DEPARTAMENTO DE ELECTRÓNICA Y AUTOMÁTICA

FACULTAD DE INGENIERÍA – UNIVERSIDAD NACIONAL DE SAN JUAN

Informe de Practica de DSP Nº3

Transformada Discreta de Fourier

Asignatura: Procesamiento Digital de Señales

Ingeniería Electrónica

Autor:

Avila, Juan Agustin – Registro 26076

1º Semestre

Año 2020

1 Enunciado

1. Implementar el algoritmo de la Transformada Discreta de Fourier (TDF) en lenguaje C con 8 y 16 puntos.

2. Verificar el algoritmo con 3 tonos de diferentes frecuencias. Experimentar primero fuera de línea y luego en línea.

3. Usar un archivo de sonido WAV y ver el espectro que genera.

4. A partir de un circuito provisto por la cátedra, realizar el análisis del contenido armónico de la red de 220V. Informar sobre los efectos de las armónicas en los sistemas eléctricos.

5. Para las experimentaciones fuera de línea un compilador de C. Para todas las experimentaciones en línea, programar el Arduino Uno.

6. Investigar alguna otra aplicación de la TDF.

7. Realizar el informe detallando cada paso realizado e incorporando en el mismo todas las gráficas e información relevantes. Armar una pequeña presentación en PPT explicando el desarrollo de la práctica.

2 Resolución.

2.1 Punto 1

Se realizó un programa en C que calcule la transformada de un archivo. En principio todas las variables estaban incluidas dentro del código, pero luego se modificó para que sea más “universal” y las variables se pasen como argumentos a la hora de ejecutar el programa. Esto agregó flexibilidad a la hora de analizar distintas señales con distintas cantidades de puntos.

El programa se debe correr de la siguiente manera:

DSP3.exe archivo_a_analizar cant_muestras offset

Por ejemplo para el archivo "Tono_50Hz.txt" con 16 muestras y un offset de 200:

"DSP3.exe Tono_50Hz 16 200"

(No se debe incluir la terminacion .txt del archivo)

La función que realiza la transformada propiamente dicha es la siguiente:

```c
void transformada(FILE *archivo_original, int offset, int N, char nombre[])
// Los argumentos son: *archivo_original es el archivo a analizar
// offset es un offset a partir del cual se toman los datos
// N es la cantidad de puntos con los cuales realiza la transformada
// nombre es el nombre del archivo original
{
    float x[N], freq, salida = 0;
    int k = 0, n = 0;
    float Im = 0, Re = 0;
    FILE *archivo_nuevo;
    char extra[50];
    snprintf(extra, sizeof(extra), "_Transformada%dPuntos", N);
    archivo_nuevo = abrir_archivo("w+", nombre, extra); //abre un nuevo archivo
    fscanf(archivo_original, "%f\n", &freq);            //obtiene la frecuencia
    fprintf(archivo_nuevo, "%.1f\n", freq);             //y la guarda en el nuevo
```

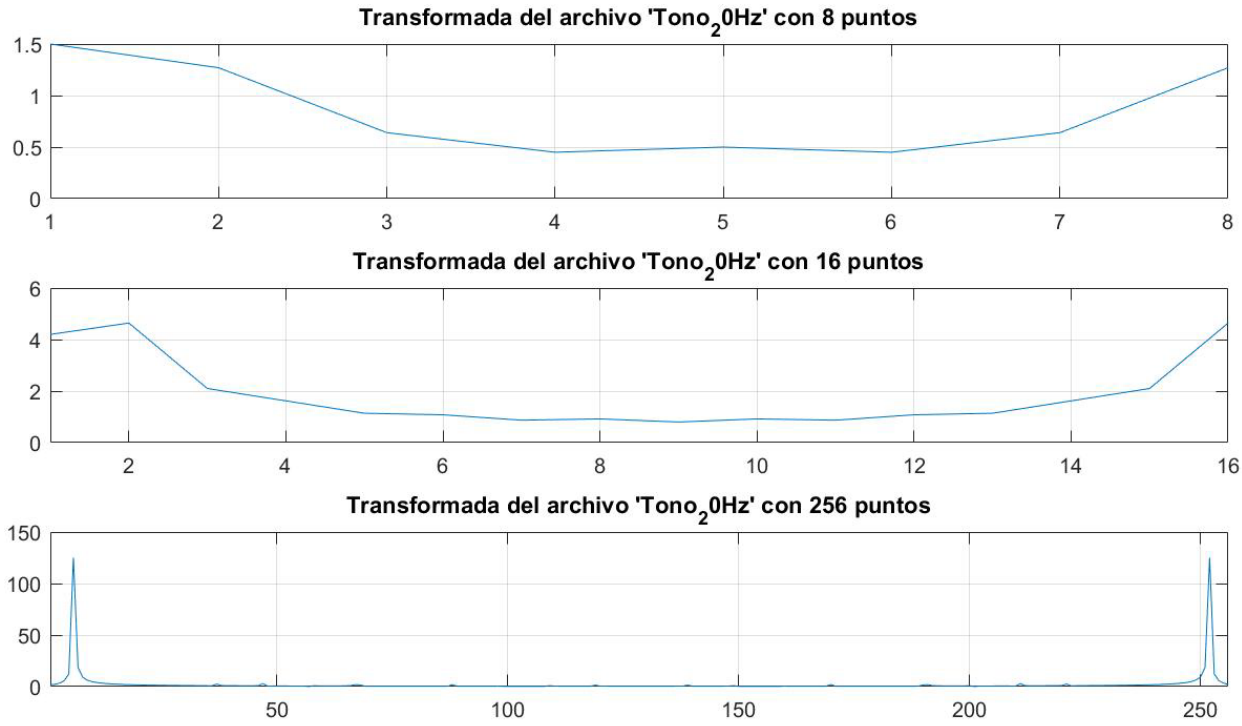
```
    for (n = 0; n < offset; n++)
    {
        x[0] = 0;
        fscanf(archivo_original, "%f\n", &x[0]); //lee el arreglo con los valores
    }
    for (n = 0; n < N; n++)
    {
        x[n] = 0;
        fscanf(archivo_original, "%f\n", &x[n]); //guarda los valores en el arreglo
    }

    //a partir de este punto es la transformada propiamente dicha
    for (k = 0; k < N; k++)
    {
        for (n = 0; n < N; n++)
        {
            Im = (Im + x[n] * (sin((2 * pi * n * k) / N))); //realiza el calculo
            Re = (Re + x[n] * (cos((2 * pi * n * k) / N)));
        }
        salida = sqrt(pow(Re, 2) + pow(Im, 2));
        fprintf(archivo_nuevo, "%.2f\n", salida); //guarda el resultado en el arch
        Im = 0;
        Re = 0;
    }
    fclose(archivo_nuevo); //cierra el archivo nuevo
    rewind(archivo_original);
}
```

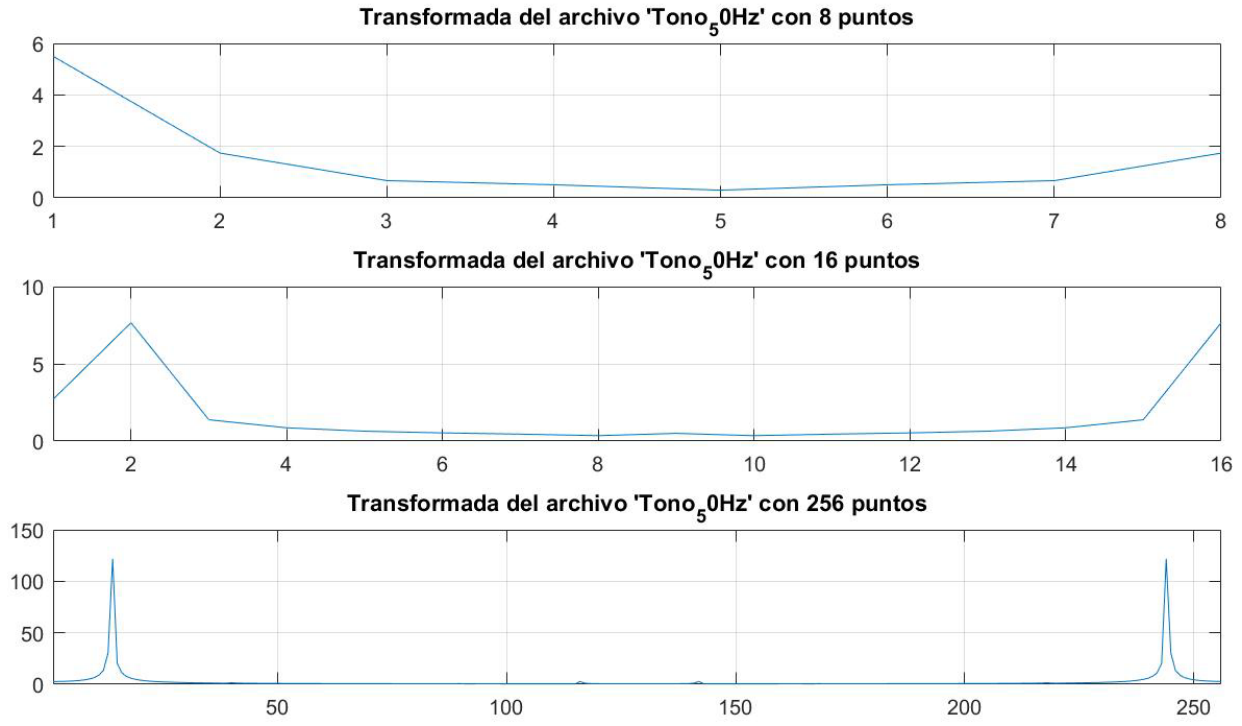
2.2 Punto 2:

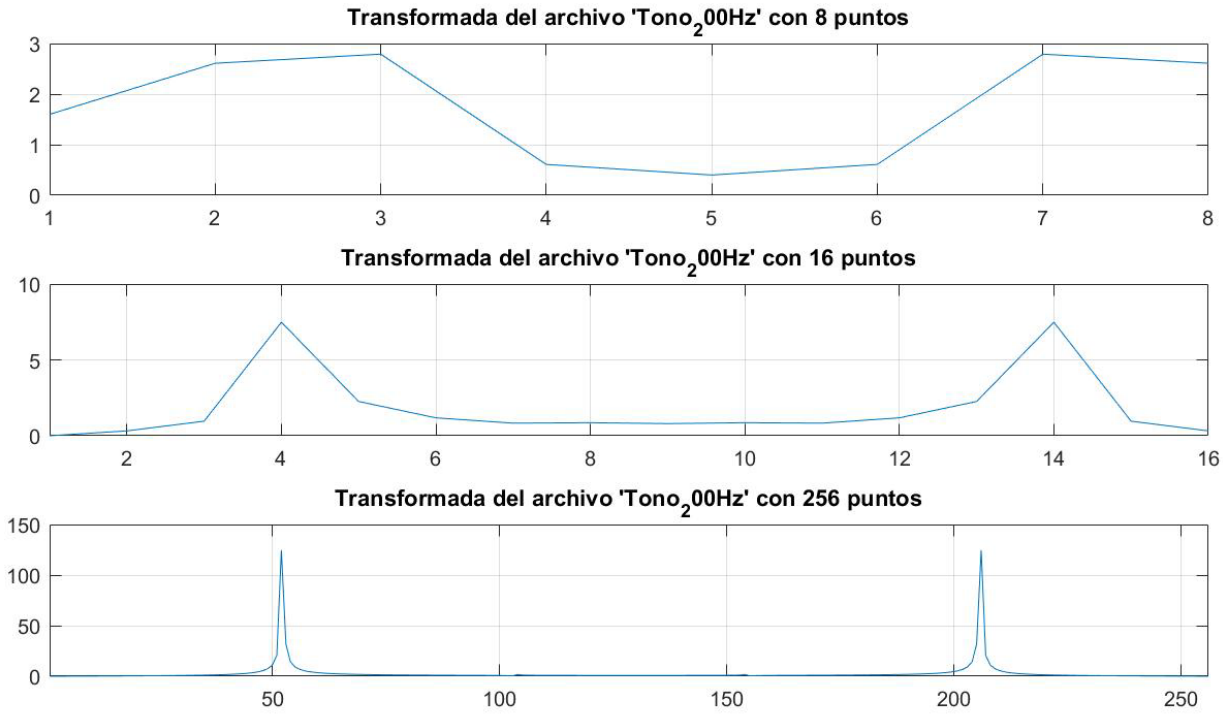
Una vez que se tuvo el código en C funcionando, se procedió a graficarlo. Para esto, se corrió el programa desde matlab modificando las distintas variables utilizadas. Para comprobar la flexibilidad del programa, además de la transformada de 8 y de 16 puntos, se realizó la transformada de 256 puntos, también para tener más resolución al analizar el espectro frecuencial. El script utilizado en matlab es el siguiente:

```
%% Punto 2
extra1="_Transformada";extra2="Puntos";
Nombres=["Tono_20Hz","Tono_50Hz","Tono_200Hz"];
puntos=[8 16 256];n=length(puntos);
for i=1:3
    figure("Name",Nombres(i));    %genera la figura
    nombre=Nombres(i); offset=20;
    for j=1:n
        subplot((n*100)+10+j);
        system("DSP3.exe "+nombre+" "+puntos(j)+" "+offset); %Corre
el archivo
        graficacionfunciones(nombre,offset,puntos(j),extra1,extra2);
    end
end
```

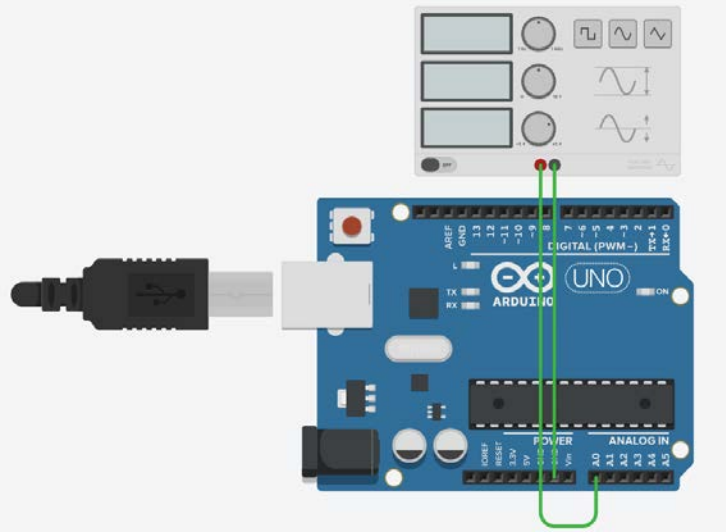
Y las salidas obtenidas fueron las siguientes:


Transformada del archivo 'Tono20Hz' con 8 puntos
Transformada del archivo 'Tono20Hz' con 16 puntos
Transformada del archivo 'Tono20Hz' con 256 puntos


Transformada del archivo 'Tono50Hz' con 8 puntos
Transformada del archivo 'Tono50Hz' con 16 puntos
Transformada del archivo 'Tono50Hz' con 256 puntos



Luego se realizó el procesamiento en línea usando TinkerCad:



Utilizando el siguiente código de arduino:

```
#include <math.h>
// Codigo para Arduino UNO – Simulador Tinkercad
const int Ts = 1; //Tiempo de Muestreo en milisegundos
const int N = 8;  //cantidad de puntos
const float pi = 3.1416;

void setup()
{
    delay(10); //solo por si el generador tiene un transitorio
```

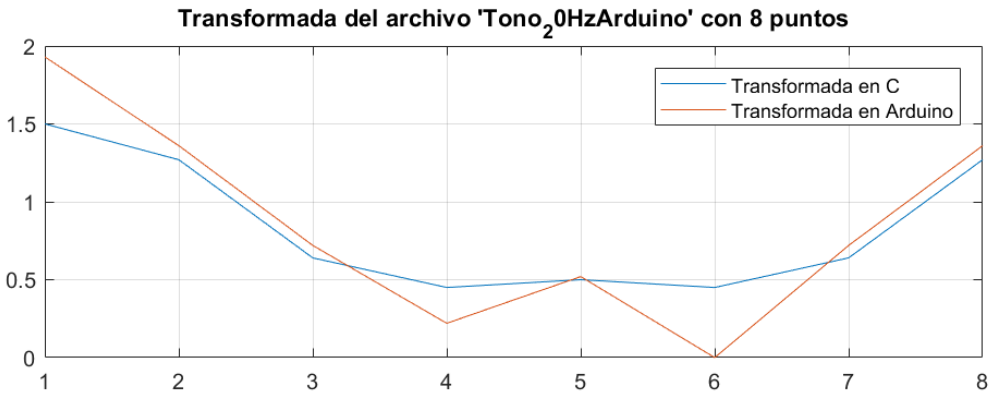
```
    Serial.begin(57600);
}

void loop()
{
    static int cont = 0;
    static float datos[N];
    if (cont < N)
    {
        datos[cont] = (float)(analogRead(A0) - 512) * 2 / 1023.0; //genera una seña
l similar a la provista por la catedra
        cont++;
    }
    else
    {
        transformada(N, datos); //cuando obtiene la cantidad de datos, calcula la t
ransf.
        cont = 0;
    }
    delay(Ts); // Espera Ts
}

void transformada(int N, float *x)
{
    float salida = 0;
    float val = 0;
    int k = 0, n = 0;
    float Im = 0, Re = 0;
    for (k = 0; k < N; k++)
    {
        for (n = 0; n < N; n++)
        {
            val = (2 * pi * n * k) / N;
            Im = Im + (x[n] * sin(val));
            Re = Re + (x[n] * cos(val));
        }
        salida = sqrt(pow(Re, 2) + pow(Im, 2));
        Serial.println(salida);
        Im = 0;
        Re = 0;
    }
}
```

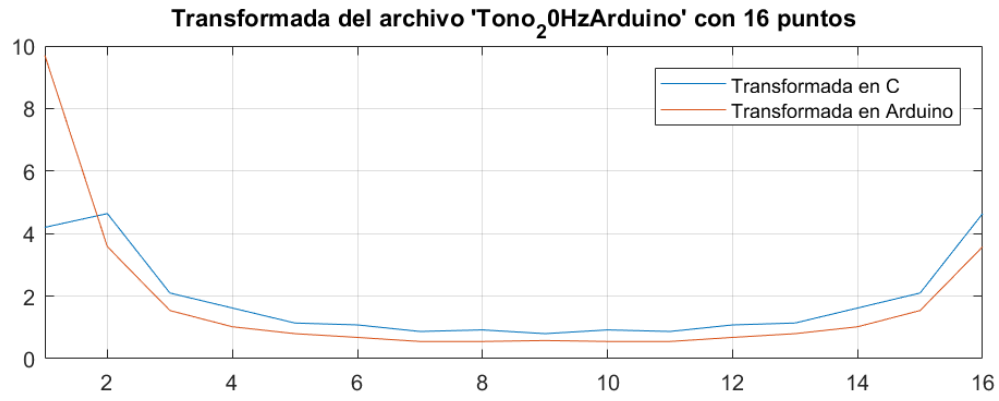
Y se reutilizó el código de matlab, en este caso graficando superpuestas las transformadas en C y en Arduino para 8 y 16 puntos:

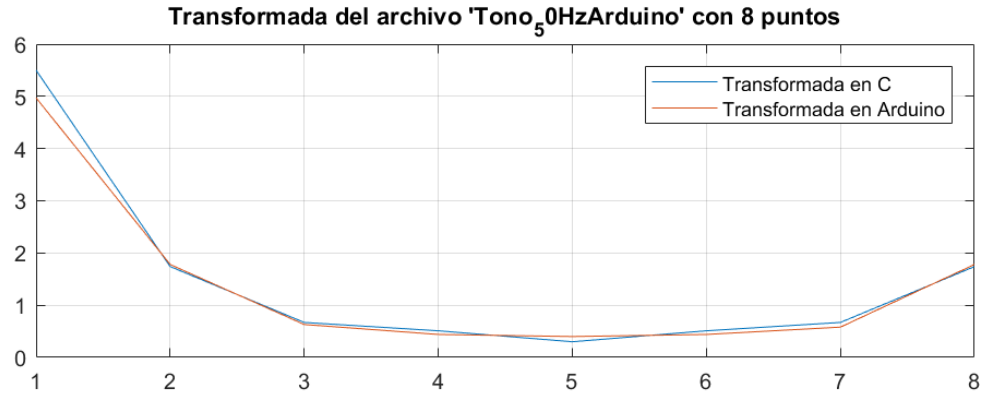
```
Nombres=["Tono_20Hz","Tono_50Hz","Tono_200Hz"];
puntos=[8 16 256]; n=length(puntos)-1;
for i=1:3
    figure("Name",Nombres(i));    %genera la figura
```

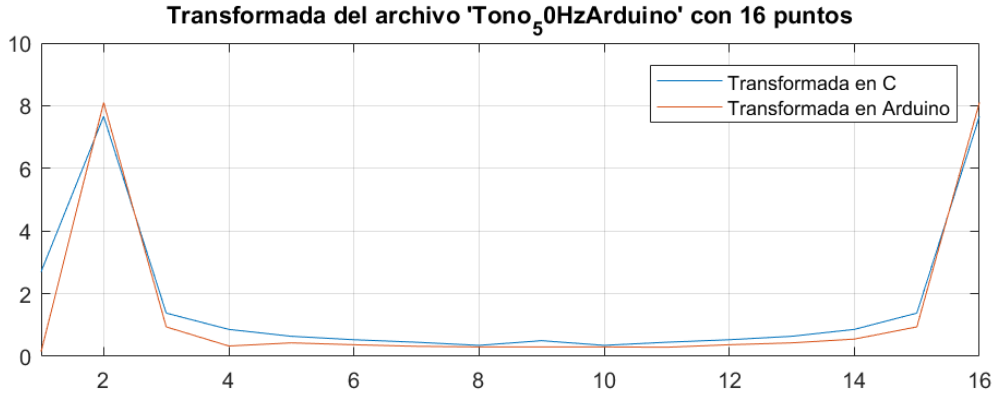
```
        nombre=Nombres(i); offset=20;
        for j=1:n
            subplot((n*100)+10+j);
            system("DSP3.exe "+nombre+" "+puntos(j)+" "+offset); %Corre
    el archivo
            graficacionfunciones(nombre,offset,puntos(j),extra1,extra2);
            hold on;

    graficacionfunciones(nombre+"Arduino",offset,puntos(j),extra1,extra2)
    ;
            legend('Transformada en C','Transformada en Arduino');
        end
    end
```

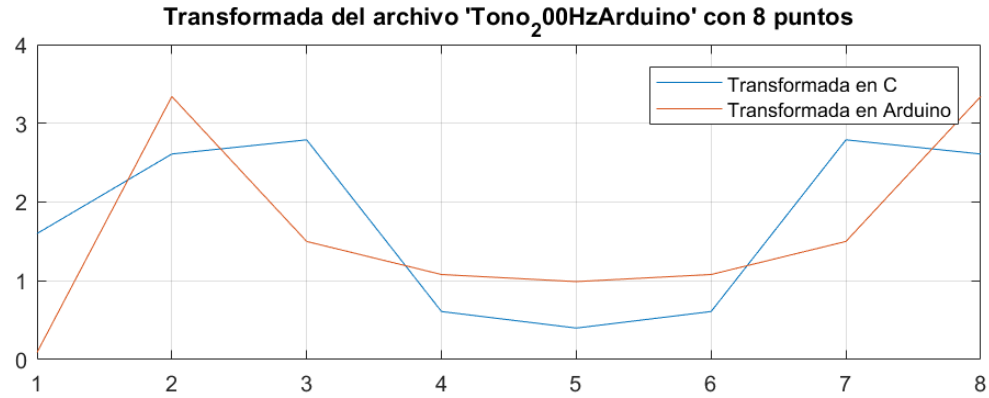
Obteniendose las siguientes graficas:

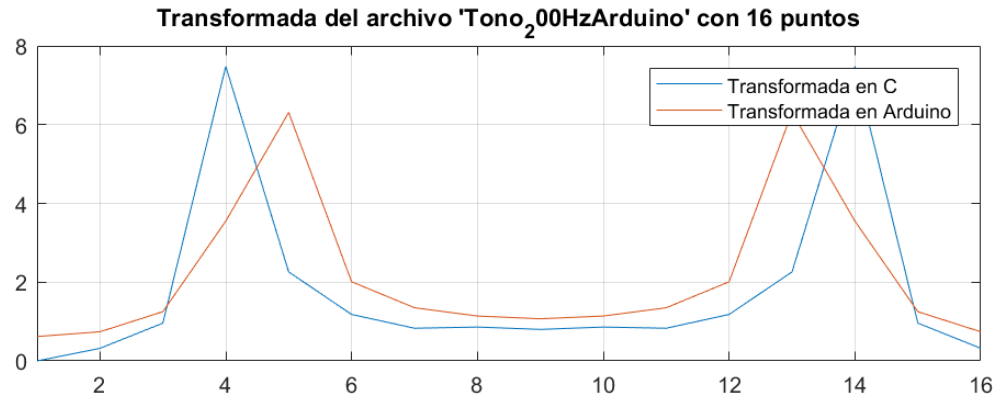





Transformada del archivo 'Tono_50HzArduino' con 8 puntos
Transformada en C
Transformada en Arduino


Transformada del archivo 'Tono_50HzArduino' con 16 puntos
Transformada en C
Transformada en Arduino


Transformada del archivo 'Tono_200HzArduino' con 8 puntos
Transformada en C
Transformada en Arduino

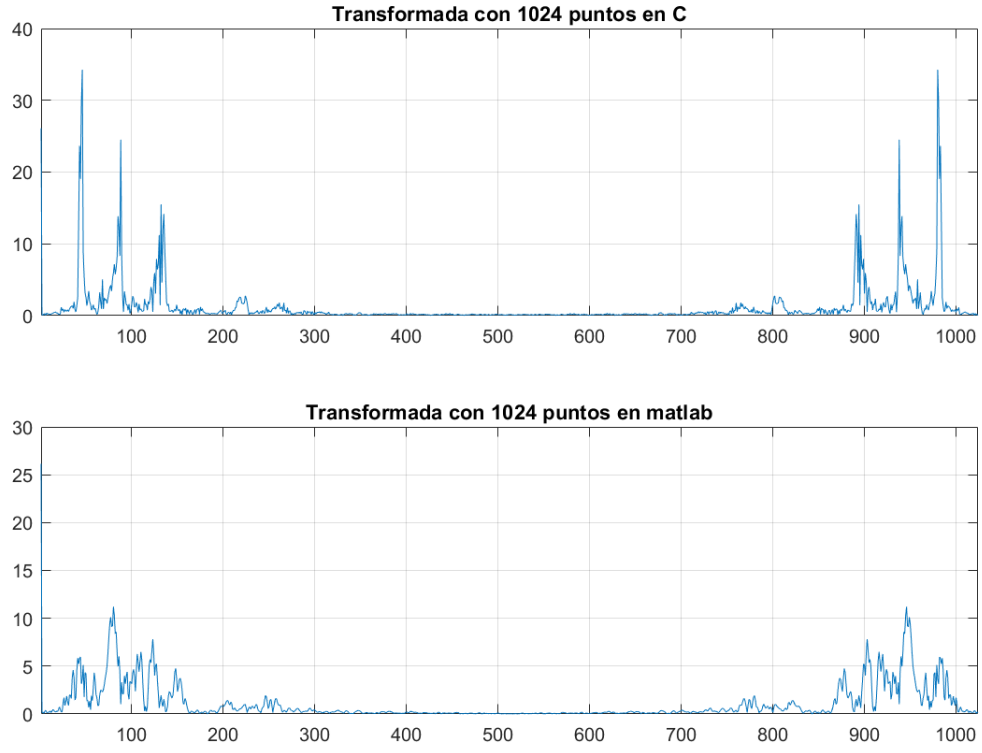

Transformada del archivo 'Tono_200HzArduino' con 16 puntos
Transformada en C
Transformada en Arduino

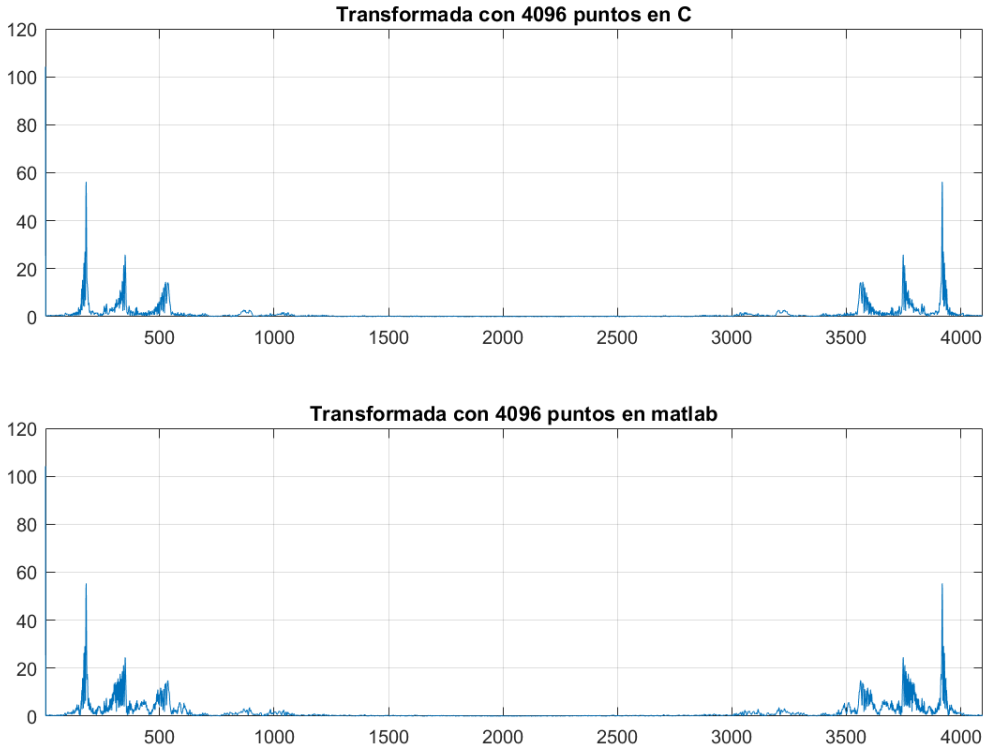
2.3 Punto 3

Se procedio a analizar un archivo wav dentro de las señales provistas por la catedra. En este caso, se utilizó el archivo “Perro.wav”, el cual se cargó en matlab y se exportó como txt. Ese archivo txt luego fue procesado con el programa en C y a la vez en matlab, y se compararon ambas transformadas. Primero se realizo la transformada con 1024 puntos, y luego con 4096. Se utilizó el siguiente código de matlab:

```
%% Punto 3
%Abrir archivo wav
[x,Fs] = audioread('Perro.wav');
%Genera y guarda los valores en un txt
arch = fopen('TonoPerro.txt', 'wt');
fprintf(arch, '%.0f\n', Fs);
for i=1:length(x)
fprintf(arch, '%.10f\n', x(i));
end
fclose(arch);

figure("Name","Ladrido de perro");    %genera la figura
nombre="Perro"; offset=1000; puntos=4096;
subplot(211);          % en la figura superior grafica la original
system("DSP3.exe "+nombre+" "+puntos+" "+offset) %Corre el archivo
graficacionfunciones(nombre,offset,puntos,extra1,extra2);
title("Transformada con "+puntos+" puntos en C");
subplot(212);          % en la figura superior grafica la original
xm=fft(x,puntos);      %calcula la DFT en matlab
plot(1:puntos,abs(xm));
title("Transformada con "+puntos+" puntos en matlab");
grid on;xlim([1 puntos]);
```

Y se obtuvieron las siguientes graficas:





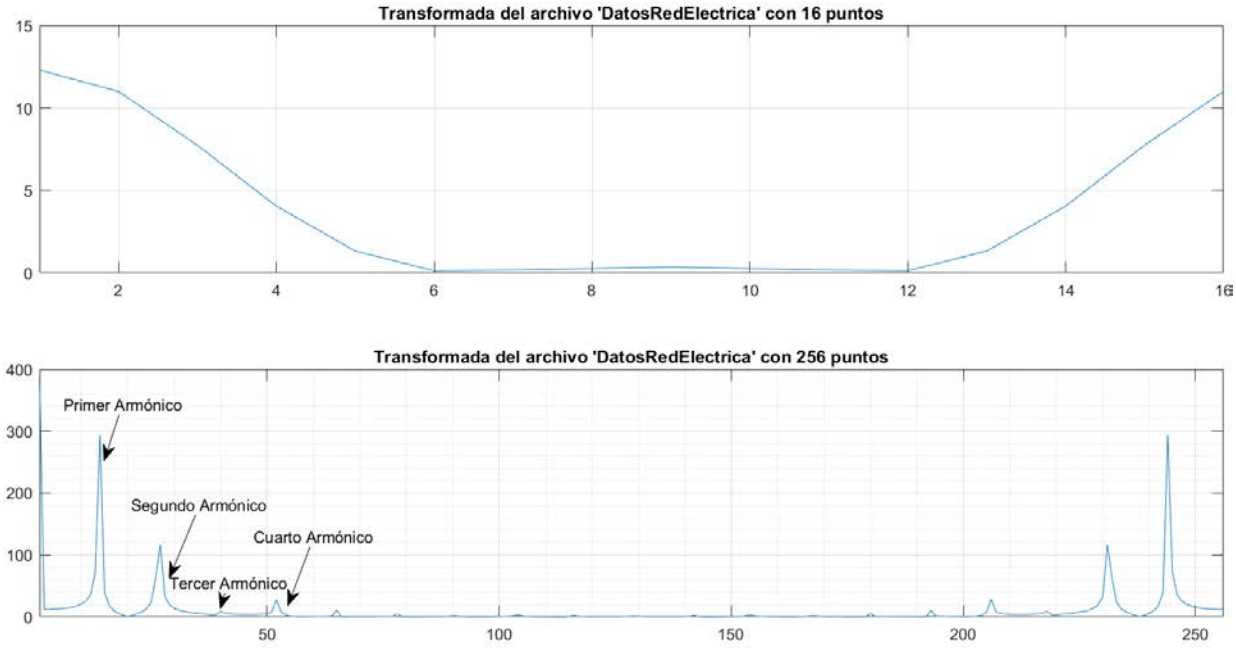
Se observa que al aumentar la cantidad de puntos, ambas transformadas tienden a igualarse.

2.4 Punto 4

Se procedio a analizar el archivo con la señal de la red eléctrica provisto por la catedra, realizando la transformada con 16 y 256 puntos. Se utilizo el siguiente script de matlab:

```
%% Punto 4
figure("Name","Transformada de Red Electrica");    %genera la figura
nombre="DatosRedElectrica"; offset=100; puntos=16;
subplot(211);
system("DSP3.exe "+nombre+" "+puntos+" "+offset)
graficacionfunciones(nombre,offset,puntos,extra1,extra2);
subplot(212);
puntos=256;
system("DSP3.exe "+nombre+" "+puntos+" "+offset)
graficacionfunciones(nombre,offset,puntos,extra1,extra2);
grid minor;
```

Y las respuestas obtenidas son las siguientes:



2.5 Punto 6

Los efectos que producen los armónicos en los circuitos eléctricos son muchos, entre los cuales se encuentran:

- Sobrecalentamientos en los conductores especialmente en el neutro de las instalaciones, debido al efecto pelicular.
- Disminución del factor de potencia de una instalación eléctrica.
- Vibraciones en cuadros eléctricos y acoplamientos en redes de telefonía y de datos.
- Deterioro de la forma de onda de la tensión, y consiguiente malfuncionamiento de los aparatos eléctricos.
- Calentamientos, degradaciones en los aislamientos, embalamientos, frenados y vibraciones en motores asíncronos.
- Degradaciones del aislamiento de los transformadores, pérdida de capacidad de suministro de potencia en los mismos.