DEPARTAMENTO DE ELECTRÓNICA Y AUTOMÁTICA

FACULTAD DE INGENIERÍA – UNIVERSIDAD NACIONAL DE SAN JUAN

Informe de Practica de DSP Nº2

Procesador de señales Arduino

Asignatura: Procesamiento Digital de Señales

Ingeniería Electrónica

Autor:

Avila, Juan Agustin – Registro 26076

1º Semestre

Año 2020

1 Enunciado

1. Armar un circuito que genere una señal periódica entre 1Hhz y 200 Hz en un rango de variación de tensión de salida de 0-5V.

2. Digitalizar la señal del circuito con 3 períodos de muestreos diferentes usando el procesador Arduino Uno.

3. Implementar un filtro promediador para mejorar la señal. Experimentar primero fuera de línea y luego en línea.

4. Implementar la siguiente ecuación: y[n]=x[n]+0.4 x[n-1]-0.6 x[n-2] a la señal.

5. A partir de un archivo de sonido WAV, implementar un filtro promediador y un ECO reproduciendo el sonido muestreado con un retraso de 0.3 segundos.

6. Para las experimentaciones fuera de línea usar cualquier compilador de lenguaje C. Para todas las experimentaciones en línea el procesador Arduino Uno.

7. Investigar alguna otra aplicación de los filtros promediadores.

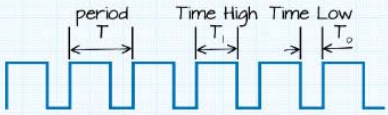
8. Realizar el informe detallando cada paso realizado e incorporando en el mismo todas las gráficas e información relevantes. Armar una pequeña presentación en PPT explicando el desarrollo de la práctica.

2 Resolución.

2.1 Punto 1

Se utilizo una calculadora de osciladores astables utilizando el CI 555 para obtener una frecuencia de 20Hz, y se obtuvo lo siguiente:

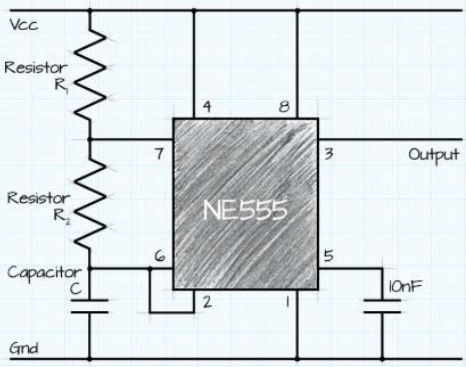
The 555 timer is capable of being used in astable and monostable circuits. In an astable circuit, the output voltage alternates between VCC and 0 volts on a continual basis.



By selecting values for R1, R2 and C we can determine the period/frequency and the duty cycle.

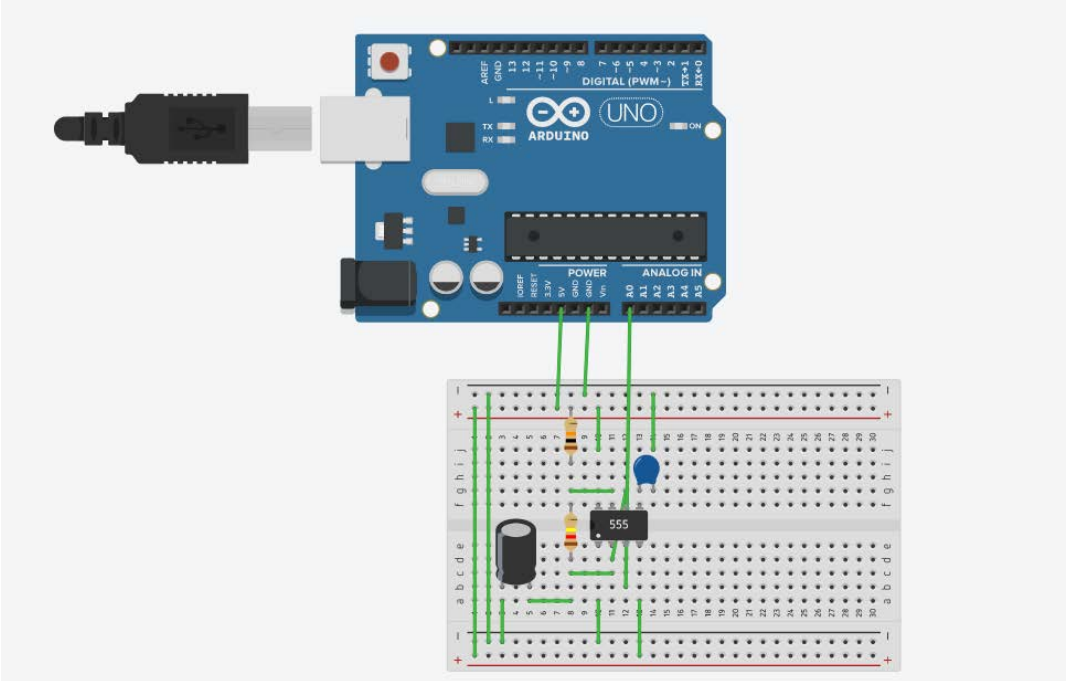
The period is the length of time it takes for the on/off cyle to repeat itself, whilst the duty cycle is the percentage of time the output is on. ie. T1/T.

In this type of circuit, the duty cycle can never be 50% or lower.



Capacitor (C)	300	nanoFarad (nF)
Resistance 1 (R_1)	10	kilohms (kΩ)
Resistance 2 (R_2)	115.25	kilohms (kΩ)
Frequency	20.000	Hertz (Hz)
Period (T)	50.000	milliseconds (ms)
Duty Cycle	52.08	%
Time High (T_1)	26.039	milliseconds (ms)
Time Low (T_0)	23.960	milliseconds (ms)

Teniendo estos datos, se procedio a realizar el circuito en TinkerCad:



2.2 Punto 2

Con el circuito armado en el simulador, se utilizó el código provisto por la catedra modificando el tiempo de muestreo. Sabiendo que la señal de entrada tiene una frecuencia de 20Hz, el tiempo de muestreo máximo es de 25ms, por lo tanto se muestreó la señal cada 5ms, 10ms y 30ms. Los resultados obtenidos en monitor serie del simulador se graficaron en matlab para poder tener escalas temporales similares.

El código utilizado es el siguiente: (incluye los puntos 3 y 4):

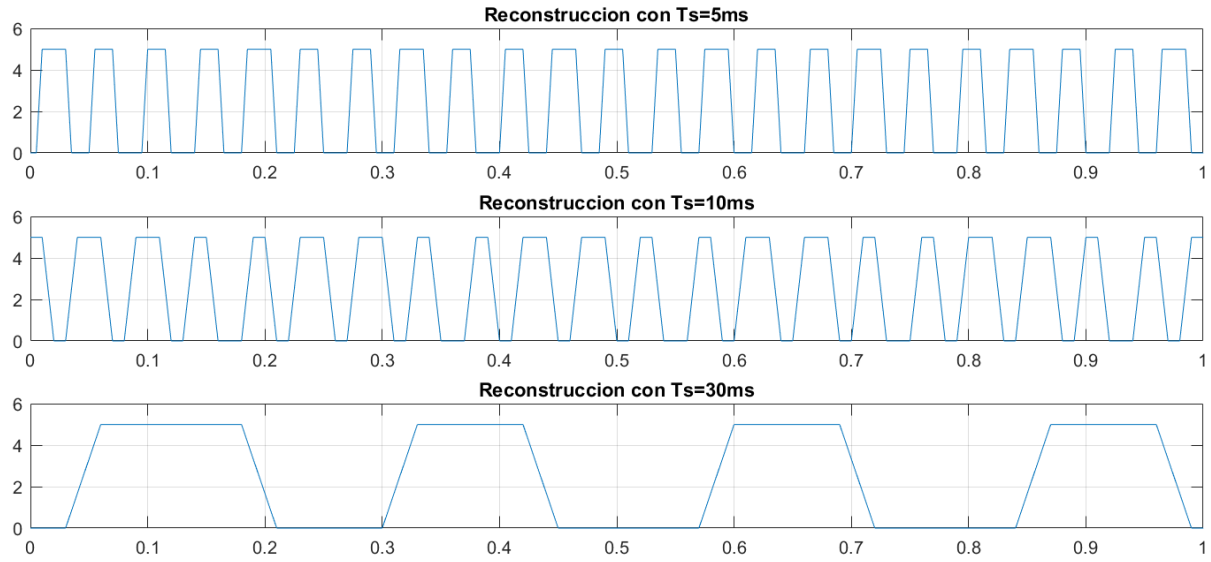
```
// Codigo para Arduino UNO – Simulador Tinkercad
int actual;
float valor_analogico_salida;
int previo=0,previo2=0;
int prom=0,punto4=0;
int Ts=10; //Tiempo de Muestreo en milisegundos
void setup()
{
 Serial.begin(57600);
}
void loop()
{
 actual=analogRead(A0);
 valor_analogico_entrada=(float)actual*5.0/1023.0;

//filtro promediador:
prom=(actual+previo)/2;
//punto 4:
punto4=actual+(previo*0.4)-(previo2*0.6);
previo2=previo;
previo=actual;


 //Serial.println("Valor Digital: ");
 Serial.print(prom);
 Serial.print(",");
 Serial.print(punto4);
 Serial.print(",");
 Serial.println(actual);
 //Serial.println("Valor Analogico: ");
 //Serial.println(valor_analogico_salida);

 delay(Ts); // Espera Ts
}
```

Y las graficas obtenidas en matlab son las siguientes:



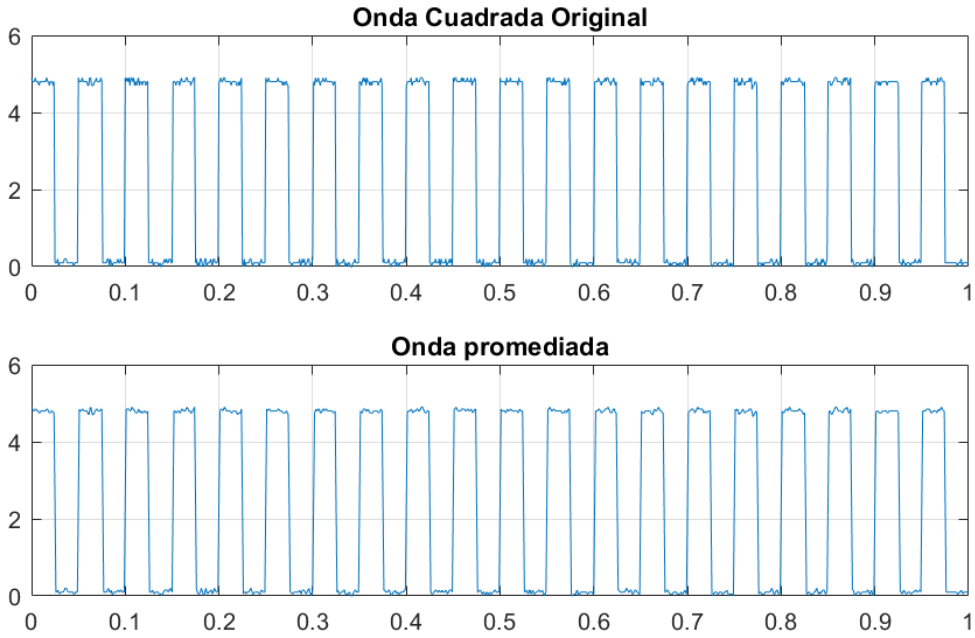
Se observa que cuando el tiempo de muestreo es mayor al tiempo limite, se produce aliasing.

2.3 Punto 3:

Fuera de línea, se realizó un programa en C que realiza un promedio de dos puntos a la señal OndaCuadrada.txt. El resultado se graficó en matlab:

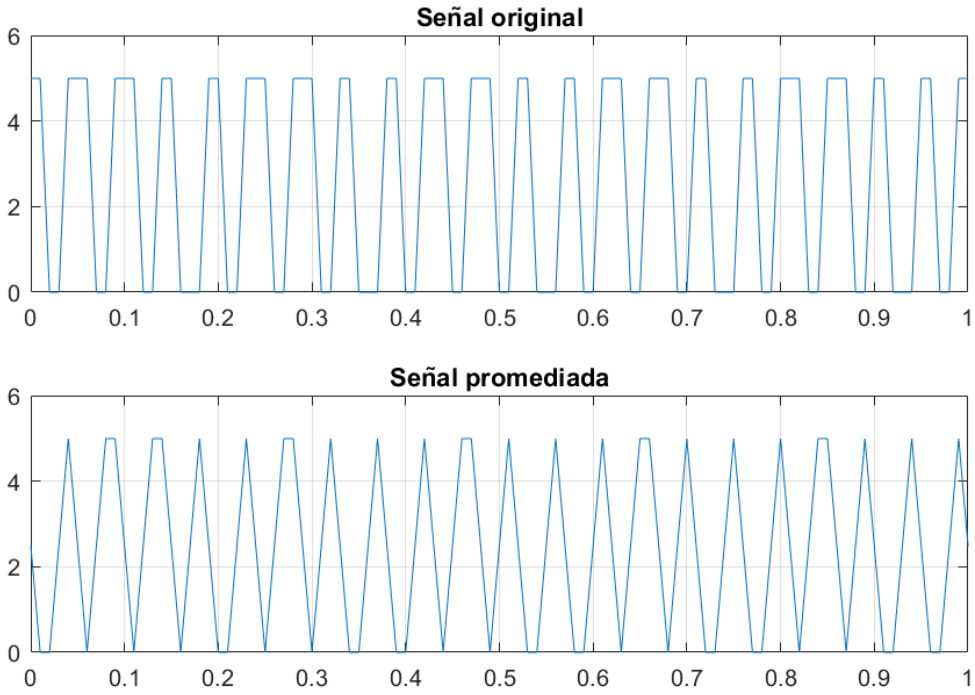
```c
void promedio(FILE *archivo_original) //Realiza el promedio
{
    float actual = 0, previo = 0, salida = 0, freq;
    FILE *archivo_nuevo;
    archivo_nuevo = abrir_archivo("w+", "Promedio");//abre un nuevo archivo
    fscanf(archivo_original, "%f\n", &freq);        //obtiene la frecuencia
    fprintf(archivo_nuevo, "%.1f\n", freq);       //y la guarda en el nuevo

    while (!feof(archivo_original))
    {
        fscanf(archivo_original, "%f\n", &actual); //Lee el valor actual
        salida = (actual + previo) / 2;            //Realiza el promedio
        previo = actual;                          //Desplaza el valor actual
        fprintf(archivo_nuevo, "%.2f\n", salida);//y lo guarda
    }
    fclose(archivo_nuevo); //cierra el archivo nuevo
    rewind(archivo_original);
}
```



Se observa que se ha eliminado levemente el ripple, esto se debe en parte a que es un promediador de dos puntos, en caso de incrementarlo se eliminaría el ripple, pero también los cambios abruptos en la señal se verían suavizados.

Luego se procedio a realizar el mismo ejercicio en el simulador de arduino con un Ts=10ms, obteniendo la siguiente salida:



Se observa que, al ser tan pocas muestras, gran parte de la señal se pierde.

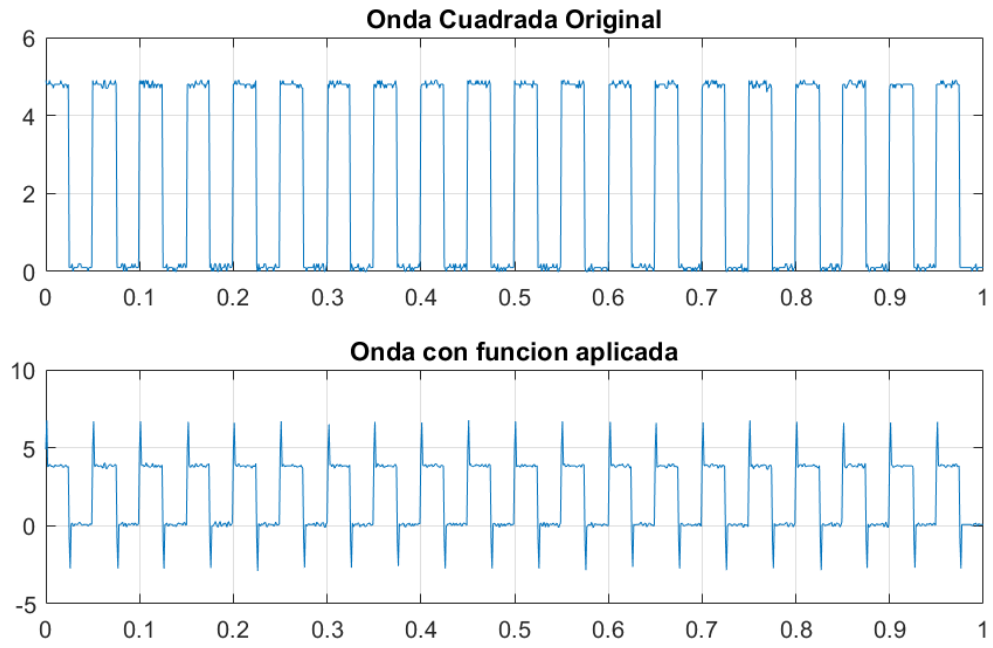
2.4 Punto 4

En el mismo programa en C del punto 3, se implementó la siguiente función:

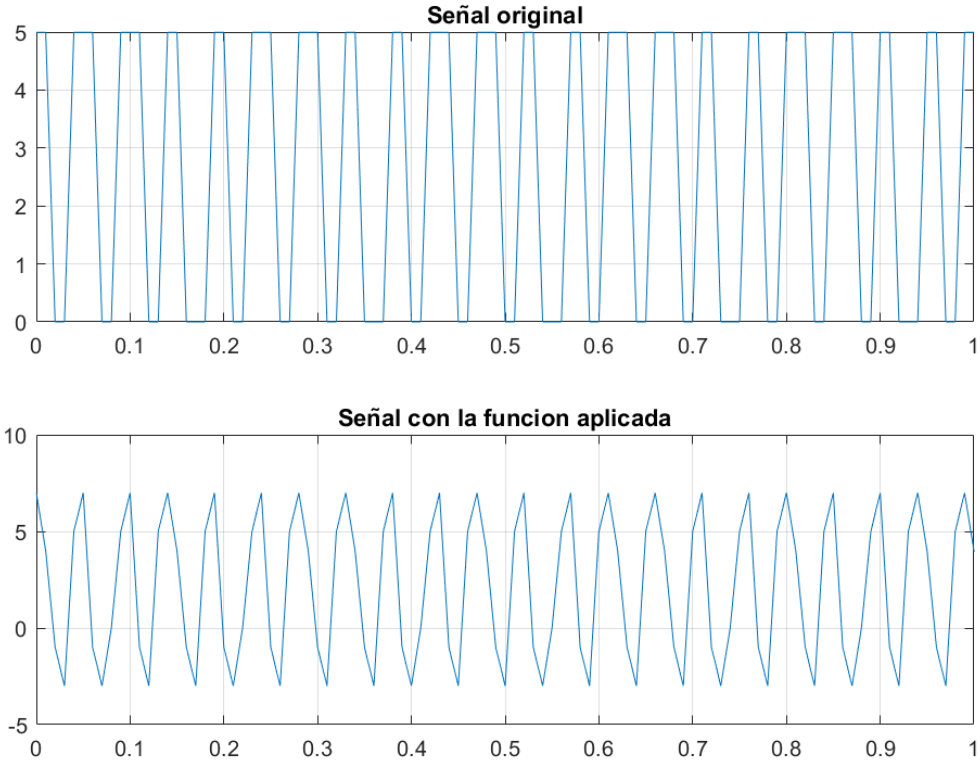
```
void funcion4(FILE *archivo_original) //resta el promedio a todo el archivo
{
    float actual = 0, previo = 0, salida = 0, previo2 = 0, freq;
    FILE *archivo_nuevo;
    archivo_nuevo = abrir_archivo("w+", "Punto4"); //abre un nuevo archivo
    fscanf(archivo_original, "%f\n", &freq);       //obtiene la frecuencia
    fprintf(archivo_nuevo, "%.1f\n", freq);        //y la guarda en el nuev
o

    while (!feof(archivo_original))
    {
        fscanf(archivo_original, "%f\n", &actual); //lee el valor actual
        salida = actual + (0.4 * previo) - (0.6 * previo2);
        previo2 = previo;                           //desplaza los valores
        previo = actual;
        fprintf(archivo_nuevo, "%.2f\n", salida); //y lo guarda
    }
    fclose(archivo_nuevo); //cierra el archivo nuevo
    rewind(archivo_original);
}
```

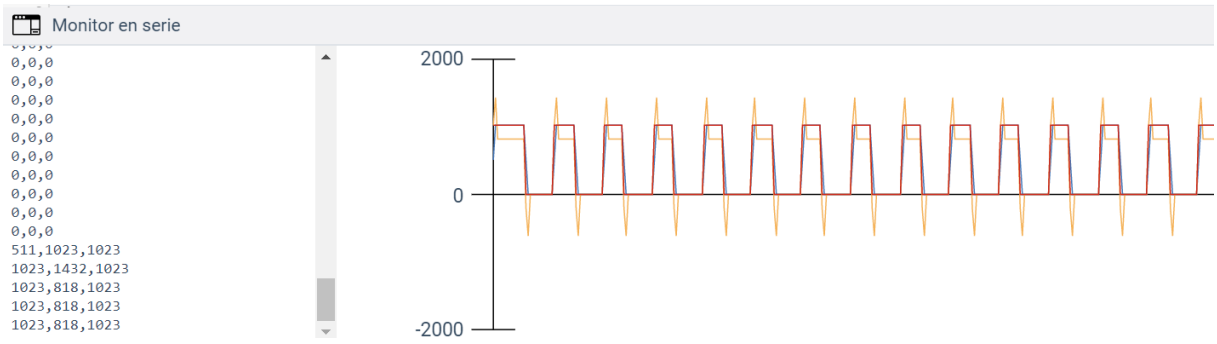
Se graficaron ambas funciones en matlab, obteniendo el siguiente resultado:



Luego, se realizó el mismo procedimiento con los datos obtenidos en arduino. El resultado fue el siguiente:



Finalmente, se adjunta la salida de las 3 funciones en simultaneo en el monitor serie (Actual, promedio y punto4):



2.5 Punto 5

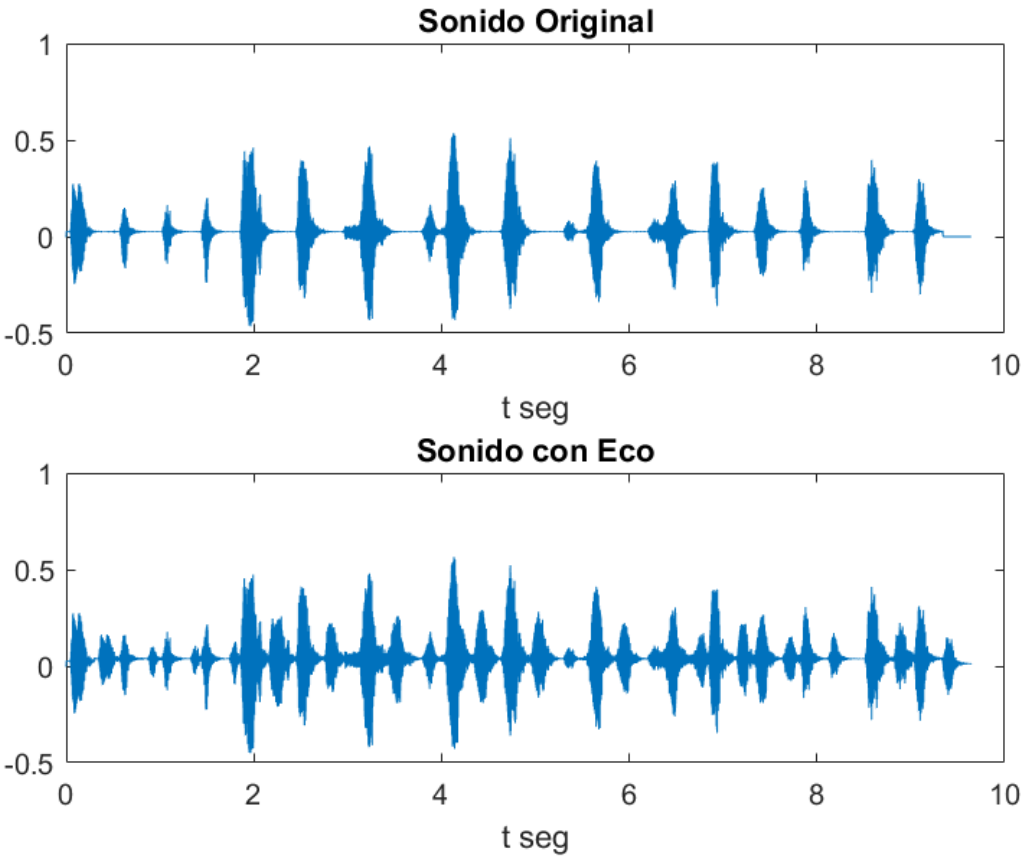
Se realizo enteramente en matlab, usando como entrada un archivo con el ladrido de un perro:

```
%% Punto 5:  archivo wav
[x,Fs] = audioread('Perro.wav');
N = length(x);

% Promediador:
previo=x(1);
prom=previo;
for i=1:length(x)
    salida=(x(i)+previo)/2;
    previo=x(i);
    prom=[prom;salida];
end
soundsc(prom,Fs);
input('Sonido promediado')
```

```
% Generación de eco mediante la suma de la versión retrasada de la
señal.
delay = 0.3 % Este valor puede ser modificado para experimentar
atten = 0.5 % Este valor puede ser modificado para experimentar
n0 = round(delay.*Fs)
n_ext = [1:1:N+n0]';
x_delay(n0+1:N+n0) = x(n_ext(n0+1:N+n0)-n0);
x_delay=x_delay(:); %se transpolo este vector para evitar errores
x_extend = x;
x_extend(N+1:N+n0) = zeros(1,n0);
y = x_extend + atten.*x_delay;  %si no aqui generaba una matriz
enorme
t_extend = (n_ext-1)./Fs;
subplot(2,1,1), plot(t_extend,x_extend);
xlabel('t seg');
title('Sonido Original');
soundsc(x,Fs);
input('Sonido con Eco')
subplot(2,1,2), plot(t_extend,y);
xlabel('t seg');
title('Sonido con Eco');
soundsc(y,Fs);
```

A continuación se muestra la gráfica del sonido original y el sonido con eco:



2.6 Punto 6