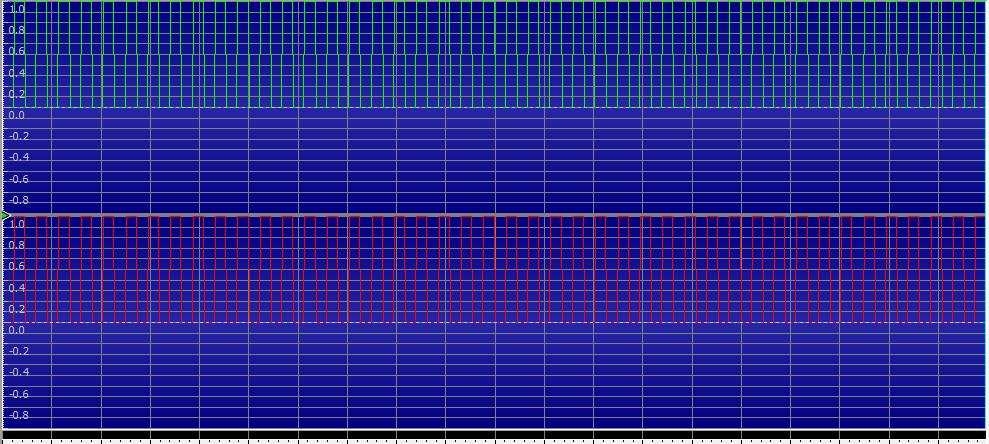
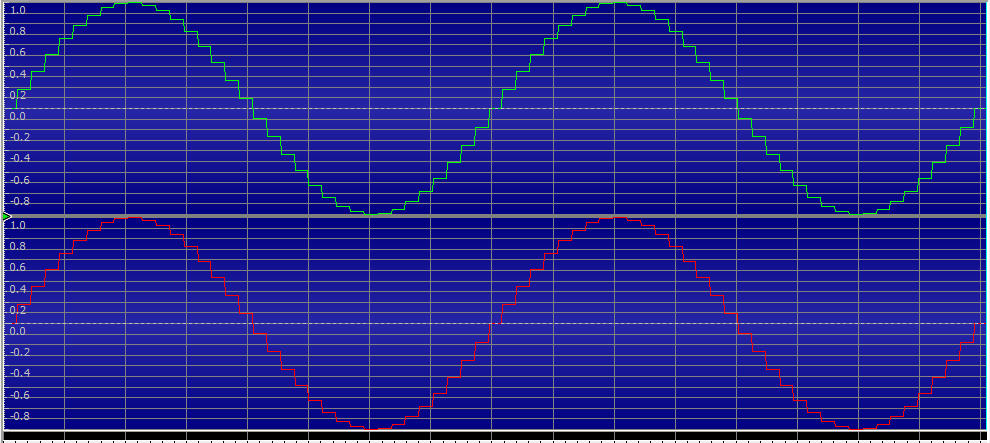
**Muestreo de una señal**

Es una de las partes del proceso de [digitalización](http://es.wikipedia.org/wiki/Digitalizaci%C3%B3n) de las señales. Consiste en tomar [muestras](http://es.wikipedia.org/wiki/Muestra_(se%C3%B1al)) de una [señal analógica](http://es.wikipedia.org/wiki/Se%C3%B1al_anal%C3%B3gica) a una [frecuencia](http://es.wikipedia.org/wiki/Frecuencia) o tasa de muestreo constante, para [cuantificarlas](http://es.wikipedia.org/wiki/Cuantificaci%C3%B3n_digital) posteriormente.

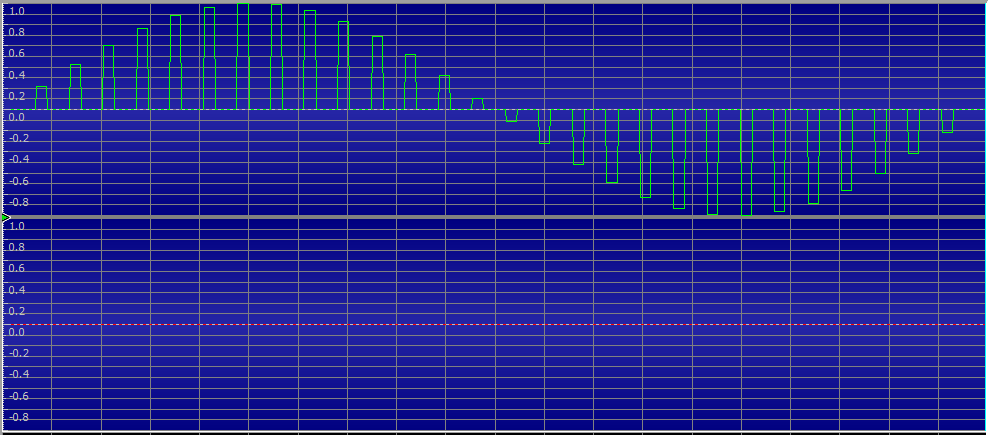
Nosotros simularemos este proceso usando la convolución y la multiplicación.

Primero tenemos un tren de impulsos y una señal seno

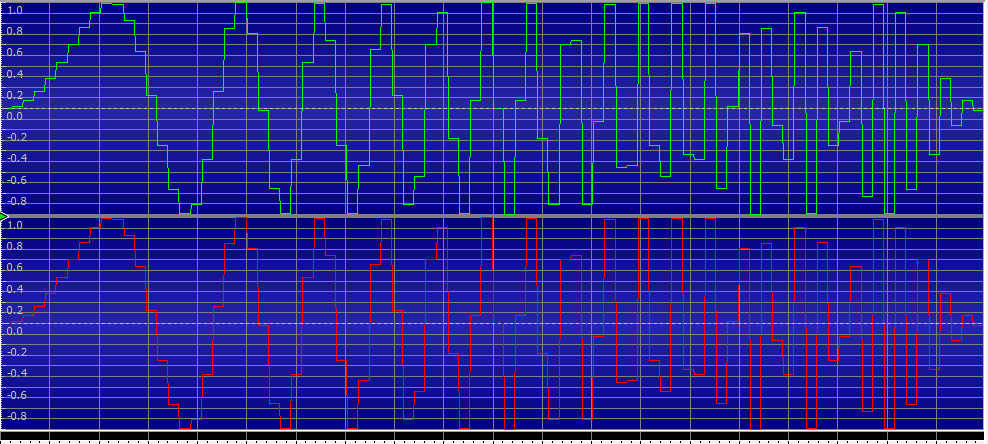




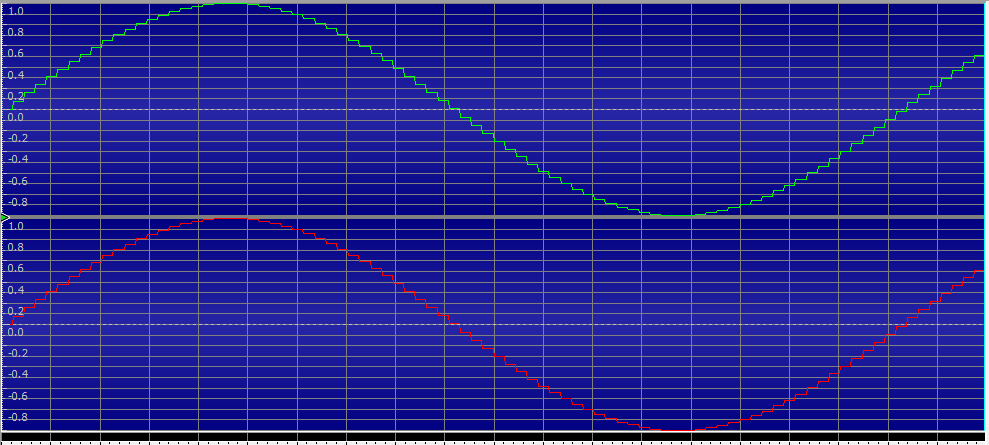
Multiplicaremos estos 2 para hacer un muestreo de la señal seno



En este punto ya tenemos la señal muestreada y discretizada. Para recuperar la señal muestreada convolucionaremos esto con una señal sampli.



Al convolucionar recuperaremos la señal



**Código:**

#include <stdio.h>

void Multi(FILE \*s1, FILE \*s2, FILE \*res);

typedef unsigned short int WORD;

typedef unsigned int DWORD;

typedef unsigned char CHAR;

typedef struct

{

CHAR riff[4];

DWORD tamano;

char wave[8];

DWORD nbytescabeza;

WORD formatopcm;

WORD monoestereo;

DWORD frecnyquist;

DWORD bytesporseg;

WORD bytesporcap;

WORD bitsmuestra;

CHAR data[4];

DWORD ndatos;

} DESCRIPTOR;

main(int argc, char \*\* argv){

FILE \*s1, \*s2, \*res;

char nom[50];

s1 = fopen(argv[1],"rb");

while(s1==NULL){

printf("Error. Archivo no encontrado!!!\n");

printf("Ingrese el nombre del archivo wav1: ");

scanf("%s",nom);

s1 = fopen(nom,"rb");

}

s2 = fopen(argv[2],"rb");

while(s2==NULL){

printf("Error. Archivo no encontrado!!!\n");

printf("Ingrese el nombre del archivo wav2: ");

scanf("%s",nom);

s1 = fopen(nom,"rb");

}

res = fopen("resultado.wav","wb");

Multi(s1, s2, res);

}

void Multi(FILE \*s1, FILE \*s2, FILE \*res){

DESCRIPTOR cabS1, cabS2, cabRes;

FILE \*hola;

hola = fopen("hola.txt", "w");

int i,j;

unsigned short int aux;

unsigned int lenS1, lenS2, bflong1, bflong2;

float \*argRes,\*argS1, \*argS2, escala, max;

fread (&cabS1, sizeof(cabS1), 1, s1);

fread (&cabS2, sizeof(cabS2), 1, s2);

cabRes = cabS1;

lenS1 = cabS1.ndatos/2;

lenS2 = cabS2.ndatos/2;

printf("Longitud: %d\t %d\n",lenS1,lenS2);

if(lenS1==lenS2){

argS1 = (float \*) malloc((lenS1/2)\*sizeof(float));

argS2 = (float \*) malloc((lenS2/2)\*sizeof(float));

argRes = (float \*) malloc((lenS2/2)\*sizeof(float));

fseek(s1, 44, SEEK\_SET);

fseek(s2, 44, SEEK\_SET);

for(i=0;i<(lenS1/2);i++){

fread(&aux, sizeof(aux),1,s1);

//fprintf(hola,"%04x\n",aux);

if(aux<=0x7FFF){

escala = aux;

argS1[i] = (escala/0x7FFF);

}else{

aux = (0xFFFF-aux);

escala = aux;

argS1[i] = -(escala/0x7FFF);

}

fread(&aux, sizeof(aux),1,s2);

if(aux<=0x7FFF){

escala = aux;

argS2[i] = (escala/0x7FFF);

}else{

aux = (0xFFFF-aux);

escala = aux;

argS2[i] = -(escala/0x7FFF);

}

//fprintf(hola,"%f\n",argS1[i]);

//fprintf(hola,"%f\n",argS2[i]);

fread(&aux, sizeof(aux),1,s1);

fread(&aux, sizeof(aux),1,s2);

}

for(i=0;i<(lenS1/2);i++){

argRes[i] = argS1[i]\*argS2[i];

max = max>argRes[i]?max:argRes[i];

}

fwrite(&cabRes,sizeof(cabRes),1,res);

for(i=0;i<(lenS1/2);i++){

if(argRes[i]>=0){

aux = (unsigned short int)(argRes[i]\*0x7FFF)/max;

fwrite(&aux,sizeof(aux),1,res);

}else{

argRes[i] = (-1)\*argRes[i];

aux = (unsigned short int)(argRes[i]\*0x7FFF)/max;

aux = (0xFFFF-aux);

fwrite(&aux,sizeof(aux),1,res);

}

aux=0;

fwrite(&aux,sizeof(aux),1,res);

}

}else{

if(lenS1<lenS2){

argS1 = (float \*) malloc((lenS2/2)\*sizeof(float));

argS2 = (float \*) malloc((lenS1/2)\*sizeof(float));

argRes = (float \*) malloc((lenS2/2)\*sizeof(float));

bflong1 = lenS2/2;

bflong2 = lenS1/2;

cabRes = cabS2;

for(i=0;i<bflong1;i++){

fread(&aux, sizeof(aux),1,s2);

//fprintf(hola,"%d\n",aux);

if(aux<=0x7FFF){

escala = aux;

argS1[i] = (escala/0x7FFF);

}else{

aux = (0xFFFF-aux);

escala = aux;

argS1[i] = -(escala/0x7FFF);

}

//fprintf(hola,"%f\n",argS1[i]);

if(!feof(s1)){

fread(&aux, sizeof(aux),1,s1);

if(aux<=0x7FFF){

escala = aux;

argS2[i] = (escala/0x7FFF);

}else{

aux = (0xFFFF-aux);

escala = aux;

argS2[i] = -(escala/0x7FFF);

}

fread(&aux, sizeof(aux),1,s1);

//fprintf(hola,"%f\n",argS2[i]);

}

//fprintf(hola,"%f\n",argS1[i]);

fread(&aux, sizeof(aux),1,s2);

}

}else{

argS1 = (float \*) malloc((lenS1/2)\*sizeof(float));

argS2 = (float \*) malloc((lenS2/2)\*sizeof(float));

argRes = (float \*) malloc((lenS1/2)\*sizeof(float));

bflong1 = lenS1/2;

bflong2 = lenS2/2;

for(i=0;i<bflong1;i++){

fread(&aux, sizeof(aux),1,s1);

//fprintf(hola,"%d\n",aux);

if(aux<=0x7FFF){

escala = aux;

argS1[i] = (escala/0x7FFF);

}else{

aux = (0xFFFF-aux);

escala = aux;

argS1[i] = -(escala/0x7FFF);

}

if(!feof(s2)){

fread(&aux, sizeof(aux),1,s2);

if(aux<=0x7FFF){

escala = aux;

argS2[i] = (escala/0x7FFF);

}else{

aux = (0xFFFF-aux);

escala = aux;

argS2[i] = -(escala/0x7FFF);

}

fread(&aux, sizeof(aux),1,s2);

//fprintf(hola,"%f\n",argS2[i]);

}

//fprintf(hola,"%f\n",argS1[i]);

fread(&aux, sizeof(aux),1,s1);

}

}

j=0;

for(i=0;i<bflong1;i++){

if(j>=bflong2){

j=0;

}

argRes[i] = argS1[i]\*argS2[j];

fprintf(hola,"%f\n",argRes[i]);

max = max>argRes[i]?max:argRes[i];

j++;

}

fwrite(&cabRes,sizeof(cabRes),1,res);

for(i=0;i<(lenS1/2);i++){

if(argRes[i]>=0){

aux = (unsigned short int)(argRes[i]\*0x7FFF)/max;

fwrite(&aux,sizeof(aux),1,res);

}else{

argRes[i] = (-1)\*argRes[i];

aux = (unsigned short int)(argRes[i]\*0x7FFF)/max;

aux = (0xFFFF-aux);

fwrite(&aux,sizeof(aux),1,res);

}

aux=0;

fwrite(&aux,sizeof(aux),1,res);

}

}

fclose(hola);

fclose(s1);

fclose(s2);

fclose(res);

}