

INFORME PRÁCTICAS PDIH CURSO 2020-2021

Realizado por: Nora Itafti Rivas

SEMINARIO 1: INTRODUCCIÓN A DOSBOX

Configuración del inicio de DOSBOX:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount c c:\users\noraa\desktop\dosbox
Drive C is mounted as local directory c:\users\noraa\desktop\dosbox\

Z:\>c:

C:\>dir
Directory of C:\.
.                <DIR>                11-03-2021 11:46
..               <DIR>                12-03-2021 11:09
BC               <DIR>                11-03-2021 11:48
JUEGOS           <DIR>                11-03-2021 11:42
S1              <DIR>                12-03-2021 10:40
    0 File(s)          0 Bytes.
    5 Dir(s)          262,111,744 Bytes free.

C:\>
```

Ejecución de juegos:

```
C:\>cd juegos

C:\JUEGOS>dir
Directory of C:\JUEGOS\.
.                <DIR>                11-03-2021 11:42
..               <DIR>                11-03-2021 11:46
VBALL            <DIR>                11-03-2021 11:42
LIVING~1.COM      61,440 11-03-2021 11:42
    1 File(s)        61,440 Bytes.
    3 Dir(s)          262,111,744 Bytes free.

C:\JUEGOS>vball
Illegal command: vball.

C:\JUEGOS>cd vball

C:\JUEGOS\VBALL>vball
```



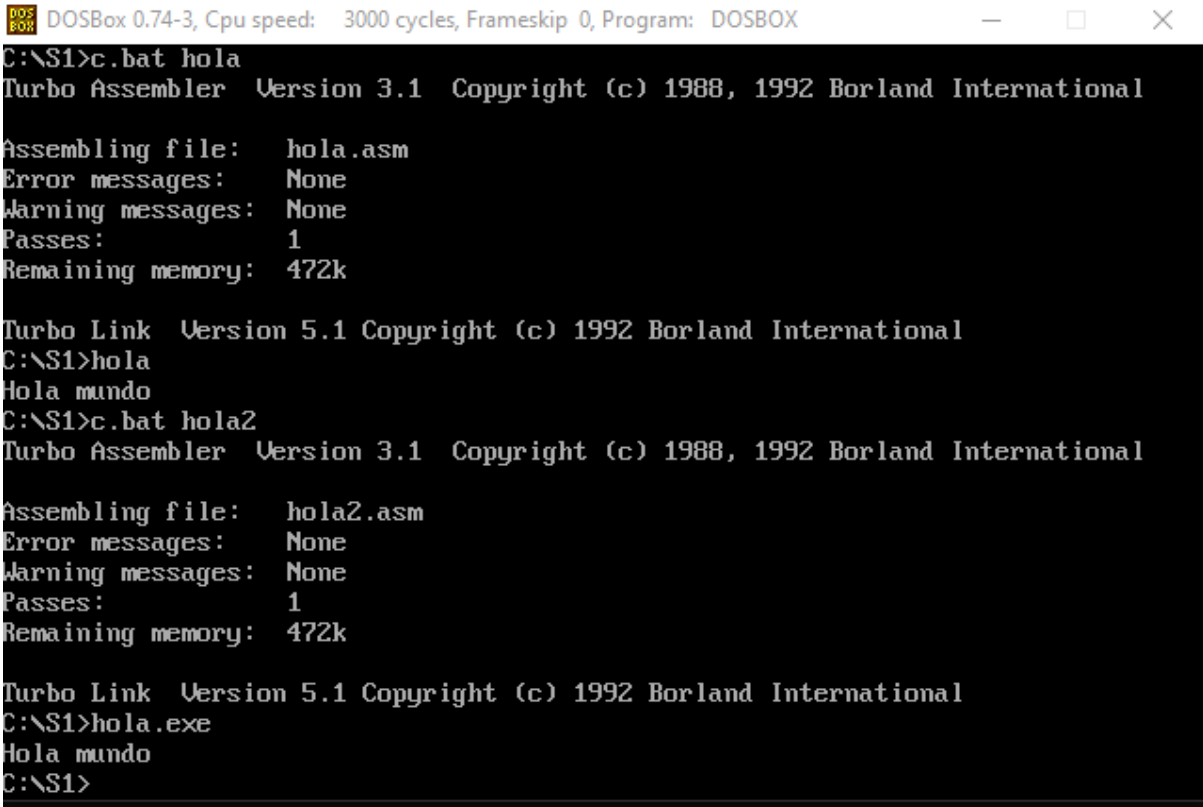
Código de ejemplo hola.asm

```
pila segment stack 'stack'
    dw 100h dup (?)
pila ends
datos segment 'data'
    msg db 'Hola mundo$'
datos ends
codigo segment 'code'
    assume cs:codigo, ds:datos, ss:pila
    main PROC
        mov ax,datos
        mov ds,ax

        mov dx,OFFSET msg
        mov ah,9
        int 21h

        mov ax,4C00h
        int 21h
    main ENDP
codigo ends
END main
```

Compilación y ejecución del ejemplo(Para la compilación se ha hecho uso del fichero c.bat):



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\S1>c.bat hola
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International

Assembling file:   hola.asm
Error messages:   None
Warning messages: None
Passes:           1
Remaining memory: 472k

Turbo Link Version 5.1 Copyright (c) 1992 Borland International
C:\S1>hola
Hola mundo
C:\S1>c.bat hola2
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International

Assembling file:   hola2.asm
Error messages:   None
Warning messages: None
Passes:           1
Remaining memory: 472k

Turbo Link Version 5.1 Copyright (c) 1992 Borland International
C:\S1>hola.exe
Hola mundo
C:\S1>
```

Código ASM de fichero del programa ejemplo de impresión del mensaje “Hola mundo” 7 veces con bucle:

```
pila segment stack 'stack'
    dw 100h dup (?)
pila ends
datos segment 'data'
    msg db 'Hola mundo$'
datos ends
codigo segment 'code'
    assume cs:codigo, ds:datos, ss:pila
    main PROC
        mov ax,datos
        mov ds,ax

        mov cx,0

    bucle:
        mov dx,OFFSET msg
        mov ah,9
        int 21h
        ;actualizar contador y comprobar condición
        inc cx
        cmp cx,7
        jne bucle

        mov ax,4C00h
        int 21h

    main ENDP
codigo ends

END main
```

Ejemplo de ejecución ejemplo 2 (con bucle):

```
C:\S1>c.bat hola2
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International

Assembling file:    hola2.asm
Error messages:     None
Warning messages:   None
Passes:             1
Remaining memory:   472k

Turbo Link Version 5.1 Copyright (c) 1992 Borland International
C:\S1>hola2.exe
Hola mundo Hola mundo Hola mundo Hola mundo Hola mundo Hola mundo Hola mundo
C:\S1>
```

PRÁCTICA 2: JUEGO PONG

Código C correspondiente:

```
// Compilar con: gcc p2.c -o p2 -lncurses
#include <ncurses.h>
#include <unistd.h>

#define DELAY 900000

int main(int argc, char *argv[]) {
    int x = 2, y = 9; //posicion pelota
    int max_y = 25, max_x = 79;
    int next_x = 0;
    int directionx = 2;
    int next_y = 0;
    int directiony = 2;
    int rows=50, cols=80;
    bool seguir = true;
    int p1=0, p2=0; //Puntuaciones de los jugadores, inicialmente valen 0

    int jugador=1;
    int xc1 = 0, yc1 = 6; //POSICION JUGADOR IZQUIERDA
    int xc2 = 78, yc2 = 6; //POSICION JUGADOR DERECHA
    int ch = 0;

    initscr();
    noecho();
    cbreak();
    nodelay(stdscr, TRUE);
    start_color();
    init_pair(1, COLOR_YELLOW, COLOR_GREEN);
    init_pair(2, COLOR_WHITE, COLOR_BLACK);
    init_pair(3, COLOR_WHITE, COLOR_BLUE);

    while(1){
        mvprintw(5, 31, " ----- ");
        mvprintw(6, 31, "|Juego Pong|");
        mvprintw(7, 31, " ----- ");
        mvprintw(8, 32, "Controles");
        mvprintw(10, 24, "s --> Desplazar hacia abajo");
        mvprintw(11, 24, "w --> Desplazar hacia arriba");
        mvprintw(12, 10, "En la parte inferior aparece el jugador que tiene el control");
        mvprintw(14, 24, "Pulsar Enter para comenzar...");
        refresh();
        ch = getch();
```

```

if ( ch == '\n'){
    seguir = false;
    while(seguir==false){
        usleep(DELAY);
        curs_set(FALSE);
        WINDOW *window = newwin(rows,cols,0,0);
        wbkgd(window, COLOR_PAIR(2));
        mvwprintw(window,y, x, "o");
        mvwprintw(window, yc1, xc1, "XX");
        mvwprintw(window, yc1+1, xc1, "XX");
        mvwprintw(window, yc1+2, xc1, "XX");
        mvwprintw(window, yc1+3, xc1, "XX");
        mvwprintw(window, yc2, xc2, "XX");
        mvwprintw(window, yc2+1, xc2, "XX");
        mvwprintw(window, yc2+2, xc2, "XX");
        mvwprintw(window, yc2+3, xc2, "XX");
        mvwprintw(window, 12, 0,
"-----");
        if(jugador==1){
            mvwprintw(window, 14, 0, "Jugador actual: JUGADOR 1");
        }
        if(jugador==2){
            mvwprintw(window, 14, 0, "Jugador actual: JUGADOR 2");
        }
        mvwprintw(window, 16, 0, " ----- ");
        mvwprintw(window, 17, 0, "|Puntuaciones|");
        mvwprintw(window, 18, 0, " ----- ");
        mvwprintw(window, 19, 0, "JUGADOR 1: %i", p1);
        mvwprintw(window, 20, 0, "JUGADOR 2: %i", p2);
        mvwprintw(window, 16, 50, " ----- ");
        mvwprintw(window, 17, 50, "|Controles|");
        mvwprintw(window, 18, 50, " ----- ");
        mvwprintw(window, 19, 50, "s --> Desplazar hacia abajo");
        mvwprintw(window, 20, 50, "w --> Desplazar hacia arriba");
        wrefresh(window);

        //Comprobamos en qué lado del campo está la pelota, para poder asignar un
jugador
        if(x > 38){
            jugador = 2;
        }
        else{
            jugador = 1;
        }

        next_x = x + directionx;
        next_y = y + directiony;

        if (next_x >= max_x || next_x < 0) {

```

```

        directionx*= -1;
    } else {
        x+= directionx;
    }

    if (next_y >= 12 || next_y < 0) {
        directiony*= -1;
    }else {
        y+= directiony;
    }

    //Realizamos los movimientos pertinentes
    ch = getch();
    if(jugador == 1){
        if (ch == 'w'){
            yc1 -= 2;
        }
        else if (ch == 's'){
            yc1 += 2;
        }
    }
    if(jugador == 2){
        if (ch == 'w'){
            yc2 -= 2;
        }
        else if (ch == 's'){
            yc2 += 2;
        }
    }

    //Puntuaciones
    if (next_x > xc2){
        next_x = xc2 - 1; //volvemos a la posición una unidad menor del máximo
        x = xc2 - 1;      // del valor d x para el jugador 2
        p1++;
    }
    else if (next_x < xc1){
        next_x = xc1 + 1; //volvemos a la posición una unidad mayor del mínimo
        x = xc1 + 1;      // del valor d x para el jugador 1
        p2++;
    }
}
}
}
}
endwin();
}

```

Resultado de la compilación:

```

[nora-P2]$> gcc p2.c -o p2 -lcurses
[nora-P2]$> ./p2

```


Resultado de la ejecución

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda

-----
|Juego Pong|
-----
Controles

s --> Desplazar hacia abajo
w --> Desplazar hacia arriba
En la parte inferior aparece el jugador que tiene el control

Pulsar Enter para comenzar...[]
```

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda

XX                                     XX
XX                                     XX
XX                                     XX
XX                                     XX

-----

Jugador actual: JUGADOR 2

-----
|Puntuaciones|
-----
JUGADOR 1: 1
JUGADOR 2: 0

-----
|Controles|
-----
s --> Desplazar hacia abajo
w --> Desplazar hacia arriba
```

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda

o

XX
XX
XX
XX
XX

XX
XX
XX
XX
XX

-----

Jugador actual: JUGADOR 1

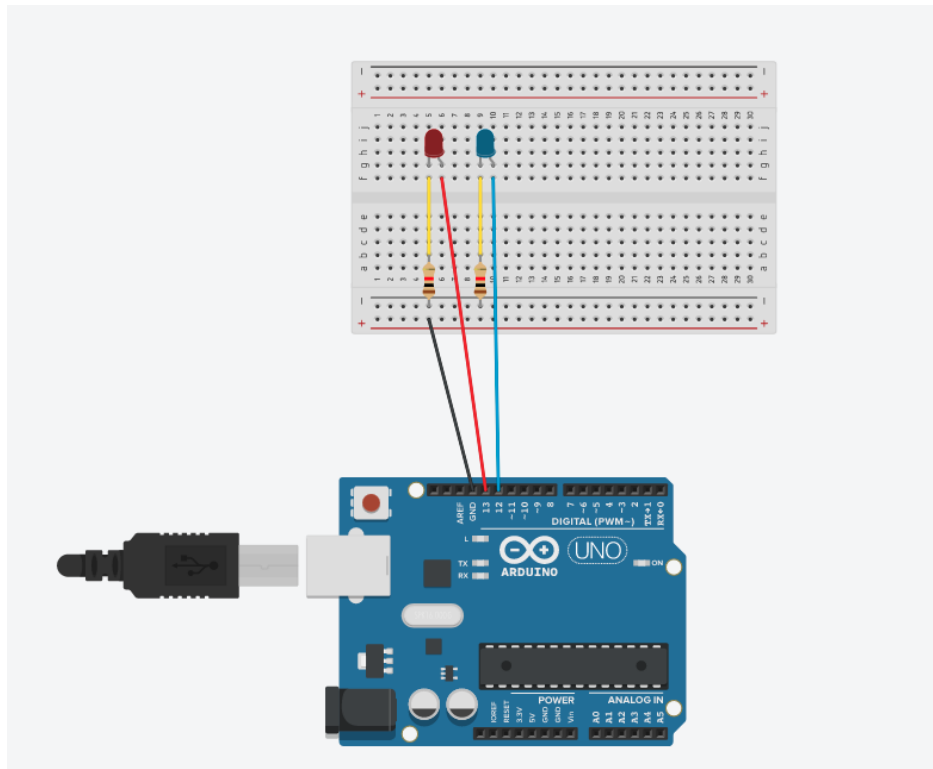
-----
|Puntuaciones|
-----
JUGADOR 1: 1
JUGADOR 2: 1

-----
|Controles|
-----
s --> Desplazar hacia abajo
w --> Desplazar hacia arriba
```

SEMINARIO 3: ARDUINO

Ejercicio 1

El diseño del circuito lo podemos ver representado en la siguiente imagen. Para realizar dicho diseño he hecho uso de la herramienta Tinkercad: <https://www.tinkercad.com/>



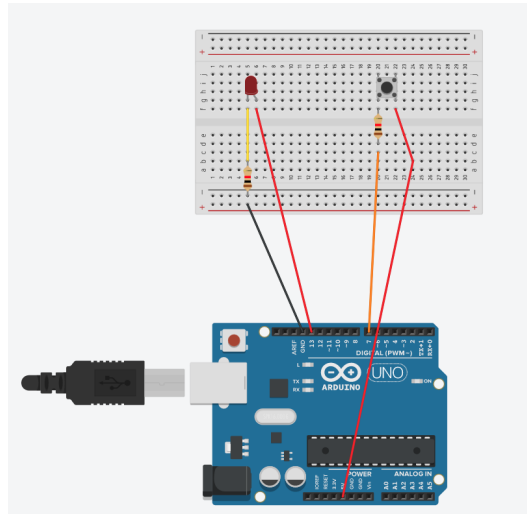
Código de ejercicio 1: código Arduino para configurar el encendido de leds asociados a 3 pines(12 y 13)

```
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
}
void loop()
{
  digitalWrite(13, HIGH);
  delay(1500);
  digitalWrite(13, LOW);

  digitalWrite(12,HIGH);
  delay(1500);
  digitalWrite(12, LOW);
}
```

Ejercicio 2:

El diseño del circuito lo podemos ver representado en la siguiente imagen.



Código de ejercicio 2: código Arduino para configurar a través de un interruptor el encendido del led asociado al pin número 13

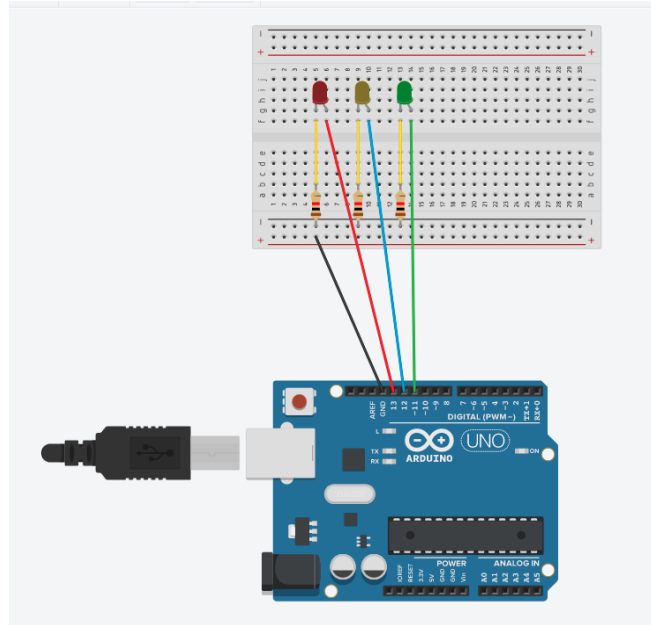
```
void setup()
{
  pinMode(7, INPUT);
  pinMode(13, OUTPUT);
}

void loop()
{
  int val = digitalRead(7); //leer si el interruptor está pulsado o no
  digitalWrite(13,!val); // Apagar el LED si el interruptor está pulsado
}
```

PRÁCTICA 3: ARDUINO

Ejercicio 1

El diseño del circuito lo podemos ver representado en la siguiente imagen.



Código ejercicio 1: código Arduino para configurar el encendido de leds asociados a 3 pines(11,12 y 13)

```
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
}

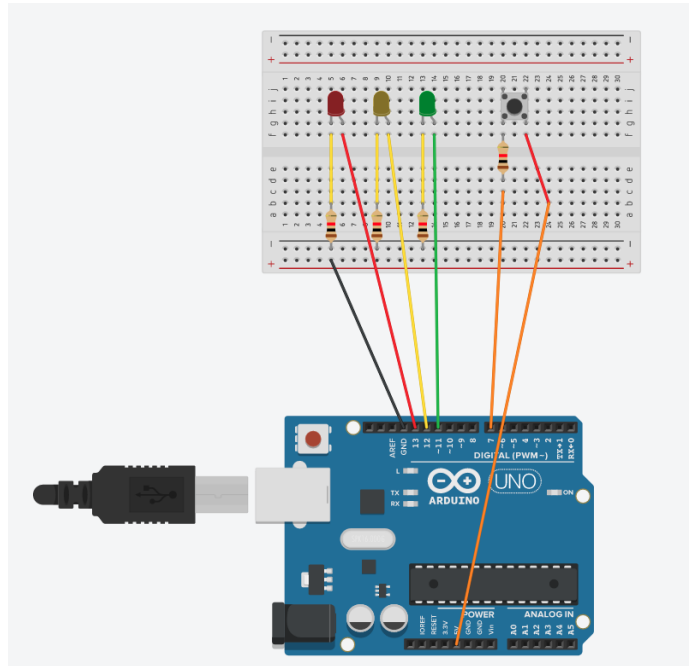
void loop()
{
  digitalWrite(13, HIGH);
  delay(1500);
  digitalWrite(13, LOW);

  digitalWrite(12,HIGH);
  delay(1500);
  digitalWrite(12, LOW);

  digitalWrite(11, HIGH);
  delay(1500);
  digitalWrite(11, LOW);
}
```

Ejercicio 2:

El diseño del circuito lo podemos ver representado en la siguiente imagen:



Código ejercicio 2: código Arduino para configurar el encendido de leds asociados a 3 pines(11,12 y 13) añadiendo el control del encendido de los mismo con un interruptor, en caso de estar pulsado el interruptor, los leds correspondientes a los pines 11 y 12 quedarán apagados, cuando el interruptor deje de estar pulsado los 3 leds continuarán encendidos

```
void setup()
{
  pinMode(7, INPUT);
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
}

void loop()
{
  int val = digitalRead(7);

  digitalWrite(13, val);
  digitalWrite(12, !val);
  digitalWrite(11, !val);
}
```

SEMINARIO 4: LKM

Código de ejemplo (hello.c):

```
/**
 * @file hello.c
 * @author Derek Molloy
 * @date 4 April 2015
 * @version 0.1
 * @brief An introductory "Hello World!" loadable kernel module (LKM) that can
display a message
 * in the /var/log/kern.log file when the module is loaded and removed. The module
can accept an
 * argument when it is loaded -- the name, which appears in the kernel log files.
 * @see http://www.derekmolloy.ie/ for a full description and follow-up descriptions.
 */

#include <linux/init.h>          // Macros used to mark up functions e.g., __init
__exit
#include <linux/module.h>        // Core header for loading LKMs into the kernel
#include <linux/kernel.h>        // Contains types, macros, functions for the kernel

MODULE_LICENSE("GPL");           ///< The license type -- this affects runtime
behavior
MODULE_AUTHOR("Nora");           ///< The author -- visible when you use modinfo
MODULE_DESCRIPTION("A simple Linux driver for the BBB."); ///< The
description -- see modinfo
MODULE_VERSION("0.1");           ///< The version of the module

static char *name = "world";      ///< An example LKM argument -- default value is
"world"
module_param(name, charp, S_IRUGO); ///< Param desc. charp = char ptr,
S_IRUGO can be read/not changed
MODULE_PARM_DESC(name, "The name to display in /var/log/kern.log"); ///<
parameter description

/** @brief The LKM initialization function
 * The static keyword restricts the visibility of the function to within this C file. The
__init
 * macro means that for a built-in driver (not a LKM) the function is only used at
initialization
 * time and that it can be discarded and its memory freed up after that point.
 * @return returns 0 if successful
 */
static int __init helloBBB_init(void){
    printk(KERN_INFO "EBB: Hello %s from the BBB LKM!\n", name);
    return 0;
}

/** @brief The LKM cleanup function
```

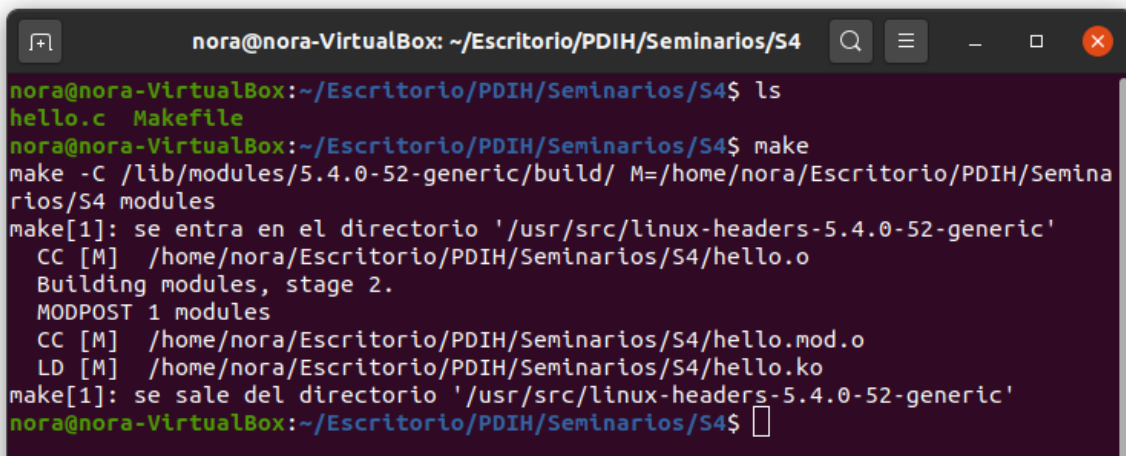
```

* Similar to the initialization function, it is static. The __exit macro notifies that if
this
* code is used for a built-in driver (not a LKM) that this function is not required.
*/
static void __exit helloBBB_exit(void){
    printk(KERN_INFO "EBB: Goodbye %s from the BBB LKM!\n", name);
}

/** @brief A module must use the module_init() module_exit() macros from
linux/init.h, which
* identify the initialization function at insertion time and the cleanup function (as
* listed above)
*/
module_init(helloBBB_init);
module_exit(helloBBB_exit);

```

Listado de lo que tenemos en la carpeta del seminario y compilación:

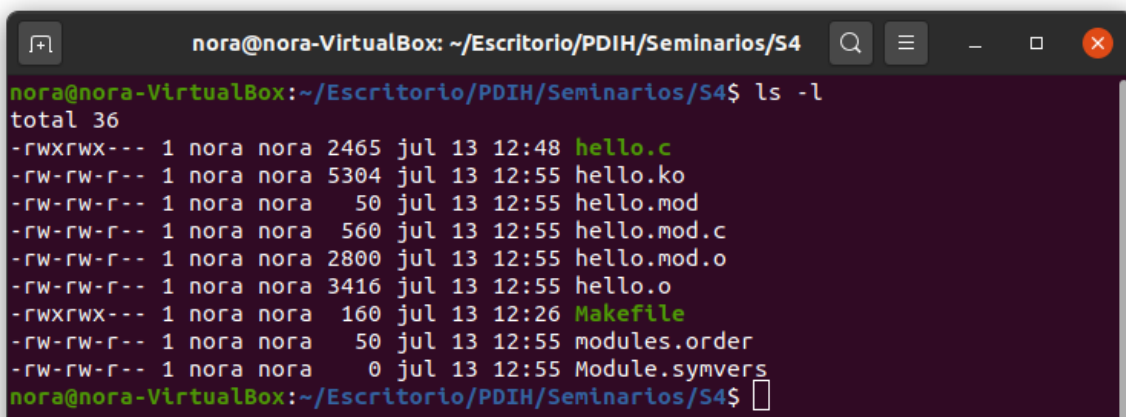


```

nora@nora-VirtualBox: ~/Escritorio/PDIH/Seminarios/S4
nora@nora-VirtualBox:~/Escritorio/PDIH/Seminarios/S4$ ls
hello.c  Makefile
nora@nora-VirtualBox:~/Escritorio/PDIH/Seminarios/S4$ make
make -C /lib/modules/5.4.0-52-generic/build/ M=/home/nora/Escritorio/PDIH/Seminarios/S4 modules
make[1]: se entra en el directorio '/usr/src/linux-headers-5.4.0-52-generic'
  CC [M]  /home/nora/Escritorio/PDIH/Seminarios/S4/hello.o
Building modules, stage 2.
MODPOST 1 modules
  CC [M]  /home/nora/Escritorio/PDIH/Seminarios/S4/hello.mod.o
  LD [M]  /home/nora/Escritorio/PDIH/Seminarios/S4/hello.ko
make[1]: se sale del directorio '/usr/src/linux-headers-5.4.0-52-generic'
nora@nora-VirtualBox:~/Escritorio/PDIH/Seminarios/S4$

```

Ahora realizamos un listado de lo que contiene el directorio tras la compilación:



```

nora@nora-VirtualBox: ~/Escritorio/PDIH/Seminarios/S4
nora@nora-VirtualBox:~/Escritorio/PDIH/Seminarios/S4$ ls -l
total 36
-rwxrwx--- 1 nora nora 2465 jul 13 12:48 hello.c
-rw-rw-r-- 1 nora nora 5304 jul 13 12:55 hello.ko
-rw-rw-r-- 1 nora nora  50 jul 13 12:55 hello.mod
-rw-rw-r-- 1 nora nora  560 jul 13 12:55 hello.mod.c
-rw-rw-r-- 1 nora nora 2800 jul 13 12:55 hello.mod.o
-rw-rw-r-- 1 nora nora 3416 jul 13 12:55 hello.o
-rwxrwx--- 1 nora nora  160 jul 13 12:26 Makefile
-rw-rw-r-- 1 nora nora  50 jul 13 12:55 modules.order
-rw-rw-r-- 1 nora nora   0 jul 13 12:55 Module.symvers
nora@nora-VirtualBox:~/Escritorio/PDIH/Seminarios/S4$

```


Insertamos el nuevo módulo en el kernel (insmod) y comprobamos el correcto funcionamiento (lsmod):

```
nora@nora-VirtualBox:~/Escritorio/PDIH/Seminarios/S4$ sudo insmod hello.ko
nora@nora-VirtualBox:~/Escritorio/PDIH/Seminarios/S4$ lsmod
Module                  Size  Used by
hello                   16384  0
vboxsf                  81920  1
vboxvideo               36864  0
nls_iso8859_1           16384  1
intel_rapl_msr          20480  0
intel_rapl_common       24576  1 intel_rapl_msr
intel_powerclamp        20480  0
crct10dif_pclmul        16384  1
ghash_clmulni_intel     16384  0
aesni_intel             372736 0
crypto_simd             16384  1 aesni_intel
snd_intel8x0             45056  4
cryptd                  24576  2 crypto_simd,ghash_clmulni_intel
snd_ac97_codec          131072 1 snd_intel8x0
glue_helper              16384  1 aesni_intel
```

Ahora, hacemos una petición de información sobre el módulo hello.ko (modinfo):

```
nora@nora-VirtualBox:~/Escritorio/PDIH/Seminarios/S4$ modinfo hello.ko
filename:               /home/nora/Escritorio/PDIH/Seminarios/S4/hello.ko
version:                0.1
description:             A simple Linux driver for the BBB.
author:                 Nora
license:                 GPL
srcversion:              5487CF5FB142463335FA790
depends:
retpoline:              Y
name:                   hello
vermagic:                5.4.0-52-generic SMP mod_unload
parm:                   name:The name to display in /var/log/kern.log (charp)
nora@nora-VirtualBox:~/Escritorio/PDIH/Seminarios/S4$
```

A continuación, lo eliminamos del kernel:

```
nora@nora-VirtualBox:~/Escritorio/PDIH/Seminarios/S4$ sudo rmmod hello.ko
nora@nora-VirtualBox:~/Escritorio/PDIH/Seminarios/S4$
```

Por último, revisaremos la salida de la función printk() en el registro de log del kernel:

```
root@nora-VirtualBox: /var/log

nora@nora-VirtualBox:~/Escritorio/PDIH/Seminarios/S4$ sudo su -
root@nora-VirtualBox:~# cd /var/log
root@nora-VirtualBox:/var/log# tail -f kern.log
Jul 13 11:37:10 nora-VirtualBox kernel: [ 94.773454] 09:37:10.324369 automount vbsvcAutomounterMountIt: Successfully mounte
Jul 13 11:37:20 nora-VirtualBox kernel: [ 104.990458] rfkill: input handler disabled
Jul 13 11:38:28 nora-VirtualBox kernel: [ 172.494531] rfkill: input handler enabled
Jul 13 11:38:51 nora-VirtualBox kernel: [ 196.389459] rfkill: input handler disabled
Jul 13 12:43:07 nora-VirtualBox kernel: [ 4052.420555] kauditd_printk_skb: 26 callbacks suppressed
Jul 13 12:43:07 nora-VirtualBox kernel: [ 4052.420556] audit: type=1400 audit(1626172987.956:38): apparmor="STATUS" operation
="apparmor_parser"
Jul 13 12:43:07 nora-VirtualBox kernel: [ 4052.424394] audit: type=1400 audit(1626172987.960:39): apparmor="STATUS" operation
rмор_parser"
Jul 13 12:43:07 nora-VirtualBox kernel: [ 4052.438168] audit: type=1400 audit(1626172987.972:40): apparmor="STATUS" operation
armor_parser"
Jul 13 12:56:17 nora-VirtualBox kernel: [ 4841.586429] EBB: Hello world from the BBB LKM!
Jul 13 12:57:30 nora-VirtualBox kernel: [ 4914.973698] EBB: Goodbye world from the BBB LKM!
```

PRÁCTICA 4: POSTSCRIPT

Ejercicio 1

Código ejercicio 1: Código Postscript que imprime como resultado de su interpretación la imagen de una casa representada mediante figuras geométricas.

```
%!  
  
/outputtext {  
  /data exch def  
  /rot exch def  
  /xfont exch def  
  /Times-Roman findfont  
  xfont scalefont  
  setfont  
  /y1 exch def  
  /x1 exch def  
  x1 y1 moveto  
  rot rotate  
  data show  
  
  rot neg rotate %sirve para ángulos negativos  
} def %definición de una función para rotar texto  
  
% x y fontsize rotation (text) outputtext  
20 420 12 90 (mi casita) outputtext  
  
% Rectángulo del fondo de la casa  
newpath  
% Start a new path  
35 400 moveto  
120  
0 rlineto  
0  
90 rlineto  
-120 0 rlineto  
closepath  
0.7 setgray  
fill  
  
35 400 moveto  
120  
0 rlineto  
0  
90 rlineto  
-120 0 rlineto  
closepath  
0 0.5 1 setrgbcolor  
0.5 setlinewidth
```

```
stroke
% finish the path (paint the lines)
%Fin rectángulo del fondo de la casa
```

```
% Ventana izquierda
```

```
newpath
```

```
% Start a new path
```

```
55 450 moveto
```

```
20
```

```
0 rlineto
```

```
0
```

```
15 rlineto
```

```
-20 0 rlineto
```

```
closepath
```

```
0.7 setgray
```

```
fill
```

```
55 450 moveto
```

```
20
```

```
0 rlineto
```

```
0
```

```
15 rlineto
```

```
-20 0 rlineto
```

```
closepath
```

```
0 0.5 1 setrgbcolor
```

```
0.5 setlinewidth
```

```
stroke
```

```
% finish the path (paint the lines)
```

```
% Ventana derecha
```

```
newpath
```

```
% Start a new path
```

```
115 450 moveto
```

```
20
```

```
0 rlineto
```

```
0
```

```
15 rlineto
```

```
-20 0 rlineto
```

```
closepath
```

```
0.7 setgray
```

```
fill
```

```
115 450 moveto
```

```
20
```

```
0 rlineto
```

```
0
```

```
15 rlineto
```

```
-20 0 rlineto
```

```
closepath
```

```

0 0.5 1 setrgbcolor
0.5 setlinewidth
stroke
% finish the path (paint the lines)

% Puerta
newpath
% Start a new path
85 400 moveto
20
0 rlineto
0
30 rlineto
-20 0 rlineto
closepath
0.7 setgray
fill

85 400 moveto
20
0 rlineto
0
30 rlineto
-20 0 rlineto
closepath
0 0.5 1 setrgbcolor
0.5 setlinewidth
stroke
% finish the path (paint the lines)

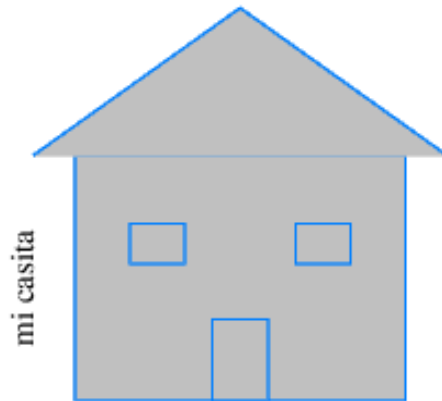
% Tejado
newpath
% Start a new path
%relleno
20 490 moveto
75 54 rlineto
75 -54 rlineto
0.7 setgray
fill

%marco
20 490 moveto
75 54 rlineto
75 -54 rlineto
0 0.5 1 setrgbcolor
0.5 setlinewidth
stroke
% finish the path (paint the lines)

```

```
showpage
% We're done
```

Resultado del código Postscript implementado:



Ejercicio 2

Código ejercicio 1: Código Postscript que imprime como resultado de su interpretación una tarjeta de visita

```
%!IPS

newpath
% Start a new path
100 400 moveto
400
0 rlineto
0
250 rlineto
-400 0 rlineto
closepath
0.9 setgray
fill
stroke
% finish the path (paint the lines)

newpath
% x y R angl angF
200 530 45 0 360 arc % dibujar un círculo gris
0.7 setgray % poner el color a gris claro
fill % rellenamos la figura recién terminada
```

```
0 setgray
1 setlinewidth
200 530 45 0 360 arc % dibujar una circunferencia negra
stroke                % no se pinta nada hasta esta instrucción

/Times-Roman findfont
15 scalefont
setfont
newpath
370 530 moveto
(Drago Kovacevic) show
stroke
0.6 setgray    %cambiar a color gris claro menos acentuado, es decir, más oscuro

newpath
350 510 moveto
(Ejecutivo de Cuentas) show
stroke

newpath
330 480 moveto
(Po de la Castellana, 141) show
stroke

newpath
310 465 moveto
(Local 2 - Edificio Cuzco IV) show
stroke

newpath
390 450 moveto
(28046 Madrid) show
stroke

newpath
290 435 moveto
(Tels. 914 174 550/669 896 224) show
stroke
newpath
370 420 moveto
(drago@serotel.es) show
stroke
showpage          % mostrar la página
```

Resultado de la implementación de código PostScript mostrado anteriormente:



PRÁCTICA 5: SONIDO

Código implementado para alcanzar los requisitos de la práctica:

```
library(tuneR)
library(seewave)
#Cargamos los archivos
perro <- readWave('C:/Users/noraa/Desktop/P5/perro.wav')
perro
gato <- readMP3('C:/Users/noraa/Desktop/P5/gato.mp3')
gato

#Representamos gráficamente las frecuencias de las señales de los archivos
recién extraídos
plot( extractWave(perro, from = 1, to = 2914304) )
plot( extractWave(gato, from = 1, to = 1724544) )

#Obtenemos los valores de las variables
str(gato)
str(perro)

#Sumamos las frecuencias de los archivos leídos anteriormente y representamos
el resultado de dicha suma
s <- pastew(perro, gato, output="Wave")
s
plot( extractWave(s, from = 1, to=4638848) )

#Asignamos el valor de la frecuencia a 44100Hz
f <- 44100

#Filtramos
sndF <- bwfilter(s, f=f, channel=1, n=1, from=10000, to=20000, bandpass=TRUE,
listen = FALSE, output = "Wave")

#Guardamos en el fichero mezcla.wav
writeWave(sndF, file.path('C:/Users/noraa/Desktop/P5/mezcla.wav') )

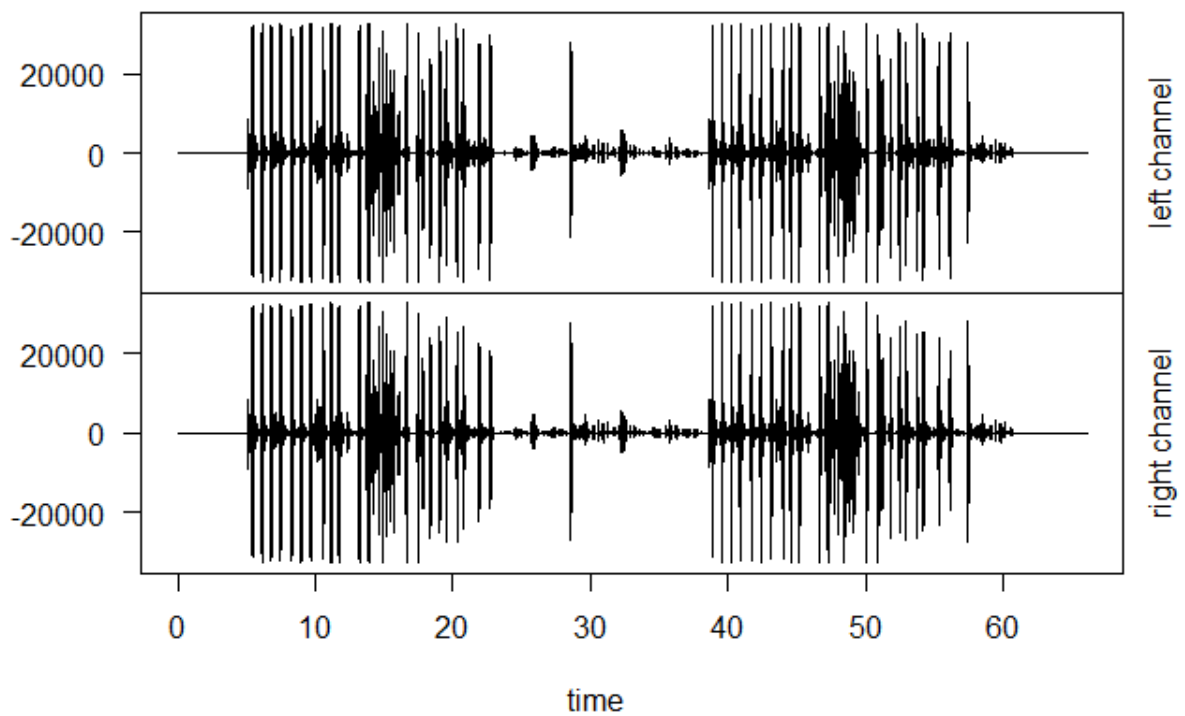
#Cargamos el archivo recién creado
mezcla <- readWave('C:/Users/noraa/Desktop/P5/mezcla.wav')

#Obtenemos los valores de la variable mezcla
str(mezcla)

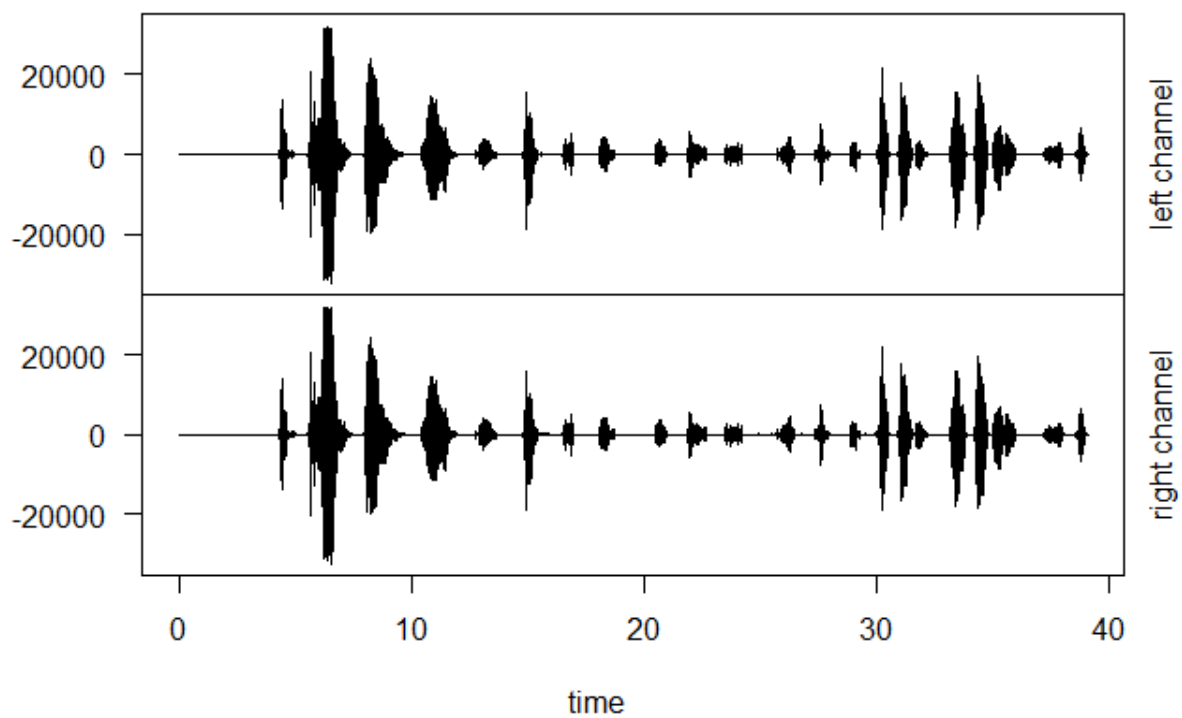
#Aplicamos eco al archivo
mezclaECO <-
echo(mezcla,f=22050,amp=c(0.8,0.4,0.2),delay=c(1,2,3),output="Wave")

#Damos la vuelta al sonido
reverse <- revw(mezcla, output="Wave")
#Guardamos en el fichero reverse.wav
writeWave(reverse, file.path("C:/Users/noraa/Desktop/P5/reverse.wav") )
```

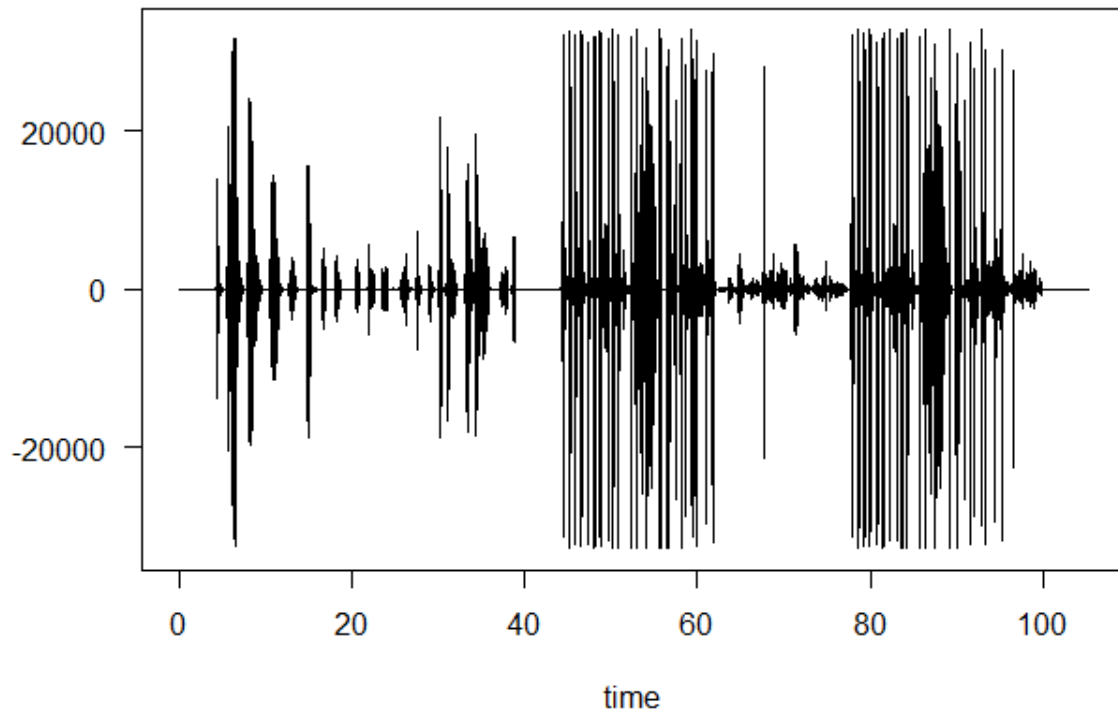
Representación gráfica de las frecuencias de las señales del archivo perro.wav:



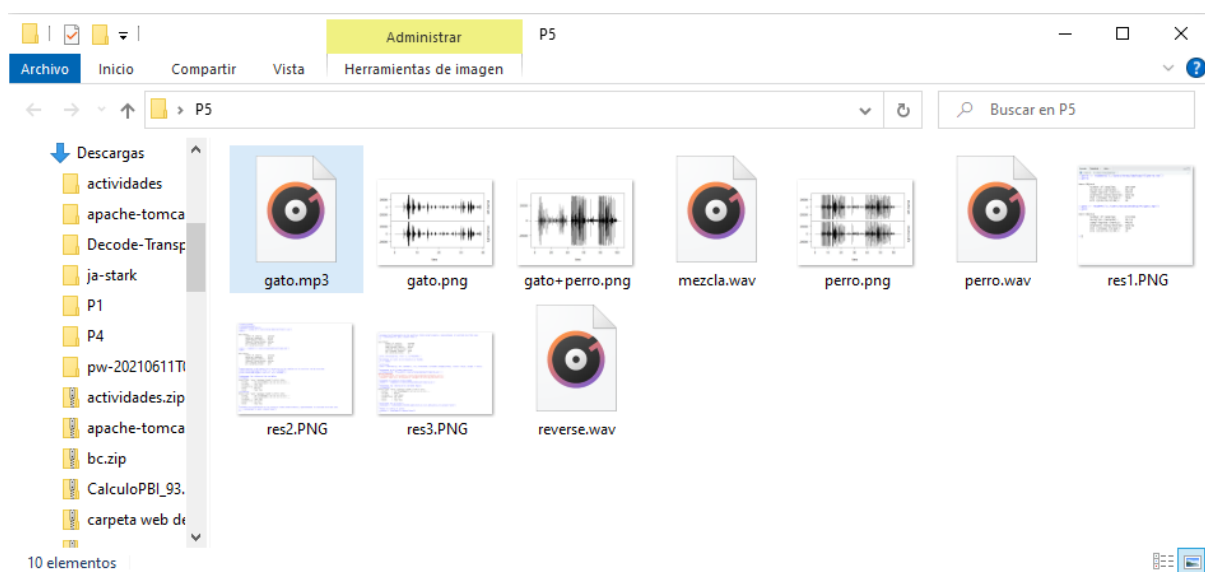
Representación gráfica de las frecuencias de las señales del archivo gato.mp3:



Representación gráfica de las frecuencias de las señales de la suma de los archivos gato.mp3 y perro.wav:



Comprobamos la correcta ejecución de la creación del fichero de mezcla:



Resultado de la ejecución del Script mostrado anteriormente(console):

```
> library(tuner)
> library(seewave)
> #Cargamos los archivos
> perro <- readwave('C:/Users/noraa/Desktop/P5/perro.wav')
> perro

Wave Object
  Number of Samples: 2914304
  Duration (seconds): 66.08
  Samplingrate (Hertz): 44100
  Channels (Mono/Stereo): Stereo
  PCM (integer format): TRUE
  Bit (8/16/24/32/64): 16

> gato <- readMP3('C:/Users/noraa/Desktop/P5/gato.mp3')
> gato

Wave Object
  Number of Samples: 1724544
  Duration (seconds): 39.11
  Samplingrate (Hertz): 44100
  Channels (Mono/Stereo): Stereo
  PCM (integer format): TRUE
  Bit (8/16/24/32/64): 16

>
> #Representamos gráficamente las frecuencias de las señales de los archivos recién extraídos
> plot( extractWave(perro, from = 1, to = 2914304) )
> plot( extractWave(gato, from = 1, to = 1724544) )
>
> #Obtenemos los valores de las variables
> str(gato)
Formal class 'wave' [package "tuner"] with 6 slots
 ..@ left      : int [1:1724544] 0 0 0 0 0 0 0 0 0 0 ...
 ..@ right     : int [1:1724544] 0 0 0 0 0 0 0 0 0 0 ...
 ..@ stereo    : logi TRUE
 ..@ samp.rate: num 44100
 ..@ bit       : num 16
 ..@ pcm       : logi TRUE
> str(perro)
Formal class 'wave' [package "tuner"] with 6 slots
 ..@ left      : int [1:2914304] 0 0 0 0 0 0 0 0 0 0 ...
 ..@ right     : int [1:2914304] 0 0 0 0 0 0 0 0 0 0 ...
 ..@ stereo    : logi TRUE
 ..@ samp.rate: int 44100
 ..@ bit       : int 16
 ..@ pcm       : logi TRUE
>
> #Sumamos las frecuencias de los archivos leídos anteriormente y representamos el resultado de dicha suma
> s <- pastew(perro, gato, output="wave")
> s

Wave Object
  Number of Samples: 4638848
  Duration (seconds): 105.19
  Samplingrate (Hertz): 44100
  Channels (Mono/Stereo): Mono
  PCM (integer format): TRUE
  Bit (8/16/24/32/64): 16

> plot( extractWave(s, from = 1, to=4638848) )
>
> #Asignamos el valor de la frecuencia a 44100Hz
> f <- 44100
>
> #Filtramos
> sndF <- bwfilter(s, f=f, channel=1, n=1, from=10000, to=20000, bandpass=TRUE, listen = FALSE, output = "wave")
>
> #Guardamos en el fichero mezcla.wav
> writewave(sndF, file.path('C:/Users/noraa/Desktop/P5/mezcla.wav') )
Warning message:
In writewave(sndF, file.path("C:/Users/noraa/Desktop/P5/mezcla.wav")) :
  channels' data will be interpreted as integers for writing the wave file
>
> #Cargamos el archivo recién creado
> mezcla <- readwave('C:/Users/noraa/Desktop/P5/mezcla.wav')
>
> #Obtenemos los valores de la variable mezcla
> str(mezcla)
Formal class 'wave' [package "tuner"] with 6 slots
 ..@ left      : int [1:4638848] 0 0 0 0 0 0 0 0 0 0 ...
 ..@ right     : num(0)
 ..@ stereo    : logi FALSE
 ..@ samp.rate: int 44100
 ..@ bit       : int 16
 ..@ pcm       : logi TRUE
>
> #Aplicamos eco al archivo
> mezclaECO <- echo(mezcla,f=22050,amp=c(0.8,0.4,0.2),delay=c(1,2,3),output="wave")
>
> #Damos la vuelta al sonido
> reverse <- revw(mezcla, output="wave")

> #Guardamos en el fichero reverse.wav
> writewave(reverse, file.path("C:/Users/noraa/Desktop/P5/reverse.wav") )
> |
```