|  |  |  |
| --- | --- | --- |
| **ipn** | **INSTITUTO POLITÉCNICO NACIONAL**  **ESCUELA SUPERIOR DE CÓMPUTO** |  |

**Teoría de Comunicaciones y Señales**

**“DTMF”**

Resumen

Implementación en lenguaje C del algoritmo DTMF para reconocimiento de las frecuencias de un teclado de un celular y saber que tecla fue presionada, creando un tablero de gato en Python para jugar entre 2 clientes.

**Por:**

**Joel Mauricio Romero Gamarra**

**René Zavala Pérez**

Profesor:

GUTIÉRREZ ALDANA EDUARDO

Diciembre 2017

**Índice**

Contenido

[Introducción: 1](#_Toc501396467)

[Análisis Teórico: 2](#_Toc501396468)

[Software (librerías, paquetes, herramientas): 3](#_Toc501396469)

[Procedimiento: 4](#_Toc501396470)

[Resultados 4](#_Toc501396471)

[Conclusiones: 9](#_Toc501396472)

[Referencias 10](#_Toc501396473)

[Código 10](#_Toc501396474)

# Introducción:

El sistema de marcación por tonos, llamado DTMF es utilizado para la señalización de las comunicaciones. En 1948, Western Electric realizó una serie de pruebas modificando uno de los aparatos telefónicos con un teclado de diez pulsadores dispuestos en dos filas. Al pulsar cualquiera de las teclas, se producían dos tonos por cada dígito, de seis tonos posibles. El experimento se basó en un diseño de Bell realizado en 1941. [1]

Los tonos de señalización DTMF también se pueden escuchar al comienzo o al final de algunas cintas de casete VHS ya que la información sobre la versión de la cinta de vídeo fue codificada con una secuencia de dichos tonos. Este código proporciona información a las máquinas de duplicación automática, como el formato, la duración y el nivel del audio, con el fin de reproducir lo más fielmente el vídeo original. [1]

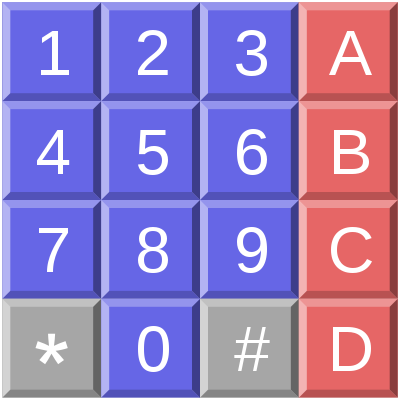


Figura 1. Teclado de DTMF

Al ser pulsada en el teléfono la tecla correspondiente al dígito que quiere marcar, se envían dos tonos, de distinta frecuencia: uno por columna y otro por fila en la que esté la tecla, que la central decodifica a través de filtros especiales, detectando qué dígito se marcó, hay que tener las siguientes consideraciones:

* Cada frecuencia puede variar ± 1.8 % de la frecuencia normal.
* Los productos de distorsión producidor por intermodulación deben tener un nivel de 20 dB por debajo de los que tienen frecuencias fundamentales.
* Al pulsar una tecla, el sonido resultante es la suma de un tono alto (columnas del teclado) con uno bajo (filas del teclado), teniendo el primero un nivel de 2 dB (25.89%) respecto del otro para compensar las pérdidas de señal en las líneas de conexión con la central telefónica.

A continuación, se muestra la tabla que representa las frecuencias que forman el teclado para formar la DTMF.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hz** | **1209** | **1336** | **1477** | **1633** |
| **697** | **1** | **2** | **3** | **A** |
| **770** | **4** | **5** | **6** | **B** |
| **852** | **7** | **8** | **9** | **C** |
| **941** | **\*** | **0** | **#** | **D** |

*Tabla 1. Tabla de frecuencias para DTMF*

En la siguiente sección se explica el procedimiento para calcular cada una de las frecuencias.

# Análisis Teórico:

Como ya se había comentado en prácticas anteriores, sabemos que con la Transformada de Fourier podemos pasar del dominio del tiempo al dominio de la frecuencia, sin embargo, sabemos que el algoritmo de la TDF es bastante costoso ya que se vuelve cuadrático, mientras que para calcular la Transformada Rápida de Fourier, al ser divide y vencerás es un algoritmo con una complejidad bastante aceptable representado por una cota O (n · log (n)), así que sería fácil escoger la FFT sobre la TDF, sin embargo, al hacer un acomodo de las muestras y un arreglo de mariposa, es un poco complicado saber el transformada para un cierto valor en el tiempo, cosa que con la TDF es bastante fácil ya que el valor en el tiempo del que queremos la TDF está representado por k en la fórmula ya conocida:

Así que calcular para un cierto valor de k, es bastante rápido, sin embargo, ¿por qué solamente necesitamos la TDF para un cierto valor?

Esto es, debido a que cada dígito del teclado está definido para unas frecuencias específicas, con una variación muy pequeña, así que ya sabemos en que parte del tiempo se esconden esas frecuencias, al tener una frecuencia de muestreo de 44,100 Hz y una duración de 0.5 segundos, quiere decir que tenemos 22,050 muestras, de las cuales, si dividimos 22,050 entre 44,100 nos da 0.5, esto quiere decir que en cada muestra hay una frecuencia de 0.5 Hz.

Así, para saber el valor de k donde necesitamos calcular la TDF debemos de realizar la siguiente operación:

Ahora, ya que sabemos en qué valor de k necesitamos calcular la TDF, tenemos un pequeño problema, y es que al recibir sonido PCM por medio de la tarjeta de sonido en lenguaje C, el sonido se recibe en 2 canales, es decir, un archivo estéreo, sin embargo, nosotros sabemos calcular la TDF para un archivo de 1 solo canal (la ITDF si se calcula para un archivo con 2 canales), así que lo que debemos hacer es lo siguiente:

* Recibir el sonido en un arreglo suficientemente grande para almacenar las muestras.
* Eliminar o ignorar todas las posiciones impares del arreglo comenzando por 1 (ya que representan al canal derecho).
* Trabajar con el arreglo resultante como un archivo monoaural.

Ya que tenemos el archivo monoaural, debemos definir las frecuencias (que ya son estáticas), y cada uno de los dígitos que son representados por 2 frecuencias (en una matriz de 4 x 4, simulando el teclado de la DTMF) para una mayor comodidad.

Ya que se tienen las frecuencias calculadas para ciertos valores de k (como sabemos, hay una parte real y una imaginaria, pero nos interesa la magnitud por el momento), debemos de saber cual es la frecuencia más alta distinta de 0, ya que nos indicará el digito que se presionó por medio del análisis de las frecuencias gracias a la TDF.

# Software (librerías, paquetes, herramientas):

Hacer una lista de TODOS los ítems utilizados en el laboratorio. Alternativamente, materiales pueden ser incluidos como parte del procedimiento.

**LIBRERÍAS:**

* asoundlib.h: Funciones utilizadas para recibir audio formato PCM
  + snd\_pcm\_open (snd\_pcm\_t \*\*pcm, const char \*name, snd\_pcm\_stream\_t stream, int mode)
  + snd\_pcm\_hw\_params\_malloc (snd\_pcm\_hw\_params\_t \*\*ptr)
  + snd\_pcm\_hw\_params\_any (snd\_pcm\_t \*pcm, snd\_pcm\_hw\_params\_t \*params)
  + snd\_pcm\_hw\_params\_set\_access (snd\_pcm\_t \*pcm, snd\_pcm\_hw\_params\_t \*params, snd\_pcm\_access\_t \_access)
  + snd\_pcm\_hw\_params\_set\_format (snd\_pcm\_t \*pcm, snd\_pcm\_hw\_params\_t \*params, snd\_pcm\_format\_t val)
  + snd\_pcm\_hw\_params\_set\_rate\_near (snd\_pcm\_t \*pcm, snd\_pcm\_hw\_params\_t \*params, unsigned int \*val, int \*dir)
  + snd\_pcm\_hw\_params\_set\_channels (snd\_pcm\_t \*pcm, snd\_pcm\_hw\_params\_t \*params, unsigned int val)
  + snd\_pcm\_hw\_params\_set\_period\_size\_near (snd\_pcm\_t \*pcm, snd\_pcm\_hw\_params\_t \*params, snd\_pcm\_uframes\_t \*val, int \*dir)
  + snd\_pcm\_hw\_params (snd\_pcm\_t \*pcm, snd\_pcm\_hw\_params\_t \*params)
  + snd\_pcm\_hw\_params\_get\_period\_size (const snd\_pcm\_hw\_params\_t \*params, snd\_pcm\_uframes\_t \*frames, int \*dir)
  + snd\_pcm\_hw\_params\_get\_period\_time (const snd\_pcm\_hw\_params\_t \*params, unsigned int \*val, int \*dir)
  + snd\_pcm\_hw\_params\_free (snd\_pcm\_hw\_params\_t \*obj)
  + snd\_pcm\_readi (snd\_pcm\_t \*pcm, void \*buffer, snd\_pcm\_uframes\_t size)

# Procedimiento:

Primero decidimos escoger Python como interacción con c, ya que tiene distintas opciones de gráficos programables, siendo solo en 2D o una interfaz en 3D, también Python es un lenguaje de programación muy amplio en cuestión de conexión con distintos lenguajes, se realizó una conexión TCP con sockets entre C y Python, tomando Python como servidor ya que en este lenguaje se montó la interfaz y se harán cambios respecto a las respuestas que se manden desde C.

Para el desarrollo del juego de gato en Python se utilizó un módulo en Python llamado Graphics, el cual te genera una ventana externa en la cual te permite dibujar líneas, cuadros, puntos etc.

Al cargar el servidor se genera una plantilla la cual te da instrucciones para que sepas que numero del DTMF es cada casilla del gato, así como iniciar un nuevo juego o terminar el juego, al recibir la instrucción del iniciar juego se borran las instrucciones con un método llamado undraw () y se generan los turnos para cada jugador, al ser un juego solo de dos personas el llamado de turnos se realizó mediante un contador, dicho contador aumenta con cada tiro y se decide el turno de cada jugador si el contador es par o impar, al pasar el sonido de la DTMF por nuestro cliente esta realiza una TDF y determina que numero de casilla se eligió, esta respuesta es enviada por el socket TCP al servidor y si la casilla ya fue seleccionada le pide al jugador que haga otra jugada, si la casilla está vacía dibuja una marca con el método Draw (), decidiendo si es Jugador 1(X) si es Jugador 2 (O), la partida termina cuando uno de los dos jugadores rellena una fila o si el tablero se llena de marcas.

Al principio se pensaba realizar una conexión con 2 clientes, pero esto causo un conflicto dentro del módulo Graphics, ya que al estar el método Draw () dentro den un ciclo causaba confusión al momento de dibujar en pantalla, es por esa razón que se decidió solo un cliente jugando por turnos.

# Resultados

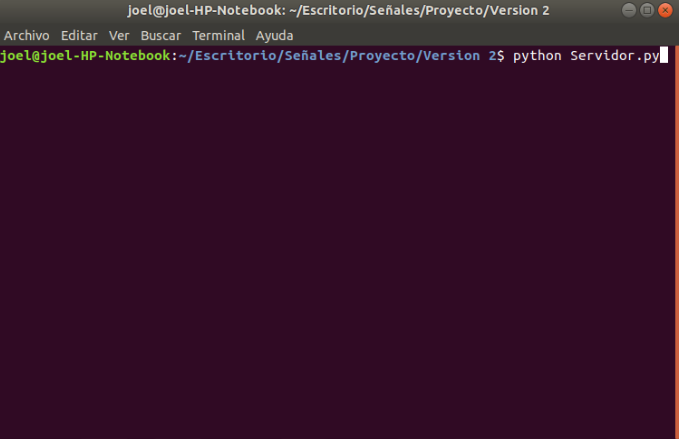


Figura 2. Ejecución del servidor.

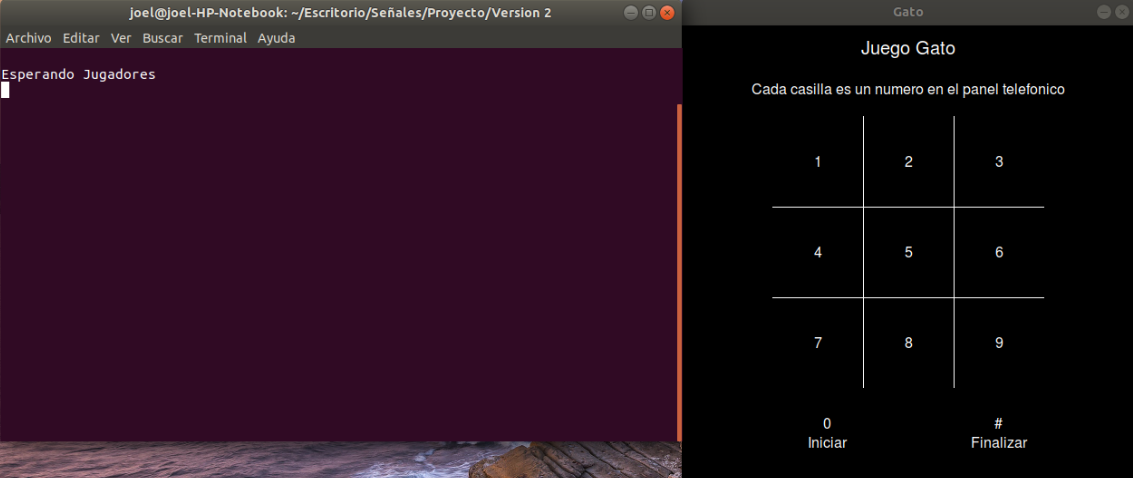


Figura 3. Servidor esperando 2 clientes para comenzar el juego

Como se puede apreciar en la Figura 3, el servidor espera jugadores mientras el tablero se muestra al usuario mostrando que tecla presionar para colocar su respectiva figura (X o O) y nos indica que para comenzar el juego hay que apretar la tecla 0, además, para finalizar la partida en cualquier momento, podemos apretar la tecla de símbolo #.

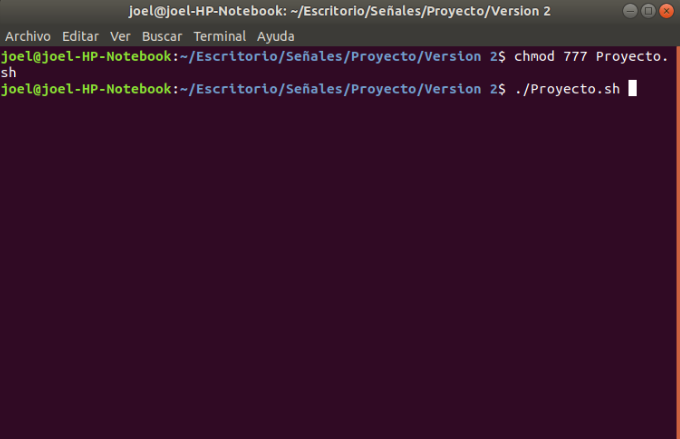


Figura 4. Ejecución del cliente por medio de un script

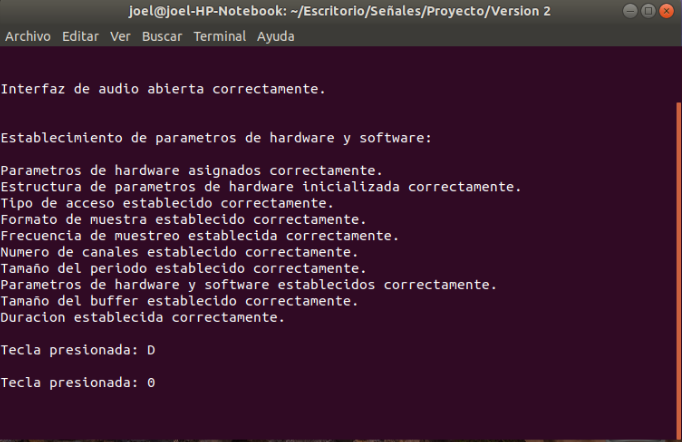


Figura 5. El cliente (jugador 1) comienza el juego al presionar 0

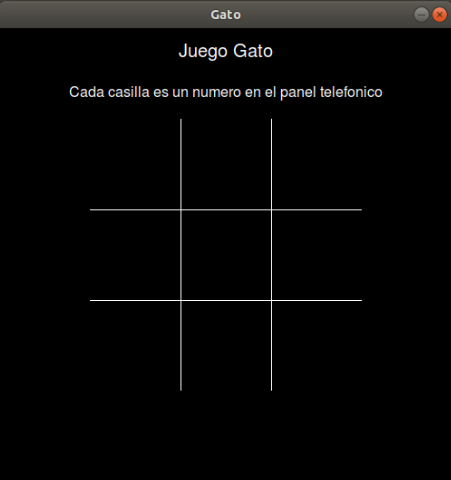


Figura 6. Tablero al iniciar el juego

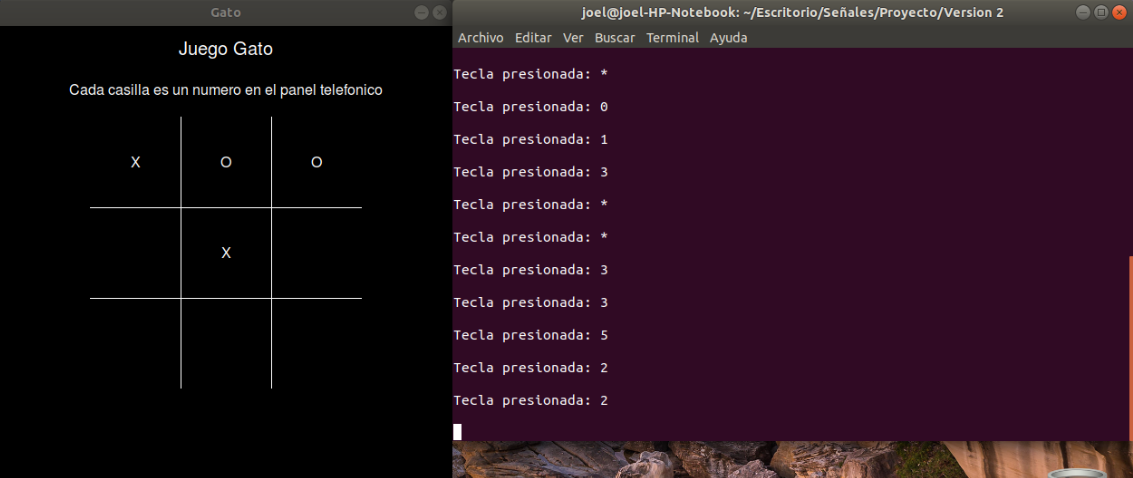


Figura 7. Juego en ejecución

Como se puede ver en la terminal del lado derecho (el cliente), se va viendo como aprieta el 0 para comenzar el juego, posteriormente se apretó el número 1 por lo que se colocó un X en la casilla 1, posteriormente, se presionó un número 3 y se colocó un O en la casilla 3, después, se presionó el 5 y se colocó un X en la posición 5 y finalmente un 2 y se colocó un O en la casilla correspondiente, debido a que el juego es por turnos.

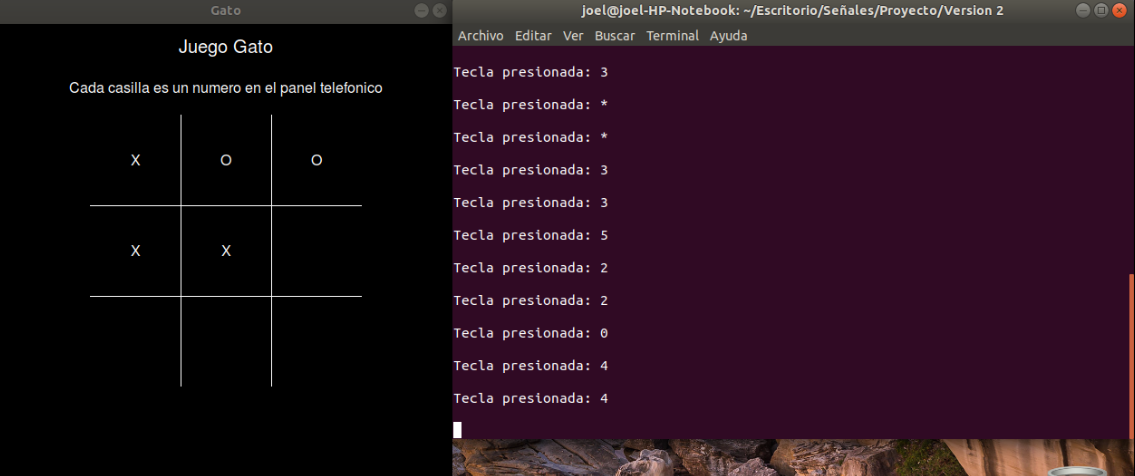


Figura 8. Juego en ejecución (2)

Como podemos ver, el tablero está como en la Figura pasada, pero se agregó un nuevo X en la posición 4 después de haber presionado el 2, y como podemos ver, si se aprietan otras teclas, no pasa nada, a menos que se presione la tecla #.

En la siguiente imagen se observa un nuevo movimiento del Jugador 2.

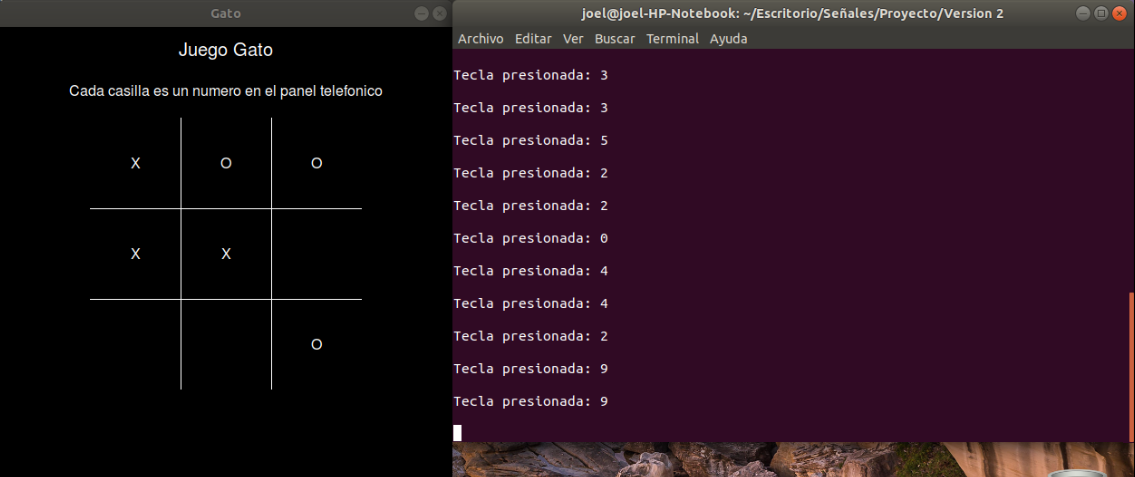


Figura 9. Juego en ejecución (3)

En la siguiente Figura se muestra la terminal cuando el usuario presiona la tecla #.

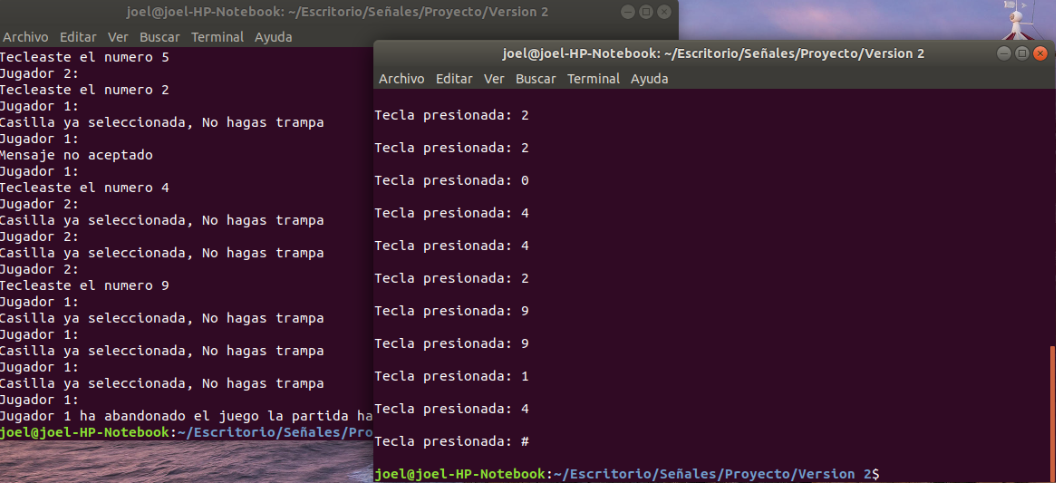


Figura 10. Juego terminado

Como se puede ver, en la terminal del lado derecho, es decir, del cliente se ve que la última tecla presionada fue #, debido a que con esta tecla se finalizaba el juego, la interfaz se cierra y en la terminal del lado izquierdo, es decir, del servidor, nos muestra el mensaje que el Jugador 1 (que es el que seguía), en vez de hacer un movimiento, abandonó el juego.

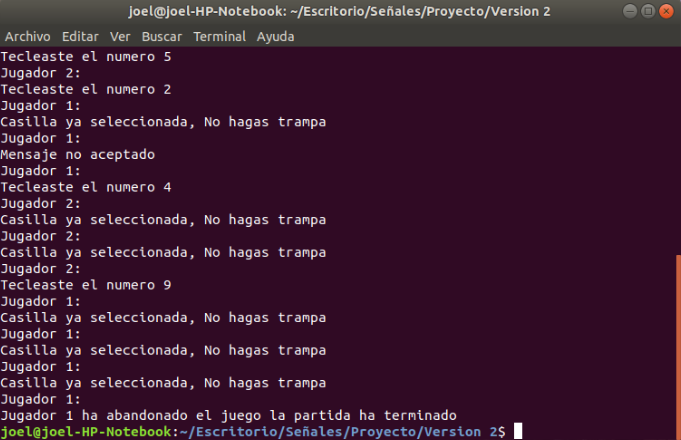


Figura 11. Terminal del servidor al finalizar el juego

# Conclusiones:

**RENÉ ZAVALA PÉREZ**

Con la realización del proyecto pude observar que el análisis de frecuencias tiene muchas aplicaciones distintas, en este caso fue el uso del DTMF para poder reconocer que tecla de nuestro teléfono fue presionada y poder realizar una acción en este caso el inicio de un juego Gato, pero si se desea en un futuro se pueden explorar mucho más cosas interesantes, como realizar proyectos por reconocimiento de voz, durante este curso he podido observar que dentro del análisis de señales se encuentra un enorme campo de estudio ya que tienes distintos problemas a resolver con el estudio de las mismas y hasta me hace pensar que la realización de proyectos actuales como con comando de voz es gracias a un análisis de frecuencias dentro de las señales

Para mí el curso fue muy importante ya que aprendí muchas funciones matemáticas programables, este curso para mi es el más completo, ya que tuve que tener conocimientos acerca de circuitos eléctricos, tuve que tener un buen manejo de lógica para la utilización de C y por último tuve que tener un buen análisis matemático para entender problemas como la convolución, TDF, TFF y más.

**JOEL MAURICIO ROMERO GAMARRA**

Este proyecto es prácticamente el más interesante que hemos realizado, ya que tiene muchas aplicaciones, por ejemplo, ahorita se hizo el análisis de frecuencias para un teclado de un teléfono con frecuencias ya definidas para un dígito, así que podemos hacer modificaciones para hacer el reconocimiento de vocales (que ya son formadas por ciertas frecuencias, pero en vez de una combinación de 2 frecuencias, son 3, pero aplica lo mismo).

Creo que el curso fue muy importante para darnos cuenta de todo lo que hemos aprendido y como aplicarlo en algo que utilizamos a diario, como lo es la telefonía celular para marcar algún número y como es que las grandes empresas (en este caso de telefonía) saben que número es el que marcamos, todo esto gracias a cursos pasados como matemáticas avanzadas para la ingeniería donde aprendimos la transformada de Fourier y ahora la aprendimos de manera discreta para hacer el análisis de frecuencias pertinente y hacer el reconocimiento de las teclas presionadas en un teclado.

# Referencias

|  |  |
| --- | --- |
| [1] | «Marcación por Tonos,» Noviembre 2017. [En línea]. Available: https://es.wikipedia.org/wiki/Marcaci%C3%B3n\_por\_tonos. [Último acceso: Diciembre 2017]. |

# Código

**Proyecto.sh**

#!/bin/bash

**if** [ -f DTMF ]; **then**

rm DTMF

**fi**

**if** [ -f Funciones**.**o ]; **then**

rm Funciones**.**o

**fi**

gcc Funciones**.**c -o Funciones**.**o -c -lm

gcc DTMF**.**c -o DTMF Funciones**.**o -lm -lasound

**.**/DTMF 3905 127**.**0**.**0**.**1

**Funciones.h**

#define TIEMPO 0.5

#define SAMPLES 22050

#define sample\_rate 44100

#define PI 3.141592653589

snd\_pcm\_t \* establecer\_parametros (snd\_pcm\_t \* manejador, **int** \* tam, **unsigned** **int** \* segundos);

**char** tecla\_presionada (**short** \* datos);

**float** TDF (**short** \* datos, **int** k);

**int** obtener\_fila (**float** \* muestras);

**int** obtener\_columna (**float** \* muestras);

**Funciones.c**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <alsa/asoundlib.h>

#include <string.h>

#include "Funciones.h"

/\*

establecer\_parametros (manejador, tam, segundos)

Recibe: Apuntador de tipo snd\_pcm\_t, tama�o de arreglo para muestras, n�mero de segundos que ser� le�do el sonido

Retorna: Manejador con todos los parametros correctos para recibir sonido

Efecto: Establecer los par�metros necesarios para recibir sonido, como: tipo de muestras, frecuencia de muestreo, numero de canales, etc.

Requerimientos: Contar con la instalaci�n de asoundlib en linux

\*/

snd\_pcm\_t \* establecer\_parametros (snd\_pcm\_t \* manejador, **int** \* tam, **unsigned** **int** \* segundos)

{

**int** flag, dir, frecuencia = sample\_rate;

**unsigned** **int** canales = 2, val;

**char** \* dispositivo = "default";

snd\_pcm\_uframes\_t muestras = SAMPLES;

snd\_pcm\_hw\_params\_t \* hw\_params;

snd\_pcm\_format\_t formato = SND\_PCM\_FORMAT\_S16\_LE; //Little Endian

snd\_pcm\_format\_t acceso = SND\_PCM\_ACCESS\_RW\_INTERLEAVED;

//Abrir la interfaz de audio para recibir sonido

flag = snd\_pcm\_open (&manejador, dispositivo, SND\_PCM\_STREAM\_CAPTURE, 0);

**if** (flag < 0)

{

printf ("No fue posible abrir el dispositivo %s: %s\n", dispositivo, snd\_strerror (flag));

exit (1);

}**else**

printf ("\n\nInterfaz de audio abierta correctamente.\n");

printf ("\n\nEstablecimiento de parametros de hardware y software:\n\n");

//Asignar par�metros de hardware

flag = snd\_pcm\_hw\_params\_malloc (&hw\_params);

**if** (flag < 0)

{

printf ("\n\nNo fue posible asignar la estructura de parametros de hardware: %s\n", snd\_strerror (flag));

exit (1);

}**else**

printf ("Parametros de hardware asignados correctamente.\n");

//Inicializar la estructura de par�metros de hardware

flag = snd\_pcm\_hw\_params\_any (manejador, hw\_params);

**if** (flag < 0)

{

printf ("\n\nNo fue posible inicializar la estructura de parametros de hardware: %s\n", snd\_strerror (flag));

exit (1);

}**else**

printf ("Estructura de parametros de hardware inicializada correctamente.\n");

//Establecer tipo de acceso

flag = snd\_pcm\_hw\_params\_set\_access (manejador, hw\_params, acceso);

**if** (flag < 0)

{

printf ("\n\nNo fue posible establecer tipo de acceso: %s\n", snd\_strerror (flag));

exit (1);

}**else**

printf ("Tipo de acceso establecido correctamente.\n");

//Establecer el formato de las muestras

flag = snd\_pcm\_hw\_params\_set\_format (manejador, hw\_params, formato);

**if** (flag < 0)

{

printf ("\n\nNo fue posible establecer formato de muestra: %s\n", snd\_strerror (flag));

exit (1);

}**else**

printf ("Formato de muestra establecido correctamente.\n");

//Establecer la frecuencia de muestreo

flag = snd\_pcm\_hw\_params\_set\_rate\_near (manejador, hw\_params, &frecuencia, &dir);

**if** (flag < 0)

{

printf ("\n\nNo fue posible establecer la frecuencia de muestreo: %s\n", snd\_strerror (flag));

exit (1);

}**else**

printf ("Frecuencia de muestreo establecida correctamente.\n");

//Establecer numero de canales (1 - Mono)

flag = snd\_pcm\_hw\_params\_set\_channels (manejador, hw\_params, canales);

**if** (flag < 0)

{

printf ("\n\nNo fue posible establecer el numero de canales: %s\n", snd\_strerror (flag));

exit (1);

}**else**

printf ("Numero de canales establecido correctamente.\n");

//Establecer el numero de frames

flag = snd\_pcm\_hw\_params\_set\_period\_size\_near (manejador, hw\_params, &muestras, &dir);

**if** (flag < 0)

{

printf ("\n\nNo fue posible establecer el tama�o del periodo: %s\n", snd\_strerror (flag));

exit (1);

}**else**

printf ("Tama�o del periodo establecido correctamente.\n");

//Establecer los parametros de hardware y software en el manejador

flag = snd\_pcm\_hw\_params (manejador, hw\_params);

**if** (flag < 0)

{

printf ("\n\nNo fue posible establecer los parametros de hardware y software: %s\n", snd\_strerror (flag));

exit (1);

}**else**

printf ("Parametros de hardware y software establecidos correctamente.\n");

//Usar un b�fer suficientemente grande para contener un periodo

flag = snd\_pcm\_hw\_params\_get\_period\_size (hw\_params, &muestras, &dir);

**if** (flag < 0)

{

printf ("\n\nNo fue posible establecer el tama�o del buffer: %s\n", snd\_strerror (flag));

exit (1);

}**else**

printf ("Tama�o del buffer establecido correctamente.\n");

\* tam = (muestras \* 4);

//Establecer un tiempo de 5 segundos para recibir sonido

flag = snd\_pcm\_hw\_params\_get\_period\_time (hw\_params, &val, &dir);

**if** (flag < 0)

{

printf ("\n\nNo fue posible establecer la duracion: %s\n", snd\_strerror (flag));

exit (1);

}**else**

printf ("Duracion establecida correctamente.\n\n");

\*segundos = ((TIEMPO \* 1000000) / val);

//Ya que se establecieron todos los parametros, se libera la estructura

snd\_pcm\_hw\_params\_free (hw\_params);

**return** manejador;

}

/\*

tecla\_presionada (datos)

Recibe: Arreglo que contiene los datos a procesar

Retorna: Char seg�n la tecla que presione el usuario

Efecto: Procesa la TDF analizando las frecuencias para saber que tecla presion� el usuario y lo devuelve como char

Requerimientos: El arreglo de los datos debe ser no vac�o

\*/

**char** tecla\_presionada (**short** \* datos)

{

**int** i, j, k, fila, columna, filas [4], columnas [4];

**int** frecuencias [8] = {697, 770, 852, 941, 1209, 1336, 1477, 1633};

**char** teclado [4][4] = {{'1', '2', '3', 'A'}, {'4', '5', '6', 'B'}, {'7', '8', '9', 'C'}, {'\*', '0', '#', 'D'}};

**float** \* muestras = (**float** \*) malloc (**sizeof** (**float**) \* 8);

**float** aux;

**for** (k = 0; k < 8; k ++)

{

aux = ((SAMPLES \* frecuencias [k]) / sample\_rate);

**if** (k < 4)

filas [k] = round (aux);

**else**

columnas [k - 4] = round (aux);

}

**for** (k = 0, j = 0; k < SAMPLES; k ++)

{

**for** (i = 0; i < 4; i ++)

**if** (k == filas [i] || k == columnas [i])

muestras [j ++] = TDF (datos, k);

}

muestras [8] = '\0';

fila = obtener\_fila (muestras);

columna = obtener\_columna (muestras);

**if** ((fila >= 0 && fila <= 3) && (columna >= 0 && columna <= 3))

**return** teclado [fila][columna];

}

/\*

TDF (datos, k)

Recibe: Arreglo que contiene los datos a procesar y entero para calcular la TDF justo en esa k

Retorna: TDF en la posici�n k del arreglo de los datos

Efecto: Procesa la TDF en un cierto valor de k

Requerimientos: El arreglo de los datos debe ser no vac�o

\*/

**float** TDF (**short** \* datos, **int** k)

{

**int** n, j;

**double** parte\_real = 0, parte\_imaginaria = 0, magnitud = 0;

**for** (n = 0, j = 0; n < SAMPLES; n ++, j += 2)

{

parte\_real += (datos [j] \* cos ((2 \* PI \* k \* n) / SAMPLES));

parte\_imaginaria -= (datos [j] \* sin ((2 \* PI \* k \* n) / SAMPLES));

}

parte\_real = pow (parte\_real, 2);

parte\_imaginaria = pow (parte\_imaginaria, 2);

magnitud = sqrt (parte\_real + parte\_imaginaria);

**return** magnitud;

}

/\*

obtener\_fila (muestras)

Recibe: Arreglo que contiene las frecuencias de la tecla leida

Retorna: El n�mero de fila donde se encuentra la tecla presionada

Efecto: Escoge la fila de la tecla que se presion�

Requerimientos: N/A

\*/

**int** obtener\_fila (**float** \* muestras)

{

**float** frecuencia\_maxima = -100000;

**int** i, fila = -1;

**for** (i = 0; i < 4; i ++)

**if** (muestras [i] > frecuencia\_maxima)

{

frecuencia\_maxima = muestras [i];

fila = i;

}

**return** fila;

}

/\*

obtener\_columna (muestras)

Recibe: Arreglo que contiene las frecuencias de la tecla leida

Retorna: El n�mero de columna donde se encuentra la tecla presionada

Efecto: Escoge la columna de la tecla que se presion�

Requerimientos: N/A

\*/

**int** obtener\_columna (**float** \* muestras)

{

**float** frecuencia\_maxima = -100000;

**int** i, columna = -1;

**for** (i = 4; i < 8; i ++)

**if** (muestras [i] > frecuencia\_maxima)

{

frecuencia\_maxima = muestras [i];

columna = (i - 4);

}

**return** columna;

}

**DTMF.c**

#include <stdio.h>

#include <stdlib.h>

#include <alsa/asoundlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include "Funciones.h"

**struct** sockaddr\_in conectServer;

**int** main(**int** argc, **char** **const** \*argv[])

{

system ("clear");

**int** i, flag, tam;

**int** socketCliente,co,sendi,r,puerto,direccion\_ip;

**char** tecla, aux = 'z';

**unsigned** **int** segundos;

snd\_pcm\_t \* manejador;

snd\_pcm\_uframes\_t muestras = SAMPLES;

**if** (argc < 3)

{

printf ("Error, faltan parametros\n");

printf ("Ejemplo: '%s 3905 127.0.0.1'\n\n", argv [0]);

exit (0);

}

socketCliente = socket (AF\_INET, SOCK\_STREAM, 0);

**if** (socketCliente < 0)

{

printf ("Error al crear el socket\n");

exit (0);

}

memset ((**void** \*)&conectServer, 0, **sizeof** (conectServer));

conectServer.sin\_family = AF\_INET;

conectServer.sin\_port = htons (atoi (argv [1]));

conectServer.sin\_addr.s\_addr = inet\_addr (argv [2]);

co = connect (socketCliente, (**struct** sockaddr \*)&conectServer, **sizeof** (conectServer));

**if** (co < 0)

{

printf ("Error de conexion\n");

exit (0);

}

manejador = establecer\_parametros (manejador, &tam, &segundos);

**short** \* datos = (**short** \*) malloc (**sizeof** (**short**) \* tam);

snd\_pcm\_prepare (manejador);

**while** (segundos > 0)

{

flag = snd\_pcm\_readi (manejador, datos, muestras);

**if** (flag < 0)

{

printf ("\n\nNo fue posible recibir audio: %s\n\n", snd\_strerror (flag));

snd\_pcm\_prepare (manejador);

}**else** **if** (flag == -EPIPE)

{

printf ("\n\nSe recibieron mas muestras de las esperadas: %s\n\n", snd\_strerror (flag));

snd\_pcm\_prepare (manejador);

}**else**

{

tecla = tecla\_presionada (datos);

**if** (aux == tecla)

{

printf ("Tecla presionada: %c\n\n", tecla);

sendi = send (socketCliente, (**void** \*)&tecla, 1, 0);

**if** (sendi < 0)

{

printf ("Error al enviar\n");

exit (0);

}

**if** (tecla == '#')

**break**;

}**else**

aux = tecla;

}

}

//Ya que se obtiene el sonido, cerramos el manejador y el descriptor del socket

snd\_pcm\_close (manejador);

close (socketCliente);

free (datos);

**return** 0;

}

**Servidor.py**

**from** socket **import** \*

**from** thread **import** \*

**from** graphics **import** \*

**import** time

**import** Tkinter

**import** sys

**def** MensajeRecibido(cadena,numJugador):

**global** i

YaExiste = 0

**for** x **in** range(0,len(lista\_casillas)):

**if** lista\_casillas[x] == cadena:

YaExiste = 1

**if** cadena == '1' **and** YaExiste == 0:

**print** "Tecleaste el numero 1"

ptCasilla1 = Point(150,150)

**if** numJugador == 1:

msgCasilla1 = Text(ptCasilla1,"X")

**else**:

msgCasilla1 = Text(ptCasilla1,"O")

msgCasilla1.setSize(12)

msgCasilla1.setTextColor('white')

msgCasilla1.draw(wind)

lista\_casillas.append(cadena)

i+=1

**elif** cadena == '2' **and** YaExiste == 0:

**print** "Tecleaste el numero 2"

ptCasilla2 = Point(250,150)

**if** numJugador == 1:

msgCasilla2 = Text(ptCasilla2,"X")

**else**:

msgCasilla2 = Text(ptCasilla2,"O")

msgCasilla2.setSize(12)

msgCasilla2.setTextColor('white')

msgCasilla2.draw(wind)

lista\_casillas.append(cadena)

i+=1

**elif** cadena == '3' **and** YaExiste == 0:

**print** "Tecleaste el numero 3"

ptCasilla3 = Point(350,150)

**if** numJugador == 1:

msgCasilla3 = Text(ptCasilla3,"X")

**else**:

msgCasilla3 = Text(ptCasilla3,"O")

msgCasilla3.setSize(12)

msgCasilla3.setTextColor('white')

msgCasilla3.draw(wind)

lista\_casillas.append(cadena)

i+=1

**elif** cadena == '4' **and** YaExiste == 0:

**print** "Tecleaste el numero 4"

ptCasilla4 = Point(150,250)

**if** numJugador == 1:

msgCasilla4 = Text(ptCasilla4,"X")

**else**:

msgCasilla4 = Text(ptCasilla4,"O")

msgCasilla4.setSize(12)

msgCasilla4.setTextColor('white')

msgCasilla4.draw(wind)

lista\_casillas.append(cadena)

i+=1

**elif** cadena == '5' **and** YaExiste == 0:

**print** "Tecleaste el numero 5"

ptCasilla5 = Point(250,250)

**if** numJugador == 1:

msgCasilla5 = Text(ptCasilla5,"X")

**else**:

msgCasilla5 = Text(ptCasilla5,"O")

msgCasilla5.setSize(12)

msgCasilla5.setTextColor('white')

msgCasilla5.draw(wind)

lista\_casillas.append(cadena)

i+=1

**elif** cadena == '6' **and** YaExiste == 0:

**print** "Tecleaste el numero 6"

ptCasilla6 = Point(350,250)

**if** numJugador == 1:

msgCasilla6 = Text(ptCasilla6,"X")

**else**:

msgCasilla6 = Text(ptCasilla6,"O")

msgCasilla6.setSize(12)

msgCasilla6.setTextColor('white')

msgCasilla6.draw(wind)

lista\_casillas.append(cadena)

i+=1

**elif** cadena == '7' **and** YaExiste == 0:

**print** "Tecleaste el numero 7"

ptCasilla7 = Point(150,350)

**if** numJugador == 1:

msgCasilla7 = Text(ptCasilla7,"X")

**else**:

msgCasilla7 = Text(ptCasilla7,"O")

msgCasilla7.setSize(12)

msgCasilla7.setTextColor('white')

msgCasilla7.draw(wind)

lista\_casillas.append(cadena)

i+=1

**elif** cadena == '8' **and** YaExiste == 0:

**print** "Tecleaste el numero 8"

ptCasilla8 = Point(250,350)

**if** numJugador == 1:

msgCasilla8 = Text(ptCasilla8,"X")

**else**:

msgCasilla8 = Text(ptCasilla8,"O")

msgCasilla8.setSize(12)

msgCasilla8.setTextColor('white')

msgCasilla8.draw(wind)

lista\_casillas.append(cadena)

i+=1

**elif** cadena == '9' **and** YaExiste == 0:

**print** "Tecleaste el numero 9"

ptCasilla9 = Point(350,350)

**if** numJugador == 1:

msgCasilla9 = Text(ptCasilla9,"X")

**else**:

msgCasilla9 = Text(ptCasilla9,"O")

msgCasilla9.setSize(12)

msgCasilla9.setTextColor('white')

msgCasilla9.draw(wind)

lista\_casillas.append(cadena)

i+=1

**else**:

**if** YaExiste == 1:

**print** "Casilla ya seleccionada, No hagas trampa"

**else**:

**print** "Mensaje no aceptado"

**def** ligarSocket(servidor, host, port):

**while** True:

**try**:

servidor.bind((host, port))

**break**

**except** error **as** e:

**print**("ERROR:", e)

**def** VaciarCasillas():

msgNumero1.undraw()

msgNumero2.undraw()

msgNumero3.undraw()

msgNumero4.undraw()

msgNumero5.undraw()

msgNumero6.undraw()

msgNumero7.undraw()

msgNumero8.undraw()

msgNumero9.undraw()

msgSubIndice1.undraw()

msgSubIndice2.undraw()

lista\_casillas = []

casillas\_jugador1 = []

casillas\_jugador2 = []

i=0

a=0

wind=GraphWin("Gato",500,500)

wind.setBackground("Black")

ptTitle=Point(250,25)

msg=Text(ptTitle,"Juego Gato")

msg.setSize(15)

msg.setTextColor('white')

msg.draw(wind)

ptSubtitulo = Point(250,70)

msgSub=Text(ptSubtitulo,"Cada casilla es un numero en el panel telefonico")

msgSub.setSize(12)

msgSub.setTextColor('white')

msgSub.draw(wind)

#Linea horizontal 1

LineaHo1 = Line(Point(100,200),Point(400,200))

LineaHo1.setOutline('white')

LineaHo1.draw(wind)

#Linea horizontal 2

LineaHo2 = Line(Point(100,300),Point(400,300))

LineaHo2.setOutline('white')

LineaHo2.draw(wind)

#Linea vertical 1

LineaVer1 = Line(Point(200,100),Point(200,400))

LineaVer1.setOutline('white')

LineaVer1.draw(wind)

#Linea vertical 2

LineaVer2 = Line(Point(300,100),Point(300,400))

LineaVer2.setOutline('white')

LineaVer2.draw(wind)

#Numero 1

ptnumero1 = Point(150,150)

msgNumero1 = Text(ptnumero1,"1")

msgNumero1.setSize(12)

msgNumero1.setTextColor('white')

msgNumero1.draw(wind)

#Numero 2

ptnumero2 = Point(250,150)

msgNumero2 = Text(ptnumero2,"2")

msgNumero2.setSize(12)

msgNumero2.setTextColor('white')

msgNumero2.draw(wind)

#Numero 3

ptnumero3 = Point(350,150)

msgNumero3 = Text(ptnumero3,"3")

msgNumero3.setSize(12)

msgNumero3.setTextColor('white')

msgNumero3.draw(wind)

#Numero 4

ptnumero4 = Point(150,250)

msgNumero4 = Text(ptnumero4,"4")

msgNumero4.setSize(12)

msgNumero4.setTextColor('white')

msgNumero4.draw(wind)

#Numero 5

ptnumero5 = Point(250,250)

msgNumero5 = Text(ptnumero5,"5")

msgNumero5.setSize(12)

msgNumero5.setTextColor('white')

msgNumero5.draw(wind)

#Numero 6

ptnumero6 = Point(350,250)

msgNumero6 = Text(ptnumero6,"6")

msgNumero6.setSize(12)

msgNumero6.setTextColor('white')

msgNumero6.draw(wind)

#Numero 7

ptnumero7 = Point(150,350)

msgNumero7 = Text(ptnumero7,"7")

msgNumero7.setSize(12)

msgNumero7.setTextColor('white')

msgNumero7.draw(wind)

#Numero 8

ptnumero8 = Point(250,350)

msgNumero8 = Text(ptnumero8,"8")

msgNumero8.setSize(12)

msgNumero8.setTextColor('white')

msgNumero8.draw(wind)

#Numero 9

ptnumero9 = Point(350,350)

msgNumero9 = Text(ptnumero9,"9")

msgNumero9.setSize(12)

msgNumero9.setTextColor('white')

msgNumero9.draw(wind)

#Sub Indice 1

ptSubIndice1 = Point(160,450)

msgSubIndice1=Text(ptSubIndice1,"0\nIniciar")

msgSubIndice1.setSize(12)

msgSubIndice1.setTextColor('white')

msgSubIndice1.draw(wind)

#Sub Indice 2

ptSubIndice2 = Point(350,450)

msgSubIndice2=Text(ptSubIndice2,"#\nFinalizar")

msgSubIndice2.setSize(12)

msgSubIndice2.setTextColor('white')

msgSubIndice2.draw(wind)

os.system("clear")

host = '127.0.0.1'

port = 3905

servidor = socket(AF\_INET, SOCK\_STREAM)

ligarSocket(servidor, host,port)

servidor.listen(1) # Espero la conexion con c

**print**("\nEsperando Jugadores")

conn,addr = servidor.accept()

**print** "Conexion Establecida"

**while** True:

**global** i

reply = conn.recv(100)

**if** len(lista\_casillas) == 9:

**print** "Juego terminado"

**break**

**if** a == 0:

**if** reply == '#':

**print** "Se ha salido del juego"

**break**

**elif** reply == '0':

VaciarCasillas()

a+=1

**else**:

**print** "Opcion no reconocida"

**else**:

**if** i%2==0:

**print** "Jugador 1:"

Jugador = 1

**if** reply == '#':

**print** "Jugador 1 ha abandonado el juego la partida ha terminado"

**break**

**else**:

MensajeRecibido(reply,Jugador)

**elif** i%2!=0:

**print** "Jugador 2:"

Jugador = 2

**if** reply == '#':

**print** "Jugador 2 ha abandonado el juego la partida ha terminado"

**break**

**else**:

MensajeRecibido(reply,Jugador)