Proyecto1: Marcación DTMF

Pamela Salazar Espinoza
Procesamiento Digital de Señales

***Abstract*— This paper describes DTMF encoding, and decoding implementation, and their application in a server-client system. The server uses the encoding to create tones with a combination of frequencies corresponding to a particular symbol. For other side, the server is able to use the decoder to listen to the tone and determine which symbol corresponds. The server implements a menu in which it gives a series of instructions to the user and waits for his response, which must be a certain tone. The user from the client types the key corresponding to the answer, and the client generates the respective tone. The server listens to it, decodes it, and generates a response based on the get symbol.**

Keywords—DTMF

I. Introducción

El algoritmo de Goertzel es una técnica para el procesamiento digital de señales (DSP) el cual identifica las componentes de frecuencia de una señal. Este fue publicado por el Dr. Gerald Goertzel1 en 1958. Goertzel es un filtro digital derivado de la transformada discreta de Fourier (DFT) que puede detectar componentes de frecuencia específicas en una señal, como por ejemplo para permitir que los circuitos de conmutación telefónica digital con tecnología DSP puedan identificar los tonos característicos generados cuando un número se marca en el sistema. [1]

II. Arquitectura de sfotware

En la figura 1 se observa el diagrama a alto nivel del sistema implementado. El servidor le comunica al usuario las instrucciones a seguir reproduciendo una serie de audios pregrabados, según corresponda, para ello hace uso de la librería playsound de Python, estas instrucciones son:

1. Espera: Primeramente, el servidor le indica al usuario que debe marcar algún tono para comenzar. De no detectar ningún tono el servidor imprime un mensaje de volver a marcar tono. Una vez detectado algún ejecuta la siguiente instrucción.
2. Introducción: Este la funcionalidad solo reproduce un menaje de bienvenida al usuario.
3. Marcar contraseña: Esta función solicita la contraseña. Esta función es recursiva y se vuelve a llamar a si misma en caso de contraseña incorrecta. De lo contrario termina y permite continuar con la ejecución del programa. Para efectos de debugging se imprime en pantalla la contraseña escuchada por el servidor (no necesariamente la correcta).
4. Opciones: Esa función reproduce un mensaje indicado marcar 1 para leer mensaje o 2 para cambiar la contraseña. En caso de que se detecte un tono que no corresponda a ninguno de estos números se reproduce un mensaje de digito correcto y la función de vuelve a llamar a sí misma.
5. Leer mensaje: Esta función lo único que hace es reproducir un mensaje que indica que no hay ningún mensaje nuevo.
6. Cambiar contraseña: Es recibe cuatro dígitos correspondientes a la nueva contraseña. Para confirmar la nueva contraseña el servidor reproduce cuatro audios pregrabados, uno para cada símbolo. Es importante aclarar que no esta implementado una base de datos donde se guarde la contraseña por lo que la próxima que se corra el servidor la contraseña volverá a ser la que esta por defecto. Esto en una aplicación real por supuesto que no debería ser así, pero para efectos de este trabajo se sale de los objetivos.

Como se mencionó anteriormente la comunicación entre el cliente y el servidor es por audio. Esto mediante un codificador y decodificador DTMF, la implementación de estos se explica con detalle en las siguientes secciones.

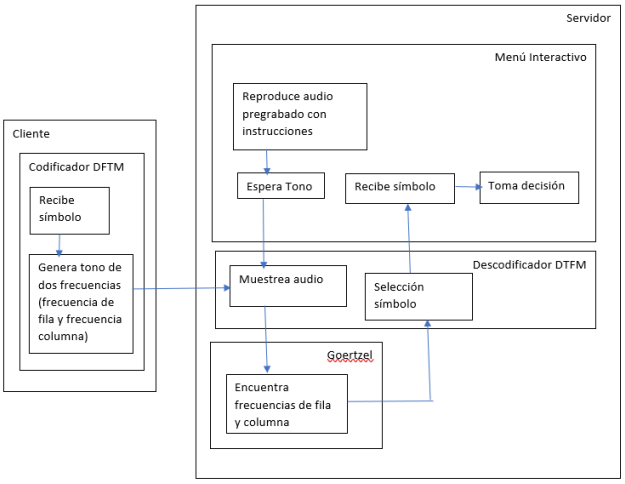

Figura 1. Arquitectura de Cliente-Servidor.

III. CODIFCADOR DTMF

Los tonos DMFT son la suma de dos funciones senoidales de diferente frecuencia. En la figura 2 se muestra la combinación de frecuencias sobre un teclado digital. La ecuación 1.

$$x(t) = \text{sen}(2\pi f_{columna}) + \text{sen}\left(2\pi f_{fila}\right) \quad (1)$$

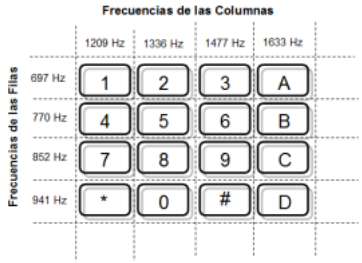

Figura 2. Matriz de teclado telefónico.

Para reproducir la señal generada el codificador la guarda en un archivo .wav con una frecuencia de muestreo de 8000 Hz y un total, de 24000 muestras. Este archivo es reproducido por Python. Cada vez que se marca un numero nuevo este se sobre escribe con la señal correcta y se reproduce. La cantidad de muestras de puede variar según se desee variar la duración del tono. Con las pruebas realizadas los valores anteriormente dichos funcionaron bien, sin embargo, esto aún se puede mejorar.

IV. DESCODIFCADOR DTMF

Como ya se ha mencionado el descodificador DTMF utiliza el algoritmo de Goertzel, sin embargo, de este se explicará en la siguiente sección. Esta en cambio se dedicará al muestreo de audio y las pruebas hechas a este bloque. Sin embargo, vale señalar de antemano que el testeo del algoritmo Goertzel no presentaron ningún error por lo que lo errores detectados corresponde precisamente al muestreo de audio.

Para detectar audio se usa librería SpeechRecognition de Python. Esta escucha el audio atreves del micrófono y lo guarda en un archivo wav con una frecuencia de muestreo de 44100 Hz, la cual es suficiente puesto a que es mas del doble que la frecuencia mayor a detectar de 1633Hz.

La librería SpeechRecognition espera un tiempo para detectar el sonido y un tiempo después de que dejar de escuchar para detener la grabación. Estos tiempos son ajustable, sin embargo, para este proyecto se usaron los valores por defecto de la misma. Una mejora importante seria realizar mas experimentos para determinar tiempos de espera óptimos. Si pasado el tiempo correspondiente el micrófono no detecta sonido alguno le indica al usuario mediante pantalla que vuelva a marcar el digito. Si detecta sonido pero las frecuencias detectadas no corresponde a ningún símbolo conocido igualmente le pide que vuelva a marca el digito. Otra mejora para considerar es que estos mensajes deberían ser dados por voz también como los correspondientes al menú principal. Para evitar errores el usuario no debería marca ningún tono hasta que se imprima un mensaje de “Escuchando”, este mensaje también debería ser implementado por voz por ejemplo con el típico mensaje de la operadora de “Marque después del tono”.

Para validar la funcionalidad de este bloque se hicieron cuatro pruebas, en las cuatro se probo generar un tono con el codificador, detectarlo con el decodificador e imprimir el símbolo detectado; sin involucrar el menú principal para esta parte. En cada prueba se probó todos los símbolos, en una se uso el micrófono del headset en un ambiente silencioso, en otra el micrófono del headset en un ambiente ruidoso (con música en el exterior), otra sin usar los headset con un ambiente ruidoso.

Las pruebas usando el micrófono del headset y la prueba usando el micrófono de la computadora en absoluto silencio no tuvieron ni un solo error. Sin embargo, la prueba en un ambiente ruidoso con el micrófono de la computadora se equivocó dos veces, marcando 4 detecto un 1 y en otra ocasión detecto un símbolo antes de marcar cualquier tono. Esta prueba se repitió y en esta segunda ocasión se equivoco marcando 7 y detectado 9.

Como se pudo ver la influencia del ruido externo puede causar un comportamiento erróneo aleatorio. Por lo que es importante tener cuidado con el canal físico de comunicación y aplicar algún filtro para reducir el ruido. Como en el caso de las headsets que ya tienen uno, evitando problemas aun en un ambiente ruidoso.

V. ALGORITMO GOERTZEL

El algoritmo Goertzel implementado está basado en su totalidad en el trabajo de [1]. En este trabajo los autores implementaron dicho algoritmo en Matlab, basado en ese código se hizo uno similar en Python. En esta sección se describe la matemática detrás del algoritmo lo cual se tomo del mismo trabajo ya mencionado. En el código se pueste encuentra un código hecho en Python para el testeo de esta implementación donde se crearon señales como las que crea el codificador, pero sin convertirlas en audio para probar el algoritmo y en todos los casos las frecuencias detectadas fueron las correctas.

El algoritmo de Groetzel para una señal x con N muestras y una frecuencia de muestreo Fs. Calcula un y(N) para una frecuencia posible determinada. Después de calcular y(N) para cada una de las frecuencias posibles se determinar que las dos frecuencias con mayor y(N) son las frecuencias correspondientes al tono. En las pruebas hechas se observó que el y(N) de las frecuencias correctas era mucho mayor que el de las incorrectas. El algoritmo implementa las siguientes

ecuaciones para frecuencia posible, decir para las frecuencias de la figura 2:

$$k = N\frac{F_{posible}}{F_s} \quad (2)$$

$$s_k(n) = x(n) + 2\cos\left(\frac{2\pi k}{N}\right)s_k(n-1) - s_k(n-2) \quad (3)$$

$$W_n^k = e^{\left(\frac{-2\pi k}{N}\right)j}$$

$$y(n) = s_k(n) - W_n^k s_k(n-1)$$

VI. Conclusiones

El algoritmo de Goertzel para descodificación DTMF es sumamente efectivo. Acertando la mayoría de las veces en ambientes poco óptimo para la comunicación como la presencia de ruido excesivo. Tomando las medidas correspondientes es perfectamente optimo. Adicionalmente en las pruebas hechas se noto que le tiempo de computo del algoritmo como tal es despreciable, puesto a que es sumamente rápido. Siendo otros factores como los tiempos de espera de respuesta del usuario los que hacen el sistema un algo lento.

Cabe aclarar que por error tanto en el codificador como en el descodificador se una "." en lugar de la "D" . Esto es un error de programación que fue detectado tarde para su corrección antes de la fecha de entrega. Pero que no afecta las conclusiones del trabajo.

References

[1] *J. A. Cortés Osorio, J. A. Mendoza Vargas, y J. A. Muriel Escobar, «Alternativa al análisis en frecuencia de la FFT mediante el algoritmo Goertzel», Sci. tech, vol. 1, n.º 44, pp. 217–222, abr. 2010.*