Proyecto 1: Marcador y Servidor Telefónico mediante DTMF

Daniel A. Rojas, *Estudiante, ITCR*
Christopher R. Russell, *Member*

***Abstract*—This document showcases a practical implementation of real-time signal processing for a DTMF voicemail server. The basic cocepts of DTMF are explained as well as the architecture of the system, incluiding challenges related to real-time audio capture drivers and controllers.**

***Index Terms*—DTMF, Goertzel Algorithm, DSP, Real-Time Audio, JACK.**

I. Introducción

El sistema de marcación por tonos o DTMF fue intrducido por AT&T como un método de señalamiento de botones mediante el canal de voz de la linea telefónica [1] y es adoptado como estándar por la recomendación Q.23 de la UIT-T. El mecanismo corresponde a asignar una par de frecuencias a cada tecla de la matriz. Las columnas se identifican con cuatro frecuencias superiores y las filas con cuatro frecuencias inferiores (Fig. 1).

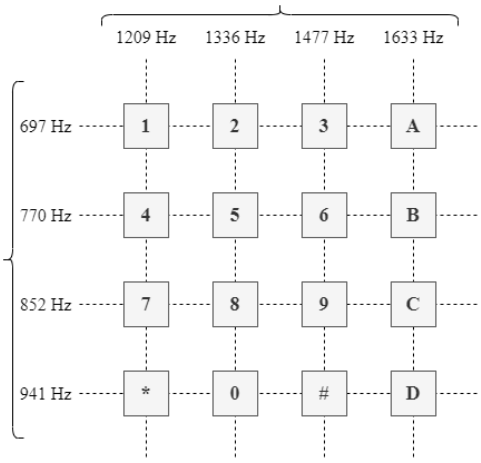

Fig. 1. Distribución de frecuencias por tecla.

Con la transición a sistemas digitales se necesitaban detectores DTMF digitales; uno de los métodos detección basado en técnicas de procesamiento digital y el empleado en el detector a desarrollar corresponde al algoritmo de Goertzel. En el presente proyecto, se explora las capacidades e implementación del algoritmo de Goertzel para el área de procesamiento digital de señales.

II. Diseño del Sistema

El sistema propuesto corresponde a un modelo de cliente-servidor: Un marcador de tonos que interactúa con un buzón de correos de voz. La comunicación entre ambos se realiza exclusivamente mediante señales de audio, es decir que emplean micrófonos y parlantes como medio físico.

A. Marcador

El marcador contempla un *script* básico de terminal en Python; un lazo infinito que captura las teclas como caracteres (`0`, `1`, `2...9`, `#`, `*`, `A...D`) e ignora los caracteres no contemplados. Luego de capturar la tecla, se muestra en pantalla el par de frecuencias correspondiente según un diccionario interno y se reproduce el tono (Fig. 2). Al llamar al *script* se debe adjuntar obligatoriamente un parámetro de duración del tono en milisegundos y opcionalmente se puede aventanar la señal mediante una ventana de Hann o una ventana triangular.

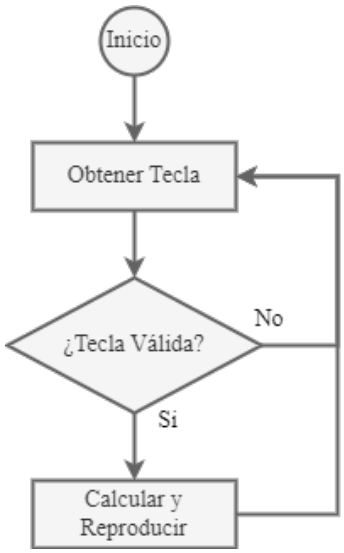

Fig. 2. Diagrama de flujo del marcador.

El *script* reproduce señales en arreglos de NumPy, y dichos arreglos se calculan mediante

$$y(n) = \frac{\sin(2\pi f_H n/F_s) + \sin(2n\pi f_L n/F_s)}{2} w(n) \quad (1)$$

donde $w(n)$ es la función de ventana, n corresponde al número de muestra, F_s es la frecuencia de muestreo y f_H junto con f_L son el par de frecuencias de la tecla. El número de muestras se calcula como $F_s D$ donde D es la duración del tono. como se esta simulando un canal de voz telefónico, se opta para el programa una frecuencia de muestreo fija de 8 kHz.

En un generador de señales ideal, las únicas componentes de frecuencia detectables corresponderían a ambos tonos DTMF, sin embargo al ser un sistema digital, es decir discreto, existirán nuevas componentes relacionadas con la frecuencia de muestreo de la señal discreta. por ello es esperable que el sistema experimente distorsión total harmónica (THD) y la forma de minimizar su impacto es contar con una frecuencia de muestreo suficientemente alta, tal que no pueda ser detectable en el servidor.

B. Servidor

El servidor se implementa siguiendo la arquitectura que se muestra en Fig 3:

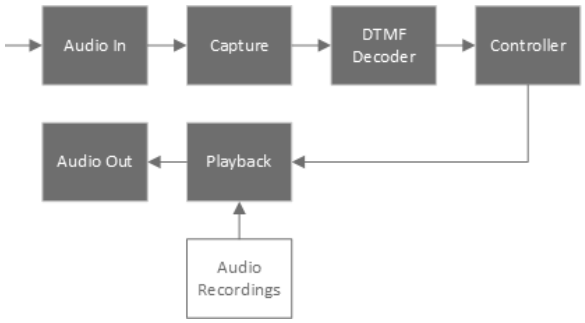

Fig. 3. Diagrama de arquitectura del servidor.

Los bloques de entrada de audio, captura y salida de audio se implementan dentro del código proporcionado para la Tarea 04 del curso, que se creo utilizando las interfaces ALSA y JACK en Linux. El bloque de captura proporciona fragmentos de audio de 1024 muestras en formato de punto flotante al decodificador DTMF para su procesamiento. El decodificador DTMF es una implementación del algoritmo Goertzel para frecuencias de 697, 770, 852, 941, 1209, 1336 y 1477 Hz. La implementación utilizada esta adaptada de la proporcionada por Kuo, Lee y Tian en los materiales complementarios de "Real-Time Digital Signal Processing: Fundamentals, Implementations and Applications, 3rd Edition" [2]. Se uso el esquema general de su implementación de punto fijo, pero se reescribió como punto flotante. La discriminación de los tonos DTMF se basa en un mecanismo de umbral simple. Los tonos DTMF detectados se envían mediante una cola de mensajes POSIX al bloque del controlador que se ejecuta en un subproceso separado. El bloque controlador implementa la maquina de estado de la interfaz de usuario en Fig 4.

La reproducción de audio se implementa con un descriptor de archivo en la memoria compartida entre el hilo del controlador y el subproceso de procesamiento original del código de JACK. para reproducir una grabación de audio, el subproceso del controlador abre el archivo asociado y coloca su descriptor de archivo en la memoria compartida. Cuando este descriptor es valido, el bucle del proceso de recepción/reproducción leerá el siguiente bloque disponible del archivo y lo escribirá en la interfaz de audio. Al llegar al final del archivo, el ciclo del proceso cerrara el archivo e invalidara su descriptor. Si no hay un descriptor valido, el reproductor generara ceros.

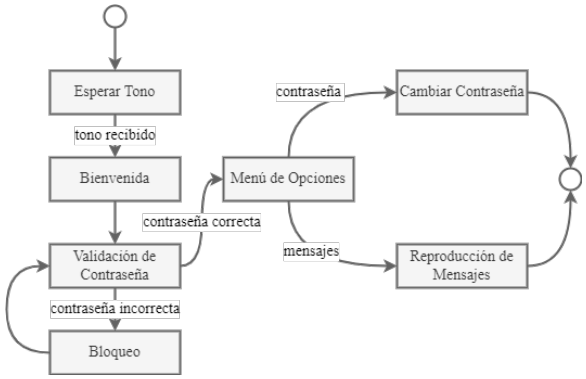

Fig. 4. Diagrama de estados del marcador.

III. Resultados

A. Desempeño del servidor

Durante las varias pruebas realizadas se obtuvo un desempeño mixto por parte del servidor, especialmente con tonos menores a 250 ms de duración. para las señales mas bajas y mas altas del rango era común identificar falsos positivos tal como `1` identificado como `7` o `4` como `*`. Sin embargo los resultados mejoran su fidelidad con tonos mas extensos.

B. Retos y Dificultades

Un inconveniente mayor que se presentó corresponde al componente JACK. Es una herramienta profesional de alta utilidad, pero su integración en tiempo real en el proyecto así como haberse ejecutado principalmente en maquinas virtuales dificultó considerablemente tener ejecuciones exitosas, pues experimenta frecuentes *xruns* que detienen esporádicamente la ejecución del controlador por tiempos indefinidos, desde un par de segundos hasta docenas de minutos.

La decodificación de señales DTMF es algo sensible al ruido extraño, lo que provoca una detección falsa ocasional o detección múltiple de tonos continuos. Se podría usar un procesamiento posterior adicional para filtrarlos. Además, el mecanismo de umbral es sensible a los cambios de volumen causados por el control automático de ganancia en el controlador de audio, y se debe desarrollar un mecanismo adaptativo mas robusto.

Fuera de ello, la maquina de estados opera bien y la captura de las teclas es generalmente correcta, es posible que los errores que se presenten sea por el mismo proceso en tiempo real de JACK.

IV. Conclusiones

El esquema de marcación DTMF pareciese irreemplazable por su simplicidad y fácil implementación en los dispositivos para clientes. El peso verdadero del esquema recae en los servidores y los métodos de procesamiento de señales que adopten. Existen alternativas al algoritmo de Goertzel que bajo el mismo principio logran discriminar coeficientes de forma más precisa, sin embargo es enteramente posible optimizar un sistema digital para que opere sin problemas con tan solo una forma básica del algoritmo el cual ya es rápido y portátil.

Un inconveniente mayor en sistemas en tiempo real es la recuperación ante fallas. En este proyecto las fallas esporádicas de JACK representaron un problema sin resolver inherente a la plataforma que debe ser resuelto con software más allá de los objetivos de este curso.

REFERENCIAS

[1] S. Selman, R. Paramesran, *Comparative analysis of methods used in the design of DTMF tone detectors*,2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications, pp. 335-339, 2007.

[2] S. Kuo, B. Lee, W. Tian, *Real-Time Digital Signal Processing: Fundamentals, Implementations and Applications*, 3rd Ed. Wiley, 2013.