

UNIVERSIDAD NACIONAL DE TRUJILLO

Facultad de Ciencias Físicas y Matemáticas

Escuela Profesional de Informática



**Sistema de seguridad para el control de acceso a una vivienda
mediante el reconocimiento de voz utilizando coeficientes
cepstrum MFCC Y DTW**

TESIS

**PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO
INFORMÁTICO**

AUTORES: Ocas Quiroz Carlos E.

Sánchez Pedro Edwin J.

ASESOR: Ms. Peralta Luján José L.

TRUJILLO - PERÚ

2019

Dedicamos esta tesis a :

Nuestras familias por el apoyo incondicional y permanente que nos brindaron durante nuestra preparación académica para poder llegar a ser profesionales competitivos y de éxito, quienes con su amor y paciencia permitieron cumplir un deseo tan esperado, y que ahora permite ser útil a nuestro país.

Agradecimientos

A Dios y a nuestros padres, ya que sin ellos nada de esto hubiera sido posible.

A nuestro asesor de tesis, Ms. José Peralta Luján, que, con su amistad, apoyo, paciencia, dedicación, consejos y observaciones en la realización de este trabajo pudimos sacar adelante esta tesis y además por darnos las herramientas necesarias para ser mejores profesionales cada día.

Al Ms. Clayder Gonzales Cadenillas, que desde un principio se mostró siempre disponible e interesado, brindándonos su apoyo y motivándonos en esta área de investigación.

A nuestros amigos, Ing. Javier Fuentes Huertas e Ing. Alexis Aspauza Lezcano, por el apoyo y consejos dados como extesistas en la carrera de Informática.

Al Dr. Jorge Guevara Días, por el apoyo y sus excelentes comentarios brindados en mejora de la investigación.

Al honorable jurado, por sus valiosas aportaciones.

A la Universidad Nacional de Trujillo, por la formación que en ella recibimos y a nuestros profesores por habernos brindado parte de su sabiduría y amistad.

Resumen

El presente trabajo de investigación busca desarrollar un sistema de seguridad para el control de acceso a una vivienda mediante el reconocimiento de voz del locutor dependiente del texto, donde el principal objetivo es identificar a cada miembro del hogar, quienes tendrán el acceso.

El sistema está conformado por 2 fases, la primera es la de extracción de características de la señal, para esto se usó una adaptación de los MFCC y la segunda es el reconocimiento automático del locutor que esta basada en la técnica DTW, además de una lógica para la toma de decisión en la identificación o no, o falsa identificación de un usuario. En relación a la configuración de los experimentos, para comprobar que el sistema sea confiable se realizaron 2 tipos de pruebas, una entre los usuarios del sistema tratando de acceder uno con el usuario del otro y otra donde personas ajenas a las almacenadas en la base de datos (intrusos) trataron de acceder.

Como resultado final, se obtuvo un sistema de seguridad con buenos resultados en el reconocimiento de voz de los miembros del hogar. De esta manera, se pretende contribuir en la industria de sistemas biométricos, en el campo de los sistemas por reconocimiento de voz, así como también su integración en sistemas domóticos.

Palabras claves: Sistema de seguridad, control de acceso, reconocimiento de voz, reconocimiento del locutor, coeficientes cepstrales, alineamiento temporal dinámico.

Abstract

The present research work seeks to develop a security system for the control of access to a dwelling by voice recognition of the speaker dependent on the text, where the main objective is to identify each member of the household, who will have access.

The system consists of 2 phases, the first is the extraction of signal characteristics, for this an adaptation of the MFCC was used and the second is the automatic recognition of the speaker that is based on the DTW technique, in addition to a logic for decision making on the identification or not, or false identification of a user. In relation to the configuration of the experiments, to verify that the system is reliable, 2 types of tests were carried out, one among the users of the system trying to access one with the user of the other and the other where people outside those stored in the database (intruders) tried to access.

As a final result, a security system was obtained with good results in the voice recognition of household members. In this way, it is intended to contribute in the biometric systems industry, in the field of voice recognition systems, as well as being able to be part of home automation systems.

Keywords: Security system, access control, voice recognition, speaker recognition, cepstral coefficients, dynamic time warping.

Lista de símbolos

Constantes:

- (1) f_s Frecuencia de muestreo.
- (2) f_{max} Frecuencia máxima.
- (3) f_{min} Frecuencia mínima.
- (4) f_N Frecuencia de Nyquist.
- (5) T_s Periodo de muestreo.
- (6) μ Tamaño de paso de adaptación del filtro adaptivo.
- (7) M Número de orden del filtro adaptivo.
- (8) β Constante de adaptación del filtro adaptivo.
- (9) c Constante de normalización del filtro adaptivo.
- (10) α Valor del filtro preénfasis.
- (11) V Longitud de ventana en la segmentación.
- (12) S Valor de desplazamiento o solapamiento en la segmentación.

- (13) T Longitud de trama para la detección de inicio y fin de una palabra.
- (14) BC Número de bandas críticas del MFCC.
- (15) k_l Índice correspondiente a la frecuencia inferior.
- (16) k_h Índice correspondiente a la frecuencia superior.
- (17) P Slope constraint del DTW.
- (18) r Tamaño de ventana de ajuste del DTW.

Variables:

- (1) $d(n)$ Señal de voz contaminada.
- (2) $s(n)$ Señal de voz.
- (3) $v_0(n)$ Señal de ruido en el micrófono primario.
- (4) $v_1(n)$ Señal de ruido en el micrófono secundario.
- (5) $w_k(n)$ Vector de pesos del filtro adaptivo.

- (6) $e(n)$ Señal de error o respuesta deseada.
- (7) E_n Energía promedio por segmento.
- (8) E_{avg} Energía promedio de toda la señal.
- (9) M_n Magnitud promedio.
- (10) Z_n Cruce por ceros.
- (11) M_{sn} Magnitud del ruido.
- (12) Z_{sn} Cruce por ceros del ruido.
- (13) μ_{Ms} Media de la magnitud del ruido.
- (14) μ_{Zs} Media del cruce por ceros del ruido.
- (15) σ_{Ms} Desviación estándar de la magnitud del ruido.
- (16) σ_{Zs} Desviación estándar del cruce por ceros del ruido.
- (17) $IZCT$ Umbral de cruce por ceros.
- (18) ITL Umbral de energía bajo.
- (19) ITU Umbral de energía alto.
- (20) IMX Energía máxima de la voz.

- (21) IMN Energía media del ruido de fondo.
- (22) $x_p(n)$ Señal de voz con preénfasis.
- (23) $x_l(n)$ Señal de voz segmentada.
- (24) $x_w(n)$ Señal de voz ponderada.
- (25) $w(n)$ Pesos de la ventana de Hamming.
- (26) N_{fft} Número de puntos de la transformada de Fourier.
- (27) $X(k)$ Espectro en frecuencia de la transformada rápida de Fourier.
- (28) $P(k)$ Espectro en potencia de la transformada rápida de Fourier.
- (29) Δf Resolución en frecuencia.
- (30) SF Separación en frecuencia.
- (31) N_p Número de puntos del espectro de potencia.
- (32) Δk Resolución por muestra del espectro.
- (33) P_{sc} Compresión del espectro en potencia.
- (34) C_m Transformada coseno discreta.
- (35) V_{mfcc} Coeficientes MFCC.

- (36) ΔC_k Coeficientes MFCC delta.
- (37) $\Delta\Delta C_k$ Coeficientes MFCC doble delta.
- (38) D_T Distancia total entre dos vectores.
- (39) D_A Distancia acumulada.
- (40) d Medida de distorsión.
- (41) z_n Patrón de referencia por la función de alineamiento.
- (42) U Umbral de decisión.
- (43) $P(D)$ Distribución de probabilidad.
- (44) μ^{intra} Media de las distancias intralocutor.
- (45) μ^{inter} Media de las distancias interlocutor.
- (46) σ^{intra} Desviación estándar de las distancias intralocutor.
- (47) σ^{inter} Desviación estándar de las distancias interlocutor.

Índice de figuras

2.1.	Órganos del cuerpo que intervienen en el aparato fonador.	15
2.2.	Corte esquemático de la laringe en un plano horizontal.	16
2.3.	Modelo fuente-filtro para la producción de la voz.	27
2.4.	Diagrama de bloques del proceso general del reconocimiento de voz.	28
2.5.	Diagrama de bloques para la etapa de entrenamiento.	29
2.6.	Diagrama de bloques para la etapa de reconocimiento.	29
2.7.	Diagrama de bloques para el módulo de preprocesamiento.	32
2.8.	Diagrama de la adquisición de voz.	33
2.9.	Diagrama de la digitalización de la señal de voz.	33
2.10.	Muestreo de una señal de voz.	37
2.11.	Señal cuantificada con 2 y 4 bits respectivamente.	38
2.12.	Señal cuantificada uniformemente.	39
2.13.	Señal cuantificada no uniformemente.	40
2.14.	Ilustración de la estructura de un filtro Wiener.	43

2.15. Cancelador de ruido adaptativo.	44
2.16. Filtro transversal.	47
2.17. Modelo de un filtro adaptativo.	51
2.18. Convergencia y error mínimo.	57
2.19. Cuadrado promedio para diferentes valores de M y μ para el LMS.	58
2.20. Valor para el número de orden M con $\mu=0.02$	58
2.21. Curva de eficiencia vs. sonido del equipo.	59
2.22. Diagrama de eliminación de ruido y normalización.	59
2.23. Diagrama de flujo del algoritmo Rabiner y Sambur.	69
2.24. Diagrama de energía.	71
2.25. Diagrama de flujo para la búsqueda del punto de inicio basado en la energía.	72
2.26. Diagrama de flujo para la búsqueda del punto de fin basado en la energía.	72
2.27. Puntos de inicio y fin de la señal de voz.	73
2.28. Diagrama del filtro de preénfasis.	74
2.29. Magnitud de la respuesta en frecuencia del filtro de preénfasis.	75
2.30. Diagrama de segmentación, ventaneo y recorte de la señal de voz.	76
2.31. División en tramas con traslape de una señal de voz.	77
2.32. Extracción de una trama mediante una ventana rectangular.	79
2.33. Ventana de Hamming.	80

2.34. Comparación de la variación en el reconocimiento entre LPC, MFCC Y PLP.	85
2.35. Proceso de extracción de características con MFCC.	85
2.36. Procedimiento para la obtención de los coeficientes MFCC.	87
2.37. Estructura general de los bancos de filtros perceptuales.	92
2.38. Gráfica del arqueado de la frecuencia con la escala Mel.	94
2.39. Curva para las bandas críticas triangulares.	95
2.40. Banco de filtros o ventanas triangulares equiespaciados en la escala Mel.	99
2.41. Banco de filtros o ventanas triangulares en Hertz con distribución logarítmica.	99
2.42. Ilustración general de alineamiento en el tiempo de dos palabras.	108
2.43. Ejemplo de alineación de patrones de voz.	110
2.44. Región de posibles trayectorias de la función de alineamiento.	112
2.45. Distancias locales generadas por los patrones de voz.	113
2.46. Conjunto de posibles trayectorias hacia el punto (n, m).	114
2.47. Técnica de alineación dinámica en el tiempo CE2-1.	116
2.48. Técnica de alineación dinámica en el tiempo UE2-1.	116
2.49. Técnica de alineación dinámica en el tiempo UELM.	117
2.50. Función warping y ventana de ajuste.	118
2.51. Slope constraint en función warping.	121
2.52. Coeficientes de pesos para la forma simétrica y la forma asimétrica.	123

2.53. Diagrama de flujo del algoritmo PD-Matching.	126
2.54. Distribuciones de probabilidad de las distancias intralocutor e interlocutor.	134
2.55. Diagrama de bloques general del proyecto.	140
3.1. Diseño del sistema empotrado y el modelo del ciclo de vida del desarrollo.	142
3.2. Ciclo de negocio de la arquitectura.	144
3.3. Modelo de referencia del sistema.	147
3.4. Código fuente de la función para grabar un audio.	149
3.5. Código fuente de la función para guardar un audio.	150
3.6. Código fuente de la función para abrir un audio.	150
3.7. Código fuente del algoritmo LMS.	151
3.8. Código fuente del algoritmo NLMS.	152
3.9. Código fuente de la función de normalización de datos.	152
3.10. Código fuente de la función del filtro de preénfasis.	153
3.11. Código fuente del algoritmo por función de energía.	154
3.12. Código fuente de la función de energía promedio.	154
3.13. Código fuente de la función de energía de tiempo corto.	155
3.14. Código fuente de la función de tasa de cruce por ceros.	155
3.15. Código fuente de la función de la media.	156
3.16. Código fuente de la función de desviación estándar.	156

3.17. Código fuente del algoritmo Rabiner y Sambur Tipo 1.	157
3.18. Código fuente del algoritmo Rabiner y Sambur Tipo 2.	158
3.19. Código fuente de la función para la segmentación de una señal.	159
3.20. Código fuente de la función de ponderado por ventana rectangular.	160
3.21. Código fuente de la función de ponderado por ventana de Hamming.	160
3.22. Código fuente de la función de relleno con ceros.	161
3.23. Código fuente de la función de la transformada rápida de Fourier.	162
3.24. Código fuente de la función para la obtención del espectro en potencia.	163
3.25. Código fuente de la función de remuestreo por bandas críticas.	164
3.26. Código fuente de la función para la obtención de coeficientes MFCC.	165
3.27. Código fuente de la función para la obtención de coeficientes MFCC delta.	166
3.28. Código fuente de la función para la obtención de coeficientes MFCC doble delta.	167
3.29. Código fuente de la función de distancia euclíadiana cuadrática.	168
3.30. Código fuente de la función de distancia de error cuadrático medio.	168
3.31. Código fuente de la función de DTW simétrico con $P = 1/2$ con ventana de ajuste.	169
3.32. Código fuente de la función de DTW simétrico con $P = 1/2$ sin ventana de ajuste.	170
3.33. Código fuente de la función para obtener el recorrido mínimo DTW.	171
3.34. Código fuente de la función para la construcción del patrón de referencia.	172
3.35. Inicio de sesión al sistema por primera vez.	173

3.36. Registro del usuario administrador del sistema.	174
3.37. Permiso de la aplicación para grabar audios en el dispositivo.	175
3.38. Permiso de la aplicación para acceder y almacenar archivos en el dispositivo.	175
3.39. Menú principal del sistema.	176
3.40. Lista de cuentas de usuario del sistema.	177
3.41. Registrar un nuevo usuario al sistema.	178
3.42. Menú de opciones para la administración de usuarios del sistema.	179
3.43. Lista de audios del patrón clave de usuario para el acceso al sistema.	180
3.44. Grabación de un nuevo audio en el sistema.	181
3.45. Opciones después de grabar un audio en el sistema.	182
3.46. Eliminar audios de un usuario del sistema.	183
3.47. Lista de los accesos de los usuarios en el sistema.	184
3.48. Interfaz para el reconocimiento del usuario y comandos por voz.	185
3.49. Grabación del comando de voz a reconocer.	186
3.50. Cancelar o enviar el comando de voz para su reconocimiento.	187
3.51. Reconocimiento del comando de voz encender.	188
3.52. Cerrar sesión del usuario en el sistema.	189
3.53. Menú de opciones antes de iniciar sesión en el sistema.	190
3.54. Respuestas después de tratar de acceder al sistema.	191

3.55. Ver micrófonos externos disponibles en red para el sistema.	192
3.56. Menú de opciones para el uso de un micrófono externo de un computador.	193
3.57. Activación del servidor y cliente para establecer la comunicación con el computador.	193
3.58. Actualizar lista de computadores conectados en red al servidor.	194
3.59. Solicitar el uso del micrófono de un computador.	195
3.60. Error en la conexión con arduino.	196
3.61. Conexión establecida con arduino.	196
3.62. Código fuente para establecer la comunicación con el dispositivo móvil en arduino.	198
3.63. Código fuente para establecer la comunicación con el arduino en android.	199
3.64. Código fuente para establecer la comunicación con el dispositivo móvil en el computador. .	200
3.65. Código fuente para enviar y recibir información al servidor en el computador.	201
3.66. Código fuente para establecer la comunicación con el computador en android.	202
3.67. Código fuente para recibir información del computador en android.	203
3.68. Código fuente para enviar información al computador en android.	204
3.69. Diseño del hardware del sistema de reconocimiento de voz.	206
3.70. Implementación del hardware del sistema de reconocimiento de voz.	207
3.71. Resultado del testeo para el <code>audio_1.wav</code> por algoritmo de función de energía.	237
3.72. Resultado del testeo para el <code>audio_1.wav</code> por algoritmo de Rabiner & Sambur tipo 1. .	237
3.73. Resultado del testeo para el <code>audio_1.wav</code> por algoritmo de Rabiner & Sambur tipo 2. .	238

3.74. Resultado del testeo para el <code>audio_2.wav</code> por algoritmo de función de energía.	238
3.75. Resultado del testeo para el <code>audio_2.wav</code> por algoritmo de Rabiner & Sambur tipo 1. . .	239
3.76. Resultado del testeo para el <code>audio_2.wav</code> por algoritmo de Rabiner & Sambur tipo 2. . .	239
3.77. Grafica de la señal de voz contaminada <code>voz_ruido.wav</code>	241
3.78. Grafica de la señal de ruido <code>ruido.wav</code>	241
3.79. Grafica de la señal después de aplicar LMS con $\mu = 0.02$	242
3.80. Grafica de la señal después de aplicar LMS con $\mu = 0.002$	242
3.81. Grafica de la señal después de aplicar NLMS con $\beta = 0.25$	243
3.82. Grafica de la señal después de aplicar NLMS con $\beta = 0.025$	243
3.83. Grafica de la señal después de aplicar NLMS con $\beta = 0.0025$	244
3.84. Grafica de la señal para el algoritmo NLMS con $\beta = 0.0025$ y Rabiner & Sambur Tipo 1 con $T = 128$	245
3.85. Código fuente del algoritmo de reconocimiento de voz para la etapa de entrenamiento. . .	248
3.86. Código fuente del algoritmo de reconocimiento de voz para la etapa de reconocimiento. . .	249
3.87. Código fuente del algoritmo de reconocimiento de voz para la etapa de toma de decisión. .	250
3.88. Código fuente de la función de cálculo del umbral de decisión.	250
3.89. Código fuente para la obtención de los umbrales de decisión para cada patrón de entrena- miento.	251
4.1. Número de errores en la respuesta del sistema para U2.	265

4.2.	Seguridad del sistema para U2.	265
4.3.	Número de errores en la respuesta del sistema para U1.	275
4.4.	Seguridad del sistema para U1.	275
4.5.	Gráfico de barras del tiempo de ejecución para el reconocimiento de voz.	277
4.6.	Gráfico de barras del tiempo de ejecución para el reajuste del umbral U1.	277
4.7.	Número de errores en la respuesta del sistema para U1 dinámico.	286
4.8.	Seguridad del sistema para U1 dinámico.	286
4.9.	Número de errores en la respuesta del sistema para U1 estático.	293
4.10.	Seguridad del sistema para U1 estático.	294
4.11.	Área de aceptación y rechazo de H0.	297

Índice de tablas

2.1.	Clasificación acústica de los sonidos.	20
2.2.	Formantes vocálicos.	21
2.3.	Aplicaciones de filtros adaptativos.	50
2.4.	Variantes del algoritmo LMS.	55
2.5.	Umbrales para la detección de inicio y fin de la señal de voz.	66
2.6.	Valores de las frecuencias para las bandas críticas triangulares.	98
2.7.	Correspondencia entre índices y frecuencias para las bandas críticas Mel.	102
2.8.	Correspondencia entre los valores para la octava banda crítica Mel.	103
2.9.	Algoritmos simétricos y asimétricos con condición de Slope Constraint $P = 0, 1/2, 1, 2$. . .	127
3.1.	Características generales del ciclo de negocio de la arquitectura.	145
3.2.	Resultados del testeo del sistema para $BC = 13, 16$ y 18 con $M = 13$	209
3.3.	Resultados del testeo del sistema para $BC = 20, 22$ y 24 con $M = 13$	210
3.4.	Resultados del testeo del sistema para $BC = 13, 16$ y 18 con $M = 15$	210
3.5.	Resultados del testeo del sistema para $BC = 20, 22$ y 24 con $M = 15$	211

3.6. Resultados del testeo del sistema para BC = 13, 16 y 18 con M = 26.	211
3.7. Resultados del testeo del sistema para BC = 20, 22 y 24 con M = 26.	212
3.8. Resultados del testeo del sistema para BC = 13, 16 y 18 con M = 39.	212
3.9. Resultados del testeo del sistema para BC = 20, 22 y 24 con M = 39.	213
3.10. Resultados del testeo del sistema para V = 28.	214
3.11. Resultados del testeo del sistema para V = 30.	215
3.12. Resultados del testeo del sistema para V = 32.	215
3.13. Resultados del testeo del sistema para P = 0 simétrico y asimétrico.	217
3.14. Resultados del testeo del sistema para P = 1 simétrico y asimétrico.	217
3.15. Resultados del testeo del sistema para P = 1/2 simétrico y asimétrico.	218
3.16. Resultados del testeo del sistema para P = 2 simétrico y asimétrico.	218
3.17. Resultados del testeo del sistema para P = 1/2 simétrico.	220
3.18. Resultados del testeo del sistema para P = 1 simétrico.	220
3.19. Resultados del testeo del sistema para P = 1 simétrico con S = 17 y 18.	222
3.20. Resultados del testeo del sistema para P = 1/2 simétrico con S = 17 y 18.	223
3.21. Resultados del testeo del sistema para P = 1 simétrico con S = 17 y 18.	224
3.22. Resultados del testeo del sistema para E = 1 y 3.	225
3.23. Resultados del testeo del sistema para V = 20 y 21.	227
3.24. Resultados del testeo del sistema para V = 22 y 23.	227

3.25. Resultados del testeo del sistema para $V = 24$ y 25	228
3.26. Resultados del testeo del sistema para $V = 26$ y 27	228
3.27. Resultados del testeo del sistema para $V = 28$ y 29	229
3.28. Resultados del testeo del sistema para $V = 30$ y 31	229
3.29. Resultados del testeo del sistema para $V = 32$	230
3.30. Resultados del testeo del sistema para $V = 27$ con $T = 192$ y 160	231
3.31. Resultados del testeo del sistema para $V = 27$ con $T = 144$	232
3.32. Resultados del testeo del sistema para $V = 28$ con $T = 192$ y 160	232
3.33. Resultados del testeo del sistema para $V = 28$ con $T = 144$	233
3.34. Resultados del testeo del sistema para $V = 30$ con $T = 192$ y 160	233
3.35. Resultados del testeo del sistema para $V = 30$ con $T = 144$	234
3.36. Resultados del testeo del sistema para distancia euclíadiana cuadrática y error cuadrático medio.	235
4.1. Resultados para obtener U_2 en el caso 1	257
4.2. Resultados para obtener U_2 en el caso 2	258
4.3. Resultados para obtener U_2 en el caso 3	259
4.4. Resultados para obtener U_2 en el caso 4	260
4.5. Resultados para obtener U_2 en el caso 5	261
4.6. Resultados para obtener U_2 en el caso 6	262

4.7. Resultados para obtener U ₂ en el caso 7	263
4.8. : Resultados para obtener U ₂ en el caso 8.	264
4.9. Resultados para obtener U ₁ en el caso 1.	267
4.10. Resultados para obtener U ₁ en el caso 2.	268
4.11. Resultados para obtener U ₁ en el caso 3.	269
4.12. Resultados para obtener U ₁ en el caso 4.	270
4.13. Resultados para obtener U ₁ en el caso 5.	271
4.14. Resultados para obtener U ₁ en el caso 6.	272
4.15. Resultados para obtener U ₁ en el caso 7.	273
4.16. Resultados para obtener U ₁ en el caso 8.	274
4.17. Comparación de numero de errores entre U ₁ = U y U ₁ = 0.51.	276
4.18. Resultados para el caso 1 con U ₁ dinámico.	280
4.19. Resultados para el caso 2 con U ₁ dinámico.	281
4.20. Resultados para el caso 3 con U ₁ dinámico.	281
4.21. Resultados para el caso 4 con U ₁ dinámico.	282
4.22. Resultados para el caso 5 con U ₁ dinámico.	283
4.23. Resultados para el caso 6 con U ₁ dinámico.	284
4.24. Resultados para el caso 7 con U ₁ dinámico.	284
4.25. Resultados para el caso 8 con U ₁ dinámico.	285

4.26. Resultados para el caso 1 con U1 estático.	287
4.27. Resultados para el caso 2 con U1 estático.	288
4.28. Resultados para el caso 3 con U1 estático.	289
4.29. Resultados para el caso 4 con U1 estático.	289
4.30. Resultados para el caso 5 con U1 estático.	290
4.31. Resultados para el caso 6 con U1 estático.	291
4.32. Resultados para el caso 7 con U1 estático.	292
4.33. Resultados para el caso 8 con U1 estático.	292
4.34. Comparación de numero de errores entre $U_1 = U$ y $U_1 = 0.51$	295
4.35. Tabla de contingencia o de frecuencias observadas.	296
4.36. Tabla de frecuencias esperadas.	296
4.37. Tabla con los valores ji-cuadrada.	298
A.1. Tabla de distribución Chi-Cuadrado.	314

Índice general

Dedicatoria	I
Agradecimientos	II
Resumen	III
Abstract	IV
Lista de símbolos	V
Índice de figuras	XVIII
Índice de tablas	XXIII
1. Introducción	1
1.1. Justificación de la investigación	6
1.2. Formulación del problema	7
	XXIV

1.3. Hipótesis	7
1.4. Objetivos	7
1.4.1. Generales	7
1.4.2. Específicos	7
1.5. Estructura de la tesis	9
2. Materiales y métodos	11
2.1. Marco teórico	11
2.1.1. Sistema de reconocimiento de voz	11
2.1.1.1. Historia del reconocedor de voz	11
2.1.1.2. Fundamentos sobre sonidos y la señal de voz	14
2.1.1.3. Modelo digital para la voz (Fuente-Filtro)	26
2.1.1.4. Funcionamiento de un sistema de reconocimiento de voz	27
2.1.2. Procesamiento o análisis de la señal de voz	30
2.1.2.1. Preprocesamiento	31
2.1.2.2. Entrenamiento	83
2.1.2.3. Reconocimiento	106
2.2. Método de la investigación	135
2.2.1. Diseño de la investigación	135
2.2.2. Variables de estudio	135

2.2.3.	Métodos y procedimientos para la recolección de datos	136
2.2.4.	Población y muestra	136
2.2.4.1.	Población	136
2.2.4.2.	Muestra	137
2.2.5.	Método de estudio	138
2.2.6.	Ánalisis estadístico de los datos	139
2.2.7.	Esquema general del proyecto	139
3.	Sistema de control de acceso por reconocimiento de voz	141
3.1.	Fase 1. Creación de la arquitectura	143
3.1.1.	Etapa 1: Tener una base técnica sólida	143
3.1.2.	Etapa 2: Entender el ciclo de negocio de la arquitectura	144
3.1.3.	Etapa 3: Definir el patrón arquitectónico y modelo de referencia	146
3.2.	Fase 2. Implementación de la arquitectura	148
3.2.1.	Desarrollo del software de aplicación	148
3.2.1.1.	Software de reconocimiento de voz	148
3.2.1.2.	Software de administración de usuarios	173
3.2.2.	Desarrollo del software del sistema	197
3.2.3.	Construcción del hardware	205
3.2.3.1.	Instrumentos	205

3.2.3.2. Diseño del hardware del sistema	205
3.2.3.3. Implementación del hardware del sistema	207
3.3. Fase 3. Testeo del sistema	208
3.3.1. Hallar BC y M	208
3.3.2. Hallar V	213
3.3.3. Hallar P	216
3.3.4. Hallar S	219
3.3.5. Hallar S	221
3.3.6. Hallar r	222
3.3.7. Hallar T	224
3.3.8. Hallar V y S	226
3.3.9. Hallar V, S, y T	230
3.3.10. Hallar d	234
3.3.11. Hallar el algoritmo para la detección de inicio y fin de la señal de voz	236
3.3.12. Hallar el algoritmo para la eliminación de ruido aditivo	240
3.4. Fase 4. Mantenimiento del sistema	247
4. Resultados y discusión de la tesis	252
4.1. Pruebas para obtener los umbrales de decisión	256
4.1.1. Prueba para obtener U2	256

4.1.1.1.	Caso 1	256
4.1.1.2.	Caso 2	257
4.1.1.3.	Caso 3	258
4.1.1.4.	Caso 4	259
4.1.1.5.	Caso 5	260
4.1.1.6.	Caso 6	261
4.1.1.7.	Caso 7	262
4.1.1.8.	Caso 8	263
4.1.2.	Prueba para obtener U1	266
4.1.2.1.	Caso 1	266
4.1.2.2.	Caso 2	267
4.1.2.3.	Caso 3	268
4.1.2.4.	Caso 4	269
4.1.2.5.	Caso 5	270
4.1.2.6.	Caso 6	271
4.1.2.7.	Caso 7	272
4.1.2.8.	Caso 8	273
4.2.	Pruebas para la construcción del patrón de referencia	279
4.2.1.	Prueba con U1 dinámico	279

4.2.1.1. Caso 1	279
4.2.1.2. Caso 2	280
4.2.1.3. Caso 3	281
4.2.1.4. Caso 4	282
4.2.1.5. Caso 5	282
4.2.1.6. Caso 6	283
4.2.1.7. Caso 7	284
4.2.1.8. Caso 8	285
4.2.2. Prueba con U1 estático	287
4.2.2.1. Caso 1	287
4.2.2.2. Caso 2	288
4.2.2.3. Caso 3	288
4.2.2.4. Caso 4	289
4.2.2.5. Caso 5	290
4.2.2.6. Caso 6	290
4.2.2.7. Caso 7	291
4.2.2.8. Caso 8	292
4.2.3. Prueba de hipótesis o de significación	296
5. Consideraciones finales	299

5.1. Conclusiones	299
5.2. Trabajos futuros	303
Referencias bibliográficas	306
A. Primer apéndice	311
A.1. Prueba de independencia Chi-Cuadrado de Pearson	311

Capítulo 1

Introducción

Hoy en día muchas personas tienen la necesidad de conocer quién y cuando alguien ha tenido acceso, ya sea a una zona de sus viviendas, a cierta información, a un servicio, etc. Por ejemplo, hay centros de trabajo en los que los trabajadores tienen turnos de noche y para poder acceder a su puesto de trabajo necesitan de una llave, ya sea porque a esas horas no hay portero, recepcionista, o simplemente porque la empresa no cuenta con ninguna persona que se encargue de abrirla, debido a esto los empleados deben turnarse para realizar esta tarea, interrumpiendo con esto su jornada laboral y perjudicando su productividad.

Ocurre también que en muchas empresas es conveniente que no todos los empleados tengan acceso a ciertas zonas, por ejemplo, cuando en algunas áreas se trabaja con materiales peligrosos, documentación importante, donde se guarda dinero o material de mucho valor, debido a esto es necesario delimitar estas zonas creando permisos distintos en función de a qué áreas se pueden o no acceder, (ALTUMTEC, 2018).

Debido a estos problemas que ocurren en el día a día, han dado origen a lo que hoy se conoce como *Sistemas de Seguridad para el Control de Acceso*, que es la actividad principal en la seguridad física e informática. El control de acceso es posiblemente la medida de seguridad más importante que se puede establecer, dado que un control bien aplicado nos permite saber quién, cuándo y a qué parte se accedió.

Es por ello que los controles de acceso facilitan el trabajo a la empresa o lugar donde sea utilizado, estos controles tanto físicos como lógicos se encuentran entre los esquemas de protección imperativos en seguridad. Los controles de acceso físico aseguran que solo el personal autorizado tenga disposición a edificios, salas, documentos, etc. Por otro lado, los controles de acceso lógico protegen las computadoras, las instalaciones de red y los sistemas de información contra amenazas de acceso no autorizado. Estos dos tipos de accesos se basan esencialmente en la autentificación del usuario mediante la cual la identidad de un individuo se verifica a través de uno de los tres medios siguientes: por algo *que conoce, que tiene o que es* (o por combinaciones de estos).

Los sistemas de seguridad han ido evolucionando conforme al avance tecnológico, los primeros sistemas de identificación fueron los *Sistemas de Identificación por Teclado*, donde el usuario utiliza un código o PIN (Número de Identificación Personal) para identificarse. Es económico y práctico porque no requiere identificadores físicos y hace innecesarios las llaves. Si bien este sistema es muy cómodo, presenta un problema que es, que los usuarios pueden compartir la clave

fácilmente, lo que compromete la seguridad, Fermax (2016).

Ante este problema aparecieron los *Sistemas de Identificación por Proximidad y Radiofrecuencia*, estos dos métodos cuentan con algo en común que es, que el usuario tiene un artículo identificador. En los *Sistemas de Proximidad*, los usuarios usan tarjetas o llaveros electrónicos para identificarse, estos no necesitan de mantenimiento, aunque sí el sistema, pero su costo es bajo. Por otro lado, los *Sistemas de Radiofrecuencia* son muy comunes en los garajes, a los que se accede con un pequeño mando a distancia, es un sistema cómodo, aunque el tipo de identificador es más costoso que el usado por proximidad. La seguridad que ofrecen estos sistemas es mayor en comparación a un *Sistema de Identificación por Teclado*, ya que un usuario, solo puede utilizar una tarjeta o un llavero a la vez, el problema es que, si perdemos la tarjeta o llavero electrónico, el sistema quedaría inservible y tendría que reconstruirse, ocasionando costos mayores, Fermax (2016).

Debido a este nuevo problema aparecen los *Sistemas de Identificación Biométrica*, la biometría está cada vez más presente en la vida cotidiana de las personas y, sobre todo, en los negocios. El control de acceso biométrico es un sistema de identificación basado en las cualidades fisiológicas del usuario, pero también puede basarse en el comportamiento del mismo, se trata de un sistema personal e intransferible, Fermax (2016). Según (Sanchez, 2015), la biometría es más cómoda y segura que los sistemas tradicionales como las contraseñas, llaves o tarjetas.

En (Cosentino, 2016) se explica que dentro de la biometría como sistema de acceso podemos diferenciar varios tipos de reconocimiento: identificación por geometría de la mano, huella digital, iris y retina y por voz.

El Reconocimiento por Geometría de Mano, identifica los parámetros dimensionales de la mano, que son únicos. Funciona muy bien, sobre todo en lugares donde no se utilizan guantes y es bastante robusta frente al vandalismo. Los motivos por el cual esta tecnología no se popularizó masivamente son estrictamente comerciales, y es que sólo había un fabricante en el mercado, lo que hizo que los fabricantes de sistemas de control de acceso no ayudaran a difundirla.

El Reconocimiento por Huella Digital, sin dudas la más popular de todas las biometrías. Es utilizado por los organismos de seguridad y gobiernos para la identificación de personas y/o sospechosos de delitos, y también en controles de acceso. No son resistentes al vandalismo y uno de los grandes inconvenientes es que la reparación del daño en el elemento sensor es bastante costoso, por lo que se recomienda utilizarlos en lugares no expuestos a estos inconvenientes. Algo similar ocurre con las inclemencias del tiempo, por lo que se deberán extremar los cuidados, si es necesario colocarlos en el exterior y sobre todo expuestos a la intemperie.

El Reconocimiento Facial, es una tecnología que año a año mejora en sus resultados y si bien actualmente existen fabricantes que ofrecen soluciones de este tipo, todavía no se consiguen equipos comerciales de bajo costo con prestaciones aceptables de identificación y velocidad. Básicamente

camiente consiste en identificar y calcular las distancias entre los diferentes accidentes faciales, reduciendo la imagen a un conjunto de coordenadas de puntos significativos.

El Reconocimiento de Iris y Retina, los equipos de este tipo de reconocimiento funcionan muy bien, tienen un costo relativamente accesible y no tienen contacto físico con el usuario, por lo que pueden ser ubicados detrás de un vidrio de manera de hacerlos resistentes al vandalismo. El inconveniente radica en que no son muy populares, por tratarse de equipos en los que la persona debe mirar adentro y si bien su ojo no tiene contacto con ningún elemento, de todas formas, suelen generar el rechazo de los usuarios.

Y por último el *Reconocimiento de Voz*, esta tecnología se encuentra en una etapa similar al reconocimiento facial y probablemente algún día sean una alternativa válida.

Como podemos observar, las tecnologías biométricas pueden ser una alternativa o un complemento de las técnicas de identificación y autenticación ya existentes. Debido a esto y a la poca producción de sistemas reconocedores de voz, esta investigación se basará en este tipo de sistema.

Con esto, finalmente podemos decir que la tecnología puede ser una herramienta sumamente útil y poderosa para reducir el riesgo a ser víctimas de intrusos en nuestras viviendas. Es por ello que nos proponemos a diseñar un sistema tecnológico que ayudará a solucionar este problema.

1.1. Justificación de la investigación

Debido a la inseguridad actual que se presenta en la ciudad de Trujillo se realizará un sistema de seguridad para una vivienda por medio de la voz, con la finalidad de proponer a los habitantes de la ciudad un sistema capaz de resguardar su vivienda.

Por otro lado, cada vez hay personas menos preparadas en la manipulación de sistemas tecnológicos, debido a la poca familiaridad que tienen con estos, por ello esta investigación propone a los habitantes de la ciudad de Trujillo un sistema simple y fácil de usar.

Si bien existen muchos tipos de sistemas de seguridad, donde los biométricos destacan, estos suelen tener un costo elevado para su implementación, debido a que los componentes que se utilizan para la captura y el procesamiento de los datos son caros, siendo así difícil la venta y compra de estos, es por ello que este proyecto servirá como una alternativa ante este problema, permitiendo a la sociedad la adquisición de estos sistemas a un precio cómodo.

Además, actualmente no existe ninguna API (Interfaz de Programación de Aplicaciones) de reconocimiento de voz que este de manera gratuita para los desarrolladores de programas informáticos que deseen usar el reconocimiento de voz en sus aplicaciones. Si bien Google ofrece gratuitamente su API de reconocimiento de voz a los desarrolladores de aplicaciones para dispositivos Android, ocurre que para el uso de esta API es necesario la conexión a internet, lo que hace

que su uso tenga esta limitante, es por ello que nuestro proyecto será una alternativa para aquellos desarrolladores que no cuentan con dinero y acceso a internet, puedan usar esta tecnología en sus programas.

1.2. Formulación del problema

¿Cómo controlar el acceso a una vivienda por reconocimiento de voz del locutor dependiente del texto?

1.3. Hipótesis

El acceso a una vivienda puede ser controlado por reconocimiento de voz del locutor dependiente del texto haciendo uso de los métodos de cuantificación de la señal de voz MFCC Y DTW.

1.4. Objetivos

1.4.1. Generales

Desarrollar un sistema de seguridad para el control de acceso a una vivienda que funcione en base al reconocimiento de voz dependiente del texto.

1.4.2. Específicos

- a) Aplicar la metodología propuesta por Tammy Noergaard para el diseño de arquitecturas para sistemas empotrados o embebidos.

- b) Analizar la transmisión, aplicación y manipulación de las señales de voz para poderlas utilizar en el desarrollo del sistema de seguridad de control de acceso.
- c) Implementar un algoritmo para la extracción de características de la señal de voz para la etapa de entrenamiento.
- d) Implementar un algoritmo para la comparación entre dos señales de voz para la etapa de reconocimiento.
- e) Implementar un algoritmo para la eliminación de ruido externo o ambiental producida por alguna fuente de ruido.
- f) Implementar un algoritmo para la detección de inicio y fin de la señal de voz.
- g) Integrar y evaluar los algoritmos de los módulos de preprocesamiento, entrenamiento y reconocimiento, de manera que se genere un modelo con menor tasa de error en el reconocimiento de voz.
- h) Desarrollar un software que funcione como interfaz con el usuario y que además realice las comparaciones y cálculos necesarios para el reconocimiento del locutor.
- i) Construir un prototipo hardware para el acceso a una vivienda.
- j) Implementar un canal de comunicación inalámbrica entre el hardware y el software.

- k) Evaluar la seguridad de nuestro sistema ante intrusos en un ambiente real.
- l) Evaluar el rendimiento y minimizar el tiempo de respuesta para el reconocimiento de voz.

1.5. Estructura de la tesis

El presente trabajo está dividido en cinco capítulos. El primer capítulo presenta los aspectos generales de la investigación realizada tal como justificación, formulación del problema, la hipótesis, los objetivos y la estructura de la tesis.

En el segundo capítulo se presenta el marco referencial teórico y soporte del tema, contemplando los conceptos básicos que se debe de tener en cuenta para el desarrollo de un sistema de seguridad para el control de acceso por reconocimiento de voz, así como algunas consideraciones con respecto al habla. Además, se dará a conocer los métodos y las etapas por las cuales debe pasar un reconocedor de voz para poder procesar y manejar las señales de voz por medio de un software. Finalmente el método empleado en la investigación.

El tercer capítulo muestra los procesos para la construcción de nuestro sistema de seguridad para el control de acceso, dividido en tres etapas: la primera la construcción del algoritmo de reconocimiento de voz y su integración en una aplicación Android, la segunda la construcción del prototipo hardware y la tercera el canal de comunicación entre los dos módulos.

El cuarto capítulo se muestran los resultados y discusión obtenida en la investigación. El quinto capítulo contiene las consideraciones finales obtenidas en esta tesis. Primero se presentan las conclusiones, seguida de las recomendaciones para futuras investigaciones relacionadas al tema en cuestión.

Por último, las referencias bibliográficas que se usaron para la investigación en esta tesis y en apéndice se presenta el instrumento que se usará para constatar la hipótesis. Finalmente la declaración jurada y la autorización de la tesis.

Capítulo 2

Materiales y métodos

2.1. Marco teórico

2.1.1. Sistema de reconocimiento de voz

2.1.1.1. Historia del reconocedor de voz

En 1870's, Alexander Graham Bell, quería construir un dispositivo que hiciera el habla visible a las personas con problemas auditivos, el cual tuvo como resultado el teléfono.

En 1880's, Tihamir Nemes, solicita permiso para una patente de desarrollo de un sistema de transcripción automática que identificara secuencias de sonidos y los imprimiera (texto). Pero fue rechazado como proyecto no realista.

En 1910's, AT&T Bell Laboratories, construye la primera máquina capaz de reconocer voz basada en plantillas de los 10 dígitos en inglés. Requería un extenso reajuste a la voz de una persona, pero una vez logrado tenía un 99 % de certeza. Por lo tanto, surge la esperanza de que el

reconocimiento de voz es simple y directo.

En 1940's y 1950's, la mayoría de los investigadores reconocen que era un proceso mucho más intrincado y sutil de lo que habían anticipado. Por lo tanto, empiezan a reducir los alcances y se enfocan a sistemas más específicos:

- Dependientes del locutor.
- Flujo discreto del habla (con espacios o pausas entre palabras).
- Vocabulario pequeño (menor o igual a 50 palabras).

Estos sistemas empiezan a incorporar técnicas de normalización del tiempo (minimizar diferencia en velocidad del habla). Además, ya no buscaban una exactitud perfecta en el reconocimiento, (Schroeder, 1998). Después IBM y CMV trabajan en reconocimiento de voz continuo, pero no se ven resultados hasta los 1970's.

En 1970's, un ambicioso proyecto para construir un sistema informático que entienda y procese el habla fue iniciado por DARPA (Jelinek, 2004), la meta era integrar conocimiento acerca del habla, lingüística e inteligencia artificial para desarrollar un *Sistema Informático del Lenguaje Hablado*. Se desarrolló un sistema informático llamado *Harpy* que integraba todas las fuentes de conocimiento en redes de estado finito, que eran entrenadas estadísticamente. En estos años crecen notoriamente métodos que hacen uso de la probabilidad y se entiende por *Reconocimiento*

Automático del Habla como buscar la palabra más probable en una señal de audio, dada alguna información de su distribución de probabilidad.

En 1980's, modernos reconocedores son lanzados al mercado, los algoritmos desarrollados en este tiempo son usados todavía en estos días como: Modelos n-grams, Mixturas Gaussianas, Modelos Ocultos de Markov, Decodificador Viterbi, etc. En 1984 es construido el primer sistema de dictado en tiempo real, por IBM.

En 1990's, con el aumento del poder de cómputo y capacidad de memoria de las computadoras existen avances en algoritmos de adaptación y entrenamiento discriminativo, a la vez se desarrollan sistemas informáticos que reconocían palabras independientemente quien fuera el hablante, algunas de estas aplicaciones fueron implantadas en empresas telefónicas. En 1995, Dragon IBM lanza su producto que reconocía palabras aisladas dependiente del hablante, era un sistema que permitía un dictado de un gran número de palabras. En 1997, Dragon IBM lanza su sistema para dictado continuo. En 2000's, teléfonos celulares ya ofrecían servicios de marcado activado por voz.

En 2010's, salió a la luz el asistente de voz Siri, realizando funciones de asistente personal a veces con su propia personalidad para iOS, macOS, tvOS y watchOS. Esta aplicación utiliza procesamiento del lenguaje natural para responder preguntas, hacer recomendaciones y realizar acciones mediante la delegación de solicitudes hacia un conjunto de servicios web que ha ido aumentando con el tiempo. También han aparecido asistentes de voz como Cortana de Microsoft y

Bixby para dispositivos Android, (Toast, 2018).

En la actualidad, se espera que con el poder de cómputo actual de las máquinas y las diversas investigaciones en el tema, se logre aumentar el desempeño. Aplicaciones actuales van desde software para celulares, robótica hasta interfaces de voz para personas discapacitadas; desafortunadamente las comparaciones de error entre humanos y máquinas todavía dan un amplio margen de diferencia, por lo que se espera lograr que éste margen disminuya en los próximos años.

2.1.1.2. Fundamentos sobre sonidos y la señal de voz

La lengua es uno de los órganos más presentes a la hora de hablar, aunque no es el único implicado. En la producción de la voz y en su percepción intervienen muchos órganos y partes del cuerpo (labios, garganta, cerebro, oídos, pulmones, boca, nariz, etc.), sin embargo, hay que tener en cuenta que estas no se emplean solamente para hablar, sino que también desempeñan otras funciones adicionales (comer, respirar, etc.).

Ahora veremos algunos conceptos empleados en el estudio de las señales de voz, así como en el funcionamiento de los sistemas generadores de voz, de tal forma que se puedan establecer las características que sirvan para realizar un correcto reconocimiento de voz. Existen varias maneras para analizar y describir el habla. Los enfoques más comúnmente usados son:

- **Articulación:** Análisis de cómo el humano produce los sonidos del habla.

- **Señal Acústica:** Análisis de la señal de voz como una secuencia de sonidos.

- **Percepción Auditiva:** Análisis de cómo el humano procesa el habla.

Los tres enfoques proveen ideas y herramientas para obtener mejores y más eficientes resultados en el reconocimiento.

a) Articulación

La voz humana se produce por medio del aparato fonatorio, también llamado aparato bucal o articulatorio, el cual está formado por un conjunto de órganos. El aparato fonador consiste en tres grupos de órganos diferenciados, tal como se ve en la Figura 2.1

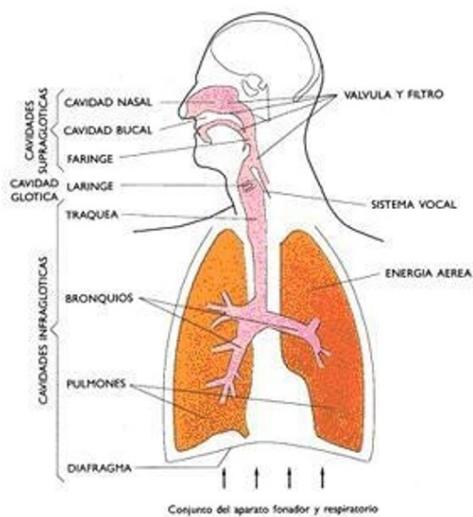


Figura 2.1: Órganos del cuerpo que intervienen en el aparato fonador.

Fuente: Cruz (2009)

- **Órganos de respiración o cavidades infraglóticas:** pulmones, bronquios y tráquea.
- **Órganos de fonación o cavidades glóticas:** laringe, cuerdas vocales y resonadores nasal, bucal y faríngeo.
- **Órganos de articulación o cavidades supraglóticas:** paladar, lengua, dientes, labios y glotis.

El proceso básico de generación de la voz es el mismo ya sea al hablar o al cantar. El cerebro envía señales a través del sistema nervioso a los músculos de la cabeza, cuello y torso de manera que se produzca la inhalación previa a la generación. Al final de la inhalación se efectúan varias acciones, primero se realiza el movimiento de los cartílagos aritenoides en la laringe acercando a las cuerdas vocales entre sí, luego el volumen de los pulmones disminuye para producir una presión de aire positiva en los mismos, después el aire comienza a fluir hacia la laringe generando una excitación y esta adquiere las formas siguientes: fonación, susurro, fricación, compresión y vibración.

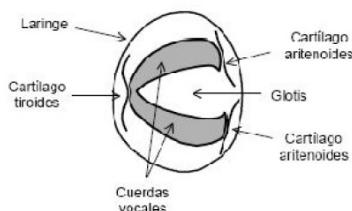


Figura 2.2: Corte esquemático de la laringe en un plano horizontal.
Fuente: Owens (1993)

- Fonación:

Este término se refiere a la oscilación de las cuerdas vocales por los movimientos de los cartílagos aritenoides, cuando el aire es conducido a través de las cuerdas vocales, éstas vibran y sus oscilaciones son dirigidas por la masa y tensión de las cuerdas. La apertura y cierre de las cuerdas secciona el pulso de aire en pulsos cuasiperiódicos llamados pulsos glotales, con una frecuencia fundamental llamada tono. Los sonidos resultantes de la fonación se llaman sonoros, mientras que sonidos con ausencia de fonación se denominan sordos. Así, por ejemplo, las vocales son sonidos sonoros y las consonantes, como la *f*, *s*, *p* y *k*, son sonidos sordos.

- Susurro:

Los susurros son generados en la laringe. Las cuerdas vocales están juntas por el cartílago aritenoides, pero en lugar de sellar completamente la glotis existe una pequeña apertura triangular entre estos cartílagos, el aire que fluye a través de esta apertura genera turbulencias, que ocasionan ruido de banda ancha, el cual sirve como señal excitadora. Además, los susurros son más débiles que las fonaciones ya que implican un menor volumen de aire, y tienen mayor energía en las altas frecuencias. Sin embargo, se producen sonidos inteligibles si todo un conjunto de sonidos con fonación es reemplazado por susurros de los mismos sonidos. Esto es común cuando se trata de hablar en voz muy baja.

- Fricación:

La fricación es similar al susurro en cuanto el flujo de aire turbulento genera ruido de banda ancha, pero existe un lugar de articulación adicional en el tracto vocal (dientes, lengua, etc.) Estos sonidos producidos son llamados fricativos, y la fricación puede ocurrir con o sin fonación.

Dado que el lugar de articulación es cerca de los labios, sólo una pequeña parte del tracto vocal está entre la fuente de excitación y el aire de salida. Esto significa que la modulación producida por el tracto vocal está limitada en extensión y complejidad. Al igual que el susurro, la fricación es más baja en amplitud que la fonación y tiene una proporción mucho más amplia de las altas frecuencias. Sin embargo, el filtrado muy limitado del tracto vocal, con sus pérdidas menores a bajas frecuencias, permite una mayor radiación de energía, por consiguiente, los sonidos fricativos son más sonoros que los susurros. La fricación puede ocurrir durante interrupciones de fonaciones, esto implica variaciones muy rápidas de amplitud y frecuencia, por lo que juega el papel de modulador.

- Comprensión:

Cuando el tracto vocal está prácticamente cerrado y una persona sigue exhalando, la presión aumenta y resulta un pequeño transitorio. La combinación de un silencio pe-

queño seguido por una ráfaga de ruido crea una excitación aperiódica. Si el transitorio es abrupto y limpio, el sonido es una oclusiva o plosiva como en el caso de *p*, si es gradual y turbulento, éste se clasifica como un sonido muy parecido al fricativo, llamado africativo, como en el caso de las letras *ch* e *y*. Las africadas, generalmente, se comportan como un estado intermedio entre oclusivas y fricativas, pero fonéticamente son secuencias de una oclusiva más una fricativa.

- **Vibración:**

La vibración es cuasiperiódica y puede ocurrir en muchos lugares del tracto vocal, por ejemplo, la vibrante *r* involucra la vibración de la lengua contra el paladar. Estas vibraciones pueden ocurrir con o sin fonación y su efecto principal es la interrupción.

Las voces del hombre y la mujer difieren debido a varios aspectos, que incluyen el tamaño de la laringe, el tono, el rango de tono, el espacio entre las cuerdas vocales y ocurrencia de problemas del habla. Antes de la pubertad, en promedio el tono y el tamaño de la laringe es el mismo tanto en hombres como en mujeres. La frecuencia fundamental de la voz (la cual está estrechamente relacionada con la percepción del tono) está alrededor de los 250 Hz y la longitud de las cuerdas vocales es de aproximadamente 10.4 mm.

Durante la pubertad, el tamaño de las cuerdas se incrementa en promedio de 5-10 mm. en el caso de los hombres y de 3-5 mm. en el caso de las mujeres. Este aumento reduce el tono

promedio a 120 Hz en el hombre y 200 Hz en la mujer. De manera que el tono más alto de la mujer comparado con el hombre es casi una octava, significa que las cuerdas vocales vibran casi el doble de veces por segundo que en los hombres.

En resumen, en el habla los formantes se determinan por el proceso de filtrado que se produce en el tracto vocal por la configuración de los articuladores. El sonido particular de la voz de cada persona está determinado por el tamaño y la forma de las cuerdas vocales, de la garganta, de la nariz, de la boca, etc. De tal manera que los sonidos se ven afectados por la forma del tracto vocal.

Dada la anterior explicación es necesaria una clasificación acústica, como la que se resume en la Tabla 2.1.

Tabla 2.1: Clasificación acústica de los sonidos.

Sonidos periódicos compuestos o complejos	Vibración de la cuerdas vocales(frecuencia fundamental, F0) y resonancia (armónicos)	Vocales, nasales, laterales
Sonidos aperiódicos impulsionales	Cierre y explosión en el tracto vocal	Oclusivas
Sonidos aperiódicos continuos	Fricción en el tracto vocal	Fricativas

Fuente: Llisterri (2018)

Los formantes son elementos que sirven para distinguir componentes del habla humana, principalmente, las vocales y sonidos sonantes. El formante con la frecuencia más baja se

llama $F1$, el segundo $F2$, el tercero $F3$, etc. Normalmente sólo los dos primeros son necesarios para caracterizar una vocal, aunque la pueden caracterizar más formantes. Generalmente, los formantes posteriores determinan propiedades acústicas como el timbre. Los dos primeros formantes se determinan principalmente por la posición de la lengua. Donde $F1$ tiene una frecuencia más alta cuanto más baja esta la lengua, es decir una mayor abertura. Para el $F2$ tiene mayor frecuencia cuanto más hacia delante está posicionada la lengua.

No todos los sonidos se componen de formantes definidos. Solamente aparecen en sonantes, que incluyen los sonidos pulmonares (vocales y nasales). Los nasales tienen un formante adicional $F3$, en torno a los 1500 Hz. Si la frecuencia fundamental es mayor que la frecuencia de los formantes, entonces el carácter del sonido se pierde y se vuelven difíciles de distinguir, por lo cual son difíciles de reconocer. En la Tabla 2.2 se muestran algunos anchos de banda entre los cuales se localizan las vocales:

Tabla 2.2: Formantes vocálicos.

Vocal	Región principal formantica
/u/	200 a 400 Hz
/o/	400 a 600 Hz
/a/	800 a 1200 Hz
/e/	400 a 600 Hz y 2200 a 2600 Hz
/i/	200 a 400 Hz y 3000 a 3500 Hz

Fuente: Wikipedia (2016)

b) Señal acústica

Un reconocedor no puede analizar los movimientos en la boca. En su lugar, la fuente de información es la señal de voz misma. El habla es una señal analógica, es decir, un flujo continuo de ondas sonoras y silencios. El conocimiento de la ciencia de la acústica se utiliza para identificar y describir los atributos del habla que son necesarios para un reconocimiento de voz efectivo, Cruz (2009).

Algunas características importantes del análisis acústico son:

- Frecuencia y amplitud:

Todos los sonidos causan movimientos entre las moléculas del aire. Los sonidos más simples son los sonidos puros, y se pueden representar gráficamente por una onda sencional.

La frecuencia es el número de vibraciones (ciclos) del tono por segundo y se mide en Hz (hertz), donde los tonos altos tienen mayor frecuencia y los tonos bajos tienen menor frecuencia.

El volumen de un sonido refleja la cantidad de aire que es forzada a moverse. Esta describe y representa como amplitud de la onda y se mide en dBC (decibéles).

- Resonancia:

La resonancia se define comúnmente como la habilidad que tiene una fuente vibrante de sonido de causar que otro objeto vibre gracias a ella. La mayoría de los sonidos incluyendo el habla tienen una frecuencia dominante llamada frecuencia fundamental, también conocida como pitch (tono) que se combina con frecuencias secundarias en el habla, la frecuencia fundamental es la velocidad a la que vibran las cuerdas vocales al producir un fonema sonoro. Sumadas a la frecuencia fundamental hay otras frecuencias que contribuyen al timbre del sonido (lo que nos permiten distinguir una trompeta de un violín, las voces de diferentes personas, etc.). Algunas bandas de la frecuencia secundarias juegan un rol importante en la distinción de un fonema de otro. Se les llama formantes y son producidas por la resonancia, tal y como vimos anteriormente.

- Estructura armónica y ruido:

El habla no es un tono puro es continuación de múltiples frecuencias y se representa como una onda compleja. Las vocales se componen de 2 o más ondas simples, ricos en frecuencias secundarias y contienen estructuras internas de ondas cíclicas y acíclicas.

Las ondas acíclicas no tienen patrones repetitivos, son llamados ruido y forman parte de todos los fonemas sonoros, consonantes y semivocales. Las frecuencias y características de estas ondas proveen información importante sobre la identidad de los fonemas.

La identidad de las consonantes también se revela por el cambio en las formantes que resultan cuando los articuladores se mueven de un fonema anterior a la consonante y de ella al siguiente fonema llamadas transiciones de formantes. Estas se analizan utilizando técnicas como FFT (Transformada Rápida de Fourier) generando espectrogramas.

La complejidad de las formas de onda de los fonemas y las constantes transiciones de un patrón a otro dificultan el análisis de los patrones utilizando las representaciones complejas de las ondas. Los patrones armónicos y de ruido se muestran con más claridad utilizando los espectrogramas de banda ancha. La localización (la distancia entre ellas) y cambio en las formantes ayudan a identificar fonemas y palabras.

- Coarticulación:

Los fonemas aparentemente tienen parámetros acústicos claramente definidos, pero más bien, los fonemas tienden a ser abstracciones implícitamente definidas por la pronunciación de palabras en un lenguaje. La forma acústica de un fonema depende fuertemente del contexto acústico en el que sucede, a este efecto se le llama coarticulación.

Los investigadores, utilizan este concepto para distinguir entre la característica conceptual de un sonido del habla y una instancia o pronunciación específica de ese fonema, Franco (2010).

c) Percepción auditiva

La variabilidad del habla producida por coarticulación y otros factores hacen del análisis de la voz extremadamente difícil.

La facilidad del humano en superar estas dificultades sugiere que un sistema basado en la percepción auditiva podría ser un buen enfoque desafortunadamente el conocimiento de la percepción humana es incompleto, lo que se sabe es que el sistema auditivo está adaptado a la percepción de la voz.

El oído humano detecta frecuencias de 20 a 20000 Hz, pero es más sensible al rango entre 1000 y 6000 Hz, además de los pequeños cambios en la frecuencia en el ancho de banda, crítico para el habla. El patrón de sensibilidad a cambios en el tono no corresponde a la escala lineal de frecuencias de ciclos por segundo de la acústica. Para representar mejor al patrón de percepción acústica, se tiene una escala llamada *Escala de Mel*, la cual es una escala logarítmica que representa los niveles de la señal.

El humano no procesa frecuencias individuales independientemente, como lo sugiere el análisis acústico. En su lugar escucha grupos de frecuencias por lo cual es capaz de distinguirlas de ruidos alrededor, Pérez et al. (2013).

2.1.1.3. Modelo digital para la voz (Fuente-Filtro)

Una forma bastante simplificada pero efectiva para comprender el mecanismo para la producción de la voz es mediante el modelo fuente-filtro, propuesto originalmente por Muller en 1848. En dicho modelo, se supone que la fuente o señal de excitación es linealmente separable de las características del tracto vocal, que se representan por medio de un filtro. Por lo tanto, la señal de voz es la salida de este filtro en respuesta a la fuente de excitación, la cual puede representarse como un generador de un tren de impulsos cuasiperiódicos para los sonidos sonoros, o como un generador de ruido aleatorio (ruido blanco) para los sonidos sordos, Rowden (1992).

Puesto que el filtro debe tener las mismas características que el tracto vocal, no es posible mantener constantes los parámetros del filtro, los cuales tienen que ser modificados para que correspondan a las variaciones que suceden al hablar. Por lo tanto, el filtro tiene parámetros variantes en el tiempo. No obstante, en la práctica, esta tasa de cambio varía lentamente en el tiempo, por lo que los parámetros tienen que ser actualizados en intervalos de 5 a 30 ms.

La Figura 2.3 muestra un diagrama de bloques del modelo fuente-filtro. El modelo es relevante porque para cada tipo de excitación, un fonema se identifica principalmente por la forma del tracto vocal, por lo que su configuración puede ser estimada al identificar el filtrado realizado por este.

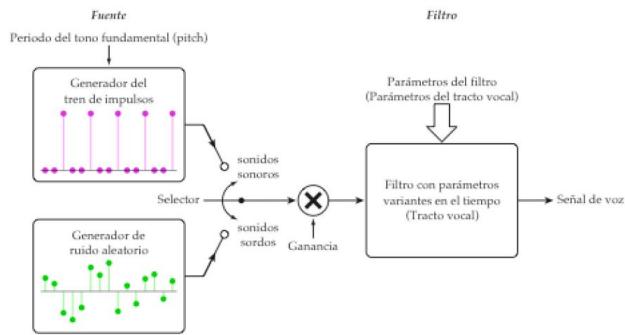


Figura 2.3: Modelo fuente-filtro para la producción de la voz.

Fuente: Rowden (1992)

2.1.1.4. Funcionamiento de un sistema de reconocimiento de voz

El reconocimiento de voz generalmente es utilizado como una interfaz entre humano y computador para algún software, tal es el caso, por ejemplo, el de los sistemas biométricos. Todo sistema de reconocimiento de voz debe cumplir las 3 tareas siguientes:

- **Preprocesamiento:** Convierte la entrada de voz a una forma que el reconocedor pueda procesar.
- **Reconocimiento:** Identifica lo que se dijo (traducción de señal a texto).
- **Comunicación:** Envía lo reconocido al sistema (hardware/software) que lo requiere.

En la Figura 2.4 se muestran los componentes de un sistema de reconocimiento de voz:



Figura 2.4: Diagrama de bloques del proceso general del reconocimiento de voz.

Fuente: Rama (2007)

Existe una comunicación bilateral en aplicaciones en las que la interfaz de voz está íntimamente relacionada al resto de la aplicación. Estas pueden guiar al reconocedor especificando las palabras o estructuras que el sistema puede utilizar. Otros sistemas tienen una comunicación unilateral.

Los procesos de preprocesamiento, reconocimiento y comunicación deberían ser invisibles al usuario de la interfaz. El usuario lo nota en el desempeño del sistema de manera indirecta como: certeza en el reconocimiento y velocidad. Estas características se utilizan para evaluar una interfaz de reconocimiento de voz.

Una vez visto el proceso general de un sistema de reconocimiento de voz, a continuación, veremos los módulos necesarios a detalle para el diseño y construcción de un sistema de reconocimiento de voz del locutor dependiente del texto.

Los diagramas de bloques que se muestran en las Figuras 2.5 y 2.6 ilustran, de forma general, los procesos que se realizan en el entrenamiento y reconocimiento de palabras aisladas, respectivamente.

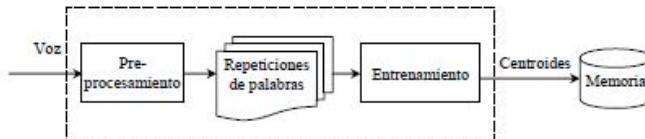


Figura 2.5: Diagrama de bloques para la etapa de entrenamiento.

Fuente: Navarrete (2013)

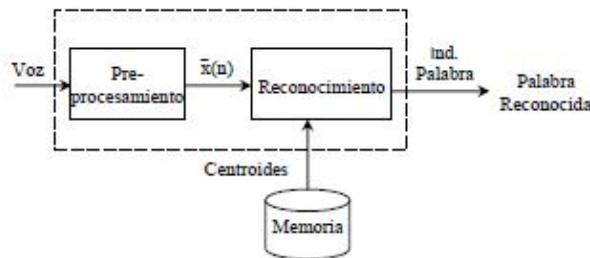


Figura 2.6: Diagrama de bloques para la etapa de reconocimiento.

Fuente: Navarrete (2013)

Si observamos las Figuras 2.5 y 2.6, existe un módulo común en ambas etapas, el preprocesamiento. Por esta razón, el preprocesamiento se manejará como un módulo independiente, para efecto del análisis.

En la Figura 2.5, durante la etapa de entrenamiento se capturan diferentes señales de voz, a la que denominaremos repeticiones de palabras. A cada repetición se le aplica preprocesamiento

para obtener una palabra delimitada, después se agrupan todas las palabras recortadas y con ellas se obtienen sus centroides correspondientes, para luego ser guardados en memoria (base de datos).

Como se observa en la Figura 2.6, en la etapa de reconocimiento se captura la palabra que se desea reconocer. A esta palabra, también se le aplica preprocesamiento, para recortarla. Utilizando los centroides, calculados durante el entrenamiento y almacenados en la memoria, se realizan las comparaciones necesarias, para efectuar el reconocimiento. El éxito del evento dependerá de ciertos parámetros estadísticos obtenidos mediante el análisis de una población de resultados.

2.1.2. Procesamiento o análisis de la señal de voz

La finalidad del procesamiento de la señal de voz es obtener una representación más útil o conveniente de la información contenida en esta. Como se mostró anteriormente, la señal de voz es una composición compleja de sonidos, pero mediante un análisis, se puede mostrar que está formada por la combinación de componentes a diferentes frecuencias, producidas por la excitación periódica o aperiódica de las resonancias en el tracto vocal, principalmente entre 80 y 8000 Hz, Rowden (1992).

Las técnicas de análisis se pueden clasificar en dos categorías: en el dominio del tiempo o en el dominio de la frecuencia. Los procedimientos en el dominio del tiempo, se denominan así, porque éstos tienen que ver directamente con la señal de voz. El atractivo de estas representaciones es la facilidad con la que se pueden implementar, no obstante, estas suministran un medio útil para

estimar otras características importantes de la señal. En contraste, los métodos en el dominio de la frecuencia involucran, ya sea de manera explícita o implícita, alguna forma de representación del espectro en frecuencia.

La mayoría de estas se basan en el modelo fuente-filtro, visto anteriormente, para la producción de la voz. Así que esencialmente, el análisis es el proceso en el cual, a partir de una señal de voz, se estiman los parámetros variantes en el tiempo del filtro, de tal manera que su objetivo principal es el estimar la respuesta en frecuencia del filtro que representa al tracto vocal.

A continuación, veremos los conceptos necesarios para la construcción de un reconocedor de voz, por cada una de las etapas que este tiene.

2.1.2.1. Preprocesamiento

En el preprocesamiento se recibe una señal de voz y se digitaliza determinando sus límites, esta señal puede contener ruido externo o no, si la contiene se aplicará un filtro de eliminación de ruido. Luego a la señal se le aplica un filtro de preénfasis, para realzar las frecuencias altas presentes en la voz. Después se divide la señal en tramas y a cada trama se le aplica una ventana para suavizar el espectro.

Es necesario aclarar que, cuando se inicializa el sistema, se determinan ciertos parámetros (umbrales) necesarios para la detección de inicio y fin de la señal de voz. Estos parámetros se cal-

culan recolectando muestras del ruido presente en el ambiente circundante. El módulo encargado de realizar este proceso se denomina ruido ambiental. Finalmente, con las tramas provenientes del ventaneo y con los umbrales originados por el ruido ambiental, se determina el inicio y fin de la señal de voz, tal como se muestra en la Figura 2.7.

Como ya se mencionó, el preprocesamiento es un módulo presente en las etapas de entrenamiento y reconocimiento. Sin embargo, existe una diferencia sutil, durante el entrenamiento cada palabra recortada se almacena, y en un archivo independiente se determinan los centroides representativos de esta, mientras que en el reconocimiento se utiliza directamente el archivo con los centroides para hacer las comparaciones. La Figura 2.7 muestra el diagrama de bloques para este módulo.

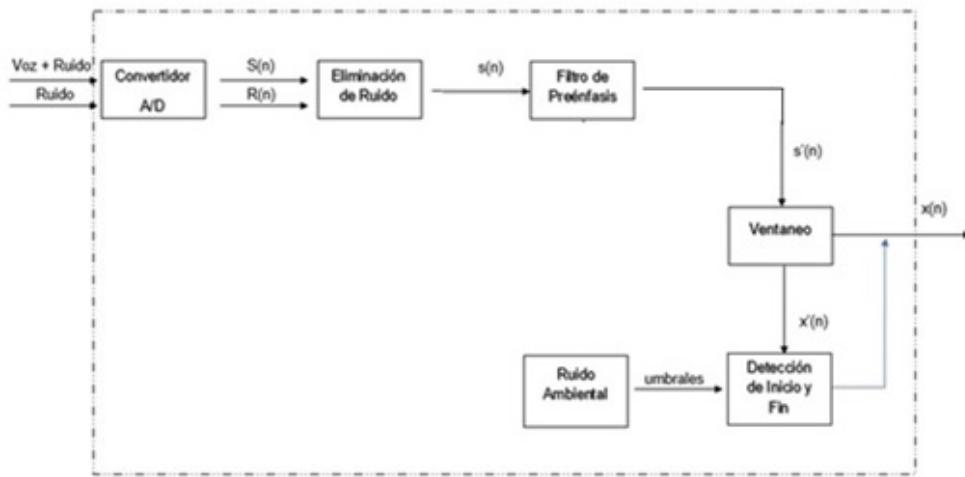


Figura 2.7: Diagrama de bloques para el módulo de preprocesamiento.

Fuente: Adaptación de Navarrete (2013)

El preprocesamiento consta de los siguientes módulos: convertidor A/D, eliminación de ruido, filtro de preénfasis, ventaneo o segmentación, detección de ruido ambiental y detección de inicio y fin de la señal de voz. Estos módulos se describen a continuación:

a) Digitalización de la señal de voz

Para poder manipular la señal de voz esta debe de ser transformada a una señal digital, así podrá ser utilizada por el software de programación, este proceso se lleva a cabo en etapas con consideraciones variables que hacen que la adquisición de la señal sea de forma correcta.

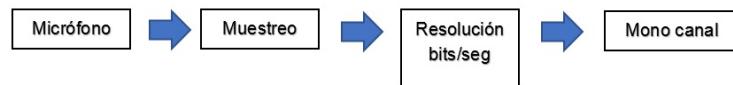


Figura 2.8: Diagrama de la adquisición de voz.

Fuente: Pérez et al. (2013)

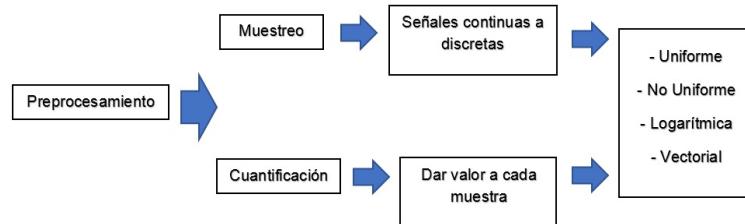


Figura 2.9: Diagrama de la digitalización de la señal de voz.

Fuente: Pérez et al. (2013)

- Obtención de información mediante un micrófono

El micrófono es un transductor electroacústico. Su función es la de transformar la presión acústica ejercida sobre su capsula por las ondas sonoras en energía eléctrica.

Como se dijo, el audio es un fenómeno analógico. Para grabar una señal de voz se hace la conversión de la señal analógica del micrófono en una señal digital por medio del conversor A/D en la tarjeta de sonido. Cuando un micrófono está operando las ondas de sonido hacen que vibre el elemento magnético del micrófono, causando una corriente eléctrica hacia la tarjeta de sonido, donde el conversor A/D básicamente graba los voltajes eléctricos en intervalos específicos.

Hay dos factores importantes durante este proceso. Primero está la taza de muestreo, es decir que tan seguido los valores de voltaje son grabados. Segundo, son los bits por segundo, en otras palabras, que tan exactamente los valores son grabados. Un tercero podría ser el número de canales (mono o estéreo), pero para las aplicaciones de reconocimiento de voz un canal (mono) es suficiente.

- Ancho de banda y muestreo

Según Becchetti and Prina (1999), el ancho de banda para las señales de voz es de aproximadamente unos 16000 Hz, pero la mayor parte de la energía se concentra en las frecuencias menores a 7000 Hz. No obstante, un ancho de banda de 3500 Hz es

suficiente para poder entender el mensaje contenido en la señal de voz.

El valor para la frecuencia de muestreo se determina mediante el teorema del muestreo de Nyquist, que establece que, si la componente en frecuencia más alta presente en la señal de voz es f_{max} , entonces para poder reconstruir apropiadamente la señal a partir de las muestras obtenidas, la frecuencia de muestreo f_s debe satisfacer la siguiente condición:

$$f_s \geq 2f_{max} \quad f_N \doteq f_{max} \quad f_s \geq 2f_N$$

En donde a f_N se le conoce como la frecuencia de Nyquist. Por lo tanto, el periodo de muestreo T_s es igual al inverso de la frecuencia de muestreo f_s :

$$T_s \doteq \frac{1}{f_s}$$

Si se emplea una frecuencia de muestreo menor que el doble de la frecuencia de Nyquist, entonces sucede un fenómeno conocido como *aliasing*, para el cual una señal senoidal a cierta frecuencia pudiera parecer otra señal de menor frecuencia, como si esta fuera un alias de la original. Si ocurriera un aliasing en la señal de voz, se insertarían algunas componentes en frecuencia no deseadas que la distorsionarían.

Puesto que el ancho de banda de una señal de voz se reduce durante la conversión A/D (analógico/digital), para su procesamiento se suelen utilizar básicamente dos frecuencias de muestreo, la primera de $f_s = 16000 \text{ Hz}$ para obtener un ancho de banda o una frecuencia de Nyquist de $f_N = 8000 \text{ Hz}$, y la segunda de $f_s = 8000 \text{ Hz}$ que corresponde a un ancho de banda o una frecuencia de Nyquist de $f_N = 4000 \text{ Hz}$, Becchetti and Prina (1999).

En cualquier caso, el muestreo periódico de una señal analógica de voz $x(t)$, proporciona una representación de esa señal mediante una secuencia de muestras $x(n)$, que se pueden obtener mediante la siguiente relación:

$$x(n) \doteq x(t)|_{t=nT_s} \quad n \doteq 0, 1, 2, 3, \dots, N_x - 1$$

En donde N_x es el número total de muestras presentes en la señal de voz. La Figura 2.10 ilustra el muestreo de una señal analógica de voz a una frecuencia de muestreo de $f_s = 8000 \text{ Hz}$, así que cada una de las $N_x = 21$ muestras se toman cada $T_s = 0.125 \text{ ms}$.

La señal discreta de voz $x(n)$ esta indizada por la variable n , que da una normalización de la señal, es decir, que ésta no contiene ninguna información explícita acerca de la frecuencia de muestreo. Aunque las muestras definen una señal discreta, se acostumbra unir éstas para que formen una curva continua para representar la señal de voz.

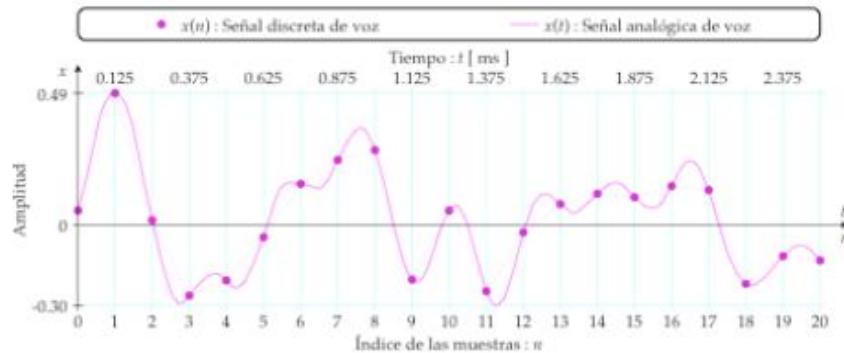


Figura 2.10: Muestreo de una señal de voz.

Fuente: Becchetti and Prina (1999)

- Resolución o cuantificación

El proceso de cuantificación es uno de los pasos que se siguen para lograr la digitalización de una señal analógica. Este proceso convierte una sucesión de muestras de amplitud continua en una sucesión de valores discretos preestablecidos según el código utilizado, Schroeder (1998).

Durante el proceso de cuantificación se mide el nivel de tensión de cada una de las muestras, obtenidas en el proceso de muestreo, y se les atribuye un valor finito de amplitud, seleccionado por aproximación dentro de un margen de niveles previamente fijado.

En la Figura 2.11 se muestra un ejemplo utilizando 2 bits donde se pueden representar 4 valores diferentes y con 4 bits 16 valores.

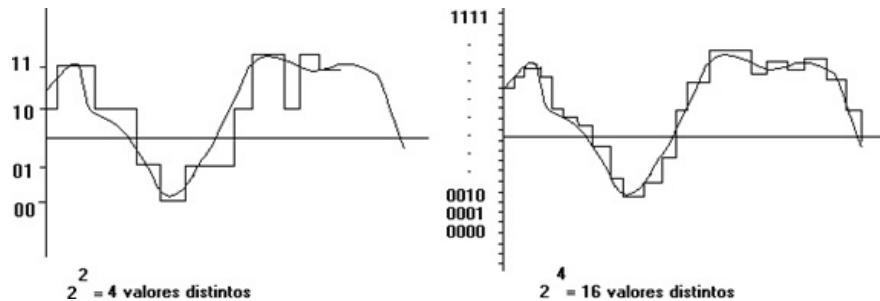


Figura 2.11: Señal cuantificada con 2 y 4 bits respectivamente.

Fuente: Pérez et al. (2013)

Esto se puede ver como la resolución:

- El error o diferencia entre la señal original y la reconstruida se percibe como ruido.
- Por lo tanto, a mayor resolución menor ruido como consecuencia.
- La resolución (número de bits por muestra) se describe generalmente en términos de la relación señal a ruido (Signal to Noise Ratio o SNR).
- A mayor SNR es mayor la fidelidad de la señal digitalizada.
- SNR es aproximadamente 2^B (B=bits/muestra)
- Es independiente de la frecuencia de muestreo.

Existen diferentes técnicas de cuantificación, a continuación, explicaremos cada una de ellas:

- Cuantificación uniforme

En los cuantificadores uniformes o lineales la distancia entre los niveles de recons-

trucción es siempre la misma, la mayoría usan un número de niveles que es una potencia de 2. No hacen ninguna suposición acerca de la señal a cuantificar, de allí que no proporcionen los mejores resultados. Su ventaja es que son más fáciles y económicos al implementarlos.

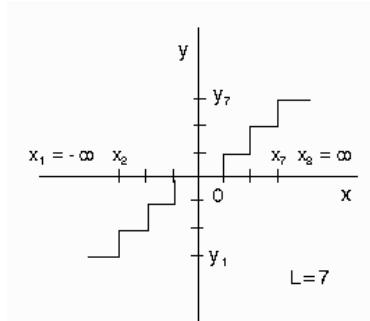


Figura 2.12: Señal cuantificada uniformemente.

Fuente: Rama (2007)

- Cuantificación no uniforme

El problema de la cuantificación uniforme es que conforme aumenta la amplitud de la señal, también aumenta el error. Si conocemos la función de la distribución de probabilidad, podemos ajustar los niveles de reconstrucción a la distribución de forma que se minimice el error cuadrático medio. Esto significa que la mayoría de los niveles de reconstrucción se den en la vecindad de las entradas más frecuentes, y consecuentemente se minimice el error.

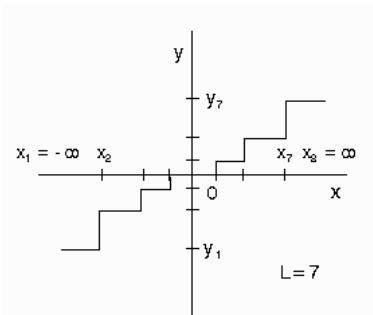


Figura 2.13: Señal cuantificada no uniformemente.

Fuente: Rama (2007)

- Cuantificación logarítmica

Las señales de voz pueden tener un rango dinámico mayor a los 60 dBC, por lo que para conseguir una alta calidad de voz se deben usar un elevado número de niveles de reconstrucción. Sin embargo, interesa que la resolución sea mayor en las partes de la señal de menor amplitud que en las de mayor amplitud. Por tanto, en la cuantificación lineal se desperdician niveles de reconstrucción y ancho de banda. Esto se puede mejorar incrementando la distancia entre los niveles de reconstrucción conforme aumenta la amplitud de la señal. Para conseguir esto se hace pasar la señal por un compresor logarítmico antes de la cuantificación. Esta señal comprimida puede ser cuantificada uniformemente. Luego a la salida del sistema la señal pasa por un expansor. A esta técnica se le llama compresión, Harrington (1999).

- Cuantificación vectorial

Este método cuantifica los datos en bloques de N muestras, cada bloque se trata

como un vector N-dimensional. La cuantificación vectorial ofrece mejores resultados que la cuantificación escalar, sin embargo, es más sensible a los errores de transmisión y lleva consigo una mayor complejidad computacional.

La cuantificación más usada, es de 8 bits, mínimo requerido para una calidad baja, puede mejorarse su SNR con una técnica no lineal de cuantificación, pero se obtienen excelentes resultados aumentando la cuantificación a 16 bits, Lee (1989).

b) Eliminación de ruido y normalización

El ruido es como una señal no deseada, es un elemento independiente de la señal, pero como consecuencia puede afectar la calidad de la misma. Para cancelar el ruido y mejorar la calidad se han desarrollado diversas técnicas de procesamiento de señal, dentro de los cuales se puede mencionar los filtros adaptativos, que es una técnica de las varias que existen hoy en día y es importante considerar los tipos de ruidos existentes en los canales de comunicación, Ebla and Inca (2016).

- Ruido aditivo: Se puede considerar todo aquel ruido procedente de distintas fuentes que coexistan en el mismo entorno acústico.
- Señales interferentes: En el caso de señales de voz, se considera señal interferente a toda aquella que proceda de otros locutores, que no sean objeto de interés.
- Reverberación: Producida por la propagación multirayectoria que se dan en los entor-

nos acústicos cerrados o semicerrados. No se trata exactamente de ruido, sino de una forma de distorsión.

- Eco: Producida generalmente por el acoplamiento entre los micrófonos y los altavoces.
Igual que en el caso anterior se trata de una forma de distorsión.
 - Diafonía: Sonido indeseado que se origina en el receptor, y es la consecuencia del acoplamiento de este canal con otro, permitiendo el paso de señales del mismo origen.
- Cancelación de ruido aditivo

En los sistemas de cancelación de ruido podemos encontrar dos tipos de filtros, los filtros digitales y los filtros adaptativos. Los filtros digitales solo pueden ser usados en ambientes de ruido estacionario, ya que es necesario conocer a priori las estadísticas del ruido, una solución a esto son los filtros adaptativos.

El propósito de los filtros digitales es separar señales que se han combinado y restaurar señales que se han distorsionado de alguna manera. La separación de señales se requiere cuando una señal se ha contaminado con una interferencia, ruido, u otras señales. Los filtros Weiner y filtros Kalman son ejemplos de filtros digitales, ver la Figura 2.14, Shubhra (2017).

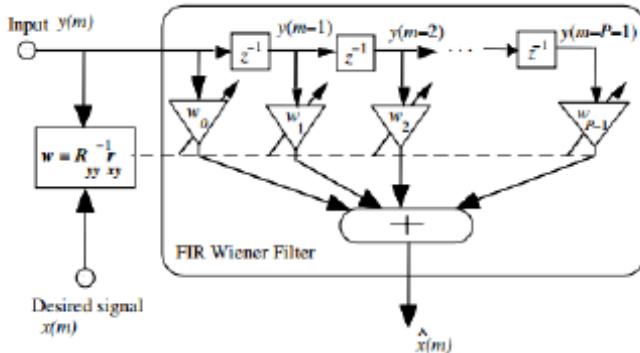


Figura 2.14: Ilustración de la estructura de un filtro Wiener.

Fuente: Shubhra (2017)

Por otro lado, los filtros adaptativos confían en el uso de una fuente de ruido para eliminar el ruido de una señal recibida. Ordinariamente, es imprudente quitar ruido de una señal recibida, debido a que semejante operación pudiera producir resultados desastrosos, causando un incremento en la potencia promedio del ruido en la salida. Sin embargo, cuando se hacen previsiones apropiadas en el control del filtrado de la eliminación del ruido, es posible realizar un sistema superior de operación comparado al filtrado directo de la señal recibida.

Básicamente, un cancelador de ruido adaptativo es un arreglo de doble entrada de lazo cerrado del sistema de realimentación como se ilustra en la Figura 2.15, se derivan las dos entradas del sistema de un par de sensores: un sensor primario y un sensor de referencia (auxiliar). Específicamente, nosotros tenemos lo siguiente:

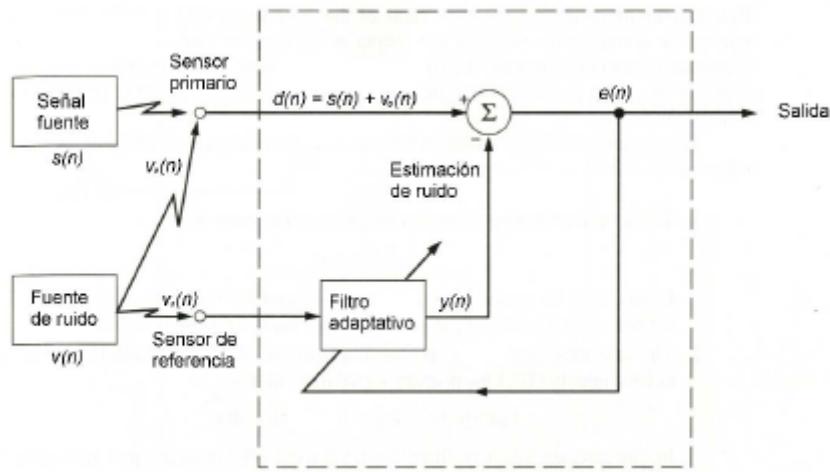


Figura 2.15: Cancelador de ruido adaptativo.

Fuente: Zelaya (2004)

1. El sensor primario recibe una señal que lleva información $s(n)$ adulterada por un ruido aditivo $v_0(n)$, como se muestra en la Ecuación (2.1).

$$d(n) = s(n) + v_0(n) \quad (2.1)$$

La señal $s(n)$ y el ruido $v_0(n)$ están sin correlación una de la otra, esto es:

$$E[s(n)v_0(n - k)] = 0 \quad \text{para todo } k \quad (2.2)$$

Donde $s(n)$ y $v_0(n)$ se suponen que son valores reales.

2. El sensor de referencia recibe un ruido $v_1(n)$ que no tiene correlación con la señal $s(n)$ pero si tiene correlación con el ruido $v_0(n)$, el cual es obtenido con el sensor

primario junto a la señal que se desea recuperar; esto es:

$$E[s(n)v_1(n - k)] = 0 \quad \text{para todo } k \quad (2.3)$$

$$E[v_0(n)v_1(n - k)] = p(k) \quad (2.4)$$

Donde, la señal es un valor real y $p(k)$ es una correlación cruzada desconocida para el retardo k . La señal de referencia $v_1(n)$ es procesada por un filtro adaptativo para producir la salida:

$$y(n) = \sum_{k=0}^{M-1} \hat{w}_k(n)v_1(n - k) \quad (2.5)$$

Donde, los $\hat{w}_k(n)$ son los valores de coeficientes (reales) ajustables del filtro adaptativo. La salida del filtro $y(n)$ es restada de la señal principal $d(n)$, conocida como la respuesta deseada para el filtro adaptativo. La señal de error se define por:

$$e(n) = d(n) - y(n) \quad (2.6)$$

Así, sustituyendo la Ecuación (2.1) en la Ecuación (2.6), se obtiene:

$$e(n) = s(n) + v_0(n) - y(n) \quad (2.7)$$

La señal de error es la que se utiliza para ajustar los valores de coeficientes del filtro adaptativo y el lazo de control alrededor de las operaciones de filtrado y sustracción están relacionados. Note que la señal que lleva la información es cierta parte de la señal $e(n)$, como se indica en la Ecuación (2.7).

La señal de error $e(n)$ constituye la salida total del sistema. De la Ecuación (2.7) se puede ver, que la componente de ruido en la salida del sistema es $v_0(n) - y(n)$.

Ahora comienza su labor el filtro adaptativo para minimizar el error cuadrático medio de la señal $e(n)$. La señal esencial $s(n)$ que lleva la información, no es afectada por el cancelador de ruido adaptativo.

Así que, minimizando el error cuadrático medio de la señal $e(n)$ es equivalente a minimizar el error cuadrático medio del ruido de salida $v_0(n) - y(n)$. Con la señal $s(n)$ que permanece esencialmente constante, esto lleva a que la minimización del error cuadrático medio de la señal, es ciertamente la misma como la maximización de la razón señal a ruido de la salida del sistema.

El uso efectivo del cancelador de ruido adaptativo requiere que se coloque el sensor de referencia en el campo de ruido del sensor primario.

- Estructura de los filtros adaptativos

El algoritmo adaptativo debe disponer del error $e(n)$ para actualizar los coeficientes, ya que $e(n)$ permite definir las mejoras del filtro y determinar la forma en que han de modificarse dichos coeficientes. La eficiencia de un filtro adaptativo lineal depende de una serie de factores, como el tipo de filtro (IIR o FIR), la estructura del mismo (transversal, de celosía o sistólico), o la función de costo usada como criterio de adaptación

(error cuadrático medio, mínimo error cuadrático, etc.).

Nuestro estudio se va a centrar en el filtro FIR (Respuesta Finita al Impulso) por varias razones:

- El error cuadrático medio para un filtro transversal es una función cuadrática de los pesos del filtro. La superficie de error es un paraboloide con sólo un mínimo, y por ello, la búsqueda del error cuadrático medio mínimo es relativamente sencilla.
- Dado que los coeficientes del filtro son limitados, se puede controlar fácilmente la estabilidad del filtro.
- Existen algoritmos para la actualización de los coeficientes que con filtros FIR son mucho más simples y eficientes.
- Las prestaciones de estos algoritmos son perfectamente conocidas en términos de convergencia y estabilidad.

De los tres tipos de estructuras de filtro que se distinguen en el contexto de un filtro adaptativo lineal, vamos a conceder toda la atención al tipo transversal.

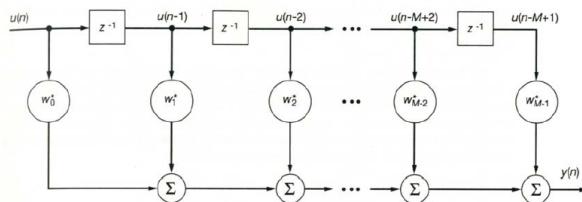


Figura 2.16: Filtro transversal.

Fuente: Zelaya (2004)

Los filtros transversales, también llamados filtros directos de pesos retardados, consisten en tres elementos básicos, como se describió en la Figura 2.16, primero los elementos de unidad de retardo, segundo el multiplicador, y tercero la sumatoria. El número de elementos de retardo usados en el filtro, mostrado como $M - 1$ en la Figura 2.16, determinan el filtro FIR, estos normalmente están referidos al orden del filtro. Los elementos de retardo son identificados por el operador de la unidad de retraso z^{-1} .

En particular, cuando z^{-1} opera en la entrada $u(n)$, la salida resultante es $u(n - 1)$. El papel de cada multiplicador en el filtro es realizar el producto del valor de entrada por un coeficiente del filtro. Así un multiplicador conectado k veces a entradas retardadas $u(n - k)$ produce la versión del escalar de que es el producto interno, $w_k^* u(n - k)$, donde w_k va desde $k = 1$ a M . El asterisco denota la conjugación compleja que asume la entrada y por consiguiente también los pesos actualizables son valores complejos.

El papel combinado de las sumatorias en el filtro es sumar los resultados de los productos individuales y producir una salida total del filtro. Para el filtro transversal descrito en la Figura 2.16, la salida está dada de la siguiente manera:

$$y(n) = \sum_{k=0}^{M-1} W_k \cdot u(n - k) \quad (2.8)$$

La ecuación es llamada sumatoria de convolución finita, en el sentido que la respuesta convoluciona el impulso de duración finita del filtro, w_n^* , con la entrada $u(n)$ del filtro,

luego cada producto individual se suma para dar como resultado $y(n)$.

La estructura transversal es la más sencilla de implementar, conduciendo a algoritmos igualmente sencillos. La estructura de celosía, presenta mejores propiedades, pues ofrece mayor robustez frente a errores de redondeo y una mayor eficiencia computacional.

Sin embargo, aumenta la complejidad de los algoritmos. Por lo tanto, en el presente trabajo de tesis se adopta la primera estructura debido a su sencillez, pero aun así con buen desempeño.

- Aplicaciones de filtros adaptivos

La habilidad de un filtro adaptativo es de operar de manera satisfactoria en un ambiente desconocido rastreando las variaciones estadísticas en el tiempo de una entrada, esto hace que los filtros adaptativos sean poderosos dispositivos para aplicaciones de procesado de señales y control. Es por eso que los filtros adaptativos han sido abundantemente aplicados en campos diversos tales como las comunicaciones, radar, sonar, sismología y la ingeniería biomédica. Aunque estas aplicaciones son de hecho bastante diferentes en naturaleza, no obstante, tienen un rasgo común básico: un vector de entrada y una respuesta deseada que se usan para calcular el error de estimación, que a su vez se usa para controlar los valores de un conjunto de coeficientes ajustables.

Los coeficientes ajustables pueden tomar la forma de pesos regulables, coeficientes de reflexión, parámetros de rotación o pesos sinápticos, dependiendo de la estructura del filtro empleada. Sin embargo, la diferencia esencial entre las varias aplicaciones de filtrado adaptativo comienza con la manera como se extrae la respuesta deseada. En este contexto, podemos distinguir cuatro clases básicas de aplicaciones de filtros adaptativos que ascienden a doce, como se describen en la Tabla 2.3. En la sección anterior se describió el caso particular del cancelador de ruido, que es la que usaremos en esta tesis.

Tabla 2.3: Aplicaciones de filtros adaptativos.

Clase de filtro adaptativo	Aplicación
I. Identificación	Identificación de Sistemas Modelado de Capas Subterráneas
II. Modelado Inverso	Deconvolución Predictiva Ecualización Adaptativa Ecualización Ciega
III. Predicción	Codificación por Predicción Lineal (LPC) Codificación Diferencial Adaptativa (ADPCM) Análisis Espectral Autorregresivo Detección de Señal
IV. Cancelación de Interferencias	Cancelación Adaptativa de Ruido Cancelación de Eco Formas de haz Adaptativas

Fuente: Zelaya (2004)

- Filtros adaptativos

Tal y como vimos en la sección anterior, los filtros adaptativos son sistemas dinámicos

lineales con estructura y parámetros variables o adaptables y tiene la propiedad para modificar los valores de sus parámetros, es decir su función de transferencia, durante el procesamiento de la señal de entrada genera una señal de salida que este sin componentes no deseados, ruido, degradación y también las señales de la interferencia.

La Figura 2.17 muestra el concepto básico de un filtro adaptativo cuyo objetivo principal es filtrar la señal de entrada, $x(n)$, de tal manera que se empareje a la señal deseada, el $d(n)$. La señal deseada, $d(n)$, se substrae de la señal filtrada, $y(n)$, para producir el error de la señal que a su vez maneja un algoritmo adaptivo que genera los coeficientes del filtro de una manera que minimiza el error de la señal. La adaptación ajusta las características del filtro a través de una interacción con el ambiente para alcanzar los valores deseados.

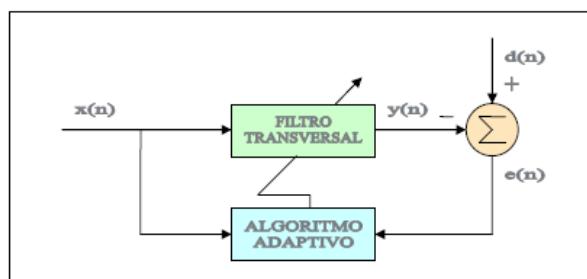


Figura 2.17: Modelo de un filtro adaptativo.

Fuente: Simón (2011)

El filtro adaptativo ajusta sus coeficientes con un objetivo, garantizar la posible convergencia más rápida a los parámetros óptimos del punto de vista del criterio adoptado. La

mayoría de algoritmos adaptivos significa modificaciones de los procedimientos iterativos normales para la solución del problema de minimización de la función de criterio en tiempo real. El más común de filtros adaptativos es el filtro transversal y es usado en, el algoritmo LMS (Mínimo Error Cuadrado Medio) y el algoritmo RLS (Mínimo Error Cuadrado Recursivo). Para lograr un buen desempeño del filtro adaptativo requiere el uso del mejor algoritmo adaptivo con baja complejidad computacional y un rango de convergencia rápido. A continuación, veremos algunos algoritmos adaptivos, pero nos centraremos más en los algoritmos LMS y NLMS.

- Algoritmo Least Mean Square (LMS)

Un acercamiento muy sincero en el cancelamiento del ruido es el uso del algoritmo LMS desarrollado por Windrow y Hoff. Este algoritmo usa un descenso de la gradiente para estimar una variación del tiempo en la señal. El método de descenso de la gradiente encuentra un mínimo (si este existe), tomando los pasos en la dirección negativa de la gradiente y hace esto para ir ajustando los coeficientes del filtro para minimizar el error. La gradiente es un operador y se aplica para encontrar la divergencia de una función, que es el error con respecto en este caso al número de coeficientes.

El algoritmo LMS se ha aceptado por varios investigadores para la aplicación en

el hardware debido a su estructura simple. Para llevarlo a cabo, las modificaciones tienen que ser hechas al algoritmo LMS original porque los bucles recursivos en la fórmula de actualización del filtro le impide ser un flujo lineal.

La Ecuación 2.9 muestra el detalle del algoritmo LMS, la evaluación de pesos:

$$w_i(n+1) = w_i(n) + \mu^* e(n)^* x(n-i) \quad (2.9)$$

Salida filtrada:

$$y(n) = \sum_{k=0}^{M-1} w_i(n)^* x(n-i) \quad (2.10)$$

La estimación del error, donde $e(n)$ es la salida deseada:

$$e(n) = d(n) - y(n) \quad (2.11)$$

Donde la salida del filtro adaptivo $y(n)$ y el error de la señal $e(n)$ son obtenidos por las Ecuaciones (2.10) y (2.11), respectivamente. En estas ecuaciones $x(n)$ es el vector de la señal de entrada, y $w(n)$ es el vector de pesos del filtro adaptivo.

De estas ecuaciones, en cada iteración la información de la mayoría de los recientes valores son requeridos y el procedimiento reiterativo se inicia con una suposición inicial w_0 . El μ es el tamaño del paso de adaptación que depende del poder de la densidad espectral de la entrada de referencia $x(n)$ y $M - 1$ es el orden del

filtro y controla la estabilidad y velocidad de la convergencia del algoritmo LMS.

- Algoritmo Normalized Least Mean Square (NLMS)

La debilidad principal del LMS del tipo convencional radica en su complejidad seleccionando un valor conveniente para el parámetro de tamaño de paso que garantiza la estabilidad. Ante este inconveniente, fue propuesto el NLMS, controlando el factor de la convergencia del LMS a través de la modificación en un parámetro de tamaño de paso de tiempo variante.

El NLMS emplea un parámetro de tamaño de paso inconstante pensado a minimizar el error de salida instantáneamente el cual converge más rápidamente que el LMS convencional. El algoritmo LMS convencional experimenta en la gradiente problemas de amplificación de ruido cuando el factor de la convergencia μ es grande. La corrección se aplicó al vector de pesos $w(n)$ al $n + 1$ de la iteración, es normalizada con respecto al cuadrado de la norma euclíadiana del vector de entrada $x(n)$ a la iteración n .

Nosotros podemos expresar el algoritmo NLMS como un algoritmo de tamaño de paso de tiempo variante, calculando el factor de convergencia μ como en la Ecuación (2.12).

$$\mu(n) = \frac{\beta}{c + \|x(n)\|^2} \quad (2.12)$$

Dónde β es la constante de adaptación del NLMS, que optimiza el rango de convergencia del algoritmo y debe satisfacer la condición $0 < \beta < 2$, y c es el término constante para la normalización y siempre es menor que 1. Los pesos del filtro se actualizan por la Ecuación (2.13)

$$w(n+1) = w(n) + \frac{\beta}{c + \|x(n)\|^2} e(n)x(n) \quad (2.13)$$

En comparación al LMS, el NLMS tiene tamaño de paso variante que hace al NLMS para converger más rápidamente. Para sacar mejores aplicaciones se han desarrollado varias variantes del LMS, ver la Tabla 2.4.

Tabla 2.4: Variantes del algoritmo LMS.

S. No	Algorithm type	Recursion (Weighted)	Reference
1.	Conventional LMS	$w_i(n+1) = w_i(n) + \mu^* e(n)^* x(n-i)$	[45], [48]
2.	NLMS	$w(n+1) = w(n) + \frac{a}{c + \ x(n)\ ^2} e(n)x(n)$	[48-49]
3.	(MN-LMS)	$W(n+1) = W(n) + \beta \frac{X(n)}{c + \ X(n)\ ^2} \mu e(n) \quad \text{Where, } 0 < \beta < 2$	[50-51]
4.	Leaky LMS	$W(n+1) = (1 - \mu\gamma)W(n) + X(n)\mu e(n) \quad \text{Where, leaky coefficient } \gamma, 0 < \gamma << 1$ $0 < \mu < (\gamma + \lambda_{max})$	[52-54]
5.	(B-LMS)	$W((k+1)L) = W(kL) + \mu \sum_{l=0}^{L-1} e(kL+l) X(kL+l)$ Where, $l = 0, 1, 2, \dots, L-1$	[55-57]
6.	(SE-LMS)	$W(n+1) = W(n) + X(n)\mu sgn[e(n)]$ Where, $sgn(\cdot) = \text{signum function}$ $sgn[e(n)] = \begin{cases} 1 & \text{for } e(n) > 0 \\ 0 & \text{for } e(n) = 0 \\ -1 & \text{for } e(n) < 0 \end{cases}$	[58-59]
7.	(SD-LMS)	$W(n+1) = W(n) + sgn[X(n)]\mu e(n) \quad \text{Where, } sgn(\cdot) = \text{signum function}$	[60-62]
8.	(SDN-LMS)	$w_k(n+1) = w_k(n) + \frac{\mu}{ x(n-k) } e(n)x(n-k) \quad \text{Where, } sgn(\cdot) = \text{signum function}$	[63]
9.	(SS-LMS)	$W(n+1) = W(n) + sgn[X(n)]\mu sgn[e(n)] \quad \text{Where, } sgn(\cdot) = \text{signum function}$	[64-65]
10.	(VS-LMS)	$w_k(n+1) = w_k(n) + \mu_k e(n)x(n-k) \quad \text{Where, } \mu_{min} < \mu < \mu_{max}$	[66-69]
11.	(SS-LMS-LT)	$W(n+1) = (1 - \mu\gamma)W(n) + sgn[X(n)]\mu sgn[e(n)]$	[89-90]
12.	(FX-LMS)	$W(n+1) = W(n) + X^{(n)}\mu e(n) \quad \text{Where, } X^{(n)} = \tilde{s}(n)X(n)$	[70-74]
13.	(FRS-LMS)	$W(n+1) = [I - \mu F]W(n) + X(n)\mu e(n) \quad \text{Where, } F = \square F_0 \text{ and } \square \text{ is constant.}$	[75-77]
14.	(H-LMS)	$W(n+1) = W(n) + X(n)\mu e(n) \quad \text{for, } 0 \leq n \leq p$ $X(n) = \frac{X(n)\mu(n)e(n)}{E[X(n)X^T(n)]} \quad \text{for, } n \geq p+1$	[78-81]

Fuente: Shubhra (2017)

Además del LMS, NLMS y sus variantes, existen otros algoritmos adaptivos como el algoritmo RLS (Mínimo Error Cuadrado Recursivo), el algoritmo CMA (Adaptivo de Módulo Constante), el algoritmo DMI (Adaptivo de Inversión de Matriz Directa), los cuales pueden verse en (Shubhra, 2017).

- Convergencia

Los coeficientes del filtro adaptativo se modifican constantemente hasta lograr que el error sea mínimo, es decir el algoritmo alcance la convergencia, dicho de otra manera, que la entrada se aproxime a la respuesta deseada. Para precisar lo que sucede, supongamos que el orden del filtro es $M = 2$, es decir los coeficientes son de la forma $w(n) = [w_1(n) w_2(n)]$, siendo ξ el error donde ξ_{min} es el mínimo.

Entonces el sistema se puede graficar en un espacio de tres dimensiones como se puede apreciar en la Figura 2.18. Cuando empieza el algoritmo, el error es grande y a medida que el tiempo transcurre, los coeficientes van descendiendo por la curva a una velocidad relativa a μ hasta llegar a ξ_{min} . Pero se debe tener en cuenta que este proceso no sucede de una manera trunca, sino similar a una pelota deslizándose por la pared interna cuando llega al fondo, pasa de largo y vuelve a subir, pero con menor impulso y luego vuelve a bajar y así sucesivamente hasta detenerse en el fondo. El tiempo que se demore en detenerse en el fondo es el tiempo de convergencia y w_0 es el punto de

convergencia.

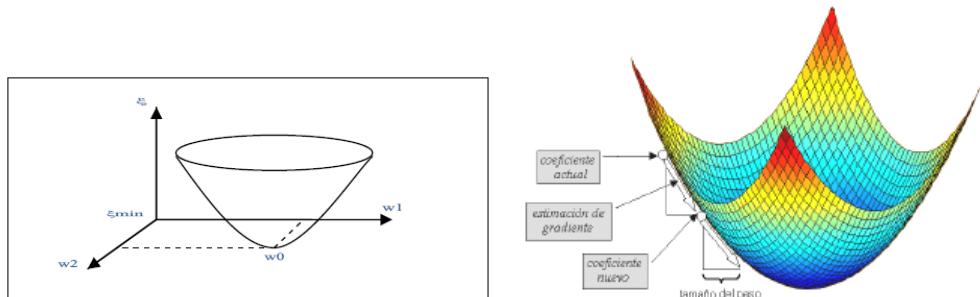


Figura 2.18: Convergencia y error mínimo.

Fuente: Simón (2011)

En (Simón, 2011) se realizaron unas pruebas para obtener el mejor número de orden M y tamaño de paso μ para el algoritmo LMS y el mejor número de orden M , la constante de adaptación β y el término constante para la normalización c para el algoritmo NLMS. Estas pruebas se realizaron con un micrófono primario ubicado a 2 metros de la fuente de ruido (equipo de sonido) y un segundo micrófono ubicado a 50 cm de la fuente de ruido. Se concluyó que cuando se utiliza un tamaño de paso grande ($\mu > 0,03$) el error en un determinado momento se eleva, el valor de W se torna inestable, por lo cual se debe elegir un valor menor.

En la Figura 2.19 se puede apreciar como la energía se va reduciendo desde $M = 32$ hasta $M = 128$, pero luego vuelve a incrementarse como se dijo anteriormente. Entonces se puede concluir que el mejor valor de M bordea el valor 128.

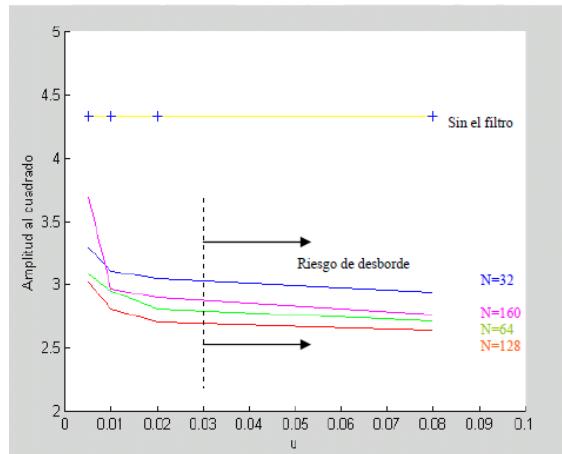


Figura 2.19: Cuadrado promedio para diferentes valores de M y μ para el LMS.

Fuente: Simón (2011)

El error varía poco para un tamaño de paso entre 0.02 y 0.08 además debe ser menor a 0.03 para evitar el desborde, por lo que se eligió el valor de 0.02 que asegura la estabilidad del filtro y mantiene el error a un nivel aceptable, ver la Figura 2.20.

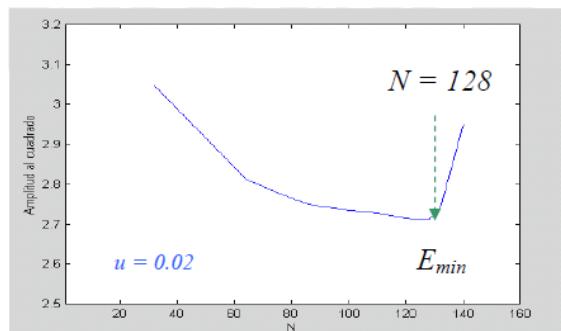


Figura 2.20: Valor para el número de orden M con $\mu=0.02$.

Fuente: Simón (2011)

En la Figura 2.21 se puede apreciar que se obtuvo una buena eficiencia hasta 81dBC de ruido a 2m de distancia de los parlantes. Para el algoritmo NLMS se propuso los valores de $M = 128$ al igual que el del LMS, con $\beta = 0,25$ y $c = 0,0001$.

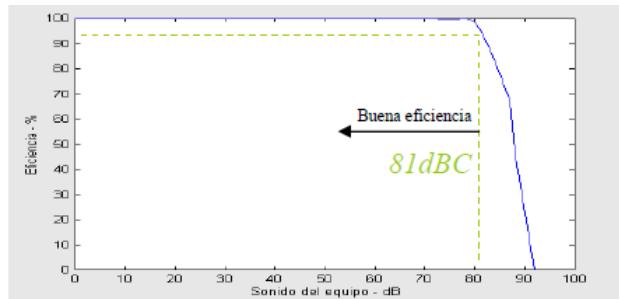


Figura 2.21: Curva de eficiencia vs. sonido del equipo.

Fuente: Simón (2011)

Finalmente, esta etapa de eliminación de ruido estará conformada por dos submódulos, el primero será la eliminación de ruido donde evaluaremos los algoritmos LMS Y NLMS para luego elegir el que tenga un mejor comportamiento y el segundo es la normalización de la señal después de haber aplicado el filtro adaptativo, debido a que puede haber un aumento en el espectro de potencia. El proceso se puede ver en la Figura 2.22.

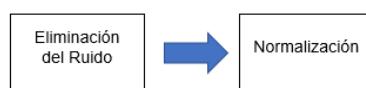


Figura 2.22: Diagrama de eliminación de ruido y normalización.

Fuente: (Pérez et al., 2013)

En Mathematics (2015) se muestra la formula para la normalización de la señal:

$[c, d]$ = rango de entrada mínimo y máximo.

$[m, n]$ = rango de salida mínimo y máximo.

Donde:

$$A = (m - n)/(c - d) \quad (2.14)$$

$$B = m - A * c \quad (2.15)$$

Entonces para la señal de entrada $E(n)$ obtenemos la salida normalizada $S(n)$:

$$S(n) = A^* E(n) + B \quad (2.16)$$

c) Detección de inicio y fin de la señal de voz

El problema de la localización del inicio y final de una muestra de voz en un fondo acústico de silencio es de vital importancia en muchas áreas del procesamiento de voz. En particular, el problema del reconocimiento de palabras está basado en la suposición de que se puede hallar la región de la muestra donde hay voz, para ser reconocida.

El poder determinar cuál es el inicio y el final de una palabra proporciona ciertas ventajas como, procesar menor cantidad de información, comparar únicamente los patrones de información y evitar confusiones a causa del ruido o señales de fondo.

La tarea de separar la voz del espacio de silencio no es una tarea tan simple como puede

parecer, esto puede suceder sólo en ambientes acústicos con tasa extremadamente alta de señal a ruido, por ejemplo, las cámaras a prueba de eco o de ruido, donde son hechas las grabaciones de alta calidad. Para dichos ambientes con alta relación señal a ruido, la energía de los sonidos de voz de más bajo nivel (fricativos débiles) excede la energía del ruido del fondo, por ende, una simple medición de la energía es suficiente para realizar las mediciones.

Sin embargo, dichas condiciones ideales de grabación no son prácticas para las aplicaciones del mundo real. Por ende, la simple medición de la energía se convierte en una condición necesaria más no suficiente para separar los fricativos débiles del fondo de silencio.

Algunos de los problemas que se presentan en la detección son:

- Silencios contenidos dentro de las palabras que tienen fonemas plosivos (*ej. /t/, /p/, /k/*) que pueden confundirse con un falso principio o fin.
- Presencia de espurias de ruido que se pueden confundir con la señal.
- Los fonemas fricativos (*ej. /f/, /th/, /h/, etc.*), ya que tienen baja energía.
- Sonidos cortos (*ej. /t/, /p/, /k/*).
- Detección de fonemas nasales al final de la palabra.
- Respiraciones del locutor, que pueden confundirse por su duración.

- Los micrófonos tienen resonancia después de pronunciar una palabra (sobre todo en vocales).
- Los niveles de ruido pueden confundirse con la señal de voz.

En este trabajo de tesis se evaluará el algoritmo por función de energía y los algoritmos propuestos por Rabiner y Sambur, la expuesta en (UNAM, 2015) la denominaremos *Tipo 1* y la de (Rabiner and Sambur, 1975) la denominaremos *Tipo 2*, para la localización de los puntos de inicio y final de grabación de voz. Dicho algoritmo es muy útil y funciona muy bien en ambientes con una relación señal a ruido de al menos 30 dBC y está basado en dos mediciones de la voz, la energía de tiempo corto y la tasa de cruces por cero. Por último, se escogerá el que tenga mejor desempeño.

- Algoritmo por función de energía

La señal digitalizada es escaneada y las zonas de silencio son removidas por medio del cálculo de energía en corto tiempo. En segmentos de 10 ms si la energía promedio es menor que un valor umbral, generalmente de valor 0.2 (proporcional a la energía promedio de la señal entera) es descartado, Velásquez (2008). A continuación, se muestran las fórmulas que se utilizarán:

$$E_n = \sum_{m=0}^{N-1} |x[m]|^2 \quad (2.17)$$

$$E_{avg} = \frac{1}{N} \sum_{k=1}^N |x[k]|^2 \quad (2.18)$$

Donde E_n es la energía promedio de cada segmento y E_{avg} es la energía promedio de la señal entera.

- Algoritmo de Rabiner y Sambur tipo 1

Tanto la energía como la magnitud son útiles para distinguir segmentos sordos y sonoros en la señal de voz. Pero existen otras maneras para identificar segmentos sonoros, hay un método denominado *cruces por cero y máximos*, se puede decir que una señal clasificada como ruido (la s es un ruido de alta frecuencia) provoca en la amplitud un cambio de signo, de esta manera se pueden localizar consonantes fricativas.

El algoritmo de Rabiner y Sambur combina la energía instantánea en una trama, con medidas de tasas de cruce por cero y confían en que un nivel alto de energía es el mejor estímulo para la detección. Se asume que la tasa de cruces por cero es bien distinta en zonas de voz con baja energía o de ruido de fondo y sería el punto adicional para determinar los límites de la voz.

- Energía de tiempo corto

La energía de la voz $E(n)$, se define como la suma de las magnitudes de la voz al cuadrado en intervalos centrados de 10 ms, todo medido sobre el intervalo de

medición, tal como se definió en la Ecuación (2.17).

En donde, $x(n)$ son las muestras de voz muestreadas a la frecuencia escogida. El escoger una ventana de 10 ms para el cálculo de la energía y el uso de la función de magnitud en lugar del cuadrado de la magnitud, se realizó porque puede causar un aumento en el espectro de la potencia, además así podemos aumentar la velocidad de computo, debido a que requerirá de una operación menos. La función de magnitud se define en la Ecuación (2.22).

- Tasa de cruces por cero

La tasa de cruces por cero de una grabación de voz $Z(n)$ se define como el número de cruces por cero en un intervalo de 10 ms. Aunque esta es altamente susceptible a ruido de 60 Hz, en la mayoría de los casos es una medida muy buena y razonable de la presencia o ausencia de silencio en la grabación.

La principal suposición que se realiza al momento de aplicar este algoritmo es que los primeros 100 ms de grabación son silencio. Es por eso que, en este intervalo de tiempo, las estadísticas de silencio en el fondo de la grabación son medidas. Estas mediciones incluyen, la media de los valores de los datos digitalizados y la desviación estándar de la tasa de cruces por cero, así como la energía promedio.

$$Z(n) = \sum_{m=-\infty}^{\infty} |[x(m)] - [x(m-1)]| r(n-m) \quad (2.19)$$

Donde:

$$\begin{aligned} x(n) &= 1, x(n) \geq 0 \\ &= -1, s(n) < 0 \end{aligned} \tag{2.20}$$

Y

$$\begin{aligned} r(n) &= \frac{1}{2N}, 0 \leq n \leq N-1 \\ &= 0, \text{ de otra manera} \end{aligned} \tag{2.21}$$

- Algoritmo para la detección de inicio

1. Por cada trama de 128 muestras, calcular las funciones: magnitud promedio M_n y cruce por ceros Z_n , partiendo de las ecuaciones (2.17), (2.19), (2.20), (2.21) estas funciones se definen a continuación:

$$M_n = \sum_{m=0}^{N-1} |x[m]| \tag{2.22}$$

$$Z_n = \frac{\sum_{m=0}^{N-2} |sign(x[m+1]) - sign(x[m])|}{2N} \tag{2.23}$$

2. Para obtener las estadísticas del ruido ambiental se considera que las primeras diez ventanas son ruido, con lo cual se tiene:

$$\begin{aligned} Ms_n &= \{M_1, M_2, \dots, M_{10}\} \\ Zs_n &= \{Z_1, Z_2, \dots, Z_{10}\} \end{aligned} \tag{2.24}$$

Donde Ms_n es la magnitud del ruido y Zs_n son los cruces por cero del ruido.

3. Calcular la media y la desviación estándar para las características del ruido y obtener los siguientes umbrales:

Tabla 2.5: Umbrales para la detección de inicio y fin de la señal de voz.

Umbbral	Nombre del umbral	Valor
$UmbSupEnrg$	Umbral Superior de Energía	$0,5^* \max\{M_n\}$
$UmbInfEnrg$	Umbral Inferior de Energía	$\mu_{Ms} + 2^* \sigma_{Ms}$
$UmbCruCero$	Umbral de cruces por cero	$\mu_{Zs} + 2^* \sigma_{Zs}$

Fuente: UNAM (2015)

$$\mu = \frac{\sum_i^n x_i}{N} \quad (2.25)$$

$$\sigma = \sqrt{\frac{\sum_i^n (x_i - \mu)^2}{N - 1}} \quad (2.26)$$

Para el caso de $UmbSupEnrg$ se usa M_n de la señal de voz (se considera a partir de las 10 tramas en adelante como señal de voz). En los siguientes dos casos, se usan los diez valores del ruido, obtenidos en el punto 2 para cada umbral. Recordar que es la media y la desviación estándar de la magnitud del ruido, así como la media y la desviación estándar del cruce por ceros del ruido.

4. Recorrer la función M_n incrementando en una unidad a n de 11 hasta que $M_n > UmbSupEnrg$. En este punto estamos garantizando presencia de señal. A este punto lo marcaremos como I_n .
5. Resulta lógico pensar que el inicio de la señal se encuentra en algún punto

anterior a I_n , por lo que ahora recorremos la función M_n desde $n = I_n$ hasta que $M_n < UmbInfEnrg$. Este punto lo marcaremos como I_e y lo reconocemos tentativamente como el inicio de la señal, determinado por la función de magnitud.

6. Ahora disminuimos n , desde $n = I_e$ hasta $n = I_e - 25$ o en su defecto $n = 11$, verificando si se presenta alguna de las siguientes condiciones en la función de cruces por cero, ya que lo que ahora buscamos es la posibilidad de que un sonido no sonoro preceda a un sonido sonoro. Sí $Z_n < UmbCruCero$ significa que no encontramos alguna porción de la señal con aumento importante de frecuencia en 25 ventanas anteriores, por lo tanto, el inicio es I_e . Sí encontramos que $Z_n > UmbCruCero$ menos de tres veces seguidas significa que solo fue una espiga de ruido, el punto de inicio sigue siendo I_e .
 7. Si encontramos que $Z_n > UmbCruCero$ al menos tres veces seguidas hemos encontrado un sonido no sonoro, entonces buscamos el punto n para el cual $Z_n > UmbCruCero$ la primera de las más de tres veces, es decir, el punto para el cual la función Z_n sobrepasa el umbral, indicando el comienzo del sonido no sonoro y desplazamos el inicio de la palabra de I_e a I_z .
- Algoritmo para la detección de fin

Para la detección de fin de la señal de voz hacemos lo mismo, pero en sentido

inverso a partir del punto (4) de la sección anterior, como si detectáramos un inicio con la señal invertida en el tiempo.

- Algoritmo de Rabiner y Sambur tipo 2

Al igual que el algoritmo anterior esta variante del algoritmo de Rabiner y Sambur también utiliza la energía de tiempo corto y la tasa de cruces por cero. La Figura 2.23 explica su funcionamiento, donde la señal se filtra en un ancho de banda entre 100 Hz y 4 KHz, con una frecuencia de muestreo de 8 KHz. De forma simple, el algoritmo de detección se basa en medidas de la energía cada 10 milisegundos (80 muestras a 8 KHz) según la fórmula.

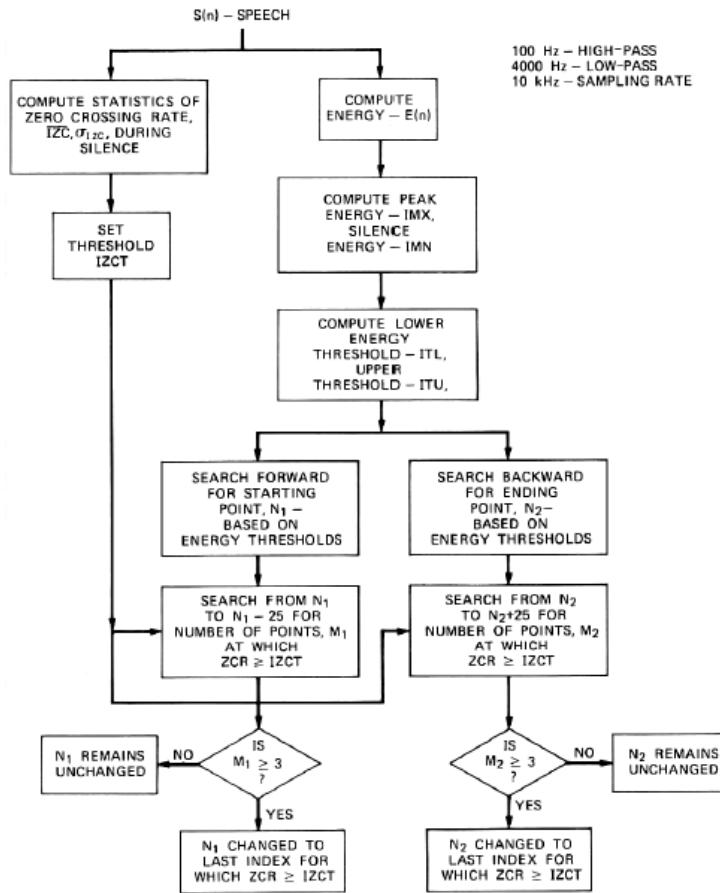


Figura 2.23: Diagrama de flujo del algoritmo Rabiner y Sambur.

Fuente: Rabiner and Sambur (1975)

Por otro lado, la tasa de cruces por cero $Z(n)$, también se calcula una vez cada trama de 10 milisegundos. El algoritmo supone que durante los 100 milisegundos primeros de la grabación no hay voz presente. Durante este periodo de silencio inicial se miden, la media IZC (Cruce por Cero Integrado), y la desviación estándar IZC , de la tasa de cruces por cero, la IMX (Energía Máxima de Voz) y la IMN (Energía Media del

Ruido de fondo). Una vez obtenida la energía de toda la grabación se establecen los siguientes umbrales:

$$IZCT = \text{MIN}(IF, \overline{IZC} + 2\sigma_{IZC}) \quad (2.27)$$

$$I1 = 0,03 \cdot (IMX - IMN) + IMN \quad (2.28)$$

$$I2 = 4 \cdot IMN \quad (2.29)$$

$$ITL = \text{MIN}(I1, I2) \quad (2.30)$$

$$ITU = 5 \cdot ITL \quad (2.31)$$

El umbral $IZCT$ tiene en cuenta el mínimo entre la tasa de cruces por cero promedios en una zona de silencio (IZC) más dos veces su desviación estándar y un umbral de tasa de cruces por cero IF , típicamente con valor 25. $I1$ simboliza la suma entre la IMN (Energía Media del Ruido de fondo) y el 3 % de la diferencia entre el IMX (*Valor Máximo de la Energía de voz*) y la IMN (Energía Media del Ruido de fondo). $I2$ es cuatro veces la IMN , ITL el mínimo entre $I1$ e $I2$, y finalmente ITU , cinco veces ITL .

En el detector sencillo que proponen Rabiner y Sambur, la decisión viene dada en función de una combinación de umbrales de energía y de tasa de cruces por cero. El diagrama de energía se puede visualizar en la Figura 2.24.

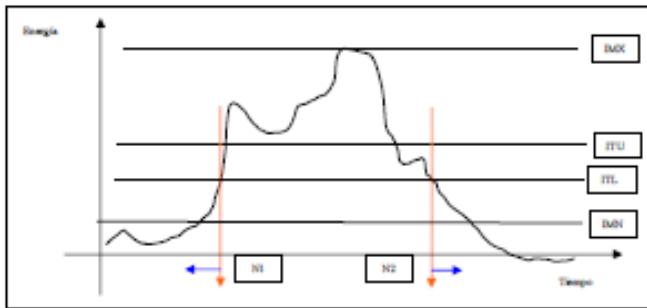


Figura 2.6. Diagrama de energía.

Figura 2.24: Diagrama de energía.

Fuente: Varela (1994)

En este caso IF es una constante de valor 25, ajustada manualmente, e IMX es el pico de energía de la grabación incluyendo la voz principal. El comienzo del pulso, trama N_1 , es el punto en el que la energía excede ITL :

- Si se supera ITL , estamos ante un posible pulso, pero todavía no se da ninguna confirmación debido a que podemos estar ante un pulso espurio.
- Una vez superado ITL , si además se supera ITU , ya se da la confirmación de pulso y se confirma también inicio de palabra. Se toma como inicio de pulso el instante de tiempo N_1 en el que supera ITL . Algo análogo ocurre con el final de pulso N_2 .

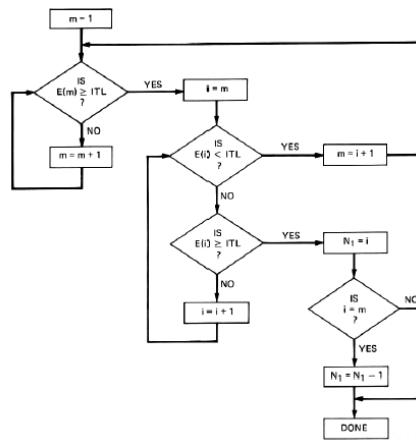


Figura 2.25: Diagrama de flujo para la búsqueda del punto de inicio basado en la energía.

Fuente: Rabiner and Sambur (1975)

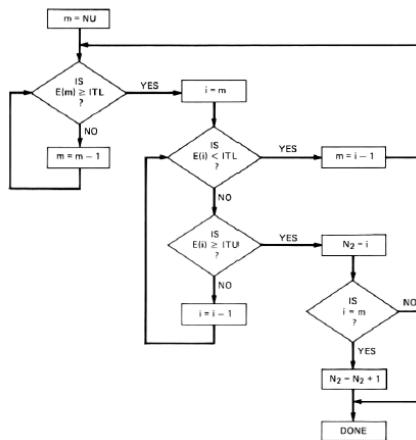


Figura 2.26: Diagrama de flujo para la búsqueda del punto de fin basado en la energía.

Fuente: Rabiner and Sambur (1975)

Estos extremos son conservativos y se refina la localización de los extremos con la tasa de cruces por cero. El algoritmo examina desde N_1 hasta $N_1 - 25$ (250 milisegundos) y si el número de veces que se excede el umbral $IZCT$ es tres o más, el punto de comienzo se retrasa al primer punto (en el tiempo) en el que excede el mencionado umbral. De lo contrario, el comienzo se mantiene en N_1 .

De igual manera se opera en el intervalo comprendido entre N_2 y $N_2 + 25$ (250 milisegundos). En cuanto a los umbrales de energía, se configuran los umbrales en función de lo robusto que se quiera ser ante espurios (por ejemplo, umbrales de energía ITL e ITU más altos). Es importante comentar que se pueden tratar los umbrales de energía que se quiera, tres, cuatro, en lugar de dos como en el caso del ejemplo (ITL e ITU).

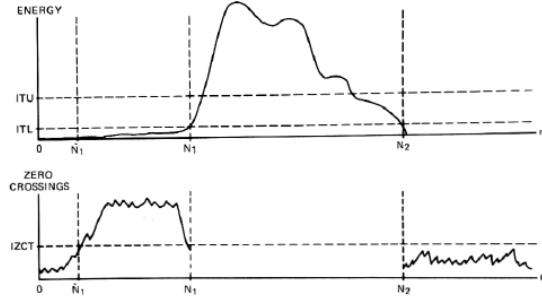


Figura 2.27: Puntos de inicio y fin de la señal de voz.

Fuente: Rabiner and Sambur (1975)

d) Filtro de preénfasis

Consiste en aplicar un filtro digital pasa altas de primer orden a la señal. Las frecuencias altas de los formantes se enfatizan por dos razones, primero para que no se pierda información durante la segmentación, ya que la mayoría de la información está contenida en las frecuencias bajas, y segundo para remover la componente DC de la señal, aplanándola espectralmente, Cook (2002).

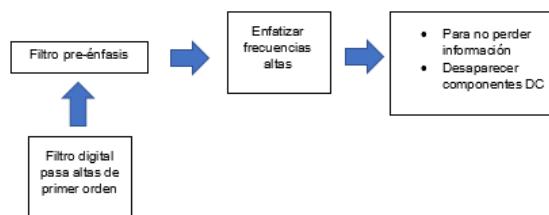


Figura 2.28: Diagrama del filtro de preénfasis.

Fuente: Pérez et al. (2013)

Las componentes de alta frecuencia presentes en la señal de voz, normalmente tienen una amplitud menor con respecto a las componentes de baja frecuencia, debido a la atenuación que ocurre durante el mecanismo de producción de la voz, por lo que se necesita, realizar un preénfasis de las componentes de alta frecuencia, a fin de obtener una amplitud similar para todas las componentes, de tal manera que el espectro de la señal sea lo más plano posible. Este filtro FIR pasa altas de primer orden, se encuentra definido por la siguiente ecuación,

siendo $x(n)$ la señal de voz digitalizada:

$$x_p = x(n) - ax(n-1) \quad n = 0, 1, 2, 3, \dots, N_x - 1 \quad (2.32)$$

$0,9 \leq a \leq 1$

Donde $x_p(n)$ es la respuesta del filtro o la señal de voz con preénfasis y α es el parámetro de preénfasis. El valor más común para el parámetro de preénfasis es $\alpha = 0,95$, el cual proporciona un incremento de unos 20 dBC de amplificación para las componentes de alta frecuencia del espectro. No obstante, el valor exacto de este parámetro no es tan crítico. La Figura 2.29 muestra la magnitud de la respuesta en frecuencia de un filtro de preénfasis para una frecuencia de muestreo de $f_s = 8000\text{Hz}$ y un parámetro de preénfasis $\alpha = 0,95$.

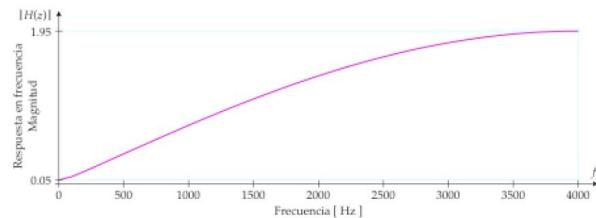


Figura 2.29: Magnitud de la respuesta en frecuencia del filtro de preénfasis.

Fuente: ?

e) Segmentación

La segmentación consiste en cortar la señal de voz en segmentos de análisis y es asumida como estacionaria en estos. Durante este proceso los segmentos son guardados cada uno como la columna de una matriz, para el posterior procesamiento de la señal de voz.

Para este proceso generalmente se emplea una ventana de Hamming de 30 ms que es aplicada a la señal de voz, enfatizada previamente con el filtro de preéñfasis. Con un desplazamiento típico 10ms entre cada ventana.

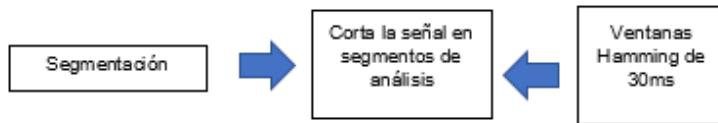


Figura 2.30: Diagrama de segmentación, ventaneo y recorte de la señal de voz.

Fuente: (Pérez et al., 2013)

- División de tramas

Los métodos tradicionales de análisis en frecuencia son bastante confiables para las señales estacionarias, pero este no es el caso de la señal de voz. Entonces, para poder emplear estas técnicas, resulta necesario dividir la señal en segmentos cortos o tramas para poder realizar un análisis en tiempo corto. Estas tramas pueden ser separadas y procesadas de manera individual, como si fueran segmentos cortos de un sonido sostenido con propiedades fijas. Este proceso se repite tantas veces como sea necesario hasta que se haya abarcado completamente la señal.

Estos segmentos o tramas de análisis, son de longitud finita, consisten de N muestras y normalmente se traslanan entre sí, teniendo S muestras de desplazamiento o separación

entre el inicio de una trama y la siguiente:

$$\begin{aligned} l &= 0, 1, 2, 3, \dots, L - 1 \\ x_l &= x(n + l \cdot S) \\ n &= 0, 1, 2, 3, \dots, N - 1 \end{aligned} \quad (2.33)$$

En donde $x_l(n)$ es la l -ésima trama y L es la cantidad o el número total de tramas presentes en la señal de voz $x(n)$. La Figura 2.31 ilustra la división en tramas con traslape de una señal de voz, para una separación entre tramas de $S = N/3$ muestras.

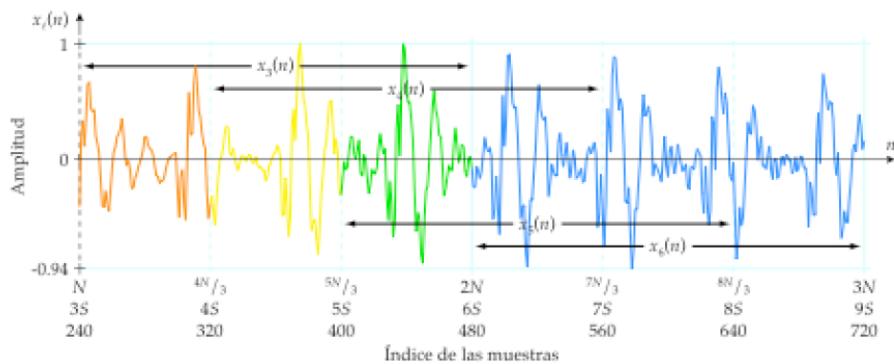


Figura 2.31: División en tramas con traslape de una señal de voz.

Fuente: ?

A partir de la figura anterior, se puede observar que la cantidad o el número total de tramas en la señal de voz es L , se puede calcular mediante la siguiente ecuación:

$$L = \frac{N_x - N}{S} + 1 \quad (2.34)$$

Si $N > S$, entonces sí ocurre el traslape y $N-S$ muestras al final de las tramas son duplicadas al inicio de la siguiente trama. Entre mayor sea el traslape, la correlación entre las tramas adyacentes también será mayor y los cambios serán más suaves. Por

otra parte, si $N < S$, entonces no hay traslape, pero además algunas porciones de la señal de voz se perderán, ya que no aparecerán en ninguna de las tramas. Por lo tanto, esta última situación resulta inaceptable en cualquier tipo de procesamiento de voz.

Entonces, la señal de voz se divide en L tramas para su análisis. Cada una de éstas se compone por N muestras con S muestras de separación entre tramas adyacentes. Sin embargo, para no perder las propiedades dinámicas o variantes en el tiempo de la señal de voz, es necesario volver a calcular otros coeficientes para la siguiente trama.

- Ponderado por la ventana de Hamming

Primero demos un vistazo a la más simple de las ventanas, la cual tiene una forma rectangular y se define como:

$$w(n) = \begin{cases} 0 & \text{para } n < 0 \\ 1 & \text{para } 0 \leq n \leq N - 1 \\ 0 & \text{para } n > N - 1 \end{cases} \quad (2.35)$$

Esta ventana rectangular se usa de manera implícita cuando se extrae una trama de N muestras a partir de una señal de voz:

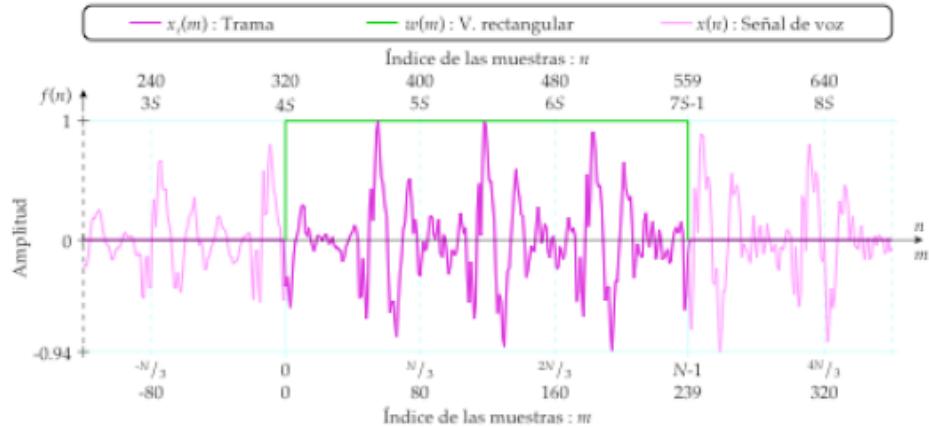


Figura 2.32: Extracción de una trama mediante una ventana rectangular.

Fuente: ?

Sin embargo, la presencia de ésta ventana ocasiona la distorsión del espectro de la señal de voz, pero si se pondera la trama por una ventana con la forma adecuada, entonces se puede reducir la distorsión en la frecuencia, aunque se deformé la señal en el tiempo.

En el reconocimiento de voz, la ventana que más se utiliza es la de Hamming:

$$w(n) = \begin{cases} 0 & \text{para } n < 0 \\ 0,54 - 0,46\cos\left(\frac{2\pi n}{N-1}\right) & \text{para } 0 \leq n \leq N-1 \\ 0 & \text{para } n > N-1 \end{cases} \quad (2.36)$$

La Figura 2.33 es la representación gráfica de la ventana de Hamming, la cual tiene valores que decaen suavemente hacia cero, aunque en los extremos, tiene una transición abrupta para el primer y el último valor de la ventana, los que son iguales a 0.08.



Figura 2.33: Ventana de Hamming.

Fuente: ?

El segmento o la trama ponderada $x_w(n)$ se obtiene mediante el producto de cada una de las muestras de la trama de análisis $x_l(n)$ por el peso correspondiente de la ventana de Hamming $w(n)$:

$$x_w(n) = x_l(n) \cdot w(n)$$

$$x_w(n) = x_l(n) \cdot \left[0,54 - 0,46 \cos \left(\frac{2\pi n}{N-1} \right) \right] \quad n = 0, 1, 2, 3, \dots, N-1 \quad (2.37)$$

El ponderado de la trama de análisis contribuye a minimizar las discontinuidades al inicio y al final de las tramas, al atenuar los valores en los extremos. De otro modo la disparidad entre la primera y la última muestra de la trama ocasionaría algunos efectos indeseables en los valores del espectro en frecuencia. No obstante, si las muestras que se encuentran cerca de los extremos de la trama describen algún evento significativo de corta duración, entonces este evento prácticamente no afecta a la trama, dado que recibe un ponderado bajo. Así que normalmente, las tramas se traslanan para asegurar que todos estos eventos se tomen en cuenta al ser cubiertos en las tramas adyacentes.

- Longitud de trama o tamaño de ventana

La selección de la longitud de la trama N es una consideración muy importante, pues por lo general, el trabajo de computo requerido durante el análisis es proporcional a esta, por lo que resulta ventajoso mantenerla tan pequeña como sea posible. Sin embargo, se deben abarcar varios periodos del tono fundamental (pitch) para asegurar que los resultados sean confiables. Además, debido al ponderado, la longitud de la trama debe ser lo suficientemente larga como para que los efectos de atenuación de la ventana de Hamming no afecten seriamente los resultados.

Adicionalmente, la longitud o la duración de la trama se determina mediante un compromiso entre las resoluciones en el tiempo y en la frecuencia, ya que la longitud de la trama es proporcional a la resolución en frecuencia, pero inversamente proporcional a la resolución en el tiempo. De manera similar, el traslape es proporcional al número de tramas en la señal de voz, pero también es proporcional a la correlación de las tramas subsecuentes. Por lo tanto, se debe buscar un balance o compromiso entre estas características contrapuestas.

Una manera para determinar V (longitud de la trama N), es haciendo el producto de la duración de la trama T por la frecuencia de muestreo f_s de la señal de voz:

$$V = (T : f_s)_{int} \quad (2.38)$$

No obstante, como el número de muestras por trama N debe ser un valor entero, se debe redondear o truncar el resultado. El número de muestras de separación entre tramas adyacentes S , generalmente se selecciona de tal manera que se obtenga una tasa de 100 tramas por segundo ($Fps = 100$), es decir una trama cada 10 ms.

$$Fps = \frac{f_s}{S} \quad \Rightarrow \quad S = \frac{f_s}{100} \quad (2.39)$$

Los valores típicos de V son de 10 a 30 ms. Las ventanas cortas han sido propuestas para estimar los parámetros del tracto vocal que varían rápidamente, mientras que las ventanas largas se usan para estimar la frecuencia del tono fundamental (pitch). Las tramas de 20 a 30 ms de duración generalmente tienen una buena relación entre ambas resoluciones, Pérez et al. (2013).

Hasta aquí hemos visto todos los componentes que conforman el módulo de preprocesamiento, a continuación, veremos el módulo de entrenamiento.

2.1.2.2. Entrenamiento

Para el entrenamiento se deben capturar diversas palabras recortadas por medio del módulo de preprocesamiento, mismas que representan las diferentes repeticiones. A partir de estas repeticiones se obtiene un conjunto de centroides que son almacenados en la memoria. Para poder obtener estos centroides los valores numéricos obtenidos del preprocesamiento deben pasar por un algoritmo de extracción de características.

a) Extracción de características

Es el proceso mediante el cual, dado un conjunto de datos se desea obtener otro conjunto de datos resumidos, pero que contengan solo la información necesaria de los datos de entrada.

Una buena técnica de extracción de características garantiza que dichos datos obtenidos son únicos para una determinada entrada, es decir que para dos señales de entrada diferentes, se obtiene dos conjuntos de características diferentes, Becchetti and Prina (1999).

Sin embargo, obtener características plenamente diferentes involucra que todas las señales sean diferentes, así se traten de dos palabras iguales, pero con diversas tonalidades y velocidades. Por lo tanto, la obtención de características debe de asegurarme, que en ambas señales existen características que son propias para ambas. De esta manera podemos obtener un grado de similaridad adecuado para el reconocimiento de la señal de voz.

Entre muchas técnicas de extracción de características, podemos mencionar las siguientes:

- Algoritmo de características Fonético Acústicas
- Algoritmo basado en la Transformada de Fourier
- Codificación Predictiva Lineal (LPC)
- Coeficientes Cepstrales en Escala Mel (MFCC)
- Predicción Perceptual Lineal (PLP)
- Algoritmo basado en Wavelet.

En (UNAM, 2015), se hace un análisis comparativo entre los algoritmos LPC, MFCC y PLP teniendo como resultado al MFCC como mejor algoritmo de extracción de características, ver la Figura 2.34. En (Guevara and Salazar, 2007), se realiza una evaluación y comparación entre los Wavelets y los MFCC, resultando también a los MFCC con mejor desempeño, pero si bien tienen la misma complejidad computacional el tiempo de ejecución de los Wavelets es menor.

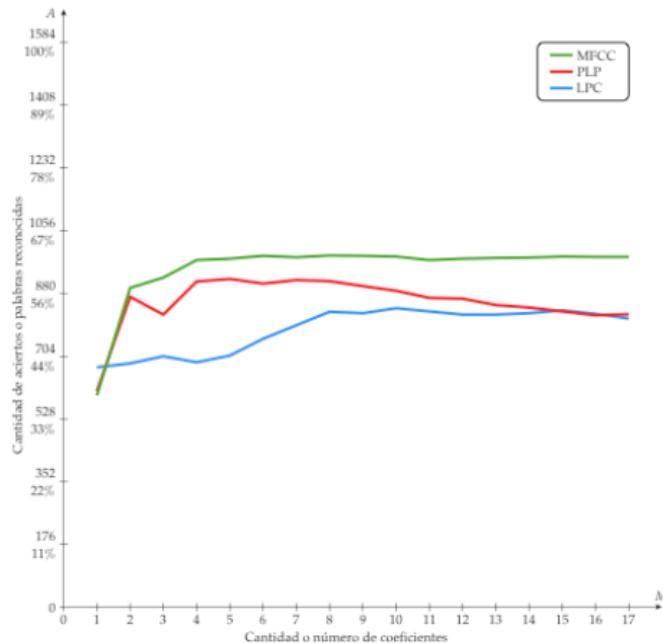


Figura 2.34: Comparación de la variación en el reconocimiento entre LPC, MFCC Y PLP.
Fuente: UNAM (2015)

Para este trabajo de Tesis teniendo lo anterior como antecedentes, hemos optado por la técnica MFCC (Coeficientes Cepstrales en la Frecuencia de Mel). La Figura 2.35 muestra el proceso del MFCC y como al final se retornan los cepstrum.



Figura 2.35: Proceso de extracción de características con MFCC.
Fuente: Rabiner and Sambur (1975)

b) Codificación o análisis Mel cepstral

En 1980 Davis y Mermelstein propusieron un nuevo tipo de representación del cepstrum, en el cual combinaron sus ventajas con los conocimientos de la percepción no lineal de la frecuencia en el sistema auditivo humano, tomando como base los estudios realizados por Zwicker en 1961. En este método de análisis, se utiliza la parte real del cepstrum, pero con una transformación no lineal de la frecuencia, pasando de la escala en Hertz a la escala de frecuencia Mel. Por eso, a estos coeficientes se les ha denominado como los Coeficientes Cepstral en la Frecuencia Mel o MFCC, Pérez et al. (2013).

La mayor parte de los sistemas han estado convergiendo hacia el uso de estos vectores, que se obtienen a partir de un procesamiento de la señal de voz empleando un banco de filtros que ha sido diseñado tomando en cuenta algunas propiedades de la percepción del sonido en el sistema auditivo humano, con lo que se busca imitar su comportamiento.

A continuación, se describe un procedimiento para obtener los coeficientes MFCC. La Figura 2.36 muestra el diagrama de bloques para este procedimiento. Los bloques muestran las operaciones necesarias y a la izquierda de éstos se muestran los parámetros de entrada requeridos en cada paso. A partir del cuarto bloque, el ponderado por la ventana de Hamming, las operaciones se realizan sobre las tramas de la señal de voz. La consiguiente reducción en el número de parámetros para cada trama se ilustra mediante la cantidad de flechas.

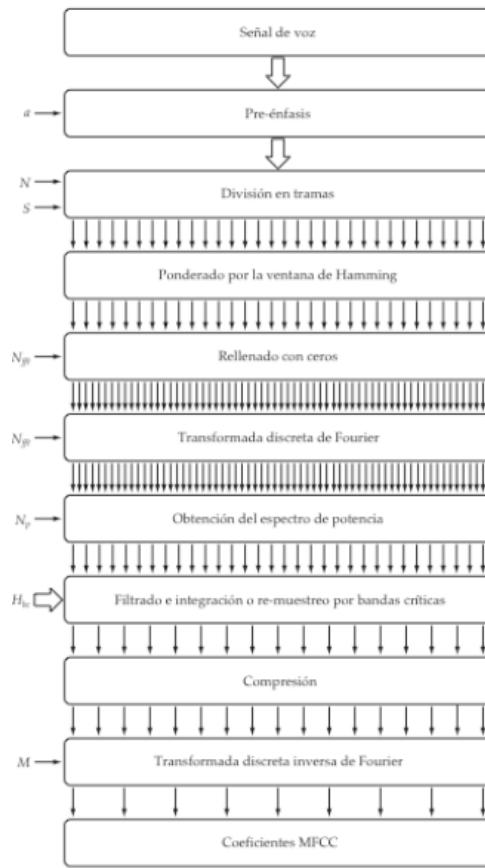


Figura 2.36: Procedimiento para la obtención de los coeficientes MFCC.

Fuente: Pérez et al. (2013)

Las etapas preénfasis, división de tramas y ponderado por la ventana de Hamming ya las vimos en la etapa de preprocessamiento, por lo que solamente veremos las etapas restantes.

- Transformada discreta de Fourier

El espectro en frecuencia de una señal discreta se puede estimar mediante DFT (Transformada Discreta de Fourier):

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi nk}{N}} \quad (2.40)$$

Esta transformada proporciona un conjunto de N valores en el dominio de la frecuencia, en donde $X(k)$ es la transformada discreta de Fourier en el dominio de la frecuencia, de la señal discreta $x(n)$ en el dominio del tiempo. Sin embargo, en la práctica, la transformada discreta de Fourier se calcula de una manera mucho más eficiente mediante FFT (Transformada Rápida de Fourier).

- Rellenado con ceros

Para facilitar el uso de la transformada rápida de Fourier, la longitud de la trama N debería de ser igual al número de puntos para la transformada rápida de Fourier N_{fft} , un número que debe ser igual a la potencia de dos más próxima:

$$N_{fft} = 2^{\lceil \log_2(N) \rceil_{int}} \quad (2.41)$$

Pero si no se cumple con esta condición, entonces se puede llenar con ceros, es decir, añadir las muestras nulas que hagan falta.

$$x_{zp}(n) = \begin{cases} x_w(n) & \text{para } n = 0, 1, 2, 3, \dots, N - 1 \\ 0 & \text{para } n = N, N + 1, \dots, N_{fft} - 1 \end{cases} \quad (2.42)$$

A partir de la Ecuación (2.40) de la transformada discreta de Fourier se tiene:

$$X(k) = \sum_{n=0}^{N_{fft}-1} x(n)e^{-j\frac{2\pi nk}{N_{fft}}} \quad (2.43)$$

Se puede apreciar que el agregar valores nulos a $x(n)$ no afecta significativamente la sumatoria para cualquier $X(k)$. Sin embargo, la longitud aumentada de la trama N_{fft} solamente proporciona una mejor descripción de la transformada discreta de Fourier, pues no incrementa la resolución en frecuencia, pues este es el único propósito de la longitud de la trama N .

- **Obtención del espectro de frecuencia**

El espectro en frecuencia de una trama de análisis se obtiene mediante el algoritmo FFT (Transformada Rápida de Fourier):

$$\begin{aligned} X(k) &= \text{FFT}\{x_{zp}(n), N_{fft}\} & k &= 0, 1, 2, 3, \dots, N_{fft} - 1 \\ & & n &= 0, 1, 2, 3, \dots, N_{fft} - 1 \end{aligned} \quad (2.44)$$

Dado que la señal de voz $x(n)$ es una secuencia de números reales, $X(k)$ y $X(N - k)$ son conjugados complejos. Puesto que la magnitud de los conjugados complejos es igual, el espectro en frecuencia resultante tiene una simetría par con respecto a la mitad

de la frecuencia de muestreo, es decir, que la magnitud de la mitad superior del espectro es un reflejo de la mitad inferior.

- Obtención del espectro de potencia

Ahora bien, no importa tanto la respuesta en frecuencia, sino la envolvente de la respuesta en frecuencia. Las características del tracto vocal se pueden estimar mediante el espectro de potencia $P(k)$ que se calcula como la magnitud del espectro al cuadrado o la suma de los cuadrados de la parte real e imaginaria del espectro:

$$\begin{aligned} P(k) &= |X(k)|^2 \\ &\quad k = 0, 1, 2, 3, \dots, N_{fft} - 1 \\ P(k) &= \text{real}[X(k)]^2 + \text{imag}[X(k)]^2 \end{aligned} \quad (2.45)$$

El espectro de potencia se compone de valores reales, enfatiza los picos en el espectro y conserva la simetría par del espectro en frecuencia. Por lo tanto, normalmente solamente se ocupa la mitad inferior del espectro, es decir los primeros N_p valores:

$$\begin{aligned} N_p &= 0,5N_{fft} \\ P(k) &= \text{real}[X(k)]^2 + \text{imag}[X(k)]^2 \quad k = 0, 1, 2, 3, \dots, N_p \end{aligned} \quad (2.46)$$

Hay que destacar que el espectro de potencia solamente proporciona información acerca de la magnitud, pues la información relacionada con la fase se descarta. Esto es consistente con el hecho de que la fase no aporta ninguna información que resulte útil. De hecho, se ha comprobado experimentalmente que la percepción de una señal reconstruida con una fase distinta es prácticamente indistinguible de la señal original, esto si se mantiene la continuidad secuencial de la fase entre las tramas.

- Filtrado e integración o remuestreo por bandas críticas

El análisis espectral revela aquellas características de la señal de voz que son consecuencia directa de la configuración del tracto vocal. Asimismo, estas características de la voz por lo general se obtienen como las respuestas o salidas de un banco de filtros.

Ahora bien, se ha encontrado que la percepción de una frecuencia en particular por parte del sistema auditivo humano está influenciada por la energía dentro de una banda crítica de frecuencias a su alrededor. En la psicoacústica, a esta propiedad se le conoce como la resolución espectral por bandas críticas.

Un método muy utilizado para implementar estos bancos, consiste en efectuar el filtrado directamente en el dominio de la transformada discreta de Fourier. Los filtros simplemente son versiones desplazadas en la escala de frecuencia de alguna ventana.

Estas ventanas no están equiespaciadas en la escala de frecuencia en Hertz, de hecho se utiliza otra escala no lineal (logarítmica) para conseguir una mejor relación entre las resoluciones en el tiempo y en la frecuencia, pues los filtros de banda angosta para baja frecuencia hacen posible la detección de las frecuencias del tono fundamental (pitch), pero el incremento en el ancho de banda de los filtros para alta frecuencia permiten una mejor resolución en el tiempo, con la que se pueden detectar las plosivas. Además, la

parte del espectro que se encuentra por debajo de 1000 Hz se procesa por una mayor cantidad de filtros, pues ésta contiene más información.

La respuesta en frecuencia del banco de filtros simula el proceso de percepción que tiene lugar dentro del oído interno del humano, por eso a este proceso de filtrado también se le denomina como ponderado perceptual. La Figura 2.37 muestra la estructura general de estos bancos de filtros perceptuales, en donde BC es la cantidad o el número de filtros o bandas críticas en el banco.

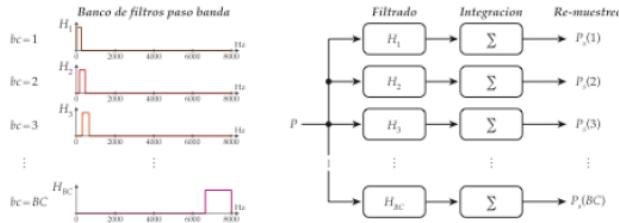


Figura 2.37: Estructura general de los bancos de filtros perceptuales.

Fuente: Pérez et al. (2013)

Entonces, al integrar las salidas individuales de los filtros, se obtiene una muestra por cada una de las bandas críticas, por lo que el espectro es remuestreado a determinados rangos de frecuencia, proporcionando así, una compresión o reducción en el número de parámetros.

El efecto neto de esta etapa es reducir la sensibilidad del espectro a la frecuencia, particularmente para las altas frecuencias, las cuales de alguna manera son enfatizadas

debido al ancho de banda más amplio de los filtros del banco.

- Cambio de la escala de frecuencia

Existen muchas formas diferentes para las ventanas que se pueden emplear en los bancos de filtros, pero todas estas se basan en alguna escala de frecuencia que se considera aproximadamente lineal para frecuencias menores de 1000 Hz, pero para frecuencias mayores que ésta, presenta un comportamiento no lineal (logarítmico).

Este cambio en la escala de frecuencia en inglés se denomina *frequency warping*, es decir, que se realiza un arqueado o una transformación de la escala de frecuencia en Hertz, a otra escala de frecuencia que exhibe un comportamiento logarítmico. Las escalas Mel, que es la que usaremos en esta tesis, y Bark son dos ejemplos, puesto que están basadas en datos experimentales de la percepción humana, existen varias aproximaciones que se pueden usar para estas. A continuación, explicaremos en qué consiste la escala de Mel.

- Escala Mel

La escala Mel fue propuesta por Stevens, Volksman y Newmann en 1940. Un Mel es una unidad de medición de la frecuencia o del tono percibido, esta no corresponde de manera lineal a la frecuencia real del tono. Se designó como punto inicial de referencia, que una frecuencia de 1000 Hz también es de 1000 Mels.

Los observadores juzgan tonos espaciados exponencialmente como tonos equiespaciados. De esta manera fue posible determinar un mapeo entre la escala de la frecuencia real (Hertz) y la escala de la frecuencia percibida (Mels).

La siguiente ecuación proporciona una equivalencia entre la escala logarítmica de la frecuencia Mel f_{mel} con respecto a la escala de frecuencia en Hertz f_{Hz} :

$$f_{mel}(f_{Hz}) = \frac{1000}{\ln(2)} \ln \left(1 + \frac{f_{Hz}}{1000} \right) \quad (2.47)$$

No obstante, se acostumbra utilizar otra función de aproximación para realizar la transformación no lineal de la frecuencia en Hertz a la escala de frecuencia Mel:

$$f_{mel}(f_{Hz}) = 1125 \ln \left(1 + \frac{f_{Hz}}{700} \right) \quad (2.48)$$

La Figura 2.38 muestra la representación gráfica de la ecuación anterior, en donde se puede apreciar el arqueado de la frecuencia al cambiar de la escala lineal de frecuencia en Hertz f_{Hz} a la escala logarítmica de la frecuencia Mel f_{mel} :

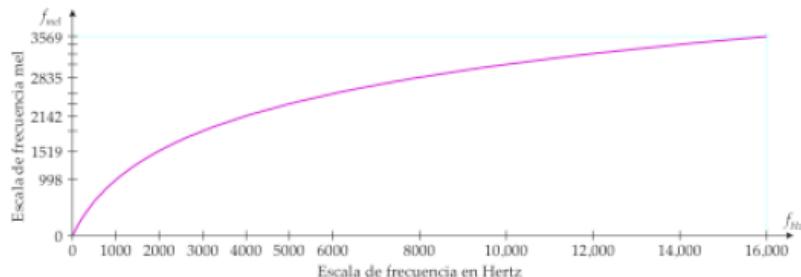


Figura 2.38: Gráfica del arqueado de la frecuencia con la escala Mel.

Fuente: Pérez et al. (2013)

- Definición de las bandas críticas triangulares con la escala Mel

El diseño de los filtros para las bandas críticas primero se realiza en la escala Mel. Los filtros utilizados en este método presentan una forma triangular, que es simétrica con respecto a su frecuencia central y tienen el mismo ancho de banda en la escala Mel.

Aunque a estas ventanas triangulares se les suele llamar filtros, estas simplemente se usan para promediar el espectro de potencia con respecto a la frecuencia central de cada una de las bandas críticas. La función que las define está dada por la siguiente ecuación y su representación gráfica se muestra en la Figura 2.39.

$$H_{bc}(f) = \begin{cases} 0 & \text{para } f \leq f_l \\ \frac{1}{(f_c - f_l)}(f - f_l) & \text{para } f_l < f < f_c \\ 1 & \text{para } f = f_c \\ \frac{-1}{(f_h - f_c)}(f - f_h) & \text{para } f_c < f < f_h \\ 0 & \text{para } f \geq f_h \end{cases} \quad (2.49)$$



Figura 2.39: Curva para las bandas críticas triangulares.

Fuente: Pérez et al. (2013)

En la Ecuación (2.49) se puede apreciar que los primeros términos (cocientes) son las pendientes de las rectas que forman la ventana triangular, y los otros términos del producto contienen a la variable independiente con su respectiva abscisa al origen. Ahora bien, la transformación no lineal de la frecuencia en Hertz a la escala de frecuencia Mel, se acostumbra realizar mediante la función de aproximación Ecuación (2.48).

El rango de frecuencias de interés abarca desde los 0 Hz hasta la frecuencia de Nyquist, sin embargo, se puede determinar un límite inferior de frecuencia f_{min} , y un límite superior de frecuencia f_{max} . Para la señal de voz, se recomienda que $f_{min} > 100\text{Hz}$ y que el límite superior de frecuencia sea menor a la frecuencia de Nyquist, aunque para una frecuencia mayor de 6800 Hz ya no hay mucha información en la señal de voz. Además, si se fija el límite inferior de frecuencia arriba de 50 o 60 Hz, se puede evitar el ruido procedente de la red de suministro o distribución de AC. Ahora bien, puesto que estos límites están definidos en Hertz, hace falta transformarlos a la escala Mel.

$$(f_{min})_{mel} = 1125 \ln \left(1 + \frac{f_{min}}{700} \right) \quad (f_{max})_{mel} = 1125 \ln \left(1 + \frac{f_{max}}{700} \right) \quad (2.50)$$

Las frecuencias centrales deben estar equiespaciadas en la escala Mel, entonces se puede calcular la separación en frecuencia entre éstas SF , mediante la siguiente fórmula:

$$SF = \frac{(f_{max})_{mel} - (f_{min})_{mel}}{BC + 1} \quad (2.51)$$

Las frecuencias centrales están separadas por múltiplos enteros de este parámetro más el valor del límite inferior de frecuencia:

$$f_c(bc)_{mel} = bc \cdot SF + (f_{min})_{mel} \quad bc = 1, 2, 3, \dots, BC \quad (2.52)$$

En donde bc denota el número correspondiente de la banda crítica. Ahora sólo hace falta transformar las frecuencias centrales de la escala Mel a la escala en Hertz mediante la siguiente fórmula, que es la función inversa de la Ecuación (2.48)

$$f_{Hz}(f_{mel}) = 700 \left[e^{\left(\frac{f_{mel}}{1125} \right)} - 1 \right] \quad (2.53)$$

Con estos valores de las frecuencias centrales en Hertz, ya se pueden determinar las frecuencias inferior y superior para cada una de las bandas críticas. La frecuencia inferior para una banda crítica bc es igual a la frecuencia central de la banda crítica anterior:

$$f_l(bc) = f_c(bc - 1) \quad (2.54)$$

La frecuencia superior para una banda crítica es igual a la frecuencia central de la banda crítica posterior:

$$f_h(bc) = f_c(bc + 1) \quad (2.55)$$

La frecuencia inferior de la primera banda crítica es igual al límite inferior de frecuencia:

$$f_l(1) = f_{min} \quad (2.56)$$

De manera similar, la frecuencia superior de la última banda crítica es igual al límite superior de frecuencia.

$$f_h(BC) = f_{max} \quad (2.57)$$

La Tabla 2.6 muestra los valores para las frecuencias inferior, central y superior para un banco de $BC = 17$ filtros, tanto en la escala Mel como en la escala en Hertz. El límite inferior de frecuencia es $f_{min} = 0$ Hz y el límite superior de frecuencia es $f_{max} = 4000$ Hz.

Tabla 2.6: Valores de las frecuencias para las bandas críticas triangulares.

bc [banda c.]	fl [mel]	fc [mel]	fh [mel]	fl [Hz]	fc [Hz]	fh [Hz]
1	0.000	119.015	238.030	0.00	78.11	164.94
2	119.015	238.030	357.045	78.11	164.94	261.46
3	238.030	357.045	476.059	164.94	261.46	368.75
4	357.045	476.059	595.074	261.46	368.75	488.01
5	476.059	595.074	714.089	368.75	488.01	620.58
6	595.074	714.089	833.104	488.01	620.58	767.94
7	714.089	833.104	952.119	620.58	767.94	931.75
8	833.104	952.119	1071.134	767.94	931.75	1113.84
9	952.119	1071.134	1190.148	931.75	1113.84	1316.24
10	1071.134	1190.148	1309.163	1113.84	1316.24	1541.23
11	1190.148	1309.163	1428.178	1316.24	1541.23	1791.33
12	1309.163	1428.178	1547.193	1541.23	1791.33	2069.34
13	1428.178	1547.193	1666.208	1791.33	2069.34	2378.37
14	1547.193	1666.208	1785.223	2069.34	2378.37	2721.88
15	1666.208	1785.223	1904.237	2378.37	2721.88	3103.72
16	1785.223	1904.237	2023.252	2721.88	3103.72	3528.18
17	1904.237	2023.252	2142.267	3103.72	3528.18	4000.00

Fuente: Pérez et al. (2013)

En la Figura 2.40 se muestra la representación gráfica, en la escala Mel, del banco de filtros triangulares mostrado en la Tabla 2.6. Las bandas críticas están equiespaciadas

y todas las bandas críticas tienen el mismo ancho de banda.

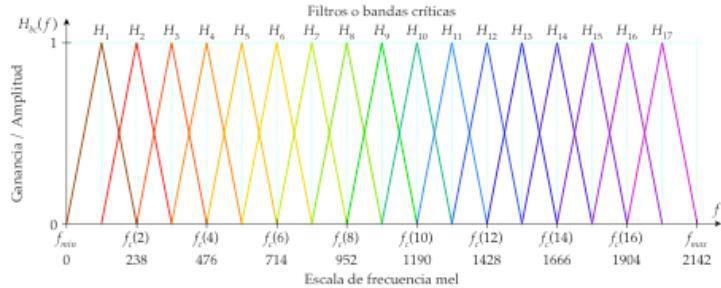


Figura 2.40: Banco de filtros o ventanas triangulares equiespaciados en la escala Mel.

Fuente: Pérez et al. (2013)

La Figura 2.41 muestra el mismo banco de filtros, pero ahora en la escala en Hertz. Puede apreciarse como la distribución de las bandas críticas ahora sigue una distribución logarítmica y también como se incrementa el ancho de banda al aumentar la frecuencia.

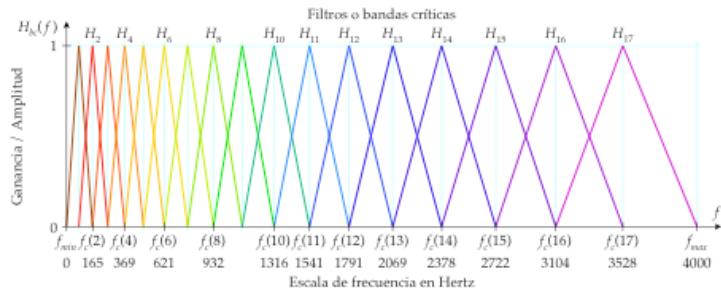


Figura 2.41: Banco de filtros o ventanas triangulares en Hertz con distribución logarítmica.

Fuente: Pérez et al. (2013)

Finalmente, sólo hace falta hacer un muestreo de las bandas críticas de acuerdo al número de puntos del espectro de potencia N_p . Ahora bien, como las bandas críticas contienen unos cuantos valores diferentes de cero, no hace falta calcular todos los valores correspondientes a cada uno de los puntos del espectro de potencia, basta con

las muestras comprendidas entre la frecuencia inferior y la superior para cada banda crítica.

A cada muestra del espectro de potencia $P(k)$ le corresponde un valor de frecuencia en Hertz, el cual es un múltiplo de la resolución en frecuencia del espectro:

$$f = k \cdot \Delta f \quad k = 0, 1, 2, 3, \dots, N_p \quad (2.58)$$

La resolución en frecuencia del espectro Δf , se puede calcular mediante el cociente de la frecuencia de muestreo f_s , entre el número de puntos para la transformada rápida de Fourier N_{fft} :

$$\Delta f = \frac{f_s}{N_{fft}} \quad (2.59)$$

Si se invierte este cociente, el nuevo factor es la resolución por muestra del espectro Δk :

$$\Delta k = \frac{N_{fft}}{f_s} \quad (2.60)$$

El cual se puede utilizar para determinar que índice o muestra corresponde a determinada frecuencia, al multiplicar esa frecuencia por este factor de equivalencia:

$$k = (f \cdot \Delta k)_{\text{int}} \quad (2.61)$$

Puesto que los índices de las muestras deben ser números enteros, se tiene que redondear el resultado a un número entero. El índice para la frecuencia inferior k_l se puede

redondear hacia arriba, pero el índice para la frecuencia superior k_h se puede redondear hacia abajo. También, si es necesario, hay que tomar en cuenta que la primera muestra del espectro de potencia representa a la frecuencia cero.

Con los índices correspondientes a las frecuencias inferior k_l y superior k_h , ya se pueden calcular los N_{bc} valores diferentes de cero para cada una de las bandas críticas.

$$N_{bc} = k_h(bc) - k_l(bc) + 1 \quad (2.62)$$

Para esto, primero se obtiene el valor correspondiente en frecuencia para la muestra k , comprendida dentro del intervalo $k_l(bc) \leq k \leq k_h(bc)$ de una banda crítica bc . Entonces, el valor correspondiente a la muestra k para el filtro o la banda crítica H_{bc} se determina mediante la Ecuación (2.58), según sea el caso.

En la Tabla 2.7 se muestran los índices correspondientes a las frecuencias inferior y superior para las $BC = 17$ bandas críticas de la Tabla 2.6 . El número de puntos para la transformada rápida de Fourier es de $N_{fft} = 256$ puntos y la frecuencia de muestreo es de $f_s = 8000$ Hz. Por lo tanto, la resolución en frecuencia es de $\Delta f = 31,25$ Hz/muestra, pero la resolución por muestra es de $\Delta k = 0,032$ muestras/Hz.

Tabla 2.7: Correspondencia entre índices y frecuencias para las bandas críticas Mel.

bc [banda c.]	fl [Hz]	kl [índice]	(kl)int [índice]	fh [Hz]	kh [índice]	(kh)int [índice]	Nbc [muestras]
1	0.00	0.00	0	164.94	5.28	5	6
2	78.11	2.50	3	261.46	8.37	8	6
3	164.94	5.28	6	368.75	11.80	11	6
4	261.46	8.37	9	488.01	15.62	15	7
5	368.75	11.80	12	620.58	19.86	19	8
6	488.01	15.62	16	767.94	24.57	24	9
7	620.58	19.86	20	931.75	29.82	29	10
8	767.94	24.57	25	1113.84	35.64	35	11
9	931.75	29.82	30	1316.24	42.12	42	13
10	1113.84	35.64	36	1541.23	49.32	49	14
11	1316.24	42.12	43	1791.33	57.32	57	15
12	1541.23	49.32	50	2069.34	66.22	66	17
13	1791.33	57.32	58	2378.37	76.11	76	19
14	2069.34	66.22	67	2721.88	87.10	87	21
15	2378.37	76.11	77	3103.72	99.32	99	23
16	2721.88	87.10	88	3528.18	112.90	112	25
17	3103.72	99.32	100	4000.00	128.00	128	29

Fuente: Pérez et al. (2013)

La Tabla 2.8 muestra la correspondencia entre los índices y las frecuencias para los puntos que definen la octava banda crítica Mel H_8 . Se muestran los índices k para las muestras del espectro de potencia y sus valores correspondientes de frecuencia en Hertz. La resolución en frecuencia es $\Delta f = 31,25 \text{ Hz/muestra}$. Para fines ilustrativos, se incluyen los valores exactos de los tres límites para los cinco casos de la Ecuación (2.49).

Tabla 2.8: Correspondencia entre los valores para la octava banda crítica Mel.

bc	f	CASO	H8(k)
[índice]	[Hz]		
24	750.00	1 : $f \leq fl$	0
kl(8)	767.94	1 : f ≤ fl	0
25	781.25	2 : $fl < f < fc$	0.0812
26	812.50	2 : $fl < f < fc$	0.2720
27	843.75	2 : $fl < f < fc$	0.4628
28	875.00	2 : $fl < f < fc$	0.6536
29	906.25	2 : $fl < f < fc$	0.8443
kc(8)	931.75	3 : f = fc	1
30	937.50	4 : $fc < f < fh$	0.9684
31	968.75	4 : $fc < f < fh$	0.7968
32	1,000.00	4 : $fc < f < fh$	0.6252
33	1,031.25	4 : $fc < f < fh$	0.4536
34	1,062.50	4 : $fc < f < fh$	0.2819
35	1,093.75	4 : $fc < f < fh$	0.1103
kh(8)	1,113.84	5 : f ≥ fh	0
37	1,156.25	5 : $f \geq fh$	0

Fuente: Pérez et al. (2013)

- Compresión

En lugar de simplemente usar la magnitud del espectro remuestreado mediante la escala Mel para calcular los MFCC, algunos investigadores han sugerido que es más provechoso el emplear una función logaritmo para calcular la energía total de las bandas críticas alrededor de las frecuencias centrales:

$$P_{sc}(bc) = \log[P_s(bc)] \quad bc = 1, 2, 3, \dots, BC \quad (2.63)$$

El procesamiento de la magnitud y la obtención de su logaritmo también se realizan en el oído humano. Es más, el empleo de la magnitud descarta la información innecesaria que proporciona la fase mientras que el logaritmo efectúa una compresión, dando como

resultado que la extracción de características sea menos sensible a las variaciones de la amplitud producidas por las resonancias espectrales.

- Transformada inversa de Fourier

El último paso en el cálculo de los MFCC consiste en realizar la transformada discreta inversa de Fourier del espectro muestreado y comprimido P_{sc} . En este paso es donde se obtienen los MFCC, además esta etapa del procedimiento tiene grandes ventajas, puesto que los valores del espectro son números reales y presentan una simetría par con respecto a la frecuencia de Nyquist, entonces se puede reducir a una DCT (Transformada Coseno Discreta) en lugar de la IFFT (Transformada Discreta Inversa de Fourier):

$$c_m = \sum_{bc=1}^{BC} P_{sc}(bc) \cos \left[\frac{\pi}{BC} \left(bc - \frac{1}{2} \right) m \right] \quad m = 0, 1, 2, 3, \dots, M \quad (2.64)$$

La transformada coseno discreta tiene la propiedad de producir características que prácticamente no están correlacionadas entre sí. De tal manera que se pueden usar matrices diagonales de covarianza en lugar de las matrices ordinarias de covarianza, con lo que se reduce en gran medida el trabajo de cómputo y el número de parámetros a estimar. Además, la transformada coseno discreta proporciona un suavizado del espectro si solamente se conservan los primeros coeficientes.

Por lo tanto, la solución final es el vector o el conjunto de coeficientes MFCC:

$$\begin{aligned}\mathbf{V}_{mfcc} &= C_m \\ m &= 1, 2, 3, \dots, M \\ \mathbf{V}_{mfcc} &= [C_1, C_2, C_3, \dots, C_M]\end{aligned}\tag{2.65}$$

El coeficiente inicial c_0 representa el logaritmo de la energía promedio en la trama y se descarta, pues ésta se puede calcular directamente a partir de la señal de voz.

- Coeficientes delta y doble deltas

Los cambios temporales en el espectro, tienen una importancia significativa en la percepción humana, una manera de capturar estos cambios es utilizando los coeficientes delta cuya finalidad es medir el cambio de los coeficientes en el tiempo, Huang et al. (2001). Las características a usar en un reconocedor usando MFCC con F_s de 16 KHz, es la siguiente:

Por frame analizado (ventaneamiento) se tendrá un vector X_k , donde:

$$X_k = \begin{pmatrix} C_k \\ \Delta C_k \\ \Delta \Delta C_k \end{pmatrix}\tag{2.66}$$

- C_k son los 13 coeficientes MFCC

- 13 coeficientes delta MFCC

$$\Delta C_k = C_{k+2} - C_{k-2}\tag{2.67}$$

- 13 coeficientes doble delta MFCC

$$\Delta \Delta C_k = \Delta C_{k+1} - \Delta C_{k-1}\tag{2.68}$$

2.1.2.3. Reconocimiento

Una vez terminada la fase de extracción de características, debemos empezar con la tarea del reconocimiento automático. Al igual que en la extracción de características, el proceso de reconocer una señal de voz por una persona también es motivo del estudio de múltiples algoritmos que realizan esta tarea, tales como el DTW (Alineamiento Temporal Dinámico), HMM (Modelos Ocultos de Markov), ANN (Redes Neuronales), etc. ver en (De Luna et al., 2006).

Este bloque dentro de los sistemas de reconocimiento de locutor, consiste en la comparación de parámetros característicos o patrones de voz. Como se explicó en el diagrama de bloques en la Figura 2.6, es necesario hacer la comparación de patrones de referencia contra los patrones de prueba y en base a una lógica de decisión reconocer al locutor.

Una característica fundamental de los sistemas de reconocimiento es la forma en que los vectores característicos son combinados y comparados con los patrones de referencia. Para poder realizar estas operaciones es necesario definir una medida de distancia o distorsión entre los vectores característicos. Si las características a medir tienen la misma longitud, pues es fácil realizar esta operación. Sin embargo, el problema se presenta si tenemos dos patrones que corresponden a la misma palabra, pero pronunciadas a diferentes velocidades, este análisis se hace inadecuado.

Es por ello que en este capítulo estudiaremos la técnica de *Alineación Dinámica en el Tiempo*

o *Alineamiento Temporal Dinámico* (DTW) para comparar estos patrones, la cual a su vez servirá para construir los patrones de referencia. También se estudian las medidas de distorsión o similitud que se emplean. Además, de la lógica de decisión para reconocer a un locutor.

a) Alineamiento Temporal Dinámico

Se sabe que una palabra es más compleja que un sonido fijo y sus características espectrales varían lentamente en el tiempo. Por lo tanto, si se analiza sobre períodos cortos, estas características se vuelven estacionarias. Como resultado del análisis, se obtienen patrones de voz para cada segmento analizado. Estos patrones, son parámetros que representan las características espectrales del tracto vocal durante ese segmento de voz. El problema de realizar una comparación punto a punto de estos patrones se asocia a las variaciones que existen al pronunciar varias veces una palabra, aún cuando la palabra ha sido pronunciada por el mismo locutor.

Tales variaciones se deben a la rapidez de pronunciación, a la duración de la palabra completa, a los diferentes acentos y estilos de pronunciación. Todas esto influye notablemente en el resultado de la comparación entre los patrones de voz. Por este motivo, se necesita llevar a cabo una alineación en el tiempo de las variaciones de pronunciación para que el resultado de la comparación de patrones tenga un significado útil al momento de tomar la decisión de reconocer a un locutor.

Para entender lo anterior, primero explicaremos el concepto de alineación en el tiempo de una palabra de prueba contra una de referencia y posteriormente la comparación de patrones de voz. En la Figura 2.42 se muestra el problema de alineación en el tiempo.

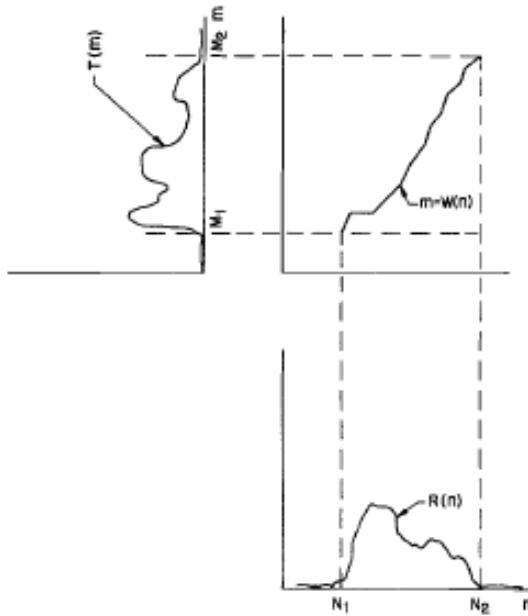


Figura 2.42: Ilustración general de alineamiento en el tiempo de dos palabras.

Fuente: Rabiner and Sambur (1975)

Se denota la palabra de referencia como $R(n)$, $0 \leq n \leq N$ y la palabra de prueba como $T(m)$, $0 \leq m \leq M$. El inicio y fin de $R(n)$ se denota como N_1 y N_2 , y el inicio y fin de $T(m)$ como M_1 y M_2 . El propósito del algoritmo de alineamiento en el tiempo es el de proveer un mapeo entre los índices temporales n y m para obtener una correspondencia en el tiempo entre las palabras de prueba y referencia. El mapeo w o también llamado función

de alineamiento (warping) entre n y m se denota como:

$$m = w(n) \quad (2.69)$$

Donde w debe satisfacer ciertas condiciones en el inicio y fin de las palabras. En el ejemplo de la Figura 2.42, se asume que los puntos de inicio y fin de las palabras están alineados en el tiempo, es decir:

$$M_1 = w(N_1) \quad (2.70)$$

$$M_2 = w(N_2) \quad (2.71)$$

A estas condiciones de las Ecuaciones (2.70) y (2.71) se les llama conjunto restringido en los límites. Como se mencionó anteriormente, el análisis de una palabra se hace en períodos cortos, generalmente de 5 a 100 ms., por lo que para cada segmento de voz de la palabra analizada se obtienen parámetros característicos (en nuestro caso, los MFCC) con los cuales se construyen los patrones de voz para el locutor que pronunció la palabra.

Ahora, si se consideran dos patrones de voz, X y Y , representados por las secuencias (x_1, x_2, \dots, x_n) y (y_1, y_2, \dots, y_m) respectivamente, donde x_n y y_m son vectores que contienen los parámetros característicos de cada segmento de voz analizado, N y M son el número de segmentos en cada palabra y no necesariamente $N = M$.

El propósito del alineamiento en el tiempo de patrones de voz es el de obtener una función

de alineamiento w entre los índices temporales n y m con la cual se logre una correspondencia entre los patrones de voz de la palabra de prueba y los patrones de voz de la palabra de referencia. En la Figura 2.43, se muestra un ejemplo de lo anterior donde se tienen los patrones de voz de la palabra de referencia en el eje de las ordenadas y los patrones de voz de la palabra de prueba en el eje de las abscisas. Se observa cómo la función de alineamiento proporciona una correspondencia entre los vectores x_n y y_m de los patrones de voz.

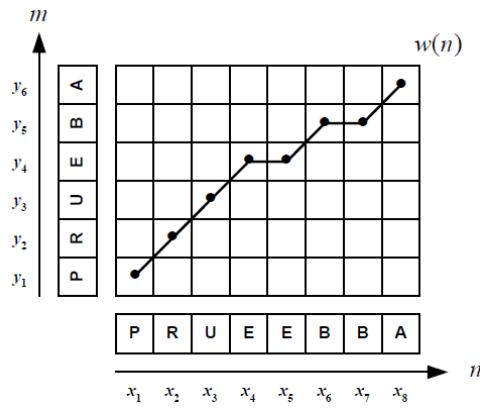


Figura 2.43: Ejemplo de alineación de patrones de voz.

Fuente: Rabiner and Sambur (1975)

La función de alineamiento debe cumplir con las Ecuaciones (2.70) y (2.71), además de cumplir con ciertas suposiciones acerca de la forma de $w(n)$. Por ejemplo, si $w(n)$ es una función lineal, el alineamiento en el tiempo es simplemente una compresión o expansión de una escala en el tiempo. Esta función lineal asume que la variación en la pronunciación de una palabra es proporcional a la duración de la misma y es independiente del sonido

hablado y por consiguiente no modela adecuadamente la forma en que se pronuncia una palabra, sobre todo no modela las variaciones de velocidad con que se pronuncian los sonidos vocales, por lo que es necesario emplear una función de alineamiento no lineal entre los patrones de voz.

Un enfoque más sofisticado y poderoso para alinear en el tiempo, consiste en restringir a $w(n)$ para que satisfaga un conjunto de condiciones locales de continuidad,

$$w(n+1) - w(n) = 0, 1, 2 (w(n) \neq w(n-1)) \quad (2.72)$$

$$w(n+1) - w(n) = 1, 2 (w(n) = w(n-1)) \quad (2.73)$$

estas ecuaciones hacen que $w(n)$ sea monotónicamente creciente con una pendiente máxima de 2 y una pendiente mínima de 0, excepto cuando la pendiente anterior fue 0, en tal caso la pendiente mínima será de 1.

Las condiciones de las Ecuaciones (2.70) y (2.71) junto con las condiciones de continuidad de las Ecuaciones (2.72) y (2.73) limitan a la función de alineamiento a seguir una trayectoria dentro del paralelogramo formado en el plano (n, m) como se muestra en la Figura 2.44. Los vértices, A y B , del paralelogramo se obtiene de la intersección de las líneas,

$$\begin{aligned} m - 1 &= 2(n - 1) && \text{Vértice A} \\ m - M &= (n - N)/2 \end{aligned} \quad (2.74)$$

y

$$m - 1 = (n - 1)/2 \quad \text{Vértice B} \quad (2.75)$$

$$m - M = 2(n - N)$$

La función de alineamiento $w(n)$, está restringida a seguir una trayectoria dentro de la región sombreada de la Figura 2.44.

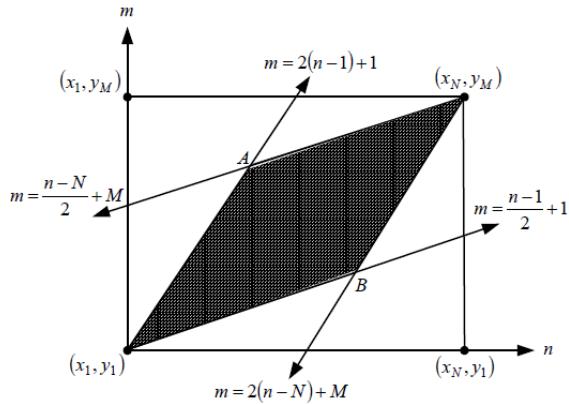


Figura 2.44: Región de posibles trayectorias de la función de alineamiento.

Fuente: Rabiner and Sambur (1975)

Para obtener la trayectoria de la función de alineamiento es necesario considerar una medida de similitud o distancia entre X y Y , la cual se define considerando una distancia local $d(x_n, y_m)$. Esta distancia local se calcula para todos los vectores en el eje n contra todos los vectores en el eje m como se muestra en la Figura 2.45. Por simplicidad esta distancia local será denotada como $d(n, m)$, donde $n = 1, 2, \dots, N$, $m = 1, 2, \dots, M$ y generalmente $N \neq M$. Entre más pequeño sea el valor de d , mayor es la similitud entre x_n y y_m .

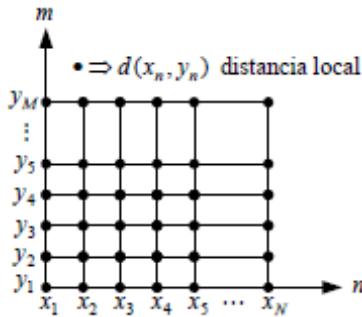


Figura 2.45: Distancias locales generadas por los patrones de voz.

Fuente: Rabiner and Sambur (1975)

Dada la función de distancia local d , la trayectoria óptima de la función de alineamiento $w(n)$ acotada por el paralelogramo de la Figura 2.44, debe de ser tal que la distancia acumulada total D_T desde el punto (x_1, y_1) al punto (x_N, y_M) siguiendo a $w(n)$ sea mínima, es decir:

$$D_T = \min_{\{w(n)\}} \sum_{n=1}^N d(n, w(n)) \quad (2.76)$$

Donde $d(n, w(n))$ es la distancia local entre el segmento n del patrón de referencia y el segmento $w(n)$ del patrón de prueba.

Hasta este punto se ha mencionado la trayectoria óptima de la función de alineamiento, sin embargo, no se ha establecido un método para obtenerla. La técnica para determinarla es mediante el método de programación dinámica. Usando esta técnica, la distancia mínima acumulada a cualquier punto en el plano (n, m) puede ser determinada en forma recursiva:

$$D_A(n, m) = d(n, m) + \min_{q \leq m} D_A(n - 1, q) \quad (2.77)$$

$n = 1, 2, \dots, N$

$m = 1, 2, \dots, M$

Donde $D_A(n - 1, q)$ es la distancia mínima acumulada al punto $(n - 1, q)$, y q puede tomar valores de $q = m, m - 1, m - 2$. Dadas las restricciones de continuidad de las Ecuaciones (2.72) y (2.73) y la Ecuación (2.77) puede ser escrita de la siguiente forma:

$$D_A(n, m) = d(n, m) + \min[D_A(n - 1, m)g(n - 1, m), \\ D_A(n - 1, m - 1), D_A(n - 1, m - 2)] \quad (2.78)$$

Donde:

$$g(n, m) = \begin{cases} 1 & w(n) \neq w(n - 1) \\ \infty & w(n) = w(n - 1) \end{cases}$$

En la Figura 2.46, se muestra un ejemplo gráfico de las restricciones de continuidad, además, se muestra una restricción no lineal ya que si la mejor trayectoria al punto $(n - 1, m)$ viene del punto $(n - 2, m)$, entonces ninguna trayectoria puede salir del punto $(n - 1, m)$, a estas restricciones se les conoce como restricciones locales del tipo Itakura.

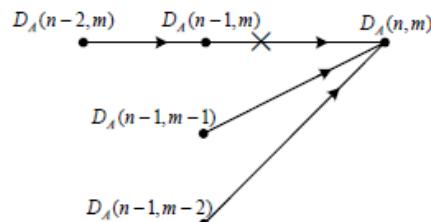


Figura 2.46: Conjunto de posibles trayectorias hacia el punto (n, m) .

Fuente: Rabiner and Sambur (1975)

La función de alineamiento se obtiene a partir de la Ecuación (2.77). Esta función contiene los índices m donde $D_A(n - 1, q)$ es mínima para $q = m, m - 1, m - 2$ y $n = 2, 3, \dots, N$,

en otras palabras, la función de alineamiento contiene los índices para los cuales existe una correspondencia entre el índice n del patrón de referencia y el índice m del patrón de prueba.

La iteración de la ecuación se realiza para todos los índices m válidos, es decir, todos los índices m dentro del paralelogramo de la Figura 2.44, y para cada valor de n , desde $n = 1$ hasta $n = N$. La solución final o distancia total está dada como:

$$D_T = D_A(N, M) \quad (2.79)$$

Las Ecuaciones (2.78) y (2.79) definen la técnica de programación dinámica para el alineamiento en el tiempo de patrones de voz. Al aplicarla a dos patrones de voz, se obtiene la medida de distancia total y la función de alineamiento. La función de alineamiento se usa en la etapa de entrenamiento, mientras que la distancia total entre los patrones de voz se emplea en la etapa de prueba.

A este método de alineación en el tiempo se le conoce como inicio y fin con restricción, rango de la pendiente 2 a 1 (Constrained Endpoints 2 to 1 slope range, CE2-1) donde se asume que existe un alineamiento perfecto entre los puntos de inicio y fin de los patrones de voz de prueba y referencia. En la Figura 2.47, se ilustra este método.

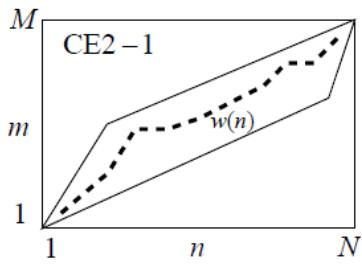


Figura 2.47: Técnica de alineación dinámica en el tiempo CE2-1.

Fuente: Rabiner and Sambur (1975)

Existen dos variantes del método explicado anteriormente. La primera variante se le conoce como inicio y fin sin restricción, rango de la pendiente 2 a 1 (Unconstrained Endpoints 2 to 1 slope range, UE2-1), ver la Figura 2.48, se utiliza cuando se tiene cierto grado de incertidumbre en la detección del inicio y fin de la palabra.

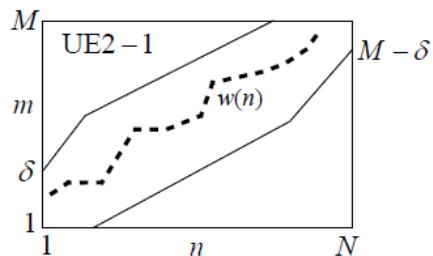


Figura 2.48: Técnica de alineación dinámica en el tiempo UE2-1.

Fuente: Rabiner and Sambur (1975)

La segunda variante se le conoce como inicio y fin sin restricción, mínimos locales (Unconstrained Endpoints Local Minimum, UELM), ver la Figura 2.49. En este método, $w(n)$

se restringe a seguir la trayectoria local óptima dentro de un rango especificado. El propósito de esta restricción adicional es el de reducir los cálculos ya que este algoritmo sigue la trayectoria local óptima para estimar la trayectoria global óptima.

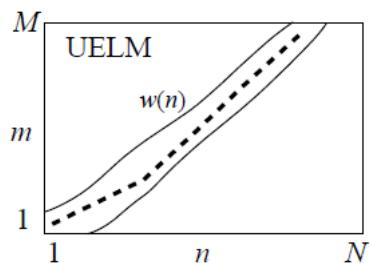


Figura 2.49: Técnica de alineación dinámica en el tiempo UELM.

Fuente: Rabiner and Sambur (1975)

En Rabiner et al. (1978) se muestran comparaciones de estas tres variantes aplicadas en el reconocimiento de palabras aisladas y se concluye que:

- La variante CE2-1 genera distancias totales más grandes que la variante UE2-1.
- El algoritmo UE2-1 sirve para reducir la distancia total y para mejorar el desempeño del método.
- El algoritmo UELM se usa para aplicaciones en las cuales solamente se conoce el inicio de la palabra, por ejemplo, reconocimiento utilizando palabras concatenadas.

Según Myers et al. (1980) se generan pequeñas pero consistentes mejoras en la exactitud de reconocimiento cuando el patrón de prueba está sobre la abscisa.

En (Furui, 1981) se muestra que se produce menor tasa de error en la verificación de locutor cuando la palabra de menor duración se utiliza sobre la abscisa, independientemente que esta sea una palabra de prueba o de referencia.

A partir de estos antecedentes, se ha optado para este trabajo de tesis el estudio de UE2-1 y UELM, a continuación explicaremos a detalle el algoritmo DTW.

- Restricciones de la función warping

Tal como vimos anteriormente, la función warping, es una medida de la fluctuación en el eje del tiempo de los patrones del habla, la función $F = c(1), c(2), \dots, c(K)$, puede ser vista como una función de mapeo del patrón A en el patrón B , quien debería conservar las estructuras lingüísticas esenciales en el patrón A y viceversa, ver la Figura 2.50.

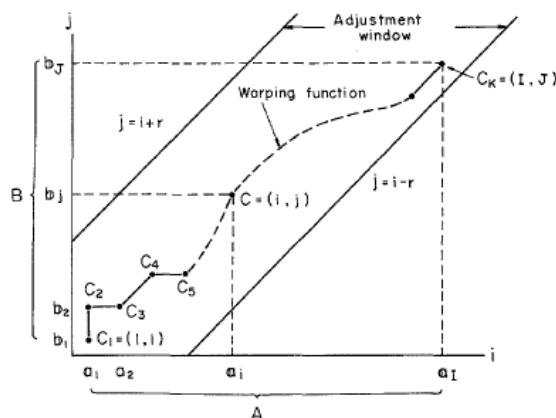


Figura 2.50: Función warping y ventana de ajuste.

Fuente: Sakoe and Chiba (1978)

Las restricciones de la función warping F , son las siguientes:

- Condición de monoticidad:

$$i(k-1) \leq i(k) \quad \text{y} \quad j(k-1) \leq j(k) \quad (2.80)$$

- Condición de continuidad:

$$i(k) - i(k-1) \leq 1 \quad \text{y} \quad j(k) - j(k-1) \leq 1 \quad (2.81)$$

como resultado de estas restricciones, la relación entre dos puntos consecutivos está dada:

$$c(k-1) = \begin{cases} (i(k), j(k-1)) \\ (i(k-1), j(k-1)) \\ (i(k-1), j(k)) \end{cases} \quad (2.82)$$

- Condición de frontera:

$$i(1) = 1, j(1) = 1 \quad \text{y} \quad i(K) = I, j(K) = J \quad (2.83)$$

- Condición de ventana de ajuste:

$$|i(k) - j(k))| \leq r \quad (2.84)$$

Donde r es un apropiado valor entero no negativo que indica el tamaño de la ventana de ajuste, esto se debe al hecho en que las fluctuaciones en el eje del tiempo, en algunos casos nunca causan excesiva diferencia.

- Condición de slope constraint:

No se deben permitir gradientes ni muy pronunciadas, ni muy suaves para la función F , pues puede causar desviaciones no deseables en el eje del tiempo, para gradientes pronunciadas puede causar una correspondencia no real entre un patrón muy corto y otro muy largo.

La condición de slope constraint es establecida como la primera derivada de la función warping en su forma discreta, así se obliga que los valores $c(k)$ se muevan en dirección hacia adelante en el eje i o en el eje j , consecutivamente m veces, entonces no se debe permitir que $c(k)$ vaya en la misma dirección a menos que haya pasado n veces por la diagonal, la intensidad efectiva del slope constraint puede ser evaluada por la siguiente medida:

$$p = \frac{n}{m} \quad (2.85)$$

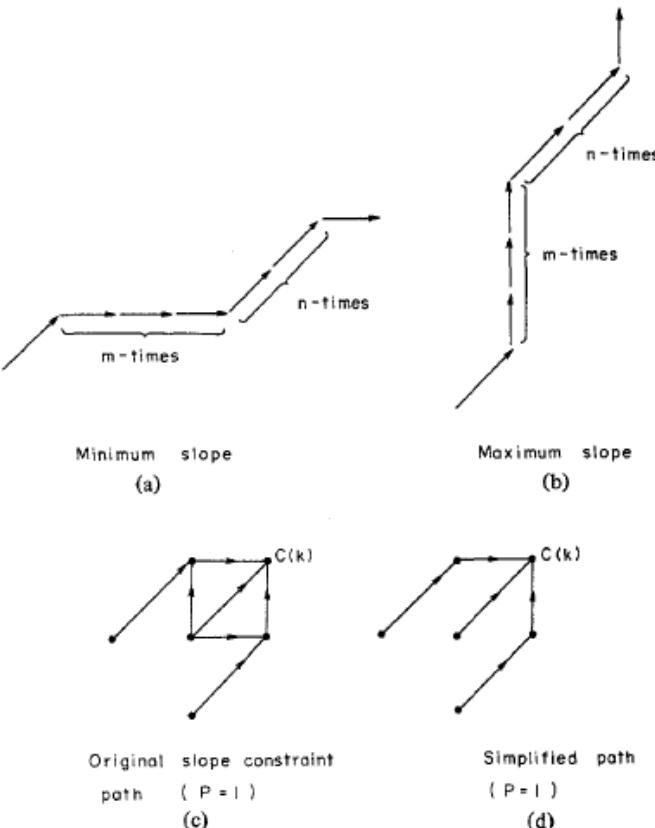


Figura 2.51: Slope constraint en función warping.

Fuente: Sakoe and Chiba (1978)

Cuando $P = 0$, no existen restricciones en la función warping, cuando $P = \infty$ que significa $m = 0$ la función warping está restringida a la diagonal $i = j$, si el slope constraint es muy severo, entonces la normalización en el tiempo no podrá trabajar adecuadamente y si el slope constraint es muy relajado, entonces la discriminación entre patrones de diversas categorías es degradada, por ello es deseable un valor ni muy pequeño, ni muy grande de p .

- Coeficientes de peso

La expresión:

$$N = \sum_{k=1}^K w(k) \quad (2.86)$$

que actúa como denominador del cálculo de la distancia normalizada:

$$D(A, B) = \min \left[\frac{\sum_{k=1}^K d(c(k)) \cdot w(k)}{\sum_{k=1}^K w(k)} \right] \quad (2.87)$$

es independiente de la función warping F , luego la distancia normalizada se puede escribir como:

$$D(A, B) = \frac{1}{N} \min \left[\sum_{k=1}^K d(c(k)) \cdot w(k) \right] \quad (2.88)$$

y puede ser efectivamente resuelto por la técnica de programación dinámica. Existen dos definiciones de coeficientes de pesos:

- Forma simétrica:

$$w(k) = (i(k) - i(k-1)) + (j(k) - j(k-1)) \quad (2.89)$$

entonces $N = I + J$, donde I y J son las longitudes de los parámetros de habla A y B respectivamente.

- Forma asimétrica:

$$w(k) = (i(k) - i(k-1)) \quad (2.90)$$

entonces $N = I$, o equivalentemente:

$$w(k) = j(k) - j(k-1) \quad (2.91)$$

entonces $N = J$.

La forma simétrica significa que, si los ejes del tiempo i y j son continuos, entonces existirá un eje temporal: $l = i + j$; en la forma asimétrica se refiere a la integración a través del eje i o j según sea el caso. Según (Sakoe and Chiba, 1978), se espera que trabaje mejor la forma simétrica que la asimétrica, pues trata las partes del vector de características de igual manera, y la forma asimétrica hace alguna exclusión cuando la función warping está en dirección del eje i o j según sea el caso, pues $w(k)$ se reduce a cero: $c(k) = c(k-1) + (0, 1)$, esto significa que algunos vectores pueden ser excluidos del análisis, pero por suerte la condición de slope constraint reduce esta situación, la diferencia de desempeño entre la forma simétrica y asimétrica puede ser gradualmente atenuada, del mismo modo que la condición de slope constraint es intensificada.

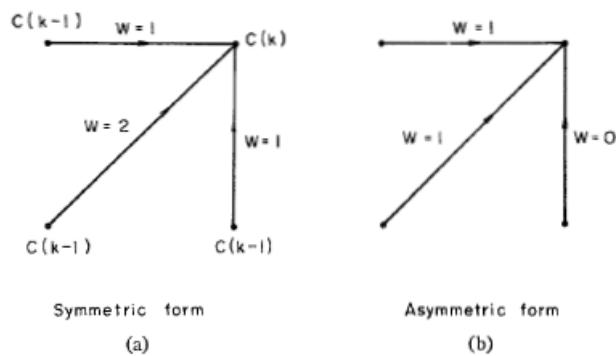


Figura 2.52: Coeficientes de pesos para la forma simétrica y la forma asimétrica.

Fuente: Sakoe and Chiba (1978)

- Algoritmo PD-Matching

El principio de PD (Programación Dinámica) puede ser aplicado a la definición de distancia normalizada en el tiempo, el algoritmo básico es descrito como sigue:

Condición inicial:

$$g_1(c(1)) = d(c(1))w(1) \quad (2.92)$$

Ecuaciones PD:

$$g_k(c(k)) = \min_{c(k-1)} [g_{k-1}(c(k-1)) + d(c(k))w(k)] \quad (2.93)$$

Distancia normalizada en el tiempo:

$$D(A, B) = \frac{1}{N} g_k(c(k)) \quad (2.94)$$

Se asume que $c(0) = (0, 0)$ y $w(1) = 2$ para la forma simétrica y $w(1) = 1$ para la forma asimétrica.

Pueden derivarse varios algoritmos prácticos al aplicar la forma simétrica y la forma asimétrica según la condición de slope constraint que utilicen, algunos de ellos se detallan a continuación:

Para el algoritmo de la forma simétrica la condición inicial de la Ecuación (2.92) quedaría así:

$$g(1, 1) = 2d(1, 1) \quad (2.95)$$

Ecuación PD: para $p = 0$ en su forma simétrica

$$g(i, j) = \min \begin{cases} g(i, j - 1) + d(i, j) \\ g(i - 1, j - 1) + 2d(i, j) \\ g(i - 1, j) + d(i, j) \end{cases} \quad (2.96)$$

Condición de restricción (ventana de ajuste):

$$j - r \leq i \leq j + r \quad (2.97)$$

Distancia normalizada en el tiempo:

$$D(A, B) = \frac{1}{N} g(I, J) \quad (2.98)$$

Las ecuaciones de PD ($g(i, j)$) se calculan en orden ascendente con respecto a las coordenadas i y j , es decir empiezan de (i, j) hasta (I, J) .

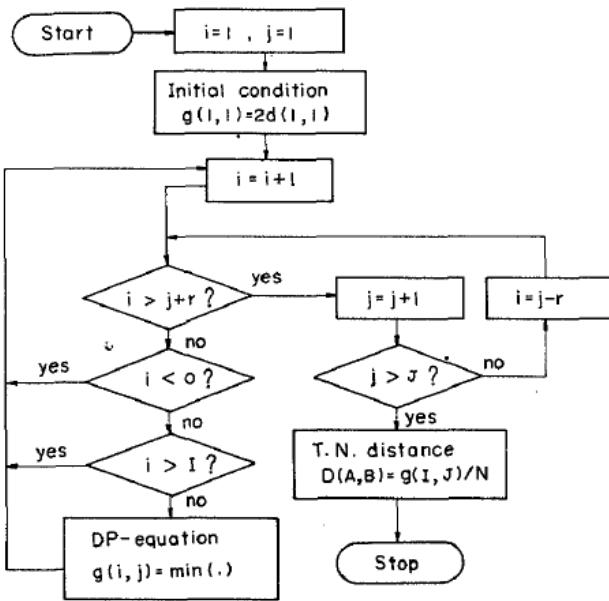


Figura 2.53: Diagrama de flujo del algoritmo PD-Matching.
Fuente: Sakoe and Chiba (1978)

Experimentos realizados en (Sakoe and Chiba, 1978) muestran que el desempeño de la forma simétrica es superior al de la forma asimétrica, pero esta diferencia disminuye a medida que la condición de slope constraint es intensificada, se puede notar también que el desempeño de la forma simétrica no es afectado por un slope constraint superior a $P = 1$, por otro lado, la forma asimétrica es visiblemente mejorada por la condición de slope constraint.

Otro experimento hecho también en (Sakoe and Chiba, 1978) evalúa el efecto de la condición de slope constraint en la forma simétrica del algoritmo PD-matching, y llega

al resultado de que cuando $P = 1$ se obtiene el mejor desempeño, y en otro experimento evalúa este algoritmo con otros varios algoritmos PD propuestos por otros autores, y muestra la superioridad del algoritmo PD-matching con slope constraint $P = 1$.

En conclusión, se tiene que los mejores resultados se obtienen con la forma simétrica del algoritmo PD-matching y con una condición de slope constraint $P = 1$.

Tabla 2.9: Algoritmos simétricos y asimétricos con condición de Slope Constraint $P = 0, 1/2, 1, 2$.

P	Schematic explanation	Symmetric	DP-equation $g(i, j) =$
		Asymmetric	
0		Symmetric	$\min \begin{bmatrix} g(i, j-1)+d(i, j) \\ g(i-1, j-1)+2d(i, j) \\ g(i-1, j)+d(i, j) \end{bmatrix}$
		Asymmetric	$\min \begin{bmatrix} g(i, j-1) \\ g(i-1, j-1)+d(i, j) \\ g(i-1, j)+d(i, j) \end{bmatrix}$
1/2		Symmetric	$\min \begin{bmatrix} g(i-1, j-3)+2d(i, j-2)+d(i, j-1)+d(i, j) \\ g(i-1, j-2)+2d(i, j-1)+d(i, j) \\ g(i-1, j-1)+2d(i, j) \\ g(i-2, j-1)+2d(i-1, j)+d(i, j) \\ g(i-3, j-1)+2d(i-2, j)+d(i-1, j)+d(i, j) \end{bmatrix}$
		Asymmetric	$\min \begin{bmatrix} g(i-1, j-3)+(d(i, j-2)+d(i, j-1)+d(i, j))/3 \\ g(i-1, j-2)+(d(i, j-1)+d(i, j))/2 \\ g(i-1, j-1)+d(i, j) \\ g(i-2, j-1)+d(i-1, j)+d(i, j) \\ g(i-3, j-1)+d(i-2, j)+d(i-1, j)+d(i, j) \end{bmatrix}$
1		Symmetric	$\min \begin{bmatrix} g(i-1, j-2)+2d(i, j-1)+d(i, j) \\ g(i-1, j-1)+2d(i, j) \\ g(i-2, j-1)+2d(i-1, j)+d(i, j) \end{bmatrix}$
		Asymmetric	$\min \begin{bmatrix} g(i-1, j-2)+(d(i, j-1)+d(i, j))/2 \\ g(i-1, j-1)+d(i, j) \\ g(i-2, j-1)+d(i-1, j)+d(i, j) \end{bmatrix}$
2		Symmetric	$\min \begin{bmatrix} g(i-2, j-3)+2d(i-1, j-2)+2d(i, j-1)+d(i, j) \\ g(i-1, j-2)+2d(i, j) \\ g(i-3, j-2)+2d(i-2, j-1)+2d(i-1, j)+d(i, j) \end{bmatrix}$
		Asymmetric	$\min \begin{bmatrix} g(i-2, j-3)+2(d(i-1, j-2)+d(i, j-1)+d(i, j))/3 \\ g(i-1, j-2)+d(i, j) \\ g(i-3, j-2)+d(i-2, j-1)+d(i-1, j)+d(i, j) \end{bmatrix}$

Fuente: Sakoe and Chiba (1978)

b) Distancias o medidas de distorsión

Un componente clave en la mayoría de los algoritmos de comparación de patrones, es formular una medida de distorsión entre dos vectores característicos. Esta medida, puede ser manejada con rigor matemático si los patrones son visualizados en un espacio vectorial.

Suponer que se tienen dos vectores característicos, x e y , definidos en un espacio vectorial X . Se define una métrica o función de distancia, d , en el espacio vectorial X , como una función de valor real, sobre el producto cartesiano $X * X$, para que una función de distancia pueda tratarse con rigor matemático, debe satisfacer las siguientes condiciones:

1. $0 \leq d(x, y) < \infty$, para $x, y \in X$ y $d(x, y) = 0$ sí y solo sí $x = y$
2. $d(x, y) = d(y, x)$ para $x, y \in X$
3. $d(x, y) \leq d(x, z) + d(z, y)$ para $x, y, z \in X$

Además, una función de distancia se denomina invariante si:

4. $d(x + z, y + z) = d(x, y)$

Las primeras tres propiedades comúnmente son conocidas como positividad, simetría y desigualdad del triángulo, respectivamente. Una métrica que contenga estas propiedades, permite un alto grado de manejo matemático. Sí una medida de distancia d , satisface solo la

propiedad de positividad, se le denomina medida de distorsión, particularmente cuando los vectores son representaciones del espectro de la señal.

Para el procesamiento de voz es importante considerar que la elección, de la medida de distancia, es significativamente subjetiva. Una medida matemática de la distancia, para ser utilizada en el procesamiento de voz, debe tener una alta correlación entre su valor numérico y su distancia subjetiva aproximada, para evaluar una señal real de voz.

Para el reconocimiento de voz, la consistencia psicofísica (los diferentes matices que se le pueden imprimir a una misma palabra) que se desea medir con la distancia, obliga a que se encuentre una medida matemática ajustada por necesidad a las características lingüísticas conocidas. Estos requisitos tan subjetivos no pueden ser satisfechos con medidas de distancia que proporcionen manejo matemático, dado que existe una enorme dificultad al querer cumplir simultáneamente ambos objetivos (subjetividad y manejo matemático), algún compromiso es inevitable. Por consiguiente, y dado que se necesita manipular matemáticamente las propiedades de esa medida de distancia, se necesita probar que estas propiedades subjetivas son lo suficientemente buenas como para lograr el reconocimiento de voz.

Por otra parte, se hablará de *medidas de distorsión* en vez de *métricas* o *distancias* debido a que se relajan las condiciones de simetría y desigualdad del triángulo. Por lo tanto, no se

debe utilizar el término distancia en sentido estricto, acorde a la definición expuesta; por otro lado, en la costumbre de la literatura de voz el término distancia es análogo a las medidas de distorsión.

Existen varios tipos de medidas de distorsión, cada una con sus características especiales, entre ellas tenemos: *Distancia Euclíadiana Cuadrática*, *Distorsión del Error Cuadrático Medio*, *Distorsión del Error Cuadrático Ponderado*, *Distancia de Itakura*, *Distancia de Mahalanobis*, etc., ver en (Navarrete, 2013).

Para este trabajo de tesis veremos solo la Distancia Euclíadiana Cuadrática y la Distancia del Error Cuadrático Medio, debido a su sencillez y poca complejidad, además de complementarse muy bien con el algoritmo DTW.

- Distancia Euclíadiana Cuadrática

La medida más conveniente y ampliamente usada para calcular distancias, es el *Error Cuadrático* o *Distancia Euclíadiana Cuadrática*, entre dos vectores, definida como:

$$d(X_1, X_2) = \|X_1 - X_2\|^2 = \sum_{j=1}^N (X_{1j} - X_{2j})^2 \quad (2.99)$$

- Distorsión del Error Cuadrático Medio

La distorsión del *Error Cuadrático Medio* (MSE) es otra de las medidas más utilizadas,

en la cual la distorsión está definida por cada dimensión y se define como:

$$d(X_1, X_2) = \frac{1}{N} (X_1 - X_2)^T (X_1 - X_2) = \frac{1}{N} \sum_{j=1}^N (X_{1j} - X_{2j})^2 \quad (2.100)$$

c) Construcción del patrón de referencia

El sistema de identificación de locutor se divide en dos etapas, etapa de prueba y etapa de entrenamiento. En la etapa de prueba, basta con obtener la distancia total D_T entre los patrones de voz. Con esta distancia total y una lógica de decisión se lleva a cabo la tarea de identificar a un locutor, esta lógica de decisión se verá más adelante. Sin embargo, para construir un patrón de referencia confiable para un locutor, en la etapa de entrenamiento, es necesario conocer la función de alineamiento, $w(n)$. Como se observa en la Figura 2.50, la función de alineamiento provee una correspondencia entre los índices de los patrones de voz. Utilizando esta correspondencia entre los patrones de voz, se puede crear un patrón de referencia z , simplemente promediando los vectores de los patrones de voz siguiendo la función de alineamiento, es decir:

$$z_n = \frac{X_n + Y_{w(n)}}{2} \quad n = 1, 2, \dots, N \quad (2.101)$$

En (Furui, 1981), el entrenamiento se hace de la siguiente manera: Primeramente, se tienen los patrones de voz para una palabra clave, pronunciada dos veces por el locutor. Se aplica la técnica de alineación dinámica en el tiempo y se obtiene la función de alineamiento para los dos patrones de voz.

Se utiliza la Ecuación (2.101), para crear el primer patrón de entrenamiento. Se repite este procedimiento para la misma palabra pronunciada por tercera ocasión, pero ahora comparada contra el primer patrón de entrenamiento. Se obtiene la función de alineamiento y utilizando de nuevo la Ecuación (2.101), se genera el segundo patrón de entrenamiento. Este procedimiento se sigue hasta que se ha generado el quinto patrón de entrenamiento, el cual ya se considera como un patrón de referencia. Para obtener un patrón de referencia más confiable, se utiliza un mayor número de repeticiones de la palabra clave.

d) Toma de Decisión

Después de realizar la comparación de patrones es necesario tener una lógica de decisión para identificar al locutor. Este es el último paso dentro de los sistemas de reconocimiento de locutor y consiste en escoger cuál patrón de referencia (locutores conocidos) mejor corresponde con el patrón de prueba (locutor desconocido).

La forma más simple de implementar lo anterior consiste en utilizar la regla del vecino más cercano, la cual opera de la siguiente manera: se tienen V patrones de referencia, R^i , $i = 1, 2, \dots, V$, donde cada patrón de referencia representa la identidad de un locutor, y para cada patrón de referencia se obtiene la distancia total D_T^i , $i = 1, 2, \dots, V$, respecto al patrón de prueba utilizando el algoritmo DTW. El patrón de referencia con la distancia mínima respecto al patrón de prueba corresponderá a la identidad del locutor.

Bueno, hasta aquí se ha considerado que el sistema es de conjunto cerrado, es decir, se tienen N usuarios y N decisiones del sistema. Sin embargo, esto da lugar a tres respuestas del sistema: que el sistema identifique a un usuario válido, que se equivoque en identificar a un usuario y que acepte a un intruso. Dado que se requiere que el sistema no acepte intrusos, se incluyen umbrales en el sistema con el fin de aumentar la seguridad del mismo. Con estos umbrales se pretende tener un sistema de N usuarios y $N + 1$ decisiones.

Es por ello que se utilizaran umbrales individuales para poder incluir en el sistema de identificación de locutor la respuesta *no identificado*. A cada locutor o patrón de referencia le corresponde un umbral. El cálculo de los umbrales se hace mediante un método explicado en (Varela, 1994), donde se define una distancia *intralocutor* y una distancia *interlocutor*, el primero se obtiene al comparar mediante el algoritmo DTW el patrón de referencia de un locutor contra las palabras habladas por el mismo, y el segundo se obtiene al comparar el patrón de referencia de un locutor contra las palabras habladas por la base de datos de locutores sin incluir las palabras que pertenecen al locutor del cual se usa el patrón de referencia.

Se supone que tanto el conjunto de distancias intralocutor, como el de distancias interlocutor pueden ser aproximados por distribuciones de probabilidad normal como se muestra en la Figura 2.54.

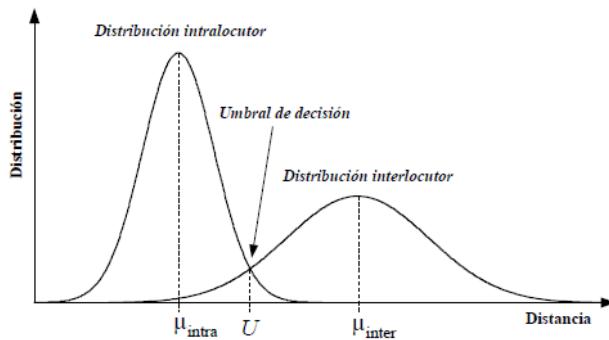


Figura 2.54: Distribuciones de probabilidad de las distancias intralocutor e interlocutor.

Fuente: Varela (1994)

El umbral U es el punto en donde el área bajo la curva de la distribución intralocutor desde U a más infinito es igual al área bajo la curva de la distribución interlocutor desde menos infinito a U , es decir, el punto donde la probabilidad de que el sistema responda *no identificado* es igual a la probabilidad de que exista una *falsa identificación*. Las distribuciones de probabilidad normal de la Figura 2.54. están dadas por la siguiente ecuación:

$$P(D) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(D-\mu)^2}{2\sigma^2}} \quad (2.102)$$

Donde μ es la media y σ es la desviación estándar. El umbral U se calcula a partir de las siguientes ecuaciones:

$$U - \mu_{intra} = n\sigma_{intra} \quad (2.103)$$

$$\mu_{inter} - U = n\sigma_{inter} \quad (2.104)$$

despejando n de las Ecuaciones (2.103) y (2.104) e igualando se obtiene el umbral U de la siguiente forma:

$$U = \frac{\sigma_{intra}\mu_{inter} + \sigma_{inter}\mu_{intra}}{\sigma_{intra} + \sigma_{inter}} \quad (2.105)$$

Si es que se tienen 10 patrones de referencia (10 usuarios del sistema), es necesario calcular un umbral por cada patrón de referencia, es decir:

$$U_i = \frac{\sigma_i^{intra}\mu_i^{inter} + \sigma_i^{inter}\mu_i^{intra}}{\sigma_i^{intra} + \sigma_i^{inter}} \quad i = 1, 2, \dots, 9 \quad (2.106)$$

2.2. Método de la investigación

2.2.1. Diseño de la investigación

De acuerdo al fin que persigue, el presente trabajo es una investigación aplicada. Por otro lado, de acuerdo al diseño de contrastación de la hipótesis es una investigación experimental.

2.2.2. Variables de estudio

- **Variable dependiente:** Control de acceso a una vivienda.
- **Variable independiente:** Sistema de seguridad por reconocimiento de voz.

El factor de medida para el control de acceso a una vivienda mediante el reconocimiento de voz consiste en el tiempo de respuesta y la tasa de precisión en el reconocimiento que se obtengan de los resultados experimentales.

2.2.3. Métodos y procedimientos para la recolección de datos

Para recolectar los datos se harán grabaciones a 10 personas escogidas aleatoriamente, para esto se contará con el micrófono del dispositivo móvil y el del computador. El procedimiento a emplear será darle a cada persona una lista de dos comandos de voz (*Abrir* y su *Nombre*), las cuales, al ser leídas, serán grabadas. El nivel de ruido de la grabación deberá ser moderado.

2.2.4. Población y muestra

Siendo una investigación experimental, la determinación de la muestra se basa en los antecedentes Pérez et al. (2013) y Varela (1994) debido a que estas investigaciones son de caso similar y en la experiencia del autor.

2.2.4.1. Población

- **Área:** Audios de señales de voz.
- **Categoría:** Palabras de idioma español.
- **Subcategoría:** Hombres con edad promedio de 23 años.

La población es infinita para esta investigación, debido al número infinito de formas distintas en que una señal de voz puede ser capturada. Se tomarán en cuenta audios donde se muestren señales de voz de palabras en español habladas por hombres con edad promedio de 23 años.

2.2.4.2. Muestra

La muestra estará conformada por un conjunto de 800 audios dividido entre los comandos de voz *Abrir* y los *Nombres de los Usuarios* (Edwin, Carlos, Josué, Édison, Gerson, Nizama, Anthony, Jhordan, Renzo y Franchesco), los cuales serán las palabras claves de acceso para 10 miembros de la vivienda, por lo que estará agrupada por el conjunto de estos 11 comandos de voz.

Dado que la población de señales de voz es infinita, se optó por usar un muestreo aleatorio simple para poblaciones desconocidas con un nivel de confianza del 95 % y un error de muestreo del 3.5 %.

$$n = \frac{z^2 pq}{e^2} \quad (2.107)$$

En el que:

- n = Tamaño de muestra.
- z = Coeficiente de confiabilidad 95 % al que corresponde (1.96).
- pq = Varianza de la población, ponemos la varianza mayor posible porque a mayor varianza hará falta una muestra mayor (0.25).
- e = Error muestral 3.5 % (0.035).

2.2.5. Método de estudio

Para llegar a los objetivos propuestos, el desarrollo de la investigación comprendió las siguientes etapas:

- a) Análisis del problema de los sistemas de control de acceso en la ciudad de Trujillo, en el Perú y el mundo, comprendiendo la situación actual y el levantamiento de los principales casos de éxito.
- b) Formulación del problema principal de la investigación y justificación de la importancia de su solución.
- c) Recopilación de herramientas, técnicas y métodos acústicos matemáticos y físicos de la voz para identificar y autenticar a los dueños de la vivienda.
- d) Levantamiento de los diferentes temas necesarios para la elaboración de la investigación.
- e) Diseño e implementación del software de reconocimiento de voz.
- f) Diseño y construcción del hardware capaz de ser controlado a través del protocolo de red inalámbrica 802.11g.
- g) Integración de las dos partes mencionadas anteriormente, así como la ejecución de pruebas para su correcto funcionamiento.

- h) La implementación de todo el sistema en un ambiente real.
- i) Realización del reporte de incidencias ocurridas mientras estaba en uso el sistema.
- i) Para el análisis de resultados, se pretende usar la técnica de Chi-Cuadrado de Pearson para dos variables nominales, a través de esta técnica se puede conocer la relación de dependencia de nuestras variables con Identificación Correcta, Falsa Identificación y No Identificación métricas usadas comúnmente para evaluar la calidad de nuestro sistema de reconocimiento de voz.

2.2.6. Análisis estadístico de los datos

Para efectuar el análisis estadístico de los datos se hará la prueba de independencia Chi- Cuadrado de Pearson para dos variables nominales.

El uso de esta prueba es que permite contrastar los resultados de una prueba o método propuesto, determinando si dos cualidades o variables referidas a individuos de una población están relacionadas.

2.2.7. Esquema general del proyecto

En la Figura 2.55 se muestra el diagrama de bloques para el esquema general de este proyecto, primero la señal voz es capturada por el micrófono del dispositivo móvil, en el caso de que se habilite la opción de filtrado de ruido en el sistema, el micrófono del computador capturara la

señal de ruido, luego el sistema se encargará de procesar y analizar la señal de voz (con la señal de ruido, si es que la hubiera) y dependiendo de la respuesta del reconocimiento de dicha señal este le enviará o no una señal al microcontrolador que controla el acceso, activando o no la cerradura eléctrica.

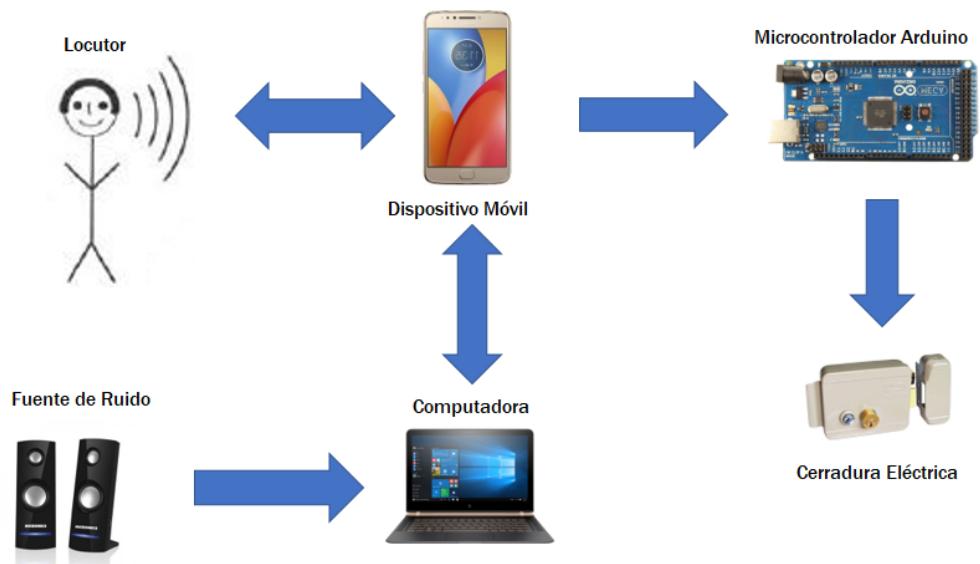


Figura 2.55: Diagrama de bloques general del proyecto.

Fuente: Elaboración propia

Capítulo 3

Sistema de control de acceso por reconocimiento de voz

Para el desarrollo del sistema se utilizará la metodología propuesta por (Noergaard, 2012) , la cual toma mucha de las claves de las metodologías conocidas como RUP (Rational Unified Process), ADD (Attribute Driven Desing), OPP (Object Oriented Process) y el MDA (Model Driven Architecture), pero las simplifica. La Figura 3.1 muestra las diferentes fases de la metodología, la cual es definida por el autor como el *Diseño del Sistema Empotrado y el Modelo del Ciclo de Vida del Desarrollo*.

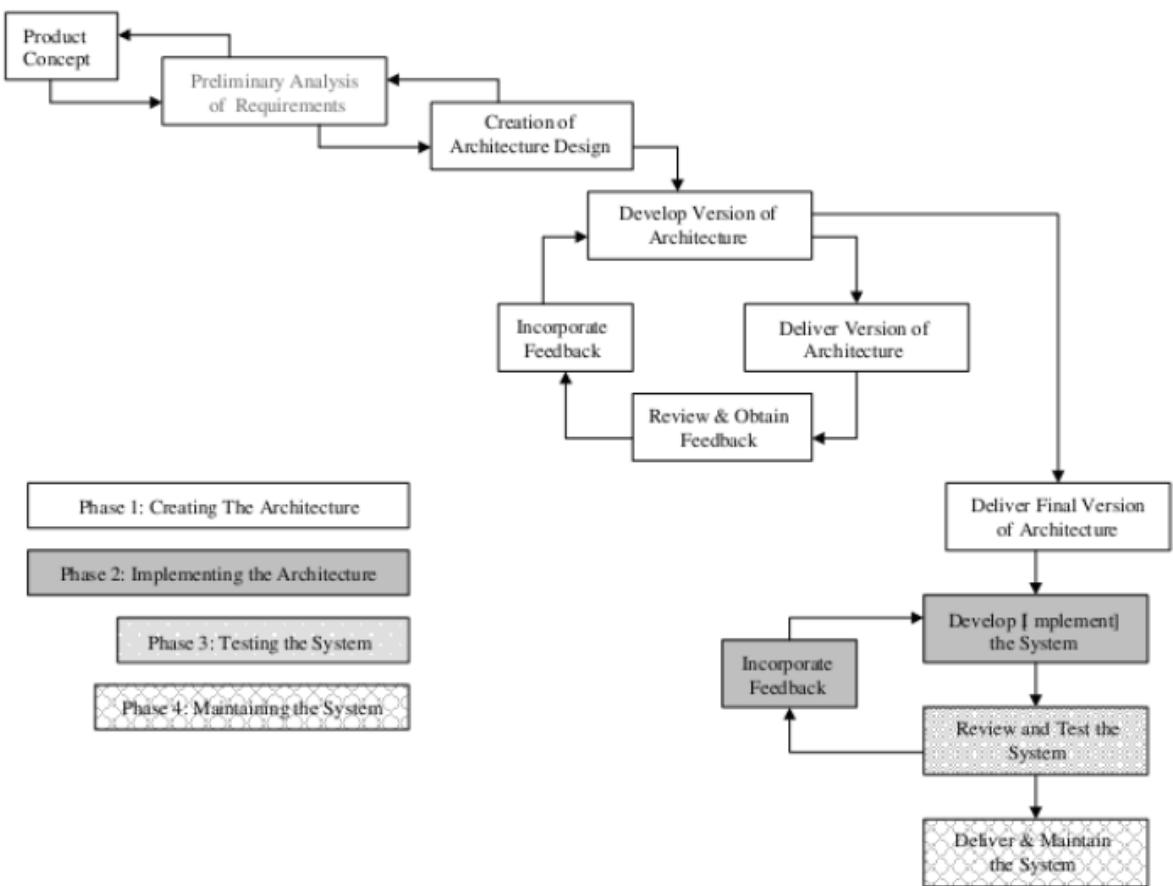


Figura 3.1: Diseño del sistema empotrado y el modelo del ciclo de vida del desarrollo.

Fuente: Noergaard (2012)

Como se puede ver en la Figura 3.1, en el diseño de un sistema empotrado, según la metodología de Noergaard (2012), se pueden distinguir 4 fases:

- **Fase 1. Creación de la arquitectura**, el cual es el proceso de planear el diseño del sistema empotrado.

- **Fase 2. Implementación de la arquitectura**, el cual es el proceso de desarrollar el sistema empotrado.
- **Fase 3. Testeo del sistema**, el cual es el proceso de testear el sistema empotrado para buscar problemas y luego solucionarlos.
- **Fase 4. Mantenimiento del sistema**, el cual es el proceso de desplegar el sistema empotrado en el mercado, y proveerlo de soporte técnico para los usuarios del dispositivo por la duración del tiempo de vida del mismo.

3.1. Fase 1. Creación de la arquitectura

Esta fase se divide en 3 etapas:

- Etapa 1: Tener una base técnica sólida.
- Etapa 2: Entender el ciclo de negocio de la arquitectura.
- Etapa 3: Definir los patrones arquitectónicos y modelos de referencia.

3.1.1. Etapa 1: Tener una base técnica sólida

Esta etapa tiene como fundamento la base cognitiva de los autores de la presente tesis y ha sido descrita tanto en el título, introducción y marco teórico.

3.1.2. Etapa 2: Entender el ciclo de negocio de la arquitectura

En la Figura 3.2 vemos que el ciclo de negocio de la arquitectura de un sistema empotrado son los diferentes tipos de influencias los que generan los requerimientos del sistema, los requerimientos a su vez generan la arquitectura, luego esta produce el sistema y el sistema resultante a su vez provee los requerimientos y capacidades devuelta a la organización para futuros diseños.

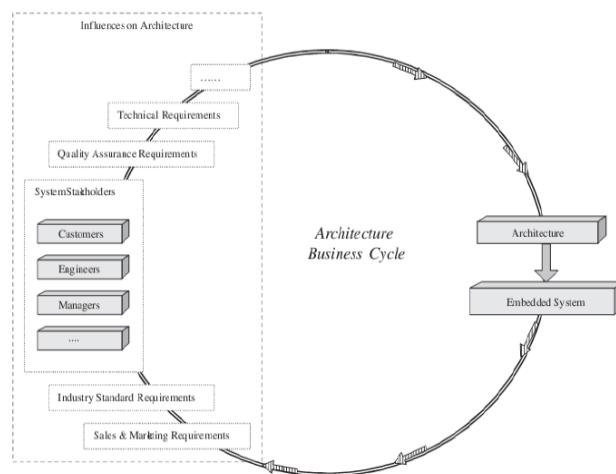


Figura 3.2: Ciclo de negocio de la arquitectura.

Fuente: Noergaard (2012)

Una vez entendido el ciclo de negocio de la arquitectura, se debe plasmar la información obtenida de varias influencias, que en otras palabras son los requerimientos. La presente metodología propone hacerlo mediante la tabla general de características del ciclo de negocio de la arquitectura o el uso de prototipos. Se eligió representar los requerimientos mediante la tabla general de características del ciclo de negocio de la arquitectura, ver la Tabla 3.1.

Tabla 3.1: Características generales del ciclo de negocio de la arquitectura.

Influencia	Característica	Descripción
Negocios	Costo	El sistema de seguridad para el control de acceso por medio de la voz, debe tener un costo de desarrollo menor a los S/. 500.
	Objetivo de Mercado	El sistema a diseñar es del tipo de módulo reconocedor de la voz y está dirigido al control de acceso, sin embargo, puede ser usado para el control de cualquier otro objeto.
	Capacidad	El sistema debe poder reconocer la voz con una tasa de acierto del 95 %. El sistema debe poder controlar los accesos para nuevos de usuarios. El sistema debe poder dar información de todos los accesos que tuvo un usuario. El sistema debe poder permitir o denegar el acceso a un usuario. El sistema debe ser capaz de trabajar en ambientes con ruido no mayores a 80 dBC.
Técnicos	Desempeño	El sistema debe ser capaz de reconocer una señal de voz en un tiempo menor o igual a 5 segundos. Sin embargo, la rapidez de respuesta del sistema dependerá de la cantidad de usuarios registrados en el sistema
	Amistad con Usuario	La interfaz de usuario del sistema no será muy compleja, y solo mostrará lo necesario para su manipulación.
	Modificabilidad	La modificación dependerá únicamente de los desarrolladores del software
	Portabilidad	El sistema podrá ser usado por cualquier dispositivo móvil Android superior a 4.4 y menor a 9.0, por lo que será portable para la mayoría de los dispositivos existentes en el mercado aproximadamente el 95 %.

Fuente: Elaboración propia

3.1.3. Etapa 3: Definir el patrón arquitectónico y modelo de referencia

El patrón arquitectónico a usar será el patrón arquitectónico en capas y el modelo de referencia será el modelo de referencia de sistemas empotrados, definido en (Noergaard, 2012). El modelo de referencia para un sistema empotrado está construido en capas y no es más que una distribución topológica de los elementos del patrón arquitectónico en capas, se define que todo sistema empotrado está formado por tres capas: de hardware, de software de sistema y de aplicación.

a) Capa de hardware

Esta capa estará formada principalmente por un microcontrolador Arduino, el cual alojará el controlador para la activación de la cerradura eléctrica por medio del módulo relé, además este se acoplará con el módulo Ethernet Shield, lo que permitirá conectarse a un router para poder establecer así la comunicación con el dispositivo móvil.

b) Capa de software de sistema

Esta capa estará formada por el controlador para la activación de la cerradura eléctrica y por el servidor para poder establecer la comunicación con la aplicación de reconocimiento de voz alojado en el dispositivo móvil.

c) Capa de software de aplicación

El sistema estará formado por hardware de propósito específico, diseñado solamente para

la demostración de la aplicación de reconocimiento de voz, el cual estará compuesto por 3 módulos: preprocesamiento, entrenamiento y reconocimiento de la señal de voz.

d) Modelo de referencia

El modelo de referencia servirá para aproximarse al posible hardware y software que deberá contener el sistema incorporando elementos relativos. La Figura 3.3 muestra el modelo de referencia del sistema de seguridad para el control de acceso por reconocimiento de voz.

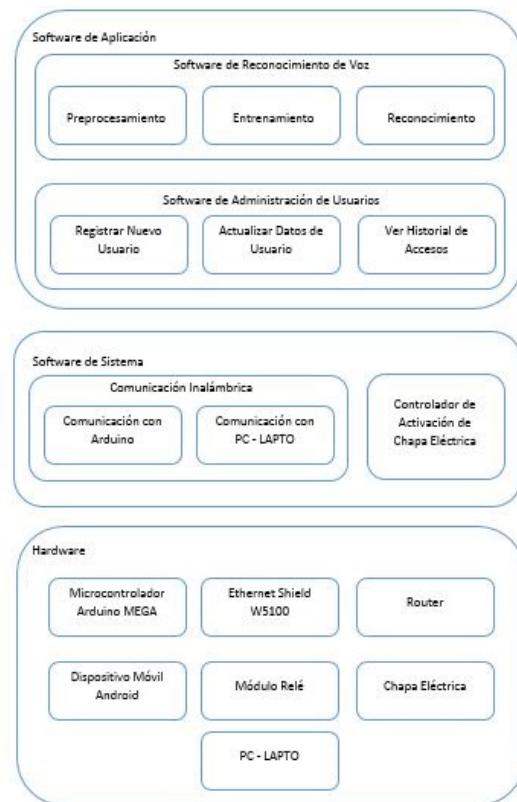


Figura 3.3: Modelo de referencia del sistema.
Fuente: Elaboración propia

3.2. Fase 2. Implementación de la arquitectura

3.2.1. Desarrollo del software de aplicación

3.2.1.1. Software de reconocimiento de voz

Primero se implementará una aplicación de escritorio en lenguaje Java, este nos permitirá evaluar y testear el algoritmo de reconocimiento de voz encontrando así las mejores funciones y los parámetros adecuados para un correcto y eficiente funcionamiento del reconocimiento de voz, para luego, finalmente ser llevado a una aplicación Android, el cual servirá como interfaz entre el usuario y el sistema.

a) Paso 1: Digitalización de la señal de voz

La Figura 3.4 muestra la función de grabar un audio por medio del micrófono de un computador, para ello se hace la petición a este, del uso del controlador para la captura del sonido con un formato específico, este formato tendrá los datos de frecuencia de muestreo f_s , cuantificación, número de canales y el tipo de ordenamiento en el que se almacenaran los bytes.

```

public TargetDataLine grabarAudio() {
    TargetDataLine targetDataLine = null;
    try {
        //Obtiene un TargetDataLine del Sistema (Controlador de captura de sonido)
        DataLine.Info dataLineInfo = new DataLine.Info(TargetDataLine.class, formato);
        targetDataLine = (TargetDataLine)AudioSystem.getLine(dataLineInfo);
        //Prepara la linea para usarla
        targetDataLine.open(formato);
        targetDataLine.start();
    } catch (LineUnavailableException ex) {
        String msj = "Controlador de microfono no disponible!";
        JOptionPane.showMessageDialog(null, msj, "Error", JOptionPane.ERROR_MESSAGE);
    }
    return targetDataLine;
}

@Override
public void run() {
    baos = new ByteArrayOutputStream();
    buffer = new byte[10000];
    int nBytesLeidos;
    //Bucle hasta que StopGrabar es "true" entonces el hilo para
    while (!stopGrabar) {
        //Lee datos en el buffer interno desde el controlador de audio
        //y luego los almacena en el buffer externo "buffer"
        nBytesLeidos = targetDataLine.read(buffer, 0, buffer.length);
        //Guarda los datos del buufer en el objeto de flujo de salida
        if (nBytesLeidos > 0) {
            baos.write(buffer, 0, nBytesLeidos);
        }
    }
    targetDataLine.close();
}

```

Figura 3.4: Código fuente de la función para grabar un audio.

Fuente: Elaboración propia

En la Figura 3.5 podemos ver la función para guardar un archivo .WAV a partir de los bytes capturados por la función anterior.

```

public void guardarAudio(String nombreArchivo) {
    try {
        File archivoSalida = new File(nombreArchivo);
        ByteArrayInputStream bais = new ByteArrayInputStream(datos.getBits());
        long nFrameArchivo = datos.getBits().length/formato.getFrameSize();
        AudioInputStream flujoEntradaAudio = new AudioInputStream(bais, formato, nFrameArchivo);
        AudioSystem.write(flujoEntradaAudio, AudioFileFormat.Type.WAVE, archivoSalida);
        archivo = archivoSalida;
    } catch (IOException ex) {
        String msj = "No se puede escribir los datos!";
        JOptionPane.showMessageDialog(null, msj, "Error", JOptionPane.ERROR_MESSAGE);
    }
}

```

Figura 3.5: Código fuente de la función para guardar un audio.

Fuente: Elaboración propia

En la Figura 3.6 se muestra la función para la lectura de un archivo .WAV convirtiendo los bytes a double, estos representan las amplitudes de la señal en el tiempo.

```

public String abrirAudio(File archivoEntrada) {
    String info = "";
    int LONGITUD_BUFFER = 1024;
    try {
        AudioInputStream flujoEntradaAudio = AudioSystem.getAudioInputStream(archivoEntrada);
        archivo = archivoEntrada;
        formato = flujoEntradaAudio.getFormat();
        int bytesPorFrame = formato.getFrameSize();
        byte[] buffer = new byte[LONGITUD_BUFFER*bytesPorFrame];
        int nBytesArchivo = bytesPorFrame*(int)flujoEntradaAudio.getFrameLength();
        datos = new Datos(nBytesArchivo, formato.isBigEndian());
        for (int pos = 0, nBytesLeidos;
             (nBytesLeidos = flujoEntradaAudio.read(buffer)) != -1;
             pos += nBytesLeidos) {
            System.arraycopy(buffer, 0, datos.getBits(), pos, nBytesLeidos);
        }
        datos.convertirByteADouble();
        info += archivoEntrada.getName()+"\n\n";
        info += " NroBytes Frames Formato" + "\n";
        info += " "+nBytesArchivo+
                " "+flujoEntradaAudio.getFrameLength()+
                " "+formato.toString()+"\n";
    } catch (IOException ex) {
        String msj = "No se puede leer los datos!";
        JOptionPane.showMessageDialog(null, msj, "Error", JOptionPane.ERROR_MESSAGE);
    } catch (UnsupportedAudioFileException ex) {
        String msj = "Audio no soportado!";
        JOptionPane.showMessageDialog(null, msj, "Error", JOptionPane.ERROR_MESSAGE);
    }
    return info;
}

```

Figura 3.6: Código fuente de la función para abrir un audio.

Fuente: Elaboración propia

b) Paso 2: Eliminación de ruido

La Figura 3.7 muestra la función del algoritmo LMS definido por las Ecuaciones (2.9), (2.10) y (2.11), teniendo como entradas a μ que es el paso de adaptación, $nord$ como el número de orden o tamaño del filtro y a_0 como vector de pesos inicial.

```

public void LMS(double mu, int nord, double[] a0) {
    verificarLongitudArreglos();
    int N = senalEntrada.length;
    double[] W = a0; //coeficientes del LMS
    if (W == null) {
        W = zeros(nord);
    }
    senalDeseada = new double[N]; //estimacion de d(n)
    double[] ruido = Utilitarios.concatenar(zeros(nord-1), senalRuido);
    double[] X;
    for (int i = 0; i < N; i++) {
        X = Utilitarios.recortar(ruido, i+nord-1, i);
        // Y = Wt*X (salida del filtro o estimacion de v1(n))
        // est_d(n) = x(n) - est_v1(n)
        senalDeseada[i] = senalEntrada[i] - FIR(W, X);
        // Wi(n+1) = Wi(n) + u*e(n)*X
        for (int j = 0; j < nord; j++) {
            W[j] = W[j] + mu*senalDeseada[i]*X[j];
        }
    }
}

```

Figura 3.7: Código fuente del algoritmo LMS.

Fuente: Elaboración propia

En la Figura 3.8 podemos ver la función del algoritmo NLMS definido por las Ecuaciones (2.10), (2.11), (2.12) y (2.13), donde β/DEN es el factor de convergencia, $nord$ el tamaño del filtro y a_0 el vector de pesos inicial.

```

public void NLMS(double beta, int nord, double[] a0) {
    verificarLongitudArreglos();
    int N = senalEntrada.length;
    double[] W = a0; //coeficientes del LMS
    if (W == null) {
        W = zeros(nord);
    }
    senalDeseada = new double[N]; //estimacion de d(n)
    double[] ruido = Utilitarios.concatenar(zeros(nord-1), senalRuido);
    double[] X;
    double DEN;
    for (int i = 0; i < N; i++) {
        X = Utilitarios.recortar(ruido, i+nord-1, i);
        // Y = Wt*X (salida del filtro o estimacion de v1(n))
        // est_d(n) = x(n) - est_v1(n)
        senalDeseada[i] = senalEntrada[i] - FIR(W, X);
        // DEN = ||x||!2 + alfa
        DEN = FIR(X, X) + 0.0001;
        // Wi(n+1) = Wi(n) + b*(X/||x||!2 + alfa)*e(n)
        for (int j = 0; j < nord; j++) {
            W[j] = W[j] + (beta/DEN)*X[j]*senalDeseada[i];
        }
    }
}

```

Figura 3.8: Código fuente del algoritmo NLMS.

Fuente: Elaboración propia

Al eliminar el ruido en la señal de voz a veces esta puede quedar con valores que sobrepasan el valor de cuantificación dado inicialmente, por lo que será necesario aplicar una normalización en la señal, esta función se muestra en la Figura 3.9 definida por las Ecuaciones (2.14), (2.15) y (2.16).

```

public static double[] normalizarDouble(double[] vEntrada, double c, double d, double m, double n) {
    double[] vNormalizado = new double[vEntrada.length];
    double A = (m-n)/(c-d);
    double B = m - A*c;

    for (int i = 0; i < vNormalizado.length; i++) {
        vNormalizado[i] = A*vEntrada[i] + B;
    }

    return vNormalizado;
}

```

Figura 3.9: Código fuente de la función de normalización de datos.

Fuente: Elaboración propia

c) Paso 3: Filtro preénfasis

En la Figura 3.10 se muestra la función para el filtro de preénfasis definida en la Ecuación (2.32), donde $vEntrada$ es el vector de datos de la señal a filtrar y α el parámetro de preénfasis.

```
public static double[] preEnfasis(double[] vEntrada, double alfa) {  
    int n = vEntrada.length;  
    double[] vSalida = new double[n];  
    vSalida[0] = vEntrada[0];  
    for (int i = 1; i < n; i++) {  
        vSalida[i] = vEntrada[i] - alfa*vEntrada[i-1];  
    }  
    return vSalida;  
}
```

Figura 3.10: Código fuente de la función del filtro de preénfasis.

Fuente: Elaboración propia

d) Paso 4: Detección de inicio y fin de la señal de voz

En la Figura 3.11 se muestra el algoritmo para la detección del inicio y fin de la voz en una señal por medio de la función de energía, definido por las Ecuaciones (2.17) y (2.18), teniendo como entradas a $tamTrama$ que es la longitud que tendrá cada trama para el análisis, a $umbral$ que indica que porcentaje tendrá el umbral de la energía promedio de la señal y a $vEntrada$ que es el vector de datos de la señal filtrada del paso anterior.

```

public static int[] eliminarSegmentosInutiles_Energia(int tamTrama, double umbral, double[] vEntrada) {
    double[] vEnergias = Preprocesamiento.getEnergias(tamTrama, vEntrada);
    int n = vEnergias.length, pi = 0, pf = vEntrada.length-1;
    double umb = umbral*Preprocesamiento.energia(vEntrada)/n;
    double e1, e2, e3;

    for (int i = 0; i < n; i++) {
        if (i+2 >= n || i+1 >= n) {
            break;
        }
        e1 = vEnergias[i]; e2 = vEnergias[i+1]; e3 = vEnergias[i+2];
        if (e1 > umb && e2 > umb && e3 > umb) {
            pi = tamTrama*i;
            break;
        }
    }
    for (int i = n-1; i >= 0; i--) {
        if (i-2 < 0 || i-1 < 0) {
            break;
        }
        e1 = vEnergias[i]; e2 = vEnergias[i-1]; e3 = vEnergias[i-2];
        if (e1 > umb && e2 > umb && e3 > umb) {
            if (i < n-1)
                pf = tamTrama*(i+1);
            break;
        }
    }
    int[] puntos = new int[2];
    puntos[0] = pi; puntos[1] = pf;

    return puntos;
}

```

Figura 3.11: Código fuente del algoritmo por función de energía.
Fuente: Elaboración propia

En la Figura 3.12 se muestra la función de energía promedio definida en la Ecuación (2.17).

```

public static double energia(double[] vEntrada) {
    double e = 0.0;
    for (int i = 0; i < vEntrada.length; i++) {
        e += Math.pow(vEntrada[i], 2);
    }
    return e;
}

```

Figura 3.12: Código fuente de la función de energía promedio.
Fuente: Elaboración propia

La Figura 3.13 muestra la función de energía de tiempo corto definida en la Ecuación (2.22).

```
public static double magnitud(double[] vEntrada) {
    double m = 0.0;
    for (int i = 0; i < vEntrada.length; i++) {
        m += Math.abs(vEntrada[i]);
    }
    return m;
}
```

Figura 3.13: Código fuente de la función de energía de tiempo corto.

Fuente: Elaboración propia

En la Figura 3.14 se muestra la función de la tasa de cruce por ceros definida por las Ecucciones (2.19), (2.20), (2.21) y (2.23).

```
public static double crucePorCero(double[] vEntrada) {
    double s = 0.0;
    int n = vEntrada.length;

    for (int i = 0; i <= n-2; i++) {
        s += Math.abs(Preprocesamiento.sign(vEntrada[i+1]) -
                      Preprocesamiento.sign(vEntrada[i]));
    }

    return (s/2)/n;
}

public static double sign(double x) {
    if (x >= 0) {
        return 1.0;
    } else {
        return -1.0;
    }
}
```

Figura 3.14: Código fuente de la función de tasa de cruce por ceros.

Fuente: Elaboración propia

En las Figuras 3.15 y 3.16 se muestran las funciones de la media y de desviación estándar

para un conjunto de datos respectivamente, definidas por las Ecuaciones (2.25) y (2.26).

```
public static double media(double[] vEntrada) {
    double m = 0.0;
    int n = vEntrada.length;
    for (int i = 0; i < n; i++) {
        m += vEntrada[i];
    }
    return m/n;
}
```

Figura 3.15: Código fuente de la función de la media.

Fuente: Elaboración propia

```
public static double desviacionEstandar(double[] vEntrada, double media) {
    double s = 0.0;
    int n = vEntrada.length;
    for (int i = 0; i < n; i++) {
        s += Math.pow(vEntrada[i]-media, 2);
    }
    return Math.sqrt(s/(n-1));
}
```

Figura 3.16: Código fuente de la función de desviación estándar.

Fuente: Elaboración propia

En la Figura 3.17 se muestra el algoritmo de Rabiner y Sambur de Tipo 1, definido por las Ecuaciones (2.22), (2.23), (2.24), (2.25) y (2.26), por otro lado en la Figura 3.18 se muestra el algoritmo de Rabiner y Sambur de Tipo 2, definido por las Ecuaciones (2.27), (2.28), (2.29), (2.30) y (2.31). Ambos tienen como entradas a *vEntrada* que es el vector de datos de la señal y *tamTrama* que es la longitud de la trama con la que se hará la evaluación, retornando el punto de inicio o fin de la voz en la señal según sea el caso.

```

public static int rabinerSambur1(int tamTrama, double[] vEntrada) {
    //PASO 1.1: Obtener la Magnitud Promedio
    double[] M = Preprocesamiento.getMagnitudes(tamTrama, vEntrada);
    //PASO 1.2: Obtener el Cruce por Ceros de la Señal
    double[] Z = Preprocesamiento.getCrucPorCeros(tamTrama, vEntrada);

    //PASO 2: Obtener Estadísticas del Ruido Ambiental
    //Para ello se escogen las 10 primeras tramas
    double[] Ms = new double[10];
    System.arraycopy(M, 0, Ms, 0, 10);
    double[] Zs = new double[10];
    System.arraycopy(Z, 0, Zs, 0, 10);

    //PASO 3: Obtener Umbrales
    double[] Mx = new double[M.length-10];
    System.arraycopy(M, 10, Mx, 0, M.length-10);
    double UmbSupEnrg = 0.5*Utilitarios.getMayor(Mx);
    double uMs = Preprocesamiento.media(Ms);
    double UmbInfEnrg = uMs + 2*Preprocesamiento.desviacionEstandar(Ms, uMs);
    double uZs = Preprocesamiento.media(Zs);
    double UmbCruCero = uZs + 2*Preprocesamiento.desviacionEstandar(Zs, uZs);

    //PASO 4: Hallar Inicio de la Palabra
    int inicio = 0;
    int in = -1;
    for (int n = 11; n < M.length; n++) {
        if (M[n] > UmbSupEnrg) {
            in = n;
            n = M.length;
        }
    }
    if (in != -1) {
        int ie = -1;
        for (int n = in; n > 10 ; n--) {
            if (M[n] <= UmbInfEnrg) {
                ie = n;
                n = 10;
            }
        }
        if (ie != -1) {
            int iz = -1; int cont = 0; int k = 11;
            if (ie-25 >= 1) {
                k = ie-25;
            }
            for (int n = ie; iz == -1 && n > k; n--) {
                if (Z[n] <= UmbCruCero) {
                    inicio = ie;
                    cont = 0;
                } else if (Z[n] > UmbCruCero) {
                    cont++;
                    if (cont < 3 && Z[n-1] < UmbCruCero) {
                        inicio = ie;
                    } else if (cont >= 3) {
                        for (iz = n; iz > k && Z[iz] > UmbCruCero; iz--) {}
                        inicio = iz;
                    }
                }
            }
        }
    }
    return inicio;
}

```

Figura 3.17: Código fuente del algoritmo Rabiner y Sambur Tipo 1.

Fuente: Elaboración propia

```

public static int rabinerSambur2(int tamTrama, double[] vEntrada) {
    //PASO 1.1: Obtener la Magnitud Promedio
    double[] M = Preprocesamiento.getMagnitudes(tamTrama, vEntrada);
    //PASO 1.2: Obtener el Cruce por Ceros de la Señal
    double[] Z = Preprocesamiento.getCrucPorCeros(tamTrama, vEntrada);

    //PASO 2: Obtener Estadísticas de Ruido Ambiental
    double[] Ms = new double[10];
    System.arraycopy(M, 0, Ms, 0, 10);
    double[] Zs = new double[10];
    System.arraycopy(Z, 0, Zs, 0, 10);

    //PASO 3: Obtener Umbrales
    double uZs = Preprocesamiento.media(Zs);
    double UmbCruCero = uZs + 2*Preprocesamiento.desviacionEstandar(Zs, uZs);
    double IF = 25/tamTrama;
    double IZCT = Math.min(IF, UmbCruCero);
    double[] Mx = new double[M.length-10];
    System.arraycopy(M, 10, Mx, 0, M.length-10);
    double IMX = Utilitarios.getMayor(Mx);
    double IMN = Preprocesamiento.media(Ms);
    double I1 = 0.03*(IMX-IMN) + IMN;
    double I2 = 4*IMN;
    double ITL = Math.min(I1, I2);
    double ITU = 5*ITL;

    //PASO 4: Hallar Inicio de la Palabra
    int inicio = 0;
    int N = -1;
    for (int m = 11; m < M.length; m++) {
        if (M[m] >= ITL) {
            for (int i = m; i < M.length; i++) {
                if (M[i] < ITL) {
                    m = i + 1;
                    i = M.length;
                } else {
                    if (M[i] >= ITU) {
                        N = i;
                        if (i == m) {
                            N--;
                        }
                        i = M.length;
                        m = M.length;
                    }
                }
            }
        }
    }
    if (N != -1) {
        int iz = -1; int cont = 0; int k = 11;
        if (N-25 >= 1) {
            k = N-25;
        }
        for (int n = N; iz == -1 && n > k; n--) {
            if (Z[n] < IZCT) {
                inicio = N;
                cont = 0;
            } else if (Z[n] >= IZCT) {
                cont++;
                if (cont < 3 && Z[n-1] < IZCT) {
                    inicio = N;
                } else if (cont >= 3) {
                    for (iz = n; iz > k && Z[iz] >= IZCT; iz--) {}
                    inicio = iz;
                }
            }
        }
    }
    return inicio;
}

```

Figura 3.18: Código fuente del algoritmo Rabiner y Sambur Tipo 2.

Fuente: Elaboración propia

e) Paso 5: Segmentación

En la Figura 3.19 se muestra la función para la segmentación de una señal definida por las Ecuaciones (2.33) y (2.34), además se aplica un ponderado por una ventana.

```
private double [][] obtenerMatrizVentanamiento(double[] datos) {
    int a = 0, b = TV, j = 0, k = 0;
    int tam = datos.length;
    int piezas = (int) Math.floor((tam-TV)/DS)+1;
    boolean todo = (tam == DS*(piezas-1) + TV);
    if (!todo) {//Verificamos si se coge toda la señal
        piezas++;
    }
    double[][] mat = new double[piezas][];
    double[] vh;

    switch (tipo) {
        case 1:
            vh = crearHamming(TV);
            break;
        default:
            vh = crearRectangular(TV);
            break;
    }
    while (b <= tam) {
        mat[j] = new double[TV];
        for (int i = a; i < b; i++, k++) {
            mat[j][k] = datos[i]*vh[k];
        }
        if(b == tam) {
            return mat;
        }
        a += DS; b += DS; j++; k = 0;
    }
    if (b > tam) {
        mat[j] = new double[TV];
        for (int i = a; i < b; i++, k++) {
            if (i < tam) {
                mat[j][k] = datos[i]*vh[k];
            } else {//extrapolacion
                mat[j][k] = 2*mat[j][k-1] - mat[j][k-2];
            }
        }
    }
    return mat;
}
```

Figura 3.19: Código fuente de la función para la segmentación de una señal.

Fuente: Elaboración propia

f) Paso 6: Ponderado por la ventana de Hamming

En las Figuras 3.20 y 3.21 se muestran las funciones de ponderado por ventana rectangular y Hamming respectivamente, definidas por las Ecuaciones (2.35), (2.36) y (2.37), donde N es el tamaño que tendrá la ventana a generar.

```
private double[] crearRectangular(int N) {  
    double[] v = new double[N];  
    for (int i = 0; i < N; i++) {  
        v[i] = 1;  
    }  
    return v;  
}
```

Figura 3.20: Código fuente de la función de ponderado por ventana rectangular.

Fuente: Elaboración propia

```
private double[] crearHamming(int N) {  
    double[] v = new double[N];  
    for (int i = 0; i < N; i++) {  
        v[i] = 0.54 - 0.46*Math.cos(2*Math.PI*i/(N-1));  
    }  
    return v;  
}
```

Figura 3.21: Código fuente de la función de ponderado por ventana de Hamming.

Fuente: Elaboración propia

g) Paso 7: Rellenado con ceros

En la Figura 3.22 se muestra la función de relleno con ceros, definida por las Ecuaciones (2.41) y (2.42), donde se tiene como entrada a $vEntrada$ que es el vector de datos de la señal (amplitudes), y da como resultado un vector de complejos $vSalida$.

```

public static Complejo[] rellenadoCeros(double[] vEntrada) {
    int N = vEntrada.length;
    int Nfft = (int) Math.pow(2, (int) Math.ceil(Math.log(N) / Math.log(2)));
    Complejo[] vSalida = new Complejo[Nfft];

    for (int n = 0; n < Nfft; n++) {
        if (n < N) {
            vSalida[n] = new Complejo(vEntrada[n], 0);
        } else {
            vSalida[n] = new Complejo(0, 0);
        }
    }

    return vSalida;
}

```

Figura 3.22: Código fuente de la función de relleno con ceros.

Fuente: Elaboración propia

h) Paso 8: Transformada rápida de Fourier

En la Figura 3.23 se muestra la función para la transformada rápida de Fourier, definida por las Ecuación (2.43) y (2.44), como podemos ver esta función es un método recursivo que tiene como entradas a *vEntrada* y *N*, que son el vector de datos de la señal (rellenado con ceros) y el tamaño de este vector respectivamente.

```

public static Complejo[] transformada(Complejo[] vEntrada, int N) {
    Complejo[] vSalida = new Complejo[N];

    if (N == 1) {
        vSalida[0] = vEntrada[0];
    } else {
        Complejo[] fe = new Complejo[N/2];
        Complejo[] fo = new Complejo[N/2];

        for (int i = 0; i < N/2; i++) {
            fe[i] = vEntrada[2*i];
            fo[i] = vEntrada[2*i+1];
        }

        Complejo[] Fe = transformada(fe, N/2);
        Complejo[] Fo = transformada(fo, N/2);
        Complejo w, aux;
        double teta;

        for (int k = 0; k < N/2; k++) {
            //Twiddle Factor (Factor Mariposa)
            teta = 2*Math.PI*k/N;
            w = new Complejo(Math.cos(teta), (-1)*Math.sin(teta));
            aux = Fo[k].multiplicar(w);
            vSalida[k] = Fe[k].sumar(aux);
            vSalida[k+N/2] = Fe[k].restar(aux);
        }
    }

    return vSalida;
}

```

Figura 3.23: Código fuente de la función de la transformada rápida de Fourier.

Fuente: Elaboración propia

i) Paso 9: Obtención del espectro en potencia

En la Figura 3.24 se muestra la función para obtener el espectro en potencia de la señal, definida por las Ecuaciones (2.45) y (2.46), donde se tiene como entrada a *ve* que es el vector de datos de la señal después de aplicarle la transformada rápida de Fourier.

```

public static double[] getEspectroPotencia(Complejo[] ve) {
    int Np = ve.length/2;
    double[] vs = new double[Np+1];
    for (int i = 0; i < vs.length; i++) {
        vs[i] = ve[i].getModulo2();
    }
    return vs;
}
public double getModulo2() {
    return Math.pow(real,2) + Math.pow(imag,2);
}

```

Figura 3.24: Código fuente de la función para la obtención del espectro en potencia.

Fuente: Elaboración propia

j) Paso 10: Filtrado e integración o remuestreo por bandas críticas

En la Figura 3.25 se muestra la función de remuestreo por bandas críticas, definido por las Ecuaciones desde (2.49) hasta (2.57), esta función tiene como entradas a f_{min} y a f_{max} que son la frecuencia mínima y máxima de la señal respectivamente.

```

public int[][] getBandasCriticasTriangulares(double fMin, double fMax) {
    double mell, mel2, mel3, hz1, hz2, hz3, f;
    double[] hm;
    H = new ArrayList<>();
    int[][] pos = new int[BC][2];

    double fMinMel = toMel2(fMin);
    double fMaxMel = toMel2(fMax);
    double pasoMel = (fMaxMel-fMinMel)/(BC+1.0);
    int nFFT = (int) Math.pow(2, (int) Math.ceil(Math.log(dimensionVentana)/Math.log(2)));
    double pasoHz = fs/nFFT;

    for (int bc = 0; bc < BC; bc++) {
        mell = bc*pasoMel + fMinMel;
        mel2 = (bc+1)*pasoMel + fMinMel;
        mel3 = (bc+2)*pasoMel + fMinMel;

        hz1 = toMelInv2(mell);
        hz2 = toMelInv2(mel2);
        hz3 = toMelInv2(mel3);

        pos[bc][0] = (int) Math.round(hz1/pasoHz);
        pos[bc][1] = (int) Math.round(hz3/pasoHz);

        //Crear Hm[k]
        hm = new double[pos[bc][1] - pos[bc][0] + 1];
        for (int p = pos[bc][0], i = 0; p <= pos[bc][1]; p++, i++) {
            f = p*pasoHz;
            if (f <= hz1) {
                hm[i] = 0;
            } else if (f > hz1 && f < hz2) {
                hm[i] = (f-hz1)/(hz2-hz1);
            } else if (f == hz2) {
                hm[i] = 1;
            } else if (f > hz2 && f < hz3) {
                hm[i] = (hz3-f)/(hz3-hz2);
            } else if (f >= hz3) {
                hm[i] = 0;
            }
        }
        H.add(hm);
    }
    return pos;
}

```

Figura 3.25: Código fuente de la función de remuestreo por bandas críticas.

Fuente: Propia

k) Paso 11: Obtención de los coeficientes MFCC

La Figura 3.26 muestra la función para la obtención de los coeficientes MFCC, definido por las Ecuaciones (2.63), (2.64) y (2.65). Estas ecuaciones definen las etapas de compresión y la transformada discreta inversa de Fourier de la señal.

```

public double[][] getResultMFCC(double fMin, double fMax) {
    double[][] mo = new double[M][senalVentaneada.length];//Matriz de Observaciones
    double[] Psc = new double[BC], P;
    double suma;

    //Paso 1. Calcular bins-MEL
    int[][] pos = getBandasCriticasTriangulares(fMin, fMax);
    for (int k2 = 0; k2 < mo[0].length; k2++) {
        //Paso 2. Transformada Discreta de Fourier por Ventanas
        //Paso 3. Obtencion del Espectro de Potencia
        P = Utilitarios.getEspectroPotencia(Fourier.fft2(senalVentaneada[k2]));
        //P = Utilitarios.getModulo(Fourier.fft2(senalVentaneada[k2]));
        for (int bc = 0; bc < BC; bc++) {
            //Paso 5. Filtrado e Integracion o Remuestreo por Bandas Criticas
            Psc[bc] = Utilitarios.getSuma(Utilitarios.multiplicar(H.get(bc),
                Utilitarios.recortar(P, pos[bc][0], pos[bc][1])));
            if (Psc[bc] != 0) { //Paso 6. Compresion
                Psc[bc] = Math.log10(Psc[bc]);
            }
        }
        for (int j = 0; j < M; j++) {
            suma = 0; //Paso 7. Transformada Discreta Inversa de Fourier
            for (int k1 = 0; k1 < BC; k1++) {
                suma += Psc[k1] * Math.cos(Math.PI*(j+1.0)*(k1+0.5)/(double)BC);
            }
            //Paso 8. Coeficientes MFCC
            mo[j][k2] = Math.sqrt(2.0/(double)BC) * suma;
        }
    }
    return mo;
}

```

Figura 3.26: Código fuente de la función para la obtención de coeficientes MFCC.

Fuente: Elaboración propia

La Figura 3.27 muestra la función para la obtención de los coeficientes MFCC delta, definido por la Ecuación (2.67).

```

public double[][] getResultMFCCDelta(double fMin, double fMax) {
    double[][] mo = new double[26][senalVentaneada.length];//Matriz de Observaciones
    double[] Psc = new double[BC], P;
    double suma;

    //Paso 1. Calcular bins-MEL
    int[][] pos = getBandasCriticasTriangulares(fMin, fMax);
    for (int k2 = 0; k2 < mo[0].length; k2++) {
        //Paso 2. Transformada Discreta de Fourier por Ventanas
        //Paso 3. Obtencion del Espectro de Potencia
        P = Utilitarios.getEspectroPotencia(Fourier.fft2(senalVentaneada[k2]));
        for (int bc = 0; bc < BC; bc++) {
            //Paso 5. Filtrado e Integracion o Remuestreo por Bandas Criticas
            Psc[bc] = Utilitarios.getSuma(Utilitarios.multiplicar(H.get(bc),
                Utilitarios.recortar(P, pos[bc][0], pos[bc][1])));
            if (Psc[bc] != 0) { //Paso 6. Compresion
                Psc[bc] = Math.log10(Psc[bc]);
            }
        }
        for (int j = 0; j < 13; j++) {
            suma = 0; //Paso 7. Transformada Inversa Discreta de Fourier
            for (int k1 = 0; k1 < BC; k1++) {
                suma += Psc[k1] * Math.cos(Math.PI*(j+1.0)*(k1+0.5)/(double)BC);
            }
            //Paso 8. Coeficientes MFCC
            mo[j][k2] = Math.sqrt(2.0/(double)BC) * suma;
        }
    }
    //Paso 9. Agregar las lra derivadas a mo
    for (int j = 0; j < mo[0].length; j++) {
        for (int i = 0; i < 13; i++) {
            if (i-2 < 0 || i+2 > 12) {
                mo[i+13][j] = mo[i][j];
            } else {
                mo[i+13][j] = mo[i+2][j] - mo[i-2][j];
            }
        }
    }
}

return mo;
}

```

Figura 3.27: Código fuente de la función para la obtención de coeficientes MFCC delta.

Fuente: Elaboración propia

La Figura 3.28 muestra la función para la obtención de los coeficientes MFCC doble delta, definido por la Ecuación (2.68).

```

public double[][] getResultMFCCDeltaDelta(double fMin, double fMax) {
    double[][] mo = new double[39][senalVentaneada.length];//Matriz de Observaciones
    double[] Psc = new double[BC], P;
    double suma;

    //Paso 1. Calcular bins-MEL
    int[][] pos = getBandasCriticasTriangulares(fMin, fMax);
    for (int k2 = 0; k2 < mo[0].length; k2++) {
        //Paso 2. Transformada Discreta de Fourier por Ventanas
        //Paso 3. Obtencion del Espectro de Potencia
        P = Utilitarios.getEspectroPotencia(Fourier.fft2(senalVentaneada[k2]));
        for (int bc = 0; bc < BC; bc++) {
            //Paso 5. Filtrado e Integracion o Remuestreo por Bandas Criticas
            Psc[bc] = Utilitarios.getSuma(Utilitarios.multiplicar(H.get(bc),
                Utilitarios.recortar(P, pos[bc][0], pos[bc][1])));
            if (Psc[bc] != 0) { //Paso 6. Compresion
                Psc[bc] = Math.log10(Psc[bc]);
            }
        }
        for (int j = 0; j < 13; j++) {
            suma = 0; //Paso 7. Transformada Inversa Discreta de Fourier
            for (int kl = 0; kl < BC; kl++) {
                suma += Psc[kl] * Math.cos(Math.PI*(j+1.0)*(kl+0.5)/(double)BC);
            }
            //Paso 8. Coeficientes MFCC
            mo[j][k2] = Math.sqrt(2.0/(double)BC) * suma;
        }
    }
    //Paso 9. Agregar las lra derivadas a mo
    for (int j = 0; j < mo[0].length; j++) {
        for (int i = 0; i < 13; i++) {
            if (i-2 < 0 || i+2 > 12) {
                mo[i+13][j] = mo[i][j];
            } else {
                mo[i+13][j] = mo[i+2][j] - mo[i-2][j];
            }
        }
    }
    //Paso 10. Agregar las 2da derivadas a mo
    for (int j = 0; j < mo[0].length; j++) {
        for (int i = 13; i < 26; i++) {
            if (i-1 < 13 || i+1 > 25) {
                mo[i+13][j] = mo[i][j];
            } else {
                mo[i+13][j] = mo[i+1][j] - mo[i-1][j];
            }
        }
    }
}

return mo;
}

```

Figura 3.28: Código fuente de la función para la obtención de coeficientes MFCC doble delta.
 Fuente: Elaboración propia

l) Paso 12: Medidas de distorsión

En las Figuras 3.29 y 3.30 se muestra las funciones de distancia euclíadiana cuadrática y la distancia de error cuadrático medio entre dos señales, definidas por las Ecuaciones (2.99) y (2.100) respectivamente.

```
public static double distanciaEuclidianaCuadratica(double[] v1, double[] v2) {
    double suma = 0.0;
    for (int i = 0; i < v1.length; i++) {
        suma += Math.pow(v1[i]-v2[i], 2);
    }
    return suma;
}
```

Figura 3.29: Código fuente de la función de distancia euclíadiana cuadrática.

Fuente: Elaboración propia

```
public static double distanciaErrorCuadraticoMedio(double[] v1, double[] v2) {
    double suma = 0;
    int N = v1.length;
    for (int i = 0; i < N; i++) {
        suma += Math.pow(v1[i]-v2[i], 2);
    }
    return suma/N;
}
```

Figura 3.30: Código fuente de la función de distancia de error cuadrático medio.

Fuente: Elaboración propia

m) Paso 13: Alineamiento temporal dinámico

En la Figura 3.31 se muestra la función DTW de tipo simétrico con slope constrain $P = 1/2$ sin ventana de ajuste para la comparación de patrones definida por las Ecuaciones (2.95), (2.96), (2.97) y (2.98). El diagrama de flujo del algoritmo puede verse en la Figura 2.53.

```

public void dtwSimetricoP12() {
    //1. Llenar las matrices de distancias.
    obtenerMatricesDistancias();
    double[][] md = mDistancia, mg = mPesos;
    int nca = md.length, ncb = md[0].length;
    double[] rama = new double[5];
    int r = Math.abs(mfcc1[0].length-mfcc2[0].length) + radio;

    //2. Algoritmo DTW
    //2.1. Condición Inicial
    mg[0][0] = 2*md[0][0];
    //2.2. Proceso con ecuación DP con ventana de ajuste
    int i = 0, j = 0; boolean fin = false;
    do{
        i = i + 1;
        if (i > j+r) {
            j = j + 1;
            if (j >= ncb) {
                //2.3. Calcular el mejor camino
                calcularCaminoDTW(mg, nca-1, ncb-1);
                //2.4. Calcular distancias
                calcularDistanciaCamino(mg);
                distanciaDTW = mg[nca-1][ncb-1]/(nca+ncb);
                fin = true;
            } else {
                i = j - r;
            }
        } else {
            if (i >= 0) {
                if (i < nca) {
                    //calcular DWT
                    rama[0] = 1000; rama[1] = 1000; rama[2] = 1000; rama[3] = 1000; rama[4] = 1000;
                    if (i-1 >= 0 && j-3 >= 0) {
                        rama[0] = mg[i-1][j-3] + 2*md[i][j-2] + md[i][j-1] + md[i][j];
                    }
                    if (i-1 >= 0 && j-2 >= 0) {
                        rama[1] = mg[i-1][j-2] + 2*md[i][j-1] + md[i][j];
                    }
                    if (i-1 >= 0 && j-1 >= 0) {
                        rama[2] = mg[i-1][j-1] + 2*md[i][j];
                    }
                    if (i-2 >= 0 && j-1 >= 0) {
                        rama[3] = mg[i-2][j-1] + 2*md[i-1][j] + md[i][j];
                    }
                    if (i-3 >= 0 && j-1 >= 0) {
                        rama[4] = mg[i-3][j-1] + 2*md[i-2][j] + md[i-1][j] + md[i][j];
                    }
                    if (i != 0 || j != 0) {
                        mg[i][j] = Utilitarios.getMenor(rama);
                    }
                }
            }
        }
    } while(!fin);
}

```

Figura 3.31: Código fuente de la función de DTW simétrico con $P = 1/2$ con ventana de ajuste.

Fuente: Elaboración propia

En la Figura 3.32 se muestra la función DTW de tipo simétrico con slope constrain $P = 1/2$

sin ventana de ajuste en la comparación de patrones.

```

public void dtwSimetricoP12() {
    //1. Llenar las matrices de distancias.
    obtenerMatricesDistancias();
    double[][] md = mDistancia, mg = mPesos;
    int nca = md.length, ncb = md[0].length;
    double[] rama = new double[5];
    int r = Math.abs(mfccl[0].length-mfcc2[0].length) + radio;

    //2. Algoritmo DTW
    //2.1. Condición Inicial
    mg[0][0] = 2*md[0][0];
    //2.2. Proceso con ecuación DP con ventana de ajuste
    for (int i = 0; i < nca; i++) {
        for (int j = 0; j < ncb; j++) {
            rama[0] = 1000; rama[1] = 1000; rama[2] = 1000; rama[3] = 1000; rama[4] = 1000;
            if (j-r <= i && i <= j+r && !(i == 0 && j == 0)) {
                if (i-1 >= 0 && j-3 >= 0) {
                    rama[0] = mg[i-1][j-3] + 2*md[i][j-2] + md[i][j-1] + md[i][j];
                }
                if (i-1 >= 0 && j-2 >= 0) {
                    rama[1] = mg[i-1][j-2] + 2*md[i][j-1] + md[i][j];
                }
                if (i-1 >= 0 && j-1 >= 0) {
                    rama[2] = mg[i-1][j-1] + 2*md[i][j];
                }
                if (i-2 >= 0 && j-1 >= 0) {
                    rama[3] = mg[i-2][j-1] + 2*md[i-1][j] + md[i][j];
                }
                if (i-3 >= 0 && j-1 >= 0) {
                    rama[4] = mg[i-3][j-1] + 2*md[i-2][j] + md[i-1][j] + md[i][j];
                }
                mg[i][j] = Utilitarios.getMenor(rama);
            }
        }
    }

    //2.3. Calcular el mejor camino
    calcularCaminoDTW(mg, nca-1, ncb-1);
    //2.4. Calcular distancias
    calcularDistanciaCamino(mg);
    distanciaDTW = mg[nca-1][ncb-1]/(nca+ncb);
}

```

Figura 3.32: Código fuente de la función de DTW simétrico con $P = 1/2$ sin ventana de ajuste.

Fuente: Elaboración propia

Si bien el algoritmo de la Figura 3.31 su tiempo de ejecución puede ser menor al algoritmo de la Figura 3.32, sin embargo este último puede tener mayor precisión en los resultados, por lo que los someteremos a evaluación.

n) Paso 14: Construcción del patrón de referencia

En la Figura 3.33 tenemos la función *indicesMinAnterior* que nos permite hallar el punto anterior con menor peso en la matriz de comparación de patrones *mg*, esta es usada por la función *calcularCaminoDTW*, la cual permite obtener el camino con menor distancia $y_{w(n)}$, definida por la Ecuación (2.101), llamada función warping o función de alineamiento.

```

private void calcularCaminoDTW(double[][] mg, int i, int j) {
    caminoDTW = null;
    caminoDTW = new ArrayList<>();
    int[] indices = new int[2];
    indices[0] = i; indices[1] = j;
    while (indices[0] >= 0 && indices[1] >= 0) {
        caminoDTW.add(indices);
        indices = indicesMinAnterior(mg, indices[0], indices[1]);
    }
}

private int[] indicesMinAnterior(double[][] mg, int i, int j) {
    int[] indices = new int[2];
    double[] aux = new double[3];
    if (i-1 >= 0) aux[0] = mg[i-1][j];
    else aux[0] = 10000;
    if (i-1 >= 0 && j-1 >= 0) aux[1] = mg[i-1][j-1];
    else aux[1] = 10000;
    if (j-1 >= 0) aux[2] = mg[i][j-1];
    else aux[2] = 10000;
    double[] r = Utilitarios.getMenorConPosicion(aux);
    switch((int)r[1]) {
        case 0: indices[0] = i-1; indices[1] = j; break;
        case 1: indices[0] = i-1; indices[1] = j-1; break;
        case 2: indices[0] = i; indices[1] = j-1; break;
    }
    return indices;
}

```

Figura 3.33: Código fuente de la función para obtener el recorrido mínimo DTW.

Fuente: Elaboración propia

En la Figura 3.34 se muestra la función para la construcción de los patrones de referencia, definido por la Ecuación (2.101).

```

public double[][] fusionarAudios2() {
    int nfmo = mfcc1.length;
    int ncmo = caminoDTW.size();
    double[][] MF = new double[nfmo][ncmo];
    double[] x, y;
    boolean invertir = mfcc1[0].length >= mfcc2[0].length;
    ArrayList pos_repet = new ArrayList();

    for (int j = ncmo-1, pi, pj, pi_a = -1, pj_a = -1; j > -1; j--) {
        pi = caminoDTW.get(j)[0];
        pj = caminoDTW.get(j)[1];
        if (pi_a == pi || pj_a == pj) {
            pos_repet.add(ncmo-1-j);
        }
        if (!invertir) {
            x = Utilitarios.getColMatriz(mfcc1, pi);
            y = Utilitarios.getColMatriz(mfcc2, pj);
        } else {
            x = Utilitarios.getColMatriz(mfcc2, pi);
            y = Utilitarios.getColMatriz(mfcc1, pj);
        }
        for (int i = 0; i < nfmo; i++) {
            MF[i][ncmo-1-j] = (x[i]+y[i])/2;
        }
        pi_a = pi;
        pj_a = pj;
    }

    double[][] MF2 = new double[nfmo][MF[0].length - pos_repet.size()];
    boolean repetido; double suma;
    for (int j = 0, jm = 0, cont; j < MF[0].length; j++) {
        repetido = false;
        cont = 0;
        while (Utilitarios.existePosicion(j+cont, pos_repet)) {
            cont++;
            repetido = true;
        }
        if (!repetido) {
            for (int i = 0; i < MF.length; i++) {
                MF2[i][jm] = MF[i][j];
            }
            jm++;
        } else {
            for (int i = 0; i < MF.length; i++) {
                suma = 0;
                for (int k = 0; k <= cont; k++) {
                    suma += MF[i][j-1+k];
                }
                MF2[i][jm-1] = suma/(cont+1);
            }
            j += (cont-1);
        }
    }
    return MF2;
}

```

Figura 3.34: Código fuente de la función para la construcción del patrón de referencia.

Fuente: Elaboración propia

3.2.1.2. Software de administración de usuarios

A continuación, veremos las interfaces gráficas de usuario del software de aplicación del sistema de control de acceso por reconocimiento de voz. En la Figura 3.35 podemos ver la primera interfaz que se mostrará después de haber instalado la aplicación en el dispositivo móvil Android, donde tendremos dos opciones *Iniciar* y *Crear Cuenta*. La primera es cuando ya se tiene un usuario del tipo *Administrador*, pero sin tener aún el registro de su patrón de voz de clave de acceso, en caso de no ocurrir esto entonces se escogerá la segunda opción.

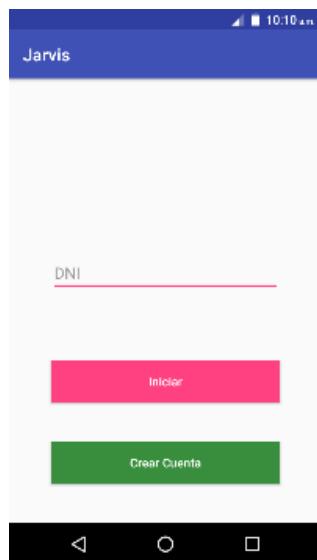


Figura 3.35: Inicio de sesión al sistema por primera vez.

Fuente: Elaboración propia

En la Figura 3.36 se muestra la interfaz para el registro del usuario *Administrador* del sistema, esto solo se hará una sola vez, después de registrarla esta opción desaparecerá.

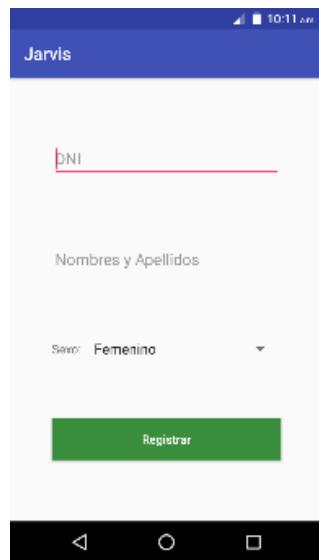


Figura 3.36: Registro del usuario administrador del sistema.

Fuente: Elaboración propia

En las Figuras 3.37 y 3.38 se muestran las interfaces de inicio de sesión al sistema por primera vez, solicitando los permisos de *grabar audio* y el *acceso a archivos multimedia* para la aplicación respectivamente, donde estos deben ser aceptados para el correcto funcionamiento del sistema.

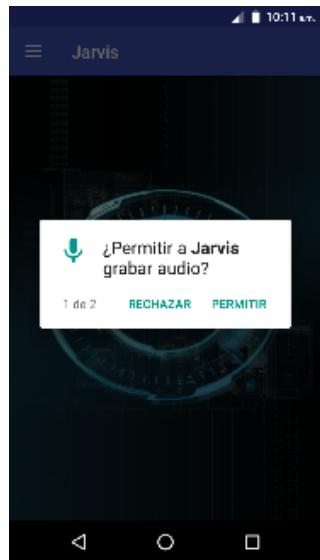


Figura 3.37: Permiso de la aplicación para grabar audios en el dispositivo.

Fuente: Elaboración propia



Figura 3.38: Permiso de la aplicación para acceder y almacenar archivos en el dispositivo.

Fuente: Elaboración propia

En la Figura 3.39 se muestra las opciones del menú principal del sistema que son: *Comandos por Voz*, *Cerrar Sesión*, *Micrófono Externo*, *Dispositivo Arduino*, *Administrador Cuentas* y *Consultar Accesos*, donde estos dos últimos solo se mostrarán para usuarios del tipo *Administrador*.

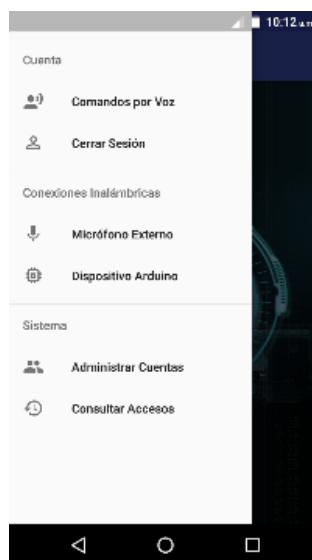


Figura 3.39: Menú principal del sistema.

Fuente: Elaboración propia

En la Figura 3.40 se puede ver la interfaz para la opción del menú principal *Administrador Cuentas*, donde se muestra una lista de las cuentas de los usuarios del sistema. La opción con el ícono de lupa es para buscar usuarios, esta búsqueda puede ser por su nombre o DNI (Documento Nacional de Identidad), la opción con el ícono del mas es para registrar un nuevo usuario al sistema, además al seleccionar un usuario de la lista podemos ver y editar sus datos, y la última opción *Sin Acceso* es para denegar el acceso al sistema al usuario.

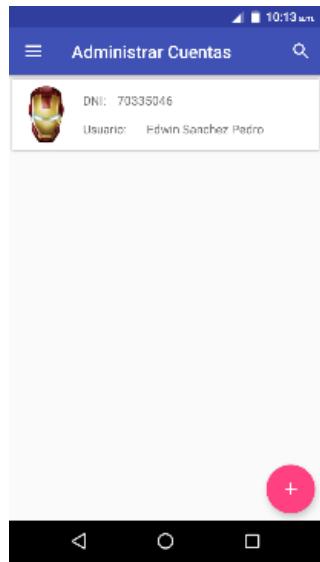


Figura 3.40: Lista de cuentas de usuario del sistema.

Fuente: Elaboración propia

En la Figura 3.41 se muestra la interfaz para registrar a un nuevo usuario al sistema, donde en *tipo* se especificará que tipo de acceso tendrá este usuario al sistema, si es *Administrador* tendrá todos los privilegios del sistema, si es *Normal* no podrá registrar ni editar los datos de los usuarios ni tampoco podrá consultar los accesos de estos al sistema.

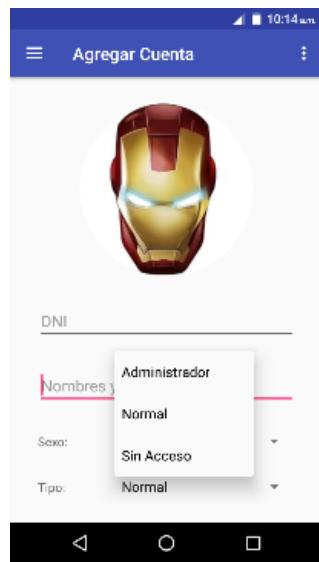


Figura 3.41: Registrar un nuevo usuario al sistema.

Fuente: Elaboración propia

En la Figura 3.42 se muestra el menú con las opciones *Guardar*, *Patrón de Voz* y *Comando de Voz*. El primero es para guardar el registro o el editado de los datos de un usuario. El segundo es para poder asignar un patrón de voz clave al usuario que le permitirá acceder al sistema. Y la última opción es para asignar los comandos de voz que realizará el sistema por parte del usuario.

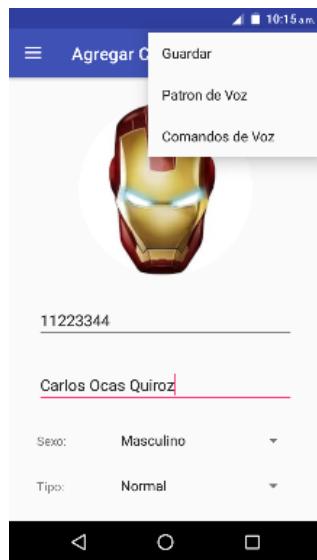


Figura 3.42: Menú de opciones para la administración de usuarios del sistema.

Fuente: Elaboración propia

En la Figura 3.43 se muestra la lista de audios de los patrones de voz que conforman la clave de acceso de un usuario, esta interfaz se muestra a través de la opción *Patrones de Voz* del menú de opciones que se vió en la Figura 3.42.

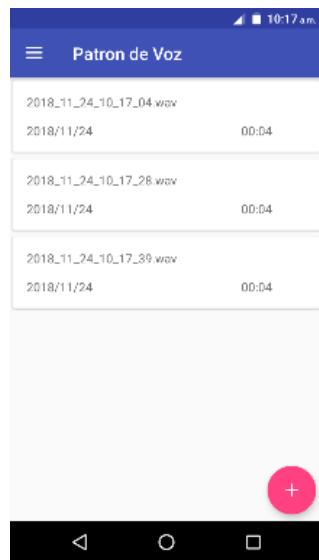


Figura 3.43: Lista de audios del patrón clave de usuario para el acceso al sistema.

Fuente: Elaboración propia

En la Figura ?? se muestra la interfaz para la grabación de un nuevo audio, a través de la opción con *ícono de mas* que se vió en la interfaz anterior en la Figura 3.43. Como podemos ver en esta interfaz al inicio se muestran tres opciones, la primera es para iniciar la grabación, la segunda es para reproducir la grabación y la tercera es para regresar a la lista de audios.

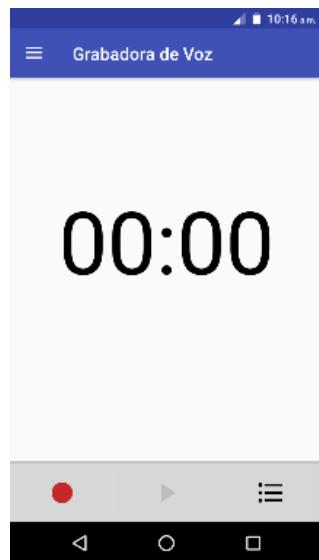


Figura 3.44: Grabación de un nuevo audio en el sistema.

Fuente: Elaboración propia

Una vez terminada la grabación se mostrarán las opciones de *cancelar* y *aceptar* el almacenamiento del audio en la base datos, *grabar* un nuevo audio, *reproducir* el audio grabado, y por ultimo ir a la *lista de audios*, ver la Figura 3.45.



Figura 3.45: Opciones después de grabar un audio en el sistema.

Fuente: Elaboración propia

También, podemos eliminar audios del sistema, para ello mantenemos presionado por un tiempo el audio a eliminar y así la vista cambiará a modo eliminación, permitiendo seleccionar uno a más audios a eliminar, al seleccionar la opción con *ícono de tacho* se eliminarán todos los audios seleccionados de la base datos, ver la Figura 3.46.

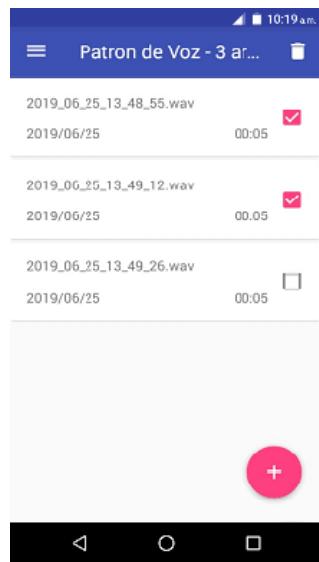


Figura 3.46: Eliminar audios de un usuario del sistema.

Fuente: Elaboración propia

En la Figura 3.47 podemos ver la lista de los accesos de los usuarios del sistema, esta se mostrará a través de la opción *Consultar Accesos* del menú principal del sistema mostrado en la Figura 3.39. En esta interfaz se puede realizar búsquedas por DNI del usuario o por fecha de acceso al sistema, a través de la opción con *ícono de lupa*.

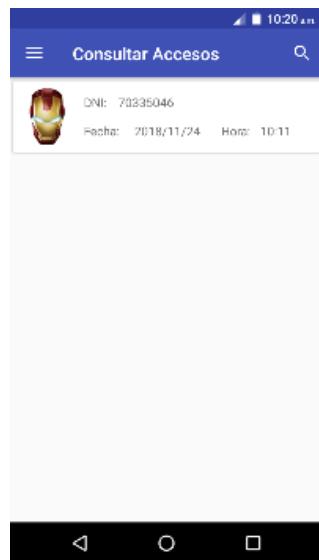


Figura 3.47: Lista de los accesos de los usuarios en el sistema.

Fuente: Elaboración propia

En la Figura 3.48 podemos observar la interfaz que nos permitirá realizar el reconocimiento del usuario para su acceso al sistema, así como los comandos de voz. A través de la opción con *ícono de cámara*, nos permitirá cambiar la foto de la cuenta que ha iniciado sesión, para ello se abrirá la cámara del dispositivo para poder tomar la foto.

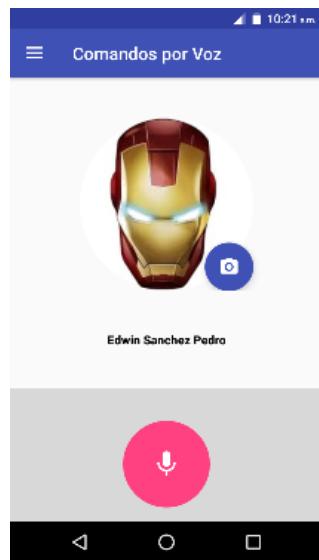


Figura 3.48: Interfaz para el reconocimiento del usuario y comandos por voz.

Fuente: Elaboración propia

En la Figura 3.49 se muestra la grabación de un comando de voz para su reconocimiento por parte del sistema, a través de la opción con *ícono de micrófono* (el tiempo máximo de grabación es de un minuto).

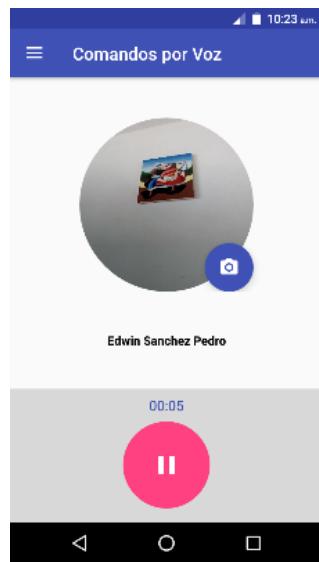


Figura 3.49: Grabación del comando de voz a reconocer.

Fuente: Elaboración propia

Una vez terminada la grabación esta puede ser cancelada o enviada para el reconocimiento del comando de voz, como se ve en la Figura 3.50.

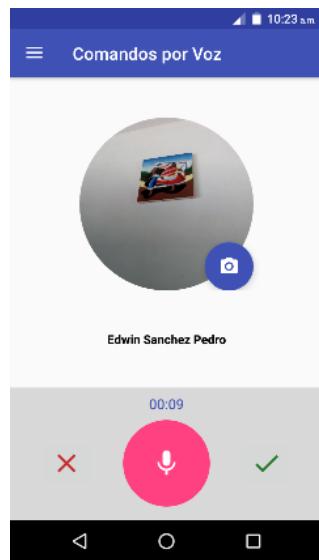


Figura 3.50: Cancelar o enviar el comando de voz para su reconocimiento.

Fuente: Elaboración propia

En el reconocimiento pueden darse los resultados de *usuario reconocido*, *usuario no reconocido*, *usted no es el usuario*, *comando reconocido* o *comando no reconocido*. En la Figura 3.51 se muestra el reconocimiento del comando de voz *encender*. Para esta aplicación solo se reconocerán los comandos *encender*, *apagar* y *jarvis*.

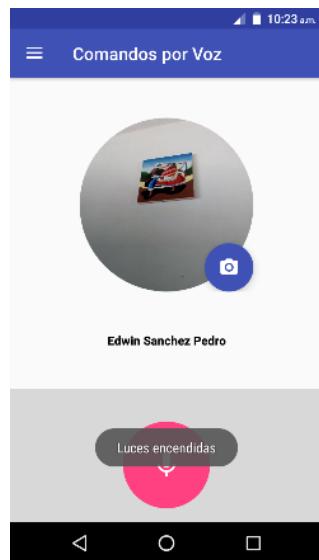


Figura 3.51: Reconocimiento del comando de voz encender.

Fuente: Elaboración propia

En la Figura 3.52 se muestra el mensaje *Sesión Cerrada* después de dar en la opción *Cerrar Sesión* del menú principal de la Figura 3.39, llevándonos a la pantalla principal.



Figura 3.52: Cerrar sesión del usuario en el sistema.

Fuente: Elaboración propia

En la Figura 3.53 se muestra el menú principal antes de iniciar sesión al sistema con una cuenta de usuario.

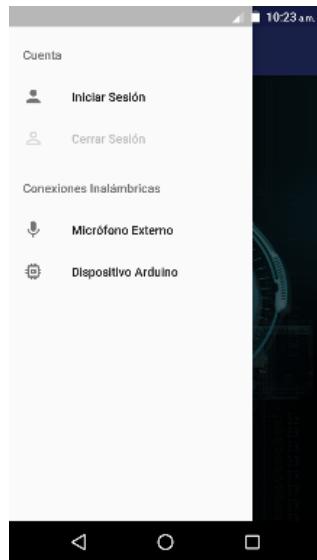


Figura 3.53: Menú de opciones antes de iniciar sesión en el sistema.

Fuente: Elaboración propia

Para iniciar sesión debemos ir a la opción *Iniciar Sesión* del menú principal, ver la Figura 3.53, donde se nos mostrará la interfaz para poder hacer la grabación del patrón clave de voz a reconocer, esta interfaz es la misma que para grabar comandos que se vió en la Figura 3.49. Si el patrón clave de acceso es incorrecto, o si el usuario todavía no se ha registrado o si un usuario quiere hacerse pasar por otro usuario se mostrará un mensaje en la interfaz, tal como se ve en la Figura 3.54.

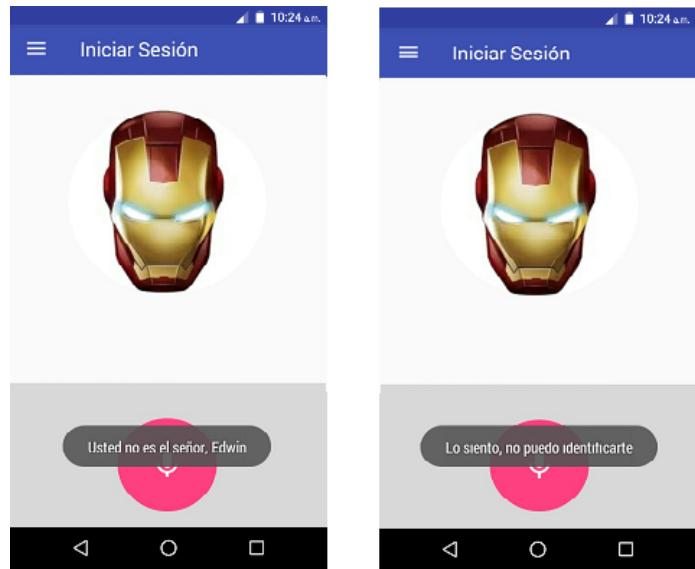


Figura 3.54: Respuestas después de tratar de acceder al sistema.

Fuente: Elaboración propia

Para hacer uso de un micrófono externo al dispositivo móvil para la grabación del ruido del entorno del ambiente y poder hacer la eliminación de este en la señal de voz contaminada, para ello debemos ir al menú principal y a través de la opción *Micrófonos Externos* que se vió en la Figura 3.53, se nos mostrará la lista de micrófonos disponibles en la red, pero primero se deberá activar el Wi-Fi del dispositivo y conectarse a una red donde se encuentre el micrófono del computador a usar, de no ser así se nos mostrará un mensaje de error, ver la Figura 3.55.



Figura 3.55: Ver micrófonos externos disponibles en red para el sistema.

Fuente: Elaboración propia

Para poder establecer la comunicación con el computador y poder usar el micrófono, debemos ir al menú de opciones, ver en la Figura 3.56, y a través de la opción *Activar*, se activará el servidor y creará un cliente local en el dispositivo móvil, para así establecer la comunicación. En la Figura 3.57 se muestra un mensaje de confirmación de que el servidor se activado correctamente y un mensaje del nombre del cliente creado localmente con su dirección IP.



Figura 3.56: Menú de opciones para el uso de un micrófono externo de un computador.

Fuente: Elaboración propia

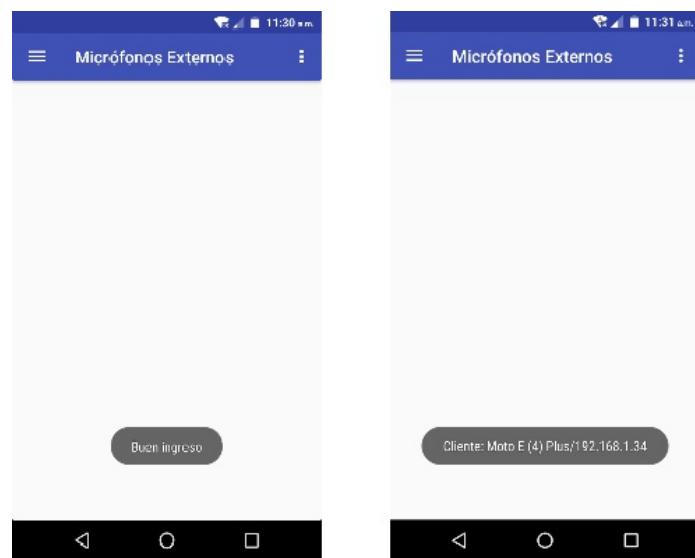


Figura 3.57: Activación del servidor y cliente para establecer la comunicación con el computador.

Fuente: Elaboración propia

Para poder ver los computadores conectados a la misma red del servidor, debemos dar en la opción *Actualizar* del menú mostrado en la Figura 3.56, así se mostrará la lista de los computadores para poder hacer el uso de su micrófono, tal como se muestra en la Figura 3.58.



Figura 3.58: Actualizar lista de computadores conectados en red al servidor.

Fuente: Elaboración propia

En la Figura 3.59 se muestra el mensaje de que un computador disponible acepto la solicitud del dispositivo móvil para hacer el uso de su micrófono para la grabación del ruido. Para desactivar el servidor y cerrar el cliente y no usar el micrófono externo para la filtración de ruido en el ambiente se tendrá que entrar al menú de las opciones que se vió en la Figura 3.56 y dar en *Desactivar*.



Figura 3.59: Solicitar el uso del micrófono de un computador.

Fuente: Elaboración propia

Finalmente, en la Figura 3.60 se muestra un mensaje de error al momento de establecer la comunicación con el Arduino para el control del hardware del sistema, a través de la opción *Dispositivo Arduino* del menú principal, esto se debe a que el Arduino se encuentra apagado. Por otro lado, en la Figura 3.61 se muestra un mensaje de una conexión correcta con el Arduino, permitiendo así el control del hardware del sistema.



Figura 3.60: Error en la conexión con arduino.

Fuente: Elaboración propia

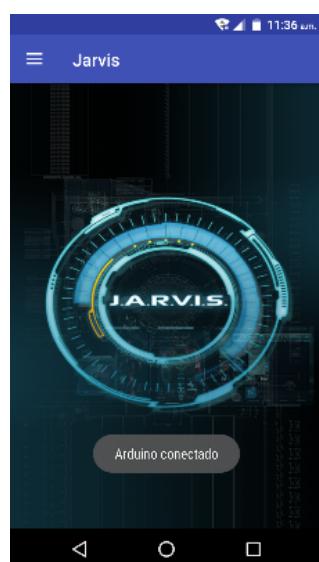


Figura 3.61: Conexión establecida con arduino.

Fuente: Elaboración propia

3.2.2. Desarrollo del software del sistema

En esta parte se mostrará los algoritmos utilizados para establecer la comunicación entre el software de aplicación y el hardware del sistema. En la Figura 3.62 se muestra el código fuente que se implementará en el Arduino para establecer la comunicación con el dispositivo móvil, para poder desde este manipular el hardware del sistema, para nuestro caso solo se realizarán las acciones de *abrir, cerrar, encender y apagar.*

```

#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress serverIP(192,168,1,105);
int serverPort = 8080;
int PIN_LED = 5;
int PIN_PUERTA = 6;
String readString;
EthernetServer server(serverPort);

void setup() {
  Serial.begin(9600);
  Ethernet.begin(mac, serverIP);
  server.begin();
  Serial.print("Server online.");
  pinMode(PIN_LED, OUTPUT);
  pinMode(PIN_PUERTA, OUTPUT);
}

void loop() {
  EthernetClient client = server.available();
  if (client) {
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        if (readString.length() < 30) {
          readString.concat(c);
        }
        if (readString == "led1") {
          digitalWrite(PIN_LED, HIGH);
          Serial.println("Luz Encendida");
          resetString();
        }
        if (readString == "led0") {
          digitalWrite(PIN_LED, LOW);
          Serial.println("Luz Apagada");
          resetString();
        }
        if (readString == "abrir") {
          digitalWrite(PIN_PUERTA, HIGH);
          Serial.println("Puerta Abierta");
          resetString();
        }
        if (readString == "cerrar") {
          digitalWrite(PIN_PUERTA, LOW);
          Serial.println("Puerta Cerrada");
          resetString();
        }
      }
      delay(1);
      client.stop();
    }
  }

  void resetString() {
    readString = "";
}
}

```

Figura 3.62: Código fuente para establecer la comunicación con el dispositivo móvil en arduino.

Fuente: Elaboración propia

En la Figura 3.63 se muestra el código fuente que se implementará en la aplicación del dispositivo móvil para la comunicación con el arduino para la manipulación del hardware del sistema.

```
public Arduino(String ip, int puerto, MainActivity mainActivity) {
    this.ip = ip; //192.168.1.105
    this.puerto = puerto; //8888
    this.dos = null;
    this.dis = null;
    this.socket = null;
    this.conectado = false;
    this.mainActivity = mainActivity;
}

public void conectar() {
    Thread t = new Thread( target: this);
    t.start();
}

@Override
public void run() {
    if (!conectado) {
        try {
            socket = new Socket(ip, puerto);
            dos = new DataOutputStream(socket.getOutputStream());
            dis = new DataInputStream(socket.getInputStream());
            conectado = true;
            mostarMensaje( msg: "Arduino conectado");
        } catch (IOException ex) {
            mostarMensaje( msg: "No se pudo conectar al arduino");
        }
    }
}

public void enviar(String msg) {
    if (conectado) {
        try {
            dos.writeBytes(msg);
            dos.flush();
        } catch(IOException ex) {
            conectado = false;
            mostarMensaje( msg: "No se pudo enviar el comando al arduino");
        }
    }
}

public void desconectar() {
    try {
        dos.close();
        dis.close();
        socket.close();
        conectado = false;
        mostarMensaje( msg: "Arduino desconectado");
    } catch (IOException ex) {
        mostarMensaje( msg: "No se pudo desconectar del arduino");
    }
}
```

Figura 3.63: Código fuente para establecer la comunicación con el arduino en android.
Fuente: Elaboración propia

En la Figura 3.64 se muestra el código fuente que se implementará en la aplicación del computador para establecer la comunicación con el dispositivo móvil para el uso del micrófono en la grabación del ruido.

```

public class Cliente extends Thread {
    private Socket socket;
    private DataInputStream entrada;
    private DataOutputStream salida;
    private boolean conectado;
    private final String usuario;
    private String otroUsuario;
    private final GUINuevoAudio guiCliente;
    public Cliente(String usuario, GUINuevoAudio gui) {
        this.conectado = false;
        this.usuario = usuario;
        this.otroUsuario = null;
        guiCliente = gui;
    }
    public boolean inicializarConexion(String ip) {
        conectarAlServidor(ip, 12354);
        if (conectado) {
            obtenerFlujos();
            enviarInformacion("ingreso@"+usuario);
        }
        return conectado;
    }
    private void conectarAlServidor(String ip, int puerto) {
        try {
            try {
                socket = new Socket(ip, puerto);
                conectado = true;
            } catch (ConnectException ex) {
                String msj = "Ningun servidor tiene esa IP!";
                JOptionPane.showMessageDialog(guiCliente, msj, "Error", JOptionPane.ERROR_MESSAGE);
            }
            catch (IOException ex) {
                String msj = "El servidor esta apagado!";
                JOptionPane.showMessageDialog(guiCliente, msj, "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
        private void obtenerFlujos() {
            try {
                entrada = new DataInputStream(socket.getInputStream());
                salida = new DataOutputStream(socket.getOutputStream());
            } catch (IOException ex) {
                String msj = "No se puede obtener los flujos!";
                JOptionPane.showMessageDialog(guiCliente, msj, "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
        private void cerrarConexion() {
            try {
                conectado = false;
                salida.close();
                entrada.close();
                socket.close();
            } catch (IOException ex) {
                String msj = "No se pudo apagar el cliente!";
                JOptionPane.showMessageDialog(guiCliente, msj, "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}

```

Figura 3.64: Código fuente para establecer la comunicación con el dispositivo móvil en el computador.
Fuente: Elaboración propia

En la Figura 3.65 se muestra la función para enviar y recibir información del servidor en el computador.

```

public void enviarInformacion(String mensaje) {
    try {
        salida.writeUTF(mensaje);
    } catch (IOException ex) {
        String error = "No se pudo enviar la informacion al servidor!";
        JOptionPane.showMessageDialog(guiCliente, error, "Error", JOptionPane.ERROR_MESSAGE);
    }
}
@Override
public void run() {
    while (conectado) {
        try {
            String mensaje = entrada.readUTF();
            if (mensaje.indexOf("buenIngreso") == 0) {
                guiCliente.buenIngreso();
            } else if (mensaje.indexOf("malIngreso") == 0) {
                cerrarConexion();
                guiCliente.malIngreso();
            } else if (mensaje.indexOf("apagar") == 0) {
                cerrarConexion();
            } else if (mensaje.indexOf("apagoServidor") == 0) {
                cerrarConexion();
                guiCliente.apagoServidor();
            } else if (mensaje.indexOf("visibles") == 0) {
                String[] usuarios = mensaje.split("@");
                String[] visibles;
                if (usuarios.length == 1) {
                    visibles = null;
                } else {
                    visibles = new String[usuarios.length-1];
                    for (int i = 0; i < visibles.length; i++) {
                        visibles[i] = usuarios[i+1];
                    }
                }
                guiCliente.cargarListaUsuarios(visibles);
            } else if (mensaje.indexOf("solicitud") == 0) {
                guiCliente.solicitud(mensaje.split("@")[1]);
            } else if (mensaje.indexOf("aceptado") == 0) {
                otroUsuario = mensaje.split("@")[1];
                guiCliente.aceptado(otroUsuario);
            } else if (mensaje.indexOf("rechazado") == 0) {
                otroUsuario = null;
                guiCliente.rechazado(mensaje.split("@")[1]);
            } else if (mensaje.indexOf("empezarGrabar") == 0) {
                guiCliente.empezarGrabar(mensaje.split("@")[1].split("_"));
            } else if (mensaje.indexOf("pararGrabar") == 0) {
                guiCliente.pararGrabar();
            } else if (mensaje.indexOf("audioRuido") == 0) {
                guiCliente.audioRuido(mensaje.split("@")[1]);
            } else if (mensaje.indexOf("saliendo") == 0) {
                otroUsuario = null;
                guiCliente.salio(mensaje.split("@")[1]);
            }
        } catch (IOException ex) {
            String msj = "Cliente no se pudo leer!";
            JOptionPane.showMessageDialog(guiCliente, msj, "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

Figura 3.65: Código fuente para enviar y recibir información al servidor en el computador.

Fuente: Elaboración propia

En la Figura 3.66 se muestra el código fuente que se implementará en la aplicación del dispositivo móvil para establecer la comunicación con el computador para el uso del micrófono para la grabación del ruido.

```

public class Servidor extends Thread {

    public static ArrayList<Atencion> clientes;
    private ServerSocket servidor;
    private boolean conectado;

    public Servidor() {
        clientes = new ArrayList<>();
        servidor = null;
        conectado = false;
    }

    public void encender(int maxClientes) throws IOException {
        servidor = new ServerSocket( port: 1234, maxClientes);
        conectado = true;
        this.start();
    }

    public void apagar() throws IOException {
        while (!clientes.isEmpty()) {
            clientes.get(0).enviarAlCliente( mensaje: "apagoServidor");
            clientes.get(0).cerrarConexion();
            clientes.remove( index: 0);
        }
        conectado = false;
        servidor.close();
        //this.stop();
    }

    @Override
    public void run() {
        while (conectado) {
            try {
                if (!servidor.isClosed()) {
                    //Esperando Conexion...
                    Socket conexion = servidor.accept();
                    //Nueva Conexion...
                    Atencion atencion = new Atencion(conexion);
                    if (atencion.obtenerFlujos()) {
                        atencion.start();
                        clientes.add(atencion);
                    }
                }
            } catch (IOException ex) {
                Log.e( tag: "ERROR", msg: "Problema de I/O en el servidor");
            }
        }
    }
}

```

Figura 3.66: Código fuente para establecer la comunicación con el computador en android.

Fuente: Elaboración propia

En la Figura 3.67 se muestra la función para recibir información del cliente (computador) y poder realizar una acción ante ella en el dispositivo móvil.

```

@Override
public void run() {
    while (conectado) {
        try {
            String mensaje = entrada.readUTF();
            if (mensaje.indexOf("ingreso") == 0) {
                user = mensaje.split( regex: "@") [1];
                if (existe(user)) {
                    enviarAlCliente( mensaje: "malIngreso");
                    salir();
                } else {
                    enviarAlCliente( mensaje: "buenIngreso");
                }
            } else if (mensaje.indexOf("apagar") == 0) {
                enviarAlCliente( mensaje: "apagar");
                salir();
            } else if (mensaje.indexOf("getVisibles") == 0) {
                cargarLista();
            } else if (mensaje.indexOf("activo") == 0) {
                visible = Boolean.parseBoolean(mensaje.split( regex: "@") [1]);
            } else if (mensaje.indexOf("online") == 0) {
                enviarAlOtroCliente(mensaje.split( regex: "@") [1], user, accion: "enviarSolicitud");
            } else if (mensaje.indexOf("aceptar") == 0) {
                String[] cad = mensaje.split( regex: "@");
                enviarAlOtroCliente(cad[1], cad[2], accion: "aceptado");
            } else if (mensaje.indexOf("rechazar") == 0) {
                String[] cad = mensaje.split( regex: "@");
                enviarAlOtroCliente(cad[1], cad[2], accion: "rechazado");
            } else if (mensaje.indexOf("empezar") == 0) {
                String[] cad = mensaje.split( regex: "@");
                enviarAlOtroCliente(cad[1], cad[2], accion: "empezar");
            } else if (mensaje.indexOf("parar") == 0) {
                enviarAlOtroCliente(mensaje.split( regex: "@") [1], mensaje: "", accion: "parar");
            } else if (mensaje.indexOf("audio") == 0) {
                String[] cad = mensaje.split( regex: "@");
                enviarAlOtroCliente(cad[1], cad[2], accion: "audioGrabado");
            } else if (mensaje.indexOf("salir") == 0) {
                String[] cad = mensaje.split( regex: "@");
                enviarAlOtroCliente(cad[1], cad[2], accion: "saliendo");
            }
        } catch (IOException ex) {
            Log.e( tag: "ERROR", ex.toString());
        }
    }
}

```

Figura 3.67: Código fuente para recibir información del computador en android.

Fuente: Elaboración propia

En la Figura 3.68 se muestra la función para enviar información al cliente (computador) para realizar el control del uso de su micrófono para la grabación del ruido en el dispositivo móvil.

```
private void enviarAlOtroCliente(String otroUsuario, String mensaje, String accion) {
    for (Atencion a : Servidor.clientes) {
        if (a.user.equals(otroUsuario)) {
            switch (accion) {
                case "empezar":
                    a.enviarAlCliente( mensaje: "empezarGrabar@"+mensaje);
                    break;
                case "parar":
                    a.enviarAlCliente( mensaje: "pararGrabar");
                    break;
                case "audioGrabado":
                    a.enviarAlCliente( mensaje: "audioRuido@"+mensaje);
                    break;
                case "enviarSolicitud":
                    a.enviarAlCliente( mensaje: "solicitud@"+mensaje);
                    break;
                case "aceptado":
                    a.enviarAlCliente( mensaje: "aceptado@"+mensaje);
                    break;
                case "rechazado":
                    a.enviarAlCliente( mensaje: "rechazado@"+mensaje);
                    break;
                case "saliendo":
                    a.enviarAlCliente( mensaje: "saliendo@"+mensaje);
                    break;
            }
        }
    }
}
```

Figura 3.68: Código fuente para enviar información al computador en android.

Fuente: Elaboración propia

3.2.3. Construcción del hardware

3.2.3.1. Instrumentos

- Un dispositivo móvil con Android 8.0, memoria RAM 3GB y CPU 4x1.4GHz.
- Un computador con Windows 10, memoria RAM 8GB y CPU Intel Core i5 2x2.60GHz.
- Un módem router pasarela con tecnología 802.11 b,g,n.
- Un Arduino Mega 2560 R3.
- Un módulo Ethernet Shield W5100 compatible con Arduino Mega.
- Un módulo relé 4 Songle 10A 250VAC.
- Un transformador DC 9-12V 1A.
- Un transformador DC 5V 500mA.
- Un transformador DC 12V 1A.
- Una cerradura eléctrica YALE.

3.2.3.2. Diseño del hardware del sistema

En la Figura 3.69 se muestra el diseño del hardware para nuestro sistema de seguridad por reconocimiento de voz, en dicho diagrama se indica la conexión e integración de cada componente que conforman el hardware del sistema.

A continuación, explicaremos el rol que cumple cada componente en el funcionamiento del sistema. El arduino mega, este microcontrolador almacena el código fuente que nos permitirá controlar la activación o desactivación de la cerradura eléctrica y el foco, a través de un dispositivo móvil android, para ello se necesita establecer una comunicación inalámbrica con este, es por eso que se utiliza el componente ethernet shield, el cuál nos permite conectar el arduino mega a una red por medio de un cable UTP con entradas RJ45, y este es conectado a un router o móden.

En cuanto al módulo relé, este componente su función es transformar el voltaje de la corriente eléctrica de 0V a 5V proveniente del arduino mega a 220V, que es usada para la activación del foco y la cerradura eléctrica, pero para este último se requiere para su funcionamiento 12V, por lo que se usa un transformador reductor de potencial.

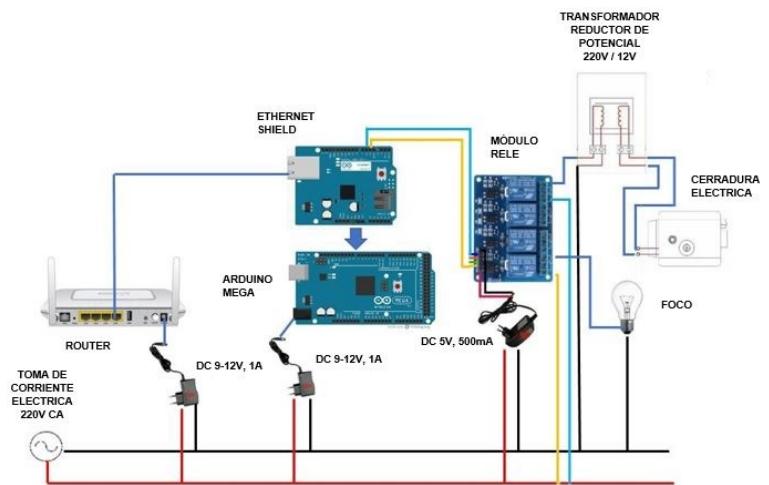


Figura 3.69: Diseño del hardware del sistema de reconocimiento de voz.

Fuente: Elaboración propia

3.2.3.3. Implementación del hardware del sistema

En la Figura 3.70 se muestra la implementación del hardware de nuestro sistema de seguridad por reconocimiento de voz.

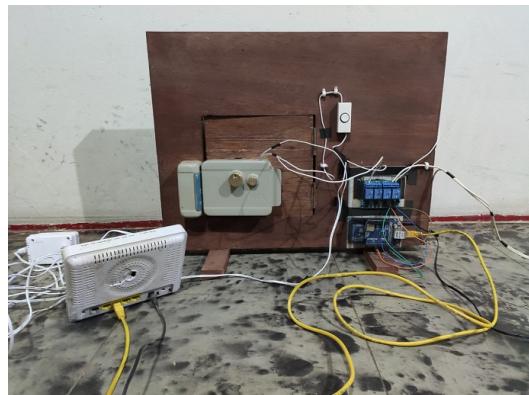


Figura 3.70: Implementación del hardware del sistema de reconocimiento de voz.
Fuente: Elaboración propia

3.3. Fase 3. Testeo del sistema

En esta etapa del desarrollo utilizaremos a 5 personas del sexo masculino con edades entre 20 y 22 años, donde cada una de ellos dirá los números del 1 al 10, repitiendo así este proceso 5 veces, el entorno de grabación será en un ambiente controlado sin ruido.

Una vez recolectado las 250 grabaciones, donde cada persona pronunció cada número 5 veces, 3 de ellos los utilizaremos para el entrenamiento (150 audios) y los 2 restantes para el testeo del sistema (100 audios). Las grabaciones se realizaron con una frecuencia de muestro de 16000 Hz, canal mono, 16 bits de cuantificación y la codificación Little-Endian.

A continuación, veremos los resultados obtenidos de las evaluaciones para cada configuración del algoritmo de reconocimiento de voz, mostrando el número de reconocimientos incorrectos obtenidos, donde *min* es el número mínimo de errores (desaciertos) obtenidos para una configuración (algoritmo con determinados parámetros), *prom* es el valor promedio de los errores obtenidos para una configuración, y finalmente *error* es el promedio de los valores *prom* obtenidos para una configuración.

3.3.1. Hallar BC y M

En este primer testeo hallaremos *BC* y *M*, parámetros del algoritmo MFCC empleado para obtener el patrón característico de una señal de voz, teniendo la siguiente configuración del algo-

ritmo:

- Filtro Preénfasis

$$\alpha = \text{Desde } 0.80 \text{ hasta } 0.97$$

- Segmentación por Hamming

$$V = 30 ; \quad S = 12, 14, 15 \text{ y } 16$$

- MFCC

$$BC = 13, 15, 18, 20, 22 \text{ y } 24 ; \quad M = 13, 15, 26 \text{ y } 39$$

- DTW

$$P = 1 \text{ Simétrico} ; \quad R = 40 ; \quad D = \text{Euclíadiana Cuadrática}$$

Tabla 3.2: Resultados del testeo del sistema para BC = 13, 16 y 18 con M = 13.

α	BC=13 M=13 V=30 P=S1 R=40				BC=16 M=13 V=30 P=S1 R=40				BC=18 M=13 V=30 P=S1 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	4	4	3	4	7	5	8	8	6	5	8	6
0.81	4	4	3	4	6	6	7	9	5	6	10	6
0.82	4	4	3	4	7	3	7	10	7	7	10	6
0.83	4	4	3	4	7	4	6	10	8	7	10	6
0.84	4	4	3	4	7	4	4	9	8	7	10	9
0.85	4	4	3	4	8	6	5	7	8	8	10	8
0.86	4	4	3	4	6	4	5	6	7	6	9	7
0.87	4	4	3	4	7	5	6	6	7	5	7	8
0.88	4	4	3	4	5	6	3	6	7	3	5	7
0.89	4	4	3	4	5	5	5	8	6	5	5	7
0.9	4	4	3	4	5	5	6	8	6	4	4	6
0.91	4	4	3	4	4	6	6	6	6	3	5	6
0.92	4	4	3	4	7	6	5	5	8	4	3	8
0.93	4	4	3	4	5	6	7	5	7	4	5	7
0.94	4	4	3	4	5	5	5	5	8	3	6	8
0.95	4	4	3	4	5	4	6	5	8	5	7	6
0.96	4	4	3	3	8	4	7	7	7	5	4	4
0.97	4	4	3	3	6	6	5	6	9	5	4	7
min	4	4	3	3	4	3	3	5	5	3	3	4
prom	4	4	3	3.8888889	6.111111	5	5.722222	7	7.111111	5.111111	6.777778	6.777778
error				3.72222222			5.958333333			6.444444444		

Fuente: Elaboración propia

Tabla 3.3: Resultados del testeo del sistema para $BC = 20, 22$ y 24 con $M = 13$.

α	BC=20 M=13 V=30 P=S1 R=40				BC=22 M=13 V=30 P=S1 R=40				BC=24 M=13 V=30 P=S1 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	5	7	6	6	4	4	3	3	5	4	3	3
0.81	5	7	7	6	4	4	3	3	5	4	3	3
0.82	5	7	8	7	4	4	3	3	5	4	3	3
0.83	7	6	5	7	4	4	3	3	5	4	3	3
0.84	6	7	7	7	4	4	3	3	5	4	3	3
0.85	6	6	8	7	4	4	3	3	5	4	3	3
0.86	6	6	7	7	4	4	3	3	5	4	3	3
0.87	5	6	7	6	4	4	3	3	5	4	3	3
0.88	6	7	6	6	4	4	3	3	5	4	3	3
0.89	6	7	8	8	4	4	3	3	5	4	3	3
0.9	8	6	5	6	4	4	3	3	5	4	3	3
0.91	8	6	5	6	4	4	3	3	5	4	3	3
0.92	7	4	4	5	4	4	3	3	5	4	3	3
0.93	6	5	6	6	4	4	3	3	5	4	3	3
0.94	7	6	5	6	4	4	3	3	5	4	3	3
0.95	7	5	6	5	4	4	3	3	4	4	3	3
0.96	5	4	6	5	4	4	3	3	4	4	3	3
0.97	3	4	5	6	4	4	3	3	4	4	3	3
min	3	4	4	2	4	4	3	3	4	4	3	3
prom	6	5.888889	6.166667	6.222222	4	4	3	3	4.833333	4	3	3
error		6.069444444				3.5			3.708333333			

Fuente: Elaboración propia

Como vemos en las Tablas 3.2 y 3.3 para $M = 13$ el mejor resultado fue con $BC = 22$ donde el error promedio fue de 3.5 y el segundo mejor fue para $BC = 24$ con un error promedio de 3.7.

Tabla 3.4: Resultados del testeo del sistema para $BC = 13, 16$ y 18 con $M = 15$.

α	BC=13 M=15 V=30 P=S1 R=40				BC=16 M=15 V=30 P=S1 R=40				BC=18 M=15 V=30 P=S1 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	4	4	3	4	7	6	8	8	7	5	9	6
0.81	4	4	3	4	7	6	7	9	7	7	10	6
0.82	4	4	3	4	8	4	7	9	7	7	11	7
0.83	4	4	3	4	7	3	5	10	8	6	10	6
0.84	4	4	3	4	7	4	4	11	9	8	10	9
0.85	4	4	3	4	8	5	5	7	7	7	11	8
0.86	4	4	3	4	7	6	4	7	7	6	9	7
0.87	4	4	3	4	6	4	4	7	7	5	6	8
0.88	4	4	3	4	6	8	3	8	7	3	5	7
0.89	4	4	3	4	5	5	5	9	7	5	6	7
0.9	4	4	3	4	6	5	6	9	7	4	3	5
0.91	4	4	3	4	4	6	6	6	6	3	5	6
0.92	4	4	3	4	7	5	5	6	5	3	3	8
0.93	4	4	3	4	5	5	5	6	7	4	5	7
0.94	4	4	3	4	5	5	7	6	9	4	6	8
0.95	4	4	3	4	5	4	5	5	9	5	6	6
0.96	4	4	3	3	6	6	7	7	7	5	4	4
0.97	4	4	3	3	6	7	5	7	8	5	4	6
min	4	4	3	3	4	3	3	5	5	3	3	4
prom	4	4	3	3.888889	6.222222	5.222222	5.444444	7.611111	7.277778	5.111111	6.833333	6.722222
error		3.722222222				6.125				6.486111111		

Fuente: Elaboración propia

Tabla 3.5: Resultados del testeo del sistema para $BC = 20, 22$ y 24 con $M = 15$.

α	BC=20 M=15 V=30 P=S1 R=40				BC=22 M=15 V=30 P=S1 R=40				BC=24 M=15 V=30 P=S1 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	6	6	7	6	5	4	3	3	3	4	3	3
0.81	6	6	6	6	5	4	3	3	3	4	3	3
0.82	7	7	6	7	5	4	3	3	3	4	3	3
0.83	7	6	6	7	5	4	3	3	3	4	3	3
0.84	6	6	6	7	5	4	3	3	3	4	3	3
0.85	6	6	7	7	5	4	3	3	3	4	3	3
0.86	6	6	8	7	5	4	3	3	3	4	3	3
0.87	5	6	7	7	4	4	3	3	3	4	3	3
0.88	6	7	7	7	4	4	3	3	3	4	3	3
0.89	6	6	7	7	4	4	3	3	3	4	3	3
0.9	7	6	6	7	4	4	3	3	4	4	3	3
0.91	8	6	4	6	4	4	3	3	4	4	3	3
0.92	7	6	5	6	4	4	3	3	4	4	3	3
0.93	6	4	6	6	4	4	3	3	4	4	3	3
0.94	7	6	6	5	4	3	3	3	4	4	3	3
0.95	6	5	5	6	4	3	3	3	3	4	3	3
0.96	5	4	6	5	4	3	3	3	3	4	3	3
0.97	4	4	5	5	4	3	3	3	3	3	3	3
min	4	4	4	5	4	3	3	3	3	3	3	3
prom	6.166667	5.722222	6.111111	6.333333	4.3888889	3.777778	3	3	3.2777778	3.944444	3	3
error	6.083333333				3.541666667				3.305555556			

Fuente: Elaboración propia

Como vemos en las Tablas 3.4 y 3.5 para $M = 15$ el mejor resultado fue con $BC = 24$ donde el error promedio fue de 3.3 y el segundo mejor fue para $BC = 22$ con un error promedio de 3.5.

Tabla 3.6: Resultados del testeo del sistema para $BC = 13, 16$ y 18 con $M = 26$.

α	BC=13 M=26 V=30 P=S1 R=40				BC=16 M=26 V=30 P=S1 R=40				BC=18 M=26 V=30 P=S1 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	14	10	7	8	14	14	16	15	14	13	13	13
0.81	11	12	10	8	17	14	15	17	15	15	13	15
0.82	10	11	9	8	17	13	15	17	16	14	9	16
0.83	12	9	7	10	16	15	12	15	16	14	11	15
0.84	9	13	8	13	14	15	13	17	16	14	12	14
0.85	15	11	9	11	14	13	11	15	12	11	9	14
0.86	13	11	10	14	14	13	13	12	11	10	9	14
0.87	12	11	9	13	18	13	9	11	11	11	11	12
0.88	15	11	11	14	17	12	11	11	10	13	11	13
0.89	13	11	10	13	16	9	11	12	12	9	12	12
0.9	14	11	11	13	17	10	13	13	17	10	12	14
0.91	14	11	12	12	18	15	12	13	14	10	13	15
0.92	12	11	12	11	19	18	13	13	15	13	18	13
0.93	14	11	10	14	15	19	15	12	14	13	16	13
0.94	14	11	13	11	15	18	13	10	13	14	16	13
0.95	15	10	9	9	16	14	13	11	15	14	16	12
0.96	14	11	8	9	14	12	12	10	12	12	15	10
0.97	13	11	8	12	18	14	13	12	13	11	12	10
min	9	9	7	8	14	9	9	10	10	9	9	10
prom	13	10.94444	9.611111	11.27778	16.05556	13.94444	12.77778	13.11111	13.66667	12.27778	12.66667	13.22222
error	11.208333333				13.972222222				12.958333333			

Fuente: Elaboración propia

Tabla 3.7: Resultados del testeo del sistema para $BC = 20, 22$ y 24 con $M = 26$.

α	BC=20 M=26 V=30 P=S1 R=40				BC=22 M=26 V=30 P=S1 R=40				BC=24 M=26 V=30 P=S1 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	15	12	10	9	5	5	4	4	4	5	4	4
0.81	12	13	11	11	5	5	4	4	4	5	4	4
0.82	14	11	13	10	5	5	4	4	4	5	4	4
0.83	15	11	11	9	5	5	4	4	4	5	4	4
0.84	13	10	11	9	5	5	4	4	4	5	4	4
0.85	14	10	11	10	5	5	4	4	4	5	4	4
0.86	13	9	10	10	5	5	4	4	4	5	4	4
0.87	12	9	11	12	5	5	4	4	4	5	4	4
0.88	12	11	8	12	5	5	4	4	4	5	4	4
0.89	12	10	9	12	5	5	4	4	4	5	4	4
0.9	12	7	11	9	5	5	4	4	4	5	4	4
0.91	14	9	12	8	5	5	4	4	4	5	4	4
0.92	12	9	12	9	5	5	4	4	4	5	4	4
0.93	13	9	10	9	5	5	4	4	4	5	4	4
0.94	12	12	8	9	5	5	4	4	4	5	4	4
0.95	13	11	7	10	5	5	4	4	4	5	4	4
0.96	13	10	8	8	5	6	4	4	4	5	4	4
0.97	11	10	9	8	5	6	5	4	4	5	4	4
min	11	7	7	8	5	5	4	4	4	5	4	4
prom	12.88889	10.16667	10.11111	9.666667	5	5.111111	4.055556	4	4	5	4	4
error		10.70833333				4.541666667				4.25		

Fuente: Elaboración propia

Como vemos en las Tablas 3.6 y 3.7 para $M = 26$ el mejor resultado fue con $BC = 24$ donde el error promedio fue de 4.2 y el segundo mejor fue para $BC = 22$ con un error promedio de 4.5.

Tabla 3.8: Resultados del testeo del sistema para $BC = 13, 16$ y 18 con $M = 39$.

α	BC=13 M=39 V=30 P=S1 R=40				BC=16 M=39 V=30 P=S1 R=40				BC=18 M=39 V=30 P=S1 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	19	16	17	18	19	21	21	19	21	14	18	19
0.81	19	16	16	17	19	22	21	19	19	17	17	18
0.82	17	19	18	18	20	23	22	22	20	14	18	17
0.83	18	20	18	16	21	22	21	23	20	14	18	20
0.84	18	18	19	16	17	23	19	26	18	15	18	19
0.85	19	17	18	17	21	22	20	20	16	12	19	19
0.86	20	18	20	21	20	19	19	19	15	12	17	18
0.87	19	17	17	18	21	21	20	16	14	13	18	17
0.88	18	18	20	19	21	22	21	20	17	13	20	17
0.89	18	15	18	17	21	18	18	17	17	15	19	18
0.9	22	16	24	18	21	20	19	18	17	14	19	17
0.91	20	15	19	17	22	21	21	20	18	18	21	19
0.92	21	16	19	16	22	24	19	17	18	18	18	18
0.93	18	18	18	15	20	22	19	16	14	18	22	17
0.94	16	19	19	17	18	23	20	17	17	20	21	17
0.95	16	19	16	14	19	22	21	18	18	16	20	18
0.96	15	20	15	15	19	22	17	16	17	15	20	17
0.97	17	19	13	16	21	24	19	17	16	15	17	17
min	15	15	13	14	17	18	17	16	14	12	17	17
prom	18.33333	17.555556	18	16.94444	20.11111	21.72222	19.83333	18.88889	17.33333	15.1.6667	18.88889	17.88889
error		17.70833333				20.13888889				17.31944444		

Fuente: Elaboración propia

Tabla 3.9: Resultados del testeo del sistema para $BC = 20, 22$ y 24 con $M = 39$.

α	BC=20 M=39 V=30 P=S1 R=40				BC=22 M=39 V=30 P=S1 R=40				BC=24 M=39 V=30 P=S1 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	18	17	17	14	11	11	11	5	12	10	9	5
0.81	17	20	19	15	11	11	11	5	12	10	9	5
0.82	18	18	17	16	11	11	11	5	12	10	9	5
0.83	20	16	19	13	11	11	11	5	12	10	9	5
0.84	20	14	17	13	11	11	11	5	12	10	9	5
0.85	18	15	18	15	11	11	11	5	12	10	9	5
0.86	17	14	18	14	11	11	11	5	12	10	9	5
0.87	14	13	16	12	11	11	11	5	12	10	9	5
0.88	17	14	16	14	11	11	11	5	12	10	9	5
0.89	17	15	15	12	11	11	11	5	12	10	9	5
0.9	16	13	17	12	11	11	11	5	12	10	9	5
0.91	17	16	17	13	11	11	11	5	12	10	9	5
0.92	16	16	16	12	11	11	11	5	12	10	9	5
0.93	16	17	14	13	11	11	11	5	12	10	9	5
0.94	16	16	17	14	11	11	11	5	12	10	9	5
0.95	16	14	14	14	11	11	11	5	12	10	9	5
0.96	16	15	14	14	11	11	11	5	11	10	10	5
0.97	15	16	14	12	11	11	11	5	11	10	10	5
min	14	13	14	12	11	11	11	5	11	10	9	5
prom	16.88889	15.5	16.38889	12.44444	11	11	11	5	11.88889	10	9.111111	5
error	15.55555556				9.5				9			

Fuente: Elaboración propia

Como vemos en las Tablas 3.8 y 3.9 para $M = 39$ el mejor resultado obtenido fue con $BC = 24$ donde el error promedio fue de 9 y el segundo mejor fue para $BC = 22$ con un error promedio de 9.5. Podemos concluir de este primer testeo del sistema que se obtuvieron mejores resultados con MFCC ($M = 13, BC = 22$) y ($M = 15, BC = 24$), que los MFCC Delta y Doble Delta.

3.3.2. Hallar V

En este segundo testeo hallaremos V , parámetro usado para la segmentación o ventaneamiento de una señal de voz, teniendo la siguiente configuración del algoritmo:

- Filtro Preénfasis

$$\alpha = \text{Desde } 0.80 \text{ hasta } 0.97$$

- Segmentación por Hamming

$$V = 28, 30 \text{ y } 32; \quad S = 12, 14, 15 \text{ y } 16$$

- MFCC

$$BC = 22 \text{ y } 24; \quad M = 13 \text{ y } 15$$

- DTW

$$P = 1 \text{ Simétrico}; \quad R = 40; \quad D = \text{Euclíadiana Cuadrática}$$

Tabla 3.10: Resultados del testeo del sistema para V = 28.

α	BC=22 M=13 V=28 P=S1 R=40				BC=24 M=15 V=28 P=S1 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	4	3	3	3	3	3	3	4
0.81	4	3	3	3	3	3	3	4
0.82	4	3	3	3	3	3	3	4
0.83	4	3	3	3	3	3	3	4
0.84	4	3	3	3	3	3	3	4
0.85	4	3	3	3	3	3	3	4
0.86	4	3	3	3	3	2	3	3
0.87	4	3	3	3	3	2	3	3
0.88	3	3	3	3	3	2	3	3
0.89	3	3	3	3	3	2	3	3
0.9	3	3	3	3	3	2	3	3
0.91	3	3	3	3	3	2	3	3
0.92	3	3	3	3	3	2	3	3
0.93	3	3	3	3	3	3	3	3
0.94	3	3	3	3	3	3	3	3
0.95	3	3	3	3	3	3	3	3
0.96	3	3	3	3	3	3	3	3
0.97	3	3	3	3	3	3	3	3
min	3	3	3	3	3	2	3	3
prom	3.4444444	3	3	3	3	2.61111111	3	3.3333333
error		3.11111111			2.986111111			

Fuente: Elaboración propia

Tabla 3.11: Resultados del testeo del sistema para V = 30.

α	BC=22 M=13 V=30 P=S1 R=40				BC=24 M=15 V=30 P=S1 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	4	3	3	3	3	3	3	4
0.81	4	4	3	3	3	4	3	3
0.82	4	4	3	3	3	4	3	3
0.83	4	4	3	3	3	4	3	3
0.84	4	4	3	3	3	4	3	3
0.85	4	4	3	3	3	4	3	3
0.86	4	4	3	3	3	4	3	3
0.87	4	4	3	3	3	4	3	3
0.88	4	4	3	3	3	4	3	3
0.89	4	4	3	3	3	4	3	3
0.9	4	4	3	3	4	4	3	3
0.91	4	4	3	3	4	4	3	3
0.92	4	4	3	3	4	4	3	3
0.93	4	4	3	3	4	4	3	3
0.94	4	4	3	3	4	4	3	3
0.95	4	4	3	3	3	4	3	3
0.96	4	4	3	3	3	4	3	3
0.97	4	4	3	3	3	3	3	3
min	4	4	3	3	3	3	3	3
prom	4	4	3	3	3.2777778	3.9444444	3	3
error				3.5		3.305555556		

Fuente: Elaboración propia

Tabla 3.12: Resultados del testeo del sistema para V = 32.

α	BC=22 M=13 V=32 P=S1 R=40				BC=24 M=15 V=32 P=S1 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	5	4	3	3	3	4	3	3
0.81	5	4	3	3	3	4	3	3
0.82	5	4	3	3	3	4	3	3
0.83	5	4	3	3	3	4	3	3
0.84	5	4	3	3	3	4	3	3
0.85	5	4	3	3	3	4	3	3
0.86	5	4	3	3	3	4	3	3
0.87	4	4	3	3	3	4	3	3
0.88	4	4	3	3	3	4	3	3
0.89	4	4	3	3	3	4	3	3
0.9	4	4	3	3	4	4	3	3
0.91	4	4	3	3	4	4	3	3
0.92	4	4	3	3	4	4	3	3
0.93	4	4	3	3	4	4	3	3
0.94	4	3	3	3	4	4	3	3
0.95	4	3	3	3	3	4	3	3
0.96	4	3	3	3	3	4	3	3
0.97	4	3	3	3	3	3	3	3
min	4	3	3	3	3	3	3	3
prom	4.3888889	3.7777778	3	3	3.2777778	3.9444444	3	3
error		3.541666667				3.305555556		

Fuente: Elaboración propia

Con los resultados de las Tablas 3.10, 3.11 y 3.12 podemos concluir que se obtuvieron mejores resultados con $V = 28$, $BC = 24$ y $M = 15$ fue el mejor con un error promedio de 2.9.

3.3.3. Hallar P

En este tercer testeo hallaremos P , parámetro del algoritmo DTW usado para la comparación de patrones característicos de las señales de voz, teniendo la siguiente configuración del algoritmo:

- Filtro Preéñfasis

$$\alpha = \text{Desde } 0.80 \text{ hasta } 0.97$$

- Segmentación por Hamming

$$V = 28 ; \quad S = 12, 14, 15 \text{ y } 16$$

- MFCC

$$BC = 24 ; \quad M = 15$$

- DTW

$$P = 0, 1/2, 1 \text{ y } 2 \text{ (Simétrico y Asimétrico)} ; \quad R = 40 ; \quad D = \text{Euclíadiana Cuadrática}$$

Tabla 3.13: Resultados del testeo del sistema para $P = 0$ simétrico y asimétrico.

α	BC=24 M=15 V=28 P=S0 R=40				BC=24 M=15 V=28 P=A0 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	15	21	26	42	25	32	34	54
0.81	15	21	26	42	25	32	34	54
0.82	15	21	27	42	25	32	34	54
0.83	15	21	27	42	25	32	34	54
0.84	15	21	27	42	25	32	34	54
0.85	15	21	27	42	25	32	34	54
0.86	15	21	28	42	25	32	34	54
0.87	15	21	28	43	25	32	34	54
0.88	15	21	28	43	25	32	34	54
0.89	15	21	28	43	25	32	34	54
0.9	15	21	28	43	25	32	34	54
0.91	15	21	28	43	25	32	34	54
0.92	15	21	28	44	25	32	34	54
0.93	16	21	28	44	25	32	34	54
0.94	16	21	28	44	25	32	35	54
0.95	16	21	28	44	24	32	35	54
0.96	16	21	28	45	24	32	35	53
0.97	16	21	28	45	24	32	36	53
min	15	21	26	42	24	32	34	53
prom	15.277778	21.055556	27.555556	43.055556	24.833333	32	34.277778	53.888889
error		26.73611111				36.25		

Fuente: Elaboración propia

Tabla 3.14: Resultados del testeo del sistema para $P = 1$ simétrico y asimétrico.

α	BC=24 M=15 V=28 P=S1 R=40				BC=24 M=15 V=28 P=A1 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	1	2	2	3	4	6	4	4
0.81	1	2	2	3	4	5	4	4
0.82	1	2	2	3	4	5	4	4
0.83	1	2	2	3	4	5	4	4
0.84	1	2	2	3	4	5	4	4
0.85	1	2	2	3	4	5	4	4
0.86	1	2	2	3	4	5	4	4
0.87	1	2	2	3	4	5	4	4
0.88	1	2	2	3	4	5	4	4
0.89	1	2	2	3	4	5	4	4
0.9	1	2	2	3	4	5	4	4
0.91	1	2	2	3	4	5	4	4
0.92	1	2	2	3	4	5	4	4
0.93	1	2	2	3	4	5	4	4
0.94	1	2	2	3	4	5	5	4
0.95	1	2	2	3	4	5	5	4
0.96	1	2	2	3	4	5	5	4
0.97	2	2	2	3	4	5	5	5
min	1	2	2	3	4	5	4	4
prom	1.0555556	2	2	3	4	5.0555556	4.2222222	4.0555556
error		2.013888889				4.333333333		

Fuente: Elaboración propia

Tabla 3.15: Resultados del testeo del sistema para $P = 1/2$ simétrico y asimétrico.

α	BC=24 M=15 V=28 P=S1/2 R=40				BC=24 M=15 V=28 P=A1/2 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	3	3	3	4	3	3	3	3
0.81	3	3	3	4	3	3	3	3
0.82	3	3	3	4	3	3	3	3
0.83	3	3	3	4	3	3	3	3
0.84	3	3	3	4	3	3	3	3
0.85	3	3	3	4	3	3	3	3
0.86	3	2	3	3	3	3	3	3
0.87	3	2	3	3	3	3	3	3
0.88	3	2	3	3	3	3	3	3
0.89	3	2	3	3	3	3	3	3
0.9	3	2	3	3	3	3	3	3
0.91	3	2	3	3	3	3	3	3
0.92	3	2	3	3	3	3	3	3
0.93	3	3	3	3	4	3	3	3
0.94	3	3	3	3	4	3	3	3
0.95	3	3	3	3	4	3	3	3
0.96	3	3	3	3	4	3	3	3
0.97	3	3	3	3	4	3	4	3
min	3	2	3	3	3	3	3	3
prom	3	2.6111111	3	3.3333333	3.2777778	3	3.0555556	3
error		2.986111111				3.083333333		

Fuente: Elaboración propia

Tabla 3.16: Resultados del testeo del sistema para $P = 2$ simétrico y asimétrico.

α	BC=24 M=15 V=28 P=S2 R=40				BC=24 M=15 V=28 P=A2 R=40			
	S=16	S=15	S=14	S=12	S=16	S=15	S=14	S=12
0.8	7	7	5	7	81	79	83	82
0.81	7	7	5	7	81	79	83	82
0.82	7	7	5	7	81	79	83	82
0.83	7	7	5	7	81	79	83	82
0.84	7	7	5	7	81	79	83	82
0.85	7	7	5	7	81	79	83	82
0.86	7	7	5	7	81	79	83	82
0.87	7	7	5	7	81	80	83	82
0.88	7	7	5	7	81	80	83	82
0.89	7	7	5	7	81	80	83	82
0.9	7	7	5	7	81	80	83	82
0.91	7	7	5	7	81	80	83	82
0.92	7	7	5	7	81	80	83	82
0.93	7	7	5	7	81	80	83	82
0.94	7	7	5	7	81	80	83	82
0.95	7	7	5	7	81	79	83	82
0.96	7	7	5	7	81	79	83	82
0.97	7	7	5	7	81	79	83	82
min	7	7	5	7	81	79	83	82
prom	7	7	5	7	81	79.444444	83	82
error		6.5				3.81.36111111		

Fuente: Elaboración propia

A partir de los resultados obtenidos y vistos en las Tablas 3.13, 3.14, 3.15 y 3.16 podemos concluir que se obtuvieron mejores resultados para $P = 1$ simétrico con un error promedio de 2.0, por lo que se cumplió con lo que se dijo en la teoría para el valor de P , y en segundo lugar para $P = 1/2$ simétrico con un error promedio de 2.9.

3.3.4. Hallar S

En este cuarto testeo hallaremos S , parámetro usado para la segmentación o ventaneamiento de una señal de voz, teniendo la siguiente configuración del algoritmo:

- Filtro Preénfasis

$$\alpha = \text{Desde } 0.80 \text{ hasta } 0.97$$

- Segmentación por Hamming

$$V = 28 ; \quad S = 12, 14, 15, 16, 17, 18, 19, 20 \text{ y } 21$$

- MFCC

$$BC = 24 ; \quad M = 15$$

- DTW

$$P = 1/2 \text{ y } 1 \text{ Simétrico} ; \quad R = 40 ; \quad D = \text{Euclíadiana Cuadrática}$$

Tabla 3.17: Resultados del testeo del sistema para $P = 1/2$ simétrico.

α	BC=24 M=15 V=28 P=S1/2 R=40								
	S=21	S=20	S=19	S=18	S=17	S=16	S=15	S=14	S=12
0.8	1	2	1	1	1	1	2	2	3
0.81	1	2	1	1	1	1	2	2	3
0.82	1	2	1	1	1	1	2	2	3
0.83	1	2	1	1	1	1	2	2	3
0.84	1	2	1	1	1	1	2	2	3
0.85	1	2	1	1	1	1	2	2	3
0.86	1	2	1	1	1	1	2	2	3
0.87	1	2	1	1	1	1	2	2	3
0.88	1	2	1	1	1	1	2	2	3
0.89	1	2	1	1	1	1	2	2	3
0.9	1	2	1	1	1	1	2	2	3
0.91	1	2	1	1	1	1	2	2	3
0.92	1	2	1	1	1	1	2	2	3
0.93	1	2	1	1	1	1	2	2	3
0.94	1	2	1	1	1	1	2	2	3
0.95	1	2	1	1	1	1	2	2	3
0.96	2	2	2	1	1	1	2	2	3
0.97	2	2	2	1	1	2	2	2	3
min	1	2	1	1	1	1	2	2	3
prom	1.1111111	2	1.1111111	1	1	1.0555556	2	2	3
error						1.586419753			

Fuente: Elaboración propia

Tabla 3.18: Resultados del testeo del sistema para $P = 1$ simétrico.

α	BC=24 M=15 V=28 P=S1 R=40								
	S=21	S=20	S=19	S=18	S=17	S=16	S=15	S=14	S=12
0.8	3	2	2	4	2	1	1	3	3
0.81	3	2	2	4	2	1	1	3	3
0.82	3	2	2	4	2	1	1	3	3
0.83	3	2	2	4	2	1	1	3	3
0.84	3	2	2	4	2	1	1	3	3
0.85	3	2	2	4	2	1	1	3	3
0.86	3	2	2	4	2	1	1	3	3
0.87	3	2	2	4	2	1	1	3	3
0.88	3	2	2	4	2	1	1	3	3
0.89	3	2	2	4	2	1	1	3	3
0.9	3	2	2	4	2	1	1	3	3
0.91	3	2	2	4	2	1	1	3	3
0.92	3	2	2	4	2	1	1	3	3
0.93	3	2	2	4	2	1	1	3	3
0.94	3	2	2	4	2	1	1	3	3
0.95	3	2	2	4	2	1	1	3	4
0.96	3	2	2	4	2	1	1	3	4
0.97	3	2	2	3	2	1	1	3	3
min	3	2	2	3	2	1	1	3	3
prom	3	2	2	3.9444444	2	1	1	3	3.1111111
error						2.339506173			

Fuente: Elaboración propia

A partir de los resultados obtenidos y vistos en las Tablas 3.17 y 3.18 podemos concluir que se obtuvieron mejores resultados para $P = 1/2$ Simétrico con un error promedio de 1.58 donde $S = 17$ y $S = 18$ obtuvieron un *prom* = 1.

3.3.5. Hallar S

En el testeo anterior no se pudo decidir el valor de $S = 17$ o $S = 18$, ya que ambos tienen de error promedio 1.0, para ello realizaremos un quinto testeo calculando la diferencia de ese error, teniendo la siguiente configuración del algoritmo:

- Filtro Preénfasis

$$\alpha = \text{Desde } 0.80 \text{ hasta } 0.97$$

- Segmentación por Hamming

$$V = 28 ; \quad S = 17 \text{ y } 18$$

- MFCC

$$BC = 24 ; \quad M = 15$$

- DTW

$$P = 1 \text{ Simétrico} ; \quad R = 40 ; \quad D = \text{Euclíadiana Cuadrática}$$

Tabla 3.19: Resultados del testeo del sistema para $P = 1$ simétrico con $S = 17$ y 18 .

α	BC=24 M=15 V=28 P=S1 R=40	
	S=17	S=18
0.8	0.1094124943	0.177730377
0.81	0.109218433	0.1777551714
0.82	0.109042847	0.177391216
0.83	0.108885997	0.177253609
0.84	0.108749186	0.177144889
0.85	0.108632233	0.177072825
0.86	0.108533699	0.177047698
0.87	0.1084665	0.177083403
0.88	0.108441072	0.177199142
0.89	0.108469676	0.177422935
0.9	0.108568203	0.177796984
0.91	0.108757903	0.178388854
0.92	0.109073457	0.179303729
0.93	0.109559355	0.180665003
0.94	0.110277193	0.182589138
0.95	0.111342347	0.185213138
0.96	0.112971013	0.18867958
0.97	0.115673769	0.193678132
min	0.108441072	0.177047698
prom	0.109412494	0.179956242
error	1	1

Fuente: Elaboración propia

Como podemos ver en la Tabla 3.19 se obtuvo mejores resultados con $S = 17$ con una diferencia de error promedio de $prom = 0,10$ mientras que con $S = 18$ se obtuvo una diferencia de error promedio de $prom = 0,17$.

3.3.6. Hallar r

En este sexto testeo hallaremos r , parámetro usado por el algoritmo DTW empleado para la comparación de patrones de las señales de voz, teniendo la siguiente configuración del algoritmo:

- Filtro Preénfasis

$$\alpha = \text{Desde } 0.80 \text{ hasta } 0.97$$

- Segmentación por Hamming

$$V = 28 ; \quad S = 17$$

- MFCC

$$BC = 24 ; \quad M = 15$$

- DTW

$$P = 1/2 \text{ y } 1 \text{ Simétrico} ; R = 20, 25, 30, 35, 40, 45, 50, 55 \text{ y } 60 ; D = \text{Euclíadiana Cuadrática}$$

Tabla 3.20: Resultados del testeo del sistema para $P = 1/2$ simétrico con $S = 17$ y 18 .

α	BC=24 M=15 V=28 S=17 P=S1/2									
	R=20	R=15	R=30	R=35	R=40	R=45	R=50	R=55	R=60	
0.8	14	5	2	2	1	1	1	1	1	
0.81	14	5	2	2	1	1	1	1	1	
0.82	14	5	2	2	1	1	1	1	1	
0.83	13	5	2	2	1	1	1	1	1	
0.84	13	5	2	2	1	1	1	1	1	
0.85	13	5	2	2	1	1	1	1	1	
0.86	13	5	2	2	1	1	1	1	1	
0.87	13	5	2	2	1	1	1	1	1	
0.88	13	5	2	2	1	1	1	1	1	
0.89	13	5	2	2	1	1	1	1	1	
0.9	13	5	2	2	1	1	1	1	1	
0.91	13	5	2	2	1	1	1	1	1	
0.92	13	5	2	2	1	1	1	1	1	
0.93	13	5	2	2	1	1	1	1	1	
0.94	14	5	2	2	1	1	1	1	1	
0.95	14	5	2	2	1	1	1	1	1	
0.96	14	5	2	2	1	1	1	1	1	
0.97	14	5	2	2	1	1	1	1	1	
min	13	5	2	2	1	1	1	1	1	
prom	13.444444	5	2	2	1	1	1	1	1	
error					3.049382716					

Fuente: Elaboración propia

Tabla 3.21: Resultados del testeo del sistema para $P = 1$ simétrico con $S = 17$ y 18 .

α	BC=24 M=15 V=28 S=17 P=S1								
	R=20	R=15	R=30	R=35	R=40	R=45	R=50	R=55	R=60
0.8	20	7	3	2	1	1	1	1	1
0.81	20	7	3	2	1	1	1	1	1
0.82	20	7	3	2	1	1	1	1	1
0.83	20	7	3	2	1	1	1	1	1
0.84	20	7	3	2	1	1	1	1	1
0.85	20	7	3	2	1	1	1	1	1
0.86	20	7	3	2	1	1	1	1	1
0.87	20	7	3	2	1	1	1	1	1
0.88	20	7	3	2	1	1	1	1	1
0.89	20	7	3	2	1	1	1	1	1
0.9	20	7	3	2	1	1	1	1	1
0.91	20	7	3	2	1	1	1	1	1
0.92	20	7	3	2	1	1	1	1	1
0.93	20	7	3	2	1	1	1	1	1
0.94	20	7	3	2	1	1	1	1	1
0.95	20	7	3	2	1	1	1	1	1
0.96	20	7	3	2	1	1	1	1	1
0.97	20	8	3	2	1	1	1	1	1
min	20	7	3	2	1	1	1	1	1
prom	20	7.0555556	3	2	1	1	1	1	1
error					4.117283951				

Fuente: Elaboración propia

Como podemos ver en la Tabla 3.20 y 3.21 con $P = 1/2$ se obtuvo mejores resultados con $R > 40$.

3.3.7. Hallar T

En este séptimo testeo hallaremos T , parámetro usado por el algoritmo de *Eliminación de Segmentos Inútiles por Energía* empleado para la etapa de *Detección de Inicio y Fin de la Señal de Voz* en las señales de voz, teniendo la siguiente configuración del algoritmo:

- Detección de Inicio y Fin de la Señal de Voz (por función de energía)

$T = 80, 96, 112, 128, 144, 160, 170, 176, 192, 200, 208, 210, 220, 224, 230, 240, 256, 260, 270$ y 272

$E = 1$ y 3 número de tramas para el cálculo de los puntos de inicio y fin de la señal de voz

- Filtro Preénfasis

$$\alpha = 0.95$$

- Segmentación por Hamming

$$V = 28 ; \quad S = 17$$

- MFCC

$$BC = 24 ; \quad M = 15$$

- DTW

$$P = 1/2 \text{ Simétrico} ; \quad R = 40 ; \quad D = \text{Euclíadiana Cuadrática}$$

Tabla 3.22: Resultados del testeo del sistema para E = 1 y 3.

T	A=0.95 BC=24 M=15 V=28 P=S1 R=40	
	E=1	E=3
272	0.1423361234	0.131170041
270	0.142265312	0.169930906
260	0.14654467	0.138318809
256	0.206047218	0.089632901
240	0.180985061	0.099563841
230	0.13636777	0.123425669
224	0.162440198	0.122705063
0.87	0.1084665	0.177083403
220	0.152063116	0.143037404
210	0.130882351	0.138373347
208	0.14477218	0.129921146
200	0.163915064	0.142663281
192	0.136109187	0.12011364
176	0.177405319	0.163441463
160	0.183944807	0.161748511
144	0.182718486	0.196339446
128	0.206047218	0.175164863
112	0.196310078	0.143973071
96	0.182718486	0.182723452
80	0.180985061	0.177249872
min	0.1308823509	0.89632901
prom	0.1660451424	0.177796984
error	1	1

Fuente: Elaboración propia

Como podemos ver en la Tabla 3.22 con $E = 3$ se obtuvo mejores resultados con un error promedio de $prom = 0,14$ teniendo como error mínimo para $T = 256$, el segundo lugar para $T = 240$ y en tercer lugar para $T = 192$, pero vemos que el valor de error para $T = 192$ tanto en $E = 1$ como para $E = 3$ ocupa el tercer lugar siendo este el más estable por lo que se escogerá a este como mejor valor para la longitud de trama para la eliminación de segmentos inútiles por energía.

3.3.8. Hallar V y S

En este octavo testeo hallaremos V y S , parámetros usados para la etapa de segmentación o ventaneamiento de la señal de voz, teniendo la siguiente configuración del algoritmo:

- Detección de Inicio y Fin de la Señal de Voz (por función de energía)

$$T = 192 ; \quad E = 3$$

- Filtro Preénfasis

$$\alpha = \text{Desde } 0.80 \text{ hasta } 0.97$$

- Segmentación por Hamming

$$V = \text{Desde } 20 \text{ hasta } 31 ; \quad S = \text{Desde } 9 \text{ hasta } 21$$

- MFCC

$$BC = 24 ; \quad M = 15$$

- DTW

$$P = 1/2 \text{ Simétrico} ; \quad R = 40 ; \quad D = \text{Euclíadiana Cuadrática}$$

Tabla 3.23: Resultados del testeo del sistema para V = 20 y 21.

α	BC=24 M=15 V=20 T=192 P=S1/2 R=40						BC=24 M=15 V=21 T=192 P=S1/2 R=40					
	S=14	S=13	S=12	S=11	S=10	S=9	S=14	S=13	S=12	S=11	S=10	S=9
0.8	4	3	6	7	13	20	3	2	4	5	10	14
0.81	4	3	6	7	13	20	3	2	4	5	10	14
0.82	4	3	6	7	13	20	3	2	4	5	10	14
0.83	4	3	6	7	13	20	3	2	4	5	10	14
0.84	4	3	6	6	13	20	2	2	4	5	10	14
0.85	4	3	6	6	13	20	2	2	4	5	10	14
0.86	4	3	6	6	13	20	2	2	4	5	10	14
0.87	4	3	6	6	13	20	2	2	4	5	10	14
0.88	4	3	6	6	13	20	2	2	4	5	10	14
0.89	4	3	7	6	13	20	2	2	4	5	10	14
0.9	4	2	6	6	13	19	2	2	4	5	10	14
0.91	4	2	6	6	13	19	2	2	4	5	10	14
0.92	4	2	6	6	13	19	2	2	4	5	10	14
0.93	4	2	7	6	13	19	2	2	4	5	10	14
0.94	4	2	7	6	13	19	2	2	4	5	10	14
0.95	4	2	7	6	13	19	2	2	4	5	10	14
0.96	4	2	7	6	13	19	2	2	4	5	10	14
0.97	4	2	7	6	13	19	3	2	4	5	10	14
min	4	2	6	6	13	19	2	2	4	5	10	14
prom	4	2.5555556	6.3333333	6.2222222	13	19.611111	2.3333333	2	4	5	10	14
error												6.222222222

Fuente: Elaboración propia

Tabla 3.24: Resultados del testeo del sistema para V = 22 y 23.

α	BC=24 M=15 V=22 T=192 P=S1/2 R=40						BC=24 M=15 V=23 T=192 P=S1/2 R=40					
	S=15	S=14	S=13	S=12	S=11	S=10	S=15	S=14	S=13	S=12	S=11	S=10
0.8	4	4	2	4	5	9	3	3	3	4	5	9
0.81	4	4	2	4	5	9	3	3	3	4	5	9
0.82	4	4	2	4	5	9	3	3	3	4	5	9
0.83	4	4	2	4	5	9	3	3	3	4	5	9
0.84	4	4	2	4	5	9	3	3	3	4	5	9
0.85	4	4	2	4	5	9	3	3	3	4	5	9
0.86	4	4	2	4	5	9	3	3	3	4	5	9
0.87	3	4	2	4	5	9	3	3	3	4	5	9
0.88	3	4	2	4	5	9	3	3	3	4	5	9
0.89	3	4	2	4	5	9	3	3	3	4	5	9
0.9	3	4	2	4	5	9	3	3	3	2	4	5
0.91	3	4	2	4	5	9	3	3	2	4	5	9
0.92	3	4	2	4	5	9	2	3	2	4	5	9
0.93	3	3	2	4	5	9	2	3	2	4	5	9
0.94	3	3	2	4	5	9	3	3	2	4	5	9
0.95	3	3	2	4	5	9	3	3	2	4	5	9
0.96	3	3	2	4	5	9	3	3	2	4	5	9
0.97	3	3	2	4	5	9	3	3	2	4	5	10
min	3	3	2	4	5	9	2	3	2	4	5	9
prom	3.38888889	3.7222222	2	4	5	9	3.2777778	3	2.5555556	4	5	9.0555556
error												4.481481481

Fuente: Elaboración propia

Tabla 3.25: Resultados del testeo del sistema para V = 24 y 25.

α	BC=24 M=15 V=24 T=192 P=S1/2 R=40						BC=24 M=15 V=25 T=192 P=S1/2 R=40					
	S=16	S=15	S=14	S=13	S=12	S=11	S=17	S=16	S=15	S=14	S=13	S=12
0.8	2	2	3	2	2	5	1	2	3	3	2	2
0.81	2	3	3	2	2	5	1	2	2	3	2	2
0.82	2	3	3	2	2	5	1	2	2	3	2	2
0.83	2	3	3	2	2	5	1	2	2	3	2	2
0.84	2	3	3	2	2	5	1	2	2	3	2	2
0.85	2	3	3	2	2	5	1	2	2	3	2	2
0.86	2	3	3	2	2	5	1	2	2	3	2	2
0.87	2	3	3	2	2	5	1	2	2	3	2	2
0.88	1	3	3	2	2	5	1	2	2	3	2	2
0.89	1	3	3	2	2	5	1	2	2	2	2	2
0.9	1	3	3	2	2	5	1	2	2	2	2	2
0.91	1	2	3	2	2	5	1	2	2	2	2	2
0.92	1	2	3	2	2	5	1	2	2	2	2	2
0.93	1	2	3	2	2	5	1	2	2	2	2	2
0.94	1	2	2	2	2	5	1	2	2	2	2	2
0.95	1	2	2	2	2	5	1	2	2	2	2	2
0.96	1	2	2	2	2	5	1	2	2	2	2	2
0.97	1	2	2	2	2	5	1	2	2	2	2	2
min	1	2	2	2	2	5	1	2	2	2	2	2
prom	1.4444444	2.5555556	2.7777778	2	2	5	1	2	2.0555556	2.5	2	2
error				2.62962963					1.925925926			

Fuente: Elaboración propia

Tabla 3.26: Resultados del testeo del sistema para V = 26 y 27.

α	BC=24 M=15 V=26 T=192 P=S1/2 R=40						BC=24 M=15 V=27 T=192 P=S1/2 R=40					
	S=17	S=16	S=15	S=14	S=13	S=12	S=18	S=17	S=16	S=15	S=14	S=13
0.8	1	2	2	3	3	2	2	1	1	2	2	2
0.81	1	2	2	3	3	2	2	1	1	2	2	2
0.82	1	2	2	3	3	2	2	1	1	2	2	2
0.83	1	2	2	3	3	2	2	1	1	2	2	2
0.84	1	2	2	3	3	2	2	1	1	2	2	2
0.85	1	2	2	3	3	2	2	1	1	2	2	2
0.86	1	2	2	3	3	2	2	1	1	2	2	2
0.87	1	2	2	3	3	2	2	1	1	2	2	2
0.88	1	2	2	3	3	2	2	1	1	2	2	2
0.89	1	2	2	3	3	2	2	1	1	2	2	2
0.9	1	2	2	3	3	2	2	1	1	2	2	2
0.91	1	2	2	3	3	2	2	1	1	2	2	2
0.92	1	2	2	2	3	2	2	1	1	2	2	2
0.93	1	2	2	2	3	2	2	1	1	2	2	2
0.94	1	2	2	2	3	2	2	1	1	2	2	2
0.95	1	2	2	2	3	2	2	1	1	2	2	3
0.96	1	2	2	2	3	2	2	1	1	2	2	3
0.97	1	1	2	3	3	2	2	1	1	2	2	3
min	1	1	2	2	3	2	2	1	1	2	2	2
prom	1	1.9444444	2	2.6666667	3	2	2	1	1	2	2	2.1666667
error				2.101851852					1.694444444			

Fuente: Elaboración propia

Tabla 3.27: Resultados del testeo del sistema para V = 28 y 29.

α	BC=24 M=15 V=28 T=192 P=S1/2 R=40						BC=24 M=15 V=29 T=192 P=S1/2 R=40					
	S=19	S=18	S=17	S=16	S=15	S=14	S=19	S=18	S=17	S=16	S=15	S=14
0.8	1	1	1	1	2	2	1	1	1	1	2	2
0.81	1	1	1	1	2	2	1	1	1	1	2	2
0.82	1	1	1	1	2	2	1	1	1	1	2	2
0.83	1	1	1	1	2	2	1	1	1	1	2	2
0.84	1	1	1	1	2	2	1	1	1	1	2	2
0.85	1	1	1	1	2	2	1	1	1	1	2	2
0.86	1	1	1	1	2	2	1	1	1	1	2	2
0.87	1	1	1	1	2	2	1	1	1	1	2	2
0.88	1	1	1	1	2	2	1	1	1	1	2	2
0.89	1	1	1	1	2	2	1	1	1	1	2	2
0.9	1	1	1	1	2	2	1	1	1	1	2	2
0.91	1	1	1	1	2	2	1	1	1	1	2	2
0.92	1	1	1	1	2	2	1	1	1	1	2	2
0.93	1	1	1	1	2	2	1	1	1	1	2	2
0.94	1	1	1	1	2	2	1	1	1	1	2	2
0.95	1	1	1	1	2	2	1	1	1	1	2	2
0.96	1	1	1	1	2	2	1	2	2	1	2	2
0.97	1	1	1	1	2	2	2	2	2	1	2	2
min	1	1	1	1	2	2	1	1	1	1	2	2
prom	1.0555556	1	1	1	2	2	1.0555556	1.1111111	1.1111111	1	2	2
error				1.342592593				1.37962963				

Fuente: Elaboración propia

Tabla 3.28: Resultados del testeo del sistema para V = 30 y 31.

α	BC=24 M=15 V=30 T=192 P=S1/2 R=40						BC=24 M=15 V=31 T=192 P=S1/2 R=40					
	S=20	S=19	S=18	S=17	S=16	S=15	S=20	S=19	S=18	S=17	S=16	S=15
0.8	2	1	1	1	1	3	2	2	1	1	1	3
0.81	2	1	1	1	1	3	2	2	1	1	1	3
0.82	2	1	2	1	1	2	2	2	1	1	1	2
0.83	2	1	2	1	1	2	2	2	1	1	1	2
0.84	2	1	2	1	1	2	2	2	1	1	1	2
0.85	2	1	2	1	1	2	2	2	1	1	1	2
0.86	2	1	2	1	1	2	2	2	1	1	1	2
0.87	2	1	2	1	1	2	2	2	1	1	1	2
0.88	2	1	2	1	1	2	2	2	1	1	1	2
0.89	2	1	2	1	1	2	2	2	1	1	1	2
0.9	2	1	2	1	1	2	2	2	1	1	1	2
0.91	2	1	2	1	1	2	2	2	1	1	1	2
0.92	2	2	2	1	1	2	2	2	1	2	1	2
0.93	2	2	2	1	1	2	2	2	1	2	1	2
0.94	2	2	2	1	1	2	2	2	1	2	1	2
0.95	2	2	2	2	1	2	2	2	2	2	1	2
0.96	2	2	2	2	1	2	2	2	2	2	1	2
0.97	2	2	2	2	1	2	2	2	2	2	1	2
min	2	1	1	1	1	2	2	2	1	1	1	2
prom	2	1.3333333	1.8888889	1.1666667	1	2.1111111	2	2	1.1666667	1.3333333	1	2.1111111
error				1.583333333					1.601851852			

Fuente: Elaboración propia

Tabla 3.29: Resultados del testeo del sistema para $V = 32$.

α	BC=24 M=15 V=32 T=192 P=S1/2 R=40					
	S=21	S=20	S=19	S=18	S=17	S=16
0.8	2	2	1	1	1	1
0.81	2	2	1	1	1	1
0.82	2	2	1	1	1	1
0.83	2	2	1	1	1	1
0.84	2	2	1	1	1	1
0.85	2	2	1	1	1	1
0.86	2	2	1	1	1	1
0.87	2	2	1	1	1	1
0.88	2	2	1	1	1	1
0.89	2	2	1	1	1	1
0.9	2	2	1	1	1	1
0.91	2	2	1	1	1	1
0.92	2	2	1	1	1	1
0.93	2	2	1	1	1	1
0.94	2	2	1	1	2	1
0.95	2	2	1	1	2	1
0.96	2	2	1	2	2	1
0.97	2	3	1	2	2	1
min	2	2	1	1	1	1
prom	2	2.0555556	1	1.1111111	1.2222222	1
error				1.398148148		

Fuente: Elaboración propia

Como podemos ver en las Tabla 3.23, 3.24, 3.25, 3.26, 3.27, 3.28 y 3.29 que con $V = 28$ se obtuvo mejores resultados con un error promedio de $error = 1,34$, el segundo lugar para $V = 29$ con un error promedio de $error = 1,37$, el tercer lugar para $V = 32$ con un error promedio de $error = 1,39$ y el cuarto lugar para $V = 30$ con un $error = 1,58$.

3.3.9. Hallar V , S , y T

En este noveno testeo hallaremos los valores finales para V y S , parámetros usados para la etapa de segmentación o ventaneamiento de la señal de voz y T , parámetro usado por el algoritmo de *Eliminación de Segmentos Inútiles por Energía* empleado para la etapa de *Detección de Inicio y Fin de la Señal de Voz*, teniendo la siguiente configuración del algoritmo:

- Detección de Inicio y Fin de la Señal de Voz (por función energía)

$$T = 144, 160 \text{ y } 192 ; \quad E = 3$$

- Filtro Preénfasis

$$\alpha = \text{Desde } 0.80 \text{ hasta } 0.97$$

- Segmentación por Hamming

$$V = 27, 28 \text{ y } 30 ; \quad S = 13, 14, 15, 16, 17, 18, 19 \text{ y } 20$$

- MFCC

$$BC = 24 ; \quad M = 15$$

- DTW

$$P = 1/2 \text{ Simétrico} ; \quad R = 40 ; \quad D = \text{Euclíadiana Cuadrática}$$

Tabla 3.30: Resultados del testeo del sistema para $V = 27$ con $T = 192$ y 160 .

α	BC=24 M=15 V=27 T=192 P=S1/2 R=40						BC=24 M=15 V=27 T=160 P=S1/2 R=40					
	S=18	S=17	S=16	S=15	S=14	S=13	S=18	S=17	S=16	S=15	S=14	S=13
0.8	2	1	1	2	2	2	2	1	1	2	2	2
0.81	2	1	1	2	2	2	2	1	1	2	2	2
0.82	2	1	1	2	2	2	2	1	1	2	2	2
0.83	2	1	1	2	2	2	2	1	1	2	2	2
0.84	2	1	1	2	2	2	2	1	1	2	2	2
0.85	2	1	1	2	2	2	2	1	1	2	2	2
0.86	2	1	1	2	2	2	2	1	1	2	2	2
0.87	2	1	1	2	2	2	2	1	1	2	2	2
0.88	2	1	1	2	2	2	2	1	1	2	2	2
0.89	2	1	1	2	2	2	2	1	1	2	2	2
0.9	2	1	1	2	2	2	2	1	1	2	2	2
0.91	2	1	1	2	2	2	2	1	1	2	2	2
0.92	2	1	1	2	2	2	2	1	1	2	2	2
0.93	2	1	1	2	2	2	2	1	1	2	2	2
0.94	2	1	1	2	2	2	2	1	1	2	2	2
0.95	2	1	1	2	2	3	2	1	1	2	2	2
0.96	2	1	1	2	2	3	2	1	1	2	2	2
0.97	2	1	1	2	2	3	2	1	1	2	2	2
min	2	1	1	2	2	2	2	1	1	2	2	2
prom	2	1	1	2	2	2.1666667	2	1	1	2	2	2
error						1.69444444						1.66666667

Fuente: Elaboración propia

Tabla 3.31: Resultados del testeo del sistema para $V = 27$ con $T = 144$.

α	BC=24 M=15 V=27 T=144 P=S1/2 R=40					
	S=18	S=17	S=16	S=15	S=14	S=13
0.8	1	1	1	2	2	2
0.81	1	1	1	2	2	2
0.82	1	1	1	2	2	2
0.83	1	1	1	2	2	2
0.84	2	1	1	2	2	2
0.85	2	1	1	2	2	2
0.86	2	1	1	2	2	2
0.87	2	1	1	2	2	2
0.88	2	1	1	2	2	2
0.89	2	1	1	2	2	2
0.9	2	1	1	2	2	2
0.91	2	1	1	2	2	2
0.92	2	1	1	2	2	2
0.93	2	1	1	2	2	2
0.94	2	1	1	2	2	2
0.95	2	1	1	2	2	2
0.96	2	1	1	2	2	2
0.97	2	1	1	2	2	2
min	1	1	1	2	2	2
prom	1.7777778	1	1	2	2	2
error				1.62962963		

Fuente: Elaboración propia

Tabla 3.32: Resultados del testeo del sistema para $V = 28$ con $T = 192$ y 160 .

α	BC=24 M=15 V=27 T=192 P=S1/2 R=40						BC=24 M=15 V=27 T=160 P=S1/2 R=40					
	S=18	S=17	S=16	S=15	S=14	S=13	S=18	S=17	S=16	S=15	S=14	S=13
0.8	1	1	1	1	2	2	1	2	1	1	3	2
0.81	1	1	1	1	2	2	1	2	1	1	3	2
0.82	1	1	1	1	2	2	1	2	1	1	3	2
0.83	1	1	1	1	2	2	1	2	1	1	3	2
0.84	1	1	1	1	2	2	1	2	1	1	3	2
0.85	1	1	1	1	2	2	1	2	1	1	3	2
0.86	1	1	1	1	2	2	1	2	1	1	3	2
0.87	1	1	1	1	2	2	1	2	1	1	3	2
0.88	1	1	1	1	2	2	1	2	1	1	3	2
0.89	1	1	1	1	2	2	1	2	1	1	3	2
0.9	1	1	1	1	2	2	1	2	1	1	2	2
0.91	1	1	1	1	2	2	1	2	1	1	2	2
0.92	1	1	1	1	2	2	1	2	1	1	2	2
0.93	1	1	1	1	2	2	1	2	1	1	2	2
0.94	1	1	1	1	2	2	1	2	1	1	2	2
0.95	1	1	1	1	2	2	1	2	1	1	2	2
0.96	1	1	1	1	2	2	1	2	1	1	2	2
0.97	2	1	1	1	2	2	1	2	1	1	2	2
min	1	1	1	1	2	2	1	2	1	1	2	2
prom	1.0555556	1	1	1	2	2	1	2	1	1	2.5555556	2
error				1.342592593				1.592592593				

Fuente: Elaboración propia

Tabla 3.33: Resultados del testeo del sistema para $V = 28$ con $T = 144$.

α	BC=24 M=15 V=27 T=144 P=S1/2 R=40					
	S=18	S=17	S=16	S=15	S=14	S=13
0.8	1	1	1	1	2	2
0.81	1	1	1	1	2	2
0.82	1	1	1	1	2	2
0.83	1	1	1	1	2	2
0.84	1	1	1	1	2	2
0.85	1	1	1	1	2	2
0.86	1	1	1	1	2	2
0.87	1	1	1	1	2	2
0.88	1	1	1	1	2	2
0.89	1	1	1	1	2	2
0.9	1	1	1	1	2	2
0.91	1	1	1	1	2	2
0.92	1	1	1	1	2	2
0.93	1	1	1	1	2	2
0.94	1	1	1	1	2	2
0.95	2	1	1	1	2	2
0.96	2	1	1	1	2	2
0.97	2	1	1	1	2	2
min	1	1	1	1	2	2
prom	1.16666667	1	1	1	2	2
error				1.361111111		

Fuente: Elaboración propia

Tabla 3.34: Resultados del testeo del sistema para $V = 30$ con $T = 192$ y 160 .

α	BC=24 M=15 V=30 T=192 P=S1/2 R=40						BC=24 M=15 V=30 T=160 P=S1/2 R=40					
	S=20	S=19	S=18	S=17	S=16	S=15	S=20	S=19	S=18	S=17	S=16	S=15
0.8	2	1	1	1	1	3	2	1	1	1	1	3
0.81	2	1	1	1	1	3	2	1	1	1	1	3
0.82	2	1	2	1	1	2	2	1	1	1	1	3
0.83	2	1	2	1	1	2	2	1	1	1	1	3
0.84	2	1	2	1	1	2	2	1	1	1	1	3
0.85	2	1	2	1	1	2	2	1	1	1	1	3
0.86	2	1	2	1	1	2	2	1	1	1	1	3
0.87	2	1	2	1	1	2	2	1	1	1	1	3
0.88	2	1	2	1	1	2	2	1	1	1	1	3
0.89	2	1	2	1	1	2	2	1	1	1	1	3
0.9	2	1	2	1	1	2	2	1	1	1	1	3
0.91	2	1	2	1	1	2	2	1	1	1	1	3
0.92	2	2	2	1	1	2	2	1	1	1	1	3
0.93	2	2	2	1	1	2	2	1	1	1	1	3
0.94	2	2	2	1	1	2	2	1	1	1	1	3
0.95	2	2	2	2	1	2	2	1	1	1	1	3
0.96	2	2	2	2	1	2	2	1	3	1	1	3
0.97	2	2	2	2	1	2	2	2	3	2	1	2
min	2	1	1	1	1	2	2	1	1	1	1	2
prom	2	1.33333333	1.8888889	1.1666667	1	2.1111111	2	1.0555556	1.2222222	1.0555556	1	2.9444444
error				1.583333333								1.546296296

Fuente: Elaboración propia

Tabla 3.35: Resultados del testeo del sistema para $V = 30$ con $T = 144$.

α	BC=24 M=15 V=30 T=144 P=S1/2 R=40					
	S=20	S=19	S=18	S=17	S=16	S=15
0.8	1	1	1	1	1	2
0.81	1	1	1	1	1	2
0.82	1	1	1	1	1	2
0.83	1	1	1	1	1	2
0.84	1	1	1	1	1	2
0.85	1	1	1	1	1	2
0.86	1	1	1	1	1	2
0.87	1	2	1	1	1	2
0.88	1	2	1	1	1	2
0.89	1	2	1	1	1	2
0.9	1	2	1	1	1	2
0.91	1	2	1	1	1	2
0.92	1	2	1	1	1	2
0.93	1	2	1	1	1	2
0.94	1	2	2	1	1	2
0.95	1	2	2	1	1	2
0.96	1	2	2	1	1	2
0.97	1	2	2	1	1	2
min	1	1	1	1	1	2
prom	1	1.6111111	1.2222222	1	1	2
error			1.305555556			

Fuente: Elaboración propia

Como podemos ver en las Tabla 3.30, 3.31, 3.32, 3.33, 3.34 y 3.35 podemos concluir que con $V = 28$ se obtuvo mejores resultados donde con $T = 192$ se obtuvo el mejor resultado con un error promedio de $error = 1,34$, sin embargo, con $V = 30$ y $T = 144$ se obtuvo un mejor resultado con un error promedio de $error = 1,30$, pero se escogerá $V = 28$ al tener un error promedio global menor que el de $V = 30$.

3.3.10. Hallar d

En este décimo testeo hallaremos d , el tipo de medida de distancia entre dos patrones de señales de voz, usado por el algoritmo DTW en la etapa de *Reconocimiento de Voz*, teniendo la siguiente configuración del algoritmo:

- Detección de Inicio y Fin de la Señal de Voz (por función de energía)

$$T = 192 ; \quad E = 3$$

- Filtro Preénfasis

$$\alpha = \text{Desde } 0.80 \text{ hasta } 0.97$$

- Segmentación por Hamming

$$V = 28 ; \quad S = 18$$

- MFCC

$$BC = 24 ; \quad M = 15$$

- DTW

$$P = 1/2 \text{ Simétrico} ; \quad R = 40 ; \quad D = \text{Euclíadiana Cuadrática y Error Cuadrático Medio}$$

Tabla 3.36: Resultados del testeo del sistema para distancia euclíadiana cuadrática y error cuadrático medio.

	BC=24 M=15 V=28 P=S18 T=192 P=S1/2 R=40	
α	Euclíadiana Cuadrática	Error Cuadrático Medio
0.8	0.335508635	0.022367242
0.81	0.328620417	0.021908028
0.82	0.321050444	0.021403363
0.83	0.312792778	0.020852852
0.84	0.303845430	0.020256362
0.85	0.294211547	0.019614103
0.86	0.283902117	0.018926808
0.87	0.272940625	0.018196042
0.88	0.261370472	0.017424698
0.89	0.249266546	0.016617770
0.9	0.236753339	0.015783556
0.91	0.224033462	0.014935564
0.92	0.211432457	0.014095497
0.93	0.199467598	0.01329784
0.94	0.188945966	0.012596398
0.95	0.181075980	0.012071732
0.96	0.177493532	0.011832902
0.97	0.179872047	0.011991470
min	0.177493532	0.011832902
prom	0.253476855	0.016898457
error	1	1

Fuente: Elaboración propia

Como podemos ver en la Tabla 3.36 con la Distancia de Error Cuadrático Medio se obtuvo mejores resultados con un error promedio de $prom = 0,016$ teniendo un error mínimo $min = 0,011$ con $\alpha = 0,96$, ademas vemos que conforme este valor va aumentando se obtiene mejores resultados verificando así el valor propuesto en la teoría por la Ecuación (2.33), pero vemos que cuando $\alpha = 0,97$ aumenta el error por lo que se escogerá $\alpha = 0,95$, valor recomendado en la teoría.

3.3.11. Hallar el algoritmo para la detección de inicio y fin de la señal de voz

Ahora pasaremos a hallar el mejor algoritmo para la etapa de Detección de Inicio y Fin de la Señal de Voz, teniendo la siguiente configuración del algoritmo:

- Eliminación de Segmentos Inútiles por Energía

$$T = 192 ; \quad E = 3$$

- Algoritmo de Rabiner & Sambur de Tipo 1

$$T = 128$$

- S - Algoritmo de Rabiner & Sambur de Tipo 2

$$T = 128$$

Las grabaciones se realizaron con una frecuencia de muestreo 16000 Hz, canal mono, 16 bits de cuantificación y la codificación Little Endian. Tanto en el `audio_1.wav` y `audio_2.wav` se pronunció la palabra manzana, la diferencia es que para el `audio_1.wav` se controló el ruido mientras para el `audio_2.wav` no.

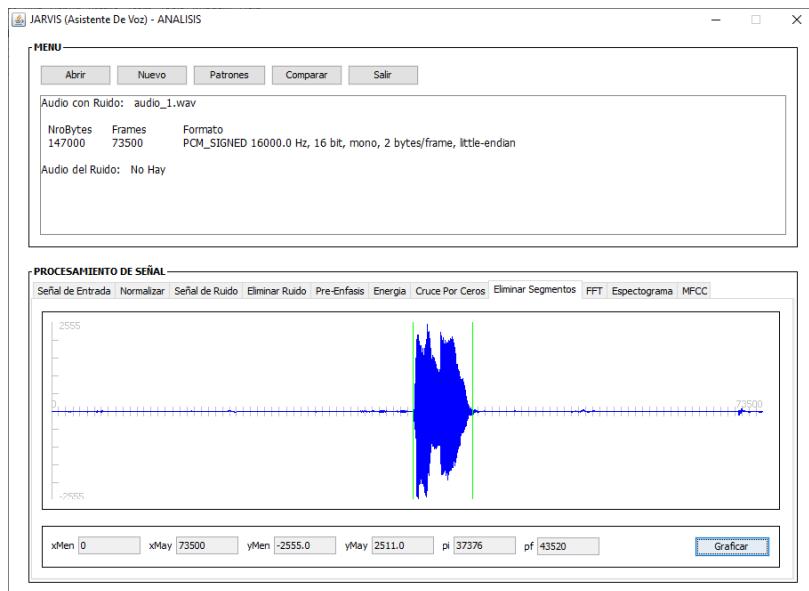


Figura 3.71: Resultado del testeo para el `audio_1.wav` por algoritmo de función de energía.
Fuente: Elaboración propia

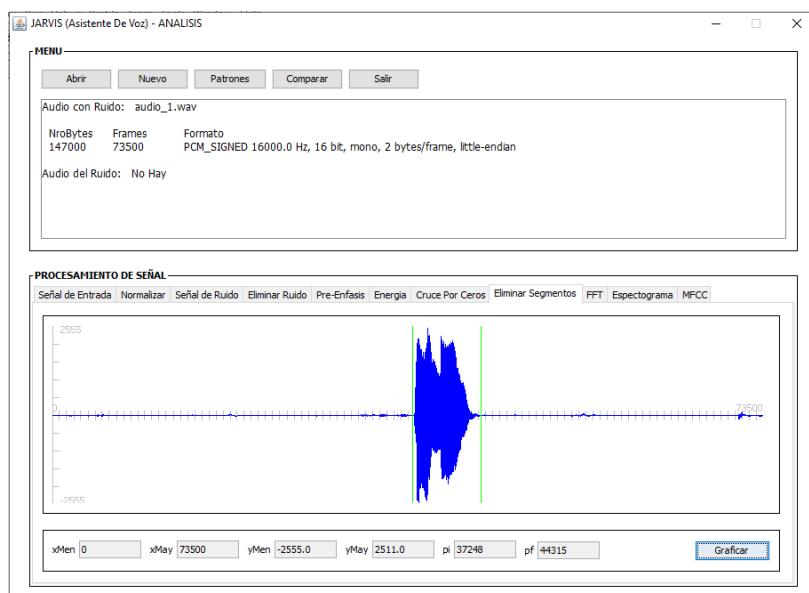


Figura 3.72: Resultado del testeo para el `audio_1.wav` por algoritmo de Rabiner & Sambur tipo 1.
Fuente: Elaboración propia

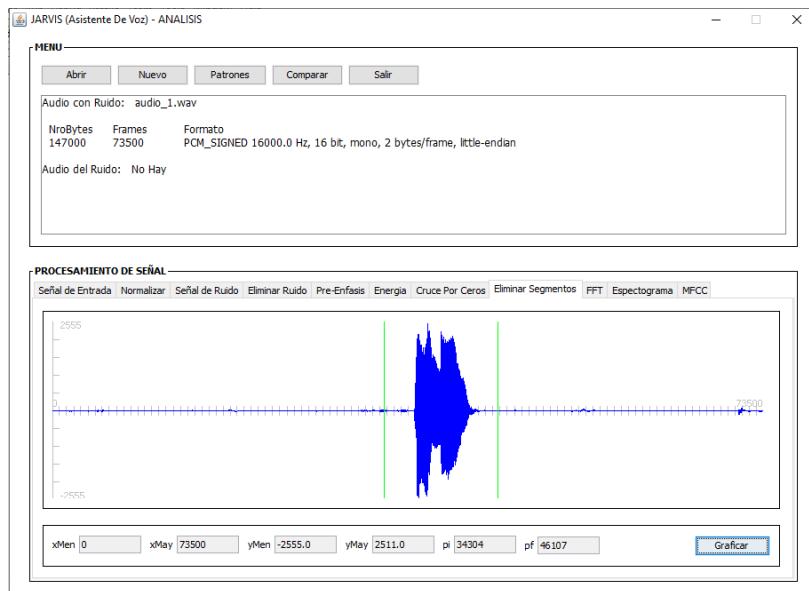


Figura 3.73: Resultado del testeо para el `audio_1.wav` por algoritmo de Rabiner & Sambur tipo 2.
Fuente: Elaboración propia

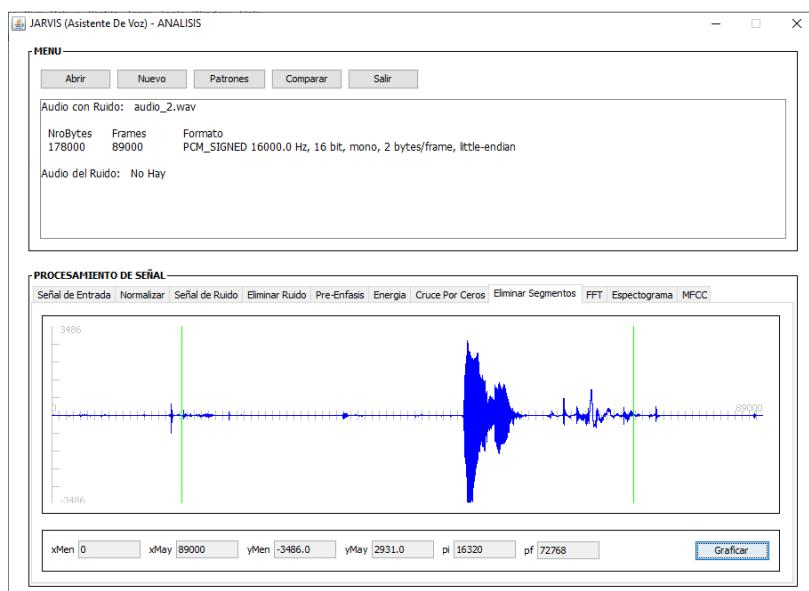


Figura 3.74: Resultado del testeо para el `audio_2.wav` por algoritmo de función de energía.
Fuente: Elaboración propia

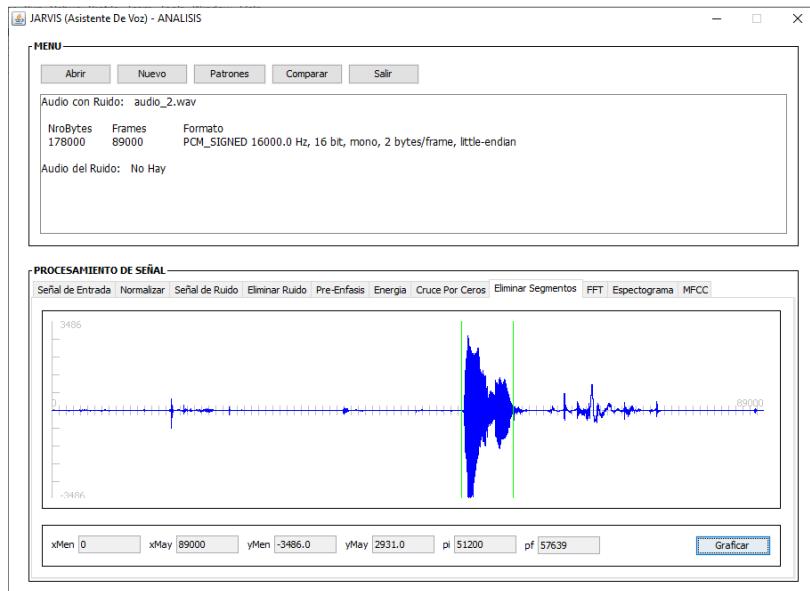


Figura 3.75: Resultado del testeо para el `audio_2.wav` por algoritmo de Rabiner & Sambur tipo 1.
Fuente: Elaboracióн propia

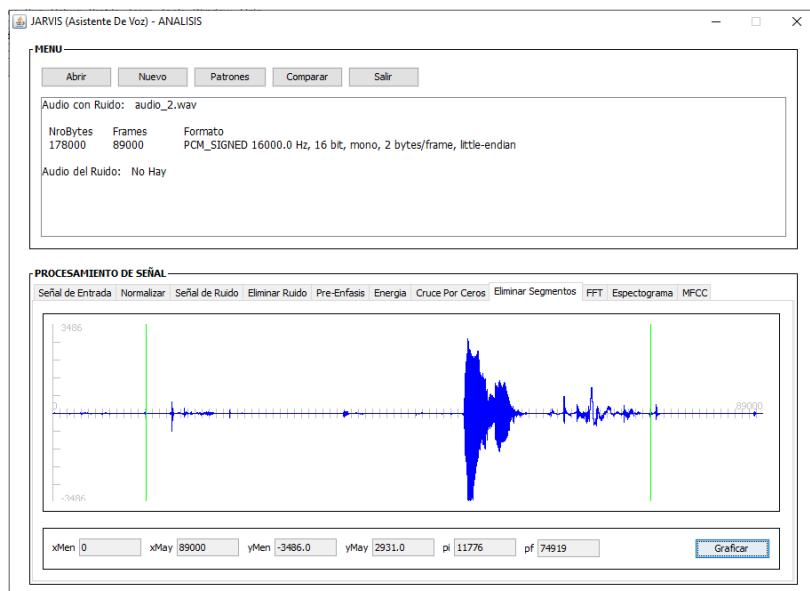


Figura 3.76: Resultado del testeо para el `audio_2.wav` por algoritmo de Rabiner & Sambur tipo 2.
Fuente: Elaboracióн propia

Para el `audio_1.wav` podemos observar en las Figura 3.71, 3.72 y 3.73 que el algoritmo por función de Energía y el de Rabiner & Sambur Tipo 1 fueron los que obtuvieron mejores resultados, pero en el `audio_2.wav` en las Figuras 3.74, 3.75 y 3.76 vemos que el algoritmo de Rabiner & Sambur Tipo 1 es superior al algoritmo por función de Energía. Por lo tanto, se escogerá el algoritmo Rabiner & Sambur Tipo 1 con $T = 128$ para la etapa de *Detección de Inicio y Fin de la Señal de Voz*.

3.3.12. Hallar el algoritmo para la eliminación de ruido aditivo

Ahora pasaremos a hallar el mejor algoritmo para la etapa de *Eliminación de Ruido*, teniendo como configuración del algoritmo de reconocimiento de voz para el sistema:

- Algoritmo LMS

$$M = 128 ; \quad \mu = 0.02 \text{ y } 0.002$$

- Algoritmo NLMS

$$M = 128 ; \quad \beta = 0.25, 0.025 \text{ y } 0.0025 ; \quad c = 0.0001$$

Estas pruebas se realizaron con un micrófono primario ubicado a 2 metros de la fuente de ruido (en este caso la melodía de un celular) y un segundo micrófono ubicado a 50 cm de la fuente de ruido. Los micrófonos que se usaron fueron de 2 laptops y la palabra que se pronuncio fue *uno*.

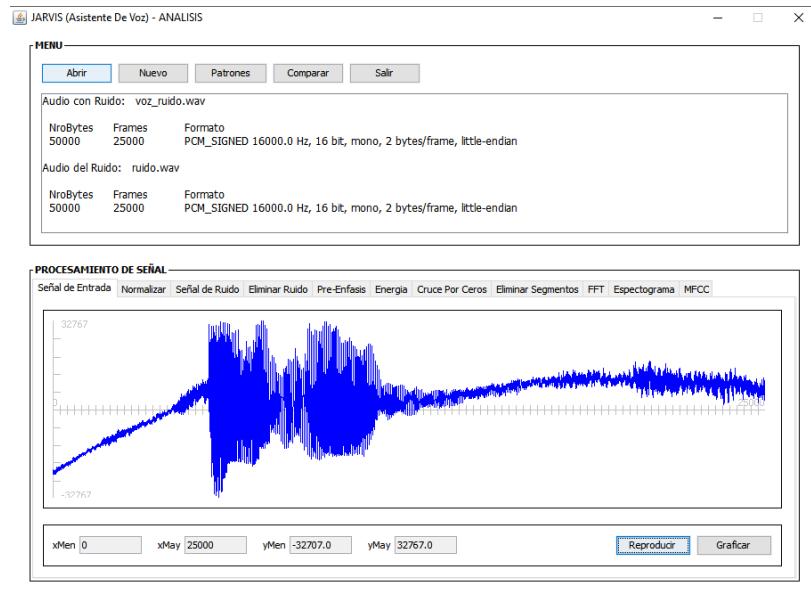


Figura 3.77: Grafica de la señal de voz contaminada voz_ruido.wav.
Fuente: Elaboración propia

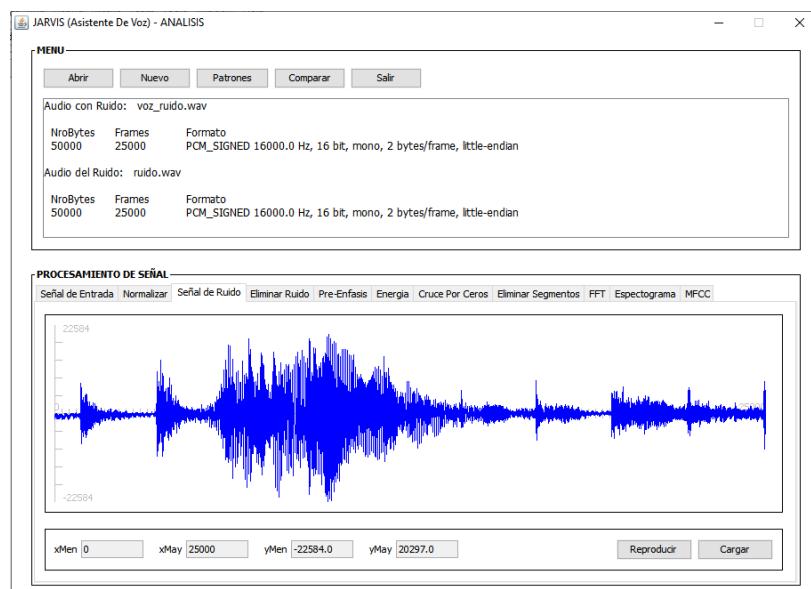


Figura 3.78: Grafica de la señal de ruido ruido.wav.
Fuente: Elaboración propia

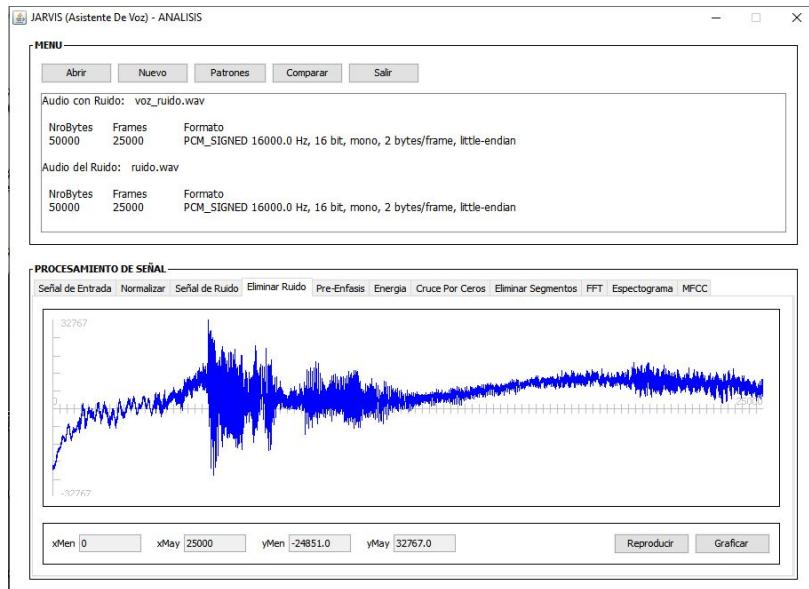


Figura 3.79: Grafica de la señal después de aplicar LMS con $\mu = 0.02$.
 Fuente: Elaboración propia

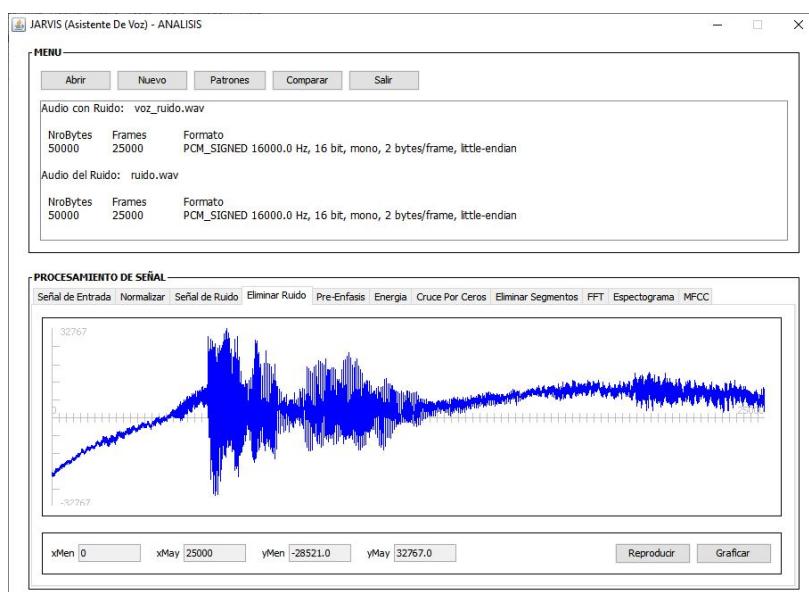


Figura 3.80: Grafica de la señal después de aplicar LMS con $\mu = 0.002$.
 Fuente: Elaboración propia

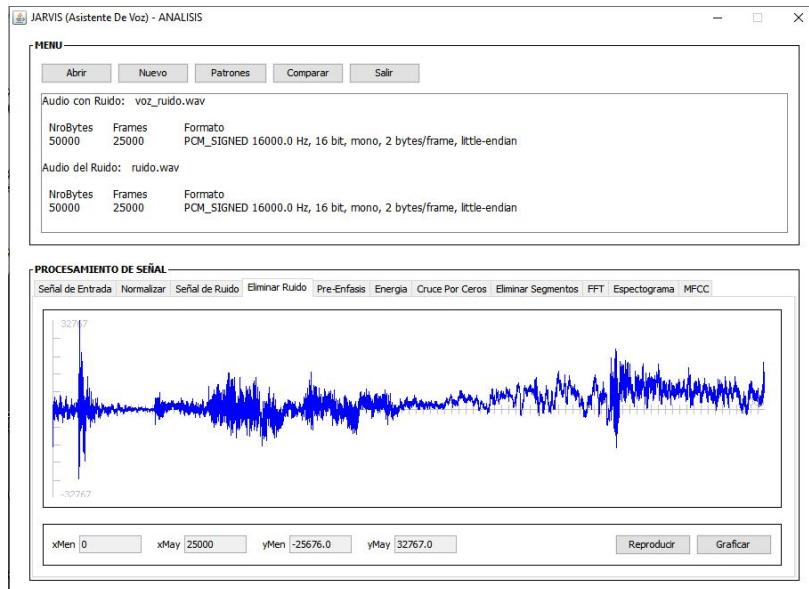


Figura 3.81: Grafica de la señal después de aplicar NLMS con $\beta = 0.25$.
Fuente: Elaboración propia

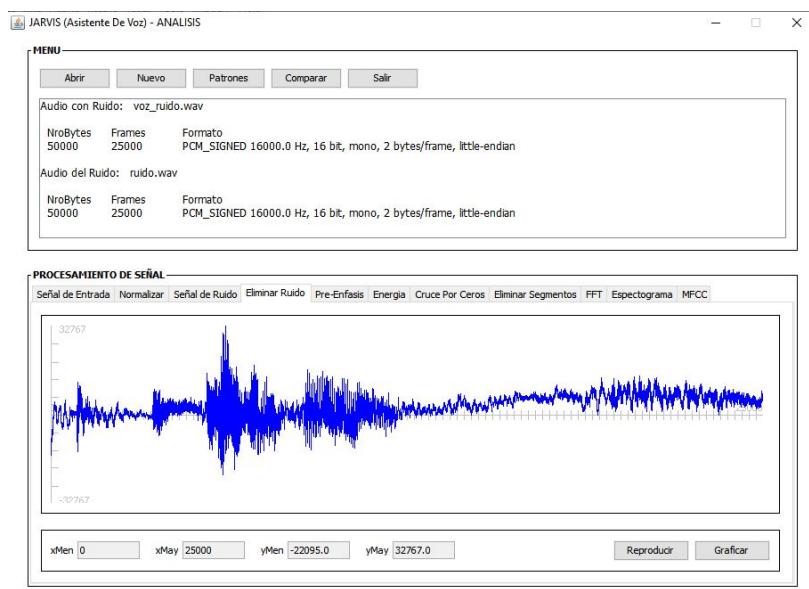


Figura 3.82: Grafica de la señal después de aplicar NLMS con $\beta = 0.025$.
Fuente: Elaboración propia

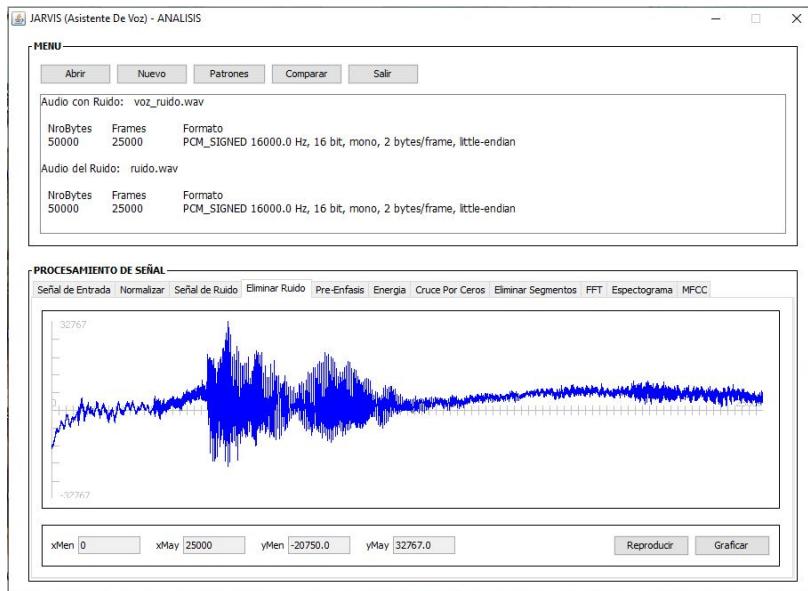


Figura 3.83: Grafica de la señal después de aplicar NLMS con $\beta = 0.0025$.
Fuente: Elaboración propia

Después de lo visto en las Figuras 3.79, 3.80, 3.81, 3.82 y 3.83 podemos concluir que el mejor resultado para la eliminación de ruido lo obtuvo el NLMS con $\beta = 0,0025$, en la Figura 3.83 podemos observar que el ruido se encuentra en una amplitud baja con respecto a la de voz, por lo que resultará un mejor caso para la etapa de *Detección de Inicio y Fin de la Señal de Voz*, ver la Figura 3.84.

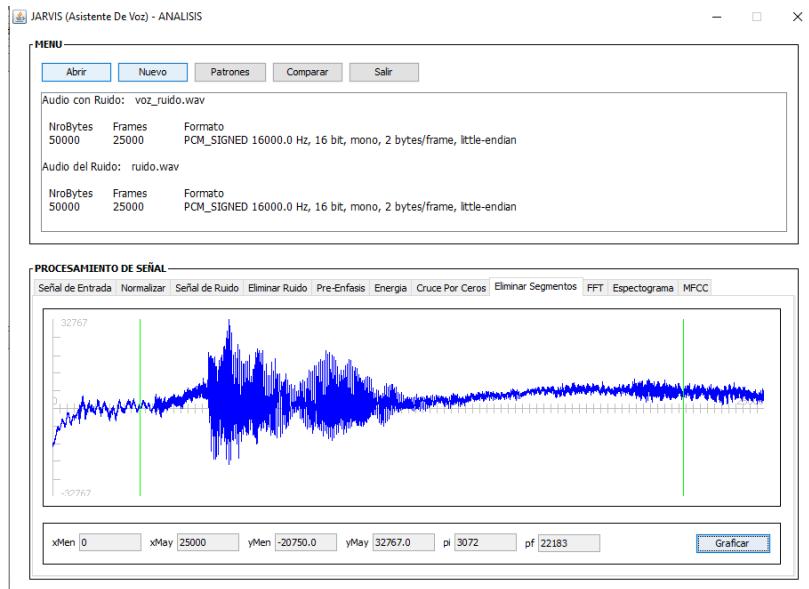


Figura 3.84: Grafica de la señal para el algoritmo NLMS con $\beta=0.0025$ y Rabiner & Sambur Tipo 1 con $T = 128$.

Fuente: Elaboración propia

Como podemos observar en la Figura 3.84 la señal no fue recortada perfectamente, el punto de fin de recorte está muy lejos del final de la palabra esto es porque en la señal todavía se tiene una energía elevada al final, esto se debe a que el sonido de la fuente de ruido tuvo mucho volumen, paso el rango de los 81 dBC que es el valor máximo para que funcione correctamente el algoritmo.

Finalmente, podemos concluir de esta etapa de testeo del sistema que la configuración que tendrá el algoritmo de reconocimiento de voz para el sistema, es el siguiente:

- Etapa de Eliminación de Ruido:

- Algoritmo NLMS

$$M = 128 ; \quad \beta = 0.0025 ; \quad c = 0.0001$$

- Etapa de Detección de Inicio y Fin de la Señal de Voz:

- Algoritmo Rabiner & Sambur Tipo 1

$$T = 128$$

- Etapa de Entrenamiento:

- Filtro Preénfasis

$$\alpha = 0.95$$

- Segmentación por Hamming

$$V = 28 ; \quad S = 18$$

- Algoritmo MFCC

$$BC = 24 ; \quad M = 15$$

- Etapa de Reconocimiento:

- Algoritmo DTW Simétrico

$$P = 1/2 ; \quad R = 40 ; \quad D = \text{Error Cuadrático Medio}$$

3.4. Fase 4. Mantenimiento del sistema

Lo que haremos en esta etapa será implementar la configuración (funciones y parámetros óptimos) del algoritmo de reconocimiento de voz (obtenida en la etapa de Testeo del Sistema) en la aplicación para el dispositivo móvil, además añadiremos a este el algoritmo para la toma de decisión, para así poder probar nuestro sistema en un ambiente real.

En la Figura 3.85 se muestra el código fuente del algoritmo para obtener el patrón característico (coeficientes MFCC) de una señal, usado en la etapa de entrenamiento.

```

public static ArchivoMFCC crearArchivoMFCC(Audio audioVoz, Audio audioRuido, String URI) {
    double men, may, max;
    double[] senalFinal = audioVoz.getDatos().getAmplitudes();

    if (audioRuido != null) {
        double[] senalEntrada = Preprocesamiento.normalizar(senalFinal, max: 32768);
        double[] senalRuido = Preprocesamiento.normalizar(audioRuido.getDatos().getAmplitudes(), max: 32768);
        LMS lms = new LMS(senalEntrada, senalRuido);
        lms.NLMS( beta: 0.0025, nroD: 128, a0: null);
        senalFinal = lms.getSenalDeseada();
        men = Utilitarios.getMenor(senalFinal);
        may = Utilitarios.getMayor(senalFinal);
        max = Utilitarios.getMaximo(may, men);
        senalFinal = Preprocesamiento.normalizarInt(senalFinal, -max, max, m: -32768, n: 32767);
    }

    senalFinal = Preprocesamiento.preEnfasis(senalFinal, alfa: 0.95);
    men = Utilitarios.getMenor(senalFinal);
    may = Utilitarios.getMayor(senalFinal);
    max = Utilitarios.getMaximo(may, men);
    if (max < 32768) {
        max = 32768;
    }
    senalFinal = Preprocesamiento.normalizarDouble(senalFinal, -max, max, m: -1, n: 1);

    int[] puntos = Preprocesamiento.eliminarSegmentosInutiles_RabinerSambur( tamTrama: 128, senalFinal, tipo: 1);
    senalFinal = Utilitarios.recortar(senalFinal, puntos[0], puntos[1]);

    Ventanamiento ventanamiento = new Ventanamiento(senalFinal, TV: 448, DS: 288, tipo: 1);

    double fs = audioVoz.getFormato().getSampleRate();
    MFCC mfcc = new MFCC(ventanamiento.getMatriz(), dimensionVentana: 448, M: 15, BC: 24, fs);
    double[][] moMFCC = mfcc.getResultMFCC( fMin: 0, fMax: fs/2);

    String ruta = audioVoz.getContexto().getFilesDir().getPath();
    String usuario = "", palabra = "";
    if (URI.contains("/")) {
        ruta += "/MFCC/Comandos/" + URI;
        usuario = URI.split(regex: "/") [0];
        palabra = URI.split(regex: "/") [1];
    } else if (!URI.equals("")) {
        ruta += "/MFCC/Patrones/" + URI;
        usuario = URI;
        palabra = "Clave";
    }
    ruta += "/" + audioVoz.getArchivo().getName().split(regex: ".wav") [0] + ".mfcc";

    ArchivoMFCC archivoMFCC = new ArchivoMFCC(ruta, usuario, palabra, umbral: 0, moMFCC);
    archivoMFCC.guardarMO();

    return archivoMFCC;
}

```

Figura 3.85: Código fuente del algoritmo de reconocimiento de voz para la etapa de entrenamiento.

Fuente: Elaboración propia

En la Figura 3.86 se muestra el código fuente del algoritmo para obtener el patrón más parecido (menor medida de distancia) para una señal, usado en la etapa de reconocimiento.

```
public static ArchivoMFCC obtenerAudioParecido(ArchivoMFCC archivoPrueba, String rutaDirectorio) {  
    ArchivoMFCC archivoMFCCParecido = null;  
    double menorDistancia = 1000000;  
  
    File[] sub_directorios = new File(rutaDirectorio).listFiles();  
    if (sub_directorios != null && sub_directorios.length > 0) {  
        for (File sub_directorio : sub_directorios) {  
            File[] archivos = sub_directorio.listFiles();  
            if (archivos != null && archivos.length > 0) {  
                for (File archivo : archivos) {  
                    ArchivoMFCC archivoMFCC = new ArchivoMFCC(archivo.getPath());  
                    archivoMFCC.abrirMO();  
                    DTW dtw = new DTW(archivoPrueba.getMo(), archivoMFCC.getMo(), radio: 40, tipoDistancia: 2);  
                    dtw.dtwSimetricoFl2();  
                    if (dtw.getDistanciaDTW() < menorDistancia) {  
                        menorDistancia = dtw.getDistanciaDTW();  
                        archivoMFCCParecido = archivoMFCC;  
                    }  
                }  
            }  
        }  
    }  
    return archivoMFCCParecido;  
}
```

Figura 3.86: Código fuente del algoritmo de reconocimiento de voz para la etapa de reconocimiento.

Fuente: Elaboración propia

En la Figura 3.87 se muestra el código fuente del algoritmo para obtener la respuesta del sistema (identificación correcta, no identificación y falsa identificación) ante la comparación de dos señales de voz, usado en la etapa de toma de decisión.

```

public static String tomaDeDecision(ArchivoMFCC archivoPrueba, ArchivoMFCC archivoEntrena, String tipoReconocedor) {
    String decision;

    DTW dtw = new DTW(archivoPrueba.getMo(), archivoEntrena.getMo(), radio: 40, tipoDistancia: 2);
    dtw.dtwSimetricoP12();
    if (tipoReconocedor.equals("locutor")) {
        if (dtw.getDistanciaDTW() > archivoEntrena.getUmbral()) { //0.45
            decision = "no identificacion";
        } else {
            if (dtw.getDistanciaDTW() < 0.31) {
                decision = "identificacion correcta";
            } else {
                decision = "falsa identificacion";
            }
        }
    } else {
        if (dtw.getDistanciaDTW() > 0.31) {
            decision = "no identificacion";
        } else {
            decision = "identificacion correcta";
        }
    }
    return decision;
}

```

Figura 3.87: Código fuente del algoritmo de reconocimiento de voz para la etapa de toma de decisión.

Fuente: Elaboración propia

En las Figuras 3.88 y 3.89 se muestran las funciones para obtener los umbrales para la toma de decisión del sistema definido por las Ecuaciones (2.105) y (2.106).

```

private static double calcularUmbralDecision(double[] intraLocutor, double[] interLocutor) {
    double u_inter = Preprocesamiento.media(interLocutor);
    double u_intra = Preprocesamiento.media(intraLocutor);
    double o_inter = Preprocesamiento.desviacionEstandar(interLocutor, u_inter);
    double o_intra = Preprocesamiento.desviacionEstandar(intraLocutor, u_intra);

    return (o_intra*u_inter + o_inter*u_intra)/(o_intra + o_inter);
}

```

Figura 3.88: Código fuente de la función de cálculo del umbral de decisión.

Fuente: Elaboración propia

```

public static void obtenerUmbralesParaDecision(@NotNull Context context) {
    File directorio = new File( pathname: context.getFilesDir().getPath() + "/MFCC/Patrones");
    ArrayList<ArchivoMFCC> archivosMFCC = new ArrayList<>();

    if (directorio.exists()) {
        File[] subDirectorios = directorio.listFiles();
        if (subDirectorios != null && subDirectorios.length > 0) {
            for (File subDirectorio : subDirectorios) {
                File[] archivos = subDirectorio.listFiles();
                if (archivos != null && archivos.length > 0) {
                    for (File archivo : archivos) {
                        ArchivoMFCC archivoMFCC = new ArchivoMFCC(archivo.getPath());
                        archivoMFCC.abrirMO();
                        archivosMFCC.add(archivoMFCC);
                    }
                }
            }
        }
    }

    if (!archivosMFCC.isEmpty()) {
        double[][] tabla = new double[archivosMFCC.size()][archivosMFCC.size()];
        for (int i = 0; i < archivosMFCC.size(); i++) {
            for (int j = i+1; j < archivosMFCC.size(); j++) {
                if (archivosMFCC.get(i).getPalabra().equals(archivosMFCC.get(j).getPalabra())) {
                    DTW dtw = new DTW(archivosMFCC.get(i).getMo(), archivosMFCC.get(j).getMo(), radio: 40, tipoDistancia: 2);
                    dtw.dtwSimetricoPl2();
                    tabla[i][j] = dtw.getDistanciaDTW();
                    tabla[j][i] = dtw.getDistanciaDTW();
                }
            }
        }
    }

    ArrayList<Double> intraLocutor, interLocutor;
    for (int i = 0; i < archivosMFCC.size(); i++) {
        intraLocutor = new ArrayList<>();
        interLocutor = new ArrayList<>();
        for (int j = 0; j < archivosMFCC.size(); j++) {
            if (i != j) {
                if (archivosMFCC.get(i).getPalabra().equals(archivosMFCC.get(j).getPalabra())) {
                    if (archivosMFCC.get(i).getUsuario().equals(archivosMFCC.get(j).getUsuario())) {
                        intraLocutor.add(tabla[i][j]);
                    } else {
                        interLocutor.add(tabla[i][j]);
                    }
                }
            }
        }
        if (interLocutor.isEmpty()) {
            archivosMFCC.get(i).setUmbral(Preprocesamiento.media(Utilitarios.convertirListToArray(intraLocutor)));
        } else {
            archivosMFCC.get(i).setUmbral(calcularUmbralDecision(Utilitarios.convertirListToArray(intraLocutor),
                                                               Utilitarios.convertirListToArray(interLocutor)));
        }
        archivosMFCC.get(i).guardarMO();
    }
}
}

```

Figura 3.89: Código fuente para la obtención de los umbrales de decisión para cada patrón de entrenamiento.

Fuente: Elaboración propia

Capítulo 4

Resultados y discusión de la tesis

En este capítulo se mostrarán los resultados obtenidos al realizar las pruebas con el sistema de seguridad para el control de acceso por reconocimiento de voz que fue diseñado e implementado anteriormente, y así poder ver si se obtuvieron los objetivos alcanzados y poder comparar las ventajas y desventajas del proyecto propuesto.

Las pruebas están divididas en 2 partes, la primera consiste en obtener los umbrales de decisión para las respuestas del sistema (Identificación Correcta, Falsa Identificación y No Identificación) y la segunda parte consiste en evaluar el algoritmo de construcción del patrón de referencia definida por la Ecuación (2.101) para la reducción del tiempo de ejecución en la respuesta del sistema.

Para la realización de las pruebas se escogió un lugar con un nivel de ruido bajo de 69 a 78 dBC por lo que no fue necesario utilizar un algoritmo para la eliminación de ruido aditivo. Además, se utilizaron a 11 personas del sexo masculino con edades entre 22 a 24 años, esto porque es evidente

que para el algoritmo le resultará más fácil diferenciar un hombre de una mujer y un niño de un adulto debido algunos aspectos físicos en el tracto vocal que se explicó en la teoría, por ello se escogió a personas del mismo sexo con edades parecidas, así al sistema le resultará más difícil realizar el reconocimiento de voz. A continuación, definiremos algunas abreviaturas que usaremos para la realización de las pruebas:

- *U1*: Umbral de decisión para reconocer la palabra
- *U2*: Umbral de decisión para reconocer al locutor
- *U*: Umbral de decisión definido por la Ecuación (2.105)
- *IC*: Respuesta del sistema de identificación correcta
- *FI*: Respuesta del sistema de falsa identificación
- *NI*: Respuesta del sistema de no identificación
- *A*: Respuesta de acierto en el reconocimiento
- *D*: Respuesta de desacuerdo en el reconocimiento
- *E*: Error de respuesta del sistema

En la ejecución de las pruebas se presentarán 8 escenarios o casos distintos, que a continuación pasaremos a explicar cada uno de ellos:

- Caso 1: Para este caso se empleó a 10 usuarios, cada uno de ellos dijo como palabra clave de acceso su nombre 40 veces (*ANTONY, CARLOS, EDINSON, EDWIN, GERSON, JORDAN, JOSUE, NIZAMA, OCAS* y *RENZO*). De las 40 grabaciones que se obtuvo por cada usuario 30 de ellas pasaran a la etapa de entrenamiento y las otras 10 para las pruebas, haciendo un total de 300 audios para la base de datos y 100 audios para la realización de la prueba.
- Caso 2: Para este caso se empleó a 10 usuarios, cada uno de ellos dijo como palabra clave de acceso *ABRIR* 40 veces. De las 40 grabaciones que se obtuvo por cada usuario 30 de ellas pasaran a la etapa de entrenamiento y las otras 10 para las pruebas, haciendo un total de 300 audios para la base de datos y 100 audios para la realización de la prueba.
- Caso 3: Para este caso se empleó a 10 usuarios, cada uno de ellos dijo como palabra clave de acceso su nombre 30 veces (*ANTONY, CARLOS, EDINSON, EDWIN, GERSON, JORDAN, JOSUE, NIZAMA, OCAS* y *RENZO*), haciendo un total de 300 audios en la base de datos para la etapa de entrenamiento. Luego cada uno de los usuarios intentará acceder diciendo la palabra *ABRIR* 10 veces, generando un total de 100 audios para la realización de la prueba.
- Caso 4: Para este caso se empleó a 10 usuarios, cada uno de ellos dijo como palabra clave de acceso *ABRIR* 30 veces, haciendo un total 300 audios en la base de datos para la etapa de entrenamiento. Luego cada uno de los usuarios intentará acceder diciendo su nombre 10 veces (*ANTONY, CARLOS, EDINSON, EDWIN, GERSON, JORDAN, JOSUE, NIZAMA,*

OCAS y *RENZO*), haciendo un total de 100 audios para la realización de la prueba.

- Caso 5: Para este caso se empleó a 11 usuarios, donde 10 de ellos dijeron como palabra clave de acceso sus nombres 30 veces (*ANTONY*, *CARLOS*, *EDINSON*, *EDWIN*, *GERSON*, *JORDAN*, *JOSUE*, *NIZAMA*, *OCAS* y *RENZO*), haciendo un total de 300 audios en la base de datos para la etapa de entrenamiento. Luego el usuario faltante (usuario no registrado) tratará de acceder con las palabras clave de acceso de los usuarios registrados, pronunciando 10 veces los nombres de cada usuario registrado, generando un total de 100 audios para la realización de la prueba.
- Caso 6: Para este caso se empleó a 11 usuarios, donde 10 de ellos dijeron como palabra clave de acceso *ABRIR* 30 veces, haciendo un total de 300 audios en la base de datos para la etapa de entrenamiento. Luego el usuario faltante (usuario no registrado) tratará de acceder con la palabra clave de acceso *ABRIR*, pronunciando 100 veces esta palabra, generando un total de 100 audios para la realización de la prueba.
- Caso 7: Para este caso se empleó a 11 usuarios, donde 10 de ellos dijeron como palabra clave de acceso sus nombres 30 veces (*ANTONY*, *CARLOS*, *EDINSON*, *EDWIN*, *GERSON*, *JORDAN*, *JOSUE*, *NIZAMA*, *OCAS* y *RENZO*), haciendo un total de 300 audios en la base de datos para la etapa de entrenamiento. Luego el usuario faltante (usuario no registrado) tratará de acceder con la palabra *ABRIR*, pronunciando 100 veces esta palabra, generando un

total de 100 audios para la realización de la prueba.

- Caso 8: Para este caso se empleó a 11 usuarios, donde 10 de ellos dijeron como palabra clave de acceso *ABRIR* 30 veces, haciendo un total de 300 audios en la base de datos para la etapa de entrenamiento. Luego el usuario faltante (usuario no registrado) tratará de acceder con los nombres de los usuarios registrados, pronunciando 10 veces los nombres de cada usuario registrado, generando un total de 100 audios para la realización de la prueba.

4.1. Pruebas para obtener los umbrales de decisión

Esta parte consta de 2 etapas, la primera será en hallar U_2 valor que nos permitirá decir si es o no el locutor ante una falsa identificación además U_1 tendrá como valor a U , una vez obtenido el mejor valor para U_2 pasaremos a la segunda etapa que consiste en hallar U_1 valor que nos permitirá decir si es o no la palabra hablada.

4.1.1. Prueba para obtener U_2

4.1.1.1. Caso 1

En la Tabla 4.1 podemos observar que $A = 100$ y $D = 0$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo el patrón de referencia correcto (es el locutor y es la palabra) para cada audio de prueba, con esto podemos decir que el algoritmo alcanzó un 100 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será IC , podemos ver en la Tabla 4.1 que conforme $U2$ aumenta de valor E disminuye, esto porque al ser $U2$ más grande permite un mayor paso de accesos al sistema, pero hay que tener mucho cuidado ya que este valor no puede ser muy grande porque podría dejar pasar a usuarios intrusos, haciendo que el sistema sea inseguro.

Tabla 4.1: Resultados para obtener U2 en el caso 1.

U1	U2	IC	FI	NI	A	D	E
U	0.20	50	50	0	100	0	50
U	0.21	57	43	0	100	0	43
U	0.22	61	39	0	100	0	39
U	0.23	65	35	0	100	0	35
U	0.24	71	29	0	100	0	29
U	0.25	77	23	0	100	0	23
U	0.26	81	19	0	100	0	19
U	0.27	84	16	0	100	0	16
U	0.28	86	14	0	100	0	14
U	0.29	89	11	0	100	0	11
U	0.30	93	7	0	100	0	7
U	0.31	95	5	0	100	0	5
U	0.32	96	4	0	100	0	4
U	0.33	99	1	0	100	0	1
U	0.34	99	1	0	100	0	1
U	0.35	100	0	0	100	0	0

Fuente: Elaboración propia

4.1.1.2. Caso 2

En la Tabla 4.2 podemos observar que $A = 100$ y $D = 0$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo el patrón de referencia correcto (es el locutor y es la palabra) para cada audio de prueba, con esto podemos decir que el algoritmo alcanzó un 100 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será IC , podemos ver en la Tabla 4.2 que conforme $U2$ aumenta de valor E disminuye, tal como en el caso anterior.

Tabla 4.2: Resultados para obtener $U2$ en el caso 2.

U1	U2	IC	FI	NI	A	D	E
U	0.20	64	36	0	100	0	36
U	0.21	70	30	0	100	0	30
U	0.22	77	23	0	100	0	23
U	0.23	83	17	0	100	0	17
U	0.24	87	13	0	100	0	13
U	0.25	91	9	0	100	0	9
U	0.26	94	6	0	100	0	6
U	0.27	96	4	0	100	0	4
U	0.28	97	3	0	100	0	3
U	0.29	97	3	0	100	0	3
U	0.30	99	1	0	100	0	1
U	0.31	99	1	0	100	0	1
U	0.32	99	1	0	100	0	1
U	0.33	99	1	0	100	0	1
U	0.34	99	1	0	100	0	1
U	0.35	100	0	0	100	0	0

Fuente: Elaboración propia

4.1.1.3. Caso 3

En la Tabla 4.3 podemos observar que $A = 0$ y $D = 100$, esto es porque los usuarios pronunciaron una palabra distinta a las que se encuentran en la base de datos.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.3 que para cualquier valor de $U2$ el valor E sigue siendo el mismo, esto es porque la respuesta depende de $U1$ que para esta prueba su valor se define por la Ecuación (2.105), en todo caso si se quiere disminuir FI el valor de $U1$ tendría que ser menor,

pero esto lo veremos en la segunda etapa. Lo importante es que $IC = 0$ por lo que el sistema no dejó pasar a ningún usuario con clave de acceso incorrecta.

Tabla 4.3: Resultados para obtener U2 en el caso 3.

U1	U2	IC	FI	NI	A	D	E
U	0.20	0	13	87	0	100	13
U	0.21	0	13	87	0	100	13
U	0.22	0	13	87	0	100	13
U	0.23	0	13	87	0	100	13
U	0.24	0	13	87	0	100	13
U	0.25	0	13	87	0	100	13
U	0.26	0	13	87	0	100	13
U	0.27	0	13	87	0	100	13
U	0.28	0	13	87	0	100	13
U	0.29	0	13	87	0	100	13
U	0.30	0	13	87	0	100	13
U	0.31	0	13	87	0	100	13
U	0.32	0	13	87	0	100	13
U	0.33	0	13	87	0	100	13
U	0.34	0	13	87	0	100	13
U	0.35	0	13	87	0	100	13

Fuente: Elaboración propia

4.1.1.4. Caso 4

En la Tabla 4.4 podemos observar que $A = 0$ y $D = 100$, esto es porque los usuarios pronunciaron una palabra distinta a las que se encuentran en la base de datos.

En cuanto a la respuesta de decisión del sistema en este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.4 que el valor de $E = 0$, por lo que para este caso el sistema alcanzó un 100 % de efectividad en su respuesta.

Tabla 4.4: Resultados para obtener U2 en el caso 4.

U1	U2	IC	FI	NI	A	D	E
U	0.20	0	0	100	0	100	0
U	0.21	0	0	100	0	100	0
U	0.22	0	0	100	0	100	0
U	0.23	0	0	100	0	100	0
U	0.24	0	0	100	0	100	0
U	0.25	0	0	100	0	100	0
U	0.26	0	0	100	0	100	0
U	0.27	0	0	100	0	100	0
U	0.28	0	0	100	0	100	0
U	0.29	0	0	100	0	100	0
U	0.30	0	0	100	0	100	0
U	0.31	0	0	100	0	100	0
U	0.32	0	0	100	0	100	0
U	0.33	0	0	100	0	100	0
U	0.34	0	0	100	0	100	0
U	0.35	0	0	100	0	100	0

Fuente: Elaboración propia

4.1.1.5. Caso 5

En la Tabla 4.5 podemos observar que $A = 90$ y $D = 10$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo un 90 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será FI , podemos ver en la Tabla 4.5 que para cualquier valor de $U2$ el valor de E sigue siendo el mismo, esto es porque la respuesta depende de $U1$ que para esta prueba es U definido por la Ecuación (2.105), para disminuir el valor de NI es necesario que $U1$ tenga un mayor valor, pero esto lo veremos en la segunda etapa. Lo importante es que $IC = 0$ por lo que el sistema no dejó pasar a ningún intruso.

Tabla 4.5: Resultados para obtener U2 en el caso 5.

U1	U2	IC	FI	NI	A	D	E
U	0.20	0	36	64	90	10	64
U	0.21	0	36	64	90	10	64
U	0.22	0	36	64	90	10	64
U	0.23	0	36	64	90	10	64
U	0.24	0	36	64	90	10	64
U	0.25	0	36	64	90	10	64
U	0.26	0	36	64	90	10	64
U	0.27	0	36	64	90	10	64
U	0.28	0	36	64	90	10	64
U	0.29	0	36	64	90	10	64
U	0.30	0	36	64	90	10	64
U	0.31	0	36	64	90	10	64
U	0.32	0	36	64	90	10	64
U	0.33	0	36	64	90	10	64
U	0.34	0	36	64	90	10	64
U	0.35	0	36	64	90	10	64

Fuente: Elaboración propia

4.1.1.6. Caso 6

En la Tabla 4.6 podemos observar que $A = 100$ y $D = 0$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo un 100 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será FI , podemos ver en la Tabla 4.6 que para $U2 \leq 0,32$ el valor de $IC = 0$, de lo contrario el número de IC aumenta, esto quiere decir que el sistema va permitiendo cada vez más el acceso a intrusos.

Tabla 4.6: Resultados para obtener U2 en el caso 6.

U1	U2	IC	FI	NI	A	D	E
U	0.20	0	46	54	100	0	54
U	0.21	0	46	54	100	0	54
U	0.22	0	46	54	100	0	54
U	0.23	0	46	54	100	0	54
U	0.24	0	46	54	100	0	54
U	0.25	0	46	54	100	0	54
U	0.26	0	46	54	100	0	54
U	0.27	0	46	54	100	0	54
U	0.28	0	46	54	100	0	54
U	0.29	0	46	54	100	0	54
U	0.30	0	46	54	100	0	54
U	0.31	0	46	54	100	0	54
U	0.32	0	46	54	100	0	54
U	0.33	1	45	54	100	0	55
U	0.34	1	45	54	100	0	55
U	0.35	5	41	54	100	0	56

Fuente: Elaboración propia

4.1.1.7. Caso 7

En la Tabla 4.7 podemos observar que $A = 0$ y $D = 100$, esto es porque el usuario pronunció una palabra distinta a las que se encuentran en la base de datos.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.7 que para cualquier valor de $U2$ el valor E sigue siendo el mismo, esto es porque la respuesta depende de $U1$ que para esta prueba es U valor definido por la Ecuación (2.105), en todo caso si se quiere disminuir FI el valor de $U1$ tendría que ser menor, pero esto no es necesario ya que $E = 1$, por lo que el sistema alcanzó un 99 % de efectividad en su respuesta. Además, lo importante es que $IC = 0$ lo que quiere decir que el sistema no dejó pasar a ningún intruso.

Tabla 4.7: Resultados para obtener U2 en el caso 7.

U1	U2	IC	FI	NI	A	D	E
U	0.20	0	1	99	0	100	1
U	0.21	0	1	99	0	100	1
U	0.22	0	1	99	0	100	1
U	0.23	0	1	99	0	100	1
U	0.24	0	1	99	0	100	1
U	0.25	0	1	99	0	100	1
U	0.26	0	1	99	0	100	1
U	0.27	0	1	99	0	100	1
U	0.28	0	1	99	0	100	1
U	0.29	0	1	99	0	100	1
U	0.30	0	1	99	0	100	1
U	0.31	0	1	99	0	100	1
U	0.32	0	1	99	0	100	1
U	0.33	0	1	99	0	100	1
U	0.34	0	1	99	0	100	1
U	0.35	0	1	99	0	100	1

Fuente: Elaboración propia

4.1.1.8. Caso 8

En la Tabla 4.8 podemos observar que $A = 0$ y $D = 100$, esto es porque el usuario pronunció una palabra distinta a las que se encuentran en la base de datos.

En cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.8 que el valor de $E = 0$, por lo que para este caso el sistema alcanzó un 100 % de efectividad en su respuesta.

Tabla 4.8: : Resultados para obtener U2 en el caso 8.

U1	U2	IC	FI	NI	A	D	E
U	0.20	0	0	100	0	100	0
U	0.21	0	0	100	0	100	0
U	0.22	0	0	100	0	100	0
U	0.23	0	0	100	0	100	0
U	0.24	0	0	100	0	100	0
U	0.25	0	0	100	0	100	0
U	0.26	0	0	100	0	100	0
U	0.27	0	0	100	0	100	0
U	0.28	0	0	100	0	100	0
U	0.29	0	0	100	0	100	0
U	0.30	0	0	100	0	100	0
U	0.31	0	0	100	0	100	0
U	0.32	0	0	100	0	100	0
U	0.33	0	0	100	0	100	0
U	0.34	0	0	100	0	100	0
U	0.35	0	0	100	0	100	0

Fuente: Elaboración propia

De esta primera prueba podemos ver que el mejor valor para $U2$ es 0.31. Esto porque, observamos que para los resultados de los casos 3, 4, 5, 7 y 8 cualquier valor que tome $U2$ entre 0.20 y 0.35 el valor de E es el mismo, por lo que no tomaremos en cuenta estos casos, entonces de las Tablas 4.1 y 4.2 de los casos 1 y 2 respectivamente podemos ver que el mejor valor para $U2$ es 0.35 dando $E = 0$ alcanzando el sistema un 100 % de efectividad en la respuesta del reconocimiento de voz, ver la Figura 4.1, sin embargo, para el caso 6 el valor de IC sería 5 esto quiere decir que de las 100 veces que el intruso intento acceder al sistema en 5 de ellas el sistema le permitió el acceso, y esto es grave ya que nosotros lo que buscamos es la seguridad, por lo que $U2$ tendrá que ser menor que 0.32 y mayor que 0.30 para que el número de FI no sea grande para los casos 1 y 2, por lo que se escogió a 0.31 como valor intermedio, ver la Figura 4.2.

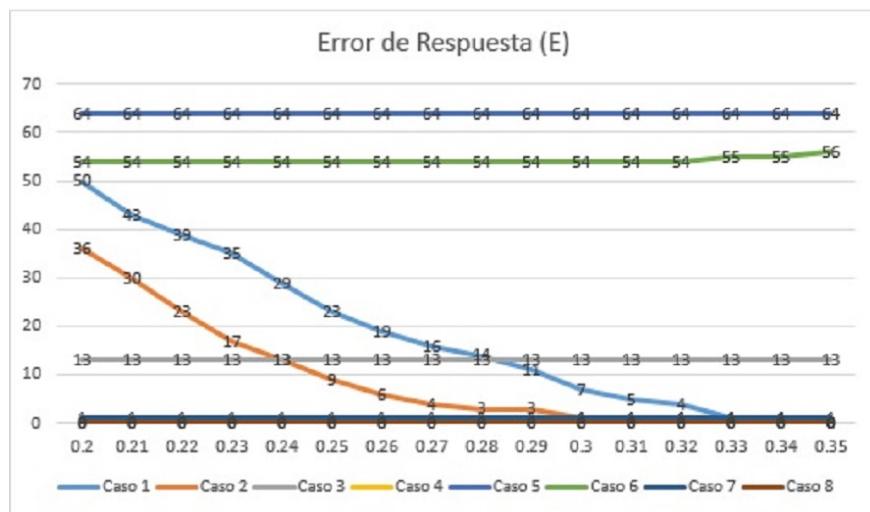


Figura 4.1: Número de errores en la respuesta del sistema para U2.

Fuente: Elaboración propia

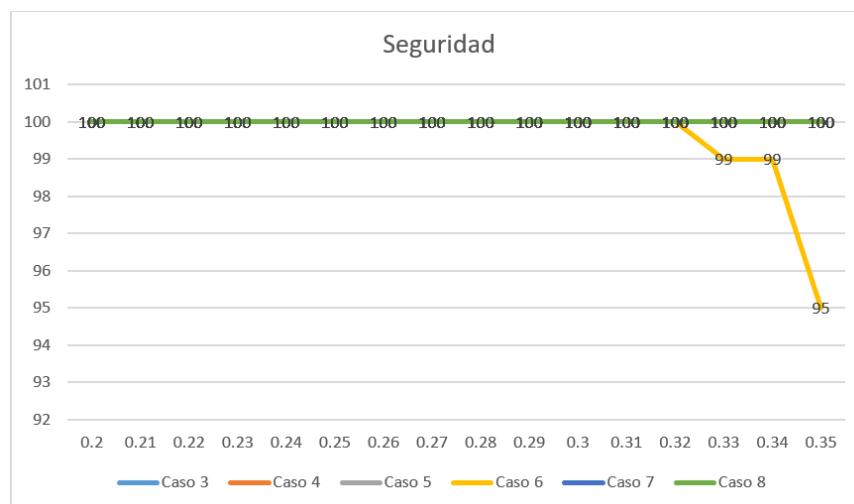


Figura 4.2: Seguridad del sistema para U2.

Fuente: Elaboración propia

4.1.2. Prueba para obtener U1

4.1.2.1. Caso 1

En la Tabla 4.9 podemos observar que $A = 100$ y $D = 0$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo el patrón de referencia correcto (es el locutor y es la palabra) para cada audio de prueba, con esto podemos decir que el algoritmo alcanzó un 100 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será IC , podemos ver en la Tabla 4.9 que para cualquier valor de $U1$ el valor de E sigue siendo el mismo, esto es porque la respuesta depende de $U2$, por lo que para disminuir el número de FI tendría que $U2$ tener un mayor valor, pero como se dijo anteriormente si este valor es muy grande el sistema permitiría el acceso a intrusos y esto es lo que no se desea. Sin embargo, $E = 5$ esto quiere decir que el sistema alcanzó un 95 % de efectividad en su respuesta.

Tabla 4.9: Resultados para obtener U1 en el caso 1.

U1	U2	IC	FI	NI	A	D	E
0.40	0.31	95	5	0	100	0	5
0.41	0.31	95	5	0	100	0	5
0.42	0.31	95	5	0	100	0	5
0.43	0.31	95	5	0	100	0	5
0.44	0.31	95	5	0	100	0	5
0.45	0.31	95	5	0	100	0	5
0.46	0.31	95	5	0	100	0	5
0.47	0.31	95	5	0	100	0	5
0.48	0.31	95	5	0	100	0	5
0.49	0.31	95	5	0	100	0	5
0.50	0.31	95	5	0	100	0	5
0.51	0.31	95	5	0	100	0	5
0.52	0.31	95	5	0	100	0	5
0.53	0.31	95	5	0	100	0	5
0.54	0.31	95	5	0	100	0	5
0.55	0.31	95	5	0	100	0	5

Fuente: Elaboración propia

4.1.2.2. Caso 2

En la Tabla 4.10 podemos observar que $A = 100$ y $D = 0$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo el patrón de referencia correcto (es el locutor y es la palabra) para cada audio de prueba, con esto podemos decir que el algoritmo alcanzó un 100 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será IC , podemos ver en la Tabla 4.10 que para cualquier valor de $U1$ el valor de E sigue siendo el mismo, tal como en el caso anterior. Sin embargo, $E = 1$ esto quiere decir que el sistema alcanzó un 99 % de efectividad en su respuesta.

Tabla 4.10: Resultados para obtener U1 en el caso 2.

U1	U2	IC	FI	NI	A	D	E
0.40	0.31	99	1	0	100	0	1
0.41	0.31	99	1	0	100	0	1
0.42	0.31	99	1	0	100	0	1
0.43	0.31	99	1	0	100	0	1
0.44	0.31	99	1	0	100	0	1
0.45	0.31	99	1	0	100	0	1
0.46	0.31	99	1	0	100	0	1
0.47	0.31	99	1	0	100	0	1
0.48	0.31	99	1	0	100	0	1
0.49	0.31	99	1	0	100	0	1
0.50	0.31	99	1	0	100	0	1
0.51	0.31	99	1	0	100	0	1
0.52	0.31	99	1	0	100	0	1
0.53	0.31	99	1	0	100	0	1
0.54	0.31	99	1	0	100	0	1
0.55	0.31	99	1	0	100	0	1

Fuente: Elaboración propia

4.1.2.3. Caso 3

En la Tabla 4.11 podemos observar que $A = 0$ y $D = 100$, esto es porque los usuarios pronunciaron una palabra distinta a las que se encuentran en la base de datos.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.11 que para $U1 \leq 0,51$ el valor de $E = 0$, pero si $U1$ aumenta entonces aumentará el número de FI y eso no es correcto, sin embargo el valor de E es pequeño por lo que esto es un buen resultado. Además, $IC = 0$ por lo que el sistema no dejó pasar a ningún usuario con clave de acceso incorrecta.

Tabla 4.11: Resultados para obtener U1 en el caso 3.

U1	U2	IC	FI	NI	A	D	E
0.40	0.31	0	0	100	0	100	0
0.41	0.31	0	0	100	0	100	0
0.42	0.31	0	0	100	0	100	0
0.43	0.31	0	0	100	0	100	0
0.44	0.31	0	0	100	0	100	0
0.45	0.31	0	0	100	0	100	0
0.46	0.31	0	0	100	0	100	0
0.47	0.31	0	0	100	0	100	0
0.48	0.31	0	0	100	0	100	0
0.49	0.31	0	0	100	0	100	0
0.50	0.31	0	0	100	0	100	0
0.51	0.31	0	0	100	0	100	0
0.52	0.31	0	1	99	0	100	1
0.53	0.31	0	1	99	0	100	1
0.54	0.31	0	1	99	0	100	1
0.55	0.31	0	1	99	0	100	1

Fuente: Elaboración propia

4.1.2.4. Caso 4

En la Tabla 4.12 podemos observar que $A = 0$ y $D = 100$, esto es porque los usuarios pronunciaron una palabra distinta a las que se encuentran en la base de datos.

En cuanto a la respuesta de decisión del sistema en este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.12 que el valor de $E = 0$ mientras $U1 \leq 0,49$, pero si este aumenta entonces aumentará el número de FI ocurriendo lo mismo que en el caso anterior.

Tabla 4.12: Resultados para obtener U1 en el caso 4.

U1	U2	IC	FI	NI	A	D	E
0.40	0.31	0	0	100	0	100	0
0.41	0.31	0	0	100	0	100	0
0.42	0.31	0	0	100	0	100	0
0.43	0.31	0	0	100	0	100	0
0.44	0.31	0	0	100	0	100	0
0.45	0.31	0	0	100	0	100	0
0.46	0.31	0	0	100	0	100	0
0.47	0.31	0	0	100	0	100	0
0.48	0.31	0	0	100	0	100	0
0.49	0.31	0	0	100	0	100	0
0.50	0.31	0	1	99	0	100	1
0.51	0.31	0	1	99	0	100	1
0.52	0.31	0	1	99	0	100	1
0.53	0.31	0	1	99	0	100	1
0.54	0.31	0	1	99	0	100	1
0.55	0.31	0	2	98	0	100	2

Fuente: Elaboración propia

4.1.2.5. Caso 5

En la Tabla 4.13 podemos observar que $A = 90$ y $D = 10$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo un 90 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será FI , podemos ver en la Tabla 4.13 que conforme aumentamos el valor de $U1$ entonces E disminuye, pero si $U1$ es muy grande el sistema no podría diferenciar las palabras que son parecidas, por ejemplo, *ENTENDER* con *ENCENDER*. Sin embargo, $IC = 0$ por lo que el sistema no dejó pasar a ningún intruso.

Tabla 4.13: Resultados para obtener U1 en el caso 5.

U1	U2	IC	FI	NI	A	D	E
0.40	0.31	0	0	100	90	10	100
0.41	0.31	0	1	99	90	10	99
0.42	0.31	0	1	99	90	10	99
0.43	0.31	0	2	98	90	10	98
0.44	0.31	0	4	96	90	10	96
0.45	0.31	0	6	94	90	10	94
0.46	0.31	0	8	92	90	10	92
0.47	0.31	0	13	87	90	10	87
0.48	0.31	0	17	83	90	10	83
0.49	0.31	0	21	79	90	10	79
0.50	0.31	0	22	78	90	10	78
0.51	0.31	0	28	72	90	10	72
0.52	0.31	0	32	68	90	10	68
0.53	0.31	0	40	60	90	10	60
0.54	0.31	0	46	54	90	10	54
0.55	0.31	0	51	49	90	10	51

Fuente: Elaboración propia

4.1.2.6. Caso 6

En la Tabla 4.14 podemos observar que $A = 100$ y $D = 0$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo un 100 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será FI , podemos ver en la Tabla 4.14 que conforme aumentamos de valor a $U1$ entonces E disminuye, tal como en el caso anterior se explicó que $U1$ no puede ser muy grande, sin embargo, se puede ver que en este caso E tiene menores valores que en el caso anterior.

Tabla 4.14: Resultados para obtener U1 en el caso 6.

U1	U2	IC	FI	NI	A	D	E
0.40	0.31	0	28	72	100	0	72
0.41	0.31	0	37	63	100	0	63
0.42	0.31	0	42	58	100	0	58
0.43	0.31	0	57	43	100	0	43
0.44	0.31	0	67	33	100	0	33
0.45	0.31	0	71	29	100	0	29
0.46	0.31	0	78	22	100	0	22
0.47	0.31	0	84	16	100	0	16
0.48	0.31	0	87	13	100	0	13
0.49	0.31	0	92	8	100	0	8
0.50	0.31	0	96	4	100	0	4
0.51	0.31	0	98	2	100	0	2
0.52	0.31	0	98	2	100	0	2
0.53	0.31	0	99	1	100	0	1
0.54	0.31	0	100	0	100	0	0
0.55	0.31	0	100	0	100	0	0

Fuente: Elaboración propia

4.1.2.7. Caso 7

En la Tabla 4.15 podemos observar que $A = 0$ y $D = 100$, esto es porque el usuario pronunció una palabra distinta a las que se encuentran en la base de datos.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.15 que para cualquier valor de $U1$ el valor de $E = 0$, para que el número de FI aumente $U1$ tiene que ser más grande, pero esto es lo que no se quiere. Sin embargo, $E = 0$ por lo que el sistema alcanzo un 100 % de efectividad en su respuesta.

Tabla 4.15: Resultados para obtener U1 en el caso 7.

U1	U2	IC	FI	NI	A	D	E
0.40	0.31	0	0	100	0	100	0
0.41	0.31	0	0	100	0	100	0
0.42	0.31	0	0	100	0	100	0
0.43	0.31	0	0	100	0	100	0
0.44	0.31	0	0	100	0	100	0
0.45	0.31	0	0	100	0	100	0
0.46	0.31	0	0	100	0	100	0
0.47	0.31	0	0	100	0	100	0
0.48	0.31	0	0	100	0	100	0
0.49	0.31	0	0	100	0	100	0
0.50	0.31	0	0	100	0	100	0
0.51	0.31	0	0	100	0	100	0
0.52	0.31	0	0	100	0	100	0
0.53	0.31	0	0	100	0	100	0
0.54	0.31	0	0	100	0	100	0
0.55	0.31	0	0	100	0	100	0

Fuente: Elaboración propia

4.1.2.8. Caso 8

En la Tabla 4.16 podemos observar que $A = 0$ y $D = 100$, esto es porque el usuario pronunció una palabra distinta a las que se encuentran en la base de datos.

En cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.16 que conforme aumentamos de valor a $U1$ entonces E aumenta, tal como se explicó anteriormente, y esto no es lo que se quiere, sin embargo el número de FI sigue siendo pequeño al igual que E y esto es bueno.

Tabla 4.16: Resultados para obtener U1 en el caso 8.

U1	U2	IC	FI	NI	A	D	E
0.40	0.31	0	0	100	0	100	0
0.41	0.31	0	0	100	0	100	0
0.42	0.31	0	0	100	0	100	0
0.43	0.31	0	0	100	0	100	0
0.44	0.31	0	0	100	0	100	0
0.45	0.31	0	0	100	0	100	0
0.46	0.31	0	1	99	0	100	1
0.47	0.31	0	1	99	0	100	1
0.48	0.31	0	1	99	0	100	1
0.49	0.31	0	1	99	0	100	1
0.50	0.31	0	2	98	0	100	2
0.51	0.31	0	3	97	0	100	3
0.52	0.31	0	3	97	0	100	3
0.53	0.31	0	3	97	0	100	3
0.54	0.31	0	3	97	0	100	3
0.55	0.31	0	3	97	0	100	3

Fuente: Elaboración propia

De esta segunda prueba podemos ver que el mejor valor para $U1$ es 0.51. Esto porque, observamos que para los resultados de los casos 1, 2 y 7 cualquier valor que tome $U1$ entre 0.40 y 0.55 el valor de E es el mismo, por lo que no tomaremos en cuenta estos casos, entonces de las tablas 4.11, 4.12 y 4.16 de los casos 3, 4 y 8 respectivamente, podemos ver que conforme $U1$ aumenta de valor E también aumenta, por otro lado, para los casos 5 y 6 si $U1$ aumenta E disminuye, por ello se escogió a 0.51 como mejor valor intermedio para los casos 3, 4, 5, 6 y 8, ver la Figura 4.3. Además, el sistema no dejó pasar en ningún caso a intrusos por lo que alcanzó un 100 % en seguridad, ver la Figura 4.4.

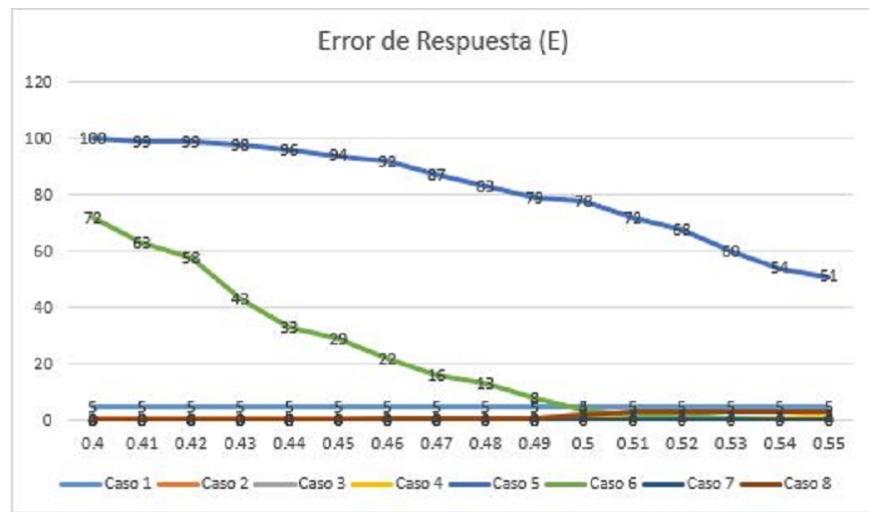


Figura 4.3: Número de errores en la respuesta del sistema para U1.

Fuente: Elaboración propia

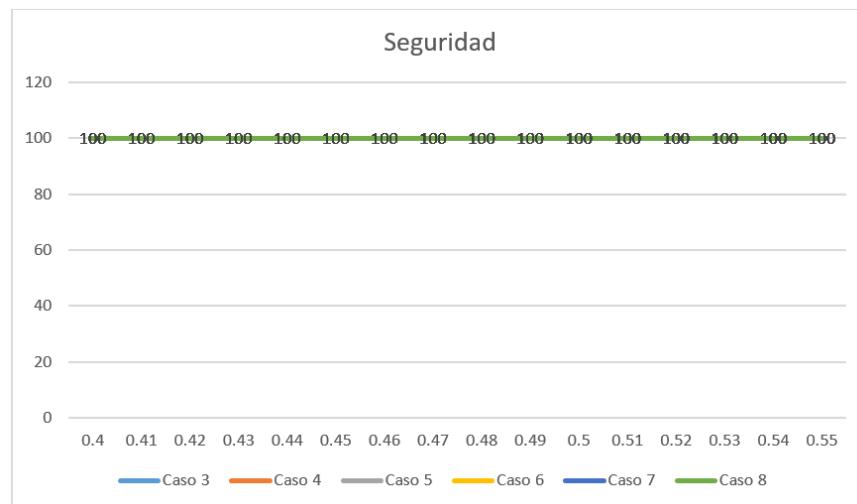


Figura 4.4: Seguridad del sistema para U1.

Fuente: Elaboración propia

De esta primera etapa se puede concluir que los valores para los umbrales U_1 y U_2 son, para $U_2 = 0,31$ sin embargo todavía no hemos escogido cual es el mejor valor para U_1 si U o 0.51 , para ello haremos una comparación del total de errores obtenidos para esto dos valores.

Tabla 4.17: Comparación de numero de errores entre $U_1 = U$ y $U_1 = 0.51$.

N°	U_2	$U_1=U$	$U_1=0.51$
1	0.31	5	5
2	0.31	1	1
3	0.31	13	0
4	0.31	0	1
5	0.31	64	72
6	0.31	54	2
7	0.31	1	0
8	0.31	0	3
Total de Error		138	84

Fuente: Elaboración propia

En la Tabla 4.17 se puede ver que para $U_1 = 0,51$ se obtuvo una menor cantidad de número de errores en la respuesta del sistema de reconocimiento de voz, debido a esto se escogerá este valor.

Ahora evaluaremos el tiempo de ejecución que toma el sistema para realizar el reconocimiento de voz y el reajuste de los umbrales, si bien este último no es necesario evaluar ya que el valor de U_1 es 0.51 un valor estático y no dinámico (calculado por la Ecuación (2.105)), lo haremos igual como información complementaria para la investigación.

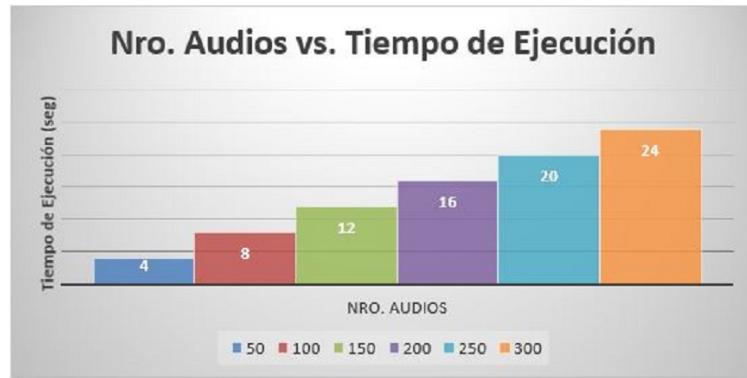


Figura 4.5: Gráfico de barras del tiempo de ejecución para el reconocimiento de voz.

Fuente: Elaboración propia

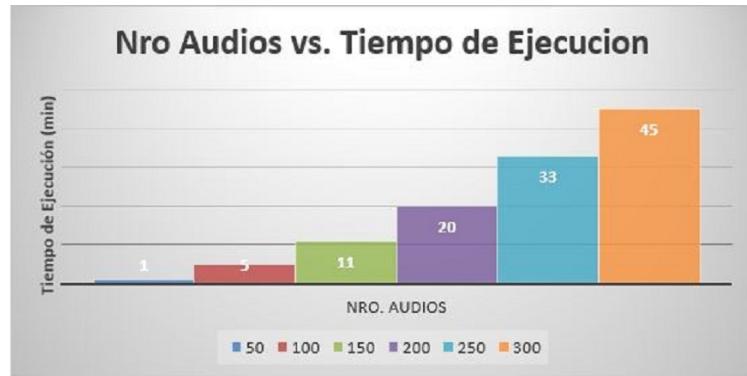


Figura 4.6: Gráfico de barras del tiempo de ejecución para el reajuste del umbral U1.

Fuente: Elaboración propia

Como podemos ver en la Figura 4.5 el gráfico muestra una pendiente de función lineal, conforme aumentamos el número de audios en la base de datos el tiempo de ejecución para la respuesta del sistema aumenta, por lo que esta puede ser calculada por la siguiente fórmula:

$$TE = txN \quad (4.1)$$

Donde:

- TE : Tiempo de ejecución para la respuesta del sistema
- t : Tiempo que demora en comparar dos audios
- N : Número total de audios en la base de datos

De la Ecuación (4.1) y los datos obtenidos que se ve en la Figura 4.5, se tiene que el valor de t es 0.08 segundos, este valor varía de acuerdo al hardware del dispositivo en donde se implemente el sistema, para nuestro proyecto hemos usado un dispositivo móvil Android Samsung J6+ con un procesador de 4 núcleos de 1.8 GHz.

El tiempo de respuesta del sistema para el reconocimiento de voz no debe ser mayor a 5 segundos tal como se definió en las características generales del ciclo de negocio de la arquitectura del sistema en la Tabla 3.1, sin embargo, si se utiliza un número mayor a 60 audios en la base de datos, entonces no se cumplira con este requisito que se planteó.

Por ello, se planteó utilizar la técnica para la construcción del patrón de referencia definida por la Ecuación (2.101) que consistía en agrupar un número de audios y dar como resultado un patrón promedio que los represente, así podremos disminuir el tiempo de respuesta del sistema al tener un número menor de patrones en la base de datos.

4.2. Pruebas para la construcción del patrón de referencia

En esta segunda parte al igual que la primera consta de 2 etapas, en la primera el umbral $U1$ se considerará con valor dinámico es decir el valor definido por la Ecuación (2.105) y para la segunda etapa el umbral $U1$ tendrá un valor estático que será 0.51, valor que fue escogido como el mejor en la primera parte. De esta manera procederemos a evaluar el algoritmo de construcción del patrón de referencia en estos dos escenarios.

A continuación, veremos los resultados que se obtuvieron en la realización de las pruebas para los 8 escenarios o casos definidos anteriormente, donde N representa el número de patrones o audios que se emplearán para la construcción del patrón de referencia.

4.2.1. Prueba con U1 dinámico

4.2.1.1. Caso 1

En la Tabla 4.18 podemos observar que $A = 100$ y $D = 0$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo el patrón de referencia correcto (es el locutor y es la palabra) para cada audio de prueba, con esto podemos decir que el algoritmo alcanzó un 100 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será IC , podemos ver en la Tabla 4.18 que se obtuvo un menor número de errores de respuesta

para $N = 5$ teniendo como resultado a $E = 15$ siendo de estos $FI = 7$ y $NI = 8$, además este valor es muy alto y no cumple con los requisitos que se propuso en la arquitectura del ciclo de negocio, por lo que este no es un buen resultado.

Tabla 4.18: Resultados para el caso 1 con U1 dinámico.

N°	U1	U2	IC	FI	NI	A	D	E
5	U	0.31	85	7	8	100	0	15
6	U	0.31	75	9	16	100	0	25
7	U	0.31	69	4	27	100	0	31
8	U	0.31	57	0	43	100	0	43
9	U	0.31	55	4	41	100	0	45
10	U	0.31	68	8	24	100	0	32

Fuente: Elaboración propia

4.2.1.2. Caso 2

En la Tabla 4.19 podemos observar que $A = 100$ y $D = 0$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo el patrón de referencia correcto (es el locutor y es la palabra) para cada audio de prueba, con esto podemos decir que el algoritmo alcanzó un 100 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será IC , podemos ver en la Tabla 4.19 que se obtuvo un menor número de errores de respuesta para $N = 6$ teniendo como resultado a $E = 13$ siendo de estos $FI = 0$ y $NI = 13$, además este valor es muy alto y no cumple con los requisitos que se propuso en la arquitectura del ciclo de negocio, por lo que este no es un buen resultado.

Tabla 4.19: Resultados para el caso 2 con U1 dinámico.

<i>Nº</i>	U1	U2	IC	FI	NI	A	D	E
5	U	0.31	86	0	14	100	0	14
6	U	0.31	87	0	13	100	0	13
7	U	0.31	76	1	23	100	0	24
8	U	0.31	60	0	40	100	0	40
9	U	0.31	71	0	29	100	0	29
10	U	0.31	78	0	22	100	0	22

Fuente: Elaboración propia

4.2.1.3. Caso 3

En la Tabla 4.20 podemos observar que $A = 0$ y $D = 100$, esto es porque los usuarios pronunciaron una palabra distinta a las que se encuentran en la base de datos.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.20 que para cualquier valor de N el valor de $E = 0$, logrando un excelente resultado de un 100 % de efectividad en la respuesta del sistema.

Tabla 4.20: Resultados para el caso 3 con U1 dinámico.

<i>Nº</i>	U1	U2	IC	FI	NI	A	D	E
5	U	0.31	0	0	100	0	100	0
6	U	0.31	0	0	100	0	100	0
7	U	0.31	0	0	100	0	100	0
8	U	0.31	0	0	100	0	100	0
9	U	0.31	0	0	100	0	100	0
10	U	0.31	0	0	100	0	100	0

Fuente: Elaboración propia

4.2.1.4. Caso 4

En la Tabla 4.21 podemos observar que $A = 0$ y $D = 100$, esto es porque los usuarios pronunciaron una palabra distinta a las que se encuentran en la base de datos.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.21 que para cualquier valor de N el valor de $E = 0$, logrando un excelente resultado igual que en el caso anterior de un 100 % de efectividad en la respuesta del sistema.

Tabla 4.21: Resultados para el caso 4 con U1 dinámico.

N°	U1	U2	IC	FI	NI	A	D	E
5	U	0.31	0	0	100	0	100	0
6	U	0.31	0	0	100	0	100	0
7	U	0.31	0	0	100	0	100	0
8	U	0.31	0	0	100	0	100	0
9	U	0.31	0	0	100	0	100	0
10	U	0.31	0	0	100	0	100	0

Fuente: Elaboración propia

4.2.1.5. Caso 5

En la Tabla 4.22 podemos observar que para $N = 8$ se obtuvo un número menor de desaciertos con $D = 7$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo un 93 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será FI , podemos ver en la Tabla 4.22 que se obtuvo un número menor de errores de respuesta

para $N = 10$ dando como resultado a $E = 95$, si bien este valor es muy malo, de estos todos fueron NI siendo $IC = 0$, por lo que el sistema no dejó pasar a ningún intruso, sin embargo, el valor de E es muy grande y no cumple con los requisitos de la arquitectura del sistema.

Tabla 4.22: Resultados para el caso 5 con U1 dinámico.

N°	U1	U2	IC	FI	NI	A	D	E
5	U	0.31	0	3	97	86	14	97
6	U	0.31	0	4	96	88	12	96
7	U	0.31	0	2	98	92	8	98
8	U	0.31	0	0	100	93	7	100
9	U	0.31	0	0	100	83	17	100
10	U	0.31	0	5	95	83	17	95

Fuente: Elaboración propia

4.2.1.6. Caso 6

En la Tabla 4.23 podemos observar que $A = 100$ y $D = 0$, esto quiere decir que el algoritmo de reconocimiento de voz alcanzó un 100 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será FI , podemos ver en la Tabla 4.23 que se obtuvo un número menor de errores de respuesta para $N = 9$ con $E = 96$, si bien este valor es muy grande todos estos fueron por NI siendo $IC = 0$, por lo que el sistema no dejó pasar a ningún intruso, sin embargo, el valor de E es muy grande y no cumple con los requisitos de la arquitectura del sistema.

Tabla 4.23: Resultados para el caso 6 con U1 dinámico.

<i>Nº</i>	U1	U2	IC	FI	NI	A	D	E
5	U	0.31	0	0	100	100	0	100
6	U	0.31	0	0	100	100	0	100
7	U	0.31	0	1	99	100	0	99
8	U	0.31	0	0	100	100	0	100
9	U	0.31	0	4	96	100	0	96
10	U	0.31	0	0	100	100	0	100

Fuente: Elaboración propia

4.2.1.7. Caso 7

En la Tabla 4.24 podemos observar que $A = 0$ y $D = 100$, esto es porque el usuario pronunció una palabra distinta a las que se encuentran en la base de datos.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.24 que para cualquier valor de N el valor de $E = 0$, logrando un excelente resultado de un 100 % de efectividad en la respuesta del sistema.

Tabla 4.24: Resultados para el caso 7 con U1 dinámico.

<i>Nº</i>	U1	U2	IC	FI	NI	A	D	E
5	U	0.31	0	0	100	0	100	0
6	U	0.31	0	0	100	0	100	0
7	U	0.31	0	0	100	0	100	0
8	U	0.31	0	0	100	0	100	0
9	U	0.31	0	0	100	0	100	0
10	U	0.31	0	0	100	0	100	0

Fuente: Elaboración propia

4.2.1.8. Caso 8

En la Tabla 4.25 podemos observar que $A = 0$ y $D = 100$, esto es porque el usuario pronunció una palabra distinta a las que se encuentran en la base de datos.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.25 que para cualquier valor de N el valor de $E = 0$, logrando un excelente resultado igual que en el caso anterior de un 100 % de efectividad en la respuesta del sistema.

Tabla 4.25: Resultados para el caso 8 con U1 dinámico.

N°	U1	U2	IC	FI	NI	A	D	E
5	U	0.31	0	0	100	0	100	0
6	U	0.31	0	0	100	0	100	0
7	U	0.31	0	0	100	0	100	0
8	U	0.31	0	0	100	0	100	0
9	U	0.31	0	0	100	0	100	0
10	U	0.31	0	0	100	0	100	0

Fuente: Elaboración propia

De esta primera prueba podemos ver que el mejor valor para N es 5. Esto porque, observamos que para los resultados de los casos 3, 4, 7 y 8 cualquier valor que tome N el valor de $E = 0$, por lo que no tomaremos en cuenta estos casos, entonces de las tablas de los casos 1, 2, 5 y 6 podemos ver que el mejor valor para N es 5, ver la Figura 4.7, si bien los valores de E son grandes, por otro lado, $IC = 0$ por lo que en ningún caso se dejó pasar a intrusos resultando tener un 100 % en seguridad del sistema, ver la Figura 4.8.

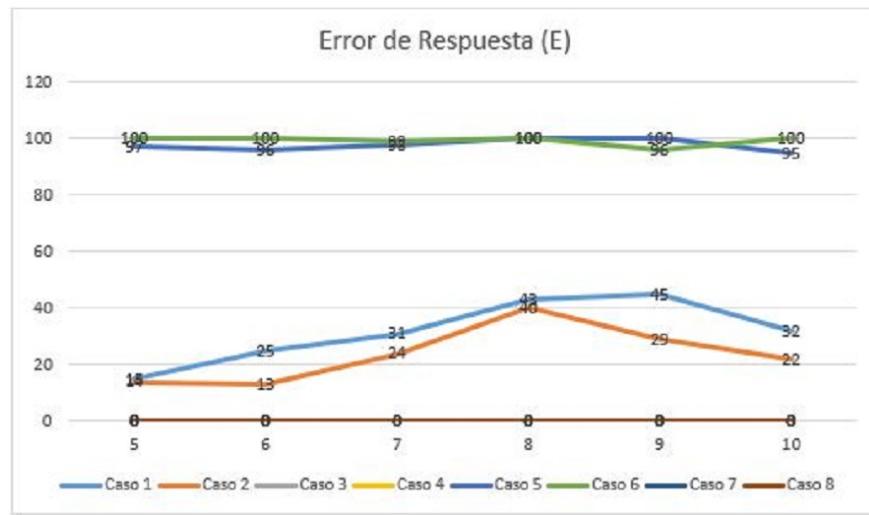


Figura 4.7: Número de errores en la respuesta del sistema para U1 dinámico.

Fuente: Elaboración propia

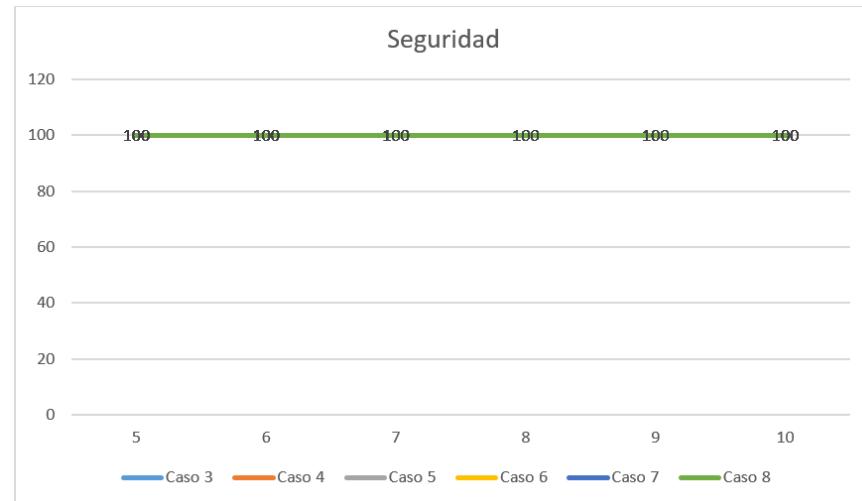


Figura 4.8: Seguridad del sistema para U1 dinámico.

Fuente: Elaboración propia

4.2.2. Prueba con U1 estático

4.2.2.1. Caso 1

En la Tabla 4.26 podemos observar que $A = 100$ y $D = 0$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo el patrón de referencia correcto (es el locutor y es la palabra) para cada audio de prueba, con esto podemos decir que el algoritmo alcanzó un 100 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será IC , podemos ver en la Tabla 4.26 que se obtuvo un menor número de errores de respuesta para $N = 5$ teniendo como resultado a $E = 8$ siendo todos de estos por FI , además este valor es muy alto y no cumple con los requisitos que se propuso en la arquitectura del ciclo de negocio, por lo que este no es un buen resultado.

Tabla 4.26: Resultados para el caso 1 con U1 estático.

N°	U1	U2	IC	FI	NI	A	D	E
5	0.51	0.31	92	8	0	100	0	8
6	0.51	0.31	89	11	0	100	0	11
7	0.51	0.31	90	10	0	100	0	10
8	0.51	0.31	81	19	0	100	0	19
9	0.51	0.31	83	17	0	100	0	17
10	0.51	0.31	91	10	0	100	0	10

Fuente: Elaboración propia

4.2.2.2. Caso 2

En la Tabla 4.27 podemos observar que $A = 100$ y $D = 0$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo el patrón de referencia correcto (es el locutor y es la palabra) para cada audio de prueba, con esto podemos decir que el algoritmo alcanzó un 100 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será IC , podemos ver en la Tabla 4.27 que se obtuvo un menor número de errores de respuesta para $N = 5$ teniendo como resultado a $E = 1$ por lo que el sistema alcanzo un 99 % de efectividad en su respuesta, cumpliendo con los requisitos que se propuso en la arquitectura del ciclo de negocio.

Tabla 4.27: Resultados para el caso 2 con U1 estático.

Nº	U1	U2	IC	FI	NI	A	D	E
5	0.51	0.31	99	1	0	100	0	1
6	0.51	0.31	99	1	0	100	0	1
7	0.51	0.31	98	2	0	100	0	2
8	0.51	0.31	99	1	0	100	0	1
9	0.51	0.31	98	2	0	100	0	2
10	0.51	0.31	99	1	0	100	0	1

Fuente: Elaboración propia

4.2.2.3. Caso 3

En la Tabla 4.28 podemos observar que $A = 0$ y $D = 100$, esto es porque los usuarios pronunciaron una palabra distinta a las que se encuentran en la base de datos.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta

será NI , podemos ver en la Tabla 4.28 que para cualquier valor de N el valor de $E = 0$, logrando un excelente resultado de un 100 % de efectividad en la respuesta del sistema.

Tabla 4.28: Resultados para el caso 3 con U1 estático.

N°	U1	U2	IC	FI	NI	A	D	E
5	0.51	0.31	0	0	100	0	100	0
6	0.51	0.31	0	0	100	0	100	0
7	0.51	0.31	0	0	100	0	100	0
8	0.51	0.31	0	0	100	0	100	0
9	0.51	0.31	0	0	100	0	100	0
10	0.51	0.31	0	0	100	0	100	0

Fuente: Elaboración propia

4.2.2.4. Caso 4

En la Tabla 4.29 podemos observar que $A = 0$ y $D = 100$, esto es porque los usuarios pronunciaron una palabra distinta a las que se encuentran en la base de datos.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.29 que para cualquier valor de N el valor de $E = 0$, logrando un excelente resultado igual que en el caso anterior de un 100 % de efectividad en la respuesta del sistema.

Tabla 4.29: Resultados para el caso 4 con U1 estático.

N°	U1	U2	IC	FI	NI	A	D	E
5	0.51	0.31	0	0	100	0	100	0
6	0.51	0.31	0	0	100	0	100	0
7	0.51	0.31	0	0	100	0	100	0
8	0.51	0.31	0	0	100	0	100	0
9	0.51	0.31	0	0	100	0	100	0
10	0.51	0.31	0	0	100	0	100	0

Fuente: Elaboración propia

4.2.2.5. Caso 5

En la Tabla 4.30 podemos observar que para $N = 8$ se obtuvo un número menor de desaciertos con $D = 7$, esto quiere decir que el algoritmo de reconocimiento de voz obtuvo un 93 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será FI , podemos ver en la Tabla 4.30 que se obtuvo un número menor de errores de respuesta para $N = 6$ dando como resultado a $E = 80$, si bien este valor es muy grande, de estos todos fueron por NI siendo $IC = 0$, por lo que el sistema no dejó pasar a ningún intruso, sin embargo, el valor de E es muy grande y no cumple con los requisitos de la arquitectura del sistema.

Tabla 4.30: Resultados para el caso 5 con U1 estático.

N°	U1	U2	IC	FI	NI	A	D	E
5	0.51	0.31	0	17	83	86	14	83
6	0.51	0.31	0	20	80	88	12	80
7	0.51	0.31	0	19	81	92	8	81
8	0.51	0.31	0	5	95	93	7	95
9	0.51	0.31	0	8	92	83	17	92
10	0.51	0.31	0	16	84	83	17	84

Fuente: Elaboración propia

4.2.2.6. Caso 6

En la Tabla 4.31 podemos observar que $A = 100$ y $D = 0$, esto quiere decir que el algoritmo de reconocimiento de voz alcanzó un 100 % de efectividad en la etapa de reconocimiento.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta

será FI , podemos ver en la Tabla 4.31 que se obtuvo un número menor de errores de respuesta para $N = 5$ con $E = 6$, siendo todos estos por NI siendo $IC = 0$, por lo que el sistema no dejó pasar a ningún intruso, sin embargo, el valor de E es un valor casi aceptable y puede cumplir con los requisitos de la arquitectura del sistema.

Tabla 4.31: Resultados para el caso 6 con U1 estático.

N°	U1	U2	IC	FI	NI	A	D	E
5	0.51	0.31	0	94	6	100	0	6
6	0.51	0.31	0	76	24	100	0	24
7	0.51	0.31	0	82	18	100	0	18
8	0.51	0.31	0	56	44	100	0	44
9	0.51	0.31	0	61	39	100	0	39
10	0.51	0.31	0	65	35	100	0	35

Fuente: Elaboración propia

4.2.2.7. Caso 7

En la Tabla 4.32 podemos observar que $A = 0$ y $D = 100$, esto es porque el usuario pronunció una palabra distinta a las que se encuentran en la base de datos.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.32 que para cualquier valor de N el valor de $E = 0$, logrando un excelente resultado de un 100 % de efectividad en la respuesta del sistema.

Tabla 4.32: Resultados para el caso 7 con U1 estático.

N°	U1	U2	IC	FI	NI	A	D	E
5	0.51	0.31	0	0	100	0	100	0
6	0.51	0.31	0	0	100	0	100	0
7	0.51	0.31	0	0	100	0	100	0
8	0.51	0.31	0	0	100	0	100	0
9	0.51	0.31	0	0	100	0	100	0
10	0.51	0.31	0	0	100	0	100	0

Fuente: Elaboración propia

4.2.2.8. Caso 8

En la Tabla 4.33 podemos observar que $A = 0$ y $D = 100$, esto es porque el usuario pronunció una palabra distinta a las que se encuentran en la base de datos.

Ahora en cuanto a la respuesta de decisión del sistema para este caso la única respuesta correcta será NI , podemos ver en la Tabla 4.33 que para $N = 9$ el valor de $E = 0$, logrando un excelente resultado de un 100 % de efectividad en la respuesta del sistema.

Tabla 4.33: Resultados para el caso 8 con U1 estático.

N°	U1	U2	IC	FI	NI	A	D	E
5	0.51	0.31	0	2	98	0	100	2
6	0.51	0.31	0	1	99	0	100	1
7	0.51	0.31	0	1	99	0	100	1
8	0.51	0.31	0	1	99	0	100	1
9	0.51	0.31	0	0	100	0	100	0
10	0.51	0.31	0	1	99	0	100	1

Fuente: Elaboración propia

De esta primera prueba podemos ver que el mejor valor para N es 5. Esto porque, observamos que para los resultados de los casos 3, 4 y 7 cualquier valor que tome N el valor de $E = 0$, por lo

que no tomaremos en cuenta estos casos, entonces de las tablas de los casos 1, 2, 5, 6 y 8 podemos ver que el mejor valor para N es 5, ver la Figura 4.9, si bien los valores de E son grandes, por otro lado, $IC = 0$ por lo que en ningún caso se dejó pasar a intrusos resultando tener un 100 % en seguridad del sistema, ver la Figura 4.10.

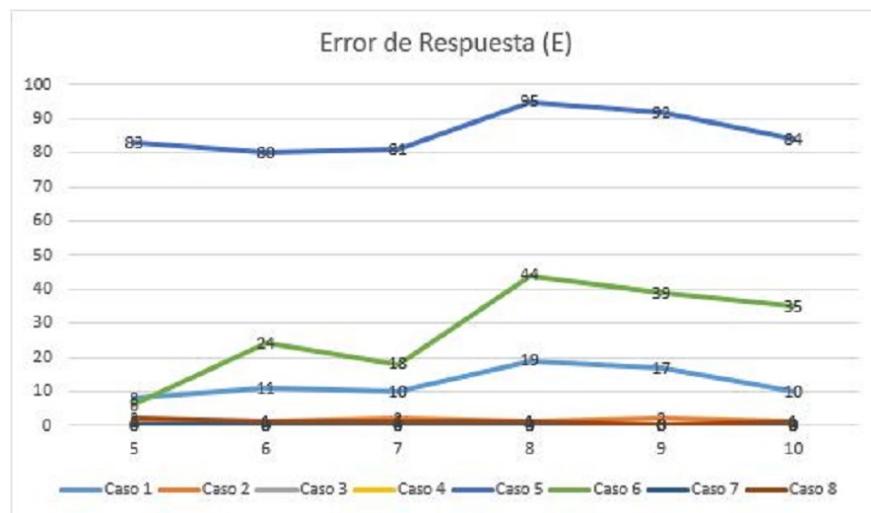


Figura 4.9: Número de errores en la respuesta del sistema para U1 estático.

Fuente: Elaboración propia

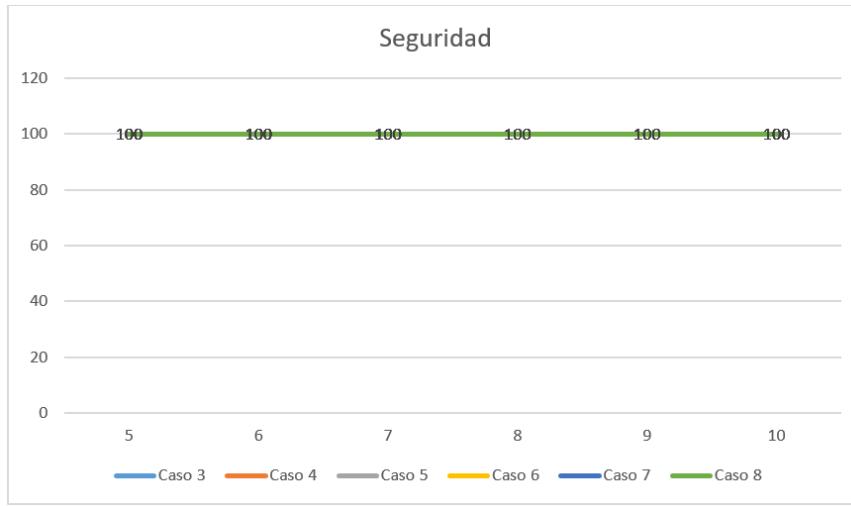


Figura 4.10: Seguridad del sistema para U1 estático.

Fuente: Elaboración propia

Después de realizar todas las pruebas para poder hallar el mejor valor para N (número de patrones para la construcción del patrón de referencia) podemos ver que se obtuvieron mejores resultados para $N = 5$, además que sigue siendo el mejor valor para $U1 = 0,51$ y no el valor definido por la Ecuación (2.101), por lo que queda totalmente comprado que este es el mejor valor.

Si bien con $N = 5$ disminuimos en la quinta parte el tiempo de ejecución en la respuesta para el reconocimiento del locutor en el sistema, todavía no hemos comparado si este tiene menos errores que $N = 1$, a continuación, en la Tabla 4.34 veremos la cantidad de errores que se obtuvieron para estos dos casos.

Tabla 4.34: Comparación de numero de errores entre $U_1 = U$ y $U_1 = 0.51$.

N°	U_1	U_2	$N=1$	$N=5$
1	0.51	0.31	5	8
2	0.51	0.31	1	1
3	0.51	0.31	0	0
4	0.51	0.31	1	0
5	0.51	0.31	72	83
6	0.51	0.31	2	6
7	0.51	0.31	0	0
8	0.51	0.31	3	2
Total de Error		84	100	

Fuente: Elaboración propia

De la Tabla 4.34 podemos ver que para $N = 1$ se tiene un *total de error* de 84 de un total de 800 pruebas haciendo una tasa de error del 10.5 %, mientras que para $N = 5$ se tiene un *total de error* de 100 haciendo una tasa de error del 12.5 %, por lo que se escogerá a $N = 1$ lo que quiere decir que no se aplicará el algoritmo de construcción de patrón de referencia definido por la Ecuación (2.101).

Si bien con $N = 5$ disminuiríamos el tiempo de respuesta del reconocimiento del locutor del sistema, este presenta un problema para el caso 1 debido a que se tiene $E = 8$ resultando un 92 % de efectividad en la respuesta del sistema incumpliendo con los requisitos definidos para la arquitectura del sistema en el ciclo de negocio, el cual se estableció como mínimo un 95 % en la tasa de acierto, requisito que si cumple para $N = 1$.

4.2.3. Prueba de hipótesis o de significación

Como se dijo anteriormente para efectuar el análisis estadístico de los datos se hará la prueba de independencia Chi- Cuadrado de Pearson para dos variables nominales, ver en Apéndice A.

1. La Tabla 4.35 muestra los resultados obtenidos del experimento con $U1 = 0,51$ y $U2 = 0,31$, estos datos son las frecuencias observadas (FO), elaborando así la tabla de contingencia.

Tabla 4.35: Tabla de contingencia o de frecuencias observadas.

CA/RV	IC	FI	NI	TOTAL
C1	95	5	0	100
C2	99	1	0	100
C3	0	0	100	100
C4	0	1	99	100
C5	0	28	72	100
C6	0	98	2	100
C7	0	0	100	100
C8	0	3	97	100
TOTAL	194	136	470	800

Fuente: Elaboración propia

2. La Tabla 4.36 muestra el cálculo de la frecuencia esperada (FE).

Tabla 4.36: Tabla de frecuencias esperadas.

CA/RV	IC	FI	NI	TOTAL
C1	24.25	17	58.75	100
C2	24.25	17	58.75	100
C3	24.25	17	58.75	100
C4	24.25	17	58.75	100
C5	24.25	17	58.75	100
C6	24.25	17	58.75	100
C7	24.25	17	58.75	100
C8	24.25	17	58.75	100
TOTAL	194	136	470	800

Fuente: Elaboración propia

3. Calcular los grados de libertad.

$$\alpha = 0,05 \text{ (nivel de confianza del } 5\%)$$

$$gl = (3 - 1)(8 - 1) = 14$$

$$X^2_{cri} = 23,6848$$

4. Plantear las hipótesis.

H_0 : La respuesta del reconocimiento de voz es independiente del caso de control de acceso.

H_1 : La respuesta del reconocimiento de voz depende del caso de control de acceso.

5. Construcción de las áreas de aceptación y rechazo.

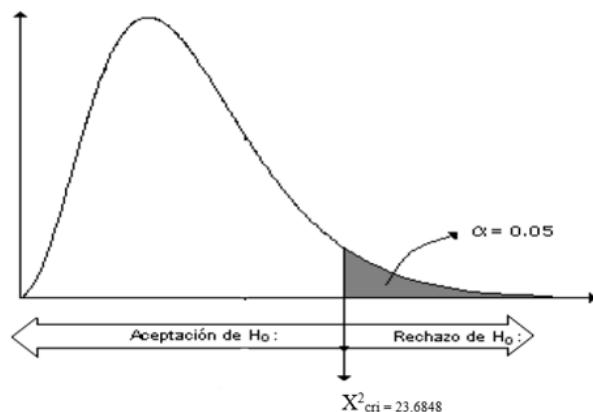


Figura 4.11: Área de aceptación y rechazo de H_0 .

Fuente: Elaboración propia

6. La Tabla 4.37 muestra los valores de la diferencia cuadrada de la frecuencia observada con

la esperada y luego dividida entre la frecuencia esperada, teniendo todo esto pasaremos a

calcular la ji-Cuadrada que no es más que la raíz cuadrada de la sumatoria de estos valores.

Tabla 4.37: Tabla con los valores ji-cuadrada.

CA/RV	IC	FI	NI
C1	206.4149485	8.47058824	58.75
C2	230.4149485	15.0588235	58.75
C3	24.25	17	28.962766
C4	24.25	15.0588235	27.2755319
C5	24.25	7.11764706	2.98829787
C6	24.25	385.941176	54.8180851
C7	24.25	17	28.962766
C8	24.25	11.5294118	24.9031915

Fuente: Elaboración propia

7. Finalmente se obtuvo un $X_{exp}^2 = 36,6772$, como podemos ver este valor es mayor al valor de $X_{cri}^2 = 23,6848$, por lo que la hipótesis nula H0 se rechaza. Entonces podemos decir que con un nivel de confianza del 5 % se encontró que la respuesta del reconocimiento de voz depende del caso de control de acceso. Por lo que se contrasta la hipótesis planteada durante el proceso de elaboración del plan de investigación, es decir, que se logró demostrar la relación entre las variables de estudio formuladas en la investigación, por lo que podemos decir que el acceso a una vivienda puede ser controlado por reconocimiento de voz del locutor dependiente del texto utilizando MFCC Y DTW.

Capítulo 5

Consideraciones finales

En esta tesis se desarrolló un sistema de seguridad por reconocimiento de voz en una plataforma de simulación, basado en la extracción de características de una palabra hablada. Se emplearon los algoritmos MFCC para construir los patrones de voz y DTW para la comparación entre ellos. El sistema es dependiente del texto y se hicieron experimentos para un sistema de conjunto cerrado y para un sistema de conjunto abierto. Se hicieron pruebas variando los valores de los parámetros de los algoritmos empleados con el fin de generar el porcentaje más alto de tasa de acierto.

5.1. Conclusiones

1. Debido a que un sistema de seguridad biométrico se divide en hardware y software, el uso de la metodología propuesta por Tammy Noergaard en (Noergaard, 2012), para el diseño de sistemas embebidos permitió seguir un orden específico entre ellos al momento de implementar los diferentes módulos necesarios para el software y hardware.

2. Se observó que el sistema funciona mejor con una frecuencia de muestreo de 16000 Hz en las grabaciones, si bien en la mayoría de reconocedores del habla se emplea una frecuencia de 8000 Hz, en nuestro caso no fue así, esto porque aparte de reconocer la palabra es necesario reconocer al locutor, por lo que se requiere más información de la señal de voz. Además de esta configuración se tuvo las siguientes características de grabación: un solo canal (mono), una resolución 16 bits y un ordenamiento de bits Little Endian.
3. En la etapa de entrenamiento se empleó el algoritmo MFCC con el que se obtuvo una mayor tasa de acierto en el reconocimiento de voz, en lugar de los MFCC delta y doble delta, ver Tablas del 3.2 al 3.9.
4. En la etapa de reconocimiento se empleó el algoritmo DTW simétrico en lugar del asimétrico, esto porque de la forma simétrica se obtiene una mayor información de la matriz de distancias permitiendo escoger la distancia con menor peso, ver Tablas del 3.13 al 3.16.
5. Para la eliminación del ruido se empleó el algoritmo NLMS, esto porque este usa un tamaño de paso de adaptación dinámico lo que permite que la señal se filtre de una mejor manera sin distorsionarla, sin embargo, este algoritmo tiene una limitación y es que funciona correctamente en ambientes con un ruido no máximo de 80 dBC.
6. Para la detección de inicio y fin de la señal de voz se usó el algoritmo de Rabiner & Sambur propuesto en (UNAM, 2015), esto porque el algoritmo definido en (Rabiner and Sambur,

1975) se requiere que la frecuencia de muestreo sea de 10 KHz para su correcto funcionamiento, sin embargo como se dijo anteriormente se optó por una frecuencia de muestreo de 16 KHz, es por esto que se obtuvo mejores resultados con el primer algoritmo, ver Figuras 3.75 y 3.76.

7. Luego de experimentar el uso de diversos algoritmos así como la variación de los parámetros de sus funciones (Sección 3.3), podemos decir que la seguridad por voz irá en aumento al paso de las diferentes etapas del sistema como filtrado, detección de inicio y fin de la señal de voz, normalización, obtención de parámetros, etc. tecnologías que vallan mejorando la confiabilidad y velocidad en la toma de decisión, estas a su vez ayudarán en la implementación de sistemas de seguridad robustos y eficientes.
8. En un sistema de seguridad por reconocimiento de voz, la toma de decisión ante un reconocimiento de un locutor, puede ser dificultoso debido a varios factores, como el nivel de ruido donde se hacen las grabaciones, la similitud de las palabras y los diferentes estados de ánimo de la persona, por ejemplo, si el usuario dice la palabra en un tono de extrema alegría o tristeza, o si está en un estado alcohólico o inconveniente, esto modificará las características de su voz, provocando una disminución en la probabilidad de reconocerlo.

Para corregir este problema se podría proponer tener más bases de datos del mismo usuario, pero en diferentes estados de ánimo, así el usuario antes de identificarse podría indicar

su estado de ánimo y el programa relacionaría su grabación en el momento, pero esto podría facilitar en gran medida el acceso a otros usuarios es por esa razón por la cual no se implementó esta solución, y se optó por que el usuario grabara de forma correcta.

9. Al utilizar umbrales ajustados se obtuvo un porcentaje de rechazo de intrusos del 100 %, ver Tablas 4.13 y 4.14. Sin embargo, existe un compromiso entre seguridad y la tasa de identificación, es decir, entre más seguro sea el sistema a intrusos, más rechazos de usuarios válidos se tendrán y por lo tanto la tasa de identificación disminuirá, ver Tablas 4.1 y 4.2.
10. Todo sistema biométrico o sistema de seguridad para el control de acceso su tiempo de respuesta no debe de ser mayor a 5 segundos, sin embargo observamos que para nuestro sistema con una cantidad mayor a 60 patrones característicos (patrón MFCC) esto ya no se cumple ver Figuras 4.5 y 4.6, debido a que el sistema toma un tiempo para poder reajustar los umbrales de decisión, además al aumentar el número de usuarios en el sistema, la similitud entre cada uno de ellos será mayor, causando que sus coeficientes MFCC podrían converger en un punto provocando una tasa de acierto menor en el reconocimiento, para evitar esto se tomó umbrales con valores estáticos $U1 = 0,51$ y $U2 = 0,31$.
11. Se implementó el algoritmo de construcción de patrón de referencia, que consiste en obtener un patrón característico a partir de un conjunto de patrones de un usuario, reduciendo así el número de comparaciones que realiza el sistema para reconocer la voz del locutor, sin

embargo no se empleo esta técnica en nuestro sistema debido a que se obtuvo una tasa de error del 8 % en el reconocimiento de los usuarios del sistema, ver Tabla 4.34.

12. Nuestro algoritmo contribuye en el recocimiento de voz del locutor con una tasa de acierto del 95 % para el caso de un sistema de conjunto cerrado y una tasa de acierto del 100 % ante intrusos en el caso de sistema de conjunto abierto, mucho mejor que los resultados de 79.63 % y 92.6 % para sistemas abierto y cerrado respectivamente obtenido por Pérez et al. (2013) y mucho más proximo y mejor al resultado de 94.44 % y 98.06 % obtenido en la investigación hecha por Varela (1994).
13. En conclusión, el objetivo general que trata sobre implementar un sistema de seguridad para el control de acceso a una vivienda que funcione en base al reconocimiento de voz que sea dependiente del texto fue conseguido a través del algortimo conformado por los métodos MFCC Y DTW.

5.2. Trabajos futuros

1. Se aprendió que para realizar un reconocedor de voz, se necesitan muchas partes tanto software como hardware que realicen cada una de sus funciones de manera adecuada, siendo el software el de mayor importancia, gracias a este se pueden programar los algoritmos para poder reconocer al locutor, pero el hardware también es importante debido a que al usar un micrófono mejor, se puede obtener una mejor calidad en la señal capturada y por lo tanto,

dejar que el software se encargue sólo del procesamiento de la señal y no tenga que desperdiciar tiempo en limpiarla, filtrarla y normalizarla.

2. El ruido externo puede ser un gran problema para el sistema al momento de realizar el reconocimiento de voz, debido a que el algoritmo que se usó funciona correctamente con niveles de ruido menores a 80 dBC lo que resulta una limitante, ante este problema una alternativa como se dijo anteriormente es en usar un mejor micrófono, pero puede que su costo sea muy elevado, por lo que se recomienda estudiar y evaluar los otros algoritmos adaptativos de eliminación de ruido que vimos en la teoría, ver Tabla 2.4.
3. Al igual que el ruido externo el eco es otro problema que ocasiona que el reconocimiento de voz no se realice correctamente, por lo que es necesario implementar un algoritmo de eliminación de eco o usar un micrófono que filtre el eco.
4. Tal como se dijo el hardware de un sistema de reconocimiento de voz es muy importante para su correcto funcionamiento, y es que como vimos en las pruebas que se realizaron en un dispositivo móvil con un procesador de 4x1.8 GHz, el tiempo de ejecución presentó una limitante y es que a partir de una base de datos mayor a 60 patrones característicos el tiempo de respuesta es mayor a 5 segundos, debido al gran número de usuarios o al número de patrones que cada uno de estos tienen, incumpliendo así uno de los requisitos principales de un sistema biométrico de tiempo real. Por lo que se recomienda implementar el algoritmo de

reconocimiento de voz en un servidor, esto porque una computadora tiene mayor velocidad de procesamiento que el de un dispositivo móvil, reduciendo así el tiempo de respuesta del sistema.

5. El sistema no soporta el reconocimiento de voz continuo solo palabras aisladas, es decir el usuario solo puede pronunciar una palabra para el reconocimiento de voz y no un conjunto de ellas (frase), por lo que se podría implementar un algoritmo para este caso. Existe una variante del algoritmo de Rabiner & Sambur el cual usa la detección de pulsos de energía, por lo que se recomienda estudiarla.
6. En este sistema se pueden producir tasas de aceptación altas si todos los usuarios son muy cooperativos, es decir, si la forma de pronunciación del locutor es muy parecida a las palabras habladas, sin embargo, en un sistema real esto no sucede. Por lo tanto, se recomienda hacer un estudio de los algoritmos basados en Redes Neuronales, el uso de probabilidades como son los Modelos Ocultos de Markov, Redes Bayesianas, etc. y cómo aplicarlos al reconocimiento de voz del locutor.
7. Puede que existan usuarios o intrusos con las voces similares, esto podría ser un problema para nuestro sistema de seguridad por reconocimiento de voz, es por ello que se recomienda integrar a nuestro sistema otro tipo de sistema biométrico como, por ejemplo, el reconocimiento facial, donde podríamos usar la cámara del dispositivo móvil.

Referencias bibliográficas

ALTUMTEC (2018). ¿qué es un sistema de control de acceso? <https://altumtec.com/que-es-un-sistema-control-de-acceso/>.

Becchetti, C. and Prina, L. (1999). *Speech Recognition: Theory and C++ Implementation*. Wiley, Canada.

Cook, S. (2002). Speech recognition howto. <http://www.tldp.org/HOWTO/Speech-Recognition-HOWTO/inside.html>.

Cosentino, L. (2016). Control de accesos. negocios de seguridad. 49:172–184.

Cruz, L. (2009). Aplicación del reconocimiento de voz de un hablante mediante una red neuronal artificial backpropagation y coeficientes lpc sobre un canal telefónico. 1:111–129.

De Luna, O., Martínez, R., and Mora, G. (2006). *Reconocimiento de Voz con Redes Neuronales, DTW y Modelos Ocultos de Markov*. redalyc.org, México.

Ebla, H. and Inca, P. (2016). *Diseño e implementación de un filtro adaptativo mediante filtros*

Wiener para la cancelación de ruido en señales de audio. Facultad de Ingeniería, Universidad Nacional de Chimborazo, Ecuador.

Fermax (2016). Sistemas de identificación en el control de accesos: ¿cuál elegir? <https://blog.fermax.com/esp/sistemas-de-identificacion-en-el-control-de-accesos-cual-elegir>.

Franco, J. (2010). *Reconocimiento de Voz para Niños con Discapacidad en el Habla.* Ingeniería en Electrónica y Computadoras, Universidad de las Américas Puebla, México.

Furui, S. (1981). Cepstral analysis technique for automatic speaker verification. 1:254–272.

Guevara, J. and Salazar, J. (2007). *Extracción de Características en el Procesamiento Digital de una Señal para el Mejoramiento del Reconocimiento Automático del Habla usando Wavelets.* Escuela de Informática, Facultad de Ciencias Físicas Y Matemáticas, Universidad Nacional de Trujillo, Perú.

Harrington, J. (1999). *Techniques in Speech Acoustics.* Kluwer Academic Publishers, Canada.

Huang, X., Acero, A., and Hon, H. (2001). *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development.* Carnegie Mellon University, EE.UU.

Jelinek, F. (2004). *Computer Speech: Recognition, Compression, Synthesis.* Springer Science & Business Media, Germany.

Lee, K. (1989). *Automatic Speech Recognition*. Springer, New York.

Llisterri, J. (2018). Las características acústicas de los sonidos del habla. http://liceu.uab.es/~joaquim/phonetics/fon_anal_acus/fon_acust.html.

Mathematics (2015). Normalisation of a range of numbers to another range. <https://math.stackexchange.com/questions/1554078/normalisation-of-a-range-of-numbers-to-another-range>.

Myers, C., Rabiner, L., and Rosenberg, A. (1980). Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. 1:623–635.

Navarrete, O. (2013). *Sistema Automatizado de Iluminación de una Casa mediante Comandos de Voz*. Universidad Nacional Autónoma de México, México.

Noergaard, T. (2012). *Embedded Systems Architecture*. Newnes, EE.UU.

Owens, F. (1993). *Signal processing of speech*. McGraw-Hill, New York.

Pérez, E., Poceros, F., and Villalobos, J. (2013). *Sistema de Seguridad por Reconocimiento de Voz*. Escuela Superior de Ingeniería Mecánica y Eléctrica, Instituto Politécnico Nacional, México.

Rabiner, L., Rosenberg, A., and Levinson, S. (1978). Considerations in dynamic time warping algorithms for discrete word recognition. 1:575–582.

Rabiner, L. and Sambur, M. (1975). An algorithm for determining the endpoints of isolated utterances. 54:297–315.

Rama, A. (2007). *Procesamiento Digital de la Señal Sonora Utilizando Matlab*. Lulu.com, México.

Rowden, C. (1992). *Speech Processing*. McGraw-Hill, EE.UU.

Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. 1:43–49.

Sanchez, H. (2015). Ventajas y desventajas de los controles biométricos.
<http://controlbiometricodigital.blogspot.com/2015/07/ventajas-y-desventajas-de-los-controles.html>.

Schroeder, M. (1998). *Statistical Methods for Speech Recognition (Language, Speech, and Communication)*. University Press Group Ltd, EE.UU.

Shubhra, D. (2017). Lms adaptive filters for noise cancellation. 7:2520–2529.

Simón, G. (2011). *Filtro adaptivo LMS y su aplicación en el reconocimiento de palabras aisladas para el control de un equipo de sonido por medio de la voz*. Facultad de Ciencias e Ingeniería, Universidad Pontificia Católica del Perú, Perú.

Toast, T. (2018). La historia de los sistemas de reconocimiento de voz. <https://www.timetoast.com/timelines/la-historia-de-los-sistemas-de-reconocimiento-de-voz>.

UNAM (2015). Laboratorio de procesamiento digital de voz, detección de inicio y fin de palabra. <http://profesores.fi-b.unam.mx/procvoz/Practicas/Practica03.pdf>.

Varela, H. (1994). *Reconocimiento Automático de Locutor y Realización de un Sistema Experimental*. CICESE, Tesis de Maestría.

Velásquez, G. (2008). *Sistema de Reconocimiento de Voz en Matlab*. Escuela de Ingeniería Mecánica Eléctrica, Facultad de Ingeniería, Universidad de San Carlos de Guatemala, Guatemala.

Wikipedia (2016). Formante. <https://es.wikipedia.org/wiki/Formante>.

Zelaya, W. (2004). *Diseño de un filtro digital adaptativo como cancelador de ruido basado en el algoritmo LMS*. Facultad de Ingeniería y Arquitectura, Universidad del Salvador, El Salvador.

Apéndice A

Primer apéndice

A.1. Prueba de independencia Chi-Cuadrado de Pearson

La prueba de independencia Chi-Cuadrado o ji-Cuadrado, nos permite determinar si existe una relación entre dos variables categóricas. Es necesario resaltar que esta prueba nos indica si existe o no una relación entre las variables, pero no indica el grado o el tipo de relación; es decir, no indica el porcentaje de influencia de una variable sobre la otra o la variable que causa la influencia.

Para comprender mejor este tema es necesario recordar cuales son los eventos independientes y cuales los dependientes.

“Dos eventos aleatorios, A y B , son eventos independientes si la probabilidad de un evento no está afectada por la ocurrencia del otro evento; por lo tanto, $p(A) = p(A | B)$.”

“Dos eventos aleatorios, A y B , son eventos dependientes si la probabilidad de un evento está

afectada por la ocurrencia del otro; por lo tanto, $p(A) \neq p(A | B)$.”

Una prueba de independencia usa la pregunta de si la ocurrencia del evento X es independiente a la ocurrencia del evento Y , por lo que el planteamiento de las hipótesis para esta prueba de independencia es:

- H_0 : La ocurrencia del evento X es independiente del evento Y .
- H_1 : La ocurrencia del evento X no es independiente del evento Y .

En las pruebas de independencia se utiliza el formato de la tabla de contingencia, y por esa razón a veces se le llama *prueba de tabla de contingencia*, o *prueba con tabla de contingencia*.

Una tabla que clasifica datos de acuerdo a dos o más categorías, relacionados con cada una de las variables cualitativas, que pueden ser o no estadísticamente independientes, se llama *tabla de contingencia*. Dicha tabla muestra todas las posibles combinaciones de categorías, o contingencias, que explican su nombre.

A la suma de todas las razones que se puedan construir al tomar la diferencia entre cada frecuencia observada y esperada, en una tabla de contingencia, elevándola al cuadrado, y luego dividiendo esta desviación cuadrada entre la frecuencia esperada, se le llama *estadístico ji-cuadrada*.

Procedimiento para elaborar una prueba de independencia:

1. Obtener la frecuencia observada (FO), proveniente de una encuesta, estudio o experimento.
2. Resumir los datos obtenidos, es decir, la frecuencia observada, en una tabla de contingencia
3. Calcular la frecuencia esperada (FE), y se calcula con la siguiente formula:

$$FE = \frac{(TotalColumna)(TotalRenglon)}{GranTotal}$$

4. Determinar el nivel de significancia (α), y los grados de libertad, con la siguiente formula:

$$gl = (\#renglones)(\#columnas)$$

5. Plantear las hipótesis.

$$H_0 : independencia$$

$$H_1 : dependencia$$

6. Construir las áreas de aceptación y rechazo.

7. Calcular ji-Cuadrada X^2

$$X_{exp}^2 = \sum_{i=1}^n \frac{(FO - FE)^2}{FE}$$

8. Tomar una decisión y emitir una conclusión en términos del problema.

A continuación, mostraremos algunos valores de la Tabla de Distribución Chi Cuadrado, donde v representa a los grados de libertad (gl) y p la probabilidad de encontrar un valor mayor o igual que el Chi-Cuadrado tabulado, también llamado como nivel de significancia (α).

Tabla A.1: Tabla de distribución Chi-Cuadrado.

v/p	0,001	0,0025	0,005	0,01	0,025	0,05	0,1	0,15	0,2	0,25	0,3	0,35	0,4	0,45	0,5
1	10,3274	9,1404	7,8794	6,6349	5,0239	3,8415	2,7055	2,0722	1,6424	1,3233	1,0742	0,8735	0,7083	0,5707	0,4549
2	13,8150	11,9827	10,5965	9,2104	7,3778	5,9915	4,6052	3,7942	3,2189	2,7726	2,4079	2,0996	1,8326	1,5970	1,3863
3	16,2660	14,3202	12,8381	11,3449	9,3484	7,8147	6,2514	5,3170	4,6416	4,1083	3,6649	3,2831	2,9462	2,6430	2,3660
4	18,4662	16,4238	14,8602	13,2767	11,1433	9,4877	7,7794	6,7449	5,9886	5,3853	4,8784	4,4377	4,0446	3,6871	3,3567
5	20,5147	18,3854	16,7496	15,0863	12,8325	11,0705	9,2363	8,1152	7,2893	6,6257	6,0644	5,5731	5,1319	4,7278	4,3515
6	22,4575	20,2491	18,5475	16,8119	14,4494	12,5916	10,6446	9,4461	8,5581	7,8408	7,2331	6,6948	6,2108	5,7652	5,3481
7	24,3213	22,9402	20,2777	18,4753	16,0128	14,0671	12,0170	10,7479	9,8032	9,0371	8,3834	7,8061	7,2832	6,8000	6,3458
8	26,1239	23,7742	21,9549	20,0902	17,5345	15,5073	13,3616	12,0271	11,0301	10,2189	9,5245	8,9094	8,3505	7,8325	7,3441
9	27,8767	25,4625	23,5893	21,6660	19,0228	16,9190	14,6837	13,2880	12,2421	11,3887	10,6564	10,0060	9,4136	8,8632	8,3428
10	29,5879	27,1119	25,1881	23,2093	20,4832	18,3070	15,9872	14,5339	13,4420	12,5489	11,7807	11,0971	10,4732	9,8922	9,4418
11	31,2635	28,7291	26,7569	24,7250	21,9200	19,6752	17,2750	15,7671	14,6314	13,7007	12,8987	12,1836	11,5298	10,9199	10,3410
12	32,9092	30,3182	28,2997	26,2170	23,3367	21,0261	18,5493	16,9893	15,8120	14,8454	14,0111	13,2661	12,5838	11,9463	11,3403
13	34,5274	31,8830	29,8193	27,6882	24,7356	22,3620	19,8119	18,2020	16,9848	15,9839	15,1187	14,3451	13,6356	12,9717	12,3398
14	36,1239	33,4262	31,3194	29,1412	26,1189	23,6848	21,0641	19,4062	18,1508	17,1169	16,2221	15,4209	14,6853	13,9961	13,3393
15	37,6978	34,9494	32,8015	30,5780	27,4884	24,9958	22,3071	20,6030	19,3107	18,2451	17,3217	16,4940	15,7332	15,0197	14,3389
16	39,2518	36,4555	34,2671	31,9999	28,8453	26,2962	23,5418	21,7931	20,4651	19,3689	18,4179	17,5646	16,7795	16,0425	15,3385
17	40,7911	37,9462	35,7184	33,4087	30,1910	27,5871	24,7690	22,9770	21,6146	20,4887	19,5110	18,6330	17,8244	17,0646	16,3382
18	42,3119	39,4220	37,1564	34,8052	31,5264	28,8693	25,9894	24,1555	22,7595	21,6049	20,6014	19,6993	18,8079	18,0860	17,3379
19	43,8194	40,8847	38,5821	36,1908	32,8523	30,1435	27,2036	25,3289	23,9004	22,7178	21,6891	20,7638	19,9102	19,1069	18,3376
20	45,3142	42,3358	39,9969	37,5663	34,1696	31,4104	28,4120	26,4976	25,0375	23,8277	22,7745	21,8265	20,9514	20,1272	19,3374
21	46,7963	43,7749	41,4069	38,9322	35,4789	32,6706	29,6151	27,6620	26,1711	24,9348	23,8578	22,8876	21,9915	21,1470	20,3372
22	48,2676	45,2041	42,7957	40,2894	36,7807	33,9245	30,8133	28,8224	27,3015	26,0393	24,9390	23,9473	23,0307	22,1663	21,3370
23	49,7276	46,6231	44,1814	41,6383	38,0756	35,1725	32,0069	29,9792	28,4288	27,1413	26,0184	25,0055	24,0689	23,1852	22,3369
24	51,1790	48,0336	45,5584	42,9798	39,3641	36,4150	33,1962	31,1325	29,5533	28,2412	27,0960	26,0625	25,1064	24,2037	23,3367
25	52,6187	49,4351	46,9280	44,3140	40,6465	37,6525	34,3816	32,2825	30,6752	29,3388	28,1719	27,1183	26,1430	25,2218	24,3366
26	54,0511	50,8291	48,2898	45,6416	41,9231	38,8851	35,5632	33,4295	31,7946	30,4346	29,2463	28,1730	27,1789	26,2395	25,3365
27	55,4751	52,2152	49,6450	46,9628	43,1945	40,1133	36,7412	34,5736	32,9117	31,5284	30,3193	29,2266	28,2141	27,2569	26,3363
28	56,8918	53,5939	50,9936	48,2782	44,4608	41,3372	37,9159	35,7150	34,0266	32,6205	31,3909	30,2791	29,2486	28,2740	27,3362
29	58,3006	54,9662	52,3355	49,5878	45,7223	42,5569	39,0875	36,8538	35,1394	33,7109	32,4612	31,3308	30,2825	29,2908	28,3361

Fuente: Pérez et al. (2013)



UNIVERSIDAD NACIONAL DE TRUJILLO

DECLARACION JURADA RR-384-2018/UNT

Los autores suscritos en el presente documento DECLARAMOS BAJO JURAMENTO que somos los responsables legales de la calidad y originalidad del contenido del Proyecto de Investigación Científica, así como, del Informe de Investigación Científica realizado.

Titulado: _____

PROYECTO DE INVESTIGACIÓN CIENTÍFICA

PROY DE TRABAJO DE INVESTIGACIÓN (PREGRADO) () TRABAJO DE INVESTIGACIÓN (PREGRADO) ()
PROYECTO DE TESIS PREGRADO () TESIS PREGRADO ()
PROYECTO DE TESIS MAESTRÍA () TESIS MAESTRIA ()
PROYECTO DE TESIS DOCTORADO () TESIS DOCTORADO ()

INFORME FINAL DE INVESTIGACIÓN CIENTÍFICA

El equipo investigador Integrado por:

Nº	APELLIDOS Y NOMBRES	FACULTAD/DEPARTAMENTO	CATEGORIA DOCENTE ASESOR	CÓDIGO Docente Número Matricula del estudiante	Autor Coautor Asesor

Trujillo, ... de de

.....
FIRMA

.....
DNI

.....
FIRMA

.....
DNI

.....
FIRMA

.....
DNI



UNIVERSIDAD NACIONAL DE TRUJILLO

CARTA DE AUTORIZACIÓN DE PUBLICACIÓN DE TRABAJO DE INVESTIGACIÓN EN REPOSITORIO DIGITAL RENATI-SUNEDU RR-384-2018/UNT

Trujillo, ... de de

Los autores suscritos del INFORME FINAL DE INVESTIGACIÓN CIENTIFICA

Titulado: _____

AUTORIZAMOS SU PUBLICACIÓN EN EL REPOSITORIO DIGITAL INSTITUCIONAL, REPOSITORIO RENATI-SUNEDU, ALICIA-CONCYTEC CON EL SIGUIENTE TIPO DE ACCESO:

- A. Acceso Abierto:
- B. Acceso Restringido (datos del autor y resumen del trabajo)
- C. No autorizo su Publicación

Si eligió la opción restringido o NO autoriza su publicación sírvase justificar:

ESTUDIANTES DE PREGRADO: TRABAJO DE INVESTIGACIÓN

TESIS

ESTUDIANTES DE POSGRADO : TESIS MAESTRÍA

TESIS DOCTORADO

DOCENTES:

INFORME DE INVESTIGACIÓN

El equipo investigador Integrado por:

Nº	APELLIDOS Y NOMBRES	FACULTAD	CONDICIÓN DOCENTE (NOMBRADO, CONTRATADO, EMÉRITO, estudiante, OTROS)	CÓDIGO Docente Número Matrícula del estudiante	Autor Coautor Asesor

.....
FIRMA

.....
DNI

.....
FIRMA

.....
DNI

.....
FIRMA

.....
DNI