**TAIVoiceMonitor - Componente de Detección de Voz Inteligente**

**¿Qué es TAIVoiceMonitor?**

TAIVoiceMonitor es un componente avanzado de Delphi/Object Pascal diseñado para la **detección inteligente de voz en tiempo real**. Su principal fortaleza radica en su capacidad de **auto-calibrarse** al ambiente sonoro y detectar automáticamente cuándo una persona comienza y termina de hablar.

**Características Principales**

**🎯 Detección Automática de Voz**

* Distingue entre ruido ambiente y voz humana
* Detecta automáticamente el inicio y final del habla
* No requiere configuración manual de umbrales

**🔧 Auto-Calibración Inteligente**

* Al activarse, analiza el ruido ambiente durante 3 segundos
* Calcula automáticamente los umbrales de sensibilidad óptimos
* Se adapta a diferentes entornos (oficina silenciosa, lugar ruidoso, etc.)

**🎙️ Captura Multi-Plataforma**

* **Windows**: Utiliza la API WaveIn de Windows
* **Android**: Usa AudioRecord de Android con manejo de permisos
* Abstrae las diferencias de plataforma para el desarrollador

**🔊 Transcripción en Tiempo Real**

* Divide el audio en fragmentos inteligentes durante el habla
* Detecta pausas naturales para crear fragmentos coherentes
* Perfecto para integrar con servicios de transcripción como OpenAI Whisper

**¿Cómo Funciona?**

**Fase 1: Calibración Automática**

1. Usuario activa el componente

2. Monitor escucha el ambiente por 3 segundos

3. Calcula el nivel promedio de ruido ambiente

4. Establece umbrales dinámicos:

- Sensibilidad de inicio = RuidoAmbiente × 4.0

- Sensibilidad de parada = RuidoAmbiente × 2.0

**Fase 2: Detección Inteligente**

1. Monitorea constantemente el nivel de audio

2. Cuando supera el umbral de inicio → "Comienza a hablar"

3. Captura todo el audio mientras habla

4. Cuando está por debajo del umbral de parada por 1 segundo → "Termina de hablar"

5. Entrega el audio completo en formato WAV

**Fase 3: Procesamiento Adicional**

* **Verificación de Wake Word**: Analiza los primeros segundos para palabras de activación
* **Fragmentos de Transcripción**: Divide el audio en chunks para transcripción en tiempo real
* **Gestión de Memoria**: Maneja eficientemente los buffers de audio

**Casos de Uso Típicos**

**🤖 Asistentes de Voz**

procedure TForm1.VoiceMonitorChangeState(Sender: TObject; aState: Boolean;

aIsValidForIA: Boolean; aStream: TMemoryStream);

begin

if aState then

ShowMessage('Escuchando...')

else

begin

ShowMessage('Procesando audio...');

// Enviar aStream a servicio de transcripción

ProcessAudioWithWhisper(aStream);

end;

end;

**📝 Dictado de Texto**

procedure TForm1.OnTranscriptionFragment(Sender: TObject; aFragmentStream: TMemoryStream);

begin

// Transcribir fragmento en tiempo real

TranscribeFragment(aFragmentStream);

end;

**🔊 Detección de Actividad de Voz (VAD)**

procedure TForm1.VoiceMonitorUpdate(Sender: TObject; const aSoundLevel: Int64);

begin

ProgressBar1.Position := aSoundLevel; // Visualizar nivel de sonido

Label1.Caption := 'Hablando: ' + BoolToStr(VoiceMonitor1.IsSpeaking, True);

end;

**Ventajas sobre Otros Enfoques**

**❌ Enfoque Manual Tradicional**

* Requiere ajustar manualmente umbrales
* No se adapta a diferentes ambientes
* Configuración compleja para el usuario final

**✅ TAIVoiceMonitor**

* **Plug & Play**: Solo activar y funciona
* **Auto-adaptativo**: Se calibra automáticamente
* **Robusto**: Funciona en ambientes variables
* **Multiplataforma**: Mismo código para Windows y Android

**Ejemplo de Implementación Básica**

procedure TForm1.FormCreate(Sender: TObject);

begin

VoiceMonitor1.OnChangeState := VoiceMonitorChangeState;

VoiceMonitor1.OnCalibrated := VoiceMonitorCalibrated;

VoiceMonitor1.OnError := VoiceMonitorError;

end;

procedure TForm1.Button1Click(Sender: TObject);

begin

VoiceMonitor1.Active := True; // ¡Eso es todo!

end;

procedure TForm1.VoiceMonitorCalibrated(Sender: TObject;

const aNoiseLevel, aSensitivity, aStopSensitivity: Integer);

begin

Memo1.Lines.Add(Format('Calibrado - Ruido: %d, Sensibilidad: %d/%d',

[aNoiseLevel, aSensitivity, aStopSensitivity]));

end;

procedure TForm1.VoiceMonitorChangeState(Sender: TObject; aState: Boolean;

aIsValidForIA: Boolean; aStream: TMemoryStream);

begin

if aState then

begin

Label1.Caption := '🎤 Escuchando...';

Label1.Font.Color := clRed;

end

else

begin

Label1.Caption := '✅ Audio capturado';

Label1.Font.Color := clGreen;

// Aquí tienes el audio completo en formato WAV

if Assigned(aStream) then

begin

SaveAudioToFile(aStream); // Guardar archivo

// o enviar a servicio de transcripción

end;

end;

end;

**Próximos Pasos**

Con esta introducción, ya comprendes el propósito y funcionamiento básico del componente. En las siguientes secciones del manual exploraremos:

* **Configuración detallada** de propiedades
* **Integración con servicios de IA** (OpenAI, Google Speech, etc.)
* **Manejo avanzado de eventos**
* **Optimización de rendimiento**
* **Resolución de problemas comunes**

¡El componente TAIVoiceMonitor te permitirá crear aplicaciones de voz sofisticadas con muy pocas líneas de código!

Te explico cada una de las propiedades del componente TAIVoiceMonitor:

**Propiedades Públicas (Solo Lectura)**

**Active: Boolean**

* Controla si el monitor de voz está activo o inactivo
* Cuando se activa, inicia la captura y calibración automática
* Al desactivarse, detiene toda la captura de audio

**IsSpeaking: Boolean**

* Indica si actualmente se está detectando voz/habla
* Se actualiza automáticamente basado en los niveles de audio y sensibilidad

**Sensitivity: Integer**

* Nivel de sensibilidad calculado automáticamente durante la calibración
* Determina el umbral mínimo para detectar el inicio del habla
* Se calcula como: ruido ambiente × SensitivityMultiplier

**StopSensitivity: Integer**

* Nivel de sensibilidad para detectar el final del habla
* Siempre menor que Sensitivity para evitar cortes abruptos
* Se calcula como: ruido ambiente × StopSensitivityMultiplier

**State: TAiMonitorState**

* Estado actual del monitor: msIdle, msRequestingPermission, msCalibrating, msMonitoring, msError

**SoundLevel: Int64**

* Nivel de sonido actual en tiempo real
* Se actualiza constantemente durante la captura

**Propiedades Configurables (Published)**

**SilenceDuration: Integer** (Predeterminado: 1000ms)

* Duración mínima de silencio requerida para considerar que terminó el habla
* Valores menores a 300ms se ajustan automáticamente a 300ms

**SensitivityMultiplier: Double** (Predeterminado: 4.0)

* Multiplicador para calcular la sensibilidad de inicio basada en el ruido ambiente
* Valores más altos = más sensible a sonidos débiles
* Valores más bajos = menos sensible, requiere más volumen

**StopSensitivityMultiplier: Double** (Predeterminado: 2.0)

* Multiplicador para calcular cuándo termina el habla
* Debe ser menor que SensitivityMultiplier para funcionar correctamente

**WakeWordDurationMs: Integer** (Predeterminado: 1000ms)

* Duración del fragmento de audio enviado para verificación de palabra de activación
* Define cuántos milisegundos del inicio del habla se capturan para análisis

**TranscriptionIntervalMs: Integer** (Predeterminado: 1500ms)

* Intervalo mínimo entre fragmentos de transcripción en tiempo real
* Controla qué tan frecuentemente se envían fragmentos para transcripción

**TranscriptionMaxWaitMs: Integer** (Predeterminado: 4000ms)

* Tiempo máximo de espera antes de forzar el envío de un fragmento de transcripción
* Evita fragmentos demasiado largos si no hay pausas naturales

**FragmentSplitRatio: Double** (Predeterminado: 0.35)

* Ratio para detectar pausas naturales y dividir fragmentos de transcripción
* Se usa cuando el nivel actual es menor al pico × ratio (momento de silencio relativo)

**Eventos**

**OnChangeState** - Se dispara al cambiar entre habla/silencio

**OnCalibrated** - Se dispara al completar la calibración automática  
**OnUpdate** - Se dispara constantemente con el nivel de sonido actual

**OnError** - Se dispara cuando ocurre un error

**OnWakeWordCheck** - Se dispara para verificar palabra de activación **OnTranscriptionFragment** - Se dispara con fragmentos de audio para transcripción en tiempo real

## Ejemplo:

**Demo Completa - TAIVoiceMonitor con IA**

Esta demo muestra una implementación completa de un asistente de voz usando TAIVoiceMonitor integrado con servicios de IA (Whisper para transcripción y OpenAI para respuestas).

**Arquitectura de la Demo**

Usuario habla → VoiceMonitor → Whisper (Wake Word) → IA Chat → Respuesta de Audio

↓ ↓ ↓ ↓ ↓

Captura Detección Verificación Procesamiento Reproducción

Audio de Voz "Sofía" del Comando del Audio

**Componentes Principales**

**1. TAIVoiceMonitor - Monitor de Voz Inteligente**

***AIVoiceMonitor: TAIVoiceMonitor;***

* Detecta automáticamente inicio/fin de voz
* Se auto-calibra al ambiente
* Genera fragmentos para transcripción en tiempo real

**2. TAIWhisper - Transcripción de Audio**

***Whisper: TAIWhisper;***

* Convierte audio a texto usando OpenAI Whisper
* Usado para verificar wake word "Sofía"
* Transcribe fragmentos en tiempo real

**3. TAiChatConnection - Conexión con IA**

***AiConn: TAiChatConnection;***

* Envía comandos de voz a servicios de IA (OpenAI, Claude, etc.)
* Maneja respuestas multimedia (texto, audio, imágenes)
* Gestiona conversaciones contextuales

**4. TMediaPlayer - Reproducción de Audio**

***MediaPlayer1: TMediaPlayer;***

* Reproduce respuestas de audio generadas por la IA

**Estados Visuales del Sistema**

La demo utiliza un enum para representar estados visuales:

TImageStatus = (

isInactivo, // Sistema apagado

isPreparing, // Calibrando ambiente

isListening, // Esperando wake word

isUserTalking, // Usuario hablando

isIaTalking // IA respondiendo

);

Cada estado cambia el ícono del botón principal para dar feedback visual inmediato.

**Flujo de Funcionamiento Detallado**

**Fase 1: Activación del Sistema**

procedure TFVoiceMonitor.StartMonitoring;

begin

MemoLog.Lines.Clear;

LblStatus.Text := 'Status: Preparando el ambiente, por favor espere';

// Configurar visualización

FCurrentSoundLevel := 0;

FMaxLevelSeen := 100;

ProgressBarLevel.Max := FMaxLevelSeen;

// Activar componentes

AnimationTimer.Enabled := True;

AIVoiceMonitor.Active := True; // ¡Inicia auto-calibración!

BtnPlay.ImageIndex := Integer(TImageStatus.isPreparing);

end;

**¿Qué sucede aquí?**

* Se limpia el log de eventos
* Se inicializa la barra de progreso para mostrar niveles de audio
* Se activa el monitor que automáticamente inicia calibración de 3 segundos
* El estado visual cambia a "Preparando"

**Fase 2: Calibración Automática**

procedure TFVoiceMonitor.AIVoiceMonitorCalibrated(Sender: TObject;

const aNoiseLevel, aSensitivity, aStopSensitivity: Integer);

begin

LblStatus.Text := 'Status: Escuchando...';

BtnPlay.ImageIndex := Integer(TImageStatus.isListening);

end;

**¿Qué sucede aquí?**

* El monitor terminó de analizar el ruido ambiente
* Calculó automáticamente los umbrales óptimos
* Sistema listo para detectar la wake word "Sofía"
* Estado visual cambia a "Escuchando"

**Fase 3: Detección de Voz**

procedure TFVoiceMonitor.AIVoiceMonitorChangeState(Sender: TObject;

aState, aIsValidForIA: Boolean; aStream: TMemoryStream);

begin

if aState then

begin

// ¡Usuario empezó a hablar!

MemoLog.Lines.Add('-> Empezaste a hablar...');

BtnPlay.ImageIndex := Integer(TImageStatus.isUserTalking);

end

else

begin

// Usuario terminó de hablar

BtnPlay.ImageIndex := Integer(TImageStatus.isListening);

if aIsValidForIA then // ¡Wake word "Sofía" detectada!

begin

// Procesar comando completo con IA

MF := TAiMediaFile.Create;

MF.LoadFromStream('nada.wav', aStream);

Res := AiConn.AddMessageAndRun('','user',[MF]);

end;

// Si no es válida, se descarta automáticamente

end;

end;

**¿Qué sucede aquí?**

* Cuando aState = True: Usuario comenzó a hablar → Estado visual "Hablando"
* Cuando aState = False: Usuario terminó → Se verifica si la wake word fue válida
* Solo si aIsValidForIA = True se procesa el comando con IA
* Audio inválido se descarta automáticamente

**Fase 4: Verificación de Wake Word**

procedure TFVoiceMonitor.AIVoiceMonitorWakeWordCheck(Sender: TObject;

aWakeWordStream: TMemoryStream; var IsValid: Boolean);

var

Res: string;

begin

// Transcribir primeros segundos del audio

Res := LowerCase(Whisper.Transcription(aWakeWordStream, 'nada.wav', ''));

// ¿Contiene "sofía"?

IsValid := AnsiContainsText(Res, 'sofía');

// Log del resultado

if IsValid then

MemoLog.Lines.Add('-->"' + Res + '"<-- → SÍ, es la palabra correcta!')

else

MemoLog.Lines.Add('-->"' + Res + '"<-- → NO, palabra incorrecta.');

end;

**¿Qué sucede aquí?**

* Se ejecuta en paralelo mientras el usuario habla
* Solo analiza los primeros 1000ms de audio (configurable)
* Usa Whisper para convertir audio a texto
* Busca la palabra "sofía" (insensible a mayúsculas)
* Resultado determina si el audio completo será procesado

**Fase 5: Transcripción en Tiempo Real**

procedure TFVoiceMonitor.AIVoiceMonitorTranscriptionFragment(Sender: TObject;

aFragmentStream: TMemoryStream);

begin

TTask.Run(

procedure

var

Whi: TAIWhisper;

Res: String;

begin

// Crear instancia separada para el hilo

Whi := TAIWhisper.Create(nil);

try

// Configurar credenciales

Whi.ApiKey := Whisper.ApiKey;

Whi.Model := Whisper.Model;

Whi.Url := Whisper.Url;

// Transcribir fragmento

Res := Whi.Transcription(aFragmentStream, 'nada.wav', '');

// Mostrar texto en tiempo real

TThread.Queue(nil,

procedure

begin

MemoPrompt.Lines.Text := MemoPrompt.Lines.Text + Res + ' ';

end);

finally

Whi.Free;

end;

end);

end;

**¿Qué sucede aquí?**

* Se ejecuta cada 1.5 segundos mientras el usuario habla
* Crea fragmentos inteligentes basados en pausas naturales
* Transcribe cada fragmento en paralelo (no bloquea la UI)
* Muestra texto en tiempo real conforme el usuario habla
* ¡El usuario ve su dictado aparecer mientras habla!

**Fase 6: Procesamiento de Respuesta IA**

procedure TFVoiceMonitor.AiConnReceiveDataEnd(const Sender: TObject;

aMsg: TAiChatMessage; aResponse: TJSONObject; aRole, aText: string);

var

MF: TAiMediaFile;

Ext, FileName: String;

begin

// La IA puede responder con múltiples tipos de archivo

if Assigned(aMsg) and (aMsg.MediaFiles.Count > 0) then

begin

for MF in aMsg.MediaFiles do

begin

Ext := LowerCase(ExtractFileExt(MF.FileName));

case Ext of

'.wav', '.mp3':

begin

// Respuesta de audio - reproducir inmediatamente

MF.SaveToFile(FileName);

MediaPlayer1.FileName := FileName;

MediaPlayer1.Play;

end;

'.jpg', '.png', '.bmp':

begin

// Respuesta con imagen - mostrar en interfaz

// Image1.Bitmap.LoadFromFile(FileName);

end;

'.txt', '.pas', '.js':

begin

// Respuesta de código/texto - mostrar en editor

// MemoCode.Lines.LoadFromStream(MF.Content);

end;

end;

end;

end;

end;

**¿Qué sucede aquí?**

* La IA procesa el comando de voz y puede responder con múltiples formatos
* **Audio**: Se reproduce automáticamente (IA hablando de vuelta)
* **Imágenes**: Se muestran en la interfaz
* **Código/Texto**: Se muestra en editores
* **Videos**: Pueden abrirse en reproductores

**Fase 7: Feedback Visual en Tiempo Real**

procedure TFVoiceMonitor.AnimationTimerTimer(Sender: TObject);

const

SMOOTHING\_FACTOR = 0.2; // Suavizado de animación

var

CurrentPos, TargetPos, NewPos: Double;

StateStr: string;

begin

// Obtener nivel de sonido objetivo

if AIVoiceMonitor.Active then

TargetPos := FCurrentSoundLevel

else

TargetPos := 0;

// Ajustar escala dinámicamente

if TargetPos > ProgressBarLevel.Max then

ProgressBarLevel.Max := Round(TargetPos \* 1.2);

// Animación suave hacia el objetivo

CurrentPos := ProgressBarLevel.Value;

NewPos := CurrentPos + Round((TargetPos - CurrentPos) \* SMOOTHING\_FACTOR);

ProgressBarLevel.Value := Min(NewPos, ProgressBarLevel.Max);

// Mostrar estado actual

case AIVoiceMonitor.State of

msCalibrating: StateStr := 'Calibrando...';

msMonitoring: StateStr := 'Monitorizando';

msRequestingPermission: StateStr := 'Pidiendo permiso...';

msError: StateStr := 'Error';

else

StateStr := 'Inactivo';

end;

end;

**¿Qué sucede aquí?**

* Timer que se ejecuta cada pocos milisegundos
* Actualiza la barra de progreso con animación suave
* Ajusta la escala automáticamente según el volumen
* Muestra el estado actual del sistema
* Proporciona feedback visual continuo al usuario

**Configuración Necesaria**

**1. API Keys**

// En el evento FormCreate o similar

Whisper.ApiKey := 'tu-openai-api-key';

AiConn.ApiKey := 'tu-openai-api-key';

**2. Configuración de Wake Word**

AIVoiceMonitor.WakeWordDurationMs := 1000; // Primeros 1000ms para verificar

**3. Configuración de Transcripción**

AIVoiceMonitor.TranscriptionIntervalMs := 1500; // Fragmento cada 1.5s

AIVoiceMonitor.TranscriptionMaxWaitMs := 4000; // Máximo 4s por fragmento

AIVoiceMonitor.FragmentSplitRatio := 0.35; // Detectar pausas al 35% del pico

**Manejo de Errores**

**Error en Monitor de Voz**

procedure TFVoiceMonitor.AIVoiceMonitorError(Sender: TObject; const ErrorMessage: string);

begin

MemoLog.Lines.Add('ERROR: ' + ErrorMessage);

ShowMessage('Ocurrió un error en el monitor de audio: ' + sLineBreak + ErrorMessage);

StopMonitoring; // Parar todo de forma segura

end;

**Error en Conexión IA**

procedure TFVoiceMonitor.AiConnError(Sender: TObject; const ErrorMsg: string;

Exception: Exception; const AResponse: IHTTPResponse);

begin

// Manejar errores de API (límites, conexión, etc.)

MemoLog.Lines.Add('Error IA: ' + ErrorMsg);

end;

**Ventajas de Esta Implementación**

**✅ Sistema Completo de Asistente de Voz**

* Wake word inteligente ("Sofía")
* Transcripción en tiempo real
* Respuestas multimedia de IA
* Feedback visual continuo

**✅ Eficiencia de Recursos**

* Solo procesa audio cuando detecta wake word válida
* Transcripción paralela no bloquea la interfaz
* Gestión automática de memoria

**✅ Experiencia de Usuario Profesional**

* Estados visuales claros
* Feedback inmediato
* Calibración automática
* Manejo robusto de errores

**✅ Flexibilidad**

* Wake word configurable
* Múltiples servicios de IA soportados
* Respuestas multimedia automáticas
* Fácil personalización

**Casos de Uso Extendidos**

**🏠 Domótica**

if AnsiContainsText(Res, 'enciende las luces') then

// Enviar comando a sistema domótico

**💼 Productividad**

if AnsiContainsText(Res, 'programa reunión') then

// Integrar con calendario

**🎮 Gaming**

if AnsiContainsText(Res, 'pausa juego') then

// Control por voz de juegos

**🚗 Automotriz**

if AnsiContainsText(Res, 'navegar a') then

// Integrar con GPS/navegación

Esta demo muestra cómo crear un asistente de voz completo y profesional con muy pocas líneas de código, aprovechando la potencia del componente TAIVoiceMonitor.

**Formulario de Configuración - TAIVoiceMonitor**

**Parámetros Configurables del TAIVoiceMonitor**

**🎯 Categoría: Detección de Voz**

**1. SilenceDuration (Integer) - *Duración de Silencio***

pascal

property SilenceDuration: Integer default DEFAULT\_SILENCE\_DURATION\_MS; *// 1000ms*

**¿Qué hace?**

* Define cuánto tiempo de silencio debe detectar para considerar que el usuario terminó de hablar
* Evita cortes prematuros si hay pausas naturales al hablar

**Configuración recomendada:**

* **Rápida**: 500-800ms (para comandos cortos)
* **Normal**: 1000-1500ms (conversación natural)
* **Lenta**: 2000-3000ms (para personas que hablan despacio)

**Control sugerido:** SpinEdit con rango 300-5000ms

**2. SensitivityMultiplier (Double) - *Multiplicador de Sensibilidad***

pascal

property SensitivityMultiplier: Double default 4.0;

**¿Qué hace?**

* Multiplica el nivel de ruido ambiente para determinar cuándo inicia la detección de voz
* Sensibilidad = RuidoAmbiente × SensitivityMultiplier

**Configuración recomendada:**

* **Ambiente silencioso**: 2.0-3.0 (oficina, casa silenciosa)
* **Ambiente normal**: 3.0-5.0 (oficina con ruido moderado)
* **Ambiente ruidoso**: 5.0-8.0 (cafetería, calle)

**Control sugerido:** FloatSpinEdit con rango 1.0-10.0, incrementos 0.5

**3. StopSensitivityMultiplier (Double) - *Multiplicador de Parada***

pascal

property StopSensitivityMultiplier: Double default 2.0;

**¿Qué hace?**

* Multiplica el nivel de ruido ambiente para determinar cuándo parar la detección
* StopSensitivity = RuidoAmbiente × StopSensitivityMultiplier
* **IMPORTANTE**: Debe ser menor que SensitivityMultiplier

**Configuración recomendada:**

* **Siempre menor** que SensitivityMultiplier
* **Diferencia típica**: 1.5-2.0 puntos menos que SensitivityMultiplier
* **Ejemplo**: Si SensitivityMultiplier=4.0, entonces StopSensitivityMultiplier=2.0-2.5

**Control sugerido:** FloatSpinEdit con validación automática

**🔊 Categoría: Wake Word (Palabra de Activación)**

**4. WakeWordDurationMs (Integer) - *Duración de Palabra Clave***

pascal

property WakeWordDurationMs: Integer default DEFAULT\_WAKE\_WORD\_DURATION\_MS; *// 1000ms*

**¿Qué hace?**

* Define cuántos milisegundos del inicio del audio se envían para verificar la wake word
* Solo analiza el principio del audio, no todo el comando

**Configuración recomendada:**

* **Wake word corta** ("OK", "Hey"): 500-800ms
* **Wake word normal** ("Sofía", "Alexa"): 1000-1500ms
* **Frase completa** ("Hola computadora"): 2000-3000ms

**Control sugerido:** SpinEdit con rango 500-3000ms

**📝 Categoría: Transcripción en Tiempo Real**

**5. TranscriptionIntervalMs (Integer) - *Intervalo de Transcripción***

pascal

property TranscriptionIntervalMs: Integer default DEFAULT\_TRANSCRIPTION\_INTERVAL\_MS; *// 1500ms*

**¿Qué hace?**

* Tiempo mínimo entre fragmentos de transcripción en tiempo real
* Evita fragmentos demasiado cortos que serían inútiles

**Configuración recomendada:**

* **Tiempo real rápido**: 1000-1500ms (transcripción muy reactiva)
* **Balance calidad/velocidad**: 1500-2500ms (recomendado)
* **Calidad prioritaria**: 2500-4000ms (fragmentos más largos)

**Control sugerido:** SpinEdit con rango 1000-5000ms

**6. TranscriptionMaxWaitMs (Integer) - *Tiempo Máximo de Espera***

pascal

property TranscriptionMaxWaitMs: Integer default DEFAULT\_TRANSCRIPTION\_MAX\_WAIT\_MS; *// 4000ms*

**¿Qué hace?**

* Tiempo máximo antes de forzar el envío de un fragmento
* Evita que una frase muy larga nunca se procese por falta de pausas

**Configuración recomendada:**

* **Siempre mayor** que TranscriptionIntervalMs
* **Típicamente**: 2-3x TranscriptionIntervalMs
* **Ejemplo**: Si TranscriptionIntervalMs=1500ms, entonces TranscriptionMaxWaitMs=4000ms

**Control sugerido:** SpinEdit con validación automática

**7. FragmentSplitRatio (Double) - *Ratio de División de Fragmentos***

pascal

property FragmentSplitRatio: Double default DEFAULT\_FRAGMENT\_SPLIT\_RATIO; *// 0.35*

**¿Qué hace?**

* Detecta pausas naturales para dividir fragmentos inteligentemente
* Cuando el nivel actual < (NivelPico × FragmentSplitRatio) = pausa detectada

**Configuración recomendada:**

* **Muy sensible a pausas**: 0.20-0.30 (divide en pausas mínimas)
* **Sensibilidad normal**: 0.30-0.40 (pausas naturales)
* **Menos sensible**: 0.40-0.60 (solo pausas pronunciadas)

**Control sugerido:** FloatSpinEdit con rango 0.1-0.8, incrementos 0.05

**Diseño del Formulario de Configuración**

**Estructura Visual Sugerida:**

pascal

TConfigForm = class(TForm)

private

*// Controles de Detección de Voz*

SpinSilenceDuration: TSpinEdit;

FloatSpinSensitivityMultiplier: TFloatSpinEdit;

FloatSpinStopSensitivityMultiplier: TFloatSpinEdit;

*// Controles de Wake Word*

SpinWakeWordDuration: TSpinEdit;

*// Controles de Transcripción*

SpinTranscriptionInterval: TSpinEdit;

SpinTranscriptionMaxWait: TSpinEdit;

FloatSpinFragmentSplitRatio: TFloatSpinEdit;

*// Controles de Presets*

ComboPresets: TComboBox;

BtnSavePreset: TButton;

BtnLoadPreset: TButton;

*// Testing en Vivo*

BtnTestCalibration: TButton;

ProgressBarTestLevel: TProgressBar;

LabelTestResults: TLabel;

end;

**Presets Recomendados:**

**🏠 Ambiente Silencioso (Casa/Oficina Privada)**

pascal

procedure LoadQuietPreset;

begin

SilenceDuration := 800;

SensitivityMultiplier := 2.5;

StopSensitivityMultiplier := 1.5;

WakeWordDurationMs := 1000;

TranscriptionIntervalMs := 1200;

TranscriptionMaxWaitMs := 3500;

FragmentSplitRatio := 0.25;

end;

**🏢 Ambiente Normal (Oficina Compartida)**

pascal

procedure LoadNormalPreset;

begin

SilenceDuration := 1000;

SensitivityMultiplier := 4.0;

StopSensitivityMultiplier := 2.0;

WakeWordDurationMs := 1000;

TranscriptionIntervalMs := 1500;

TranscriptionMaxWaitMs := 4000;

FragmentSplitRatio := 0.35;

end;

**🔊 Ambiente Ruidoso (Lugar Público)**

pascal

procedure LoadNoisyPreset;

begin

SilenceDuration := 1500;

SensitivityMultiplier := 6.0;

StopSensitivityMultiplier := 3.0;

WakeWordDurationMs := 1500;

TranscriptionIntervalMs := 2000;

TranscriptionMaxWaitMs := 5000;

FragmentSplitRatio := 0.45;

end;

**Validaciones Importantes:**

pascal

procedure ValidateConfiguration;

begin

*// StopSensitivity debe ser menor que Sensitivity*

if StopSensitivityMultiplier >= SensitivityMultiplier then

StopSensitivityMultiplier := SensitivityMultiplier - 0.5;

*// MaxWait debe ser mayor que Interval*

if TranscriptionMaxWaitMs <= TranscriptionIntervalMs then

TranscriptionMaxWaitMs := TranscriptionIntervalMs \* 2;

*// SilenceDuration mínimo*

if SilenceDuration < 300 then

SilenceDuration := 300;

*// FragmentSplitRatio en rango válido*

if FragmentSplitRatio < 0.1 then FragmentSplitRatio := 0.1;

if FragmentSplitRatio > 0.8 then FragmentSplitRatio := 0.8;

end;

**Testing en Vivo:**

pascal

procedure TestCurrentConfiguration;

begin

*// Aplicar configuración temporal*

TestVoiceMonitor.SilenceDuration := SpinSilenceDuration.Value;

TestVoiceMonitor.SensitivityMultiplier := FloatSpinSensitivityMultiplier.Value;

*// ... etc*

*// Activar por 10 segundos para probar*

TestVoiceMonitor.Active := True;

TestTimer.Interval := 10000; *// 10 segundos*

TestTimer.Enabled := True;

BtnTestCalibration.Text := 'Probando... (10s)';

BtnTestCalibration.Enabled := False;

end;

**Consejos de UI/UX:**

**📊 Indicadores Visuales**

* **Barra de progreso** mostrando niveles de sonido en tiempo real
* **Indicadores de umbral** (líneas en la barra mostrando Sensitivity/StopSensitivity)
* **Estado visual** del monitor (Calibrando/Escuchando/Error)

**🔧 Controles Inteligentes**

* **Sliders con etiquetas** para valores más intuitivos
* **Validación en tiempo real** con mensajes de ayuda
* **Botón "Restaurar Defaults"** para volver a valores por defecto

**💾 Persistencia**

* **Guardar configuración** automáticamente
* **Múltiples perfiles** (Casa, Oficina, Móvil)
* **Importar/Exportar** configuraciones

**🎯 Asistente de Configuración**

* **Wizard inicial** que guíe la primera configuración
* **Test de ambiente** automático que sugiera presets
* **Explicaciones contextuales** para cada parámetro

**Código de Ejemplo - Aplicar Configuración:**

pascal

procedure ApplyConfigurationToVoiceMonitor(VoiceMonitor: TAIVoiceMonitor);

begin

VoiceMonitor.SilenceDuration := SpinSilenceDuration.Value;

VoiceMonitor.SensitivityMultiplier := FloatSpinSensitivityMultiplier.Value;

VoiceMonitor.StopSensitivityMultiplier := FloatSpinStopSensitivityMultiplier.Value;

VoiceMonitor.WakeWordDurationMs := SpinWakeWordDuration.Value;

VoiceMonitor.TranscriptionIntervalMs := SpinTranscriptionInterval.Value;

VoiceMonitor.TranscriptionMaxWaitMs := SpinTranscriptionMaxWait.Value;

VoiceMonitor.FragmentSplitRatio := FloatSpinFragmentSplitRatio.Value;

end;

Este formulario de configuración permitirá a los usuarios personalizar completamente el comportamiento del VoiceMonitor según su ambiente específico y necesidades de uso.