UNIVERSIDAD AUTÓNOMA METROPOLITANA
UNIDAD AZCAPOTZALCO

DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

SRCV: SISTEMA COMPUTACIONAL INTERACTIVO BASADO EN RECONOCIMIENTO DE COMANDOS DE VOZ PARA APLICACIONES EDUCATIVAS

PROYECTO TERMINAL QUE PRESENTA:

DÍAZ MENDOZA ALEJANDRO

MATRÍCULA: 206303018

PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACIÓN

ASESOR DEL PROYECTO:

DR. ARMANDO JIMÉNEZ FLORES

MÉXICO, D.F. ENERO, 2012

RESUMEN

Actualmente, los reconocedores de voz han adquirido mayores grados de precisión y desempeño, facilitando el desarrollo de herramientas de comunicación hombre-máquina cada vez más amigables. Por consiguiente, resulta atractivo el desarrollo de herramientas computacionales basadas en voz para usuarios infantiles, con el fin de apoyarlos en su proceso de aprendizaje mediante estímulos gráficos y acústicos.

El presente trabajo consiste en el desarrollo de un sistema de reconocimiento de comandos de voz, denominado SRCV, que hace uso de la herramienta **HTK *(Hidden Markov Model ToolKit)*,** basada en modelos estadísticos conocidos como **Modelos Ocultos de Markov (*Hidden Markov Models*, HMMs).** El objetivo del SRCV es obtener una herramienta computacional interactiva con fines educativos que pueda ser utilizada por niños en etapa preescolar, para estimular sus habilidades cognitivas (memoria y razonamiento de conceptos básicos sobre tamaño, forma, colores y otros). El sistema SRCV utiliza una base de datos creada con las voces del usuario y un conjunto de reactivos construidos con imágenes y voces pregrabadas

El sistema SRCV genera preguntas con voces pregrabadas y gráficos, mediante una **Interfaz Gráfico/acústica**, y acepta respuestas mediante palabras simples a través un micrófono. Las respuestas son analizadas y evaluadas mediante un bloque de **Reconocimiento de Comandos de Voz** y un **Intérprete de Comandos.**

Durante las pruebas realizadas con el sistema SRCV participaron dos interlocutores: el primero con sus patrones de voz registrados en la base de datos, el segundo sin estar registrado en la base mencionada. Las pruebas consistieron en pronunciar, de manera repetida, un conjunto de palabras seleccionadas del vocabulario del reconocedor de voz. Los resultados obtenidos muestran un nivel de precisión promedio del 91%, para el primer interlocutor, y un porcentaje del 63% para el segundo interlocutor, cuyos patrones de voz no están almacenados en la base de datos del reconocedor de voz.

AGRADECIMIENTOS

Quiero dedicar este proyecto a mis padres, Alejandro y Gabriela, por todo el apoyo, cariño y confianza que me han dado a lo largo de toda mi vida, y que sin duda, han sido personas claves en la culminación de mis estudios.

Agradecer a la Universidad Autónoma Metropolitana Azcapotzalco por haberme abierto sus puertas y darme una formación académica de calidad.

También agradecer a mi tutor, el Dr. Armando Jiménez Flores, por la oportunidad que me ha brindado de realizar el Proyecto de Fin de Carrera, así como por su dedicación, ayuda, paciencia y comprensión en el transcurso del mismo.

Nunca desistas de un sueño.
Sólo trata de ver las señales que te lleven a él.
Paulo Coelho

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN

En el proceso de búsqueda de herramientas que faciliten y hagan más amigable el trabajo de las personas, se han desarrollado sistemas con interfaces basadas en visión, tacto, oído, y órgano fonador (voz). Este proyecto está planeado para diseñar e implementar una herramienta computacional, basada en comandos de voz, que ayude a niños de edad preescolar a desarrollar sus capacidades de aprendizaje mediante sonidos, formas, colores y caracteres alfanuméricos, aun cuando todavía no sepa leer ni escribir. A dicha herramienta se le ha denominado **SRCV**: **S**istema Computacional Interactivo basado en **R**econocimiento de **C**omandos de **V**oz para Aplicaciones Educativas.

El sistema SRCV será capaz de aceptar comandos de voz (palabras simples dentro del "*vocabulario del sistema*"), como respuesta a preguntas sencillas, apoyadas con imágenes para ser comprensibles por los usuarios (niños en edad preescolar). El SRCV analizará las respuestas apoyándose en un Banco de Reactivos.

La estructura interna del SRCV constará de los siguientes bloques:

- Interfaz Gráfico-acústica
- Reconocedor de Comandos de Voz
- Intérprete de Comandos
- Generador de Preguntas y Respuestas

A continuación se describen dichos bloques:
Interfaz Gráfico-acústica: Se encargará de interactuar con el usuario, mostrando aleatoriamente preguntas previamente almacenadas en un Banco de Reactivos y apoyadas con gráficos (figuras geométricas, símbolos y colores, entre otros) de fácil comprensión, y aceptando respuestas vía un micrófono.

Reconocedor de Comandos de Voz: Este bloque devolverá como resultado el texto correspondiente a la mejor hipótesis de la palabra pronunciada como respuesta por un usuario, para ello el reconocedor comparará la respuesta apoyándose en patrones de voz almacenados en una Base de Datos y determinará que palabra es la que más se aproxima a la pronunciada.

Intérprete de Comandos: Analizará y validará si las respuestas del usuario son correctas. El bloque se apoyará en un Banco de Reactivos de donde obtendrá la respuesta correcta para poder compararla con la respuesta dicha por el usuario.

Generador de Preguntas y Respuestas: Se encargará de seleccionar y enviar al bloque Interfaz Gráfico-acústica la pregunta a mostrar al usuario. También, se encargará de enviar a dicha Interfaz el mensaje de evaluación (audio e imagen) de la respuesta del usuario (si fue correcta o no).

Las pruebas que se realizarán sobre el reconocedor de comandos de voz serán en un entorno libre de ruido y consistirán en la repetición de palabras reconocidas por el sistema, un número determinado de veces, para así obtener el porcentaje de precisión del reconocedor. Para ello se necesitará la intervención de al menos 2 interlocutores, uno que haya prestado su voz en la creación de los "*patrones de voz*", y el otro sin haber participado en dicha creación, lo cual nos permitirá observar las diferencias en los resultados obtenidos entre un interlocutor y otro.

Extensivamente, el potencial de esta herramienta puede resultar de gran utilidad para personas con una discapacidad motriz que les impida manejar un teclado y un *mouse* convencionales. Para ello habrá que diseñar un programa específico para estos casos, el cual no está contemplado en este proyecto.

Actualmente, se dispone de algunas herramientas enfocadas al desarrollo de aplicaciones basadas en reconocimiento de voz, tales como, SAPI SDK de Microsoft y HTK **[1]**, del Departamento de Ingeniería de la Universidad de Cambridge, entre otras. Hoy en día, la mayoría de los sistemas reconocedores en funcionamiento se basan en la técnica de "*modelos ocultos de Markov*", debido a que requieren menos memoria física, ofrecen mejor tiempo de respuesta y una menor tasa de error con respecto a otras técnicas utilizadas en reconocimiento de voz.

1.1. Antecedentes

En los párrafos siguientes se citan algunos proyectos realizados en otras instituciones, relacionados con el reconocimiento de voz, donde se utilizan herramientas basadas en Modelos Ocultos de Markov similares a la herramienta HTK.

Uso de Reconocimiento de Voz en un Juego Electrónico para la Rehabilitación de Niños con el Problema de Lenguaje Dislalia [2]. En este trabajo se desarrolló un juego electrónico con reconocimiento de voz para estimular a niños con problemas de lenguaje dislalia. El juego se realizó utilizando el lenguaje C, las *librerías* gráficas SDL y los modelos acústicos se desarrollaron utilizando el software de reconocimiento de voz HTK.

Aplicaciones del Software de Reconocimiento de Voz en el Aprendizaje de la Lectoescritura [3]. El principal objetivo de este trabajo es comprobar la viabilidad del software de reconocimiento de voz como apoyo al aprendizaje de la lecto-escritura, y sus efectos sobre la comprensión y la lectura de palabras en niños y niñas de primer curso de primaria. Para ello se seleccionaron 14 participantes de

primaria con diferente nivel lector y fueron distribuidos en dos grupos según estuvieran por encima o debajo de la mediana. Se realizaron dos sesiones semanales de trabajo con este software durante un total de 14 semanas. Los resultados muestran la viabilidad del uso de este sistema en la enseñanza ordinaria, así como una clara mejoría de los participantes que tenían un nivel de lectura más bajo.

Tecnologías de Reconocimiento por Voz y su Aplicabilidad en Videojuegos [4]. En este proyecto se integra un Sistema de Reconocimiento de Voz en un videojuego mediante el control del *avatar* de un juego basado en comandos de voz. Utiliza los Modelos Ocultos de Markov y las actividades del *avatar* son limitadas ya que se restringe el conjunto de frases reconocibles por el sistema a una gramática específica, mejorando también así el resultado del reconocimiento. Para facilitar el entrenamiento y mejorar la tasa de reconocimiento, se dispone de una aplicación web que permite al usuario realizar grabaciones de voz de una serie de frases que contienen palabras de la gramática del lenguaje que se utiliza en el juego; reconoce comandos en idioma español e inglés.

Diferentes grupos han desarrollado aplicaciones interesantes y útiles para el apoyo del aprendizaje de un segundo idioma.

Pronunciation Power [5]. El cual proporciona al usuario una serie de herramientas que le permiten aprender la pronunciación del idioma Inglés. Crea una gráfica de onda sonora del sonido que se desea verificar, para esto, el usuario debe grabar su voz y comparar la gráfica generada con la gráfica de la "forma correcta" de pronunciación. La desventaja de este método es que cuando hay un error en la pronunciación este no es establecido explícitamente. Se requiere de práctica, pues se debe ver la representación gráfica de la señal de voz y la de "lo correcto" y determinar cómo y porque ocurrió el error. Esto puede resultar relativamente fácil para una persona que tenga experiencia, pero puede resultar tedioso. Además, no se sabe con base en qué criterio es elegida la representación gráfica como correcta. Una pronunciación correcta puede variar mucho en su apariencia debido a las diferencias naturales en el tracto vocal humano que puede hacer que el usuario crea que algo está mal.

Otro ejemplo interesante es de la compañía Language Connect, la cual usa **IBM Via Voice [5]**. El software, diseñado para la enseñanza de inglés, toma como entrada cada palabra, frase u oraciones complejas, entonces responde al usuario y da un puntaje sobre la pronunciación realizada. Este software, que utiliza reconocimiento de voz, es muy poderoso y tiene muy buen nivel de reconocimiento. Sin embargo, el puntaje que recibe el estudiante no es muy explícito, es decir, se sabe que existe un error pero no se indica en qué o dónde hay que corregir.

1.2. Justificación

El propósito de este proyecto es desarrollar un sistema computacional educativo e interactivo, por medio de comandos de voz (palabras aisladas) e imágenes, como apoyo a niños en etapa preescolar con el cual se logre estimular las habilidades cognitivas (la memoria, conceptos básicos como tamaño, forma, colores y otros). Para ello, se propone el diseño de una interfaz que sea amigable y de fácil uso que realice una serie de preguntas didácticas al niño en forma de audio (ya que por lo regular los niños en esta edad no saben leer), y apoyadas con elementos visuales.

La motivación para desarrollar este sistema se basa en el hecho de que, de acuerdo con investigaciones sobre el tema, la edad más importante para la estimulación de estas habilidades se da desde los dos años.

El tipo de sistema que se pretende desarrollar servirá de ayuda al proceso de aprendizaje del niño, pero nunca de sustituto a la labor pedagógica del profesor.

1.3. Relevancia del Proyecto

Debido a la importancia del habla en la vida del ser humano es importante fortalecer las habilidades lingüísticas e intelectuales de los niños centrando la atención en el apoyo y desarrollo de su educación escolar. Esto es particularmente necesario, considerando que existe un número considerable de niños y niñas que presentan dificultades en el aprendizaje escolar y requieren, por lo tanto de un apoyo más didáctico.

Actualmente, no existen muchas aplicaciones de reconocimiento de voz, en particular enfocadas al estímulo y fortalecimiento de conocimientos que reduzcan así el impacto del problema de aprendizaje en los niños. Este aspecto, es una de las fortalezas del SRCV porque ayuda al niño a aprender conceptos nuevos desde etapas tempranas, mediante la interacción con el sistema y la utilización de solamente su voz.

Es importante el uso del reconocimiento de voz en un juego infantil, ya que permite tener una nueva opción en el proceso de enseñanza educacional. Esto, considerando la importancia de los juegos en la vida del niño, ya que a través de ellos aprende a escuchar, reforzar sus conocimientos y a desarrollar nuevas habilidades.

1.4. Objetivos

1.4.1. General

Diseñar y construir un sistema computacional interactivo con reconocimiento de comandos de voz que de manera didáctica, ayude a niños en edad preescolar durante su proceso de aprendizaje.

1.4.2. Particulares

Estudiar, comprender y aplicar los fundamentos del análisis y reconocimiento de señales de voz para crear un tipo de interfaz hombre-máquina que facilite su interacción, utilizando la técnica de Modelos Ocultos e Markov.

Aplicar conocimientos fundamentales adquiridos en la carrera de Ingeniería en Computación para realizar cada una de las etapas de desarrollo de un sistema computacional, comenzando por un estudio de requerimientos hasta la implementación de los algoritmos y programas necesarios para el SRCV.

Colaborar en el diseño de herramientas computacionales que faciliten la interacción de la computadora con usuarios que no saben o no pueden escribir, como los niños en etapa preescolar o personas con alguna discapacidad motriz. Mediante la adaptación de interfaces adecuadas, estas herramientas también pueden ser útiles en el control de mecanismos, tales como equipos electrodomésticos e industriales.

1.5. Alcances del Proyecto

El Sistema propuesto en este proyecto reconocerá palabras aisladas (dígitos, colores, figuras, etc.) emitidas por el o los interlocutores que contribuyeron en la generación del *corpus* (base de datos de grabaciones y etiquetas), de voces y que serán los usuarios con mejor porcentaje de reconocimiento, además dicho reconocimiento deberá ser aceptable (igual o mayor al 85%).

Las diferencias con respecto a las capacidades máximas del diseño, y las que se implementarán, radican en la cantidad de usuarios y comandos, ya que el sistema a realizar reconocerá a un solo interlocutor o a un número limitado de estos, así como también contará con una cantidad reducida de comandos. Otra diferencia es el idioma, el cual se podría extender al inglés.

1.6.Organización del Documento

El presente documento esta organizado de la siguiente manera:

Capítulo 2. FUNDAMENTOS TEÓRICOS: Abarca los antecedentes necesarios para la comprensión de este trabajo. Se abordan los aspectos más importantes de la anatomía y fisiología como lo es el aparato fonador humano, así como la base teórica del Reconocimiento de Voz y los Modelos Ocultos de Markov.

Capítulo 3. LA HERRAMIENTA HTK: Se presenta la herramienta con la que diseñaremos el reconocedor que será utilizado por el SRCV. Se da una introducción de dicha herramienta y se habla de las etapas en el proceso de diseño de un reconocedor de voz con HTK.

Capítulo 4. DESCRIPCIÓN DEL SISTEMA SRCV: Dentro de este capítulo se muestra el diagrama conceptual y el diagrama a bloques del sistema SRCV, así como la descripción y especificación técnica de cada uno de esos bloques.

Capítulo 5. DISEÑO DEL SISTEMA SRCV: Se aborda el diseño de los bloques descritos en el capítulo anterior. Se explica el desarrollo del reconocedor de comandos de voz mediante la utilización de la herramienta HTK, así como de los bloques restantes del sistema con la ayuda del entorno de desarrollo Visual Studio.

Capítulo 6. PRUEBAS Y RESULTADOS: En este capítulo se muestran los resultados obtenidos después de haber realizado las pruebas de reconocimiento de voz con la ayuda de los interlocutores.

Capítulo 7. CONCLUSIONES: En este último capítulo se habla de los objetivos planteados y los objetivos alcanzados tras el desarrollo del presente proyecto, los resultados obtenidos, así como los trabajos futuros que pueden darle continuidad al proyecto.

CAPÍTULO 2
FUNDAMENTOS TEÓRICOS

2.1. Introducción

El reconocimiento de voz es el estudio de las señales de voz y las técnicas de procesado de estas señales. Las señales se digitalizan con el propósito de manipular su información, lo cual es llamado procesamiento digital de voz. El procesamiento digital de voz se puede dividir en varias categorías, la de nuestro interés es el reconocimiento de voz. El problema es identificar las palabras pronunciadas, sin importar el hablante. Bajo este esquema, se pre-procesan las señales de voz, se obtienen las características (patrones de voz), y finalmente se utilizan en el proceso de reconocimiento.

2.2. Antecedentes

A continuación se presenta una línea de tiempo referente a la historia del reconocimiento de voz.

1910. AT&T Bell *Laboratories* construye la primera máquina, basada en plantillas, capaz de reconocer voces de los 10 dígitos del Inglés. Requiere un extenso *entrenamiento* de la voz de una persona, pero una vez logrado tiene un 99% de certeza. Surge la esperanza de que el reconocimiento de voz sea simple y directo.

Las primeras investigaciones en el desarrollo de estos sistemas fueron realizadas en la década de los 50´s. Los estudios trataron de explotar las ideas fundamentales de la *fonética acústica*.

Durante la década de los 60´s los estudios se enfocaron, principalmente, a los problemas de segmentación, clasificación y reconocimiento de patrones.

En los 70´s se mejoró la tecnología de reconocimiento para palabras aisladas y continuas. Se hicieron reconocedores que aceptaban un vocabulario más extenso. También se desarrollaron técnicas de reconocimiento como: *time warping*, modelado probabilístico y el algoritmo de retropropagación.

En la década de los 80´s hubo un cambio en la tecnología, del enfoque basado en reconocimiento de patrones a métodos de modelado probabilístico, como el método de cadenas ocultas de Markov (HMM). Las redes neuronales se reintrodujeron para resolver problemas de reconocimiento de voz.

En la actualidad, existen diversos factores que contribuyen al mejoramiento y el progreso de los sistemas de Reconocimiento de Voz, como los HMM y las redes

neuronales. Se han realizado grandes esfuerzos para desarrollar una base de datos de voz con un vocabulario grande, el cual pueda ser usado en el entrenamiento, desarrollo y prueba de estos sistemas. Por otra parte, el establecimiento de estándares para la evaluación del desempeño en el reconocimiento permite hacer comparaciones entre distintos sistemas. Gracias a los avances en la tecnología computacional, los sistemas pueden ser probados en tiempo real sin la necesidad de *hardware* adicional **[5]**.

2.3. Aparato Fonador

Los seres humanos generamos la voz por medio de las cuerdas vocales, el aire que fluye desde los pulmones y un proceso de filtrado producido por las cavidades resonantes del tracto vocal. La **Figura 1** muestra un esquema del aparato fonador humano **[6]**.

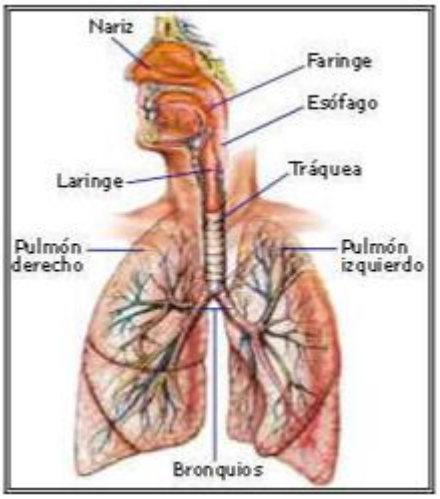

Figura 1. Aparato Fonador Humano

El aparato fonador se divide en tres partes fundamentales:

Cavidades infraglóticas: Está conformada por el diafragma, los pulmones y la tráquea. Tienen como misión proporcionar la corriente de aire respirada, necesaria para producir el sonido.

Cavidad glótica: Está formada por la laringe donde se encuentran las cuerdas vocales, responsables de producir la vibración básica para la generación de la voz. Cuando las cuerdas vocales se encuentran separadas, el aire pasa libremente y prácticamente no se produce ningún tipo de sonido (como en la respiración). Cuando la glotis comienza a cerrase, el aire proveniente de los pulmones que alcanza a atravesarla experimenta una especie de turbulencia, emitiéndose así un ruido. Cuando las cuerdas vocales se cierran completamente, comienzan a vibrar, produciéndose un sonido. La frecuencia e intensidad del sonido que se genera depende de varios factores, algunos de ellos son el tamaño y la masa de las

cuerdas, la tensión que se les aplique o la velocidad del flujo de aire proveniente de los pulmones.

Cavidades supraglóticas: Conformada por 4 cavidades, la faríngea, nasal, bucal y labial.

En la producción del habla intervienen los siguientes elementos:

- Una fuente de energía, la cual proviene del aire a presión que se expulsa en la exhalación.
- Un órgano vibratorio, que son las cuerdas vocales.
- Una caja de resonancia, que son las fosas nasales, la cavidad bucal y la faringe.
- Un sistema de articulación del sonido en el cual entran el paladar, la lengua, los dientes, los labios y las mandíbulas.

A través de la voz, los seres humanos somos capaces de producir sonidos con frecuencias en un rango desde los 100 Hz hasta los 10 kHz. Sin embargo, en una señal de voz tanto la información verbal como la sección más significativa del espectro de la voz se encuentran contenidas a partir de los 300 Hz hasta los 4 kHz. Tomando en cuenta únicamente la inteligibilidad de la voz, entonces el rango de frecuencias que contendría esta información sería desde los 500 Hz a los 2.5 kHz.

Algunos de los parámetros más importantes de la voz son:

- **Tono o *pitch*:** Se refiere al número de veces por segundo que las cuerdas vocales se unen durante la fonación. Los hombres presentan un tono cerca de 110 ciclos por segundo, mientras que las mujeres manifiestan un tono de 180 a 220 ciclos por segundo. El periodo de *pitch* es uno de los parámetros más importantes para la caracterización de la voz.
- **Intensidad de la voz:** Es la amplitud de la señal de voz. La intensidad depende de que tan juntas estén las cuerdas vocales entre sí, de la cantidad de presión de aire por debajo de la laringe, de la frecuencia fundamental de la voz y la resonancia de la estructura del tracto vocal.
- **Timbre:** Es la característica que nos permite diferenciar a un sonido de otro, aunque estos tengan el mismo tono e intensidad. Esta propiedad está determinada principalmente por el contenido harmónico y las dinámicas características del sonido.
- **Formantes:** Son las frecuencias de resonancia del tracto vocal, es decir, frecuencias donde las harmónicas tienen mayores amplitudes.

De acuerdo con la forma en que se producen los sonidos por el tracto vocal, estos pueden clasificarse en dos grandes grupos. Sonidos vocalizados o sonoros y no vocalizados o sordos.

- **Sonidos vocalizados o sonoros:** Se producen por la vibración de las cuerdas vocales cuando el aire pasa a través de las mismas. Tienen un comportamiento cuasi periódico, ciclo vibratorio que comienza cuando el aire separa las cuerdas vocales y termina cuando estas se cierran cortando el aire y liberando un pulso. El tiempo que transcurre entre cada uno de esos pulsos es el periodo fundamental o *pitch* definido anteriormente. Los sonidos vocalizados tienen un alto contenido energético y una estructura armónica fina en frecuencia. Las vocales del idioma español y algunas consonantes tienen este comportamiento.

- **Sonidos no vocalizados o sordos:** Se producen cuando las cuerdas vocales están totalmente abiertas y el aire no encuentra obstrucción al pasar por el tracto vocal. Los sonidos no vocalizados tienen un comportamiento no periódico, además de un bajo contenido energético y un espectro más compensado en frecuencia. Algunas consonantes del idioma español se pueden considerar no vocalizadas.

2.4.Reconocedor de Voz

Los sistemas de reconocimiento de voz generalmente consideran que la señal de voz es la representación de algún mensaje codificado como una secuencia de uno o más símbolos. Para el reconocimiento de dicha secuencia, la señal de voz es segmentada en espacios iguales de tiempo, donde cada segmento es parametrizado, la función de la parametrización consiste en representar los eventos acústicos relevantes de la señal de voz en términos de un conjunto compacto de parámetros.

Para que una señal pueda ser procesada, si la señal es analógica (continua), esta debe ser convertida a una señal digital (valores discretos) haciendo un muestreo en el tiempo y obteniendo por tanto una señal en tiempo discreto y posteriormente cuantificando sus valores para obtener un "*conjunto discreto*". Esta secuencia de vectores parametrizados se asume que formará una representación exacta de la forma de onda de voz basada en la duración cubierta por un solo vector (típicamente de 10 ms). La forma de onda puede ser considerada como estacionaria (que no cambia con el tiempo), a pesar de que esto no es estrictamente verdadero, es una aproximación razonable. Las representaciones paramétricas comúnmente usadas son espectros alisados o coeficientes de predicción lineal, además de otras representaciones derivadas de éstas.

El objetivo de los reconocedores consiste en efectuar un mapeo entre secuencias de vectores de voz y secuencias de símbolos fundamentales. Las dificultades que se presentan son:

- Primero, el mapeo de símbolos de la voz no es uno a uno debido a que diferentes símbolos subyacentes pueden tener sonidos de voz similares. Además, existen grandes variaciones en las ondas de voz debidas a la variabilidad del hablante por su estado de ánimo, medio ambiente, timbre de voz, entre otros.

- Segundo, los límites entre símbolos no pueden ser identificados explícitamente de la forma de onda de la señal de voz. Por lo que no es trivial tratar la forma de onda de voz como una secuencia concatenada de patrones.

La dificultad para delimitar la frontera de las palabras puede reducirse restringiendo la tarea del reconocedor a sólo palabras. Lo cual implica que la forma de onda de voz corresponda a un solo símbolo seleccionado de un vocabulario fijo. A pesar de que este hecho representa un problema simple es un tanto artificial, no obstante tiene un amplio rango de aplicaciones.

2.5.Técnicas para el Reconocimiento de Voz

Existen tres métodos sobresalientes que se enfocan al reconocimiento automático de voz con máquinas **[7]**:

- **El Fonético-acústico.** En este método la máquina intenta decodificar la señal de voz de manera secuencial con base en características acústicas observadas de la señal y las relaciones conocidas entre las características acústicas y los símbolos fonéticos. Sin embargo, se tienen algunos problemas para tener un sistema de reconocimiento de voz exitoso ya que se requiere un conocimiento amplio de las propiedades acústicas de las unidades fonéticas.

- **El de Inteligencia artificial.** Es un método híbrido que explota ideas y conceptos del método acústico fonético y del de reconocimiento de patrones. Involucra conceptos como el de redes neuronales.

- **El de Reconocimiento de Patrones.** En este método los patrones de voz se usan directamente sin determinación de características explícitas. Se tienen dos pasos: entrenamiento de patrones de voz y reconocimiento de patrones de voz a través de la comparación de patrones. Este método es frecuentemente utilizado en reconocedores de voz, debido a la simplicidad de uso, facilidad de entendimiento y por su riqueza en matemáticas y teoría de comunicaciones. Este método es el que se utilizará para el reconocimiento de voz en este proyecto.

2.6.Reconocimiento Empleando Comparación de Patrones

Su principal ventaja inmediata reside en que no es necesario descubrir características espectrales de la voz a nivel fonético, lo que evita desarrollar etapas complejas de detección de *formantes*, rasgos distintivos de los sonidos, tono de voz, etc.

Esto está muy bien para un número finito de palabras, cuyo número no sea muy grande. Si queremos implementar esto para un completo entendimiento de nuestro lenguaje esto sería inútil, por ejemplo si pronunciásemos la palabra *queso* tendríamos que hacerlo exactamente igual que en la grabación. Tendríamos que decir *queso* con la misma velocidad, con el mismo tono...etc. Si la palabra *queso* se pronuncia muy rápido, habría que ajustar los tiempos de inicio y de final. En la **Figura 2** se muestra el proceso de reconocimiento de voz

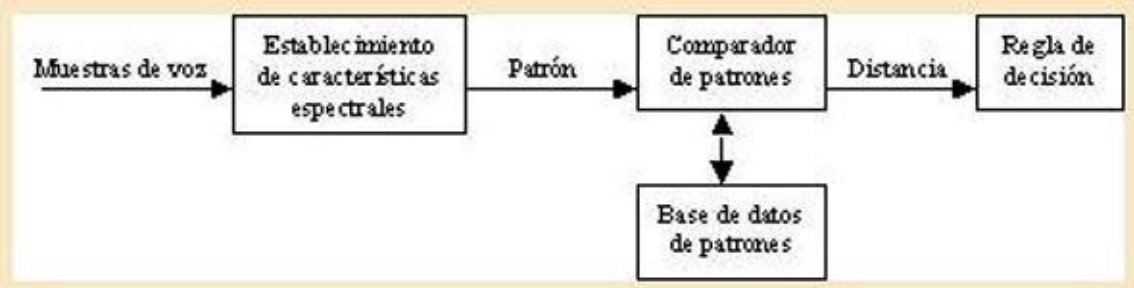

Figura 2. Proceso de Reconocimiento por Comparación de Patrones

Además la base de datos que necesitaríamos sería tan grande, que los sistemas serian ineficientes en cuanto al tiempo de reconocimiento. Cuando el sistema intenta buscar la palabra, basará su funcionamiento en el establecimiento de una distancia matemática entre patrones, de tal manera que se puede calcular lo cercano que se encuentra la palabra pronunciada con cada patrón. Por lo tanto la necesidad de aplicar este sistema es única y exclusivamente a ciertos casos donde el número de palabras necesarias sea pequeño.

También se pueden constituir los grupos de patrones por unidades más básicas como los *fonemas* o silabas. Al grabar estos sonidos en la base de datos, se obtendrán sus características espectrales (patrones) **[8]**. En la **Figura 3** se observa el proceso de comparación de patrones.

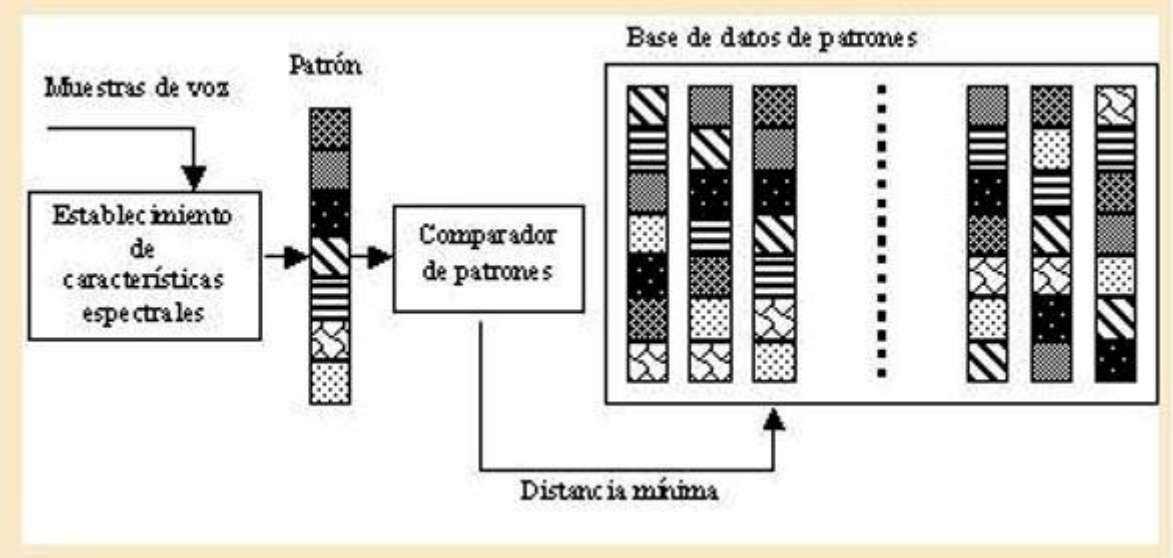

Figura 3. Obtención de la palabra reconocida mediante comparación de patrones

El reconocimiento basado en la técnica de comparación de patrones debe pasar por las siguientes etapas:

- **Adquisición.** La señal a procesar es digitalizada y almacenada en un formato que pueda ser interpretado posteriormente por un sistema de cómputo. La digitalización de la voz se realiza a una frecuencia de muestreo determinada.

- **Preprocesamiento.** Aquí se realizan una serie de pasos para lograr normalizar la señal de voz y poder trabajar con ella posteriormente. Los pasos del preprocesamiento son:

  Detección de inicio y fin: Es necesario implementar un algoritmo detector con el objetivo de eliminar silencios al comienzo y al final de la señal de voz. Esto se realiza a través de un cálculo de energía y detección de umbrales de actividad de la señal.

  Preénfasis: Resalta determinadas componentes de frecuencia relevantes para un determinado procesamiento, por lo tanto tiende a blanquear el espectro y hacerlo más plano removiendo componentes espectrales no deseadas.

  Segmentación y ventaneo: La señal de voz es segmentada a intervalos de 20 a 30 ms, tiempo durante el cual la señal se considera cuasi estacionaria y puede ser aproximada matemáticamente. Luego, es sometida a una ventana de *Hamming* con el objeto de eliminar errores debido a discontinuidades generadas en la segmentación.

- **Extracción de Características.** Aquí es obtenida la información relevante de la señal de voz y capturada en vectores característicos (patrones) con la finalidad de obtener elementos de clasificación que presenten rasgos

distintivos con respecto a los generados por otra palabra o sonido. Un método eficiente para extraer características y que es el más utilizado actualmente en reconocedores comerciales son los Coeficientes Cepstrales en escala de Mel (*MFCC*, por sus siglas en inglés), este método es robusto, además hace uso de la "*Transformada de Fourier*" para obtener las frecuencias de la señal. El objetivo es desarrollar un conjunto de características basadas en criterios perceptuales, diversos experimentos muestran que la percepción de los tonos en los humanos no está dada en una escala lineal, esto hace que se trate de aproximar el comportamiento del sistema auditivo.

- **Clasificación.** En este bloque se lleva a cabo el reconocimiento. La clasificación consiste en comparar el patrón a reconocer (señal de entrada) con una serie de plantillas o patrones que se encuentran en la base de datos, de manera que se identifique al comando de voz en cuestión.

Esto proporciona las bases para introducir el reconocimiento basado en Modelos Ocultos de Markov.

2.7.Modelos Ocultos de Markov (HMM)

Se introdujeron inicialmente a finales de la década de 1960 y principios de los años 1970. Los métodos estadísticos de los Modelos Ocultos de Markov, se han vuelto más populares en los últimos años debido a dos razones principales. Los modelos son muy ricos en estructura matemática y pueden formarse las bases teóricas para usarse en un amplio rango de aplicaciones. Segundo, los modelos al aplicarse apropiadamente para diversas aplicaciones, trabajan muy bien en la práctica.

Dentro de la tecnología del habla, se opta por trabajar en una gran mayoría de casos con Modelos Ocultos de Markov, debido a que requieren menos memoria física, ofrecen mejor tiempo de respuesta y una menor tasa de error con respecto a otras herramientas matemáticas utilizadas en reconocimiento de voz **[7]**. Sin pretender entrar en detalle y de forma simplificada, podemos representar un HMM como un *autómata* de estados finitos como en la **Figura 4** en donde el proceso de transición entre estados está gobernado por probabilidades. Cabe mencionarse que el autómata se rige bajo la probabilidad de transición que presente el fenómeno, es decir, que tan probable es que se mueva al estado siguiente o bien que continúe en el punto de equilibrio. Dentro de la **Figura 4** dicha probabilidad de transición es representada con los números que identifican los enlaces.

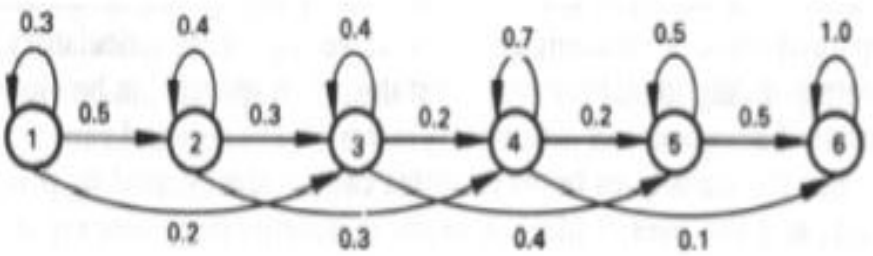

Figura 4. Representación de los Modelos Ocultos de Markov como un autómata

2.7.1. Reconocimiento de Voz Basado en HMM

Los modelos ocultos de Markov son modelos estadísticos que pueden representar procesos aleatorios paramétricos como lo es la señal de voz. Estos son el enfoque más popular y con mayor éxito en el ámbito del reconocimiento de voz.

Un modelo oculto de Markov está compuesto de dos elementos básicos: un proceso de Markov y un conjunto de distribuciones de probabilidad de salida. Los estados del proceso de Markov están ocultos pero son observables de una manera indirecta a partir de la secuencia de vectores con información espectral extraídos de la señal de voz de entrada **[9]**.

Los HMM en reconocimiento de voz se utilizan teniendo en cuenta dos hipótesis:

- La voz se puede dividir en segmentos, estados en los que la señal de voz se puede considerar estacionaria. Es decir, en la ventana de análisis la señal mantiene la estructura de principio a fin. Se asume que las transiciones entre segmentos contiguos son instantáneas. En la **Figura 5** se muestra el proceso de reconocimiento basado en HMM.

- La probabilidad de observación de que un vector de características se genere depende sólo del estado actual y no de símbolos anteriores. Esta es una suposición de Markov de primer orden, denominada hipótesis de independencia. Ninguna de estas hipótesis es cierta para la señal de voz. Sin embargo, hasta el momento, los HMM estándar son los que se utilizan en la mayoría de los reconocedores de voz actuales.

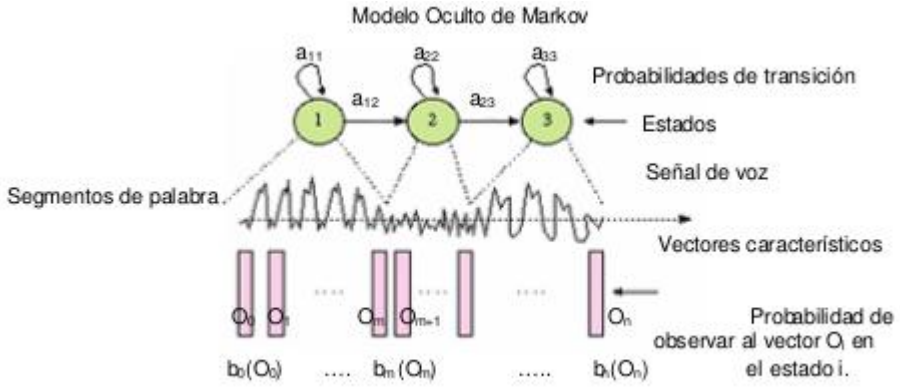

Figura 5. Esquema de funcionamiento del proceso de reconocimiento basado en HMM

2.7.2. Definición Formal de un HMM

Por conveniencia, usamos la notación compacta:

$$\lambda = (A, B, \pi)$$

Este conjunto de parámetros, definen una medida de probabilidad para O, por ejemplo, P(O | λ) .Donde:

- **λ** representa el modelo.
- **A** representa la matriz de distribución de probabilidad de cada estado de transición.
- **B** representa la matriz de observación de los símbolos de la distribución de probabilidad
- **π** representa la matriz de distribución del estado inicial

2.7.3. Los Tres Problemas Básicos de un HMM

Los modelos ocultos de Markov, se caracterizan por tres problemas que hay que resolver para que resulten modelos útiles en aplicaciones reales:

- **Problema 1: Dada la secuencia de observaciones O = O1 O2 . . . OT, y un modelo λ = (A,B,π), ¿Cómo podemos calcular eficientemente P(O/λ), es decir, la probabilidad de la secuencia de observaciones dado el modelo?**
  Consiste en la evaluación del problema, concretamente, dado un modelo y una secuencia de observaciones, cómo calculamos la probabilidad de que la secuencia de observaciones fue producida por el modelo. Podemos ver el problema como la cuantificación de que también un modelo dado empata con la secuencia de observaciones dadas. Este último punto de vista es extremadamente usado. Por ejemplo, si nosotros consideramos el caso en el cual estamos tratando de escoger entre diferentes modelos compitiendo, la solución al problema uno nos permite seleccionar el modelo que empata mejor con las observaciones.

- **Problema 2: Dada la secuencia de observaciones O = O1 O2 . . . OT, y el modelo λ, ¿Cómo seleccionamos una secuencia de observaciones que corresponde a Q=q1 q2 . . . qT, la cual sea óptima?**
  Aquí, intentamos descubrir la parte oculta del modelo, es decir, la secuencia correcta de estados (**Q=q1 q2 . . . qT**).que mejor modele la secuencia de observación dada. El algoritmo de Viterbi permite encontrar la secuencia de estados más probable en un Modelo oculto de Markov a partir de una secuencia de observación.

- **Problema 3: ¿Cómo ajustamos los parámetros del modelo λ= (A, B, π), para maximizar P(O/λ)?.**
  Es en el que intentamos optimizar los parámetros del modelo, de tal forma que describa mejor cómo ocurre una secuencia de observaciones dada. La secuencia de observaciones usada para ajustar los parámetros del modelo se llama secuencia de entrenamiento debido a que es usada para entrenar los Modelos ocultos de Markov.

  Consideremos el siguiente reconocedor de palabras aisladas. Para cada palabra de un vocabulario (de W palabras) queremos diseñar un HMM diferente. Representamos la señal de voz de una palabra dada como una secuencia discreta de vectores que codifican las características esenciales de la palabra. La primera tarea es construir modelos individuales para cada palabra, usando la solución al problema 3 para estimar los parámetros óptimos.

  Para la comprensión del significado de los estados del modelo, usamos la solución al problema 2. Con esto conseguimos segmentar cada modelo en estados, y entonces estudiar las propiedades de los vectores de observación. El objetivo es refinar el modelo para mejorar su capacidad de modelar la secuencia recibida.

  Finalmente, una vez que el conjunto de W modelos han sido diseñados y optimizados, el reconocimiento de una palabra se realiza usando la solución al problema 1. Se asigna una puntuación a cada modelo basada en la secuencia observada de entrada, y se elige como palabra reconocida aquella cuya puntuación sea máxima**[10]**.

CAPÍTULO 3
LA HERRAMIENTA HTK

3.1.Introducción

HTK (Hidden Markov Models ToolKit) es un grupo de módulos para construir y operar Modelos Ocultos de Markov. Aunque puede modelar cualquier tipo de serie temporal de datos, HTK está diseñado para construir herramientas de procesado del habla basadas en HMM, especialmente reconocedores de voz. No obstante, se ha utilizado con éxito en otras tareas (*síntesis* de voz, reconocimiento de caracteres y secuenciación de cadenas de ADN).

Fue originalmente desarrollado en el Grupo de Robótica, Visión y Habla del Departamento de Ingeniería de la Universidad de Cambridge (CUED) a principios de los años 90. Existe una página web de dicha herramienta y en ella se encuentra un manual ampliamente detallado y aportes constantes de los distintos grupos de investigación que están utilizándolo. También está disponible el código fuente en lenguaje C.

El diseño de esta herramienta tiene el propósito principal de construir Modelos Ocultos de Markov para el procesamiento de voz, específicamente para el diseño de reconocedores de voz. Como primer paso las herramientas del HTK se utilizan para estimar los parámetros del conjunto de HMMs utilizando archivos de entrenamiento que contienen la pronunciación y su *transcripción* asociada, Posteriormente las pronunciaciones desconocidas se transcriben a través de las herramientas del HTK para su reconocimiento. La **Figura 6** muestra las etapas de procesamiento para el diseño de Reconocedores de voz.

La mayoría de las funcionalidades del HTK están construidas en módulos de librerías. Las herramientas del HTK están diseñadas para correr bajo el estilo de comandos en línea. Proporcionando el nombre de la herramienta, los archivos y argumentos opcionales; los valores opcionales están siempre separados del nombre de la opción por un espacio. Como ejemplo consideremos la siguiente herramienta hipotética:

HFxx –T 1 –f 34.3 –a –s archivo archivo1 archivo2

Se tienen dos argumentos principales archivo1 y archivo2 y cuatro argumentos opcionales. Las opciones siempre se escriben con el nombre de la opción que corresponde a una letra seguida del valor de la opción. Por ejemplo la opción **–f** es un número real, el valor de la opción **T** es un número entero y el valor de **–s** es una cadena. En el caso de la opción **–a** no sigue ningún valor, ya que corresponde a una

bandera para habilitar o deshabilitar alguna característica de la herramienta. Las opciones cuyo nombre es una letra mayúscula tienen el mismo significado para todas las herramientas. Por ejemplo **–T** se utiliza para controlar la ubicación de salida de la herramienta HTK. La operación de la herramienta se puede controlar por parámetros almacenados en un archivo de configuración. Por ejemplo:

HFxx –C config –f 34.3 –a –s archivo archivo1 archivo2

En este caso a través de la opción **–C**, la herramienta hipotética **HFxx** cargará los parámetros que se encuentran en el archivo de configuración llamado **config** durante el procedimiento de inicialización **[1]**.

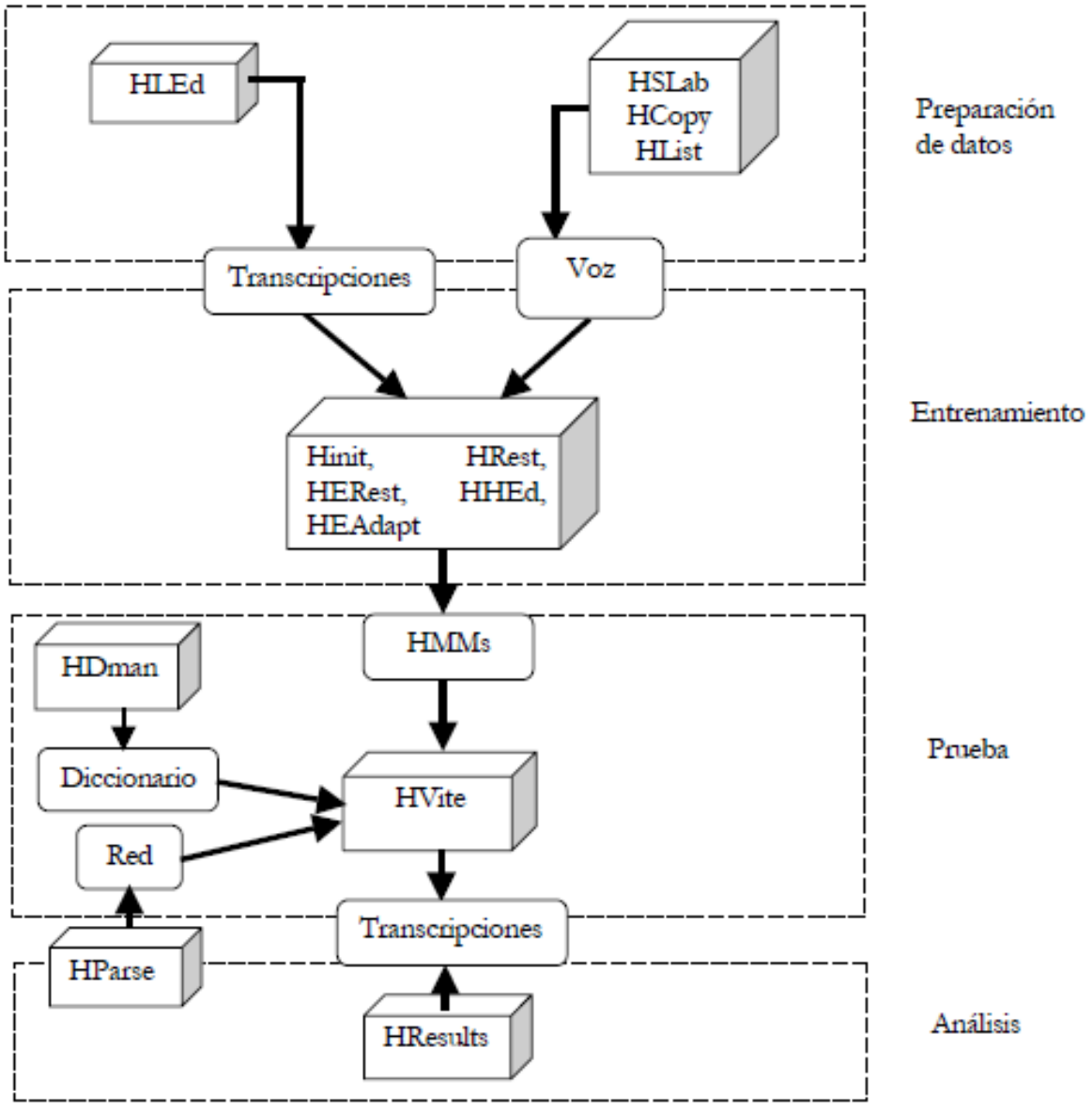

Figura 6. Etapas de procesamiento para el diseño de Sistemas de Reconocimiento de Voz con HTK

3.2.Preparación de Datos

Para construir el conjunto de HMMs, debemos tener un conjunto de archivos de datos (señales de voz) en el formato apropiado y su transcripción asociada. Así mismo se deben tener los *"archivos etiquetados"*.

En caso de requerir realizar el grabado de los archivos de voz, se puede utilizar la herramienta **HSLab**.

La herramienta **HCopy** se utiliza para parametrizar los datos, copia uno o más archivos fuentes en un archivo de salida (concatena archivos), colocando las variables de configuración apropiadas.

HList se puede usar para checar el contenido de cualquier archivo de voz y se puede usar para verificar los resultados de cualquier conversión antes de procesar una gran cantidad de datos.

La herramienta **HLEd** es un editor de etiquetas diseñado para realizar las transformaciones requeridas de los archivos etiquetados. Esta herramienta puede colocar los archivos de salida en un solo Archivo de etiquetas maestro **MLF** (Master Label File), el cual es conveniente para los procesos posteriores.

3.3.Herramientas de Entrenamiento

El siguiente paso en la construcción del sistema consiste en definir la topología (estructura) requerida para cada HMM, escribiendo una definición del prototipo. Se puede utilizar cualquier topología deseada, la definición del HMM se puede almacenar en forma externa como un texto simple a través de un editor de texto. El propósito de la definición del prototipo es solo especificar las características generales de la topología del HMM. Los parámetros serán calculados posteriormente por las herramientas de entrenamiento. El proceso de entrenamiento, se lleva a cabo en etapas. Primero se debe crear un conjunto de modelos inicial.

La herramienta **HInit** y **HRest** proporcionan un entrenamiento del estilo de palabras aisladas. Cada uno de los HMMs requeridos se genera individualmente. **HInit** lee en todos los datos de entrenamiento y deja todos los ejemplos del *fonio* requerido. Después calcula iterativamente un conjunto inicial de valores de parámetros utilizando el procedimiento *k-media* segmentada. En el primer ciclo, los datos de entrenamiento están segmentados uniformemente, cada estado del modelo se empata con el correspondiente segmento de dato y se estima la media y la varianza. En el segundo ciclo, así como en los sucesivos, la segmentación uniforme se reemplaza por el alineamiento usando el *"algoritmo de Viterbi"*. Los valores de los parámetros iniciales calculados por **HInit** son más adelante reestimados por la herramienta **HRest**. Nuevamente los datos etiquetados se

utilizan, pero esta vez el procedimiento k-media segmentada se reemplaza por el procedimiento de reestimación Baum-Welch.

Una vez creado el conjunto inicial de modelos, la herramienta **HERest** se utiliza para realizar un entrenamiento integrado utilizando el conjunto de entrenamiento completo. **HERest** realiza una simple reestimación Baum Welch del conjunto total de modelos de fonios HMM simultáneamente. Para cada expresión de entrenamiento, los modelos de fonios correspondientes se concatenan y después el algoritmo forward-backward se utiliza para acumular las estadísticas de ocupación de estados, medias y varianzas para cada Modelo Oculto de Markov en la secuencia. Cuando se han procesado todos los datos de entrenamiento, las estadísticas acumuladas se utilizan para calcular la reestimación de los parámetros de los Modelos ocultos de Markov.

La filosofía de la construcción de un sistema en HTK consiste en que los HMMs deben ser refinados incrementalmente. Es decir una progresión típica se inicia con un simple conjunto de Gausianas de modelos de fonios independientes del contexto, y después iterativamente se refinan expandiéndolas para incluir dependencia del contexto y usar múltiples mezclas de componentes de distribución Gausiana. La herramienta **HHEd** es un editor de definición de HMM la cual clonará los modelos en conjuntos dependientes del contexto, aplicando una variedad de parámetros uniendo e incrementando el número de componentes mezclados en distribuciones específicas. El proceso consiste en modificar un conjunto de HMMs en estados utilizando **HHEd** y después reestimar los parámetros del conjunto modificado utilizando **HERest** después de cada estado. Para mejorar el desempeño para hablantes específicos, la herramienta **HEAdapt** y **HVite** se puede utilizar para adaptar HMMs a mejores modelos de características de hablantes particulares utilizando pequeñas cantidades de datos de entrenamiento o adaptación. El resultado final será un sistema adaptado.

3.4.Herramienta de Reconocimiento

HTK proporciona una herramienta de reconocimiento llamada **HVite**, la cual usa un algoritmo de Viterbi para realizar el reconocimiento de voz. **HVite** toma como entrada una red, describiendo la secuencia de palabras permitida, la definición del diccionario definiendo cómo se debe pronunciar cada palabra y un conjunto de HMMs. Su forma de operación consiste en convertir la red de palabras para una red de fonios y después unir a la definición de HMM adecuada para cada instancia de fonio. El reconocimiento se realiza para un conjunto de archivos de voz almacenados.

Las redes de palabras necesarias para manejar **HVite** son generalmente ciclos (*loops*) simples de palabras en las cuales cualquier palabra puede seguir de cualquier otra. La herramienta **HParse** se utiliza para convertir el lenguaje de

especificación de la gramática en una notación equivalente de red de palabras. Finalmente, se requiere de la construcción de un diccionario a través de la herramienta **HDMan**.

3.5.Herramienta de Análisis

Cuando ya se ha construido un reconocedor con base en HMM, se necesita evaluar su desempeño. Esto generalmente se realiza utilizando el reconocedor para transcribir algunas oraciones de prueba pregrabadas y conjuntando la salida del reconocedor con las referencias correctas de las transcripciones. Esta comparación se realiza con la herramienta **Hresults**, la cual utiliza programación dinámica para alinear las dos transcripciones, después cuenta errores de sustituciones, eliminaciones e inserciones. Esta herramienta presenta la opción de generar una matriz de confusión.

CAPÍTULO 4

DESCRIPCIÓN DEL SISTEMA SRCV

En el presente capítulo se describe la estructura del Sistema Reconocedor de Voz, así como cada uno de los bloques que constituye a dicho Sistema.

4.1. Descripción Conceptual

El sistema SRCV es capaz de aceptar palabras simples, vía un micrófono, como respuesta a preguntas sencillas hechas mediante voces grabadas apoyadas con gráficos comprensibles por los usuarios (niños en edad preescolar). El SRCV analiza las respuestas con base en un reconocedor de voz apoyado en un banco de reactivos y una base de datos de patrones de voz.
En la **Figura 7** se muestra el diagrama conceptual del SRCV.

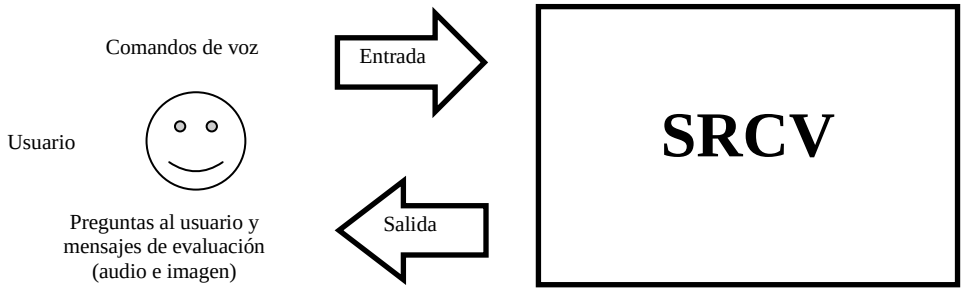

Figura 7. Diagrama conceptual del SRCV

4.2. Estructura Interna

La estructura interna del SRCV está compuesta por los siguientes bloques:

- Interfaz Gráfico-acústica
- Reconocedor de Comandos de Voz
- Intérprete de Comandos
- Generador de Preguntas y Respuestas

Interfaz Gráfico-acústica: Bloque encargado de interactuar directa y fácilmente con el usuario, mostrando una serie de preguntas previamente grabadas y apoyadas con objetos gráficos de fácil comprensión (formas geométricas, símbolos alfanuméricos y colores, entre otros), y aceptando respuestas de voz.

Reconocedor de Comandos de Voz: A partir de las respuestas de voz dadas por el usuario, este bloque se encarga de identificar la respuesta recibida, utilizando una base de datos.

Intérprete de Comandos: Su trabajo es analizar y verificar si las respuestas del usuario son correctas. La función de este bloque está apoyada por un Banco de Reactivos.

Generador de Preguntas y Respuestas: Este bloque utiliza un Banco de Reactivos para seleccionar y enviar a la interfaz gráfico-acústica la pregunta a mostrar al usuario. También se encarga de enviar a la interfaz gráfico-acústica los mensajes de evaluación de las respuestas del usuario.

4.3.Funcionamiento del SRCV

El sistema presenta al *usuario* una **Interfaz Gráfico-acústica** que le permitirá interactuar con el sistema mediante *comandos de voz*. Esta interfaz hará preguntas sencillas al *usuario* (niño en etapa preescolar), mediante voz y figuras en color. Las preguntas son enviadas a la interfaz por parte de un **Generador de Preguntas y Respuestas** que a su vez utiliza un **Banco de Reactivos** que contiene un conjunto de preguntas con sus respectivas respuestas. Las respuestas del usuario, son recibidas y transferidas por la interfaz gráfico-acústica hacia el **Reconocedor de Comandos de Voz**. En esta etapa se comparan los *comandos de respuesta* contra *patrones de referencia* almacenados en una **Base de Datos de Patrones de Comandos de Voz**.

Si la respuesta corresponde a un patrón de la base de datos se enviará el *comando de voz reconocido* al **Intérprete de Comandos**, de lo contrario se enviará un *comando de no reconocimiento*. El intérprete de comandos recibe, del generador de preguntas y respuestas, el *identificador del reactivo (IR)* que ha sido enviado al usuario para que busque la respuesta correspondiente a dicho reactivo y la compare contra el comando de voz reconocido. De dicha comparación se obtiene un *resultado de evaluación* que es enviado hacia el generador de preguntas y respuestas el cual produce un mensaje a la interfaz gráfico-acústica para que el usuario reciba la evaluación de su respuesta.

En la **Figura 8** se muestra el funcionamiento de trabajo del SRVC.

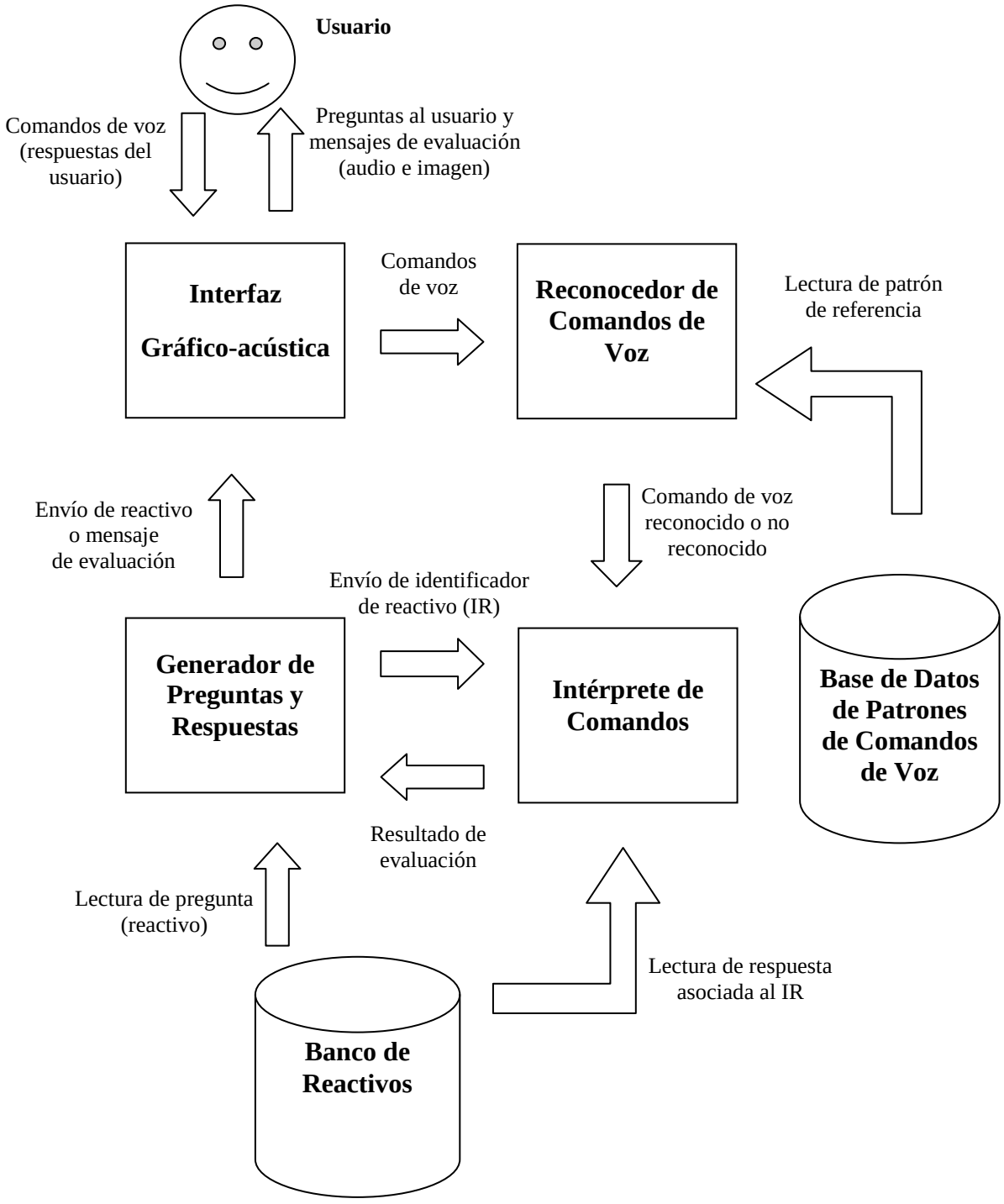

Figura 8. Diagrama de bloques del SRCV

El bloque encargado del reconocimiento de voz es desarrollado con la ayuda de librerías que contiene la herramienta HTK, con las que se apoyaron las tareas de reconocimiento y captura de voz principalmente.

4.4.Especificaciones Técnicas

Interfaz Gráfico/acústica: Las entradas a este módulo son dos: la pregunta (reactivo seleccionado) o la respuesta al usuario (mensaje de evaluación), enviada por el generador de preguntas y respuestas en modo gráfico-acústico, y la respuesta de voz generada por el usuario (pronunciando algún número del 0 al 9, color, figura geométrica, posición, etc.). Asimismo, existen dos salidas de este módulo: una de ellas corresponde a la generación de una pregunta que deberá recibir el usuario en forma gráfico-acústica, y la otra salida son los comandos de voz que se envían al reconocedor de comandos de voz. Los datos acústicos de salida son voces grabadas que complementan las respuestas al usuario con gráficos mostrados en la pantalla.

Reconocedor de Comandos de Voz: Una de las entradas recibe archivos de audio tomados de la base de datos de patrones de comandos de voz. Otra entrada recibe también archivos de audio correspondientes a las respuestas pronunciadas por el usuario. Este módulo reconocedor de voz contiene una salida hacia el intérprete de comandos, encargado de enviar un número identificador del comando de voz reconocido, o en su caso, un valor que indique que la respuesta del usuario es irreconocible.

Intérprete de Comandos: Este módulo requiere tres entradas: una recibe el número identificador del comando de respuesta reconocido, otra entrada recibe un número identificador de la respuesta correcta, asociada con el número identificador del reactivo en uso, que es recibido por una tercera entrada. Como salida se genera un valor (verdadero o falso) que le indica al generador de preguntas y respuestas si la respuesta es correcta o incorrecta.

Generador de Preguntas y Respuestas: Tiene dos entradas: en una se recibe un reactivo seleccionado bajo algún criterio establecido previamente, y en la otra se recibe un valor (verdadero o falso) correspondiente a la respuesta del usuario. Asimismo, se tiene una salida que sirve para enviar el número identificador del reactivo en uso al intérprete de comandos y otra salida encargada de enviar a la interfaz gráfica los datos de los objetos a graficar en la pantalla y el archivo de voz asociado a dichos gráficos (ya sea durante la realización de preguntas al usuario o evaluación de sus respuestas).

CAPÍTULO 5

DISEÑO DEL SISTEMA SRCV

5.1. Base de Datos de Patrones de Comandos de Voz

Para obtener los patrones que se utilizan durante el proceso de reconocimiento primeramente hay que grabar las señales de voz de las palabras a las cuales se les extraerán dichos patrones característicos. Las palabras grabadas fueron "si", "no", "quiero salir", números del 1 al 10, los colores, figuras geométricas, las letras, y algunos objetos como libro, lápiz, reloj, vaso, etc. Estas grabaciones se realizaron en un entorno libre de ruido (ambiente controlado) utilizando la herramienta **HSLab** de HTK bajo el formato de audio .SIG (formato propio de HTK), esto se explicará en el siguiente apartado 5.2 Construcción de un Reconocedor con HTK. Cada palabra fue cuidadosamente grabada en medio de un silencio de inicio y un silencio final. Las palabras grabadas en la base de datos de entrenamiento, se repitieron 20 veces cada una. En general para la base de datos de entrenamiento se grabaron, aproximadamente, 1300 señales de voz. Mencionar que las señales de voz pertenecen a un solo interlocutor masculino.

Teniendo las grabaciones correspondientes a las palabras que estarán incluidas en el vocabulario del reconocedor podemos comenzar con la construcción del mismo. La lista de las palabras reconocidas en el sistema pueden consultarse en el Anexo 2.

5.2. Reconocedor de Comandos de Voz

Nuestro objetivo es centrarnos en un reconocedor simple de palabras aisladas. Se describirán los pasos necesarios para la creación de un reconocedor basado en palabras aisladas utilizando la herramienta HTK **[11]**. Las etapas principales son las siguientes:

- Creación de una base de datos de entrenamiento. Cada elemento del vocabulario se graba varias veces y se etiqueta con su correspondiente palabra, para crear un *corpus*.

- Análisis acústico. Las formas de onda grabadas deben ser convertidas a vectores de coeficientes MFCC.

- Definición de los modelos. Debemos definir un prototipo de HMM para cada elemento del vocabulario.

- Entrenamiento de los modelos. Cada HMM es inicializado y entrenado utilizando las etiquetas y vectores de coeficientes.

- Definición de la gramática. Se define qué puede ser reconocido por la aplicación.

- Evaluación. Se evalúa el reconocedor con entrada de micrófono o con un conjunto de secuencias de prueba.

5.2.1. Organización del Espacio de Trabajo

Aunque no es necesario, se recomienda crear en el directorio donde se encuentran los ejecutables de las herramientas HTK, la estructura de directorios mostrada en la **Figura 9** y que se describe a continuación:

- **datos** almacenará los datos relativos al entrenamiento. Estos datos son las grabaciones voz (**datos\sig**), las etiquetas (**datos\lab**) y los vectores de coeficientes MFCC (**datos\mfcc**).

- **análisis** almacenará archivos de configuración (.conf) que serán utilizados para la etapa de análisis acústico (extracción de características para obtener los patrones de voz) y reconocimiento.

- **entrenamiento** almacenará una lista de los archivos que contienen los vectores de coeficientes que se utilizarán para el entrenamiento.

- **modelo** almacenará los HMMs del reconocedor. Contendrá una carpeta para los prototipos (**modelo\proto**), una para los modelos inicializados (**modelo\hmm0**) y otra para los modelos entrenados (**modelo\hmm1**).

- **definicion** almacenará los archivos que definen la gramática y el vocabulario.

- **Prueba** almacenará los archivos que conciernen a las pruebas: aquí se almacenarán las señales (**prueba\sig**) y los vectores MFCC (**prueba\mfcc**) que usaremos para las pruebas.

Figura 9. Estructura del espacio de trabajo

5.2.2. Creación del Corpus de Voz

Primero, debemos grabar las señales de voz con las que entrenaremos los modelos. Cada señal debe ser etiquetada, es decir, asociada con un texto que describa su contenido. La grabación y el etiquetado pueden hacerse con **HSLab**, aunque pueden usarse aplicaciones externas. La **Figura 10** muestra este proceso.

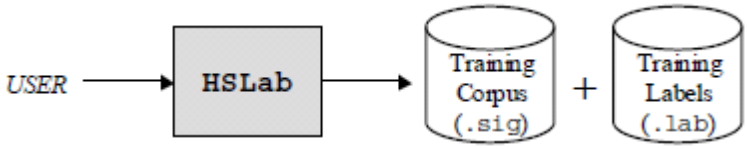

Figura 10. Proceso de grabación y etiquetado

Para crear y etiquetar archivos de voz, nos situamos en el directorio de HTK y ejecutamos en la línea de comandos:

HSLab cero.sig

donde cero.sig es el nombre del archivo que se creará. Con esto aparecerá la interfaz gráfica de **HSLab** que se muestra en la **Figura 11**.

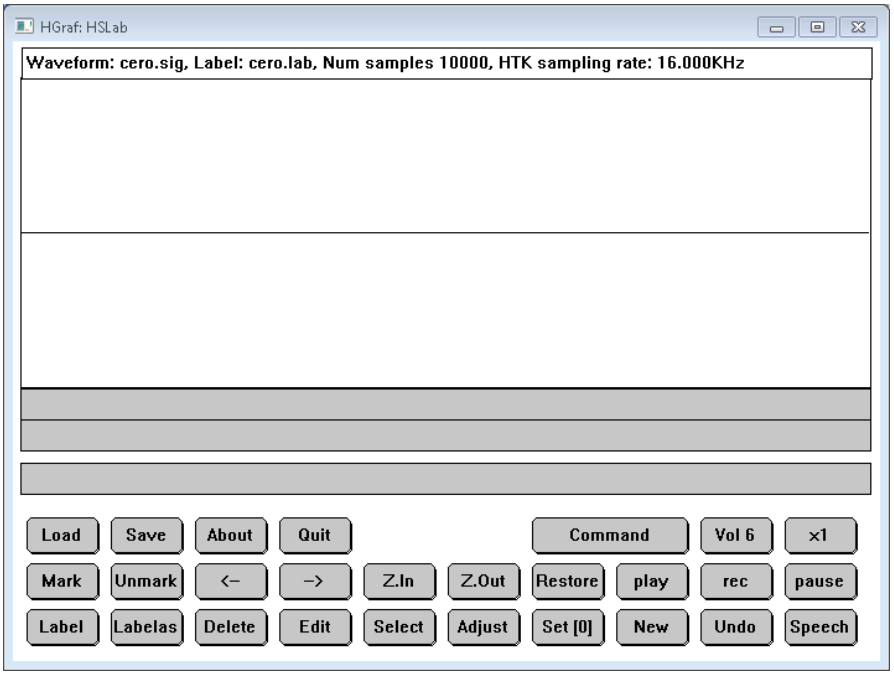

Figura 11. Interfaz de la herramienta HSLab

Presionamos el botón de *rec*, grabamos la señal y pulsamos *stop*. Automáticamente se crea un archivo temporal llamado **cero_0.sig** en el directorio actual. Si realizamos una nueva grabación, se graba en un segundo archivo temporal llamado **cero_1.sig**. Los archivos de señal se almacenan en un formato específico de HTK (.sig).

Para etiquetar las formas de onda grabadas, pulsamos *Mark*, elegimos la región que queremos etiquetar y pulsamos el botón *Labelas*. Ahora escribimos el nombre de la etiqueta y pulsamos *intro*. A parte de la palabra o secuencia de palabras que hayamos grabado, debe haber un silencio inicial y otro final. Los etiquetaremos con la etiqueta *sil*. Repetimos el proceso asignando etiquetas a las distintas palabras que se han dicho, pulsando sobre *play* para oír la zona marcada cuando sea necesario. En la **Figura 12** puede verse la interfaz de **HSLab** con la grabación y etiquetado de la palabra cero.

Las regiones etiquetadas no se deben solapar entre sí, pero puede haber un pequeño salto entre ellas. Los silencios entre palabras no hacen falta etiquetarlos. Cuando toda la señal ha sido etiquetada, pulsamos *Save* e *intro*, con lo que se creará un archivo de etiquetas llamado **cero_0.lab**. Tras esto, podemos cerrar la herramienta, pulsando *Quit*.

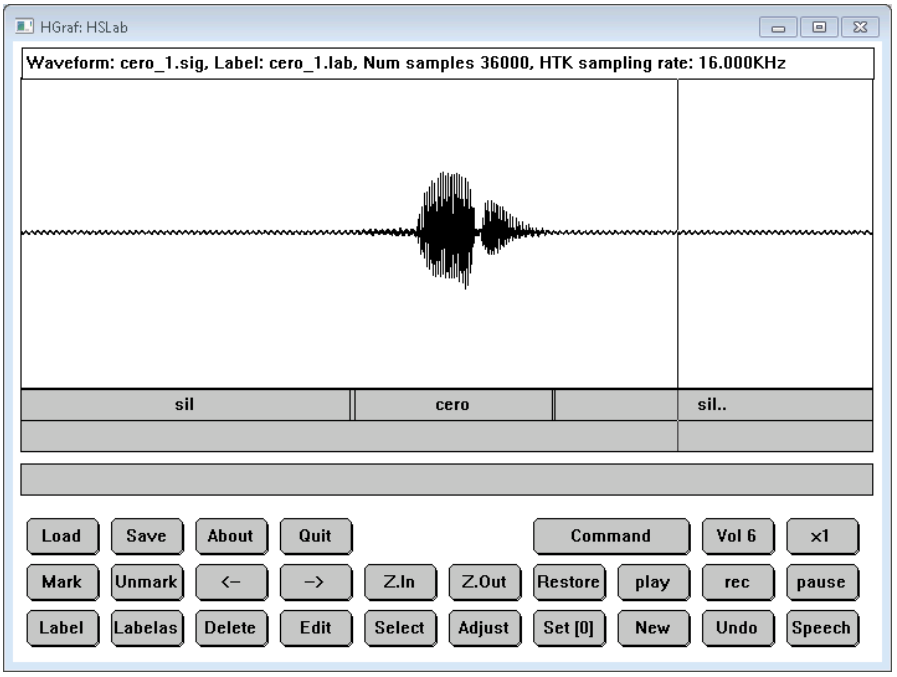

Figura 12. Interfaz de HSLab, grabación y etiquetado de una señal de voz

Tras haber grabado y etiquetado la señal de voz, renombramos los archivos .sig y .lab generados como nos convenga. Al final, todos los archivos .sig que vayamos a utilizar en el entrenamiento de los modelos irán en el directorio **datos\sig** y los archivos .lab a la carpeta **datos\lab**.

5.2.3. Análisis Acústico

Las herramientas de reconocimiento del habla no pueden procesar rectamente las formas de onda de la voz. Tienen que ser representadas de una manera más eficiente y compacta, es decir se tienen que extraer vectores de coeficientes de cada señal de voz. La conversión se realiza mediante la herramienta **HCopy**. Se muestra el proceso de extracción en la **Figura 13**.

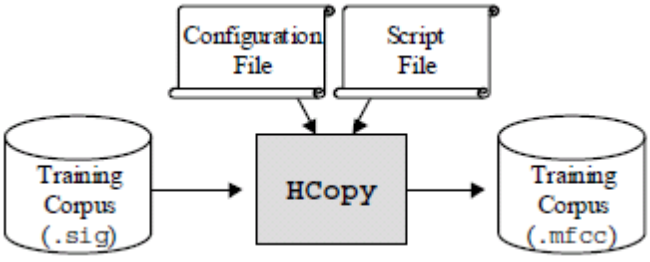

Figura 13. Proceso de extracción de los vectores de coeficientes

El comando a ejecutar es el siguiente:

HCopy -C analisis\analisis.conf -S datos\lista.txt

El parámetro **-C** indica que **HCopy** cargará parámetros almacenados en un archivo de configuración. El parámetro **–S** indica que se utilizará una lista de archivos. Los archivos implicados deben estar preparados antes de ejecutar **HCopy**.

- **analisis.conf** es el archivo de configuración.

- **lista.txt** es una lista de parejas fuente-destino que indica la conversión a realizar desde las grabaciones de voz a los vectores MFCC.

Contenido de analisis.conf

El archivo de configuración es un documento de texto (pueden añadirse comentarios mediante el símbolo #). En nuestros ejemplos utilizaremos el archivo mostrado en la **Figura 14.**

```
#
# Example of an acoustical analysis configuration file
#
SOURCEFORMAT = HTK            # Gives the format of the speech files
TARGETKIND = MFCC_0_D_A       # Identifier of the coefficients to use

# Unit = 0.1 micro-second :
WINDOWSIZE = 250000.0         # = 25 ms = length of a time frame
TARGETRATE = 100000.0         # = 10 ms = frame periodicity

NUMCEPS = 12                  # Number of MFCC coeffs (here from c1 to c12)
USEHAMMING = T                # Use of Hamming function for windowing frames
PREEMCOEF = 0.97              # Pre-emphasis coefficient
NUMCHANS = 26                 # Number of filterbank channels
CEPLIFTER = 22                # Length of cepstral liftering

# The End
```

Figura 14. Contenido del archivo de configuración para la extracción de características

Aunque pueden ponerse directamente en la línea de comandos de **HCopy**, es más práctico crear un archivo de texto que liste las parejas origen-destino que debe transformar la herramienta (ver **Figura 15**). El archivo se especifica con el argumento **-S**. El origen serán los archivos de voz grabados con anterioridad (.sig) y el destino los archivos de coeficientes espectrales que se crearán (.mfcc).

```
datos/sig/cero_1.sig datos/mfcc/cero_1.mfcc
datos/sig/cero_2.sig datos/mfcc/cero_2.mfcc
datos/sig/cero_3.sig datos/mfcc/cero_3.mfcc
datos/sig/cero_4.sig datos/mfcc/cero_4.mfcc
datos/sig/cero_5.sig datos/mfcc/cero_5.mfcc
datos/sig/cero_6.sig datos/mfcc/cero_6.mfcc
datos/sig/cero_7.sig datos/mfcc/cero_7.mfcc
datos/sig/cero_8.sig datos/mfcc/cero_8.mfcc
datos/sig/cero_9.sig datos/mfcc/cero_9.mfcc
datos/sig/cero_10.sig datos/mfcc/cero_10.mfcc
datos/sig/cero_11.sig datos/mfcc/cero_11.mfcc
datos/sig/cero_12.sig datos/mfcc/cero_12.mfcc
datos/sig/cero_13.sig datos/mfcc/cero_13.mfcc
datos/sig/cero_14.sig datos/mfcc/cero_14.mfcc
datos/sig/cero_15.sig datos/mfcc/cero_15.mfcc
datos/sig/cero_16.sig datos/mfcc/cero_16.mfcc
datos/sig/cero_17.sig datos/mfcc/cero_17.mfcc
datos/sig/cero_18.sig datos/mfcc/cero_18.mfcc
datos/sig/cero_19.sig datos/mfcc/cero_19.mfcc
datos/sig/cero_20.sig datos/mfcc/cero_20.mfcc
datos/sig/uno_1.sig datos/mfcc/uno_1.mfcc
datos/sig/uno_2.sig datos/mfcc/uno_2.mfcc
```

Figura 15. Parte del contenido de lista.txt para la extracción de características

Al ejecutar el comando, se generarán en el directorio **datos\mfcc** los archivos listados arriba que contienen los vectores de coeficientes MFCC que usaremos en el reconocimiento.

5.2.4. Definición de los HMM

Debemos modelar cada una de las distintas palabras del vocabulario sin olvidar el silencio utilizando un HMM. En HTK, un HMM se define en un fichero de texto (ver **Figura 16**). El prototipo que usaremos se almacena en **modelo\proto** y se llamará **hmm_x**, donde "**x**" es la palabra a la que modela. Habrá un prototipo por cada palabra además del prototipo que modela al silencio. Su contenido es el siguiente:

```
~o <VecSize> 39 <MFCC_0_D_A>
~h "x"
<BeginHMM>
      <NumStates> 6
      <State> 2
            <Mean> 39
                  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
                  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
                  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
            <Variance> 39
                  1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
                  1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
                  1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      <State> 3

      (…)

      <State> 5
            <Mean> 39
                  0.0 0.0 (...) 0.0
            <Variance> 39
                  1.0 1.0 (...) 1.0
      <TransP> 6
            0.0 0.5 0.5 0.0 0.0 0.0
            0.0 0.4 0.3 0.3 0.0 0.0
            0.0 0.0 0.4 0.3 0.3 0.0
            0.0 0.0 0.0 0.4 0.3 0.3
            0.0 0.0 0.0 0.0 0.5 0.5
            0.0 0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Figura 16. Definición del prototipo de un HMM

Comentaremos el significado de cada línea:

- **~o <VecSize> 39 <MFCC_0_D_A>** indica la longitud del vector de coeficientes y el tipo de coeficientes.
- **~h "x" <BeginHMM> (...) <EndHMM>** encierra la definición del HMM llamado "x".

- **<NumStates> 6** especifica el número total de estados, los cuatro activos y los dos transitorios.

- **<State> i** introduce a la especificación de las características del estado i-ésimo. La matriz diagonal que representa la probabilidad de generación en ese estado puede especificarse poniendo simplemente los elementos de la diagonal: un vector de medias y otro de varianzas para las 39 gaussianas. Los estados 1 y 6 no se describen, pues no tienen función de observación.

- **<Mean> 39** especifica el vector de medias de 39 elementos de la función de observación del estado al que acompaña. Estos coeficientes serán entrenados posteriormente y su valor cambiará, pero ahora, que estamos definiendo la estructura, son inicializados a cero.

- **<Variance> 39** especifica el vector de varianzas de 39 elementos de la función de observación del estado al que acompaña. Son inicializados a uno, aunque serán entrenados posteriormente.

- **<TransP> N** especifica la matriz (**Figura 17**) de transición de estados, de dimensiones N × N :

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{bmatrix}$$

Figura 17. Matriz de transición de estados

- Los valores nulos indican transiciones no permitidas entre estados. El resto de valores pueden ser inicializados arbitrariamente, pero los elementos de cada línea deben sumar 1. Se recomienda, en la medida de lo posible, que todos los elementos no nulos de cada línea tengan el mismo valor, para que ninguna transición parta con ventaja en el proceso de entrenamiento.

- Este prototipo debe copiarse para cada palabra que vayamos a usar, simplemente cambiando el nombre del archivo (**hmm_x**) y la cabecera (**~h "x"**). Todos estos modelos se almacenarán en **modelo\proto** que será el punto de partida para entrenar los modelos.

La definición del prototipo puede verse gráficamente en la **Figura 18**

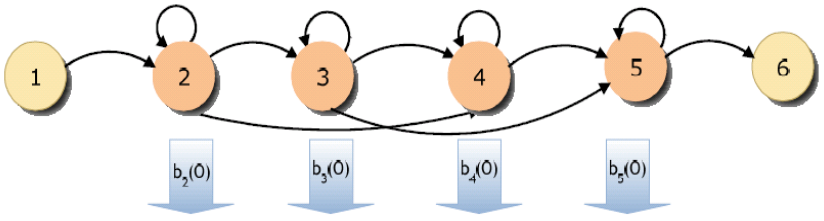

Figura 18. Prototipo de un HMM como autómata

5.2.5. Entrenamiento de los Modelos

Antes de comenzar a entrenar los modelo debe llevarse a cabo el proceso de inicialización (ver **Figura 19**), los parámetros de los HMM deben ser inicializados correctamente utilizando los datos de entrenamiento con el objetivo de permitir que el algoritmo de entrenamiento converja rápidamente. Para la inicialización usaremos **HInit**.

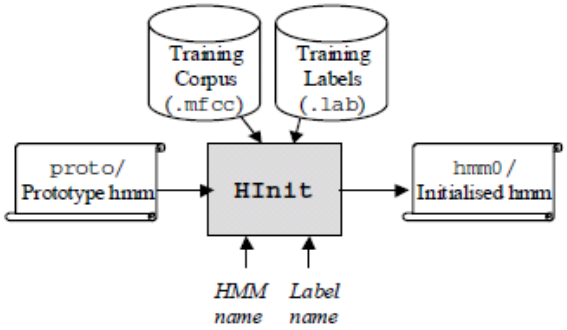

Figura 19. Proceso de Inicialización

La ejecución en línea de comandos es la siguiente:

HInit -S entrenamiento\entrenamiento.txt -M modelo\hmm0 –H modelo\proto\hmm_x -l xlab -L datos\lab\ x

Esta línea debe repetirse para cada uno de los HMM que queramos inicializar.

- **x** es el nombre del HMM que usaremos.
- **modelo\proto\hmm_x** es el nombre del fichero de definición de un HMM, creado en el paso anterior, con el parámetro **–H** indicamos que **modelo\proto** es el directorio fuente.
- **modelo\hmm0** es el directorio donde se guardaran los modelos inicializados, con el parámetro **–M** indicamos que **modelo\hmm0** es el directorio destino.
- **-L datos\lab** indica el directorio donde almacenamos las etiquetas.
- **-l xlab** indica qué etiqueta debe ser utilizada.
- **entrenamiento\entrenamiento.txt** es una lista completa de los vectores de MFCC que vamos a usar en el entrenamiento.

En la **Figura 20** se observa parte del contenido del archivo entrenamiento.txt:

```
datos/mfcc/cero_1.mfcc
datos/mfcc/cero_2.mfcc
datos/mfcc/cero_3.mfcc
datos/mfcc/cero_4.mfcc
datos/mfcc/cero_5.mfcc
datos/mfcc/cero_6.mfcc
datos/mfcc/cero_7.mfcc
datos/mfcc/cero_8.mfcc
datos/mfcc/cero_9.mfcc
datos/mfcc/cero_10.mfcc
datos/mfcc/cero_11.mfcc
datos/mfcc/cero_12.mfcc
datos/mfcc/cero_13.mfcc
datos/mfcc/cero_14.mfcc
datos/mfcc/cero_15.mfcc
datos/mfcc/cero_16.mfcc
datos/mfcc/cero_17.mfcc
datos/mfcc/cero_18.mfcc
datos/mfcc/cero_19.mfcc
datos/mfcc/cero_20.mfcc
datos/mfcc/uno_1.mfcc
datos/mfcc/uno_2.mfcc
```

Figura 20. Parte del contenido de entrenamiento.txt para las etapas de inicialización y re-estimación

El siguiente paso consiste en obtener los modelos entrenados que serán utilizados en la etapa de reconocimiento.

Para ello se utiliza la herramienta **HRest,** que estima los valores óptimos de los parámetros de los HMM (probabilidades de transición, vectores de medias y varianzas de cada función de observación), ver **Figura 21**. Cada vez que la ejecutemos, se producirán varias iteraciones del método de Baum-Welch hasta que converja.

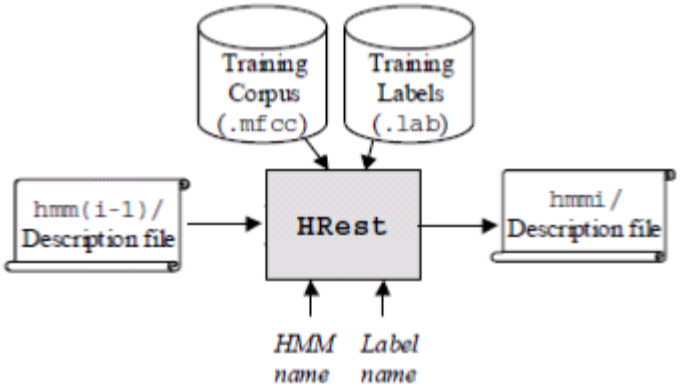

Figura 21. Proceso de re-estimación

El comando es el siguiente:

HREST -S entrenamiento/entrenamiento.txt -M modelo/hmm1 –H modelo/hmm0/hmm_x -l xlab -L datos/lab/ x

Al igual que en la inicialización, éste comando debe ejecutarse para cada HMM.

5.2.6. Definición de la Gramática

Se define la secuencia de palabras que pueden reconocerse. En HTK la gramática se escribe en un fichero de texto, acorde con ciertas reglas sintácticas como se muestra en la **Figura 22**. En este caso será bastante simple ya que se trata de la gramática de un reconocedor de palabras aisladas. Ejemplo:

```
$palabra = palabra1 | palabra2 | palabra 3 | (…);

( { silencio } [ $palabra ] { silencio } )
```

Figura 22. Ejemplo de gramática para un reconocedor de palabras aisladas en HTK

Esto dice a HTK que la variable **$palabra** puede remplazarse por palabra1, palabra2, palabra3, etc.

Las llaves **{ }** denotan cero o varias repeticiones (sin silencio, un silencio corto o largo). Los corchetes **[ ]** alrededor de $palabra denotan cero o una ocurrencia (si no se pronuncia ninguna palabra, es posible reconocer sólo silencio). El símbolo **|** denota alternativa y los paréntesis **( )** encierran la sintaxis de la gramática.

La gramática descrita no puede ser usada por HTK y antes de utilizarla para ejecutar el reconocedor, debe ser compilada con el módulo **HParse** para crear una red gramatical.

El comando a ejecutar es el siguiente:

HParse definicion\gramatica.txt definicion\red.slf

Creación de un diccionario

El sistema debe saber a qué HMM corresponde cada una de las variables del archivo de gramática. Estas variables se almacenan en el directorio definición en un archivo de texto llamado diccionario.txt, un ejemplo de su contenido se observa en la **Figura 23**:

```
palabra1 nombre_HMM_palabra1
palabra2 nombre_HMM_palabra2
palabra3 nombre_HMM_palabra3
.
.
.
```

Figura 23. Ejemplo del contenido un diccionario de palabras en HTK

- Los elementos de la columna izquierda se refieren a los nombres de las variables en el fichero de gramática.
- Los elementos de la columna derecha se refieren a los nombres de los HMM (introducido por los **~h** en los archivos de definición de los HMM).

5.2.7. Reconocimiento

Reconocimiento Usando una Entrada Grabada
Cuando vamos a pasar al reconocedor una secuencia previamente grabada, dividimos el proceso en dos pasos como se observa en la **Figura24.**

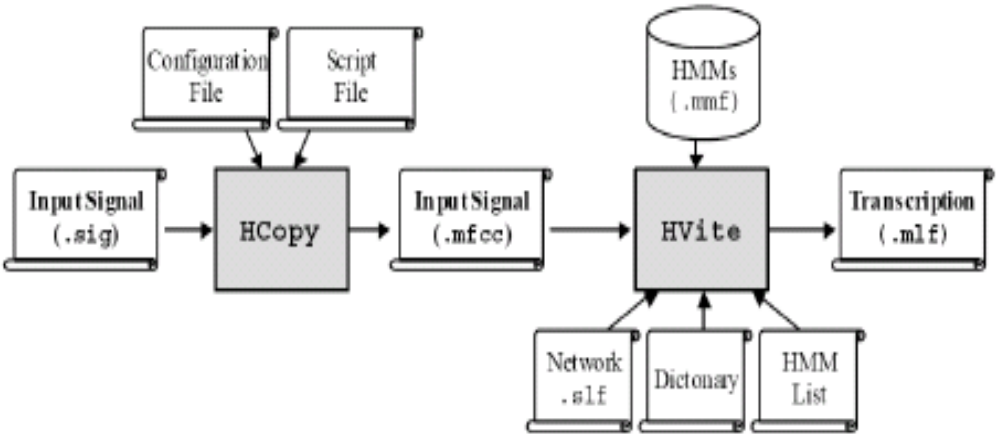

Figura 24. Proceso de reconocimiento con entrada de voz grabada

El archivo grabado es guardado en **prueba\sig**, y es transformado en una serie de vectores acústicos (MFCC) al igual que hicimos para las secuencias de entrenamiento (con **HCopy**). No etiquetamos la señal, pues vamos a usarla para que el reconocedor la transcriba. El resultado se almacena en **prueba\mfcc**.

El cálculo se realiza mediante la herramienta **HVite**:

HVITE -H modelo\hmm1\hmm_x -i prueba\reco.mlf –w definicion\red.slf definicion\diccionario.txt definicion\listahmm.txt prueba\mfcc\entrada.mfcc

- **-H modelo\hmm1\hmm_x** es el hmm_x entrenado. Debe añadirse este argumento para cada hmm_x que tengamos en el directorio **modelo\hmm1**.
- **-w definicion\red.slf** es la red definida.
- **definicion\diccionario.txt** es el diccionario definido.
- **definicion\listahmm.txt** es la lista de los HMM entrenados.
- **prueba\mfcc\entrada.mfcc** es el archivo de entrada que queremos que sea reconocido.
- **-i prueba\reco.mlf** es el archivo de salida transcrita por el reconocedor.

Una transcripción de ejemplo se muestra en la **Figura 25**.

```
#!MLF!#
"prueba\mfcc\entrada.rec"
0 4900000 silencio -2394.001465
4900000 12000000 palabra2 -5159.434570
12000000 18300000 silencio -3289.197021
```

Figura 25. Ejemplo de una transcripción durante el Reconocimiento

En ejemplo de transcripción se observa que se han reconocido los eventos silencio-palabra2-silencio, sucedidos en los tiempos (en nanosegundos) que aparecen a la izquierda, y con las probabilidades logarítmicas que figuran a la derecha.

Reconocimiento en Tiempo Real

Una forma más interactiva de probar el reconocedor es usar las opciones de "entrada en vivo" de **HVite**. El comando es el siguiente:

HVITE -C analisis\directin.conf -H modelo\hmm1\hmm_x –w definicion\red.slf definicion\diccionario.txt definicion\listahmm.txt

El archivo **directin.conf** tiene el mismo contenido que **analisis.conf** salvo que hay que agregar dos parámetros más:

- **AUDIOSIG = -1** define la señal que usaremos para controlar el funcionamiento del reconocedor. Un valor negativo indica que es controlado por la pulsación de la tecla *intro*.
- **SOURCEKIND = HAUDIO** indica que la entrada de audio será la entrada de línea (micrófono).

Al ejecutar el módulo **HVite** aparecerá el indicador ***READY*[1]>** en la pantalla, lo cual significa que comienza el reconocimiento en tiempo real. El proceso finalizará cuando pulsemos alguna tecla. En ese momento, se muestra la transcripción decidida por el reconocedor, tras lo que se mostrará un nuevo indicador (***READY*[2]>)** esperando una nueva entrada. En la **Figura 26** se muestra dicho proceso.

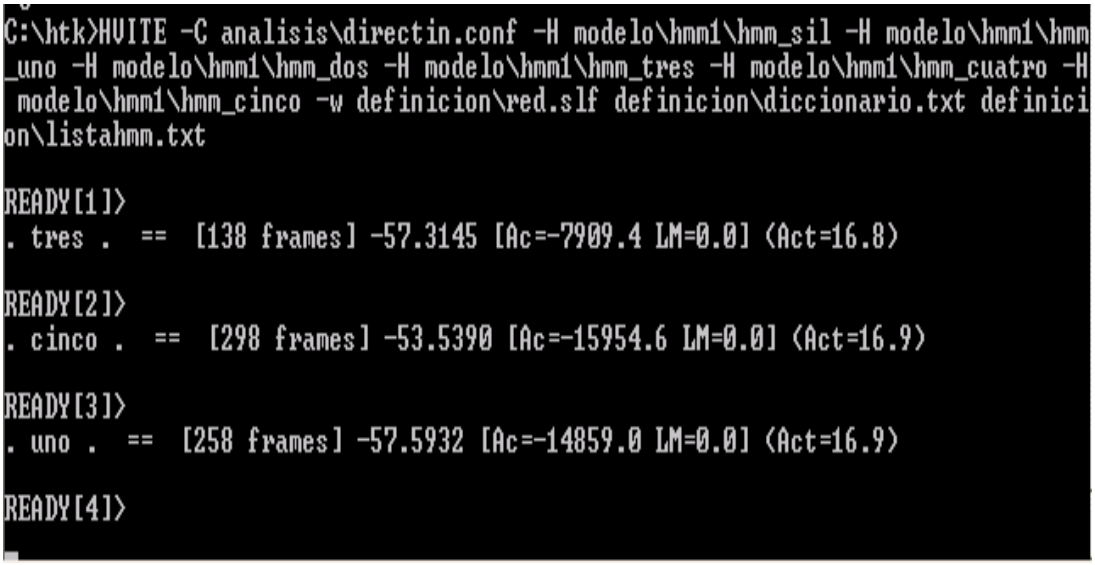

Figura 26. Salida en la etapa de reconocimiento en tiempo real.

Nota: En el reconocimiento en tiempo real puede evitarse el tener que oprimir la tecla *intro* cada vez que se quiera obtener qué se ha reconocido. Para ello hay que comentar en el archivo **directin.conf** la línea **AUDIOSIG = -1** y colocar la línea **USESILDET=T** la cual hará que el reconocedor devuelva una salida cada vez que detecte una señal de voz.

5.3. Banco de Reactivos

El Banco de Reactivos contiene un conjunto de preguntas con sus respectivas respuestas relacionadas con los números, colores, letras, figuras etc. Sirve de apoyo al Generador de Preguntas y Respuestas ya que éste selecciona y envía a la Interfaz Gráfico-acústica la pregunta a mostrar al usuario, también el Banco de Reactivos apoya al Intérprete de Comandos en el proceso de validar si la respuesta dada por el usuario es correcta en referencia a la pregunta que se le está haciendo.

Cada reactivo está conformado por un archivo de audio en formato *wav* que es la pregunta que se realiza al usuario, un archivo de texto que contiene la respuesta y un archivo de imagen asociada a la pregunta.

Las preguntas fueron realizadas con ayuda del sintetizador *Ivona Reader* **[12]** en su versión de prueba y se hizo uso de una voz femenina.

Los reactivos fueron divididos en grupos (números, colores, figuras, letras, objetos) como se muestra en la **Figura 27** y almacenados en carpetas para una mejor organización y facilitación en el manejo de dichos reactivos.

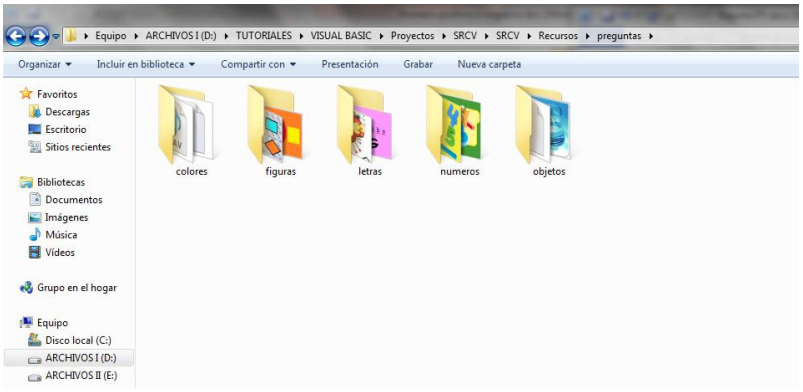

Figura 27. Organización del Banco de Reactivos

La nomenclatura utilizada en los archivos (ver **Figura28**) que conforman al reactivo son p**csc** para los archivos de audio, r**csc** para los archivos de respuesta e i**csc** para las imágenes donde:

- **csc** es un consecutivo correspondiente al número de pregunta de cada grupo de reactivos.
- **p** indica que se trata de una pregunta.
- **r** indica que se trata de una respuesta.
- **i** indica que se trata de una imagen asociada al archivo de pregunta.

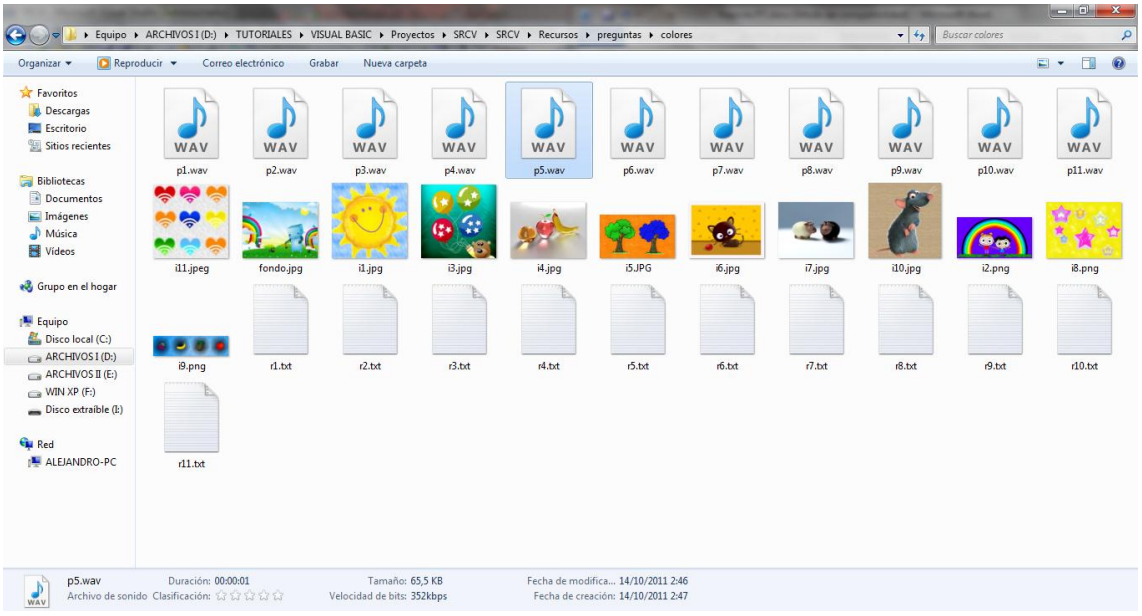

Figura 28. Nomenclatura de los archivos dentro de un grupo de reactivos

5.4.Interfaz Gráfico-acústica

Tanto el diseño de este bloque como el de los siguientes que se presentarán en el capítulo (Generador de Preguntas y Respuestas e Intérprete de Comandos) fueron diseñados sobre el sistema operativo Windows 7 de 32 bits, bajo el ambiente de desarrollo Microsoft Visual Studio 2010. El lenguaje de programación utilizado para la codificación de los bloques es Visual Basic .NET.

El diseño de este bloque consta de cuatro formularios los cuales son:

- **Formulario de Inicio:** Muestra la ventana de bienvenida al usuario.
- **Formulario de Menú:** Muestra las opciones de juego que contiene el sistema (números, colores, figuras, letras, objetos).
- **Formulario de Preguntas:** Muestra una serie de preguntas relacionadas con la opción elegida en el Menú.
- **Formulario de Resultados:** Muestra los aciertos y fallos obtenidos.

A continuación se muestra la forma de crear un proyecto en Visual Basic .NET, y agregar los formularios necesarios para crear una interfaz de usuario, así como una breve explicación acerca del entorno de desarrollo.

Iniciamos el entorno de desarrollo como se muestra en la **Figura 29**.

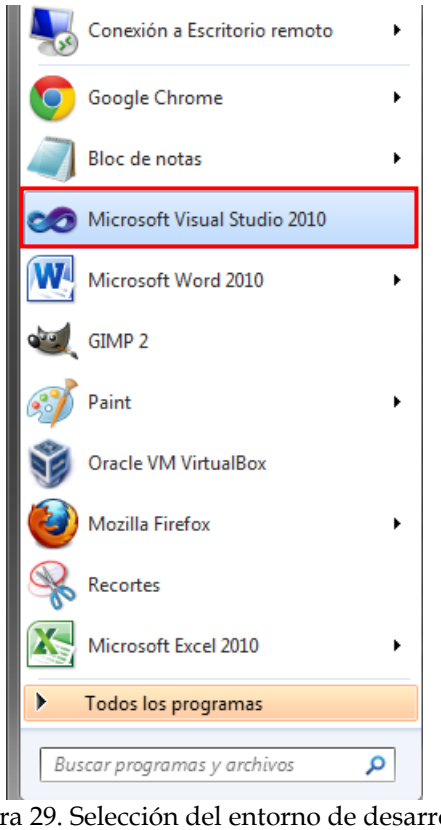

Figura 29. Selección del entorno de desarrollo

Una vez iniciado damos *click* en el menú **Archivo** y después en **Nuevo Proyecto** como se muestra en la **Figura 30**.

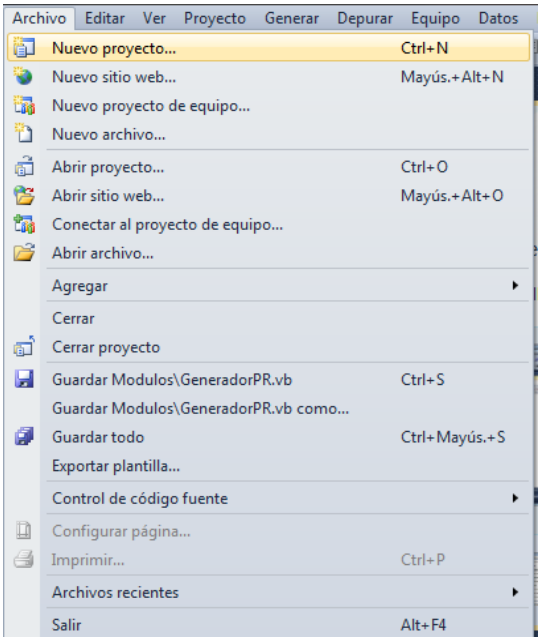

Figura 30. Creación de un proyecto en Visual Studio

En la ventana **Nuevo proyecto** (ver **Figura 31**) damos *click* en **Plantillas instaladas** ahora seleccionamos **Visual Basic**, después seleccionamos **Aplicación de *Windows Form*** y en la parte donde dice **"Nombre"** pondremos el nombre del proyecto. Damos *click* en Aceptar.

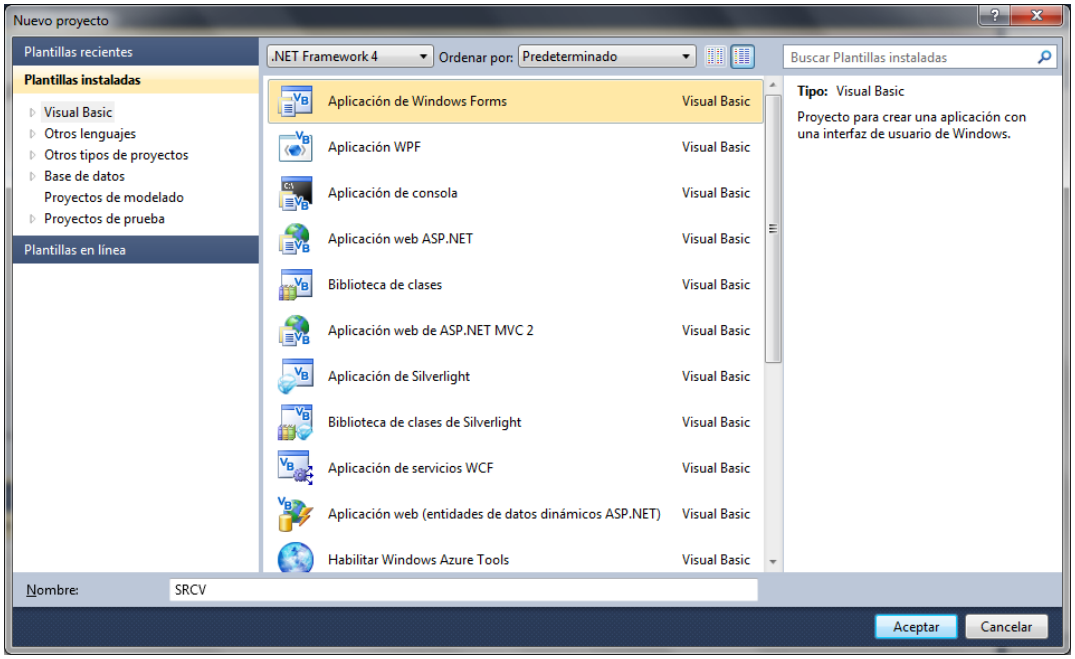

Figura 31. Selección de la aplicación a utilizar

Con esto se ha creado un proyecto y se muestra el espacio de trabajo (ver **Figura 32**). Distinguimos que en automático se habrá cargado un formulario.

En el lado superior derecho de la pantalla se encuentra el panel llamado **Explorador de Soluciones** en el que se indica el proyecto actual y muestra los ficheros que lo componen.

En inferior derecho se encuentra el panel llamado **Propiedades** el cual se utiliza para conocer o modificar las características del formulario, o los elementos contenidos en el mismo (**controles**).

Los controles son los elementos que insertamos dentro de un formulario y que van a permitir la interacción entre el usuario y el sistema, algunos de estos controles son botones, cuadros de texto, etiquetas, cuadros desplegables, es decir todos aquellos elementos que vemos en los formularios de una aplicación. La lista de controles disponibles se encuentra a la izquierda, en el panel **Cuadro de herramientas**.

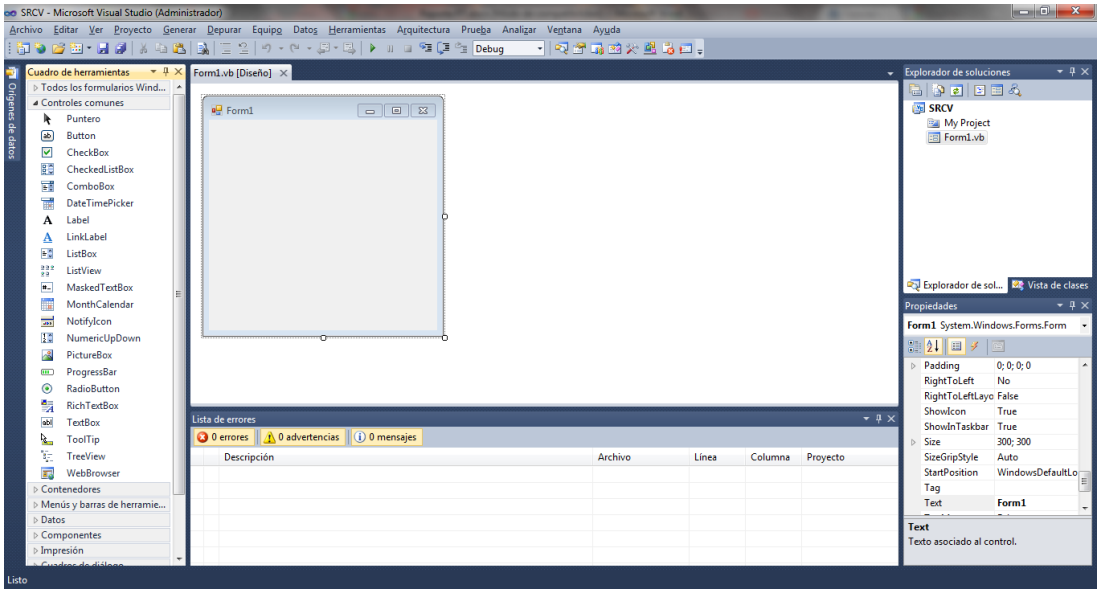

Figura 32. Espacio de trabajo en Visual Studio

Si se requiere añadir un formulario al proyecto damos *click* en el **menú Proyecto** y después seleccionamos **Agregar *Windows Forms*** como se muestra en la **Figura 33**.

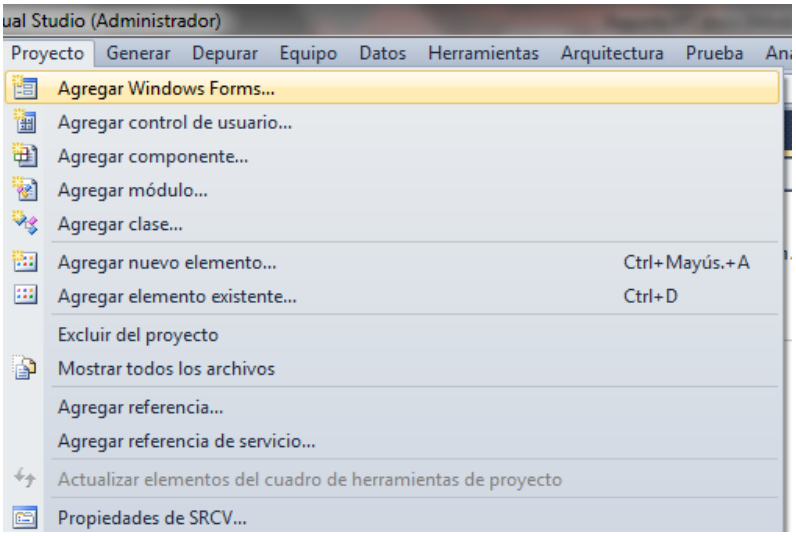

Figura 33. Agregación de un formulario al proyecto

En la ventana **Agregar nuevo elemento** (ver **Figura 34**) seleccionamos *Windows Forms*, en **Nombre** escribimos el nombre del formulario y damos *click* en el **botón Agregar**.

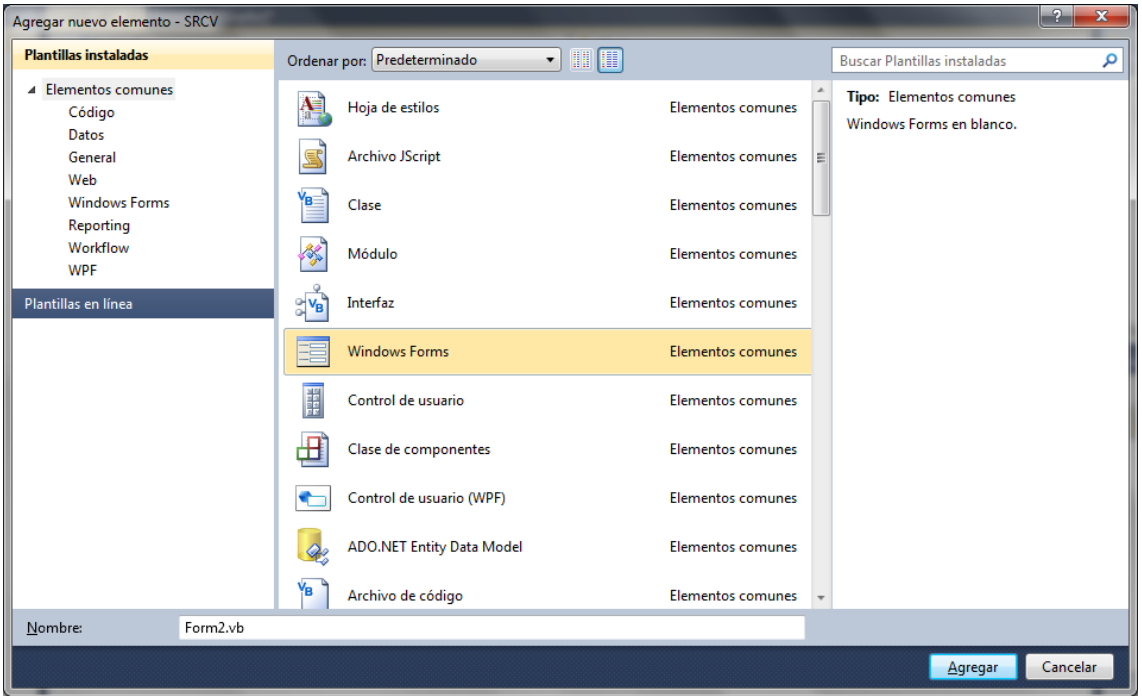

Figura 34. Selección del nuevo formulario

En el Panel **Explorador de Soluciones** (ver **Figura 35**) se observa que el formulario ha sido agregado al proyecto.

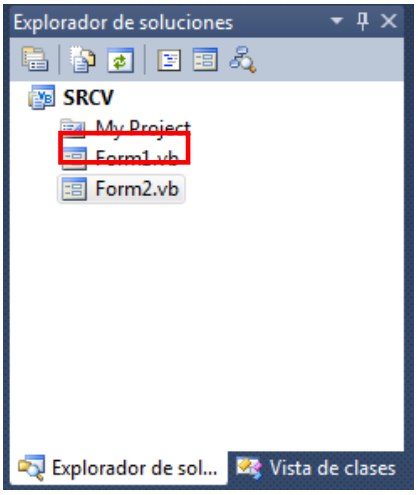

Figura 35. Visualización del nuevo formulario en el Panel Explorador de Soluciones

Para añadir controles al formulario como se muestra en la **Figura 36** utilizaremos el panel **Cuadro de herramientas**. Por ejemplo, para añadir una etiqueta (*Label*), una caja de texto (*TextBox*) y un botón (*Button*), simplemente haremos doble-*click* sobre esos elementos de la barra de herramientas y se añadirán al formulario o podemos arrastrarlos hasta nuestro formulario.

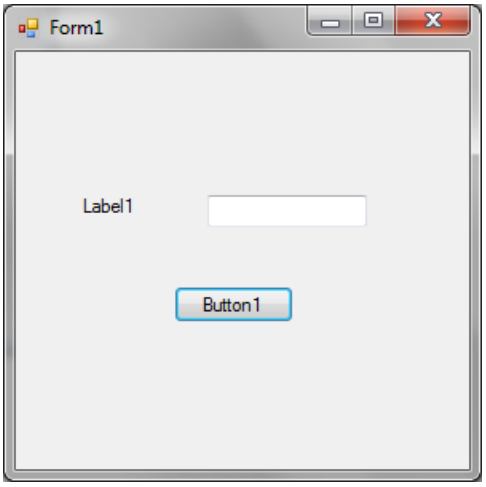

Figura 36. Ejemplo de controles dentro de un formulario

Como vemos, por defecto el entorno de desarrollo pone unos nombres genéricos a los controles, como *label1*, *button1*, es decir utiliza el tipo de control y lo va numerando tantas veces como controles iguales tengamos en el formulario (*label1*, *label2*, *label3*,...). Esto es solo para poner un nombre inicial, ya que siempre los controles deben tener un nombre que los identifique para poder trabajar con ellos durante la programación.

Ahora vamos a cambiar el texto que contiene el botón *Button1*. Para cambiar el texto hay que utilizar el panel de Propiedades, en esta ocasión el elemento que nos interesa de esa ventana de propiedades es *Text*, escribimos en esta propiedad la palabra **Mostrar,** como se muestra en la **Figura 37,** y cuando se pulse *Intro* o tabulador veremos que el texto del botón se ha actualizado.

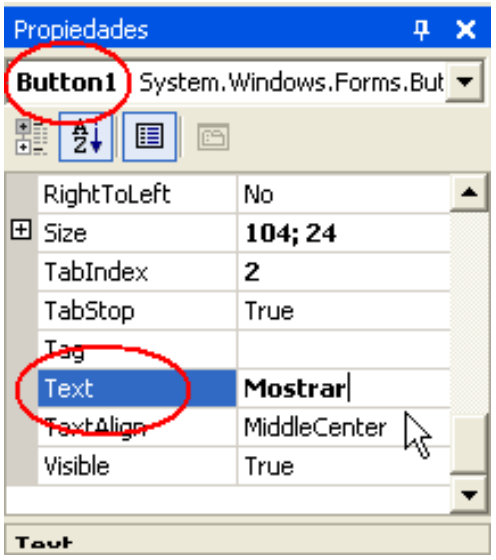

Figura 37. Ejemplo de propiedades del control Button

Hacemos lo mismo con la etiqueta *Label*, primero hay que seleccionarla para hacer que se muestren las propiedades de la etiqueta (ver **Figura 38**). Escribimos la palabra **Nombre** en la propiedad *Text* y pulsamos *intro* o tabulador.

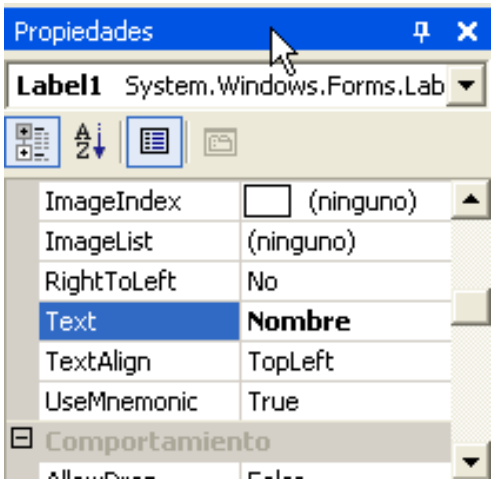

Figura 38. Ejemplo de propiedades del control Label

Ahora vamos a ver la manera de agregar código. Por ejemplo si queremos agregar código al control botón hacemos doble *click* en él, ahora se mostrará una nueva ventana, en este caso la ventana de código del botón donde se muestra el ***procedimiento*** llamado *Button1_**Click*** que contendrá una serie de instrucciones para llevar a cabo alguna acción (ver **Figura 39**). También se observa que este procedimiento pertenece a la ***Clase*** *Form1* que en automático toma el nombre del formulario.

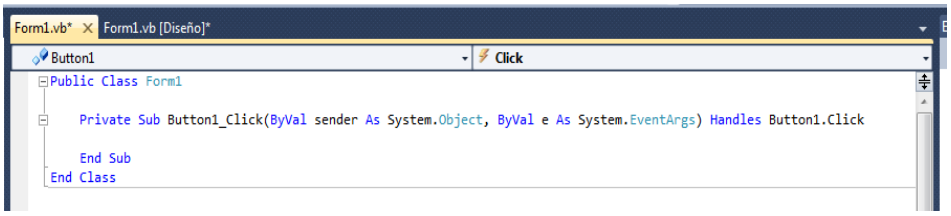

Figura 39. Procedimiento del evento *click* para el control *Button*

Aquí empieza la programación. Que nosotros pulsemos doble click en el botón y aparezca un fragmento de código significa lo siguiente: VB .NET interpreta que se quiere poner código para realizar alguna acción cuando se haga *click* sobre el botón. Por lo tanto muestra el **procedimiento** o parte de código que se va a ejecutar cada vez que suceda esto. A esta forma de trabajar se le llama "***programación orientada a eventos***". Es decir, cuando se produzca el ***evento*** de pulsar el botón (*click*) se ejecutará este código. Por lo tanto, vemos que los controles además de tener propiedades (que modifican su aspecto) también atienden a una serie de eventos (*click*, doble *click*, movimiento del ratón, presión de una tecla etc.). Los eventos a los que atienden los controles los podemos ver en la ventana de propiedades, seleccionando el icono de un rayo como se observa en la **Figura 40**.

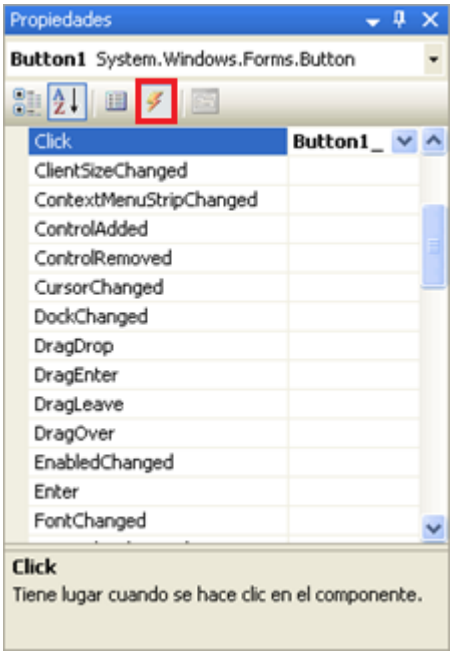

Figura 40. Lista de eventos disponibles para el control *Button*

Después de seleccionar un evento Visual Studio nos llevará a la ventana de código, en el desplegable de arriba a la izquierda se muestran los objetos que hemos añadido en nuestro formulario. Seleccionamos *Button1* y ahora en el desplegable de la derecha seleccionamos el evento *click* como se muestra en la **Figura 41**. Mencionar que existen numerosos eventos para cada control los cuales responden a diferentes señales.

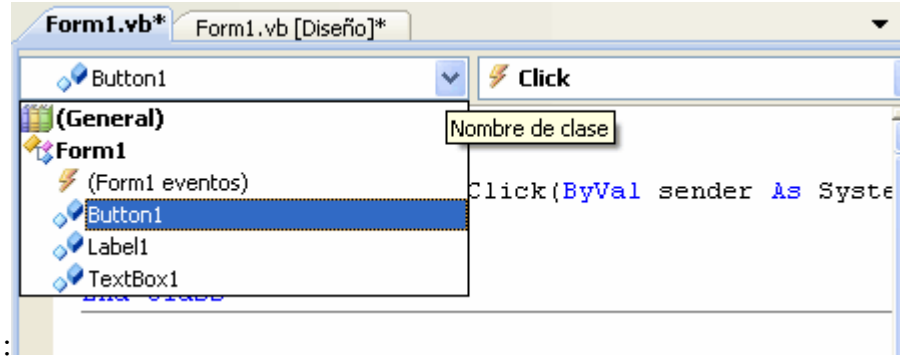

Figura 41. Vista de los controles añadidos al Formulario de trabajo

Volvemos a la ventana de código. Ésta ventana en la parte superior indica que estamos trabajando con *Form1.vb* (nombre del archivo que contiene el código fuente), debajo nos indica que hemos seleccionamos el control *Button1* y a la derecha que estamos trabajando con su evento *click* (ver **Figura 42**).

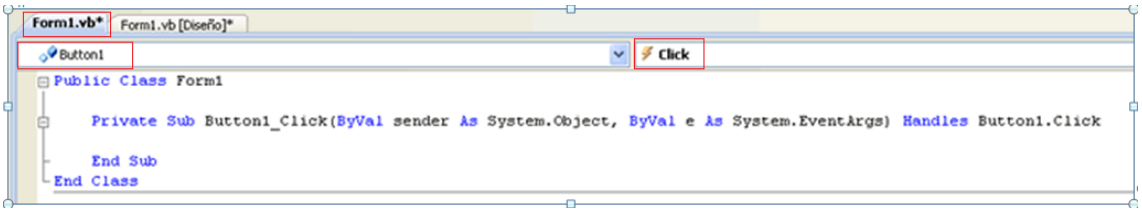

Figura 42. Ventana de código del formulario llamado Form1

Por lo tanto mediante una serie eventos colocados en cada formulario y conocimientos básicos sobre VB .NET se puede lograr obtener una Interfaz Gráfica en la que un usuario pueda interactuar con el sistema.

A continuación se muestra el algoritmo y diagrama de flujo de cada formulario:

Formulario de Inicio

Algoritmo:

INICIO
¿Comenzar con el juego?
Si (respuesta=”si”)
Mostrar Formulario de Menú
CC Si (respuesta=”no”)
Salir de juego
CC regresa a INICIO
FIN SI
FIN

Se muestra la secuencia de trabajo del formulario mediante un diagrama de flujo (ver **Figura 43**).

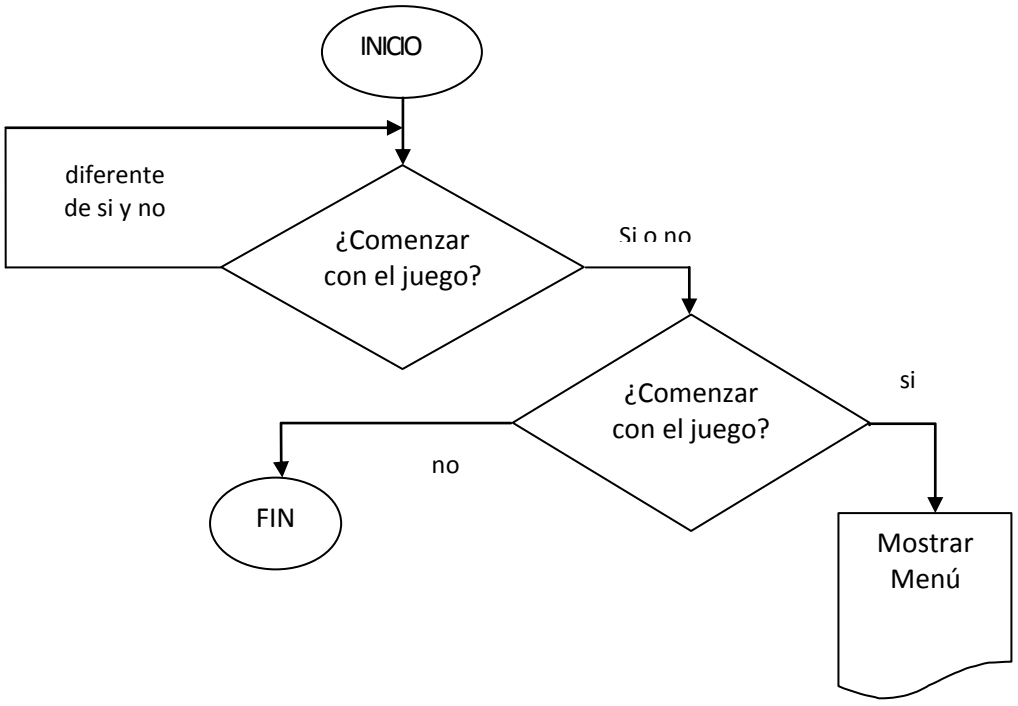

Figura 43. Diagrama de flujo – Formulario de Inicio

Formulario de Menú

Algoritmo:

INICIO
¿Jugar con los números?
Si (respuesta=”si”)
Mostrar preguntas asociadas
CC Si (respuesta=”no”)
¿Jugar con los colores?
Si (respuesta=”si”)
Mostrar preguntas asociadas
CC Si (respuesta=”no”)
¿Jugar con las figuras?
Si (respuesta=”si”)
Mostrar preguntas asociadas
CC Si (respuesta=”no”)
¿Jugar con las letras?
Si (respuesta=”si”)
Mostrar preguntas asociadas
CC Si (respuesta=”no”)
¿Jugar con los objetos?
Si (respuesta=”si”)
Mostrar preguntas asociadas
CC Si (respuesta=”no”)
¿Quieres salir del juego?
Si (respuesta=”si”)
Salir del juego
CC Si (respuesta=”no”)
Regresa a INICIO
FIN SI
FIN

Se muestra la secuencia de trabajo del formulario mediante un diagrama de flujo (ver **Figura 44**).

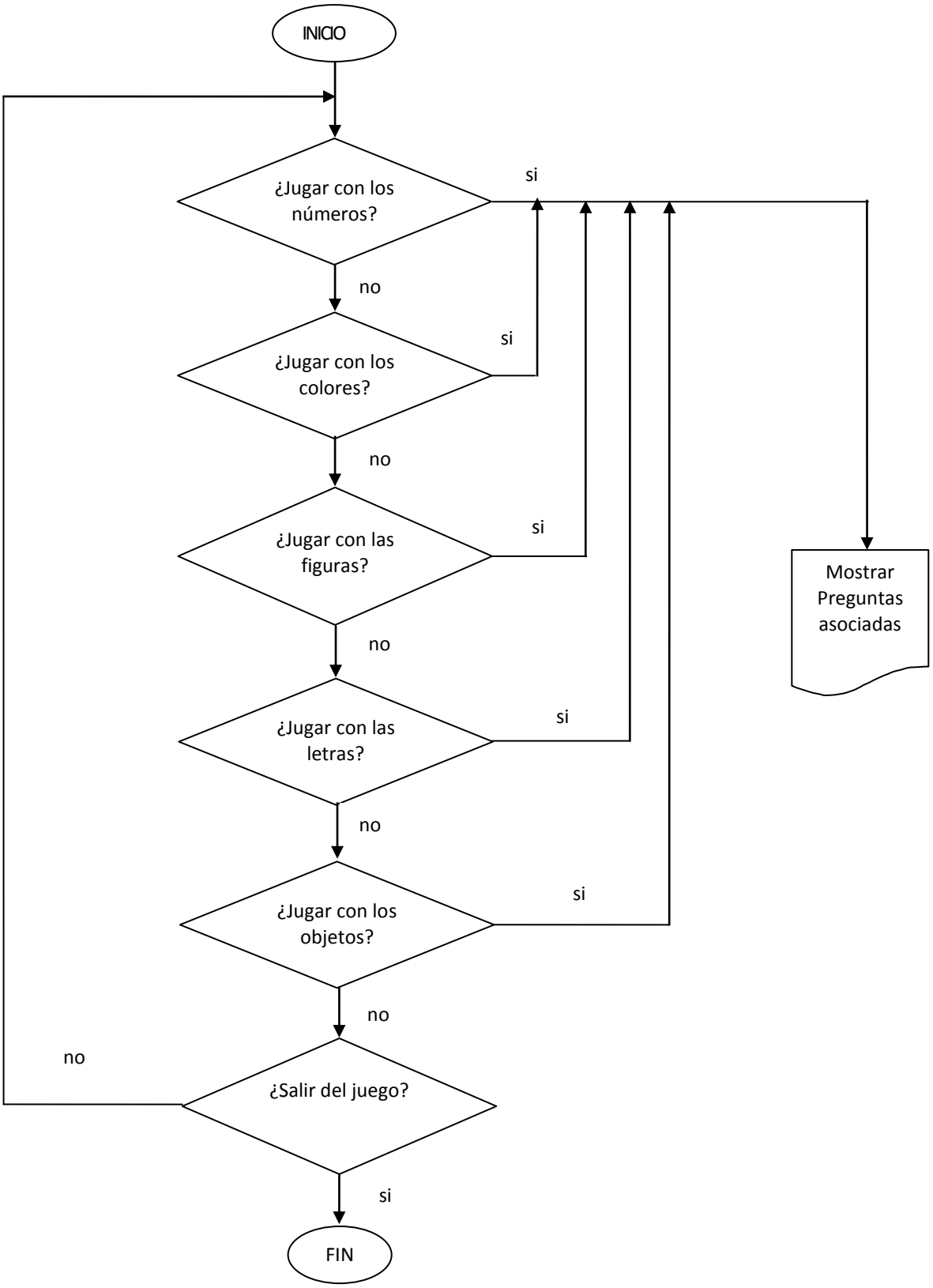

Figura 44. Diagrama de flujo – Formulario de Menú

Formulario de Preguntas

Algoritmo:

INICIO

Se realiza una pregunta al usuario
SI (respuesta = “salir”)
Salir del juego
CC Analiza respuesta
SI (respuesta es correcta)
Incrementa número de aciertos
SI (existen más preguntas)
Muestra siguiente pregunta
CC Muestra los Resultados
CC incrementa número de fallos
Repite pregunta
FIN SI

FIN

Se muestra la secuencia de trabajo del formulario mediante un diagrama de flujo (ver **Figura 45**).

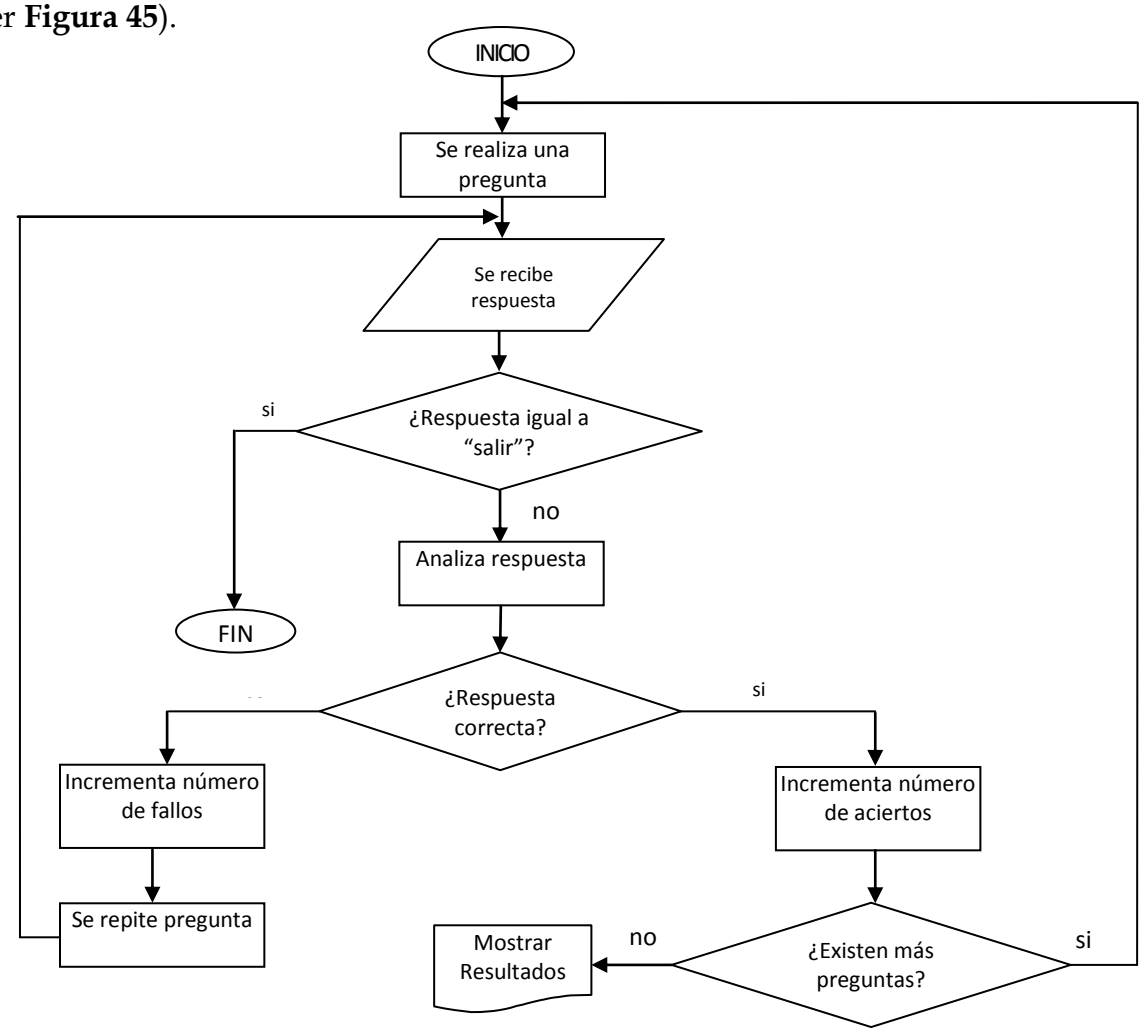

Figura 45. Diagrama de flujo – Formulario de Preguntas

Formulario de Resultados

Algoritmo:

INICIO
Mostrar Resultados
¿Regresar al Menú?
Si (respuesta =”si”)
Muestra Menú
CC Si (respuesta = “no”)
¿Salir del juego?
Si (respuesta =”si”)
Salir del juego
CC Si (respuesta =”no”)
Regresa a INICIO
FIN SI
FIN

Se muestra la secuencia de trabajo del formulario mediante un diagrama de flujo (ver **Figura 46**).

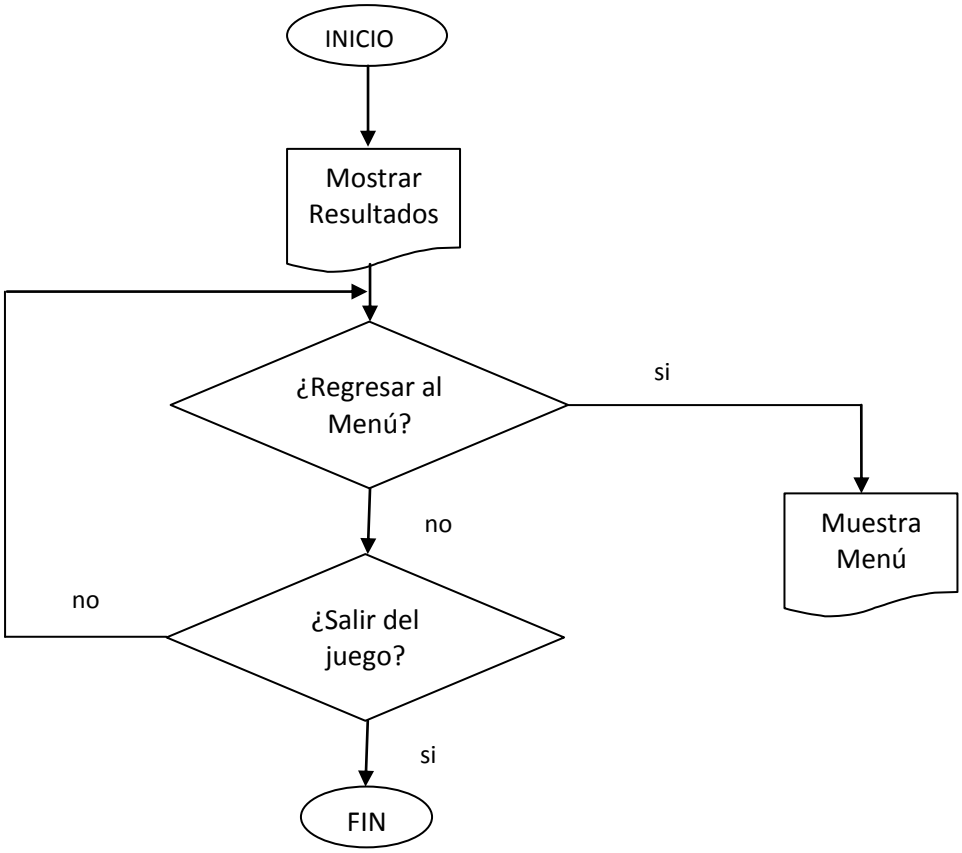

Figura 46. Diagrama de flujo – Formulario de Resultados

El código fuente de este bloque puede verse en el Anexo 2.1

5.5.Generador de Preguntas y Respuestas

Se diseñó una clase llamada **"GeneradorPR"** la cual contiene una metodo (similar a un procedimiento pero éste recibe uno o un juego de parámetros de entrada que regularán la acción a realizar y, posiblemente devuelva un valor de retorno de algún tipo) del mismo nombre, el cual se encuentra sobrecargado, es decir, en la clase existen dos métodos con el mismo nombre, y dependiendo de los parámetros que reciba, el sistema sabrá que metodo utilizar para realizar la funcionalidad requerida. El metodo GeneradorPR puede recibir un parámetro de tipo *Boolean* (este es el generador de respuestas) ó 2 parámetros, uno de tipo *String* y otro de tipo *Integer* (este es el generador de preguntas).

El metodo que solo recibe un parámetro se encarga de dar a conocer al usuario si la respuesta que dio es correcta o no, reproduciendo un archivo de audio.

El segundo metodo se encarga de crear la serie de preguntas a ser mostradas en la interfaz al usuario al usuario.

A continuación se muestra el algoritmo y diagrama de flujo de la secuencia de trabajo de este bloque.

Algoritmo:

INICIO
Se crea pregunta con Generador de preguntas
Se envía pregunta a la interfaz
Se analiza respuesta
¿Respuesta correcta?
SI (respuesta es correcta)
Ejecutar Generador de respuestas
Presenta aviso de acierto
SI (¿Existen más preguntas?)
Se crea pregunta con Generador de preguntas
Regresar a primer paso
CC terminar
CC Ejecutar Generador de respuestas
Se presenta aviso de error
Se pregunta de nuevo
Regresar a Se analiza respuesta
FIN SI
FIN

Se muestra la secuencia de trabajo del formulario mediante un diagrama de flujo (ver **Figura 47**).

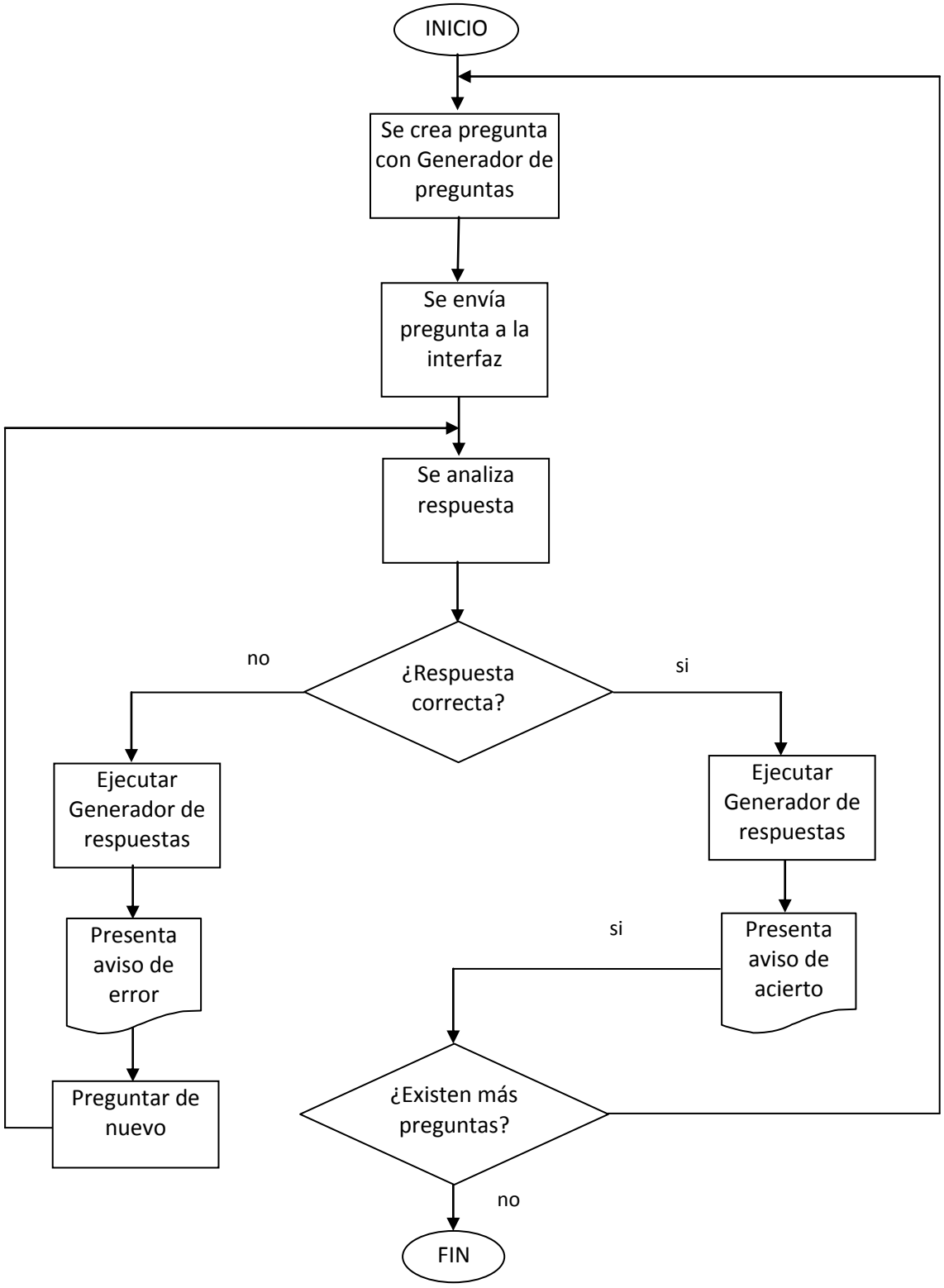

Figura 47. Diagrama de flujo – Generador de Preguntas y Respuestas.

El código fuente de este bloque puede verse en el Anexo 2.2

5.6.Intérprete de Comandos

Su diseño consiste en una Clase llamada "**Interprete**" y cuenta con un método del mismo nombre, dicho método recibe 3 parámetros los cuales servirán en la comparación que se hace entre la salida del bloque Reconocedor de Comandos Voz (resultado de una respuesta del usuario) y la respuesta correcta (almacenada en el Banco de Reactivos) para la pregunta en cuestión.

Los parámetros enviados al Interprete son: el **"comando reconocido"**, el **"tipo de pregunta"** que se está haciendo (números, colores, etc.) y el **"número de pregunta"**. Mediante estos parámetros, primeramente el Intérprete busca en el Banco de Reactivos la respuesta correcta utilizando el tipo y número de pregunta, obtenida la respuesta el Intérprete la compara con el parámetro comando reconocido para devolver un valor que indica si la respuesta del usuario fue correcta o no.

A continuación se muestra el algoritmo y diagrama de flujo de la secuencia de trabajo de este bloque

Algoritmo:

INICIO
Obtener respuesta correcta
Comparar respuesta con comando
SI (¿Son iguales?)
Valor de retorno = True
CC Valor de retorno= False
FIN SI
FIN

Se muestra la secuencia de trabajo del formulario mediante un diagrama de flujo (ver **Figura 48**).

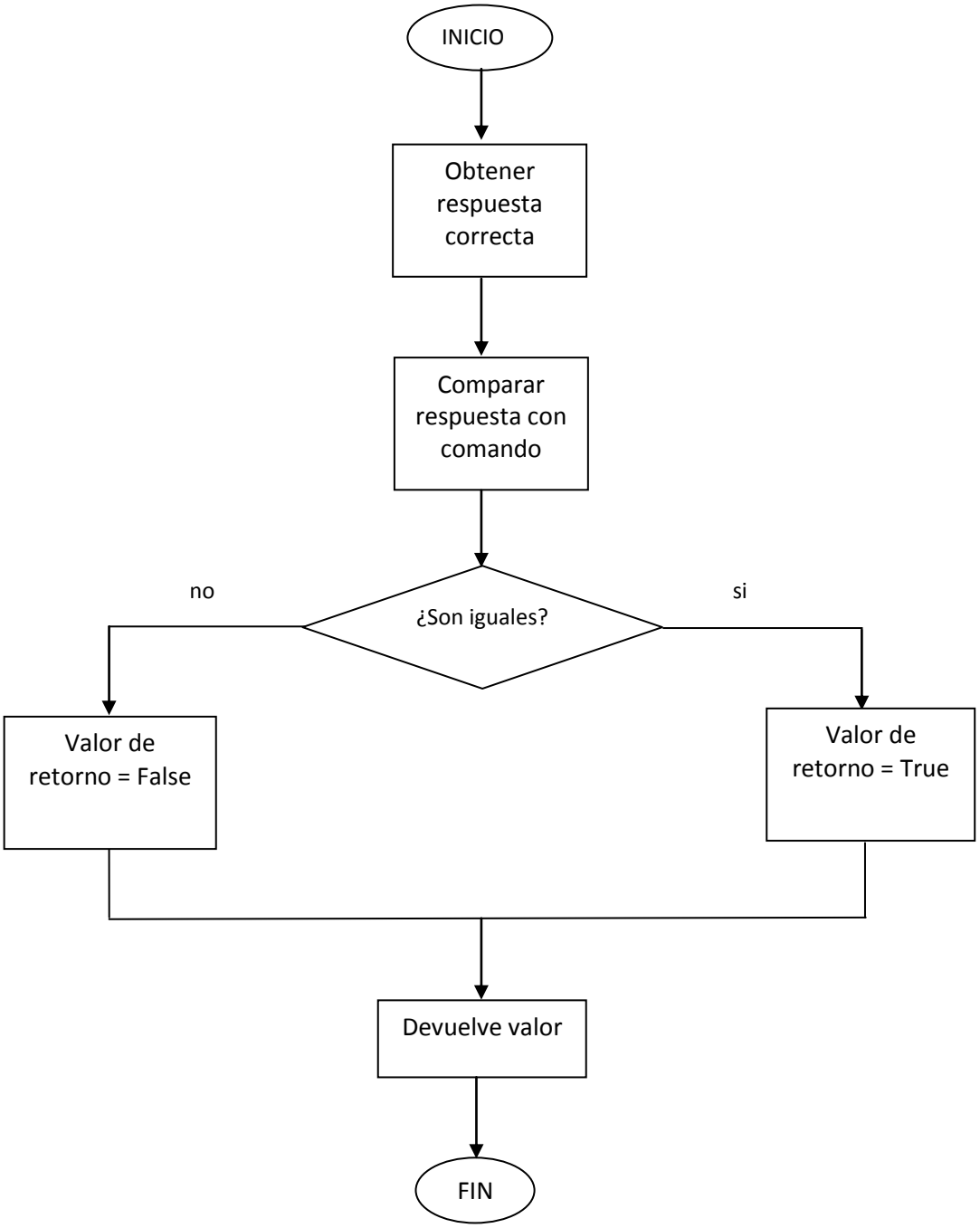

Figura 48. Diagrama de flujo – Interprete de Comandos

El código fuente de este bloque puede verse en el Anexo 2.3

CAPÍTULO 6

PRUEBAS Y RESULTADOS

La metodología empleada para evaluar el desempeño del sistema de reconocimiento de voz se basa en la medición del porcentaje de palabras reconocidas correctamente. Este porcentaje se determinará como sigue:

$$\text{Porcentaje de precisión} = \frac{(\text{num de veces reconocida} * 100)}{\text{num de veces pronunciada}}$$

Dónde:

- **Porcentaje de precisión** es el porcentaje de reconocimiento por palabra.
- **num de veces reconocida** es la cantidad de veces que una palabra fue reconocida correctamente
- **num de veces pronunciada** es la cantidad de veces que una palabra fue pronunciada para la realización de las pruebas.

Las pruebas del reconocedor desarrollado se realizaron con 2 usuarios con las siguientes características:

- **Usuario 1**: Hombre, 23 años, sus patrones de voz se encuentran almacenados en la base de datos.
- **Usuario 2**: Mujer, 21 años, sus patrones de voz no se encuentran almacenados en la base de datos.

Las pruebas consistieron en la pronunciación, mediante micrófono, de 20 palabras pertenecientes al vocabulario del reconocedor y repetidas 10 veces cada una de ellas. No se utilizó ningún otro dispositivo especial, se buscó que el entorno de pruebas estuviera libre de ruido lo más posible. No se restringieron modos de lectura a los hablantes. La intención era realizar una prueba en condiciones normales.

El equipo donde se realizaron las pruebas tiene las siguientes características: Windows 7 de 32 bits, procesador Athlon II doble núcleo a 2.9 GHz, 2 GB de memoria RAM, tarjeta de sonido VIA HD Audio Deck.

En la **Tabla 1** y **Tabla 2** se muestran los resultados obtenidos para cada uno de los usuarios.

Usuario 1:

Palabra pronunciada	Número de veces reconocida	Porcentaje de precisión
si	9	90 %
no	9	90 %
uno	8	80 %
dos	9	90 %
tres	8	80 %
seis	10	100 %
diez	8	80 %
azul	10	100 %
rosa	10	100 %
rojo	8	80 %
gris	10	100 %
negro	10	100 %
rectangulo	9	90 %
triangulo	9	90 %
reloj	10	100 %
lápiz	10	100 %
foco	10	100 %
be	8	80 %
ce	8	80 %
ge	9	90 %
Promedio	9,1	91 %

Tabla 1. Resultados de reconocimiento del usuario 1

Usuario 2:

Palabra pronunciada	Número de veces reconocida	Porcentaje de precisión
si	8	80%
no	7	70%
uno	6	60%
dos	5	50%
tres	6	60%
seis	7	70%
diez	6	60%
azul	7	70%
rosa	6	60%
rojo	6	60%
gris	6	60%
negro	7	70%
rectangulo	6	60%
triangulo	6	60%
reloj	6	60%
lápiz	7	70%
foco	7	70%
be	6	60%
ce	5	50%
ge	6	60%
Promedio	**6,3**	**63%**

Tabla 2. Resultados de reconocimiento del usuario 2

En la **Figura 49** se muestra la gráfica **Comparación de Resultados** donde se observa claramente que el usuario 1 tiene un mayor porcentaje de reconocimiento en cada una de las palabras que se repitieron para la realización de las pruebas.

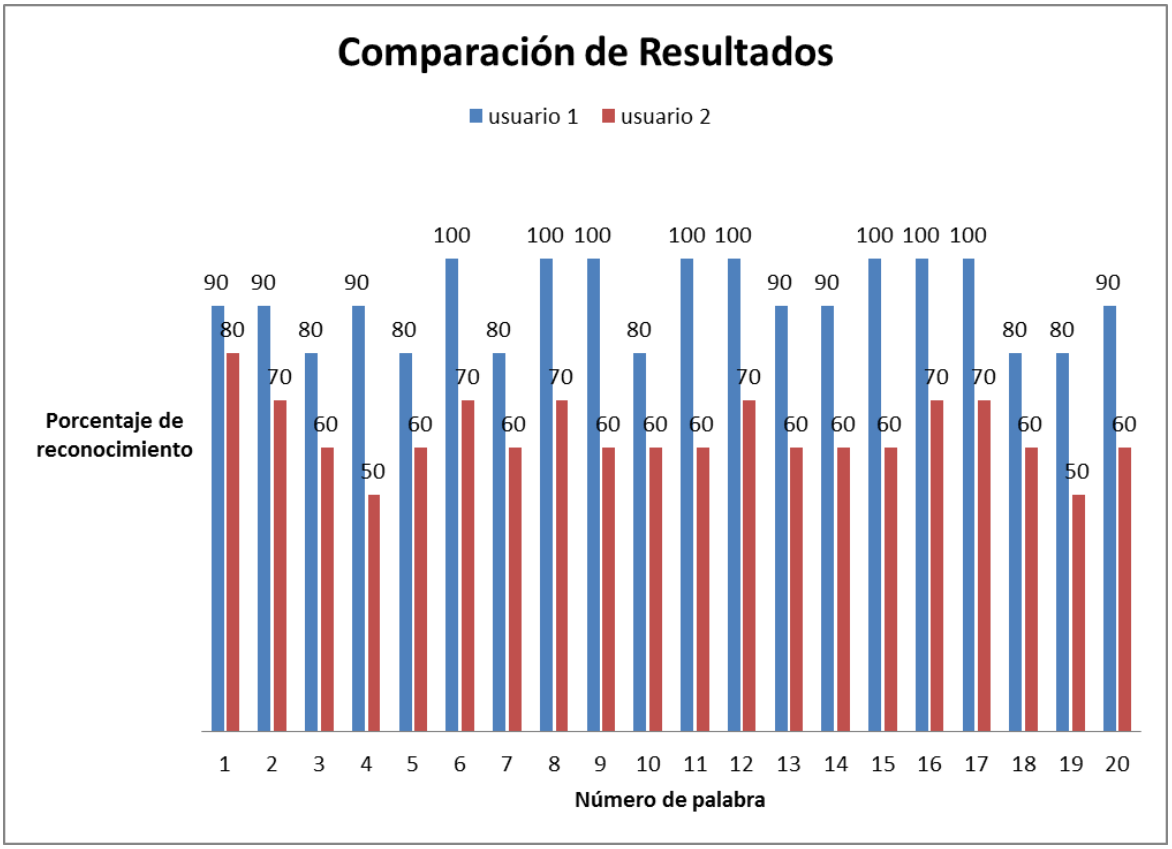

Figura 49. Comparación entre los resultados obtenidos por los usuarios en las pruebas

Promediando el porcentaje de precisión de cada palabra se obtiene un reconocimiento del 91% para el usuario 1 y 63% para el usuario 2.

El porcentaje promedio de reconocimiento del usuario 2 es menor debido a que los modelos no fueron entrenados con la voz de este usuario y por lo tanto no existen patrones de referencia con su voz en la base de datos.

Por último mencionar que los resultados pueden variar con el ruido en el ambiente, el micrófono y la tarjeta de audio de la PC.

CAPÍTULO 7
CONCLUSIONES

El objetivo de este proyecto fue desarrollar un sistema reconocedor de voz y aplicarlo en una interfaz gráfica interactiva, con la finalidad de apoyar a niños en edad preescolar en el reforzamiento de sus conocimientos, así como estimular habilidades intelectuales. Se busca fomentar el desarrollo de este tipo de aplicaciones que puedan ser útiles en el ámbito educativo para miles de personas.

Mediante la realización de las pruebas hechas con el reconocedor de voz se obtuvieron resultados satisfactorios, ya que, se obtuvo un porcentaje de reconocimiento mayor al 90%, en el caso de un usuario que participó en el entrenamiento del sistema. Los resultados muestran que la eficiencia del reconocedor depende de los patrones de voz almacenados en la base de datos donde se apoya el reconocedor.

Podemos decir que los objetivos planteados se han cumplido, puesto que, se ha conseguido implementar un sistema de reconocimiento de habla que puede considerarse aceptable, aunque se encuentre limitado, por ahora, en el número de usuarios y el tipo de comandos. Por otra parte se propuso, de manera opcional, que el sistema contara con un módulo de entrenamiento donde un nuevo usuario entrenara al sistema con su voz, y de esta manera, obtener un mejor porcentaje de reconocimiento. Este objetivo se eliminó, finalmente, debido a que su consecución requiere de un mayor tiempo.

Como posibles trabajos futuros que den continuidad al presente proyecto esta la implementación e integración de un módulo de entrenamiento con el cual se obtendrán nuevos patrones de referencia, mejorando así los resultados de reconocimiento para un número mayor de usuarios. También, extender el vocabulario del sistema con el fin de ampliar la gama de reactivos que incrementen las posibilidades de interacción con el usuario y los beneficios del aprendizaje. Además, se podría hacer que el sistema funcionara para el idioma inglés, para ello se deberán obtener los patrones de referencia necesarios.

Por último, es importante resaltar que el uso de la interfaz de voz, desarrollada en este trabajo, aporta un nuevo medio para interactuar con el entorno y pretende sustituir a los elementos clásicos de interacción, como lo son el ratón y el teclado. En el futuro, las interfaces de voz pueden dar lugar a entornos multimedia donde la voz juegue un papel más relevante, ya que hoy se cuenta con grandes avances en el campo del reconocimiento de voz.

BIBLIOGRAFÍA

[1] S. Young y G. Evermann, "An overview of the HTK Toolkit," en The HTK Book, version 3.3, Cambridge University, USA: Microsoft Corporation, 2005, cap. 2, pp. 14-21.

[2] R. del Rosario, "Uso de Reconocimiento de Voz en un Juego Electrónico para la Rehabilitación de Niños con el Problema de Lenguaje Dislalia," Tesis de Licenciatura, Facultad de Matemáticas, Universidad Autónoma de Yucatán, Yucatán, México, 2008, Disponible: http://www.tizimin.uady.mx/tesis/tesis_final_reyna.pdf

[3] M. Fernández, M. Peralbo y M. Mayor, "Aplicaciones del Software de Reconocimiento de Voz en el Aprendizaje de la Lectoescritura," Artículo, Universidad de la Coruña y Universidad de Salamanca, España, 2005, Disponible: http://www.loleweb.com/documentos/comunicacion_salamanca.pdf

[4] P. Caloto, M. Moranchel y A. Ruiz, "Tecnologías de Reconocimiento por Voz y su Aplicabilidad en Videojuegos, "Tesis de Licenciatura, Dep. de Ingeniería del Software e Inteligencia Artificial, Universidad Complutense de Madrid, Madrid, España, 2010, Disponible: http://eprints.ucm.es/11295/1/SistemasInform%C3%A1ticos.pdf

[5] N. Aguas, "Verificación de Pronunciación Basada en Tecnología de Reconocimiento de Voz para un Ambiente de Aprendizaje," Tesis Profesional, Dep. de Ing. en Sistemas Computacionales, Universidad de las Américas Puebla, Puebla, México, 1999, Disponible:http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/aguas_g_n/portada.html

[6] A. Mantilla, "análisis, reconocimiento y síntesis de voz Esofágica," Tesis Doctoral, Sección de estudios de Posgrado e Investigación, IPN Escuela Superior de Ingeniería Mecánica y Eléctrica, D.F., México, 2007.

[7] M. García, "Reconocedor de Voz Adaptado," Tesis de Maestría, Universidad Autónoma Metropolitana unidad Azcapotzalco, D.F., México, 2005.

[8] J. Bernal y J. Bobadilla, "Fundamentos informáticos relacionados con el tratamiento de voz," en Reconocimiento de voz y fonética acústica, 1era ed. D.F., México: Alfaomega, 2000.

[9] I. Villamil, "Aplicaciones en Reconocimiento de Voz utilizando HTK," Tesis de Licenciatura, Dep. de Electrónica, Pontifica Universidad Javeriana, Santa Fe de Bogotá, Colombia, 2005,
Disponible: http://www.javeriana.edu.co/biblos/tesis/ingenieria/tesis95.pdf

[10] A. Abejón, "Reconocimiento de Idioma en Voz Espontánea mediante Reconocimiento Fonético Multilingüe en Paralelo y Modelado Estadístico del Idioma," Tesis de Licenciatura, Dep. de Ingeniería Informática, Universidad

Autónoma de Madrid, Madrid, España, 2007, Disponible: http://arantxa.ii.uam.es/~jms/pfcsteleco/lecturas/20070926AlejandroAbejon.pdf

[11] N. Moreau, HTK (v.3.1): Basic Tutorial, Technische Universität Berlin, 2002, Disponible: http://agp1.hx0.ru/arts/HTK_basic_tutorial.pdf

[12] Sintetizador Ivona Reader, Disponible: http://www.ivona.com/en/reader/

ANEXOS

Anexo 1. Vocabulario de Comandos de Voz del Sistema SRCV

A continuación se muestra una tabla de las palabras reconocidas por el Sistema.

CATEGORÍA					
Números	**Colores**	**Figuras**	**Letras**	**Objetos**	**Otros**
uno	amarillo	circulo	a, b, c,	celular	si
dos	azul	cuadrado	d, e, f,	foco	no
tres	blanco	rectángulo	g, h, i	lápiz	quiero salir
cuatro	café	triangulo	j, k, l,	lentes	
cinco	gris		m, n, o,	libro	
seis	morado		p, q, r,	llave	
siete	naranja		s, t, u,	pelota	
ocho	negro		v, w, x	pluma	
nueve	rojo		y, z	reloj	
diez	rosa			vaso	
	verde				

Tabla 3. Comandos de Voz reconocidos por el Sistema SRCV

Nota: La lista mostrada en la **Tabla 3** corresponde con la lista de patrones de voz existentes en la base de datos.

Anexo 2. Interfaz del Sistema

Se Presentan las pantallas mostradas durante la ejecución del sistema, así como su secuencia. Además, se menciona el funcionamiento de cada una de las pantallas.

Ventana de bienvenida al Sistema

El sistema da la bienvenida al usuario y pregunta si quiere comenzar con el juego. Si la respuesta es "si" continua hacia el menú, si es "no" se cierra la aplicación, si la respuesta es una palabra diferente a las anteriores el sistema pide que se repita la respuesta.

Figura 50. Ventana de Inicio

Ventana de Menú

Se muestran las opciones de juego con números, colores, figuras, letras u objetos.

Inicialmente el *mouse* se sobrepone en la opción de Números y el sistema pregunta si se quiere jugar con esta opción, si se responde que "si" se pasa a la parte de realización de preguntas correspondientes a la opción elegida, si la respuesta es "no" el mouse pasa automáticamente a la siguiente opción de juego y el sistema pregunta si se quiere jugar con dicha opción. En el menú también está la posibilidad de salir del juego ya sea posicionando el mouse sobre la opción inferior (la imagen de una cara) en el menú, o pronunciando "quiero salir" que en los dos casos el sistema preguntará si en verdad se quiere salir del juego.

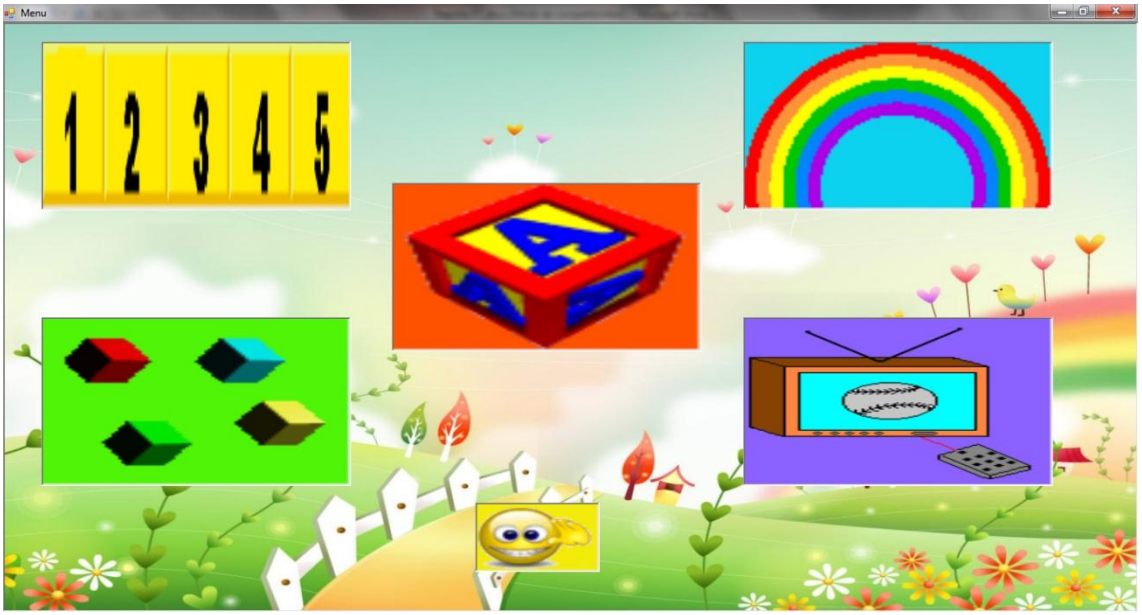

Figura 51. Ventana de Menú

Ventana de Preguntas
Se van presentando, una a una, las preguntas correspondientes a la opción elegida por el usuario.

Cada vez que responda el sistema indicará si fue una respuesta correcta o no, e irá registrando los aciertos y fallos que haya obtenido el usuario.

Si el usuario tiene 3 errores consecutivos en la misma pregunta el sistema pasa a la siguiente.

Durante la serie de preguntas esta siempre la opción de regresar al menú (posicionando el mouse en el recuadro inferior) o salir del juego pronunciando "quiero salir".

A continuación se muestran algunas pantallas de preguntas:

Figura 52. Ventana de preguntas - ejemplo 1

Figura 53. Ventana de preguntas - ejemplo 2

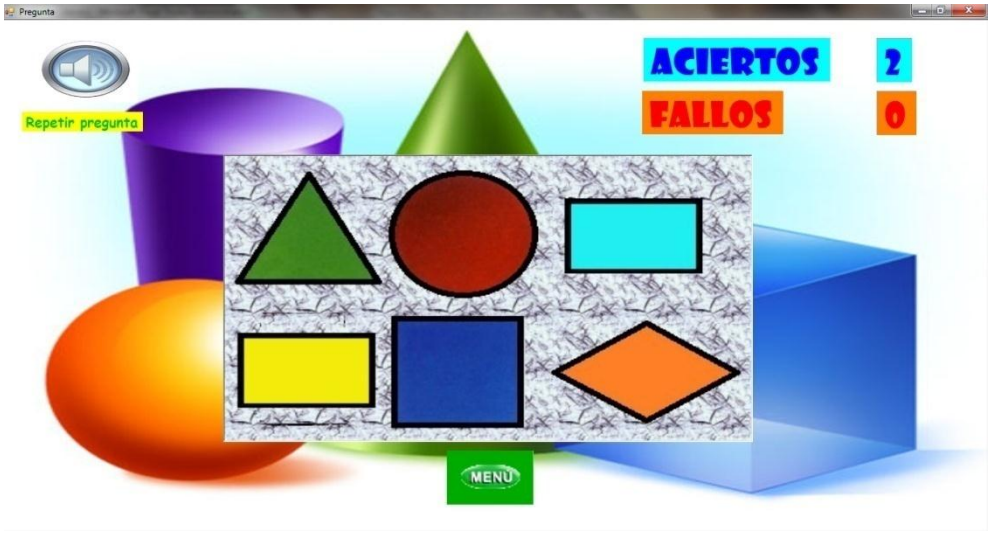

Figura 54. Ventana de preguntas - ejemplo 3

Ventana de Resultados

Finalmente después de realizadas todas las preguntas de la opción elegida se muestra la pantalla de resultados, donde el usuario puede ver los aciertos y fallos obtenidos en la serie de preguntas realizada.

Se da la opción de regresar al menú a salir del juego.

Figura 55. Ventana de Resultados

Anexo 3. Código Fuente del SRCV

En este anexo se presenta el código fuente del sistema que ha sido escrito en leguaje Visual Basic .NET

En referencia al capítulo 5, Diseño del Sistema SRCV, mostraremos el código realizado para los bloques: Interfaz Gráfico-acústica, Generador de Preguntas y Respuestas e Intérprete de Comandos.

Anexo 3.1. Interfaz Gráfico-acústica

Bloque encargado de interactuar directa y fácilmente con el usuario, mostrando una serie de preguntas previamente grabadas y apoyadas con objetos gráficos de fácil comprensión (formas geométricas, símbolos alfanuméricos y colores, entre otros), y aceptando respuestas de voz.

Este bloque se encuentra codificado como:

- Formulario de Inicio
- Formulario de Menú
- Formulario de Preguntas
- Formulario de Resultados

Formulario de Inicio

```
Public Class frmInicio
    'variables que serviran para tener conocimiento de la medida del formulario
    Dim ancho, alto As Integer

    Private Sub Inicio_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        'Se verifica que no existan archivos que produzcan un mal funcionamiento del sistema
        If My.Computer.FileSystem.FileExists("Recursos\reconocedor\0001.rec") Then
            My.Computer.FileSystem.DeleteFile("Recursos\reconocedor\0001.rec")
        End If
        If My.Computer.FileSystem.FileExists("Recursos\reconocedor\comando\0001.rec") Then
            My.Computer.FileSystem.DeleteFile("Recursos\reconocedor\comando\0001.rec")
        End If

        'Se ejecuta el reconocedor de voz
        Reconocedor.ejecutar()

        'Se escalan todos los elementos al tamaño justo correspondiente a la pantalla
        ancho = Me.Width
        alto = Me.Height
        Me.WindowState = FormWindowState.Maximized
        Me.Scale(New SizeF(CSng(Me.Width / ancho), CSng(Me.Height / alto)))
        ancho = Me.Width
        alto = Me.Height

        'Se ejecuta el timer del formulario, la funcion del timer es ejecutar una
        'serie de instrucciones cada cierto tiempo que uno especifique
        Me.miTimer.Start()
```

```
    'Se manda a llamar a un archivo de sonido
    Dim rwmp As New Reproductor
    rwmp.wmp.URL = "Recursos\sonidos\bienvenido.wav"

    'Se da una pausa para que el audio se reproduzca y despues sean liberados
    'los recursos utilizados por el objeto
    Pausa(5)

    'se liberan los recursos utilizados
    rwmp.Dispose()
End Sub

Private Sub miTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles miTimer.Tick
    'Todo el codigo que se encuentre dentro de este bloque sera ejecutado
    'cada cierto tiempo especificado en el timer

    'Lo que hace es forzar al propio SO a que termine o ejecute todo aquello que tiene pendiente
    Application.DoEvents()

    'variable que contendra la palabra detectada por el reconocedor
    Dim s As String

    'El trbajo de este procedimiento es devolver la palabra
    'reconocedor detecto
    s = leerComando()

    If s = "si" Then
        'Se detiene el timer para que ya no ejecute mas el codigo
        Me.miTimer.Dispose()
        Me.miTimer.Stop()

        'Se oculta el formulario de inicio
        Me.Hide()

        'Se crea el formulario menu y se manda a visualizar
        Dim fmenu As New frmMenu
        fmenu.Show()

    ElseIf s = "no" Then
        'Cierra la aplicacion
        End
    ElseIf s <> "" Then 'Si no se dijo la palabra "si" o "no"
        Dim rwmp As New Reproductor
        rwmp.wmp.URL = "Recursos\sonidos\responde_de_nuevo.wav"

        'Se llama al reconocedor para que el sistema espere por una nueva respuesta
        Reconocedor.ejecutar()
        Pausa(5)
        rwmp.Dispose()
    End If
End Sub

Private Sub frmInicio_ResizeEnd(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.ResizeEnd
    'Se escalan todos los elementos al tamaño correspondiente al formualrio
    'si el tamaño del formulario cambia
    Dim c As Control
    For Each c In Me.Controls
        c.Scale(New SizeF(CSng(Me.Width / ancho), CSng(Me.Height / alto)))
    Next
```

```
    ancho = Me.Width
    alto = Me.Height
  End Sub

  Private Sub frmInicio_FormClosing(ByVal sender As Object, ByVal e As
              System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
    'Se ejecuta este codigo cuando se cierra el formulario
    Reconocedor.detener()
    End
  End Sub

End Class
```

Formulario de Menú

```
Public Class frmMenu

  'variable para llamar a archivos de audio
  Dim rwmp As New Reproductor

  'variables que funcionan como indicadores durante el
  'funcionamiento del sistema
  Public Shared opcionMenu As String = "numeros"
  Dim salir As Boolean
  Dim ancho, alto As Integer

  Public Sub New()
    ' Llamada necesaria para el diseñador.
    InitializeComponent()
    ' Agregue cualquier inicialización después de la llamada a InitializeComponent().

    'Se cargan las imagenes para mostrar en el menu
    ptbNumeros.ImageLocation = "Recursos\imagenes\numeros.png"
    ptbColores.ImageLocation = "Recursos\imagenes\arcoiris.png"
    ptbFiguras.ImageLocation = "Recursos\imagenes\figuras.png"
    ptbLetras.ImageLocation = "Recursos\imagenes\letras.png"
    ptbObjetos.ImageLocation = "Recursos\imagenes\objetos.png"
    ptbSalir.ImageLocation = "Recursos\imagenes\emoticon_salir.png"
    ptbNumeros2.ImageLocation = "Recursos\imagenes\numeros.gif"
    ptbNumeros2.Hide()
    ptbColores2.ImageLocation = "Recursos\imagenes\arcoiris.gif"
    ptbColores2.Hide()
    ptbFiguras2.ImageLocation = "Recursos\imagenes\figuras.gif"
    ptbFiguras2.Hide()
    ptbLetras2.ImageLocation = "Recursos\imagenes\letras.gif"
    ptbLetras2.Hide()
    ptbObjetos2.ImageLocation = "Recursos\imagenes\objetos.gif"
    ptbObjetos2.Hide()
    ptbSalir2.ImageLocation = "Recursos\imagenes\emoticon_salir.gif"
    ptbSalir2.Hide()
  End Sub

  Private Sub FrmMenu_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    ancho = Me.Width
    alto = Me.Height
    Me.WindowState = FormWindowState.Maximized
    Me.Scale(New SizeF(CSng(Me.Width / ancho), CSng(Me.Height / alto)))
    ancho = Me.Width
```

```
    alto = Me.Height

    'Posiciona el mouse sobre la opcion Numeros en el menu
    Dim x As Integer = Me.ptbNumeros.Location.X
    Dim y As Integer = Me.ptbNumeros.Location.Y
    Dim al As Integer = Me.ptbNumeros.Size.Height
    Dim an As Integer = Me.ptbNumeros.Size.Width
    Windows.Forms.Cursor.Position = New Point(x + CInt(an / 2), y + CInt(al / 2))
End Sub

Private Sub ptn_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs) Handles
                                    ptbNumeros.MouseEnter, ptbColores.MouseEnter, _
    ptbFiguras.MouseEnter, ptbLetras.MouseEnter, ptbObjetos.MouseEnter, ptbSalir.MouseEnter
    'Este codigo se ejecuta cada vez que el mosue entra sobre una de las opciones en el menu

    If sender Is ptbNumeros Then 'Se identifica si se entro a la opcion Numeros
        'Se oculta la imagen estatica y se muestra un gif de la imagen
        ptbNumeros.Hide()
        ptbNumeros2.Show()
        rwmp.wmp.URL = "Recursos\sonidos\jugarnumeros.wav"

        'Se indica al sistema que se esta sobre la opcion Numeros
        opcionMenu = "numeros"
    ElseIf sender Is ptbColores Then 'Se identifica si se entro a la opcion Colores
        ptbColores.Hide()
        ptbColores2.Show()
        rwmp.wmp.URL = "Recursos\sonidos\jugarcolores.wav"

        'Se indica al sistema que se esta sobre la opcion Colores
        opcionMenu = "colores"
    ElseIf sender Is ptbFiguras Then 'Se identifica si se entro a la opcion Figuras

        ptbFiguras.Hide()
        ptbFiguras2.Show()
        rwmp.wmp.URL = "Recursos\sonidos\jugarfiguras.wav"

        'Se indica al sistema que se esta sobre la opcion Figuras
        opcionMenu = "figuras"
    ElseIf sender Is ptbLetras Then 'Se identifica si se entro a la opcion Letras
        ptbLetras.Hide()
        ptbLetras2.Show()
        rwmp.wmp.URL = "Recursos\sonidos\jugarletras.wav"

        'Se indica al sistema que se esta sobre la opcion Letras
        opcionMenu = "letras"
    ElseIf sender Is ptbObjetos Then 'Se identifica si se entro a la opcion Objetos
        ptbObjetos.Hide()
        ptbObjetos2.Show()
        rwmp.wmp.URL = "Recursos\sonidos\jugarobjetos.wav"

        'Se indica al sistema que se esta sobre la opcion Objetos
        opcionMenu = "objetos"
    ElseIf sender Is ptbSalir Then 'Se identifica si se entro a la opcion Salir
        salir = True
        'opcionMenu = "numeros"
        ptbSalir.Hide()
        ptbSalir2.Show()
        rwmp.wmp.URL = "Recursos\sonidos\terminar.wav"
    End If
```

```
End Sub

Private Sub ptb_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs) Handles
                                    ptbNumeros2.MouseLeave, _
    ptbColores2.MouseLeave, ptbFiguras2.MouseLeave, ptbLetras2.MouseLeave, ptbObjetos2.MouseLeave,
    ptbSalir2.MouseLeave
    'Este codigo se ejecuta cada vez que el mosue sale de una de las opciones en el menu

    If sender Is ptbNumeros2 Then 'Se identifica si se salio de la opcion Nuemros
        'Se detiene la reproduccion de audio
        rwmp.wmp.Ctlcontrols.stop()

        'Se oculta el gif de la imagen y se muestra la imagen estatica
        ptbNumeros2.Hide()
        ptbNumeros.Show()
    ElseIf sender Is ptbColores2 Then 'Se identifica si se salio de la opcion Colores
        rwmp.wmp.Ctlcontrols.stop()
        ptbColores2.Hide()
        ptbColores.Show()
    ElseIf sender Is ptbFiguras2 Then 'Se identifica si se salio de la opcion Figuras
        rwmp.wmp.Ctlcontrols.stop()
        ptbFiguras2.Hide()
        ptbFiguras.Show()
    ElseIf sender Is ptbLetras2 Then 'Se identifica si se salio de la opcion Letras
        rwmp.wmp.Ctlcontrols.stop()
        ptbLetras2.Hide()
        ptbLetras.Show()
    ElseIf sender Is ptbObjetos2 Then 'Se identifica si se salio de la opcion Objetos
        rwmp.wmp.Ctlcontrols.stop()
        ptbObjetos2.Hide()
        ptbObjetos.Show()
    ElseIf sender Is ptbSalir2 Then 'Se identifica si se salio de la opcion Salir

        'Indicador para el sistema de si el usuairo quiere salir o no del juego
        salir = False
        rwmp.wmp.Ctlcontrols.stop()
        ptbSalir2.Hide()
        ptbSalir.Show()
    End If

End Sub

Private Sub miTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles miTimer.Tick
    Application.DoEvents()
    Dim s As String
    s = leerComando()

    If s = "quiero_salir" Then
        'variable que indica si se quiere salir del jeugo
        salir = True

        rwmp.wmp.URL = "Recursos\sonidos\terminar.wav"

        'El mouse se posiciona sobre la opcion salir en el menu
        Dim x As Integer = Me.ptbSalir.Location.X
        Dim y As Integer = Me.ptbSalir.Location.Y
        Dim an As Integer = Me.ptbSalir.Size.Width
        Dim al As Integer = Me.ptbSalir.Size.Height
        Windows.Forms.Cursor.Position = New Point(x + CInt(an / 2), y + CInt(al / 2))
```

```
    Reconocedor.ejecutar()

ElseIf s = "si" And salir Then
    'Se detiene el reconocedor y se cierra el juego solo si
    'se dijo "si" y la variable salir contiene el valor True
    Reconocedor.detener()
    End

ElseIf s = "si" Then

    'Indicador para el sistema de si el usuairo quiere salir o no del juego
    salir = False

    Reconocedor.detener()
    Me.miTimer.Dispose()
    Me.miTimer.Stop()
    Me.Hide()

    'Se asigna el numero total de preguntas correspondiente a la opcion elegida
    'por el usuario en el menu
    frmPregunta.totalP = CInt((My.Computer.FileSystem.GetFiles("Recursos\preguntas\" & opcionMenu).Count –
                                1) / 3)

    'Se llama a la funcion para asignar un numero aleatorio de pregunta
    frmPregunta.np = aleatorio(frmPregunta.totalP)

    'Lleva la cuenta del numero de preguntas que se han realizado
    frmPregunta.contador = frmPregunta.contador + 1

    'Se declara variable para usar la funcion generadorPR
    Dim gpr As New GeneradorPR

    'A esta funcion se le da como parametro la opcion del menu elegida por el usario
    'y un numero aleatorio de pregunta
    gpr.generadorPR(opcionMenu, frmPregunta.np)

ElseIf s = "no" Then

    'Se identifica en que opcion del menu se encuentra el sistema para posicionar el
    'mouse a la siguiente opcion cuando se dijo "no"
    If opcionMenu = "numeros" Then
        Dim x As Integer = Me.ptbColores.Location.X
        Dim y As Integer = Me.ptbColores.Location.Y
        Dim an As Integer = Me.ptbColores.Size.Width
        Dim al As Integer = Me.ptbColores.Size.Height
        Windows.Forms.Cursor.Position = New Point(x + CInt(an / 2), y + CInt(al / 2))
    ElseIf opcionMenu = "colores" Then
        Dim x As Integer = Me.ptbLetras.Location.X
        Dim y As Integer = Me.ptbLetras.Location.Y
        Dim an As Integer = Me.ptbLetras.Size.Width
        Dim al As Integer = Me.ptbLetras.Size.Height
        Windows.Forms.Cursor.Position = New Point(x + CInt(an / 2), y + CInt(al / 2))
    ElseIf opcionMenu = "letras" Then
        Dim x As Integer = Me.ptbFiguras.Location.X
        Dim y As Integer = Me.ptbFiguras.Location.Y
        Dim an As Integer = Me.ptbFiguras.Size.Width
        Dim al As Integer = Me.ptbFiguras.Size.Height
        Windows.Forms.Cursor.Position = New Point(x + CInt(an / 2), y + CInt(al / 2))
    ElseIf opcionMenu = "figuras" Then
```

```
            Dim x As Integer = Me.ptbObjetos.Location.X
            Dim y As Integer = Me.ptbObjetos.Location.Y
            Dim an As Integer = Me.ptbObjetos.Size.Width
            Dim al As Integer = Me.ptbObjetos.Size.Height
            Windows.Forms.Cursor.Position = New Point(x + CInt(an / 2), y + CInt(al / 2))
        ElseIf opcionMenu = "objetos" And salir = False Then
            Dim x As Integer = Me.ptbSalir.Location.X
            Dim y As Integer = Me.ptbSalir.Location.Y
            Dim an As Integer = Me.ptbSalir.Size.Width
            Dim al As Integer = Me.ptbSalir.Size.Height
            Windows.Forms.Cursor.Position = New Point(x + CInt(an / 2), y + CInt(al / 2))
        ElseIf salir Then
            Dim x As Integer = Me.ptbNumeros.Location.X
            Dim y As Integer = Me.ptbNumeros.Location.Y
            Dim an As Integer = Me.ptbNumeros.Size.Width
            Dim al As Integer = Me.ptbNumeros.Size.Height
            Windows.Forms.Cursor.Position = New Point(x + CInt(an / 2), y + CInt(al / 2))
        End If
        salir = False
        Reconocedor.ejecutar()
    ElseIf s <> "" Then
        'Se reproduce el archivo de audio
        rwmp.wmp.URL = "Recursos\sonidos\responde_de_nuevo.wav"
        Reconocedor.ejecutar()
    End If
End Sub

Private Sub frmMenu_VisibleChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles
                                        Me.VisibleChanged
    'El codigo se ejecuta cada vez que se visualiza el menu
    If Me.Visible = True Then
        Reconocedor.detener()
        Reconocedor.ejecutar()
        Me.miTimer.Start()

        'este codigo posiciona el mouse sobre la opcion Numeros en el menu
        Dim x As Integer = Me.ptbNumeros.Location.X
        Dim y As Integer = Me.ptbNumeros.Location.Y
        Dim al As Integer = Me.ptbNumeros.Size.Height
        Dim an As Integer = Me.ptbNumeros.Size.Width
        Windows.Forms.Cursor.Position = New Point(x + CInt(an / 2), y + CInt(al / 2))
    End If
End Sub

Private Sub FrmMenu_ResizeEnd(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.ResizeEnd
    Dim c As Control
    For Each c In Me.Controls
        c.Scale(New SizeF(CSng(Me.Width / ancho), CSng(Me.Height / alto)))
    Next
    ancho = Me.Width
    alto = Me.Height
End Sub

Private Sub FrmMenu_FormClosing(ByVal sender As Object, ByVal e As
                                System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
    Reconocedor.detener()
    End
End Sub
```

```
End Class
```

Formulario de Preguntas

```
Public Class frmPregunta

    'Variables que seran usadas durante la ejecucion como indicadores
    'por lo cual el sistema tomara acciones en referencia a estos valores
    Public Shared aciertos, fallos, maxFallos, np, totalP, contador As Integer
    Dim evaluacion, irMenu, salir As Boolean
    Dim ancho, alto As Integer

    Public Sub New()
        ' Llamada necesaria para el diseñador.
        InitializeComponent()
        ' Agregue cualquier inicialización después de la llamada a InitializeComponent().

        'Se carga la imagen y el gif de la opcion "regresar al menu"
        'en la realizacion de la preguntas
        ptbMenu1.ImageLocation = "Recursos\imagenes\menu.png"
        ptbMenu2.ImageLocation = "Recursos\imagenes\menu.gif"
        ptbMenu2.Hide()
    End Sub

    Private Sub frmPregunta_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    MyBase.Load
        'Se muestra la imagen estatica de la opcion "regresar al menu" y se
        'oculta el gif de la imagen
        ptbMsj.Hide()
        ptbMenu2.Hide()

        'Se asigna el el numero de aciertos y fallos que tiene el usuario
        'durante la realizacion de las preguntas
        labAciertos.Text = aciertos.ToString
        labFallos.Text = fallos.ToString

        ancho = Me.Width
        alto = Me.Height
        Me.WindowState = FormWindowState.Maximized
        Me.Scale(New SizeF(CSng(Me.Width / ancho), CSng(Me.Height / alto)))
        ancho = Me.Width
        alto = Me.Height
    End Sub

    Private Sub repetir_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
                repetir.Click
        'Se asiga el valor False para saber que el usuario no quiere regresar
        'al menu o salir del juego
        irMenu = False
        salir = False

        Reconocedor.detener()
        Reconocedor.ejecutar()

        'Se reproduce de nuevo la pregunta
        Dim rwmp As New Reproductor
        rwmp.wmp.URL = "Recursos\preguntas\" & frmMenu.opcionMenu & "\p" & np & ".wav"
```

```
    Pausa(4)
    rwmp.Dispose()
End Sub

Private Sub miTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
            miTimer.Tick
    Application.DoEvents()
    Dim s As String
    s = leerComando()

    'Se indica al sistema que no haga nada mientras la variable "s" este vacia
    If s <> String.Empty Then
        If s = "quiero_salir" Then
            Reconocedor.ejecutar()
            salir = True
            Dim rwmp As New Reproductor
            rwmp.wmp.URL = "Recursos\sonidos\terminar.wav"
            Pausa(3)
            rwmp.Dispose()

        ElseIf s = "si" And salir Then
            'Si el usuario quiere salir del juego
            Reconocedor.detener()
            End
        ElseIf s = "si" And irMenu Then '
            'Si el usuario quiere regresar al menu
            Application.OpenForms.Item("frmMenu").Show()

            'Se reinician las variables relacionadas con el control
            'de la realizacion de las preguntas
            salir = False
            aleatorio(0)
            frmPregunta.contador = 0
            frmPregunta.aciertos = 0
            frmPregunta.fallos = 0
            frmPregunta.maxFallos = 0
            Reconocedor.detener()

            'Se detiene el timer del formulario de preguntas
            Me.miTimer.Dispose()
            Me.miTimer.Stop()
            Me.Dispose()

        ElseIf irMenu Or salir Then
            'Si se dijo una palabra diferente a "si" despues de que el sistema pregunta
            'si se quiere regresar al menu o salir del juego

            Reconocedor.ejecutar()
            irMenu = False
            salir = False

            'Se reprodice de nuevo la pregunta
            Dim rwmp As New Reproductor
            rwmp.wmp.URL = "Recursos\preguntas\" & frmMenu.opcionMenu & "\p" & np & ".wav"
            Pausa(4)
            rwmp.Dispose()

        Else
```

```
            'Se crea la variable para usar la funcion Interprete
            Dim int As New Interprete

            'Esta funcion recibe el tipo de pregunta a realizar y un numero aleatorio
            'devuelve True si la respuesta del usuario es correcta o False si es incorrecta
            evaluacion = int.interprete(s, frmMenu.opcionMenu, np)

            If evaluacion = True Then
                'Se muestra una imagen correspondiente a una respuesta correcta
                ptbMsj.ImageLocation = "Recursos\imagenes\emoticon_bien.gif"
                ptbMsj.SizeMode = PictureBoxSizeMode.StretchImage
                ptbMsj.Show()
                Me.miTimer.Dispose()
                Me.miTimer.Stop()
            Else
                'Se muestra una imagen correspondiente a una respuesta incorrecta
                ptbMsj.ImageLocation = "Recursos\imagenes\emoticon_fallo.gif"
                ptbMsj.SizeMode = PictureBoxSizeMode.StretchImage
                ptbMsj.Show()
            End If

            'Se llama al generador de preguntas y respuestas, se le da como parametro un boolean
            'el cual utiliza para indicar al usuario si fue correcta o no su respuesta y en el
            'caso de que si, se encarga de mostrar la siguiente pregunta
            Dim gpr As New GeneradorPR
            gpr.generadorPR(evaluacion)

            'Limpia la imagen mostrada en la pregunta
            ptbMsj.ImageLocation = ""
            ptbMsj.Hide()

            'Se asigna el numero de aciertos y fallos del usuario
            labAciertos.Text = aciertos.ToString
            labFallos.Text = fallos.ToString
        End If
    End If
End Sub

Private Sub ptbMenu1_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs) Handles
ptbMenu1.MouseEnter

    'Se ejecuta este codigo cuando el mouse entra en la opcion "regresar al menu"
    'durante la fase de realizacion de preguntas
    irMenu = True
    salir = False
    ptbMenu1.Hide()
    ptbMenu2.Show()
    Dim rwmp As New Reproductor
    rwmp.wmp.URL = "Recursos\sonidos\menu.wav"
End Sub

Private Sub ptbMenu2_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs) Handles
ptbMenu2.MouseLeave
    'Este codigo se jecuta cuando el mouse sale de la opcion "regresar al menu"
    ptbMenu2.Hide()
    ptbMenu1.Show()
End Sub
```

```
Private Sub frmPregunta_VisibleChanged(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.VisibleChanged
    'El codigo se ejecuta cada vez que se realiza una nueva pregunta
    If Me.Visible = True Then
        Reconocedor.ejecutar()
        Me.miTimer.Start()
    End If
End Sub

Private Sub frmPregunta_ResizeEnd(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.ResizeEnd
    Dim c As Control
    For Each c In Me.Controls
        c.Scale(New SizeF(CSng(Me.Width / ancho), CSng(Me.Height / alto)))
    Next
    ancho = Me.Width
    alto = Me.Height
End Sub

Private Sub frmPregunta_FormClosing(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
    Reconocedor.detener()
    End
End Sub

End Class
```

Formulario de Resultados

```
Public Class frmResultados

    'Variables que se usaran como indicadores durante la ejecucion del sistema
    Dim ancho, alto As Integer
    Dim irMenu, salir As Boolean

    'Se crea variable para poder llamar a archivos de audio
    Dim rwmp As New Reproductor


    Private Sub frmResultados_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    MyBase.Load

        'Se cargan las imagenes correspondientes al formulario de resultados
        ptbMenu1.ImageLocation = "Recursos\imagenes\menu.png"
        ptbMenu2.ImageLocation = "Recursos\imagenes\menu.gif"
        ptbMenu2.Hide()
        ptbSalir1.ImageLocation = "Recursos\imagenes\emoticon_salir.png"
        PtbSalir2.ImageLocation = "Recursos\imagenes\emoticon_salir.gif"
        PtbSalir2.Hide()
        rwmp.wmp.URL = "Recursos\sonidos\resultados.wav"

        'Se asigna el numero de aciertos y fallos que se obtuvieron durante
        'la realizacion de las preguntas
        labAciertos.Text = frmPregunta.aciertos.ToString
        labFallos.Text = frmPregunta.fallos.ToString
        'Se escalan los objetos al tamaño correspondiente al formulario
        ancho = Me.Width
        alto = Me.Height
```

```
    Me.WindowState = FormWindowState.Maximized
    Me.Scale(New SizeF(CSng(Me.Width / ancho), CSng(Me.Height / alto)))
    ancho = Me.Width
    alto = Me.Height
    ptbMenu2.Hide()
    PtbSalir2.Hide()

    'Se inicis el Timer del formulario
    Me.miTimer.Start()
End Sub

Private Sub ptbMenu1_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs) Handles
 ptbMenu1.MouseEnter
    'Este codigo se ejecuta cada vez que el mouse esta sobre la opcion
    '"regresar al menu"

    rwmp.wmp.URL = "Recursos\sonidos\menu.wav"
    Reconocedor.detener()
    Reconocedor.ejecutar()

    'Se indica al sistema que se quiere regresar al menu
    irMenu = True
    ptbMenu1.Hide()
    ptbMenu2.Show()
End Sub

Private Sub ptbMenu2_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs) Handles
ptbMenu2.MouseLeave
    'Se ejecuta cuando el mouse deja de posicionarse sobre la opcion "regresar al menu"

    rwmp.wmp.Ctlcontrols.stop()
    Reconocedor.detener()
    irMenu = False
    ptbMenu2.Hide()
    ptbMenu1.Show()
End Sub

Private Sub ptbSalir1_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs) Handles
ptbSalir1.MouseEnter
    rwmp.wmp.URL = "Recursos\sonidos\salir.wav"
    Reconocedor.detener()
    Reconocedor.ejecutar()

    'Se indica al sistema que se quiere salir del juego
    salir = True
    ptbSalir1.Hide()
    PtbSalir2.Show()
End Sub

Private Sub ptbSalir2_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs) Handles
PtbSalir2.MouseLeave
    rwmp.wmp.Ctlcontrols.stop()
    Reconocedor.detener()
    salir = False
    PtbSalir2.Hide()
    ptbSalir1.Show()
End Sub

Private Sub miTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles miTimer.Tick
```

```
        Application.DoEvents()
        Dim s As String
        s = leerComando()
        If s <> String.Empty Then
            If s = "si" Then
                If irMenu Then
                    'Se visualiza el menu y el formulario actual de resultados es destruido
                    Application.OpenForms.Item("frmMenu").Show()
                    Me.miTimer.Dispose()
                    Me.miTimer.Stop()
                    Me.Dispose()
                ElseIf salir Then
                    'Se cierra la aplicacion
                    End
                End If

            Else
                If irMenu Then
                    'Si se dijo una palabra diferente de "si" y estando posicionado en la opcion
                    'regresar a menu, el mouse pasa automaticamente a la opcion salir
                    Dim x As Integer = Me.ptbSalir1.Location.X
                    Dim y As Integer = Me.ptbSalir1.Location.Y
                    Dim an As Integer = Me.ptbSalir1.Size.Width
                    Dim al As Integer = Me.ptbSalir1.Size.Height
                    Windows.Forms.Cursor.Position = New Point(x + CInt(an / 2), y + CInt(al / 2))
                ElseIf salir Then
                    'Se pasa automaticamente a la opcion "regresar a menu"
                    Dim x As Integer = Me.ptbMenu1.Location.X
                    Dim y As Integer = Me.ptbMenu1.Location.Y
                    Dim an As Integer = Me.ptbMenu1.Size.Width
                    Dim al As Integer = Me.ptbMenu1.Size.Height
                    Windows.Forms.Cursor.Position = New Point(x + CInt(an / 2), y + CInt(al / 2))
                End If
            End If
        End If
    End Sub

    Private Sub frmResultados_ResizeEnd(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.ResizeEnd
        Dim c As Control
        For Each c In Me.Controls
            c.Scale(New SizeF(CSng(Me.Width / ancho), CSng(Me.Height / alto)))
        Next
        ancho = Me.Width
        alto = Me.Height
    End Sub

    Private Sub frmResultados_FormClosing(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
        Reconocedor.detener()
        End
    End Sub

End Class
```

Anexo 3.2. Intérprete de Comandos

Su trabajo es analizar y verificar si las respuestas del usuario son correctas. La función de este bloque está apoyada por un Banco de Reactivos.

```
Public Class Interprete

  Function interprete(ByVal comando As String, ByVal tPreguntas As String, ByVal nPregunta As Integer) As Boolean

    'variable que devuelve la funcion e indica si la respuesta dada
    'por el usuario es correcta o no
    Dim evaluacion As Boolean = False

    'Se usa para guardar la respuesta correspondiente a la pregunta
    'que se esta realizando
    Dim respuesta As String = String.Empty

    'Se busca la respuesta a la pregunta hecha
    respuesta = My.Computer.FileSystem.OpenTextFileReader("Recursos\preguntas\" & tPreguntas & "\r" & nPregunta
    & ".txt").ReadLine

    'Se compara la respuesta con lo que se dijo
    If comando = respuesta Then
      evaluacion = True
    End If

    'devuelve True si es correcta la respuesta o False si es incorrecta
    Return evaluacion
  End Function

End Class
```

Anexo 3.3. Generador de Preguntas y Respuestas

Este bloque utiliza un Banco de Reactivos para seleccionar y enviar a la interfaz gráfico-acústica la pregunta a mostrar al usuario. También se encarga de enviar a la interfaz gráfico-acústica los mensajes de evaluación de las respuestas del usuario.

```
Public Class GeneradorPR
  Sub generadorPR(ByVal tPreguntas As String, ByVal nPregunta As Integer)
    'Se crea la variable con la cual se llama a la nueva preegunta
    Dim fp As New frmPregunta

    'ayudan a obtenr el nombre de la imagen correspondiente a la pregunta
    Dim nomArchivo As String = String.Empty, nomImagen As String

    'Se busca el nombre de la imagen a mostrar en la pregunta
    For Each nomArchivo In My.Computer.FileSystem.GetFiles("Recursos\preguntas\" & tPreguntas)
      nomImagen = nomArchivo.Substring(nomArchivo.LastIndexOf("\") + 1)
      If nomImagen(0) = "i" Then
        If nomImagen(1) = nPregunta.ToString Then
          Exit For
        ElseIf nomImagen(1) & nomImagen(2) = nPregunta.ToString Then
          Exit For
        End If
      End If
```

```
    Next

    'Se visualiza la imagen en el formulario
    nomImagen = nomArchivo.Substring(nomArchivo.LastIndexOf("\") + 1)
    fp.ptb.SizeMode = PictureBoxSizeMode.StretchImage
    fp.ptb.ImageLocation = "Recursos\preguntas\" & tPreguntas & "\" & nomImagen

    'Se ajusta la imagen al tamaño del cuadro que la contiene
    fp.ptbFondo.SizeMode = PictureBoxSizeMode.StretchImage

    'Se visualiza el fondo
    fp.ptbFondo.ImageLocation = "Recursos\preguntas\" & tPreguntas & "\fondo.jpg"

    'Se verifica si no hay una instancia del formulario pregunta
    If Application.OpenForms.Item("frmPregunta") Is Nothing = False Then
        Application.OpenForms.Item("frmPregunta").Dispose()
    End If
    'Se llama al formualrio
    fp.Show()

    'Se llama al archivo de sonido el cual es una pregunta para el usuario
    Dim rwmp As New Reproductor
    rwmp.wmp.URL = "Recursos\preguntas\" & tPreguntas & "\p" & nPregunta & ".wav"
End Sub

Sub generadorPR(ByVal evaluacion As Boolean)
    If evaluacion Then
        'Se llama al archivo de sonido el cual indica que la respuesta
        'ha sido correcta
        Dim rwmp As New Reproductor
        rwmp.wmp.URL = "Recursos\sonidos\muy_bien.wav"

        'lleva la cuenta de los erroes consecutivos en una misma pregunta
        'cuando su valor sea 3 el sistema pasa a la siguiente pregunta
        frmPregunta.maxFallos = 0

        'Se incrementa en uno los aciertos
        frmPregunta.aciertos += 1
        Pausa(4)
        rwmp.Dispose()

        If frmPregunta.contador < frmPregunta.totalP Then

            'Si todavia no se llega a preguntar el numero total de preguntas
            'el contador se incrementa en uno
            frmPregunta.contador = frmPregunta.contador + 1

            'Se obtiene un numero aleatorio de pregunta
            frmPregunta.np = aleatorio(frmPregunta.totalP)

            'Se llama a la nueva pregunta
            Dim gpr As New GeneradorPR
            gpr.generadorPR(frmMenu.opcionMenu, frmPregunta.np)
        Else
            'Este codigo se ejecuta cuando ya se han realizado todas las
            'preguntas de la opcion elegida en el menu
            Dim fr As New frmResultados
            fr.Show()
```

```
            'Se reinician las variables con las que se contrala la
            'realizacion de las preguntas
            aleatorio(0)
            frmPregunta.contador = 0
            frmPregunta.aciertos = 0
            frmPregunta.fallos = 0

            'Se destruye el formulario de pregunta
            Application.OpenForms.Item("frmPregunta").Dispose()
        End If

    Else
        'Se ejecuta este codigo si la respuesta es incorrecta
        frmPregunta.fallos += 1
        frmPregunta.maxFallos += 1

        'Se llama al archivo de sonido el cual indica que la respuesta
        'fue incorrecta
        Dim rwmp As New Reproductor
        rwmp.wmp.URL = "Recursos\sonidos\incorrecto.wav"
        Reconocedor.detener()
        Reconocedor.ejecutar()
        Pausa(4)
        rwmp.Dispose()

        'Se verifica si se ha tenido 3 errores en una misma pregunta
        'si es asi se pasa a la siguiente pregunta siempre y cuando
        'halla mas pregunstas que hacer, si no el sistema pasa al
        'formulario de resultados
        If frmPregunta.maxFallos = 3 Then
            frmPregunta.maxFallos = 0
            If frmPregunta.contador < frmPregunta.totalP Then
                frmPregunta.contador = frmPregunta.contador + 1
                frmPregunta.np = aleatorio(frmPregunta.totalP)
                Dim gpr As New GeneradorPR
                gpr.generadorPR(frmMenu.opcionMenu, frmPregunta.np)
            Else
                Dim fr As New frmResultados
                fr.Show()
                aleatorio(0)
                frmPregunta.contador = 0
                frmPregunta.aciertos = 0
                frmPregunta.fallos = 0
                Application.OpenForms.Item("frmPregunta").Dispose()
            End If
        End If
    End If
End Sub

End Class
```

GLOSARIO

vocabulario del sistema: son todas aquellas palabras que pueden ser reconocidas por él sistema.

patrones de voz: son el resultado de la extracción de características distintivas de una señal de voz que sirven como modelos de referencia para ser utilizados durante el proceso de reconocimiento.

modelos ocultos de Markov: es una de las técnicas que se ha utilizado con más éxito en el reconocimiento de voz ya permite modelar adecuadamente la gran variabilidad en el tiempo de la señal de voz.

librería: en ciencias de la computación, una librería es un conjunto de subprogramas utilizados para desarrollar software, contienen código y datos, que proporcionan servicios a programas independientes**.**

entrenamiento: es una fase durante el desarrollo de un reconocedor de voz y consiste en la creación de modelos o patrones de cada una de las unidades (silabas, palabras, frases) con la que se va a comparar el habla.

fonética acústica: es la rama de la lingüística que estudia la producción y percepción de los sonidos de una lengua.

conjunto discreto: es todo aquel conjunto que contiene un número finito de valores. Ejemplo: se tiene una urna con esferas del 1 al 5, por lo tanto sus valores discretos serán 1, 2, 3, 4 o 5 y ningún otro.

formantes: Técnicamente los formantes son bandas de frecuencia donde se concentra la mayor parte de la energía sonora de un sonido.

fonema: Puede calificarse al fonema como la unidad mínima del lenguaje oral, ya que se trata de los sonidos del habla que permiten diferenciar entre las palabras de una lengua.

transformada de Fourier: se utiliza para analizar la señal de voz en el dominio de la frecuencia.

autómata: es un modelo matemático que realiza cómputos en forma automática sobre una entrada para producir una salida.

síntesis: La síntesis de habla es la producción artificial de habla humana. Un sistema usado con este propósito recibe el nombre de sintetizador de habla y puede llevarse a cabo en software o en hardware.

transcripción: es la conversión de lengua hablada a caracteres escritos.

archivo etiquetado: se refiere a asociar en un archivo (por ejemplo de texto) cada unidad lingüística (palabra, silaba, fonema etc.) contenidas en un archivo de audio de acuerdo con las ondas de voz.

fonio: también llamado fon, fon es una unidad de medida logarítmica y adimensional (similar al decibelio) que se usa para indicar la sonoridad con que se percibe un sonido dado.

k-media: El algoritmo de k-means es el referente principal entre los diversos métodos para seleccionar grupos representativos entre los datos.

algoritmo de Viterbi: permite encontrar las secuencias de estados más probable en un Modelo oculto de Márkov.