DISEÑO DE DISPOSITIVO PARA EL RECONOCIMIENTO DE CARACTERES VOCÁLICOS, PARA ORDENAR COMANDOS AL TELEVISOR.

JAVIER ALFREDO BÁEZ DÁVILA.

UNIVERSIDAD DE SAN BUENAVENTURA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C.
2006

DISEÑO DE DISPOSITIVO PARA EL RECONOCIMIENTO DE CARACTERES VOCÁLICOS, PARA ORDENAR COMANDOS AL TELEVISOR.

JAVIER ALFREDO BÁEZ DÁVILA.

Trabajo de grado para optar por el título de
Ingeniero Electrónico

ASESOR
Ing. Pedro Luís Muñoz

UNIVERSIDAD DE SAN BUENAVENTURA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C.
2006

NOTA DE ACEPTACIÓN

_____

_____

_____

PRESIDENTE DEL JURADO

_____

JURADO

_____

JURADO

BOGOTÁ, MAYO DE 2006

A Dios, mis padres, hermanos, y amigos, que me apoyaron y depositaron su confianza durante el tiempo en que realicé mis estudios profesionales.

AGRADECIMIENTOS

El autor expresa sus agradecimientos a:

El Ing. Andrés Quiroga, por su colaboración incondicional.

Al asesor de investigación Ing. Pedro Muñoz, por su constante apoyo y sus valiosas orientaciones.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

GLOSARIO

- ✓ RAH: RECONOCIMIENTO AUTOMÁTICO DEL HABLA.
- ✓ DTW: ALINEAMIENTO TEMPORAL DINÁMICO.
- ✓ HMM: MODELOS OCULTOS DE MARKOV.
- ✓ NN: REDES NEURONALES.
- ✓ FIR: RESPUESTA FINITA AL IMPULSO.
- ✓ IIR: INFINITA RESPUESTA AL IMPULSO.
- ✓ PIC: MICROCONTROLADOR DE MICROCHIP.
- ✓ CPU: UNIDAD CENTRAL DE PROCESO.
- ✓ CAD: CONVERSOR ANÁLOGO DIGITAL.
- ✓ CDA: CONVERSOR DIGITAL ANÁLOGO.
- ✓ RISC: COMPUTADOR DE JUEGO DE INSTRUCCIONES REDUCIDO.
- ✓ CISC: COMPUTADORES DE JUEGO DE INSTRUCCIONES COMPLEJO.
- ✓ SISC: COMPUTADORES DE JUEGO DE INSTRUCCIONES ESPECÍFICO.
- ✓ FLASH: MEMORIA VOLÁTIL DE BAJO CONSUMO, QUE SE PUEDE ESCRIBIR Y BORRAR.
- ✓ GPR: REGISTRO DE PROPÓSITO GENERAL.
- ✓ SFR: REGISTROS DE FUNCIONES ESPECIALES.
- ✓ BSR: REGISTRO DE SELECCIÓN DE BANCO.
- ✓ PC: CONTADOR DE PROGRAMA.
- ✓ EPROM: ERASABLE PROGRAMMABLE READ ONIY MEMORY
- ✓ EEPROM: ELECTRICAL ERASABLE PROGRAMMABLE READ ONIY MEMORY
- ✓ ALU: UNIDAD ARITMÉTICA LÓGICA.

- ✓ E/S: ENTRADA/SALIDA.
- ✓ MSSP: MASTER SYNCHRONOUS PORT.
- ✓ SPI: SERIAL PERIPHERAL INTERFACE.
- ✓ I²C: INTER-INTEGRATED CIRCUIT.
- ✓ USART: ADAPTADOR DE COMUNICACIÓN SERIE SÍNCRONA Y ASÍNCRONA.
- ✓ AUSART: TRANSMISOR ASINCRONO UNIVERSAL DIRECCIONABLE DEL RECEPTOR.
- ✓ PWM: MODULADOR DE ANCHURA DE IMPULSOS
- ✓ SCL: TERMINAL DE ENTRADA DE RELOJ.
- ✓ SDA: TERMINAL DE ENTRADA/SALIDA.
- ✓ PTC: ENTRADA DEL RELOJ EXTERNO.
- ✓ RAM: MEMORIA VOLÁTIL.
- ✓ ROM: MEMORIA NO VOLÁTIL, DE SÓLO LECTURA.
- ✓ WATCHDOG: PERRO GUARDIAN.
- ✓ SLAVE: ESCLAVO.
- ✓ MASTER: MAESTRO.
- ✓ VDD: VOLTAJE DE ALIMENTACIÓN.
- ✓ LED: DIODO EMISOR DE LUZ.
- ✓ CIB: CONSEJO DE INVESTIGACIÓN BONAVENTURIANO.
- ✓ USB: UNIVERSIDAD DE SAN BUENAVENTURA.

RESUMEN

Este proyecto presenta, un sistema para el reconocimiento de caracteres vocálicos, para ordenar comandos al televisor; Mediante la técnica de alineamiento temporal dinámico DTW, que permite independencia del intervalo de tiempo de cada muestra de voz. Los resultados obtenidos por el dispositivo, demuestran que el uso de esta técnica permite obtener un 80% de clasificación correcta.

Al igual, describe la realización de un prototipo capaz de interpretar comandos vocales basados en un microcontrolador (PIC 18f452), como lo es el hardware mínimo necesario así como el enfoque y la implementación software necesario para llevar a cabo este proyecto.

Por medio de este dispositivo se podrá obtener respuesta a las órdenes transmitidas por voz al televisor, encendido, apagado, canal arriba y abajo, aumento y disminución de volumen. Este modo de relación liberará completamente al usuario del uso de la vista y de las manos (es decir de la pantalla y del control), y le deja libertad de movimientos.

TÍTULO DEL PROYECTO

DISEÑO DE DISPOSITIVO PARA EL RECONOCIMIENTO DE CARACTERES VOCÁLICOS, PARA ORDENAR COMANDOS AL TELEVISOR.

INTRODUCCIÓN

Debido a la presente necesidad de comunicación hombre-máquina, se ha inducido a la técnica del reconocimiento automático del habla (RAH), tanto en empresas tecnológicas como en Universidades. Por ejemplo, nuevos productos de control por voz como: robots industriales o electrodomésticos, sistemas de ayuda a discapacitados, acceso y navegación por base de datos, operaciones y transacciones comerciales, operaciones telefónicas automáticas, etc.

Comprendiendo está necesidad se pretende crear un sistema de ayuda para todas las personas que usan televisor en sus vidas como recreación y distracción, para obtener una manipulación más sencilla; por medio de la técnica de Reconocimiento Automática de Habla (RAH), utilizando para ello Alineamiento Temporal Dinámico (DWT). La importancia de desarrollar esta técnica, se debe, a que las características lingüísticas difieren en forma marcada cuando se requiera llevar hacia un reconocimiento más completo. El enfoque que se ha dado en un principio ha sido el reconocer palabras aisladas, es decir que las palabras se pronuncian entre pausas pequeñas.

Este documento describe la realización de un prototipo capaz de interpretar comandos vocales basados en un microcontrolador (PIC 18f452), como lo es el hardware mínimo necesario así como el enfoque y la implementación software necesario para llevar a cabo este proyecto.

1. PLANTEAMIENTO DEL PROBLEMA

1.1 ANTECEDENTES

Actualmente, gran parte de los laboratorios y empresas dedicadas a tareas relacionadas con las tecnologías del reconocimiento automático del habla (RAH), tanto en USA, Japón como Europa, centran sus esfuerzos en el desarrollo de sistemas con capacidad para comprender el habla. Este desarrollo se conoce internacionalmente como Comprensión de la Lengua Hablada (Spoken Language Understanding). Hasta hace muy poco, la mayoría de los esfuerzos estuvieron encaminados hacia la obtención de un mayor conocimiento sobre diversos paradigmas como son los Modelos Ocultos de Markov (HMM), las Redes Neuronales (NN), o la combinación de Redes Neuronales y Modelos Ocultos de Markov (HMM + NN), entre otros, que permitan reconocer habla con la mayor tasa posible a nivel acústico, es decir, descodificar la señal acústica y generar la secuencia de palabras que más probablemente habría producido la secuencia de símbolos acústicos de entrada.

Dos de los objetivos más importantes dentro de lo que se ha desarrollado el reconocimiento automático del habla (RAH) en los últimos tiempos son:

- Traducción automática vocal, que permita traducir una conversación entre 2 locutores que hablen lenguas diferentes, en tiempo real, así como traducción automática a partir de texto.
- Sistemas de Interacción Vocal capaces de mantener un diálogo hombre-máquina y traducir la petición vocal del usuario a una serie de órdenes a la máquina, permitiendo incluso que ésta le responda o le interrogue para completar el significado de la pregunta. Estos sistemas se desarrollan normalmente en entorno telefónico.

Actualmente no existe desarrollo de un dispositivo con estas características, en el banco de proyectos de grado en la universidad de San Buenaventura.

1.2 DESCRIPCIÓN DEL PROBLEMA.

Este proyecto muestra la problemática y la necesidad actual del Reconocimiento Automático del Habla (RAH), como que el locutor habla con una naturalidad total, con dudas, posibles repeticiones, paradas, etc. La dificultad del reconocimiento es máxima. El fin es obtener una manipulación más sencilla de los sistemas de televisión, gracias a un sistema desarrollado por RAH; no sólo desde un punto de vista investigativo sino también técnico.

La combinación del reconocimiento automático del habla y de la comprensión del lenguaje incrementa enormemente el número y el rango de aplicaciones potenciales para ambas tecnologías. El reconocimiento automático del habla sin comprensión del mensaje hablado conduce a una simple trascripción textual de las palabras habladas, pero si añadimos una interpretación a esas palabras estaremos abriendo un amplio abanico de posibilidades en la interacción hombre-máquina. La tecnología para el procesamiento del lenguaje natural sin reconocimiento de habla necesita que el usuario escriba su petición al sistema, utilizando sus manos, ojos y centrando su atención en dicho proceso. Si el usuario se viese liberado del uso de ellos, le permitiría, con una mayor eficiencia, utilizar pantallas de visualización, manejar ratones, solucionar problemas de forma interactiva mientras realiza otras tareas que mantienen ocupados sus ojos y sus manos, e interactuar naturalmente con sistemas o aplicaciones remotas de cierta complejidad. Al emplear sistemas capaces de procesar el lenguaje natural hablado, el usuario puede centrarse más en la solución de su problema y menos en la propia interacción con el sistema.

El diseño de una interfase hombre-máquina con este tipo de capacidades y su integración en una aplicación muchas veces ya existente no es una tarea sencilla, sobre todo por la escasez de experiencias hasta el momento, es decir, otros

sistemas donde observar y aprender. Uno de los problemas principales a la hora de implementar estos sistemas es la necesidad de datos para entrenar y evaluar los mismos. Algunos de los problemas o cuestiones que se plantean a la hora de desarrollar una aplicación que utilice esta tecnología son:

- Elección de un *dominio inicial* que justifique la necesidad de utilizar esta tecnología.
- Decidir la *arquitectura* para integrar ambas tecnologías: reconocimiento de habla y procesamiento del lenguaje.
- Conocimiento y uso de las características del *habla espontánea* generada cuando el usuario desea conseguir algún objetivo (goal-directed spontaneous speech).
- Cómo *evaluar* estos sistemas de comprensión del lenguaje hablado (Spoken Language Understanding Systems).

Por todo ello, la tarea del sistema de reconocimiento de habla no es sencilla, y además, es costosa, tanto en memoria como en cálculo.

1.2.1 FORMULACIÓN DEL PROBLEMA.

¿Cómo diseñar el dispositivo que permite el reconocimiento automático del habla y así obtener la respuesta que se desea para ordenar comandos al televisor?

1.3 JUSTIFICACIÓN

La creciente necesidad de mejorar la comunicación hombre – máquina, ha inducido a la técnica del reconocimiento automático del habla (RAH), hacia un inusitado interés, debido a esto, se plantea una ayuda para toda persona que usa televisor en sus vidas como recreación y distracción, para obtener una manipulación más sencilla.

Este modo de relación libera completamente al usuario del uso de la vista y de las manos (es decir de la pantalla y control remoto), dándole libertad de movimientos y permitiéndole una mayor eficiencia para realizar otras actividades centrándose más en la solución de problemas y menos en la propia interacción con el sistema.

Con el desarrolloó de este proyecto, se pretende realizar una forma de comunicación como es el lenguaje hablado hacia las máquinas. Desarrollando la técnica de reconocimiento automática de habla (RAH), cuya función es obtener una secuencia de palabras, pronunciadas por el locutor de forma continúa (sin pausas); y realizando alineamiento temporal dinámico (DWT), cuyo objetivo es identificar las palabras pronunciadas por un locutor de entre todas las posibles de un vocabulario.

El tratamiento mediante el que se va a reconocer se puede dividir en tres etapas: grabación, parametrización y comparación. En la primera etapa se hace la grabación (proceso en tiempo real) del comando a reconocer. En segundo lugar se hace la parametrización en frecuencia de la señal para obtener así la forma compacta y fácilmente tratable en que están almacenadas las muestras de comparación. Finalmente, en la comparación se elige la referencia más cercana a la señal recién procesada, completándose el proceso de reconocimiento. Para llevar a cabo la implementación se utilizo el microcontrolador 18f452, añadiendo

un integrado-memoria ME24LC256, para que de esta manera contribuya con el almacenamiento de las palabras.

1.4 OBJETIVOS

1.4.1 OBJETIVO GENERAL

Diseñar e implementar un dispositivo de reconocimiento de caracteres vocálicos (voz), para ordenar comandos a un televisor.

1.4.2 OBJETIVOS ESPECÍFICOS

- Desarrollar un sistema que ordene comandos por voz a un televisor mediante el Reconocimiento Automático del Habla utilizando Alineamiento Temporal Dinámico (DWT).

- Desarrollar un sistema de ayuda por voz, para todo tipo de persona con acceso a televisor sin necesidad de levantarse o manipular su control remoto; (encender, apagar, canal arriba, canal abajo, aumento y disminución de volumen).

- Implementar un sistema R.A.H. que tenga la capacidad de reconocer palabras o comandos manteniendo un nivel adecuado frente a cambios de usuario.

- Implementar el software necesario para obtener el reconocimiento de la voz.

- Implementar un prototipo capaz de interpretar comandos vocales basados en un microcontrolador (PIC 18f452), obteniendo el mejor resultado posible.

1.5 ALCANCES Y LIMITACIONES DEL PROYECTO

1.5.1 ALCANCES

El desarrolloó del dispositivo de Reconocimiento Automático del Habla (RAH), para ordenar comandos a un televisor, será realizado hasta la implementación del prototipo obteniendo las ventajas que se esperan de la comunicación hombre-máquina.

Por medio de este dispositivo se podrá obtener respuesta a las órdenes transmitidas por voz al televisor, encendido, apagado, canal arriba y abajo, aumento y disminución de volumen. Este modo de relación liberará completamente al usuario del uso de la vista y de las manos (es decir de la pantalla y del control), y le deja libertad de movimientos.

Las operaciones de codificación y descodificación quedan reducidas a un mínimo, principalmente cuando se utiliza el lenguaje natural. Estas y otras ventajas se traducen en una gran variedad de aplicaciones actuales y de futuro. En el campo industrial, es interesante poder controlar una máquina-herramienta, un robot, o proporcionar datos a un sistema de diseño o producción asistido por ordenador manteniendo las manos y la vista libre, e incluso poder comunicar observaciones durante una inspección o un control de calidad conservando la libertad de movimientos.

1.5.2 LIMITACIONES

Este sistema presenta problemas de cobertura léxica entre otros, que permiten reconocer la voz con la mayor tasa posible a nivel acústico, es decir, descodificar la señal acústica y generar la secuencia de palabras que más probablemente habría producido la secuencia de símbolos acústicos de entrada. Pero eso no es suficiente para descodificar el significado del mensaje reconocido acústicamente.

Sin duda alguna esta de las más grandes limitaciones, puesto que el sistema es muy sensible al tono de voz del interlocutor; a un usuario cuyos comandos de referencia no estén en la base de datos no le va a ser posible usar correctamente el sistema; la fiabilidad del reconocimiento disminuye conforme aumenta el número de comandos; no se reconocen comandos erróneos, ya que siempre buscará un emparejamiento con uno de los comandos predefinidos.

Para resolver el problema de la Comprensión del Habla, es necesaria la combinación de distintas fuentes de conocimiento lingüístico (fonológico, morfológico, léxico, sintáctico, semántico, pragmático) y de comportamiento humano (desde el punto de vista de modelos de diálogo hombre-máquina, modelos de usuario, etc.) y estrategias para llevar a cabo esa combinación desde un punto de vista computacional con eficiencia.

Por todo ello, la tarea del sistema de reconocimiento de habla no es sencilla, y además, es costosa, tanto en memoria como en cálculo.

2. MARCO DE REFERENCIA

2.1 MARCO CONCEPTUAL

2.1.1 Reconocimiento Automática de Habla.

En términos generales, los sistemas RAH, son la implementación algorítmica de un análisis detallado de las características del habla, y que dependerá del lenguaje usado y de otros conceptos que no deben excluirse (frecuencia fundamental, energía del tramo de señal, etc.). Un sistema RAH puede contemplar: Reconocimiento de palabras aisladas, identificación de palabras clave en discurso continuo, Reconocimiento de palabras conectadas y reconocimiento de discurso continúo.

2.2.1.1 Dificultades del RAH.

- **Variabilidad acústica:**
  - locutor: diferentes actitudes/emociones, ritmos, entonación.
  - ruido de ambiente
  - canal

- **Variabilidad fonética**
  - idiolectos
  - coarticulación
  - supresión/reducción de fonemas: “ir a comé”

- **Existencia de información no acústica**
  - léxica
  - sintáctico/semántica
  - pragmática/diálogo

2.1.2 Alineamiento Temporal Dinámico (DWT).

DWT (Dynamic Time Warping), puede traducirse como "comparación de patrones usando algoritmos de programación dinámica". Consiste en comparar los parámetros extraídos de la palabra a reconocer con aquellos que pertenecen al vocabulario almacenado a partir de la base de datos. El resultado de esta operación es una medida de distancia entre la muestra a reconocer y la "más cercana" a ella del vocabulario.

Mediante esta técnica podemos representar a la señal vocal mediante parámetros que varían en el tiempo los cuales están relacionados con la función de transferencia del tracto vocal y las características de la fuente sonora. Otra ventaja es que no requiere demasiado tiempo de procesamiento, lo cual es importante a la hora de la implementación.

2.1.3 La Voz.

La voz humana es producida en la laringe, cuya parte esencial, la glotis, constituye el verdadero órgano de fonación humano. El aire procedente de los pulmones, es forzado durante la espiración a través de la glotis, haciendo vibrar los dos pares de cuerdas vocales, que se asemejan a dos lengüetas dobles membranáceas. Las

cavidades de la cabeza, relacionadas con el sistema respiratorio y nasofaríngeo, actúan como resonadores.

2.1.3.1 Modelo de Producción de la Voz.

El modelo clásico del tracto vocal se compone de un filtro variable en el tiempo, un generador de ruido aleatorio y de un generador de impulsos. Los parámetros del filtro varían en función de la acción consciente que se realiza al pronunciar una palabra.

2.1.4 Micrófono.

Convierte la onda sonora en una señal eléctrica (forma de onda)

2.1.5 Amplificación.

Es el tipo más común de acondicionamiento, para conseguir la mayor precisión posible de la señal de entrada deber ser amplificada de modo que su máximo nivel coincida con la máxima tensión que el convertidor pueda leer.

2.1.6. Digitalización.

Esta etapa adapta el valor medio nulo de la señal de entrada, que centran la señal respecto al margen de operación del conversor analógico digital.

2.1.6.1 Conversor A/D (CAD).

Los microcontroladores que incorporan un Conversor A/D (Analógico/Digital) pueden procesar señales analógicas, tan abundantes en las aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del CAD diversas señales analógicas desde los pines del circuito integrado.

2.1.6.2 Conversor D/A (CDA).

Transforma los datos digitales obtenidos del procesamiento del computador en su correspondiente señal analógica que saca al exterior por una de los pines del circuito integrado. Existen muchos efectores que trabajan con señales analógicas.

2.1.6.3 Filtrado.

El fin de un filtro es eliminar las señales no deseadas de la señal que se está observando.

2.1.7 Procesamiento digital de señales.

El procesamiento de señales trata de la representación, transformación y manipulación de señales y de la información que contienen. Por ejemplo, podríamos desear separar dos o más señales que se han combinado de alguna forma, o podríamos querer realzar alguna componente de la señal o algún parámetro de un modelo de señal. Este procesamiento se puede realizar aplicando tecnología analógica en tiempo continuo, o como se ha ido difundiendo cada vez más, aplicando procesamiento en tiempo discreto mediante programas y procesadores.

2.1.7.1 Análisis en el Dominio del Tiempo.

Para analizar el comportamiento temporal de un filtro digital se considera que a la entrada se le aplica una secuencia determinada. Para ello se utilizan una serie de funciones elementales, que generan cada una de ellas distintas secuencias.

2.1.7.2 La Transformada Z.

La transformada Z se utiliza para el análisis de filtros digitales lineales e invariantes en el tiempo. Transforma ecuaciones de diferencias en expresiones algebraicas, lo que simplifica los cálculos.

2.1.7.3 Filtro digital.

El análisis de un filtro digital es el proceso de determinar la respuesta de un filtro ante una dada excitación. El *diseño* de un filtro digital es el proceso de sintetizar e implementar un filtro digital de tal manera que cumpla con las especificaciones prescriptas.

2.1.7.4 Filtros Recursivos (IIR).

Los filtros digitales recursivos también se denominan "de Respuesta Infinita al Impulso", y también se los identifica con la sigla IIR (Infinite Impulse Response). La respuesta de un filtro IIR es función de la excitación y también de las respuestas anteriores.

La siguiente es la expresión de un filtro digital recursivo, lineal, invariante en el tiempo y causal:

$$y(nT) = \sum_{i=0}^{N} a_i x(nT - iT) - \sum_{i=1}^{N} b_i y(nT - iT)$$

2.1.8 Programación Dinámica.

La programación dinámica es una técnica de resolución que encuentra aplicación en numerosos problemas de optimización, pero que no se limita a estos.
El tipo de problemas de optimización abordables por programación dinámica es, principalmente, aquél en el que las soluciones factibles se pueden descomponer en una secuencia de elementos (caminos en un grafo, series de decisiones, etc.) y cuya función objetivo satisface cierta condición de separabilidad y monotonía.

2.1.9 Memoria I^2C

El funcionamiento de la memoria I^2C , se divide en dos procesos de trabajo:

- Proceso de escritura
- Proceso de lectura

2.1.10 Microcontrolador.

Un microcontrolador es un computador completo (microprocesador + E/S +memoria + otros periféricos), aunque de limitadas prestaciones, que esta contenido con el chip de un circuito integrado programable y se destina a gobernar

una sola tarea con el programa que reside en su memoria, sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores.

2.1.10.1 PIC18F452

El PIC18F452 ofrece un ambiente amistoso del desarrollo del recopilador de ' C ', 256 octetos de EEPROM, Uno mismo-programando, un ICD, 2 funciones de capture/compare/PWM, 8 canales del convertidor (DE ANALÓGICO A DIGITAL) de analógico a digital 10-bit, el puerto serial síncrono se puede configurar como el interfaz periférico serial 3-wire (SPI™) o el autobús Inter-Integrado de dos hilos del circuito (I²C™) y el transmisor asincrónico universal direccionable del receptor (AUSART). Todas estas características hacen ideal para el equipo de fabricación, la instrumentación y supervisar, la adquisición de datos, la energía que condiciona, el control del medio ambiente, el telecom y los usos del consumidor audio/video.

2.1.10.2 Memoria de Programa.

El contador de Programa PC, tiene un tamaño de 21 bits y proporciona la dirección de la instrucción a la que se accede. Con 21 bits se puede direccional hasta 2 Mbytes de memoria de programa. La memoria de programa la utilizaremos en el proyecto para el código.

Los microcontroladores dotados de memoria EEPROM una vez instalados en el circuito, pueden grabarse y borrarse cuantas veces se quiera sin ser retirados de dicho circuito. Para ello se usan “grabadores en circuito” que confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo.

2.2 MARCO TEÓRICO

2.2.1 Reconocimiento Automático del Habla (RAH).

2.2.1.1 Historia.

La historia del reconocimiento del habla empezó en el año 1870. Alexander Graham Bell, quiso desarrollar un dispositivo que fuera capaz de proporcionar la palabra visible para la gente que no escuchara. Bell no tuvo éxito creando este dispositivo, sin embargo, el esfuerzo de esta investigación condujo al desarrollo del teléfono. Más tarde, en los años 30, Tihamer Nemes, científico húngaro quiso patentar el desarrollo de una máquina para la trascripción automática de la voz. La petición de Nemes fue negada y a este proyecto lo llamaron poco realista.

Fue hasta 1950, 80 años después del intento de Bell, cuando se hizo el primer esfuerzo para crear la primera máquina de reconocimiento del habla. La investigación fue llevada a los laboratorios de AT&T. El sistema tuvo que ser entrenado para reconocer el discurso de cada locutor individual mente, pero una vez especializada la maquina tenia una exactitud de un 99% de reconocimiento.

El primer sistema de reconocimiento del habla fue desarrollado en 1952 sobre una computadora analógica que reconocía dígitos del 0 al 9, este sistema era dependiente del locutor. Los experimentos dieron una exactitud de reconocimiento del 98%. Más tarde, se creo un sistema que reconocía consonantes y vocales.

Durante los 60´s, los investigadores que trabajaban en el área de reconocimiento de voz empezaron a comprender la complejidad del desarrollote una verdadera

aplicación dentro del reconocimiento del habla, y se comenzaron a realizar aplicaciones con vocabularios pequeños, dependientes del locutor y con palabras de flujo discreto. El flujo discreto es la forma como hablan los locutores, es decir, con pequeñas pausas entre palabras y frases. También, durante 1960, la Universidad de **Carnegie Mellon** y la compañia **IBM**, empezaron una investigación en reconocimiento del habla continua. El impacto de esta investigación se reflejó hasta después de los años 70´s.

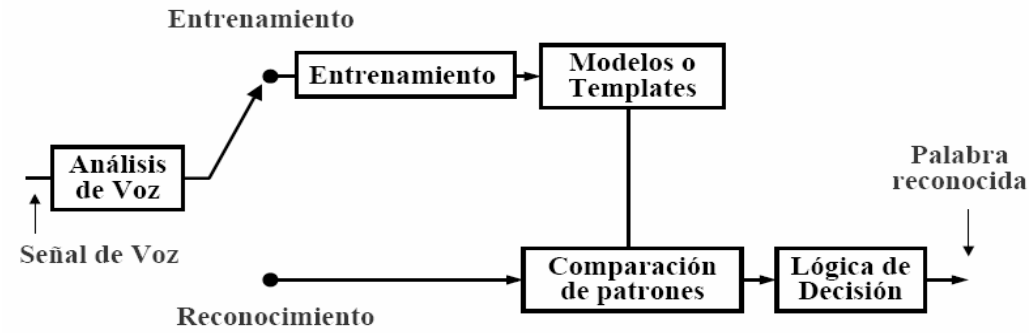
Para los 70´s, se desarrolló el primer sistema de reconocimiento del habla comercial. Se mejoraron las aplicaciones de los sistemas dependientes del locutor que requerían una entrada discreta y tenían un vocabulario pequeño. Por otra parte la **Advanced Research Projects Agency (ARPA),** de la sección Americana de Defensa se mostró interesada en la investigación del reconocimiento del habla. ARPA comenzó investigaciones enfocándose al habla continua, usando vocabularios más extensos. También se mejoró la tecnología de reconocimiento para palabras aisladas y continuas. En esta misma época se desarrollaron técnicas para el reconocimiento del habla tales como **Time Warping**, modelo probabilística y el algoritmo de retropropagación.

Durante los 80´s, el reconocimiento del habla se favoreció por tres factores: el crecimiento de computadoras personales, el apoyo de ARPA y los costos reducidos de aplicaciones comerciales. El mayor interés durante este periodo de tiempo era el desarrolló de vocabularios grandes. En 1985, un vocabulario de 100 palabras era considerado grande. Sin embargo en 1986, hubo uno de 20.000 palabras. También durante esta época hubo grandes avances tecnológicos, ya que se cambió del enfoque basado en reconocimiento de patrones a métodos de modelado probabilísticos, como los Modelos Ocultos de Markov (HMM).

Para los años 90´s, los costos de reconocimiento del habla continuaron decreciendo y los vocabularios extensos comenzaron a ser normales. También las aplicaciones independientes del locutor y de flujo continuo (es decir sin pausas significantes) comenzaron hacer más comunes.

Aunque han existido a lo largo de la corta historia del reconocimiento automático del habla distintos enfoques, los más utilizados y que han proporcionado los mejores resultados han sido los probabilísticos, basados en la Teoría de la Decisión de Bayes, la Teoría de la Información y las Técnicas de Comparación de Patrones y de Programación Dinámica.

Figura 1. Reconocimiento de Voz basado en Comparación de Patrones.



Fuente: Reconocimiento Automático de Voz basado en Técnicas de Comparación de Patrones.pdf

Dentro de este enfoque, por sistema de reconocimiento automático del habla entendemos un sistema de cierta complejidad capaz de descodificar los sonidos u otra información de nivel superior que forman parte de una determinada señal de habla. Dicha descodificación puede realizarse de diferentes formas, utilizando diferentes técnicas y con unos determinados requisitos de partida para la señal de habla a decodificar. En el fondo, se trata de ser capaces de generar un conjunto

de patrones (asociados a partes de habla) que puedan ser comparados con la señal acústica de entrada (a reconocer) devolviendo la secuencia de estos patrones que con mayor probabilidad representan a la misma.

El proceso de reconocimiento automático de habla tiene como función obtener la secuencia de palabras asociada a la frase en lenguaje natural (LN) de entrada. La frase es pronunciada por el locutor de forma continua, es decir, sin pausas entre las palabras, y a menudo, tiene problemas de agramaticalidad, incluye elementos propios del habla espontánea como son las interjecciones, falsos comienzos, repeticiones, etc. El sistema de reconocimiento de habla, por su parte, presenta problemas de cobertura léxica y sintáctica (estructural). Por todo ello, la tarea del sistema de reconocimiento de habla no es sencilla, y además, es costosa, tanto en memoria como en cálculo.

2.2.1.2 El Problema del Reconocimiento Automático del Habla.

La cuestión principal ha sido determinar cuáles son las posibles causas que hacen tan difícil llevar a cabo un reconocimiento automático del habla en condiciones generales, de forma que se pueden buscar soluciones parciales a cada uno de los problemas y conseguir una solución global lo más óptima posible. Algunas de estas causas son[1]:

- Las variaciones de fonación, debidas a los hablantes. Ninguna persona habla igual, es decir, los sonidos producidos por diferentes personas no suenan igual (mantienen ciertas relaciones formánticas pero no son copias exactas entre los diferentes locutores).

[1] José Colás Pasamontes. http://elies.rediris.es/elies12. 14/10/2005, 11:22.

- Las ambigüedades acústicas. A veces no es posible mapear eventos acústicos a sus símbolos fonéticos correspondientes, no es posible una buena decodificación al no disponer de todas las fuentes de conocimiento que una persona utiliza durante el proceso de una conversación.
- Variaciones de producción y que le desvían del registro teórico ideal. Por ejemplo:

    - Falta de cuidado al pronunciar algunas palabras. A veces ciertas palabras de duración breve se omiten (palabras función como preposiciones, conjunciones, etc.) o se transforman en sonidos extraños. También, a veces la velocidad que adquiere un hablante al pronunciar las palabras es demasiado elevada de forma que no existe una clara transición entre las diferentes sílabas, llegando a la fusión u omisión de algunas de ellas.
    - Variaciones fonéticas. Las frecuencias de los formantes, el “locus” y las duraciones de las transiciones pueden cambiar a lo largo del tiempo, lo que produce un cierto alejamiento de los patrones o reglas utilizadas durante el reconocimiento.
    - Coarticulación. Las características acústicas de los sonidos se ven afectadas por el contexto en el que se encuentran. La mayoría de estos efectos se traducen en alófonos que tiene cada fonema. Ello supone la necesidad de tener múltiples patrones que tengan en cuenta estas variaciones.
    - Variaciones temporales. La duración de una palabra e incluso de los sonidos puede cambiar, generando la necesidad de realizar alineamientos dinámicos, que permitan tener en cuenta a estas posibles variaciones.

- Ruidos e interferencias. Las personas podemos reconocer habla en condiciones adversas, es decir, con baja relación señal/ruido e incluso en presencia de otros sonidos interferentes, gracias a las propias características del oído humano y a la audición binaural (todavía no bien entendida).

Debido a todos estos problemas y algunos más, toda tarea de reconocimiento automático de habla continua, se enfrenta a la necesidad de tener que tomar decisiones a pesar de la falta de información asociada a cada uno de ellos, teniendo en cuenta que en cualquier sistema existe una interdependencia entre las decisiones tomadas en diferentes niveles. Si fuese posible reconocer alófonos o palabras con una alta tasa de acierto, no sería necesario utilizar técnicas de decisión que necesitan ciertos retardos, técnicas de corrección de errores y en métodos estadísticos. Además, la solución a estos problemas no parece ser posible en un futuro cercano, con lo que los sistemas de reconocimiento deben tratar con una gran cantidad de hipótesis a nivel de alófonos, de palabras o de frases, e idealmente tienen que tener en cuenta las "restricciones de alto nivel" aportadas por la sintaxis, la semántica y la pragmática textual.

La Teoría de la Decisión Estadística nos enseña cómo minimizar la probabilidad de cometer errores durante el reconocimiento, es decir, encontrar la secuencia de palabras que tienen la mayor probabilidad de estar asociada a la secuencia de observaciones acústicas de entrada. A través del Teorema de Bayes sobre la probabilidad condicional, el problema anterior se puede volver a escribir de modo que la búsqueda de la secuencia de palabras se convierte en un problema de buscar la secuencia de palabras que producen un máximo de probabilidad a priori (modelo de lenguaje) y que además, producen la secuencia de observaciones con máxima probabilidad (modelo acústico). Es decir ha dividido el problema en dos

posibles soluciones (un problema de decodificación lingüística y otro de decodificación acústica).

Para resolver el nuevo problema necesitaremos modelar las restricciones propias de la lengua a través del modelo gramatical y la probabilidad de observar la secuencia de observaciones acústicas cuando el locutor o locutores pronuncien la secuencia de palabras de la frase, probabilidad estimada durante la fase de entrenamiento de los HMM. La decisión acerca de las palabras reconocidas se toma mediante el uso de un procedimiento de optimización que utiliza información de diversas fuentes: el modelo de lenguaje, los modelos acústico-fonéticos asociados a los distintos alófonos que darán lugar a las palabras, y el diccionario que indica la composición de las palabras a reconocer según los alófonos modelados. A este procedimiento de optimización se le conoce como procedimiento de "búsqueda en un espacio de estados" que estará definido por la interacción de las diferentes fuentes de conocimiento. A veces, por razones de coste computacional y de memoria, se simplifica el espacio de búsqueda, eliminando la interrelación inherente entre el modelo lingüístico y el acústico, con el consiguiente aumento de la entropía del sistema y el riesgo de aumentar el error de reconocimiento (sistemas no integrados).

2.2.1.3 Técnicas más usadas en los Sistemas de Reconocimiento Automático del Habla. (R.A.H.)

Dependiendo del reconocedor que deseemos implementar, se pueden usar distintas técnicas. Las más usadas son:

2.2.1.3.1 Dynamic Time Warping (DTW).

(Es la utilizada en este proyecto de grado), puede traducirse como "comparación de patrones usando algoritmos de programación dinámica". Consiste en comparar los parámetros extraídos de la palabra a reconocer con aquellos que pertenecen al vocabulario almacenado a partir de la base de datos. El resultado de esta operación es una medida de distancia entre la muestra a reconocer y la "más cercana" a ella del vocabulario. La distancia puede ser matizada o mejorada usando información adicional, como pueden ser unas reglas gramaticales, tablas de frecuencia de aparición estadísticas, etc.

2.2.1.3.2 Modelos ocultos de Markov (HMM).

El modelado estocástico de la señal de habla, proporciona los mejores resultados hasta la fecha tanto para el reconocimiento de habla aislada como continua y para independencia del locutor. En el fondo la filosofía de comparación de patrones subyace en este tipo de aproximación al problema pero difiere en la forma en la que se obtienen los patrones, el tipo de patrón, la medida de distancia y la forma de realizar el alineamiento temporal utilizando estos últimos. Ahora, utilizamos un algoritmo de alineamiento no lineal (Programación Dinámica) conocido como algoritmo de Viterbi, capaz de “alinear” la secuencia de vectores de entrada o índices de un codebook con el conjunto de patrones estocásticos (HMM) que representan las palabras del diccionario, en forma de la probabilidad de que esa secuencia sea observada (generada) por los distintos Modelos Ocultos de Markov.

2.2.1.3.3 Redes Neuronales (NN)

Las redes neuronales son estructuras de procesamiento paralelo de información, formadas por numerosos nodos simples conectados entre sí mediante pesos y

agrupados en diferentes capas, entre las que se deben distinguir la capa de entrada y la capa de salida. Debido a su naturaleza intrínsecamente no lineal, a su capacidad de clasificación, y sobre todo a la capacidad que tienen para aprender una determinada tarea a partir de pares, observación-objetivo sin hacer suposición alguna sobre el modelo subyacente, se han convertido en una de las herramientas más atractivas para la solución del problema del reconocimiento de habla. Hoy en día se han conseguido resultados comparables a los obtenidos con otros métodos ya clásicos como los HMM. Sin embargo, presentan diferentes problemas o inconvenientes como pueden ser: desconocimiento a priori de la estructura de capas y número de nodos necesarios para cada problema; un tiempo a veces excesivamente elevado para su entrenamiento y la posibilidad de quedar "anclados" en mínimos locales de las funciones de coste usadas durante el entrenamiento de la red. Además, la señal de habla requiere de métodos con capacidad de proceso en dos dimensiones, espacio y tiempo, y las redes neuronales, por sí solas, sólo tienen capacidad de procesado espacial. Ello nos obliga a combinar técnicas de Programación Dinámica así como HMM con estas redes, consiguiendo modelar la variable tiempo, permitiendo no sólo la clasificaciones muy acertadas de las entradas de la red sino además la segmentación de la señal de entrada. Sin embargo, se han probado otras soluciones que incorporen a las redes algún tipo de memoria (finita, lazos de realimentación, o ambas), pero dificulta en gran medida el análisis de estas redes debido a su carácter no lineal.

2.2.2. La Voz.

La voz humana es producida en la laringe, cuya parte esencial, la glotis, constituye el verdadero órgano de fonación humano. El aire procedente de los pulmones, es forzado durante la espiración a través de la glotis, haciendo vibrar los dos pares de

cuerdas vocales, que se asemejan a dos lengüetas dobles membranáceas. Las cavidades de la cabeza, relacionadas con el sistema respiratorio y nasofaríngeo, actúan como resonadores.

CARACTERÍSTICAS DE LA VOZ HUMANA

- ESPECTRO:

El timbre de la voz humana es muy rico en armónicos, habiéndose observado espectros conteniendo hasta 35 armónicos diferentes.

- DURACIÓN:

Duración variable, con duración máxima fija (tiempo que una persona puede espirar aire sin inspiración). La duración de las vocales es relativamente grande. La duración promedio de una vocal es de 0.2 seg. A 1seg.

- ENVOLVENTE:

En la producción de una vocal se siguen los periodos de ataque, extinción y estado estacionario. Durante los tres hay un cambio considerable en el espectro acústico.

El estado estacionario muestra el menor cambió en las componentes en frecuencia. El ataque y la extinción son relativamente largos, aunque pueden variar sobre un amplio rango. Esto es debido a la naturaleza altamente resonante de la cavidad bucal.

2.2.2.1 La voz como sonido.

La física ha establecido que para que exista sonido se requieren tres elementos:

1. Un cuerpo que vibre.
2. Un medio elástico que vibre (las ondas sonoras son mecánicas que se propagan por la expansión y compresión del propio medio).

3. Una caja de resonancia que amplifique esas vibraciones, permitiendo que sean percibidas por el oído.

La voz humana cumple con las tres condiciones señaladas:

1. El cuerpo elástico que vibra son las cuerdas vocales.
2. El medio elástico es el aire.
3. La caja de resonancia está formada por parte de la garganta, por la boca y por la cavidad nasal.

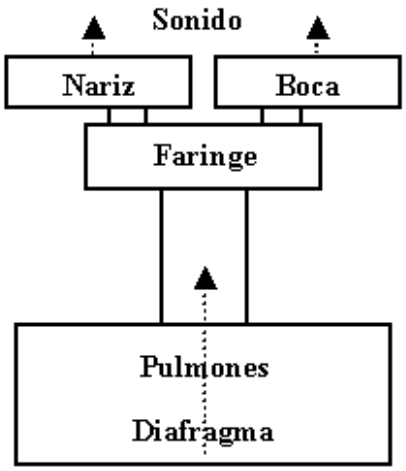
2.2.2.2 Aparato fonador humano.

El aparato fonador lo componen 3 grupos de órganos diferenciados:

1. Órganos de respiración (Cavidades infraglóticas: pulmones, bronquios y tráquea).
2. Órganos de fonación (Cavidades glóticas: laringe, cuerdas vocales y resonadoras -nasal, bucal y faríngeo-).
3. Órganos de articulación (cavidad supraglóticas: paladar, lengua, dientes, labios y glotis).

Además, el correcto funcionamiento del aparato fonador lo controla el sistema nervioso central. Específicamente, se sabe que el control del habla se realiza en el área de Broca situada en el hemisferio izquierdo de la corteza cerebral

Figura 2. Sistema Vocal Humano



El sistema vocal humano puede dividirse en tres partes:

- Aparato respiratorio: donde se almacena y circula el aire. Nariz, traquea, pulmones y diafragma.
- Aparato de fonación: donde el aire se convierta en sonido. Laringe y cuerdas vocales.
- Aparato resonador: donde el sonido adquiere sus cualidades de timbre que caracterizan cada voz. Cavidad bucal, faringe, paladar óseo, senos maxilares y frontales.

Fuente: http://laenciclopedialibre/La voz humana.htm. 11/10/2005, 16:44.

2.2.2.3 La producción de habla.

La fonación se realiza durante la **espiración**, cuando el aire contenido en los pulmones, sale de éstos y, a través de los bronquios y la tráquea, llega a la laringe.

En la laringe se encuentran las cuerdas vocales. Las cuerdas vocales no tienen forma cordófona sino que se trata de una serie de repliegues o labios.

Hay 4 cuerdas vocales:

- 2 superiores (bandas ventriculares), que no participan en la articulación de la voz.

- 2 inferiores, las verdaderas cuerdas vocales, responsables de la producción de la voz.

Las dos cuerdas inferiores son dos pequeños músculos elásticos:

- Si se abren y se recogen a los lados, el aire pasa libremente, sin hacer presión: respiramos.
- Si, por el contrario, se juntan, el aire choca contra ellas, produciendo el sonido que denominamos *voz*.

Hay 3 mecanismos básicos de producción de voz:

- Vibración de las cuerdas que produce los sonidos tonales o sonoros (vocales, semivocales, nasales, etc.).
- Las interrupciones (totales o parciales) en el flujo de aire que sale de los pulmones que da lugar a los sonidos "sordos" (fricativas, etc.)
- La combinación de vibración e interrupción, como las oclusivas sonoras (en español 'b', 'd' y 'g').

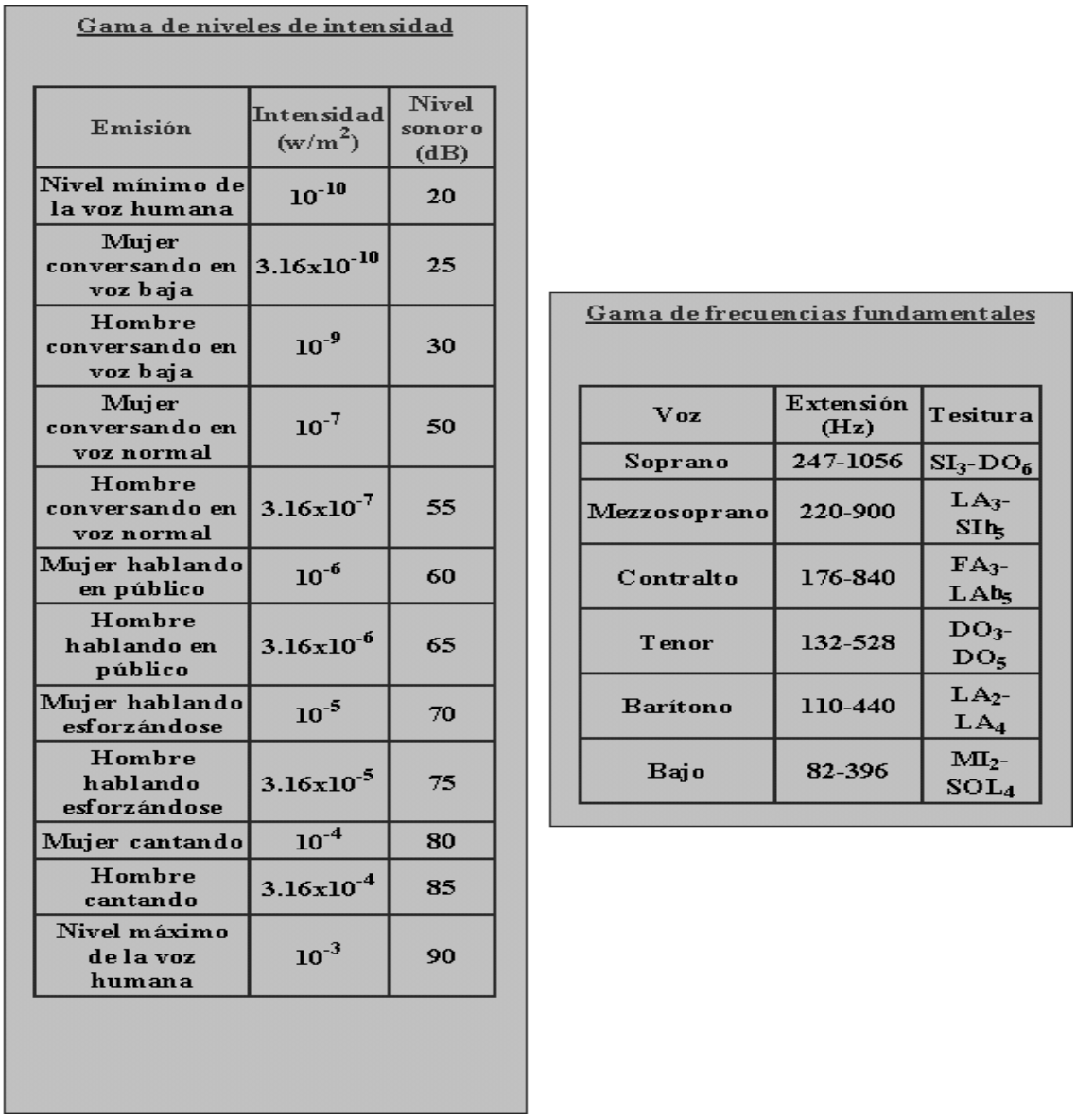
El rango vocal lo determina la flexibilidad de las cuerdas vocales, que permite diferenciar los distintos tipos de voces (en *canto*: tenor, soprano, contralto, ...), en función de la altura, intensidad y timbre.

El sonido producido en las cuerdas vocales es muy débil; por ello, debe ser amplificado. Esta amplificación tendrá lugar en los resonadores nasal, bucal y faríngeo, donde se producen modificaciones que consisten en el aumento de la frecuencia de ciertos sonidos y la desvalorización de otros.

La voz humana, una vez que sale de los resonadores, es moldeada por los articuladores (paladar, lengua, dientes, labios y glotis), transformándose en

sonidos del habla: fonemas, sílabas, palabras, etc. La posición concreta de los articuladores determinará el sonido que emita la voz.

Figura 3. Gama de nivel de intensidad y frecuencia de la voz.



Gama de niveles de intensidad

Emisión	Intensidad (w/m^2)	Nivel sonoro (dB)
Nivel mínimo de la voz humana	10^{-10}	20
Mujer conversando en voz baja	$3.16x10^{-10}$	25
Hombre conversando en voz baja	10^{-9}	30
Mujer conversando en voz normal	10^{-7}	50
Hombre conversando en voz normal	$3.16x10^{-7}$	55
Mujer hablando en público	10^{-6}	60
Hombre hablando en público	$3.16x10^{-6}$	65
Mujer hablando esforzándose	10^{-5}	70
Hombre hablando esforzándose	$3.16x10^{-5}$	75
Mujer cantando	10^{-4}	80
Hombre cantando	$3.16x10^{-4}$	85
Nivel máximo de la voz humana	10^{-3}	90

Gama de frecuencias fundamentales

Voz	Extensión (Hz)	Tesitura
Soprano	247-1056	SI_3-DO_6
Mezzosoprano	220-900	LA_3-SIb_5
Contralto	176-840	FA_3-LAb_5
Tenor	132-528	DO_3-DO_5
Barítono	110-440	LA_2-LA_4
Bajo	82-396	MI_2-SOL_4

Fuente: http://laenciclopedialibre/La voz humana.htm. 11/10/2005, 16:44.

2.2.3 Procesamiento digital de señales.

El procesamiento de señales trata de la representación, transformación y manipulación de señales y de la información que contienen. Por ejemplo, podríamos desear separar dos o más señales que se han combinado de alguna forma, o podríamos querer realzar alguna componente de la señal o algún parámetro de un modelo de señal. Este procesamiento se puede realizar aplicando tecnología analógica en tiempo continuo, o como se ha ido difundiendo cada vez más, aplicando procesamiento en tiempo discreto mediante programas y procesadores[2].

Si las señales a tratar son analógicas, deberán ser convertidas en una secuencia de muestras, a fin de ser procesadas mediante algún algoritmo. Luego, de ser necesario serán vueltas a convertir en señales analógicas. Un ejemplo de esto es el filtrado de señales de audio.

Es común que se denomine a esta forma de procesamiento, indistintamente, como procesamiento digital de señales o procesamiento de señales en tiempo discreto. Una buena parte del procesamiento de señales involucra el proceso de una señal para obtener otra señal: es el caso del *filtrado digital*.
Otra buena parte del procesamiento de señales comprende la *interpretación de señales.* En este caso no se intenta obtener una señal de salida, sino una caracterización de la señal de entrada. Un ejemplo de este tipo de procesamiento es el reconocimiento de voz.

[2] J. Proakis, Ch.D.G. Manolakis; "Tratamiento Digital de Señales", Tercera Edición, Madrid, Prentice - Hall, 1998.

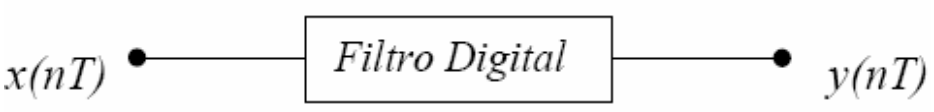
.

2.2.3.1 Filtros Digitales.

Un filtro digital puede ser representado mediante el siguiente diagrama de bloques:

$$y(nT) = Rx(nT)$$

Figura 4. Diagrama de bloques Filtro digital.



Fuente: J. Proakis, Ch.D.G. Manolakis; "Tratamiento Digital de Señales", Tercera Edición, Madrid, Prentice - Hall, 1998.

x(nT) es la secuencia de entrada -la excitación del filtro- e y(nT*)* es la respuesta del filtro ante la excitación x(nT).

El análisis de un filtro digital es el proceso de determinar la respuesta de un filtro ante una dada excitación. El *diseño* de un filtro digital es el proceso de sintetizar e implementar un filtro digital de tal manera que cumpla con las especificaciones prescriptas[3].

2.2.3.1.1 Características de los Filtros Digitales.

Los filtros digitales deben cumplir con las siguientes propiedades:

- Invariancia en el tiempo
- Causalidad
- Linealidad

[3] J. Proakis, Ch.D.G. Manolakis; "Tratamiento Digital de Señales", Tercera Edición, Madrid, Prentice - Hall, 1998.

- Invariancia en el tiempo.

Partiendo del reposo, y teniendo en cuenta que: $x(nT) = y(nT) = 0 \forall\ n < 0$

Un filtro digital es invariante en el tiempo si, para cualquier posible excitación, se cumple que:

$$Rx(nT - kT) = y(nT - kT)$$

Ejemplos:

a) $y(nT) = 2nTx(nT)$

$$\left.\begin{aligned} Rx(nT - kT) &= 2nTx(nT - kT) \\ y(nT - kT) &= 2(nT - kT)x(nT - kT) \end{aligned}\right\}$$ Son distintos ⇒ NO es Invariante en el Tiempo

b) $y(nT) = Rx(nT) = 12x(nT - T) + 11x(nT - 2T)$

$$\left.\begin{aligned} Rx(nT - kT) &= 12x(nT - kT - T) + 11x(nT - kT - 2T) \\ y(nT - kT) &= 12x(nT - kT - T) + 11x(nT - kT - 2T) \end{aligned}\right\}$$ son iguales ⇒ es Invariante en el Tiempo

- Causalidad.

Para que un filtro digital sea causal, su salida en un instante dado no puede depender de valores posteriores de la excitación.

Es decir: para un par de excitaciones tales que: $x1(nT) = x2(nT) \forall\ n \le k$ y $x1(nT) \neq x2(nT) \forall\ n > k$

Si el filtro es causal se debe cumplir que: $Rx1(nT) = Rx2(nT) \forall\ n \le k$

Ejemplos:

a) $y(nT) = Rx(nT) = 3x(nT - 2T) + 3x(nT + 2T)$

Para *n=k* tenemos: $\left.\begin{array}{l} Rx1(kT) = 3x1(kT-2T) + 3x1(kT+2T) = \\ = 3x1((k-2)T) + 3x1((k+2)T) \\ Rx2(kT) = 3x2((k-2)T) + 3x2((k+2)T) \end{array}\right\}$ son distintos ⇒

⇒ NO Causal

b) $y(nT) = Rx(nT) = 2x(nT-T) - 3x(nT-2T)$

Si $n \le k \Rightarrow (n-1) < k$ y: $n-2) < k$

$\therefore x1(nT-T) = x2(nT-T)$ y: $x1(nT-2T) = x2(nT-2T)$ $\forall\, n \le k$

$\Rightarrow Rx1(nT) = Rx2(nT)$ $\forall\, n \le k \Rightarrow$ El filtro es Causal

- <u>Linealidad.</u>

Un filtro digital es lineal si se cumple que:

$$\begin{cases} R\alpha x(nT) = \alpha Rx(nT) \\ R\left[x1(nT) + x2(nT)\right] = Rx1(nT) + Rx2(nT) \forall \alpha, x1(nT) y x2(nT) \end{cases}$$

O bien:

$$R\left[\alpha x1(nT) + \beta x2(nT)\right] = \alpha Rx1(nT) + \beta Rx2(nT) \forall \alpha, \beta, x1(nT) y x2(nT)$$

Ejemplos:

a) $y(nT) = Rx(nT) = 7x2(nT-T)$

$R\alpha x(nT) = 7\alpha\, 2x2(nT-T) \neq \alpha Rx(nT) = 7\alpha x2(nT-T) \Rightarrow$ No Lineal

b) $y(nT) = Rx(nT) = (nT)2x(nT+2T)$

$$R[\alpha x1(nT) + \beta x2(nT)] = (nT)2[\alpha x1(nT + 2T) + \beta x2(nT + 2T)] =$$
$$= \alpha(nT)2x1(nT + 2T) + \beta(nT)2x2(nT + 2T) =$$
$$= \alpha Rx1(nT) + \beta Rx2(nT) \Rightarrow \text{Es Lineal}$$

2.2.3.1.2 Caracterización de Filtros Digitales.

2.2.3.1.2.1 Filtros No Recursivos (FIR).

Los filtros digitales no recursivos también se denominan "de Respuesta Finita al Impulso", y también se los identifica con la sigla FIR (Finite Impulse Response).
La expresión general de un filtro FIR es la siguiente:

$$y(nT) = \sum_{i=-\infty}^{\infty} a_i x(nT - iT)$$

Considerando que el filtro es causal tenemos que: *a–1 = a–2 = ... = 0,* por lo tanto:

$$y(nT) = \sum_{i=0}^{\infty} a_i x(nT - iT)$$

Si además se considera que se parte del reposo: $x(nT) = 0 \forall\, n < 0$ y si sólo una cantidad finita de coeficientes es distinta de cero: $ai = 0 \forall\, i > N$ Se llega a la siguiente expresión del filtro no recursivo:

$$y(nT) = \sum_{i=0}^{N} a_i x(nT - iT)$$ Donde *N*: orden del filtro

Esta expresión indica que el valor de salida actual de un filtro FIR es función de la entrada actual y de las N entradas anteriores.

2.2.3.1.2.2 Filtros Recursivos (IIR).

Los filtros digitales recursivos también se denominan "de Respuesta Infinita al Impulso", y también se los identifica con la sigla IIR (Infinite Impulse Response). La respuesta de un filtro IIR es función de la excitación y también de las respuestas anteriores.

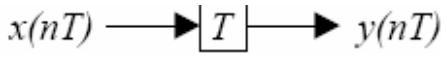
La siguiente es la expresión de un filtro digital recursivo, lineal, invariante en el tiempo y causal:

$$y(nT) = \sum_{i=0}^{N} a_i x(nT - iT) - \sum_{i=1}^{N} b_i y(nT - iT)$$

2.2.3.1.2.2 Redes de Filtrado Digital

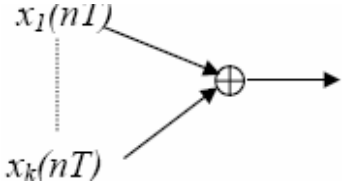
La estructura de un filtro digital puede representarse gráficamente mediante una red en la que se combinan los siguientes elementos básicos:

Figura 5. Estructuras de Filtros digitales (Retardo unitario, Sumador, Multiplicador).

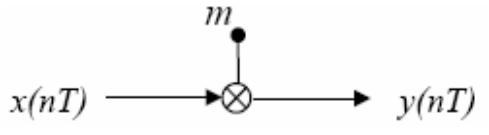
Retardo Unitario: 

$y(nT) = x(nT - T)$

Sumador:



$$y(nT) = \sum_{i=1}^{k} x_i(nT)$$

Multiplicador: 

$y(nT) = mx(nT)$

Fuente: http://laenciclopedialibre/Filtro digital.htm. 08/12/2005, 10:23

2.2.3.2 Análisis en el Dominio del Tiempo.

Para analizar el comportamiento temporal de un filtro digital se considera que a la entrada se le aplica una secuencia determinada. Para ello se utilizan una serie de funciones elementales, que generan cada una de ellas distintas secuencias. Estas funciones son las siguientes:

- Impulso unitario: $S(nT) = \begin{cases} 1, n = 0 \\ 0, n \neq 0 \end{cases}$
- Escalón unitario: $S(nT) = \begin{cases} 1, n \geq 0 \\ 0, n < 0 \end{cases}$
- Rampa unitaria: $r(nT) = \begin{cases} nT, n \geq 0 \\ 0, n < 0 \end{cases}$
- Exponencial: $e^{\alpha nT}$
- Sinusoide: $sen(wnT)$

La respuesta temporal de un filtro digital se puede determinar resolviendo la correspondiente ecuación de diferencias.

Ejemplo:

Hallar la respuesta temporal al impulso de filtro digital cuya ecuación de diferencias es:

$$y(nT) = x(nT) + e^{\alpha}\, y(nT - T)$$

Partiendo del reposo: $y(0) = 1 + e^{\alpha}\, y(-T) = 1$

$$y(T) = 0 + e^{\alpha}\, y(0) = e^{\alpha}$$

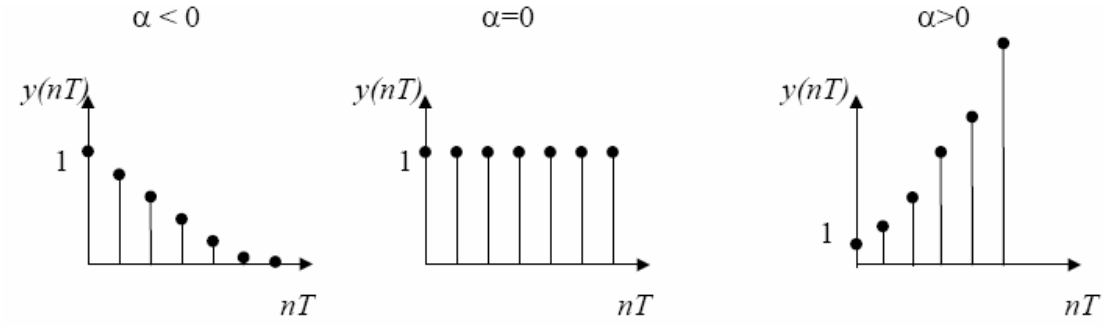
$$y(2T) = 0 + e^{\alpha}\, y(T) = e^{2\alpha}$$

.....

$$y(nT) = e^{n\alpha}$$

Las siguientes gráficas representan la respuesta temporal del filtro en función de α:

Figura 6. Respuesta temporal del filtro digital (Ejemplo).



Fuente: http://es.wikipedia.org/wiki/Imagen:Filt_elect_pend.PNG. 18/11/2005. 14:29.

2.2.3.2.1 Sumatoria de Convolución.

La respuesta de un filtro digital a una excitación arbitraria puede ser expresada en términos de la respuesta del filtro a una entrada impulsiva. Para ello se debe realizar la convolución entre la señal de entrada y la respuesta impulsiva.

Una señal de entrada $x(nT)$ puede ser expresada de la siguiente forma:

$$x(nT) = \sum_{k=-\infty}^{\infty} x_k(nT) \quad \text{donde:} \qquad x_k(nT) = \begin{cases} x(kT), si\ n = k \\ 0, si\ n \neq k \end{cases}$$

También se podría expresar de la siguiente manera:

$$x_k(nT) = x(kT)\ \delta(nT - kT) \qquad \text{Por lo tanto:} \quad x(nT) = \sum_{k=-\infty}^{\infty} x(kT)\ \delta(nT - kT)$$

Partiendo de esta última expresión, vamos a considerar un filtro digital lineal, invariante en el tiempo, tal que su respuesta impulsiva sea:

$$h(nT) = R\delta(nT)$$

y su respuesta a una entrada arbitraria $x(nT)$ sea: $y(nT) = Rx(nT)$

$$\Rightarrow y(nT) = R\sum_{k=-\infty}^{\infty} x(kT)\ \delta(nT - kT) = \sum_{k=-\infty}^{\infty} x(kT)\ R\,\delta(nT - kT) = \sum_{k=-\infty}^{\infty} x(kT)\ h(nT - kT)$$

Si el filtro es causal $\Rightarrow h(nT) = 0\ \forall\ n < 0$, por lo tanto:

$$y(nT) = \sum_{k=-\infty}^{n} x(kT)\ h(nT - kT) = \sum_{k=0}^{\infty} h(kT)\ x(nT - kT)$$

Si además se considera que: $x(nT) = 0\ \forall\ n < 0$, la respuesta del filtro digital queda expresada mediante la siguiente sumatoria de convolución:

$$y(nT) = \sum_{k=0}^{n} x(kT)\ h(nT - kT) = \sum_{k=0}^{n} h(kT)\ x(nT - kT)$$

2.2.3.2.2 Estabilidad.

Un filtro digital es estable si para cualquier excitación acotada se obtiene una salida acotada, es decir:

$$|y(nT)| < \infty \quad \forall\ n \qquad \text{Para} \qquad |x(nT)| < \infty \quad \forall\ n$$

Vamos a tratar de encontrar la forma de determinar si un filtro es estable:

$|y(nT)| \leq \sum_{k=-\infty}^{\infty} |h(kT)| \cdot |x(nT-kT)|$ → El módulo de una suma es menor o igual que la suma de los módulos; y además el módulo de un producto es menor o igual que el producto de los módulos.

Si $|x(nT)| \leq M < \infty \qquad \forall n \Rightarrow |y(nT) \leq M| \sum_{k=-\infty}^{\infty} |h(kT)|$

Por lo tanto, si: $\sum_{k=-\infty}^{\infty} |x(kT)| < \infty \Rightarrow |y(nT)| < \infty \qquad \forall n$

Condición de estabilidad.

2.2.3.3 La Transformada Z.

La transformada Z se utiliza para el análisis de filtros digitales lineales e invariantes en el tiempo. Transforma ecuaciones de diferencias en expresiones algebraicas, lo que simplifica los cálculos[4].

- Definición:

$F(z) = \sum_{n=-\infty}^{\infty} f(nT)\ z^{-n}$

Para cualquier z, tal que F(z) converge,

Z: variable compleja

$z = x + jy$

- Notación:

$$F(z) = Zf(nT)$$

[4] J. Proakis, Ch.D.G. Manolakis; "Tratamiento Digital de Señales", Tercera Edición, Madrid, Prentice - Hall, 1998.

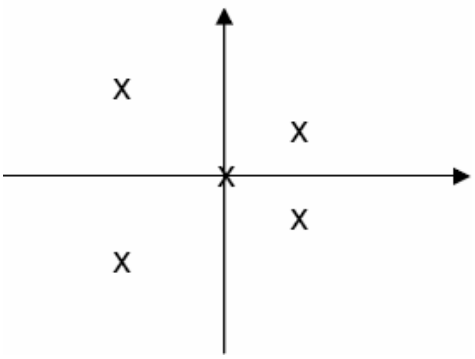
2.2.3.3.1 Propiedades de la Transformada Z.

a) Región de convergencia

- El tipo de funciones que se utiliza en filtros digitales son funciones meromórficas (sus únicas singularidades son polos).
- Hay más de una serie que converge.

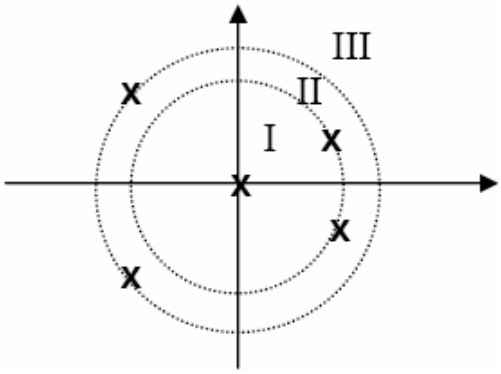
Por ejemplo, dado el siguiente diagrama de polos y ceros:

Figura 7. Región de convergencia.



Se puede ver que hay tres regiones de convergencia: I, II y III

Figura 8. Regiones de convergencia



Fuente: J. Proakis, Ch.D.G. Manolakis; "Tratamiento Digital de Señales", Tercera Edición, Madrid, Prentice - Hall, 1998.

Se asume que la serie que interesa es una que converge en la región III, es decir:

$$R1 \le |z| \le R2$$

Donde: $\begin{matrix} |z| = R1 \\ R2 \to \infty \end{matrix}$ es el círculo que pasa por los polos más alejados del origen

b) Linealidad

Si a y b son constantes, y *Z f(nT) = F(z)* y *Z g(nT) = G(z)*, entonces:

$$Z\left[af(nT) + bg(nT)\right] = aF(z) + bG(z)$$

c) Translación

$$Zf(nT + mT) = z^m F(z)$$

d) Cambio de escala complejo

$$Z\left[w^{-n} f(nT)\right] = F(wz)$$ Donde: w es un número complejo.

e) Diferenciación compleja

$$Z\left[nT\ f(nT)\right] = -Tz\ \frac{dF(z)}{dz}$$

f) Convolución real

$$Z\left(\sum_{k=-\infty}^{\infty} f(kT)g(nT - kT)\right) = F(z)G(z)$$

O bien

$$Z\left(\sum_{k=-\infty}^{\infty} f(nT - kT)g(nT)\right) = F(z)G(z)$$

2.2.3.3.2 Transformada Z unilateral.

De manera análoga a la transformada de Laplace, la transformada Z unilateral es definida así:

$$F(z) = \sum_{n=0}^{\infty} f(nT)\ z^{-n} = Z_I f(nT)$$

$Z \neq Z_I$ Solo si $f(nT) \neq 0$ para $n < 0$

Como se trabaja con funciones que son cero para $n < 0$, no es necesario hacer la distinción.

2.2.3.3.3 La Transformada Z inversa.

Si $F(z)$ converge en algún anillo abierto, tal como se ha visto anteriormente al definir la transformada Z, entonces es posible obtener $f(nT)$ de la siguiente manera:

$$f(nT) = \frac{1}{2\pi j} \oint_T F(z) z^{n-1} dz = Z^{-1} F(z)$$

$$f(z)z^{n-1} = F_0(z) = \frac{N(z)}{\sum_{i=1}^{k} (z - p_i)^{mi}}$$ donde k y m son enteros positivos.

Aplicando el teorema del residuo se obtiene: $f(nT) = \sum res_{z=pi} \left[F_0(z)\right]$

La notación es la siguiente: $f(nT) = Z^{-1} F(z)$

2.2.3.3.4 Aplicación de la Transformada Z.

Usando la transformada Z, un filtro digital puede ser caracterizado mediante una función de transferencia discreta en el tiempo, que juega el mismo rol que la función de transferencia contínua en el tiempo para un filtro analógico[5].

[5] J. Proakis, Ch.D.G. Manolakis; "Tratamiento Digital de Señales", Tercera Edición, Madrid, Prentice - Hall, 1998.

Considerando un filtro digital lineal e invariante en el tiempo, tal como el siguiente:

$$y(nT) = \sum_{k=-\infty}^{\infty} x(kT)h(nT-kT) \Rightarrow Zy(nT) = Zh(nT)\cdot Zx(nT) \Rightarrow Y(z) = H(z)\cdot X(z)$$

2.2.3.3.4.1 Obtención de Frecuencia H(z).

Para un filtro digital causal y recursivo tenemos:

$$y(nT) = \sum_{i=0}^{N} a_i x(nT-iT) - \sum_{i=1}^{N} b_i y(nT-iT)$$

Por lo tanto:

$$Zy(nT) = \sum_{i=0}^{N} a_i z^{-i} Zx(nT) - \sum_{i=1}^{N} b_i z^{-i} Zy(nT)$$

$$\Rightarrow y(z) = X(z)\sum_{i=0}^{N} a_i z^{-i} - Y(z)\sum_{i=1}^{N} b_i z^{-i}$$

$$\Rightarrow y(z)\left(1+\sum_{i=0}^{N} b_i z^{-i}\right) = X(z)\sum_{i=1}^{N} a_i z^{-i} \quad \text{Como } H(z) = \frac{Y(z)}{X(z)} \Rightarrow$$

$$\Rightarrow H(z) = Y(z) = \frac{\sum_{i=0}^{N} a_i z^{-i}}{1+\sum_{i=1}^{N} b_i z^{-i}} = \ldots factorizando \ldots = \frac{H_0 \prod_{i=0}^{N} (z-z_i)}{\prod_{i=1}^{N} (z-p_i)}$$

Donde z_i son los cero y p_i son los polos.

2.2.3.3.4.2 Análisis en el dominio del tiempo

Si se desea obtener la respuesta temporal de un filtro digital caracterizado por una función transferencia discreta H(z), ante una cierta excitación X(z), se hace lo siguiente:

$$y(nT) = Z^{-1}\left[H(z)X(z)\right]$$

2.2.3.3.4.3 Análisis en el dominio de la frecuencia.

La respuesta estacionaria de un filtro analógico cuya función de transferencia es H(s) , se calcula de la siguiente manera:

$$\lim_{t\to\infty} y(t) = \lim_{t\to\infty} R\ u(t)sen(wt) = M(w)sen\left[wT + \theta(t)\right]$$

Donde:

$M(w) = \left|H(jw)\right|$: Ganancia.

$\theta(w) = \arg H(jw)$: Desplazamiento de fase.

Si consideramos un filtro digital de orden N, la respuesta a una excitación senoidal es:

$$y(nT) = Z^{-1}\left[H(z)X(z)\right]$$

donde:

$$X(z) = Z\left[u(nT)sen(wnT)\right] = \frac{zsen(wnT)}{(z - e^{jwT})(z - e^{-jwT})}$$

Es decir:

$$y(nT) = \oint_T H(z)X(z)z^{n-1}dz = \sum res\left[H(z)X(z)z^{n-1}\right]$$

Para n > 0 tenemos:

$$y(nT) = \sum_{i=1}^{N} \sum_{z=pi} res\left[H(z)\right]x(pi)pi^{n-1} + \frac{1}{2j}\left[H(e^{jwT})e^{jwnT} - H(e^{-jwT})e^{-jwT}\right]$$

Como $|pi| < 1$, este término tiende a cero para $n \to \infty$

Por lo tanto:

$$y(nT) = \frac{1}{2j}\left[H(e^{jwT})e^{jwnT} - H(e^{-jwT})e^{-jwT}\right]$$

Como: $H(e^{jwT}) = H^*(e^{-jwT})$ y si: $H(e^{jwT}) = M(w)e^{j\theta(w)}$

Donde: $M(w) = \left|He^{j\theta(w)}\right|$ y $\theta(w) = \arg H(e^{jwT})$

$$\Rightarrow y(nT) = M(w)sen\left[wnT + \theta(w)\right]$$

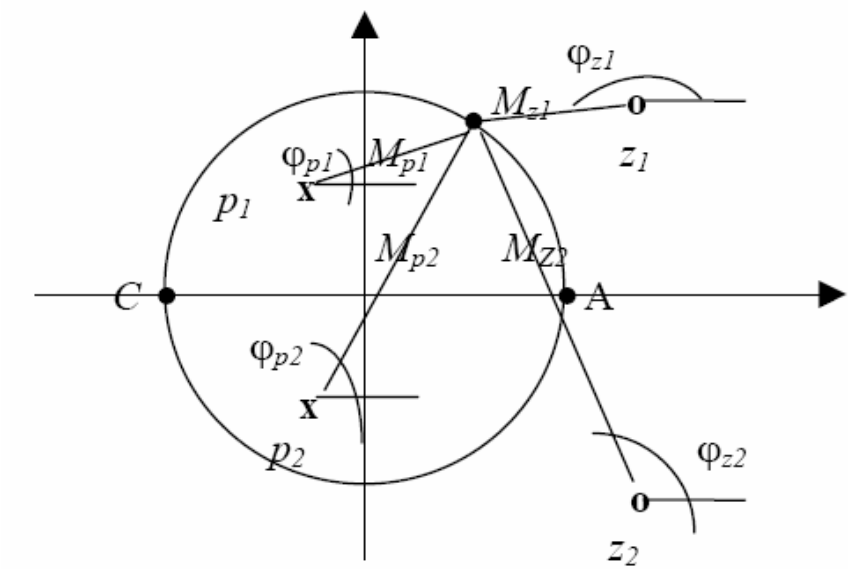
$$H(e^{jwT}) = M(w)e^{j\theta(w)} = \frac{H_0 \prod_{i=1}^{N}(e^{jwT} - z_i)}{\prod_{i=1}^{N}(e^{jwT} - p_i)} \qquad Donde: \begin{cases} e^{jwT} - z_i = M_{zi}e^{j\varphi ziT} \\ e^{jwT} - p_i = M_{pi}e^{j\varphi piT} \end{cases}$$

Así obtenemos:

$$M(w) = \frac{H_0 \prod_{i=1}^{N} M_{zi}}{\prod_{i=1}^{N} Mp_i} \qquad \theta(w) = \sum_{i=1}^{N} \varphi_{zi} - \sum_{i=1}^{N} \varphi_{pi}$$

Por lo tanto, *M(ω)* y θ(ω) pueden obtenerse dibujando los fasores en el plano, y midiendo sus magnitudes y sus ángulos. Para un filtro recursivo de segundo orden tenemos:

Figura 9. Plano de fasores para filtro recursivo de 2º orden.



El punto $A \Rightarrow w = \infty$

$C \Rightarrow w = \pi / T(frecuencia\ de\ Nyquist)$

Fuente: J. Proakis, Ch.D.G. Manolakis; "Tratamiento Digital de Señales", Tercera Edición, Madrid, Prentice - Hall, 1998.

Una vuelta completa alrededor del origen corresponde a un incremento de frecuencia de $ws = \frac{2\pi}{T}$, donde $ws = frecuencia\ de\ muestreo.$

Como:

$H(e^{jwT}) = H(e^{j(w+kws)T}) = H(e^{jwT}) \Rightarrow H(e^{jwT})$ es periódica con período ω*S*

A los fines prácticos se trabaja con un período –ω*S* , ω*S* llamado *banda base*.

2.2.3.4 Realizabilidad.

A fin de poder ser implementada mediante un filtro recursivo, una función transferencia debe satisfacer las siguientes condiciones:

1) Debe ser una función real de *z* con coeficientes reales.

2) Sus polos deben estar dentro del círculo unidad en el plano Z.

3) El grado del polinomio del numerador debe ser de grado menor o igual al grado del polinomio del denominador.

2.2.3.5 Formas de realización de un filtro digital.

2.2.3.5.1 Realización Directa.

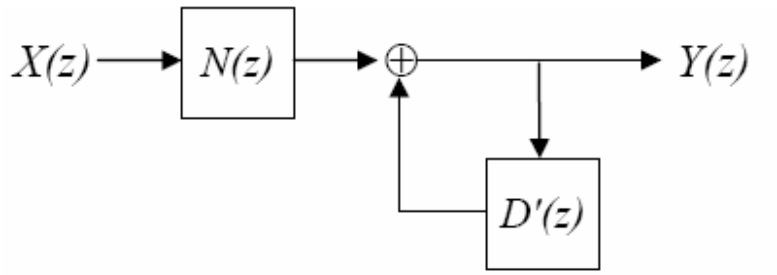
$$\frac{Y(z)}{X(z)} = H(z) = \frac{N(z)}{D(z)} = \frac{N(z)}{1 + D'(z)} \qquad Donde: \quad N(z) = \sum_{i=0}^{N} a^i z^{-i}$$

$$D(z) = \sum_{i=1}^{N} b^i z^{-i}$$

$$\Rightarrow Y(z)\left[1 + D'(z)\right] = X(z)N(z)$$

$$Y(z) = X(z)N(z) - Y(z)D'(z)$$

Figura 10. Realización directa de un filtro digital.



Fuente: J. Proakis, Ch.D.G. Manolakis; "Tratamiento Digital de Señales", Tercera Edición, Madrid, Prentice - Hall, 1998.

2.2.3.5.2 Realización Directa Canónica.

Minimiza la cantidad de retardos: la cantidad de retardos es igual al orden del filtro.

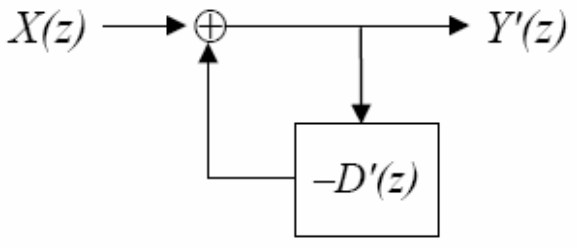
$$\frac{Y(z)}{X(z)} = H(z) = \frac{N(z)}{D(z)} = \frac{N(z)}{1 + D'(z)} \qquad Se\ \ puede\ \ \exp resar\ como: Y(z) = N(z)Y'(z)$$

Donde: $$Y'(z) = \frac{X(z)}{1 + D'(z)} \Rightarrow Y'(z) = \left[1 + D'(z)\right] = X(z)$$

$$\Rightarrow Y'(z) = X(z) - Y'(z)D'(z)$$

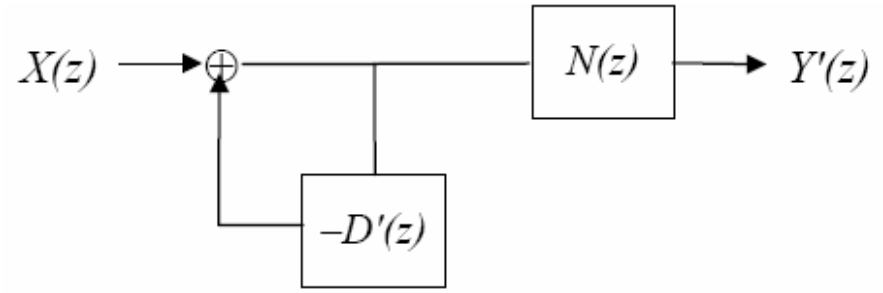
Figura 11. a), b), Realización directa canónica de un filtro digital.

a)



$$\therefore\ \ Y(z) = N(z)Y'(z)$$

b)



Fuente: J. Proakis, Ch.D.G. Manolakis; "Tratamiento Digital de Señales", Tercera Edición, Madrid, Prentice - Hall, 1998.

2.2.3.6 Aproximaciones de filtros analógicos.

Un filtro digital recursivo se puede aproximar usando alguna de las siguientes aproximaciones de filtros analógicos[6]:

- Butterworth
- Tschebyscheff
- Elípticos
- Bessel

2.2.3.6.1 Conceptos Básicos.

La función transferencia de un filtro analógico se puede expresar como:

$$\frac{V_0(s)}{V_i(s)} = H(s) = \frac{N(s)}{D(s)}$$

Donde: N(s) y D(s) son polinomios en función de $s = \sigma + j\omega$

[6] Antonio Pertence Junior; "Amplificadores Operacionales y Filtros Activos", McGraw-Hill, 1991.

- Atenuación en dB: $A(w) = 20\log\left|\frac{V_i(jw)}{V_0(jw)}\right| = 20\log\left|\frac{1}{H(jw)}\right| = 10\log L(w^2)$

Donde: $L(w^2) = \frac{1}{H(jw)\quad H(-jw)}$

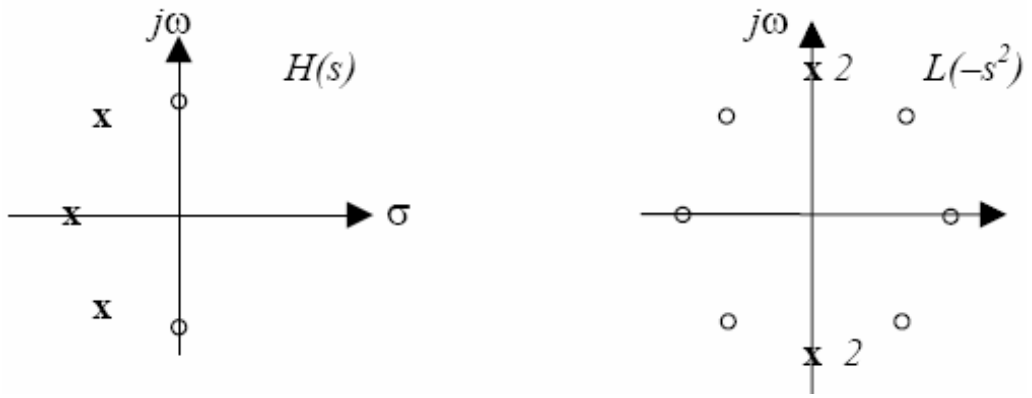
- Desplazamiento de fase: $\theta(jw) = \arg\ \ H(jw)$
- Retardo de grupo: $\tau = \frac{-d\theta(jw)}{dw}$

De aquí surgen las curvas características de Atenuación, Fase y Retardo en función de ω.

Con $w = \frac{s}{j}$ hacemos: $L(-s^2) = \frac{D(s)\ \ D(-s)}{N(s)\ \ N(-s)}$ Función de atenuación.

Un par de diagramas típicos de polos y ceros de $H(s)$ y $L(-s^2)$ son como los siguientes:

Figura 12. Diagrama típico de polos y ceros.



2.2.3.6.2 Aproximación de Butterworth.

La aproximación más simple para un pasábajos es la de Butterworth.

Se asume que: $L(w^2) = B_1 w^2 + B_2 w^4 + ... + B_n w^{2n}$

Tal que: $\lim_{w^2 \to \infty} L(w^2) = 1 \quad \Rightarrow \quad L(0) = 1 \quad \therefore \quad B_0 = 1$

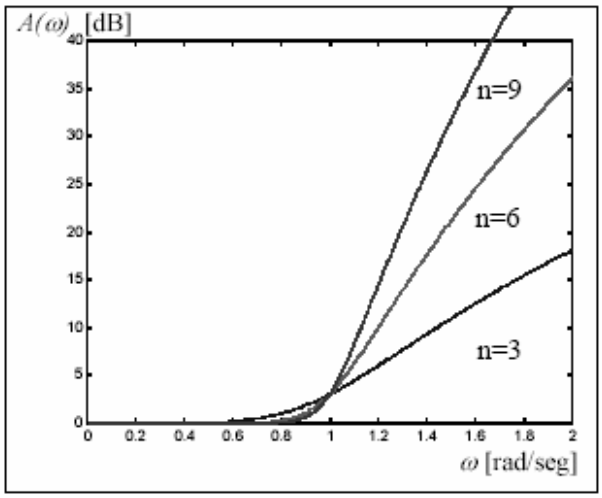
Se puede llegar a que $B_1 = B_2 = ... = B_n \quad \therefore \quad L(w^2) = 1 + B_n w^{2n}$

Como para un pasábajos normalizado: $A(w) = -3dB$ a $w = 1[rad/seg]$

$$\Rightarrow B_n = 1 \quad \therefore \quad L(w^2) = 1 + w^{2n}$$

La atenuación de un pasábajos Butterworth normalizado es: $A(w) = 10\log(1 + w^{2n})$, y en la siguiente figura se presentan las curvas de atenuación en función de *n* (orden del filtro).

Figura 13. Curvas de atenuación.



Fuente: http://laenciclopedialibre/Filtro digital.htm. 08/12/2005, 10:23

2.2.3.6.2.1 Función Transferencia Normalizada.

Con $w = \frac{s}{j}$ tenemos: $L(-s^2) = 1 + (-s^2)^n = \prod_{k=1}^{2n} (s - s_k)$

Donde: $\begin{cases} s_k = e^{j(2k-1)\pi/2n} & para\ n\ par \\ s_k = e^{j(k-1)\pi/n} & para\ n\ impar \end{cases}$

Como $|s_k| = 1$, los ceros de $L(-s^2)$ están en el círculo $|s| = 1$

La función transferencia normalizada puede ser expresada como:

$H_N(s) = \frac{1}{\prod_{i=1}^{n} (s - p_i)}$ Donde: p_i son los ceros de $L(-s^2)$ en el semiplano izquierdo.

2.2.3.6.3 Otras aproximaciones de filtros analógicos.

Además de la aproximación de Butterworth, existen otras aproximaciones para filtros analógicos, de las cuales las siguientes son las más conocidas[7].

- Aproximación de Tschebyscheff

– La atenuación en la banda de paso oscila entre 0 y un máximo permitido
– La atenuación en la banda de rechazo aumenta monotónicamente.

[7] J. Proakis, Ch.D.G. Manolakis; "Tratamiento Digital de Señales", Tercera Edición, Madrid, Prentice - Hall, 1998.

- Aproximación Elíptica

– La atenuación en la banda de paso oscila entre 0 y *Ap*.
– La atenuación en la banda de rechazo oscila entre y *Aa*.

- Aproximación de Bessel

– Tiene una respuesta lineal en fase (a diferencia de las tres anteriores).

2.2.3.6.4 Transformaciones.

Partiendo de un filtro pasábajos normalizado, se pueden obtener filtros pasábajos, pasáaltos, pasábandas y rechazabandas desnormalizados. Para tal fin se utilizan transformaciones de la forma:

$$s = f(\bar{s})$$

PasáBajos $\rightarrow$ Pasa Bajos: $s = \lambda \bar{s}$ Pasa Bajos $\rightarrow$ Pasa Altos: $s = \frac{\lambda}{\bar{s}}$

PasáBajos $\rightarrow$ Pasa Banda: $s = \frac{1}{B}(s + \frac{w_0^2}{\bar{s}})$

PasáBajos $\rightarrow$ Rechaza Banda: $s = \frac{B\bar{s}}{s^2 + w_0^2}$

2.2.3.6.5 Realización de un filtro digital.

2.2.3.6.5.1 *Realización Directa*.

$$\frac{Y(z)}{X(z)} = H(z) = \frac{N(z)}{D(z)} = \frac{N(z)}{1 + D'(z)}$$

Donde:

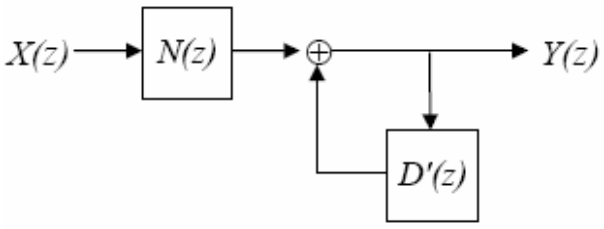
$$N(z) = \sum_{i=0}^{N} a^i z^{-i}$$

$$D'(z) = \sum_{i=1}^{N} b^i z^{-i}$$

$$\Rightarrow Y(z)\left[1 + D'(z)\right] = X(z)N(z)$$

$$Y(z) = X(z)N(z) - Y(z)D'(z)$$

Figura 14. Filtro, Realización directa



Ejemplo:

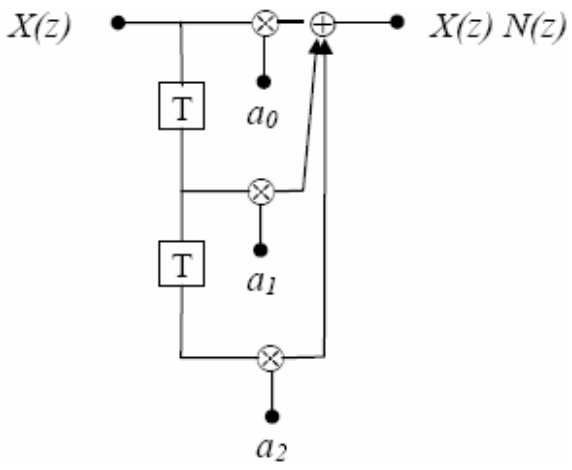
$$N(z) = a_0 + a_1 z^{-1} + a_2 z^{-2}$$

$$D'(z) = b_1 z^{-1} + b_2 z^{-2}$$

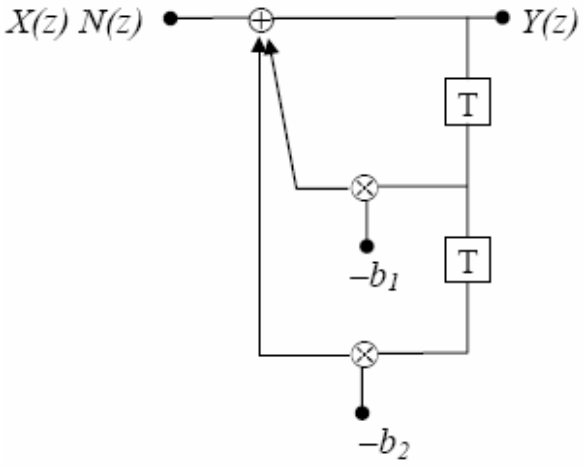
Figura 15. a)Filtro realización directa (Ejemplo), b)Filtro completo.

a)

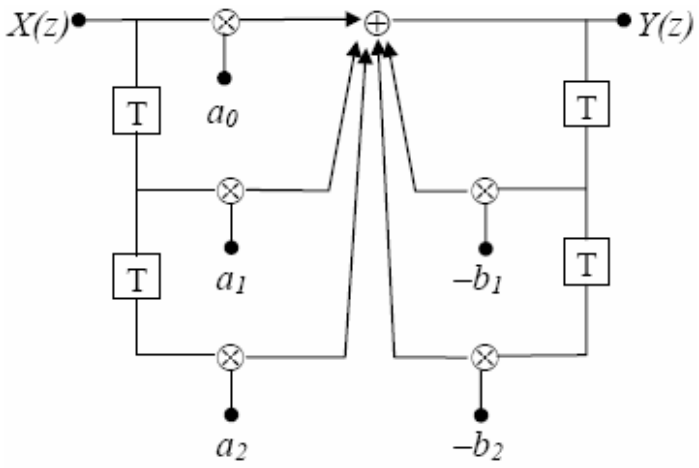
$N(z) \Rightarrow$



$D'(z) \Rightarrow$



b) Filtro completo:



Fuente: J. Proakis, Ch.D.G. Manolakis; "Tratamiento Digital de Señales", Tercera Edición, Madrid, Prentice - Hall, 1998.

2.2.3.6.5.1 Realización Directa Canónica.

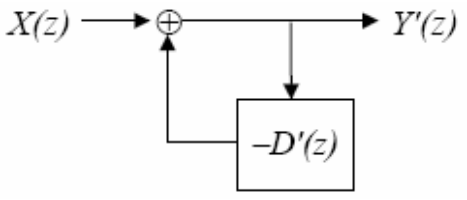
Minimiza la cantidad de retardos: la cantidad de retardos es igual al orden del filtro.

$\frac{Y(z)}{X(z)} = H(z) = \frac{N(z)}{D(z)} = \frac{N(z)}{1 + D'(z)}$ Se puede expresar como: $Y(z) = N(z)Y'(z)$

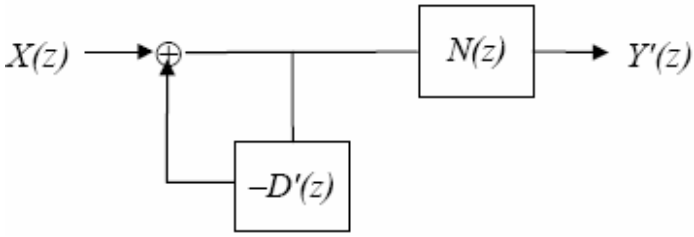
Donde: $Y(z) = \frac{X(z)}{1 + D'(z)} \Rightarrow Y'(z) = \left[1 + D'(z)\right] = X(z)$

$$\Rightarrow Y'(z) = X(z) - Y'(z)D'(z)$$

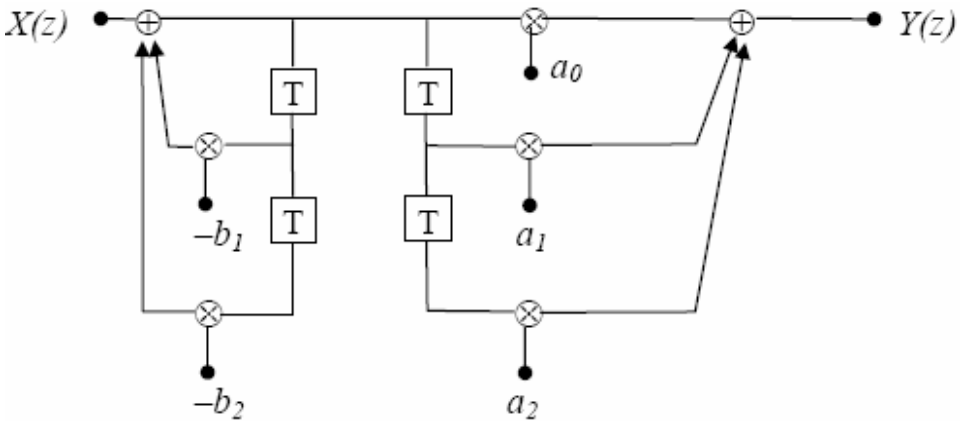
Figura 16. Filtro, Realización directa canónica.



$$\therefore Y(z) = N(z)Y'(z) \rightarrow$$



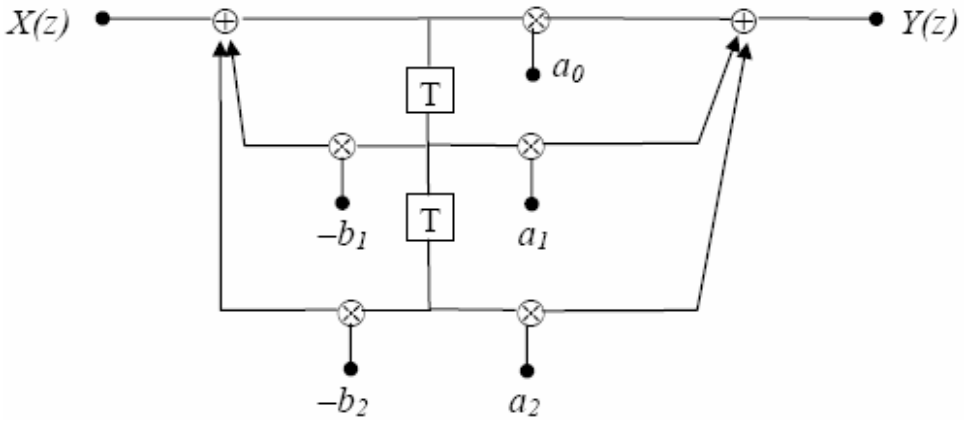
Volviendo al ejemplo anterior:



Fuente: J. Proakis, Ch.D.G. Manolakis; “Tratamiento Digital de Señales”, Tercera Edición, Madrid, Prentice - Hall, 1998.

Como lo que está almacenado en cada par de retardos son los mismos valores, el filtro puede quedar finalmente así:

Figura 17. Filtro, solución canónica.



2.2.4 Programación Dinámica.

La estrategia algorítmica conocida como programación dinámica es una técnica de resolución que encuentra aplicación en numerosos problemas de optimización, pero que no se limita a estos.

El tipo de problemas de optimización abordables por programación dinámica es, principalmente, aquél en el que las soluciones factibles se pueden descomponer en una secuencia de elementos (caminos en un grafo, series de decisiones, etc.) y cuya función objetivo satisface cierta condición de separabilidad y monotonía. No obstante, la técnica de programación dinámica también encuentra aplicación a la búsqueda de otras estructuras óptimas (árboles óptimos).

La programación dinámica guarda cierta relación con divide y vencerás. Divide y vencerás basa su eficiencia en la división de un problema en dos o más subproblemas de talla similar cuyas soluciones se combinan. La división de un problema en subproblemas de talla diferente conduce de forma natural a un planteamiento recursivo. Este planteamiento resulta, por regla general, en un

algoritmo eficiente cuando la recursión da lugar a instancias del mismo problema pero de talla similar y, además, no hay llamadas repetidas en el árbol de llamadas recursivas. El método de programación dinámica se aplica, precisamente, en problemas que adolecen de estos inconvenientes. Es decir, la
programación dinámica permite abordar problemas en los que la división recursiva de un problema en subproblemas resulta desfavorable desde la óptica de divide y vencerás. Para estos, la programación dinámica ofrece un repertorio de técnicas que permiten reducir la complejidad computacional propia de una resolución puramente recursiva[8].

Lo fundamental consiste en evitar la repetición de cálculos en las llamadas recursivas mediante el almacenamiento de los resultados que ya han sido calculados para su posterior reutilización. Esta técnica se conoce como (memorización). Por otra parte, una transformación recursivo-iterativa permite, en ocasiones, reducciones adicionales de complejidad espacial: no sólo ahorra la memoria que ocupa la pila de llamadas a función, sino que permite reducir el propio tamaño de la tabla de soluciones precalculadas. Se desarrollan los conceptos fundamentales al hilo de la resolución de un par de problemas: el calculó del camino mas probable en un grafo acíclico muy estructurado y el problema del desglose optimó de una cantidad de dinero. El algoritmo presentara un coste computacional diferente, con lo que ilustra la importancia del modelado en el diseñó de la solución.

2.2.4.1 Esquema de Programación Dinámica.

Los ejemplos desarrollados son ilustrativos del proceso que se sigue frecuentemente al solucionar un problema de programación dinámica:

[8] José C. Segura, M. Carmen Bentez. Algoritmos robustos de parametrización de la voz para reconocimiento automático del habla.pdf.

1. Modelado. Formulación del problema en términos de optimización sobre un espacio de soluciones factibles que son secuencias de elementos de cierto conjunto.

2. Ecuación recursiva. Planteamiento de una ecuación recursiva que soluciona el problema de obtención del valor optimó de la función objetivo.

3. Memorización. Detección de problemas de repetición de cálculos y resolución mediante memorización de resultados en una tabla indexada por argumentos de la función recursiva.

4. Transformación recursivo-iterativa. Obtención de un grafo (acíclico) de dependencias entre resultados de las llamadas recursivas. Diseño de un algoritmo iterativo a partir de un recorrido de los vértices del grafo en orden topológico y el almacenamiento de resultados asociados a los vértices en una tabla. Si el grafo de dependencias está estructurado, puede que no sea necesario mantener en memoria la tabla completa de resultados, con el consiguiente ahorro de espacio.

5. Calculo de la solución factible óptima. La solución factible óptima está compuesta por una serie de decisiones simples. Estas decisiones son los vértices o las aristas de un camino (u otra estructura) en el grafo de dependencias. La técnica de punteros hacia atrás o un proceso sobre el almacén de resultados permite recuperar los elementos que constituyen la solución óptima.

2.2.4.1.1 Modelado.

Conjunto de estados

El esquema de programación dinámica es de aplicación en problemas de optimización cuyo conjunto de soluciones factibles está formado por secuencias

de elementos $(d_1, d_2, ..., d_n)$. Sea D el conjunto de componentes con el que se forman las secuencias. Es decir, $X \subseteq D^*$, donde D^* es la clausura de D bajo una operación de concatenación. Por regla general, D es el conjunto de definiciones de instancias del problema que estamos resolviendo y recibe el nombre de conjunto de estados.

Función objetivo

La función objetivo f proporciona un valor real para cada secuencia. Buscamos

a) la secuencia (o una de las secuencias) que proporciona el valor optimó de la función objetivo:

$$(\hat{d}_1, \hat{d}_2, ..., \hat{d}_n) = \arg \underset{(d_1, d_2, ..., d_n) \in X}{opt} f((d_1, d_2, ..., d_n)),$$

b) y/o dicho valor:

$$\underset{(d_1, d_2, ..., d_n) \in X}{opt} f((d_1, d_2, ..., d_n)).$$

Aunque el propósito es resolver el primer problema, conviene empezar diseñando una ecuación recursiva para el segundo de los problemas. Resolver esa ecuación recursiva proporciona, mediante técnicas estándar, la solución del primero.

Función de descomposición de estados

Las secuencias válidas se pueden definir a partir de una (función de descomposición)

$\delta : D \to \rho(D)$:

$$X = \left\{ (d_1, d_2, ..., d_n) \middle| d_{i-1} \in \delta(d_i), 1 \prec i \leq n \right\}$$

La ecuación recursiva se plantea el calculó del valor óptimo considerando que la solución óptima está formada por un prefijo $(\hat{d}_1, \hat{d}_2, ..., \hat{d}_{n-1})$ y un ultimó elemento $\hat{d}_n$. Este ultimó elemento puede tomar diferentes valores y particionar el conjunto de prefijos validos.

Estados finales e iniciales

Sea f el conjunto de valores que puede adoptar d_n, el ultimó elemento de una secuencia de X. Si denotamos con $X(a)$, para cualquier elemento $a \in D$, el conjunto,

$$X(a) = \left\{ (d_1, d_2, ..., d_n) \middle| d_n = a;\ d_{i-1} \in \delta(d_i), 1 \prec i \leq n \right\},$$

Tenemos

$$X = \bigcup_{a \in f} X(a)$$

El conjunto f es el **conjunto de estados finales.**

Si ahora se denota con τ al conjunto de valores posibles para d_1 en cualquier secuencia factible, se puede plantear una ecuación recursiva:

$$X(a) = \begin{cases} \{(a)\}, & si\ a \in \tau; \\ \left\{ (d_1, d_2, ..., d_n) \middle| d_n = a;\ (d_1, d_2, ..., d_{n-1}) \in X(a'), a' \in \delta(a) \right\}, & si\ a \in \tau. \end{cases}$$

El conjunto τ es el **conjunto de estados iniciales** y determina los casos base en la definición recursiva del conjunto X.

Subestructura óptima: el principio de optimalidad

Hemos de tener en cuenta que no toda función objetivo es susceptible de conducir a una ecuación recursiva, así que hemos de tener claro bajo qué condiciones es posible abordar el problema con programación dinámica.

Una clave es la observación de la denominada subestructura óptima o principio de optimalidad. Este principio puede formularse, en principio, así: (en la solución óptima del problema, $(\hat{d}_1, \hat{d}_2, ..., \hat{d}_n)$, las secuencias de la forma $(\hat{d}_1, \hat{d}_2, ..., \hat{d}_i)$ para $i \lhd n$ son soluciones óptimas de problemas de la misma naturaleza pero diferente talla).

En el siguiente apartado se enuncia bajo qué condiciones se observa la existencia de subestructura óptima.

2.2.4.1.2 Ecuación Recursiva.

Separabilidad de la función objetivo

La función objetivo, f debe ser separable: si $(d_1, d_2, ..., d_n)$ es una solución factible, la función objetivo debe poder expresarse como

$$f((d_1, d_2, ..., d_n)) = \begin{cases} h(d_1), & si\ \ n = 1; \\ f((d_1, d_2, ..., d_{n-1})) \oplus g(d_{n-1}, d_n), & si\ \ n \rhd 1; \end{cases}$$

Donde g y h son funciones auxiliares, $g : D \times D \to R$, y $\oplus$ es un operador $R \times R \to R$ al que se denomina **operador de combinación de resultados**.

Monotonía del operador de combinación de resultados

Para seguir con la derivación de la ecuación recursiva es preciso que el operador de combinación de resultados satisfaga una determinada condición de monotonía.

Para expresarla, necesitamos distinguir problemas de minimización de problemas de maximización:

- Si el problema es de minimización, el operador $\oplus$ debe ser monótono no decreciente por la izquierda:

$$a \le a' \Rightarrow a \oplus b \le a' \oplus b,$$

Para todo $a, a', b \in R$. Con ello tenemos

$$\left(\min_{a \in R} a \right) \oplus b = \min_{a \in R} (a \oplus b),$$

- Si el problema es de maximización, el operador $\oplus$ debe ser monótono no creciente por la izquierda:

$$a \ge a' \Rightarrow a \oplus b \ge a' \oplus b$$

Para todo $a, a', b \in R$. Con ello tenemos

$$\left(\max_{a \in R} a \right) \oplus b = \max_{a \in R} (a \oplus b)$$

Podemos volver al planteamiento general y afirmar que si el operador de combinación de resultados observa la condición de monotonía apropiada, entonces:

$$\underset{d_{n-1} \in \delta(a)}{opt} \left(\underset{(d_1, d_2, ..., d_{n-1}) \in X(d_{n-1})}{opt} \left(f((d_1, d_2, ..., d_{n-1})) \oplus g(d_{n-1}, d_n) \right) \right) =$$

$$\underset{d_{n-1}\in\delta(a)}{opt}\left(\left(\underset{(d_1,d_2,...,d_{n-1})\in X(d_{n-1})}{opt}\left(f((d_1,d_2,...,d_{n-1}))\right)\right)\oplus g(d_{n-1},d_n)\right).$$

Se nota que con $F(a)$ a $opt_{(d_1,d_2,...,d_{n-1})\in X(d_{n-1})} f((d_1,d_2,...,d_{n-1}))$. Tenemos

$$F(a)=\begin{cases} h(a), & si\ \ a\in\tau; \\ \underset{b\in\delta(a)}{opt}\left(F(b)\oplus g(b,a)\right), & si\ \ a\notin\tau; \end{cases}$$

$$f((d_1,d_2,...,d_n))=\underset{a\in f}{opt}\,F(a).$$

Esquema recursivo

El esquema algorítmico de resolución de la ecuación recursiva se muestra en la figura 18. Es una simple trascripción de las dos últimas ecuaciones.

Figura 18. Esquema recursivo de programación dinámica.

```
1  scheme RecursiveDynamicProgramming
2    auxiliary functions
3      δ: D → P(D)
4      g: D × D → ℝ
5      h: D → ℝ
6      function F(a)
7        if a ∈ I:
8          return h(a)
9        else:
10         return opt_{b∈δ(a)} (F(b) ⊕ g(b, a))
11   return opt_{a∈F} F(a)
```

2.2.4.1.3 Memorización.

Una resolución directa de la ecuación de recurrencia suele plantear problemas serios de ineficiencia debidos a la repetición de llamadas recursivas a la función F con los mismos argumentos. La técnica de memorización evita la repetición de cálculos: el almacenamiento de los resultados conocidos en una tabla indexada por los argumentos de la función en cada llamada. El esquema recursivo con memorización se muestra en la figura 19.

Figura 19. Esquema recursivo de programación dinámica con memorización.

```
1  scheme RescursiveDynamicProgrammingWithMemoization
2    var
3      result: vector de ℝ indexado por D
4    auxiliary functions
5      δ: D → P(D)
6      g: D × D → ℝ
7      h: D → ℝ
8      function F(a)
9        if a ∉ result:
10         if a ∈ I:
11           result[a] ← h(a)
12         else:
13           for b ∈ δ(a):
14             if b ∉ result:
15               F(b)
16           result[a] ← opt_{b∈δ(a)} (result[b] ⊕ g(b, a))
17       return result[a]
18   return opt_{a∈F} F(a)
```

Fuente: introducción a los Algoritmos voraces.pdf

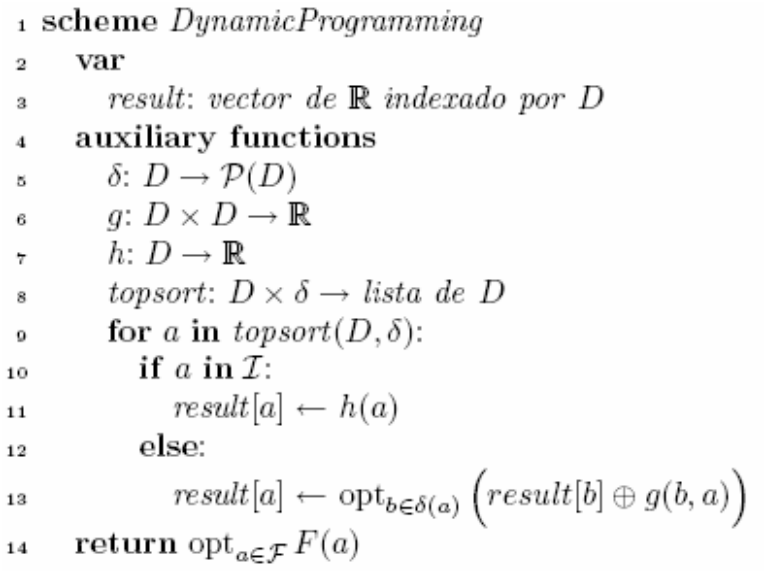
2.2.4.1.4 Transformación recursivo-iterativa.

La ecuación recursiva induce cierto grafo al que denominamos grafo de dependencias entre resultados. Los vértices son los estados, es decir, los elementos de D. Las aristas son pares (d',d) donde $d' \in \delta(d)$. Hay una correspondencia entre soluciones factibles y caminos del grafo entre el conjunto de vértices iniciales, τ , y el de vértices finales, F .

Dicho camino se expresaba como una secuencia de estados. Un error fácil de cometer es confundir la secuencia de elementos con la que expresamos una solución factible con una secuencia de estados. Es posible que expresemos una solución como una secuencia de aristas del grafo de dependencias. En los dos modelados del problema del desglose de moneda, por ejemplo, cada solución factible se describía en términos de la secuencia de aristas que conformaba un camino en el grafo de dependencias.

Encontrar el conjunto de estados es, quizá, una de las dificultades más notables a la hora de modelar un problema para abordar su resolución por programación dinámica. El grafo de dependencias es acíclico, así que permite un recorrido de sus vértices en orden topológico. Resolver F(a) para los diferentes valores de a en ese orden indicado conduce a una transformación recursivo-iterativa de la ecuación. Una versión iterativa resulta de interés en tanto que elimina el sobrécoste de las llamadas recursivas. El coste temporal es proporcional al numeró de aristas en el grafo, y el coste espacial es proporcional al numeró de vértices.

Figura 20. Esquema iterativo de programación dinámica.



```
1  scheme DynamicProgramming
2    var
3      result: vector de ℝ indexado por D
4    auxiliary functions
5      δ: D → 𝒫(D)
6      g: D × D → ℝ
7      h: D → ℝ
8      topsort: D × δ → lista de D
9      for a in topsort(D, δ):
10       if a in ℐ:
11         result[a] ← h(a)
12       else:
13         result[a] ← opt_{b∈δ(a)} (result[b] ⊕ g(b, a))
14   return opt_{a∈ℱ} F(a)
```

Fuente: introducción a los Algoritmos voraces.pdf

Si el grafo de dependencias está estructurado, puede resultar factible reducir la complejidad espacial del algoritmo. Si el grafo es multietapa, por ejemplo, basta con almacenar los resultados asociados a dos etapas: la actual y la previa. Grafos con estructura simple pueden conducir a algoritmos con un consumo de memoria 0(1).

2.2.4.1.5 Calculo de la solución factible óptima.

Calcular el valor optimó de la función objetivo puede resultar meramente instrumental para alcanzar nuestro verdadero propósito: calcular la secuencia que hace optimó el valor de la función objetivo, y que es

$$(\hat{d}_1, \hat{d}_2, ..., \hat{d}_n) = \underset{(d_1, d_2, ..., d_n) \in X}{opt} f((d_1, d_2, ..., d_n)) = \underset{a \in F}{opt} F(a).$$

Podemos conocer la secuencia de decisiones óptimas calculando secuencialmente estos valores:

$$\hat{d}_n = \arg \underset{a \in F}{opt} F(a),$$

$$\hat{d}_{n-1} = \arg \underset{b \in \delta(\hat{d}_n)}{opt} F(\hat{d}_n),$$

$$\hat{d}_{n-2} = \arg \underset{b \in \delta(\hat{d}_n - 1)}{opt} F(\hat{d}_{n-1}),$$

.
.
.

$$\hat{d}_1 = \arg \underset{b \in \delta(\hat{d}_2)}{opt} F(\hat{d}_2).$$

Las técnicas de recuperación del camino basadas en el almacenamiento de punteros atrás hacen sencilla la recuperación del camino en el grafo de dependencias. La figura 21 muestra un esquema que aplica esta técnica.

Figura 21. Esquema iterativo de programación dinámica con recuperación de la solución óptima mediante punteros hacia atrás.

```
1  scheme FactibleSolutionByDynamicProgrammingAndBacktracing
2    var
3      result: vector de ℝ indexado por D
4      backp: vector de D indexado por D
5    auxiliary functions
6      δ: D → 𝒫(D)
7      g: D × D → ℝ
8      h: D → ℝ
9      topsort: D × δ → lista de D
10     for a in topsort(D, δ):
11       if a in ℐ:
12         result[a] ← h(a)
13         backp[a] ← None
14       else:
15         result[a] ← opt_{b∈δ(a)} (result[b] ⊕ g(b, a))
16         backp[a] ← arg opt_{b∈δ(a)} (result[b] ⊕ g(b, a))
17     â ← opt_{a∈ℱ} F(a)
18     d ← [â]
19     while backp[â] ≠ None:
20       â ← backp[a]
21       d.append(â)
22     d.reverse()
23   return d
```

Fuente: introducción a los Algoritmos voraces.pdf

Por regla general, el espacio que requiere esta otra versión para almacenar los punteros hacia atrás elimina la eventual reducción de complejidad espacial alcanzada en el algoritmo anterior, es decir, requiere memoria orden del numeró de vértices del grafo de dependencias.

2.2.4.2 Alineamiento temporal no lineal.

2.2.4.2.1 Ecuación recursiva.

Ya hemos formalizado el problema el término de calculó del valor mínimo de una función sobre cierto conjunto de secuencias. Derivemos la ecuación recursiva que permite calcular la DTW entre a y b. Pensemos en qué opciones tenemos para el ultimó par de puntos alineados: sólo una. Por definición, los últimos puntos alineados son a_m y b_n. Pensemos, pues, en qué opciones tenemos para el penúltimo par:

- Puede que a_{m-1} se alinee con b_n,
- O que b_{n-1} se alinee a_m,
- o que a_{m-1} se alinee con b_{n-1}.

En el primer caso se produce un nuevo subproblema: alinear $a_{1:m-1}$; con $b_{1:n}$; en el segundo, alinear $a_{1:m}$ con $b_{1:n-1}$; y en el tercer caso, alinear $a_{1:m-1}$ con $b_{1:n-1}$. Ya se puede adivinar la recursión. Es decir, paso a paso:

$$DWT(a,b) = \min_{((i_1,j_1),(i_2,j_2),\ldots,(i_p,j_p))\in X(a_{1:p},b_{i:p})} \sum_{1\le k\le p} d(a_{i_k},b_{j_k})$$

$$= \min_{(i_p-1,j_{p-1})\in\{(m-1,n),(m,n-1),(m-1,n-1)\}}$$

$$\left(\min_{((i_1,j_1),(i_2,j_2),\ldots,(i_{p-1},j_{p-1}))\in X(a_{1:p-1},b_{i:p-1})} \left(\sum_{1\le k\le p} d(a_{i_k},b_{j_k}) \right) + d(a_m,b_n) \right)$$

$$= \min_{(i_{p-1}, j_{p-1} \in \{(m-1,n),(m,n-1),(m-1,n-1)\})} DWT(a_{1:i_{p-1}}, b_{1:j_{p-1}}) + d(a_m, b_n)$$

$$= \min \begin{Bmatrix} DWT(a_{1:m-1}, b_{1:n}) + d(a_m, b_n), \\ DWT(a_{1:m}, b_{1:n-1}) + d(a_m, b_n), \\ DWT(a_{1:m-1}, b_{1:n-1}) + d(a_m, b_n) \end{Bmatrix}.$$

Son particularmente sencillos los casos en que se alinea una secuencia formada con un sólo punto con otra de uno o más puntos:

$$DWT(a_1, b_{1:j}) = \sum_{1 \le k \le j} d(a_i, b_k) = DWT(a_1, b_{1:j-1}) + d(a_1, b_j),$$

$$DWT(a_{1:i}, b_1) = \sum_{1 \le k \le i} d(a_k, b_1) = DWT(a_{1:i-1}, b_1) + d(a_i, b_1).$$

Y resulta trivial alinear una secuencia de un solo punto con otra de un solo punto:

$$DWT(a_1, b_1) = d(a_1, b_1)$$

Abusando de la notación, mostramos la ecuación recursiva con $DWT(i, j)$ para hacer referencia ha $DWT(a_{1:i}, b_{1:j})$:

$$DWT(i,j) = \begin{cases} d(a_1, b_1), & \\ DWT(1, j-1) + d(a_1, b_j), & si\ \ i = 1\ \ y\ \ j = 1; \\ DWT(i-1, 1) + d(a_i, b_1), & si\ \ i = 1\ \ y\ \ j \triangleright 1; \\ \min \begin{Bmatrix} DWT(i-1, j) + d(a_i, b_j), \\ DWT(i, j-1) + d(a_i, b_j), \\ DWT(i-1, j-1) + d(a_i, b_j) \end{Bmatrix} & \begin{matrix} si\ \ i \triangleright 1\ \ y\ \ j = 1; \\ \\ ,si\ \ i \triangleright 1\ \ y\ \ j \triangleright 1; \end{matrix} \end{cases}$$

Fuente: introducción a los Algoritmos voraces.pdf

Podemos refinar la ecuación para evitar la realización de tres cálculos de distancia euclidea en la minimización cuando una es suficiente:

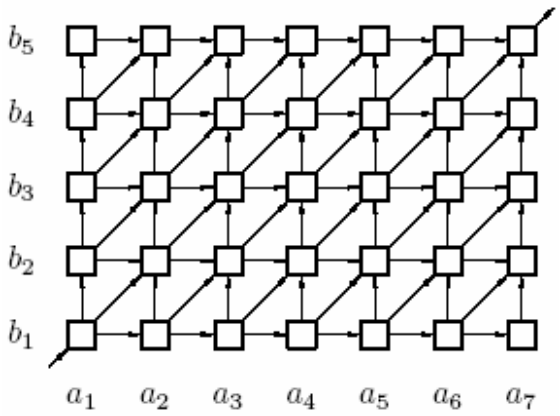
$$DWT(i,j)=\begin{cases} d(a_1,b_1), & \\ DWT(1,j-1)+d(a_1,b_j), & si\ \ i=1\ \ y\ \ j=1; \\ DWT(i-1,1)+d(a_i,b_1), & si\ \ i=1\ \ y\ \ j\triangleright 1; \\ \min\begin{Bmatrix} DWT(i-1,j), \\ DWT(i,j-1), \\ DWT(i-1,j-1) \end{Bmatrix}+d(a_i,b_j), & si\ \ i\triangleright 1\ \ y\ \ j=1; \\ & ,si\ \ i\triangleright 1\ \ y\ \ j\triangleright 1; \end{cases}$$

2.2.4.2.2 Algoritmo iterativo.

En la figura 22 se muestra el grafo de dependencias inducido por la ecuación recursiva. Es un grafo mallado similar al que subyace al calculó de la distancia de Levenshtein.

Se puede efectuar un recorrido por filas, columnas, diagonales.

Figura 22. Grafo de dependencias en el calculó de la distancia de alineamiento temporal no lineal entre dos secuencias de puntos de tallas.



He aquí un ejemplo para el algoritmo.

dtw.py

```
1 def d(v, w): # distancia euclídea entre dos puntos.
2   return sqrt((v[0]-w[0])**2 + (v[1]-w[1])**2)
```

```
3
4 def DTW(a, b):
5   result = {}
6   result[0,0] = d(a[0], b[0])
7   for i in range(1, len(a)):
8     result[i,0] = result[i-1,0] + d(a[i], b[0])
9   for j in range(1, len(b)):
10    result[0,j] = result[0,j-1] + d(a[0], b[j])
11    for i in range(1, len(a)):
12      result[i,j] = min(result[i-1,j], result[i,j-1], result[i-1,j-1]) + d(a[i],b[j])
13  return result[len(a)-1,len(b)-1]
```

Fuente: introducción a los Algoritmos voraces.pdf

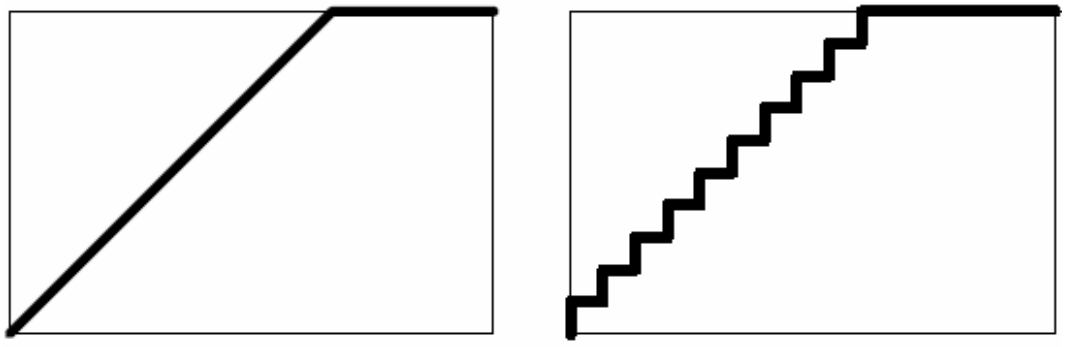
2.2.4.2.3 Normalización del resultado.

La función DTW, tal cual ha sido definida, presenta un problema: tiende a producir valores más altos cuando se le suministran secuencias largas aunque parecidas que cuando se le presentan secuencias cortas y muy diferentes. Es un efecto producido porque en el primer caso participan más sumandos en el sumatorio. Para que la medida de DTW sea inmune a este efecto es necesario trabajar con la distancia promedio entre pares alineados. Está es una definición de la distancia de alineamiento temporal no lineal que incorpora esta idea del promedio:

$$DWT(a,b) = \min_{((i_1,j_1),(i_2,j_2),\ldots,(i_p,j_p))\in X(a,b)} \frac{\sum_{1\le k\le p} d(a_{i_k}, b_{j_k})}{p},$$

Es decir, dividir la distancia entre el numeró de emparejamientos entre pares de puntos que hace el alineamiento optimó. Pero hay un problema con esta definición: conduce a un algoritmo de programación dinámica que requiere tiempo $O(m^2 n)$, pues el numeró de emparejamientos varia entre $\max(m,n)$ y $m+n$ como se ve en la figura 23 y se ha de encontrar el alineamiento optimó para cada posible valor.

Figura 23. Representación esquemática de (a) el camino con el menor numeró de emparejamientos, $\max(m,n)$ y (b) el camino con el mayor numeró de emparejamientos, $m+n$.



Fuente: introducción a los Algoritmos voraces.pdf

Una definición alternativa es esta otra:

$$DWT(a,b) = \min_{((i_1,j_1),(i_2,j_2),\ldots,(i_p,j_p))\in X(a,b)} \frac{\sum_{1\le k\le p} d(a_{i_k}, b_{j_k})}{m+n},$$

Esta definición cuenta con la ventaja de que m + n es un valor constante (no depende el del alineamiento optimó) y, por tanto,

$$DWT(a,b) = \frac{\min_{((i_1,j_1),(i_2,j_2),\ldots,(i_p,j_p))\in X(a,b)} \sum_{1\le k\le p} d(a_{i_k}, b_{j_k})}{m+n} = \frac{DWT(a,b)}{m+n}.$$

Sin embargo, el valor finalmente calculado no es un promedio de distancias entre pares de puntos: el numeró de pares de puntos que forman un alineamiento entre a y b está comprendido entre max(m, n) y m + n, y siempre dividimos por m + n.
Una técnica frecuentemente utilizada para obtener un verdadero promedio consiste en hacer que los alineamientos entre pares de puntos que corresponden a aristas (diagonales) en el grafo de dependencias pesen el doble que las otras. Es decir, se plantea la resolución de esta otra ecuación recursiva:

$$DWT(i,j) = \begin{cases} 2d(a_1,b_1), & \\ DWT(1,j-1) + d(a_1,b_j), & si\ i=1\ \ y\ \ j=1; \\ DWT(i-1,1) + d(a_i,b_1), & si\ i=1\ \ y\ \ j \triangleright 1; \\ \min\begin{cases} DWT(i-1,j) + d(a_i,b_i), \\ DWT(i,j-1) + d(a_i,b_i), \\ DWT(i-1,j-1) + 2d(a_i,b_i) \end{cases} & si\ i \triangleright 1\ \ y\ \ j=1; \\ & ,si\ i \triangleright 1\ \ y\ \ j \triangleright 1; \end{cases}$$

y devolver como distancia entre a y b el valor DTW(a, b)/(m + n). Se observa en la práctica que esta distancia normalizada produce mejores resultados que la que no tiene en cuenta la normalización.

2.2.5 Microcontroladores PIC.

2.2.5.1 Microcontrolador.

Un microcontrolador es un computador completo (microprocesador + E/S +memoria + otros periféricos) (Figura 24), aunque de limitadas prestaciones, que esta contenido con el chip de un circuito integrado programable y se destina a gobernar una sola tarea con el programa que reside en su memoria, sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores[9].

Figura 24. Microcontrolador.

2.2.5.2 Diferencia entre microprocesador y microcontrolador.

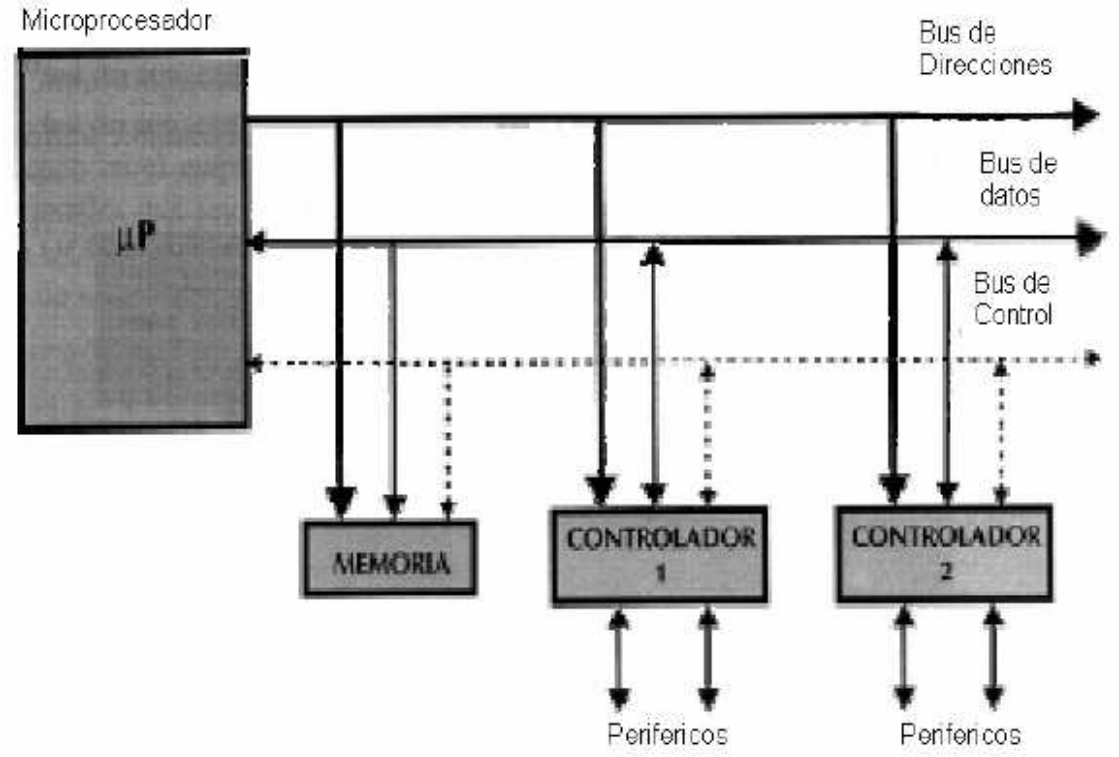
El microprocesador es un circuito integrado que contiene la unidad central de procesador (CPU), también llamada procesador, de un computador. La CPU está

[9] ANGULO, Usategui José María, ANGULO, Martínez Ignacio; "Microcontroladores PIC, Diseño practico de aplicaciones", Segunda Edición, Montevideo, McGraw-Hill, 1999.

formada por la unidad de control, que interpreta las instrucciones, y el camino de datos que las ejecuta.

Los pines de un microprocesador sacan al exterior las líneas de sus buses de direcciones, datos y control, para permitir conectarle con la Memoria y los módulos de E/S y configurar un computador implementando por varios circuitos integrados. Se dice que un microprocesador es un sistema abierto porque su configuración es variable de acuerdo con la aplicación a la que destine (Figura 25)

Figura 25. Microprocesador

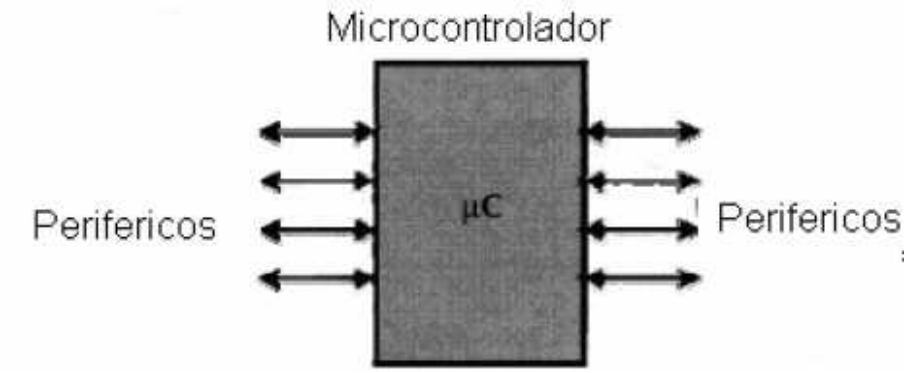


Si sólo se dispone de un modelo de microcontrolador, este debería tener muy potenciados todos sus recursos para poderse adaptar a las exigencias de las diferentes aplicaciones. En la práctica cada fabricante de microcontroladores

oferta un elevado número de modelos diferentes, desde lo más sencillo hasta lo más poderosos. Es posible seleccionar la capacidad de las memorias, el número de líneas de E/S, la cantidad y potencia de los elementos auxiliares, la velocidad de funcionamiento, etc. Por todo esto, un aspecto muy destacado del diseño es la selección del microcontrolador a utilizar.

El microcontrolador es un sistema cerrado. Todas las partes del computador están contenidas en su interior y solo salen al exterior las líneas que gobiernan los periféricos (Figura 26).

Figura 26. Microcontrolador



2.2.5.3 Arquitectura de un microcontrolador.

La arquitectura tradicional de computadoras y microprocesadores se basa en el esquema propuesto por John Von Neumann, en el cual la unidad central de proceso, o CPU, esta conectada a una memoria única que contiene las instrucciones del programa y los datos[10] (Figura 27).

El tamaño de la unidad de datos o instrucciones esta fijado por el ancho del bus de la memoria. Es decir que un microprocesador de 8 bits, que tiene además un bus de 8 bits que lo conecta con la memoria, deberá manejar datos e instrucciones de una o más unidades de 8 bits (bytes) de longitud. Cuando deba acceder a una

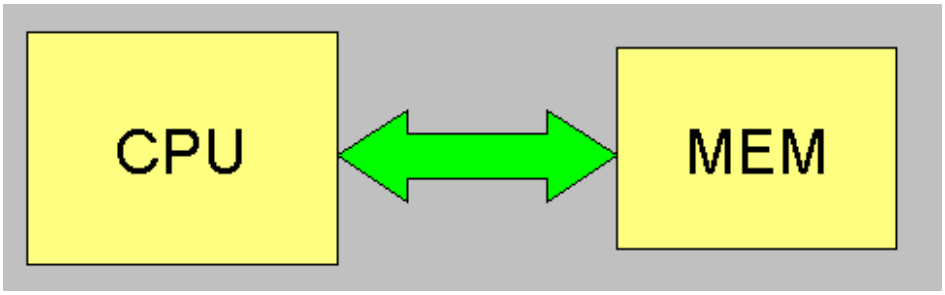
[10] http://es.wikipedia.org/wiki/Arquitectura_von_Neumann. 06/11/2005, 10:27.

instrucción o dato de más de un byte de longitud, deberá realizar más de un acceso a la memoria. Por otro lado este bus único limita la velocidad de operación del microprocesador, ya que no se puede buscar de memoria una nueva instrucción, antes de que finalicen las transferencias de datos que pudieran resultar de la instrucción anterior. Es decir que las dos principales limitaciones de esta arquitectura tradicional son:

• Que la longitud de las instrucciones está limitada por la unidad de longitud de los datos, por lo tanto el microprocesador debe hacer varios accesos a memoria para buscar instrucciones complejas.
• Que la velocidad de operación (o ancho de banda de operación) esta limitada por el efecto de cuello de botella que significa un bus único para datos e instrucciones que impide superponer ambos tiempos de acceso.

La arquitectura von Newmann permite el diseño de programas con código automodificable, práctica bastante usada en las antiguas computadoras que solo tenían acumulador y pocos modos de direccionamiento, pero innecesaria, en las computadoras modernas. (Figura 27)

Figura 27. Arquitectura Von Newmann.



Fuente: http://es.wikipedia.org/wiki/Arquitectura_von_Neumann. 06/11/2005, 10:27.

La arquitectura Harvard y sus ventajas:

La arquitectura conocida como Harvard, consiste simplemente en un esquema en el que el CPU esta conectado a dos memorias por intermedio de dos buses separados. Una de las memorias contiene solamente las instrucciones del programa, y es llamada Memoria de Programa. La otra memoria solo almacena los datos y es llamada Memoria de Datos (Figura 28). Ambos buses son totalmente independientes y pueden ser de distintos anchos. Para un procesador de Set de Instrucciones Reducido, o RISC (Reduced Instrucción Set Computer), el set de instrucciones y el bus de la memoria de programa pueden diseñarse de manera tal que todas las instrucciones tengan una sola posición de memoria de programa de longitud. Además, como los buses son independientes, el CPU puede estar accediendo a los datos para completar la ejecución de una instrucción, y al mismo tiempo estar leyendo la próxima instrucción a ejecutar.
Se puede observar claramente que las principales ventajas de esta arquitectura son:

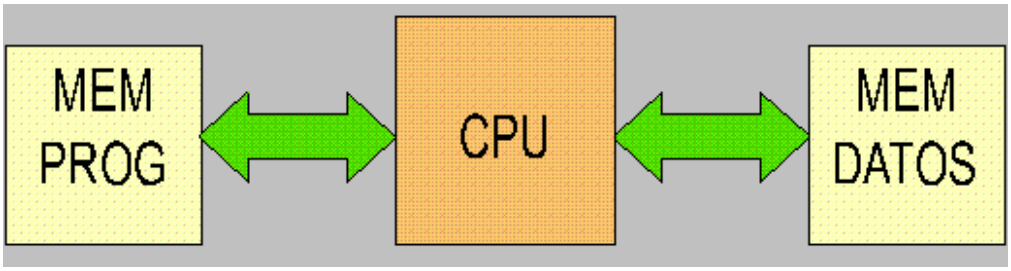
• Que el tamaño de las instrucciones no está relacionado con el de los datos, y por lo tanto puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa, logrando así mayor velocidad y menor longitud de programa.

• Que el tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad de operación.

Una pequeña desventaja de los procesadores con arquitectura Harvard, es que deben poseer instrucciones especiales para acceder a tablas de valores constantes que pueda ser necesario incluir en los programas, ya que estas tablas

se encontraran físicamente en la memoria de programa (por ejemplo en la EPROM de un microprocesador). (Figura 28)

Figura 28. Arquitectura Harvard.



Fuente: http://es.wikipedia.org/wiki/Arquitectura_von_Neumann. 06/11/2005, 10:27.

Los microcontroladores PIC 18FXXX poseen arquitectura Harvard, con una memoria de datos de 1536 Bytes, 256 bytes de EEPROM, una memoria de programa de 16.384 Instrucciones.

2.2.5.4. El procesador.

Es el elemento más importante del microcontrolador y determina sus principales características, tanto a nivel hardware como software. Se encarga de direcionar la memoria de instrucciones, recibir la instrucción en su curso, su descodificación y la ejecución de la operación que implica dicha instrucción, así como la búsqueda de los operándoos y el almacenamiento del resultado.
Existen tres orientaciones en cuanto a la arquitectura o funcionalidad de los procesadores actuales.

CISC: Un gran número de procesadores usados en los microcontroladores están basados en la filosofía CISC (Computadores de Juego de Instrucciones

Complejo). Disponen de más de 80 instrucciones máquina en su repertorio algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos para una ejecución.
Existen tres orientaciones en cuanto a la arquitectura y funcionalidad de los procesadores actuales.

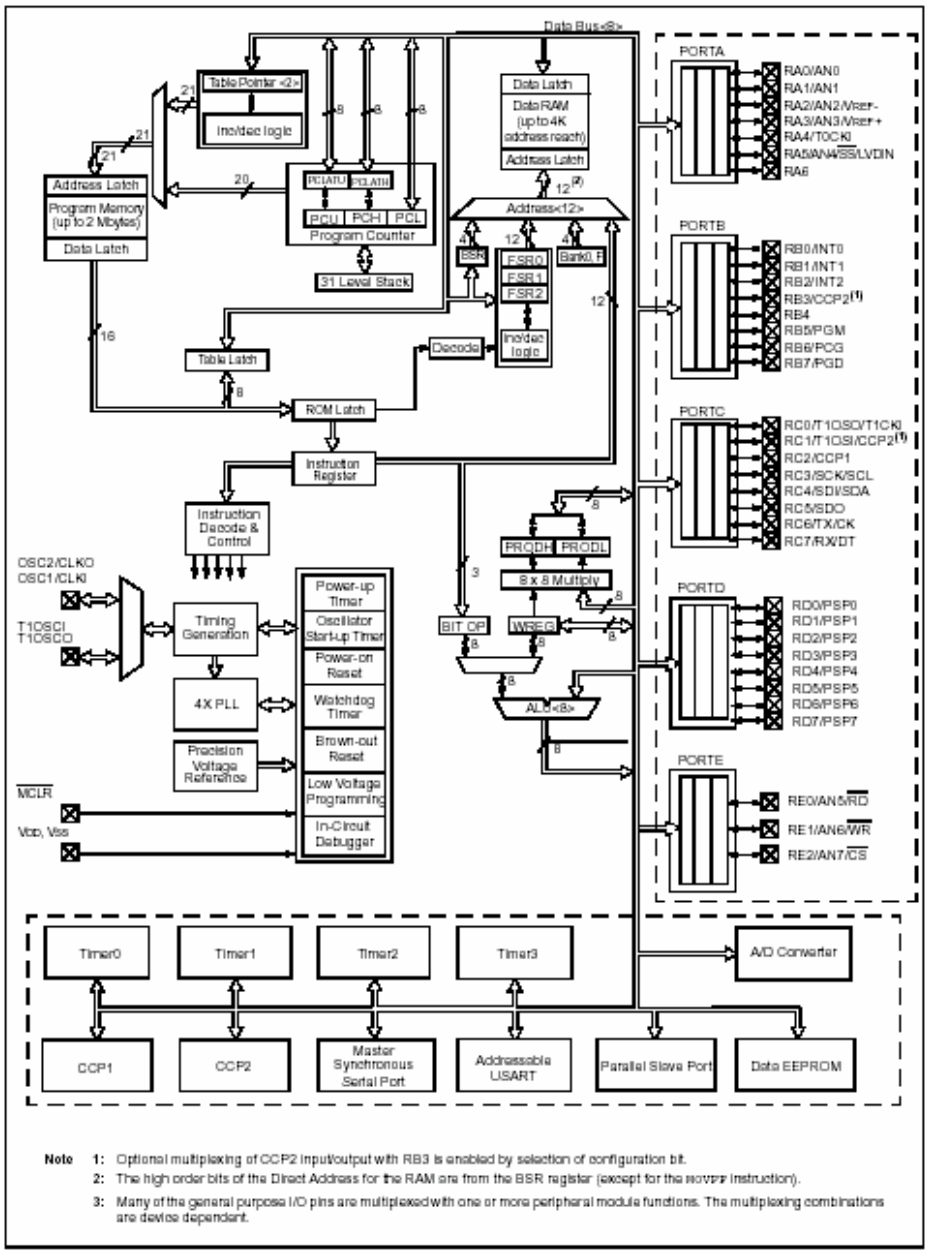
RISC: Tanto la industria de los computadores comerciales como la de los microcontroladores están decantándose hacia la filosofía RISC (Computadores de Juego de Instrucciones Reducido). En estos procesadores el repertorio de instrucciones maquine es muy reducido y las instrucciones son simples y, generalmente, se ejecutan en un ciclo.
La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador.

SISC: (Computadores de Juego de Instrucciones Especifico): En los microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser muy reducido, es "especifico", es decir, las instrucciones se adaptan a las necesidades de la aplicación prevista.

2.2.5.4.1 El microcontrolador del PIC18F452.

El PIC18F452 tiene un procesador de alto rendimiento RISC es capaz de procesar hasta 10MIPS (con un reloj de 40Mhz) con instrucciones de 16 Bits y un camino de datos de 8 bits: Además incorpora un multiplicador de 8x8 en hardware de un solo ciclo.
La memoria de programa puede direccional hasta 32KB (FLASH) la memoria de datos hasta 1.5KB, i una EEPROM de 256 bytes.
Las interrupciones disponen de diferentes niveles de prioridad.

Figura 29. Diagrama de Boques PIC18F452.



Note 1: Optional multiplexing of CCP2 input/output with RB3 is enabled by selection of configuration bit.
2: The high order bits of the Direct Address for the RAM are from the BSR register (except for the MOVFF instruction).
3: Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.

Fuente: Microchip. [En línea]. Pagina Web, URL < http:// microchip.com >. 24/09/2005, 16:14

2.2.5.5 Organización de la Memoria.

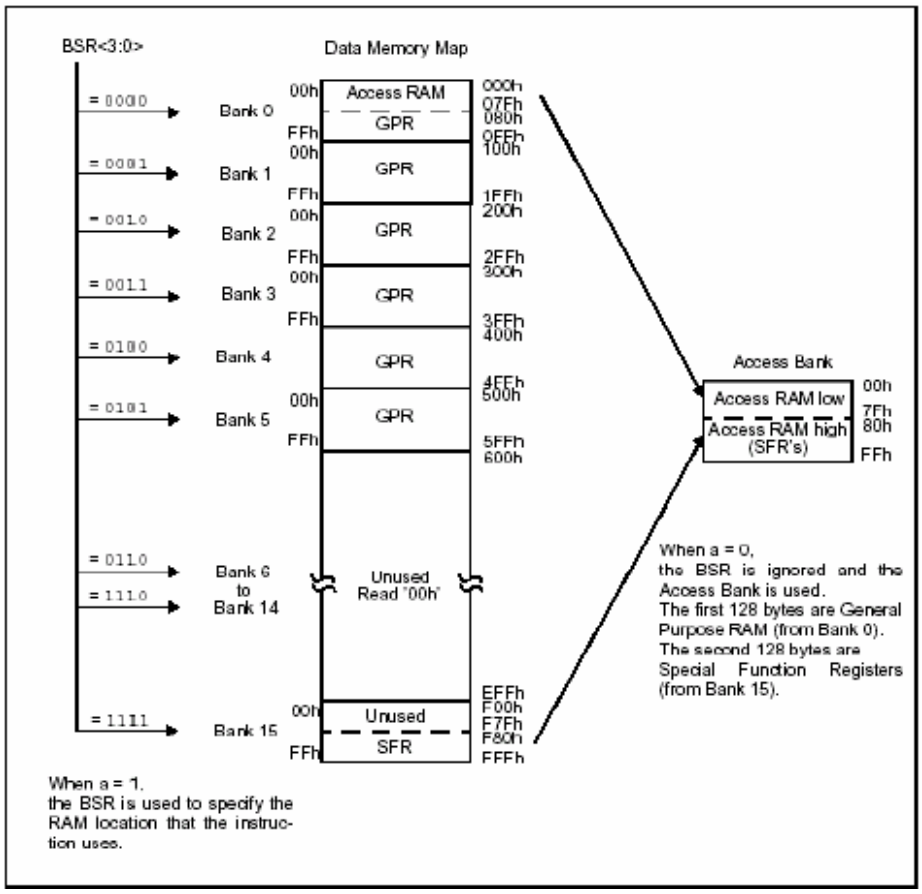
2.2.5.5.1 Memoria Interna (RAM).

La memoria interna de datos, también llamada archivo de registros (register file), cada registro de la memoria tiene una dirección de 12bits, es decir que obtenemos una capacidad de 4096 bytes. El mapa de memoria se divide en 16 bancos de 256 bytes, cada uno.

La memoria de datos está dividida en dos áreas. Una de ellas corresponde al banco de Registro de Propósito General (GPR), y la otra dedicada a los Registros de Funciones Especiales (SFR), que controlan los recursos y periféricos del microcontrolador. Las dos áreas están repartidas en Bancos, que se seleccionan mediante ciertos bits destinados a ese propósito que se hallan en el Registro de Selección de Banco (BSR). Cuando se realiza un acceso a una posición situada fuera de los bancos. Se ignoran los bits del BSR.

El Registro de Selección del banco se emplea en conmutación de bancos en el área de la memoria de datos, los 4 bits bajos del BSR determina a que banco se accederá. (Figura 30)

Figura 30. Organización de la memoria Interna (RAM) en la familia PIC18F452.



Fuente: Embedded Control Handbook, Microchip

2.2.5.5.2 Memoria de Programa.

El contador de Programa PC, tiene un tamaño de 21 bits y proporciona la dirección de la instrucción a la que se accede. Con 21 bits se puede direccional hasta 2 Mbytes de memoria de programa. La memoria de programa la utilizaremos en el proyecto para el código.

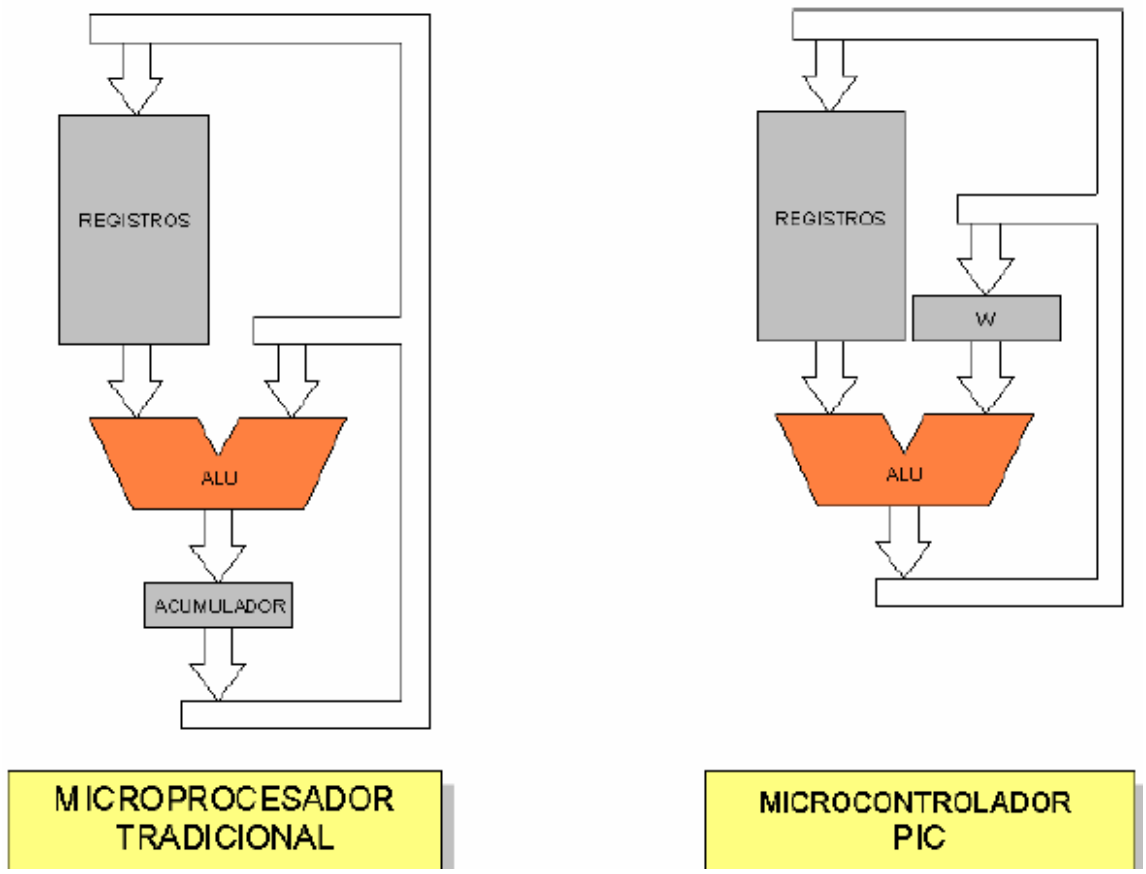
Los microcontroladores dotados de memoria EEPROM una vez instalados en el circuito, pueden grabarse y borrarse cuantas veces se quiera sin ser retirados de dicho circuito. Para ello se usan “grabadores en circuito” que confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo.

El número de veces que puede grabarse y borrarse una memoria EEPROM es finito, por lo que no es recomendable una reprogramación continúa. Son muy idóneos para la enseñanza y la Ingeniería de diseño.
Se va extendiendo en los fabricantes la tendencia de incluir una pequeña zona de memoria EEPROM en los circuitos programables para guardar y modificar cómodamente una serie de parámetros que adecuan el dispositivo a las condiciones del entorno.

La EEPROM la utilizaremos para almacenar las secuencias a enviar para el sensor a través del PUERTO B.
En los microcontroladores tradicionales todas las operaciones se realizan sobre el acumulador. La salida del acumulador está conectada a una de las entradas de la Unidad Aritmética y Lógica (ALU), y por lo tanto éste es siempre uno de los dos operándoos de cualquier instrucción. Por convención, las instrucciones de simple operando (borrar, incrementar, decrementar, complementar), actúan sobre el acumulador. La salida de la ALU va solamente a la entrada del acumulador, por lo tanto el resultado de cualquier operación siempre quedara en este registro. Para operar sobre un dato de memoria, luego realizar la operación siempre hay que mover el acumulador a la memoria con una instrucción adicional.

En los microcontroladores PIC, la salida de la ALU va al registro W y también a la memoria de datos, por lo tanto el resultado puede guardarse en cualquiera de los dos destinos. En las instrucciones de doble operando, uno de los dos datos

siempre debe estar en el registro W, como ocurría en el modelo tradicional con el acumulador. En las instrucciones de simple operando el dato en este caso se toma de la memoria (también por convención). La gran ventaja de está arquitectura es que permite un gran ahorro de instrucciones ya que el resultado de cualquier instrucción que opere con la memoria, ya sea de simple o doble operando, puede dejarse en la misma posición de memoria o en el registro W, según se seleccione con un bit de la misma instrucción. Las operaciones con constantes provenientes de la memoria de programa (literales) se realizan solo sobre el registro W. (Figura 31, Figura 32)



MICROPROCESADOR TRADICIONAL

MICROCONTROLADOR PIC

Figura 31. Representa un diagrama simplificado de la arquitectura interna del camino de los datos en el CPU de los microcontroladores PIC. Este diagrama puede no representar con exactitud el circuito interno de estos microcontroladores, pero es exacto y claro desde la óptica del programador.
Figura 32. Representa el mismo diagrama para un microprocesador ficticio de arquitectura tradicional. Se puede observar que la principal diferencia entre ambos radica en la ubicación del registro de trabajo, que para los PIC's se denomina W (Working Register), y para los tradicionales es el Acumulador (A).

2.2.5.5.3 Contador de programa.

Este registro, normalmente denominado PC, es totalmente equivalente al de todos los microprocesadores y contiene la dirección de la próxima instrucción a ejecutar. Se incrementa automáticamente al ejecutar cada instrucción, de manera que la secuencia natural de ejecución del programa es lineal, una instrucción después de la otra. Algunas instrucciones que llamaremos de control, cambian el contenido del PC alterando la secuencia lineal de ejecución.
El PC es un registro de 21bits en los 18F452, lo que permite direccional 2Mbytes de memoria de programa.

2.2.5.6 Puertos de Entrada/Salida.

La principal utilidad de los p ines que posee la cápsula que contiene un microcontrolador es soportar las líneas de E/S que comunican al computador interno con los periféricos exteriores.
Según los controladores de periféricos que posea cada modelo de microcontrolador, las líneas de E/S se destinan a proporcionar el soporte a las señales de entrada, salida y control.

Los microprocesadores PIC18F452 tienen cinco puertos de entrada/salida paralelo de usos generales llamados PUERTO A, PUERTO B, PUERTO C, PUERTO D y PUERT E. Algunos pines de estos puertos están multiplexados con funciones específicas de algunos periféricos. Se puede la función mediante el software.
Cada puerto tiene tres registros de operación:

• El registro 'TRIS' que determina la dirección del puerto ('1' = entrada, '0' = salida).
• El registro 'PORT' que establece los niveles leídos a los pines del puerto.
• El registro 'LAT' como alternativa al registro anterior.

2.2.5.7 Perro guardián o "Watchdog".

Cuando el computador personal se bloquea por un fallo del software u otra causa, se pulsa el botón del reset y se reinicializa el sistema. Pero un microcontrolador funciona sin el control de un supervisor y de forma continuada las 24 horas del día. El Perro guardián consiste en un temporizador que, cuando se desborda y pasa por 0, provoca un reset automáticamente en el sistema.
Se debe diseñar el programa de trabajo que controla la tarea de forma que refresque o inicialice al Perro guardián antes de que provoque el reset. Si falla el programa o se bloquea, no se refrescará al Perro guardián y, al completar su temporización, "ladrará y ladrará" hasta provocar el reset.

2.2.5.8 Estado de reposo o de bajo consumo.

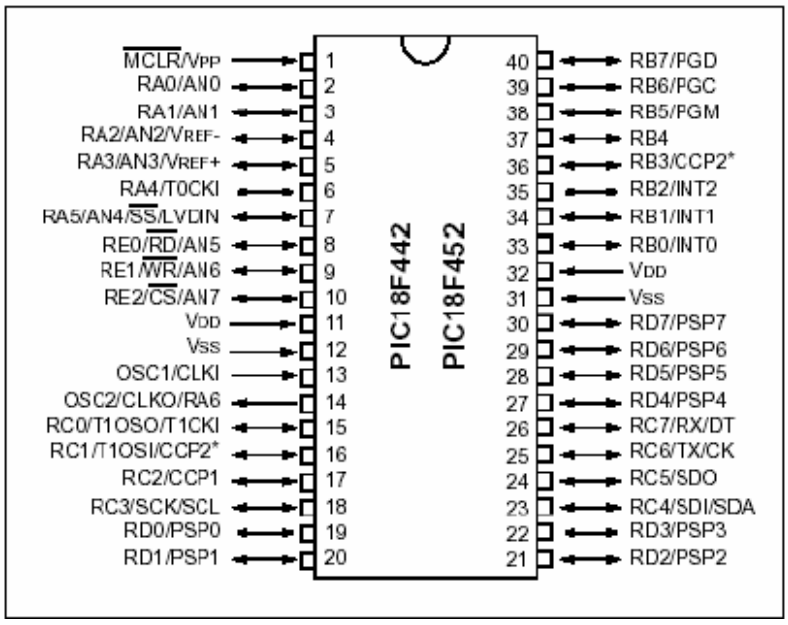
Son abundantes las situaciones reales de trabajo en que el microcontrolador debe esperar, sin hacer nada, a que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, (factor clave en los aparatos portátiles), los microcontroladores disponen de una instrucción especial (SLEEP en los PIC), que les pasa al estado de reposo o de bajo consumo, en el cual los requerimientos de potencia son mínimos. En dicho estado se detiene el

reloj principal y se “congelan” sus circuitos asociados, quedando sumido en un profundo “sueño” el microcontrolador. Al activarse una interrupción ocasionada por el acontecimiento esperado, el microcontrolador se despierta y reanuda su trabajo.

2.2.1.9 Encapsulado.

El PIC18F452 esta disponible en varios formatos se utilizara el encapsulado DIP. Este es el encapsulado más empleado en montajes por taladro pasante en placa. Este puede ser cerámico (marrón) o de plástico (negro). Un dato importante en todos los componentes es la distancia entre patillas que poseen, en los circuitos es de vital importancia este dato, así en este tipo el estándar se establece en 0,1 pulgadas (2,54 mm). (Figura 33)

Figura 33. Encapsulado DIP del PIC18F452.



Fuente: Microchip. [En línea]. Pagina Web, URL < http:// microchip.com >. 24/09/2005, 16:14

Tabla 1. Pines del microcontrolador.

Nº de Pines	Nombre	Descripción
1	MCLR	Es un reset a nivel bajo.
2	AN0	AN0 recibe la salida del sensor
4	RA2	Entrada de referencia de la A/D.
5	RA3	Entrada de referencia de la A/D
11,32	Vdd	Alimentación positiva
12,31	Vss	Masa de referencia
13	OSC1	Entrada del oscilador de cristal
14	OSC2	Salida del oscilador de cristal.
15	RC0	Entrada de referencia para la A/D externa.
17	RC2	Entrada de referencia para la A/D externa.
18	RC3	Entrada de referencia para la A/D externa.
21	RD2	Entrada de referencia para la A/D externa.
23	RC4	Entrada de referencia para la A/D externa.
24	RC5	Entrada de referencia para la A/D externa.
25	TX	Transmisión asíncrona USART (comunicación con el PC)
26	RX	Recepción asíncrona USART (comunicación con el PC)
35	RB2	Salida del pulso ST hacia el sensor.
36	RB3	Salida del pulso CLK hacia el sensor.

Fuente: Microchip. [En línea]. Pagina Web, URL < http:// microchip.com >. 24/09/2005, 16:14

2.2.5.10 Oscilador.

Todos los microcontroladores disponen de un circuito oscilador que genera una onda cuadrada de alta frecuencia, que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema.

Generalmente, el circuito de reloj está incorporado en el microcontrolador y solo se necesitan unos pocos componentes exteriores para seleccionar y estabilizar la frecuencia de trabajo. Dichos componentes suelen consistir en un cristal de cuarzo junto a elementos pasivos o bien un resonador cerámico.

Aumentar la frecuencia de reloj supone disminuir el tiempo en que se ejecutan las instrucciones pero lleva aparejado un incremento del consumo de energía.

Tabla 2. Tipos de osciladores.

LP	32-200Khz	Cristal bajo consumo.
XT	100Hz - 4Mhz.	Cristal o Resonador
HS	4Mhz - 25Mhz	Alta velocidad o resonador
HS+PLL	40Mhz Max	HS con PLL habilitado
RC	4Mhz Max	Malla RC externa
RCIO	4Mhz	RC Interno son salida clock
EC	40Mhz Max	Clock Externo
ECIO	40Mhz Max	EC con salida de clock

Fuente: Microchip. [En línea]. Pagina Web, URL < http:// microchip.com >. 24/09/2005, 16:14

2.2.5.11 Puertos de Comunicación.

Para dotar al microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistema, buses de redes y poder adaptarlos con otros elementos bajo otras normas y protocolos algunos modelos disponen de recursos que se lo permiten.

2.2.5.11.1 Módulo MSSP.

EL módulo MSSP (Master Synchronous Port) es un puerto de comunicación serial síncrona half y full duplex y orientado para comunicaciones con componentes externos.
Puede funcionar en dos modos de comunicación:

• SPI Serial Peripheral Interface
• I2C Inter-Integrated Circuit.

2.2.5.11.2 Módulo Usart.

El Puerto USART (Universal Synchronous Asynchronous Receiver Transmitter) es un puerto serial orientado a establecer comunicación con equipos con protocolo EIA-232.
Hasta 2 puertos seriales Usart pueden estar incorporados en algunos PIC18.Cada puerto serial puede ser configurado en forma distinta.
Puede funcionar en los siguientes modos:

- Asíncrono - Full Duplex
- Síncrono Master - Half Duplex
- Síncrono Slave - Half Duplex

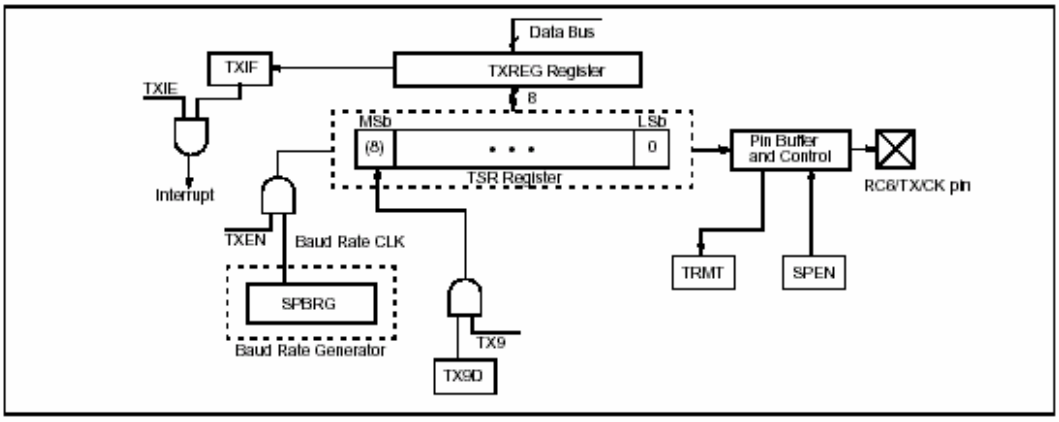
2.2.5.11.2.1 Asíncrono (full-duplex).

La comunicación es bidireccional. La patota RC6/TX actúa como línea de transmisión y la RC7/RX como línea de recepción. Cada dato lleva un bit de inicio y otro de stop. En este modo se transmiten tramas de 10 bits con codificación NRZ, es decir tenemos un bit de inicio a nivel bajo, 8 bits de datos y un bit final de trama a nivel alto.

2.2.5.11.2.2 Sincrono (semiduplex).

Comunicación unidireccional. Una sola línea para los datos que se implementan sobre la patita RC7/RX. En el modo master la señal de reloj sale por el p in RC6/TX. En el modo esclavo entra por ella. (Figura 34).

Figura 34. Diagrama de bloques del transmisor de la USART.



Fuente: http--www_warburtech_com-microcontrollers-40_pin_pic_microcontrollers-pic18f452_microcontroller.htm. 24/09/2005, 16:32.

2.2.5.12 Características PIC 18F452.

El PIC utilizado para realizar este proyecto es un PIC de la familia 18FXXX (18F452). Es un PIC de la gama alta.

El PIC18F452 ofrece un ambiente amistoso del desarrolló del recopilador de 'C',256 octetos de EEPROM, Uno mismo-programando, un ICD, 2 funciones de capture/compare/PWM, 8 canales del convertidor (DE ANALÓGICO A DIGITAL) de analógico a digital 10-bit, el puerto serial síncrono se puede configurar como el interfaz periférico serial 3-wire (SPI™) o el autobús Inter-Integrado de dos hilos del circuito (I²C™) y el transmisor asincrónico universal direccionable del receptor (AUSART). Todas estas características hacen ideal para el equipo de fabricación, l instrumentación y supervisar, la adquisición de datos, la energía que condiciona, el control del medio ambiente, el telecom y los usos del consumidor audio/video.

En la siguiente figura podemos ver la imagen real de este microcontrolador.

Figura 35. PIC 18F452.

Las características principales de este PIC son las siguientes:

• Tecnología CMOS.

• Procesador RISC.

• Memoria Flash de 32k bytes.

• Memoria de instrucciones de 16384.

• Memoria RAM de 1536 bytes.

• Memoria EEPROM de 256 bytes.

• Bus datos de 8 bits, bus de instrucciones de 16 bits.

• Módulo de puerto serie síncrono (3-wire y I2C).

• USART direccionable, soporta RS-485 y RS-232.

• Módulo de puerto paralelo.

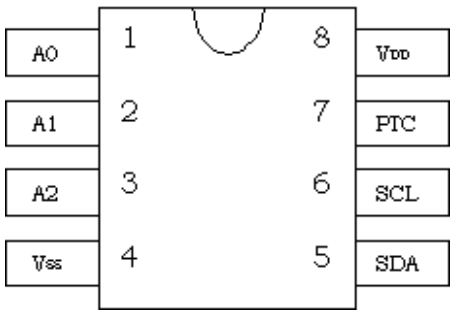
• Módulo conversor A/D de 10 bits.

En la siguiente figura podemos ver cual es la disposición del pastillaje del PIC:

2.2.5.12 Memoria I^2C .

En la figura 24 se ve la distribución de pines y características de la memoria. EEPROM I^2C .

Figura 36. Encapsulado I^2C .



Fuente: Microchip. [En línea]. Pagina Web, URL < http:// microchip.com >. 24/09/2005, 16:14.

En los terminales V_{DD} y V_{SS} se conecta la alimentación.

El terminal SCL es la entrada de reloj y el SDA es el terminal de entrada/salida de datos.

Los terminales A0-A2, son los utilizados para configurar la dirección de identificación del esclavo.

El terminal PTC es la entrada de reloj externo, utilizado en la escritura de la memoria. Este dispositivo posee un oscilador interno, por tanto no es necesario aplicar una señal externa de reloj para la escritura de la memoria.

Las características eléctricas especificadas por el fabricante son las siguientes:

- Tecnología de fabricación CMOS.
- Organización de la memoria en 256 posiciones de memoria de 8 bits cada una.
- Duración mínima de la información de 10 años.
- Ciclos de borrado/escritura de 1.000.000.
- Alimentación de +2.5V a 6V. El terminal V_{DD} es el positivo de alimentación y V_{SS} el negativo.
- Consumo durante el funcionamiento de 2 mA.
- Consumo en standby 4 µA típico.
- Frecuencia en la señal de reloj de 100 KHz.

El funcionamiento de la memoria I^2C , se divide en dos procesos de trabajo:

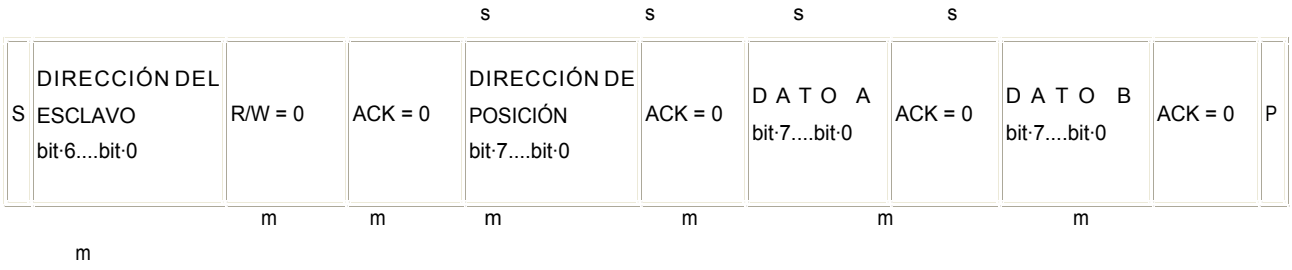
2.2.5.12.1 Proceso de escritura.

Este proceso de trabajo, realiza la escritura de dos bytes cuyas posiciones son consecutivas. Primero el maestro envía una señal de identificación de la memoria a escribir (los 7 bits que identifican a cada uno de los 128 posibles dispositivos), indicando mediante el bit R/W = 0 que se va a realizar una escritura sobre la memoria. Tras el acuse de recepción por parte de la memoria, se le indica la dirección en la que se desea escribir el primer dato que sigue. Sólo después de la recepción y del acuse de recepción de los dos octetos y del envío del bit de stop, se desencadena el mecanismo de borrado/escritura de la memoria que emplea unos 20 ms para realizar el borrado y luego la escritura de estos dos octetos (10 ms por octeto), el incremento de la dirección de la posición del primer octeto al segundo es automático. Este mecanismo de escritura se pone en práctica con

ayuda de un oscilador controlado por una red RC externa o interna (externo a través del terminal PTC). Durante el tiempo que dura el proceso de escritura, si el maestro manda información, la memoria no responderá con la señal de acuse de recepción, por tanto, el maestro tiene constancia del estado de la memoria.

Aunque se ha explicado el proceso de escritura con dos octetos, es obvio que si al mandar el primer octeto el maestro ejecuta un bit de stop, sólo se escribirá este octeto en la posición de memoria indicada. En la figura 37 se encuentra representado este proceso; generado por el maestro.

Figura 37. Trama de Escritura.



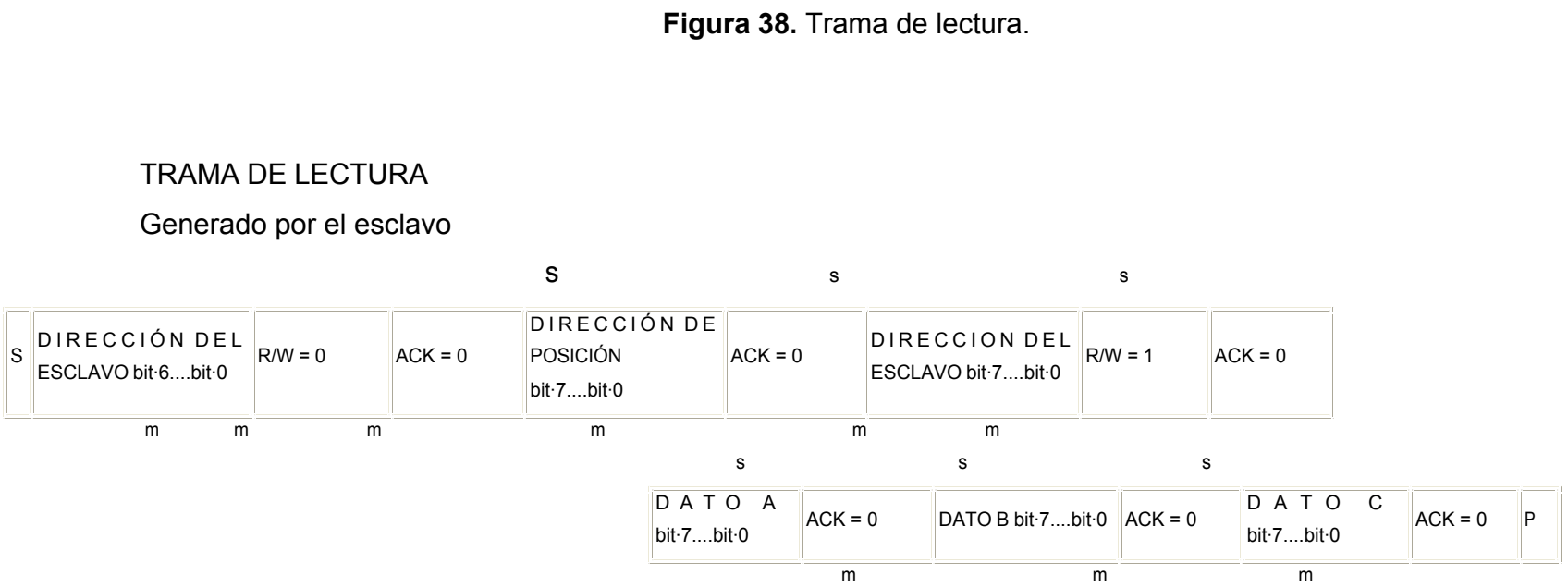
Fuente: www.comunidadelectronicos.com. 09/12/2005, 11:46.

Existe otro modo de escribir en está memoria que consiste en escribir una página completa, compuesta por 8 bytes. El tiempo empleado en este proceso de borrado/escritura de estos 8 bytes, es de unos 28 ms consiguiendo de está forma una reducción en el tiempo total cuando se necesita escribir varios bytes seguidos. El proceso de escritura es el mismo que en el caso de dos bytes, la única salvedad es el número de bytes enviados por el maestro, siendo también automático el incremento de la dirección que se va a escribir.

2.2.5.12.2 Proceso de lectura.

Está memoria tiene dos formas de lectura.
CASO A: el maestro lee la memoria después de haber indicado la dirección de la primera posición a leer. En la figura 18 se puede ver como el inicio de la trama es idéntico al proceso de escritura, pero tras enviar la dirección de la primera posición de memoria, se vuelve a enviar la dirección de identificación de la memoria, poniendo en está ocasión el bit R/W = 1.

Figura 38. Trama de lectura.



Fuente: www.comunidadelectronicos.com. 09/12/2005, 11:46.

Como se observa en la figura 16, tras el bit ACK de acuse de recepción por parte del esclavo, es él el que controla la línea de datos, enviando el dato contenido en la posición de memoria indicada. Tras el octeto enviado por el esclavo, el maestro tiene que generar el bit de acuse de recepción.

El incremento en la dirección de la posición de memoria es, como en el caso de la escritura, automático. Cuando el valor de la posición de la memoria llegue a su máximo valor (255), automáticamente pasará a tomar el valor cero y continuará incrementándose cíclicamente.
El modo de detener la lectura de la memoria es cuando el maestro no activa el bit de acuse de recepción del último octeto leído (ACK = 1). A continuación el maestro debe generar el bit de stop.

CASO B: el maestro lee directamente la memoria después de la orden de lectura. En este caso, se envía una señal al esclavo en modo de lectura y, sin ninguna otra forma de proceso, se le envían sus datos a partir de la última dirección a la que se tuvo acceso por una escritura o por una lectura. No es necesario entonces indicar la dirección de inicio y, por este hecho, se puede ganar tiempo en el intercambio.

El modo de detener la lectura de la memoria es idéntico al caso anterior.
Tras la aplicación de la tensión de alimentación, el registro donde se contiene la dirección de la posición de la memoria estará a cero, pudiendo emplear este caso sabiendo que la dirección de inicio será cero.

3. METODOLOGÍA

3.1 Enfoque de la investigación.

La metodología desarrollada en este proyecto de investigación consta de dos partes

- RECOLECCIÓN DE DATOS
- DESARROLLÓ DE LA INVESTIGACIÓN

La recolección de datos se fundamentó en el uso de la voz para interactuar con dispositivos; facilidad que está ganando adeptos conforme avanzan las técnicas que lo permiten. Dado que existen muchos sistemas en los que no es justificable el sobreprecio que conlleva la incorporación de equipos capaces de atacar con las técnicas más modernas el problema del reconocimiento de la voz resulta interesante explorar las posibilidades que ofrecen microcontroladores en el campo de la interacción vocal.

El desarrolló de la investigación describe la realización de un prototipo capaz de interpretar comandos vocales basados en un microcontrolador de la familia 18FXX2. Describe el hardware mínimo necesario así como el enfoque y la implementación software necesario para llevar a cabo la tarea. En este proyectó se presenta el desarrolló de los sistemas de Reconocimiento Automático del Habla (RAH); los cuales no sólo realizan una decodificación acústica del mensaje hablado emitida por un locutor sino que además, intentan extraer el significado de la misma, con el fin de realizar una acción o una traducción, resultado del entendimiento de la palabra reconocida.

3.2 línea de investigación de usb / sub-línea de facultad / campo temático del programa.

Por medio del Consejo de Investigación Bonaventuriano -CIB- que trabaja la conceptualización de las Líneas Institucionales de Investigación. Se estableció las siguientes líneas para este proyecto:

Tecnologías actuales y sociedad / Procesamiento de señales digitales y analógicas / Análisis y procesamiento de señales.

FUENTE: CIB. Documento sobre Líneas Institucionales de Investigación (USB).

3.3 Técnicas de recolección de información.

Al momento de recolectar la información necesaria para el reconocimiento de caracteres vocálicos para ordenar comandos al televisor, se tuvieron en cuenta aspectos como la actual necesidad de mejorar la comunicación hombre – máquina, que induce a la técnica del reconocimiento automático del habla (RAH).
En términos generales, los sistemas RAH, son la implementación algorítmica de un análisis detallado de las características del habla, y que dependerá del lenguaje usado y de otros conceptos que no deben excluirse (frecuencia fundamental, energía del tramo de señal, etc.).

El primer paso antes de implementar un sistema RAH, es investigar sobre la dependencia del hablante con el sistema, lo cual, permite cimentar la base teórico práctica para la implementación posterior.

La implementación de nuestro reconocedor, consta de tres etapas bien definidas: Etapa de muestreo y cuantificación de la señal de voz, grabación y detección de las muestras y la etapa final de reconocimiento.

El interés principal de la investigación se centra en una correcta implementación de los algoritmos clásicos de extracción de características y reconocimiento de palabras.
Para la etapa de reconocimiento de palabras, se uso el algoritmo de alineamiento temporal dinámico (DTW); el cual, es capaz de discriminar entre palabras con duración temporal independientes de la señal.

La respuesta será visualizada al momento de hacer la interacción directa con el dispositivo implementado para este propósito y adaptado al televisor.

3.4 Población y muestra.

La población tomada para este trabajo, se centra en los diferentes fabricantes y tipos de televisores que existen, y la muestra son todas aquellas personas que poseen y utilizan televisores en sus vidas ya sea como información, distracción etc.

3.5 Hipótesis.

La complejidad de los sistemas de reconocimiento de voz nos han inducido a mejorar la comunicación hombre – máquina, junto con su precisión; por lo tanto se planteo una ayuda para toda persona que usa televisor en sus vidas como

recreación y distracción, obteniendo una manipulación más sencilla; teniendo en cuenta que la transmisión de las señales de voz para la grabación, debe realizarse de forma adecuada, es decir de forma continua, clara y sin pausas.

El dispositivo de reconocimiento de caracteres vocálicos, para ordenar comandos al televisor, dará resultados, entre un 85% y 95% de exactitud, de las órdenes transmitidas a ejecutar.

3.6 Variables.

3.6.1 Variables independientes.

Dentro de las variables independientes se encuentran:

- La Voz
    - ✓ Duración.
    - ✓ Espectro.
    - ✓ Envolvente.

3.6.2 Variables dependientes.

Dentro de las variables dependientes se encuentran:

- La programación del PIC.
- El entrenamiento del dispositivo.
- Las pruebas del dispositivo.
- Resultado esperado.

4. PRESENTACIÓN Y ANÁLISIS DE RESULTADOS

4.1 Programa.

Este es el algoritmo desarrollado para ejecución de este proyecto y así obtener el mejor resultado para el funcionamiento del dispositivo.
Este fue realizado en el programador PIC C Compiler.

```
#include "18f452.h"
#fuses HS,NOWDT,NOPROTECT,NOLVP,NOBROWNOUT
#use delay(clock=20000000)
#define EEPROM_SCL  PIN_D0
#define EEPROM_SDA  PIN_D1
#include "24256.h"
#include "define.h"
#include "math.h"

// Q = 1
/// filtro de 500 Hz
float const a0_200 = 0.000108742834*1000, a2_200 = -0.000108742834*1000;
float const b1_200 = -0.0532157331, b2_200 =0.68666962;
/// filtro de 700 Hz
float const a0_300 = 0.0000405180964*1000, a2_300 = -0.0000405180964*1000;
float const b1_300 = 1.03838009899, b2_300 = 0.6866696422;
/// filtro de 900 Hz
float const a0_600 = 0.0000023646323112*1000, a2_600 =-
0.0000023646323112*1000;
```

```
float const b1_600 = 1.644646540176, b2_600 = 0.682480657156;
float T[2];

const long lpPalabra1_200 = 0x00;
const long lpPalabra1_300 = 0x1C;
const long lpPalabra1_600 = 0x38;

const long lpPalabra2_200 = 0x54;
const long lpPalabra2_300 = 0x70;
const long lpPalabra2_600 = 0x8C;

const long lpPalabra3_200 = 0xA8;
const long lpPalabra3_300 = 0xC4;
const long lpPalabra3_600 = 0xE0;

const long lpPalabra4_200 = 0xFC;
const long lpPalabra4_300 = 0x118;
const long lpPalabra4_600 = 0x134;

const long lpPalabra5_200 = 0x150;
const long lpPalabra5_300 = 0x16C;
const long lpPalabra5_600 = 0x188;

void rotar(float);
float d(float v,float w);
float dtw(float *a , float *b);
float min(float var_a, float var_b , float var_c);
void WriteEepromVector(int addr,int *vector);
void ReadEepromVector(int addr,int *vector );
```

```
long N;
long const MUESTRAS = 896;
int buffer[896];
short rec;
float distancia[5];

void detec();
void record();
void comparador();
void param(float *matriz_200, float *matriz_300 ,float *matriz_600);

// vectores de palabras

float palabra_mem_200[7];
float palabra_mem_300[7];
float palabra_mem_600[7];

float palabra_test_200[7];
float palabra_test_300[7];
float palabra_test_600[7];

#inline
void retardo()
{
        long i;
        int j;
        for(i=0 ; i<25000 ;i++)
        {
                for(j=0 ; j<10 ;j++);
```

```
        }
}
struct
{
   boolean reconociendo;
   boolean b1;
   boolean b2;
   boolean palabra1;
   boolean palabra2;
   boolean palabra3;
   boolean palabra4;
   boolean palabra5;
}ptb;
#byte ptb = 0xF81

struct

{

   boolean entrena1;
   boolean entrena2;
   boolean entrena3;
   boolean entrena4;
   boolean entrena5;
   boolean b5;
   boolean b6;
   boolean b7;

}
```

```
ptc;
#byte ptc = 0xF82


#INT_TIMER1
void tmr1_isr()
{
      set_timer1(65229);///2kHz
      if(N < MUESTRAS)
  {
            buffer[N] = read_adc();
            N++;
      }
      else
  {
            rec = 0;
            disable_interrupts(INT_TIMER1);
      }
}
void main()

{


      int i,min;
      float  menor;


  setup_adc_ports( RA0_ANALOG);    // configuracion de conversor AD
```

```
setup_adc( ADC_CLOCK_DIV_32 );
set_adc_channel(0);



setup_timer_1 ( T1_INTERNAL | T1_DIV_BY_8 );    // config de interupcion...
enable_interrupts(GLOBAL);
enable_interrupts(INT_TIMER1);

set_tris_b(0b00000110);   // configuracion del los puertos....
set_tris_c(0b11111111);

init_ext_eeprom();       // conf de la memoria eeprom.....
for(;;)

{

   detec();     /// funcion de detección de voz en espera....
   ptb.reconociendo = 1;    /// enciende el led....
   record();              ////GRABA ....
      if (ptc.entrena1 == 0)      /// si el pin ptc.entrena está en cero hace el
entrenamiento de la primera palabra

   {

     //entrena
     param(palabra_mem_200 ,palabra_mem_300 ,palabra_mem_600);
           WriteEepromVector(lpPalabra1_200,palabra_mem_200 );
        WriteEepromVector(lpPalabra1_300,palabra_mem_300 );
```

```
        WriteEepromVector(lpPalabra1_600,palabra_mem_600 );

}

else if (ptc.entrena2 == 0)

{
  //entrena
  param(palabra_mem_200 ,palabra_mem_300 ,palabra_mem_600);
        WriteEepromVector(lpPalabra2_200,palabra_mem_200 );
      WriteEepromVector(lpPalabra2_300,palabra_mem_300 );
        WriteEepromVector(lpPalabra2_600,palabra_mem_600 );
}

else if (ptc.entrena3 == 0)

{

  //entrena
  param(palabra_mem_200 ,palabra_mem_300 ,palabra_mem_600);
        WriteEepromVector(lpPalabra3_200,palabra_mem_200 );
      WriteEepromVector(lpPalabra3_300,palabra_mem_300 );
        WriteEepromVector(lpPalabra3_600,palabra_mem_600 );
}

else if (ptc.entrena4 == 0)

{
```

```
    //entrena
    param(palabra_mem_200 ,palabra_mem_300 ,palabra_mem_600);
          WriteEepromVector(lpPalabra4_200,palabra_mem_200 );
       WriteEepromVector(lpPalabra4_300,palabra_mem_300 );
          WriteEepromVector(lpPalabra4_600,palabra_mem_600 );
     }
  else if (ptc.entrena5 == 0)
  {
    //entrena
    param(palabra_mem_200 ,palabra_mem_300 ,palabra_mem_600);
          WriteEepromVector(lpPalabra5_200,palabra_mem_200 );
       WriteEepromVector(lpPalabra5_300,palabra_mem_300 );
          WriteEepromVector(lpPalabra5_600,palabra_mem_600 );
  }
  else    /// de no estar ningun pin en del puerto c en cero
  {
    //reconoce
          param(palabra_test_200  ,palabra_test_300  ,palabra_test_600);  ///
parametriza palabra entrante
          comparador(); //// Compara la palabra con la muetras grabadas en
menoria por las funciones anteriores

   menor = distancia[0];   /// rutina de verificación de distancia mínima...
          min = 0;
          for(i=1;i<5;i++)
          {
               if(distancia[i] < menor)
               {
```

```
                menor = distancia[i];      /// revisa el uno por uno vector de resultados de la dtw
                    min = i;               /// y de termina cual es el menor
            }
        }

        if(menor < 95)                 /// dependiendo del valor de la menor distancia obtenida anteriormente
        {                          /// se van a prender los leds
            if( min == 0  )            /// ¨min¨ es una variable que dice cual de las distancias fue la menor
            {                  /// dependiendo de cual haya sido la menor
                ptb.palabra1=1;       /// encenderá el led
                retardo();           /// y después de un retardo
                ptb.palabra1=0;       /// lo apagara
            }
            if( min == 1 )
            {
            ptb.palabra2=1;
            retardo();
            ptb.palabra2=0;
            }
            if( min == 2 )
            {
            ptb.palabra3=1;
            retardo();
            ptb.palabra3=0;
            }
            if( min == 3 )
```

```
            {
            ptb.palabra4=1;
            retardo();
            ptb.palabra4=0;
            }
            if( min == 4 )
            {
            ptb.palabra5=1;
            retardo();
            ptb.palabra5=0;
            }
            }
            else
            {
                  ptb.reconociendo = 0;        /// cuando menor no cumple con el mínimo valor requerido para
                  retardo();                   /// para que la palabra en memoria tenga una similitud apreciable
                  ptb.reconociendo = 1;      /// el led de reconocimiento parpadea
                  retardo();
                  ptb.reconociendo = 0;
                  retardo();
                  ptb.reconociendo = 1;
                  retardo();
                  ptb.reconociendo = 0;
            }
      }
      ptb.reconociendo = 0;
```

```
  }
}
//// funcion que haya el valor mínimo entre dos valores....
float min(float var_a, float var_b , float var_c)
{
	if(var_a < var_b)
	{
		if(var_a < var_c)
		{
			return var_a;
		}
	}
	if( var_b < var_a )
	{
		if(var_b < var_c)
		{
			return var_b;
		}
	}
	return var_c;
}
/// funcion de alineamiento dinámico temporal
float dtw(float *a , float *b)
{
	float result[7][7];
	int i,j;

	result[0][0] = d(*(a),*(b));
	for (i=1 ; i<7 ; i++ )
```

```
        {
                result[i][0] = result[i-1][0] + d(*(a+i),*(b));
        }
        for (j=1 ; j<7 ; j++ )
        {
                result[0][j] = result[0][j-1] + d(*(a),*(b+j));
                for(i=1 ; i<7 ; i++ )
                {
                        result[i][j]    =    min(result[i-1][j], result[i][j-1], result[i-1][j-1])  +
d(*(a+i),*(b+j));
                }
        }
        return result[i-1][j-1];
}

// distancia euclidea
float d(float v,float w)
{
        float cuadrado;
        cuadrado = v - w;
        cuadrado = cuadrado * cuadrado;
        return sqrt(cuadrado);
}

// grabación
void record()
{
  //hace captura de datos
        N=0;  /// variable que lleva en conteo de las muestras
```

```
        rec = 1;
        enable_interrupts(INT_TIMER1);   /// funcion que habilita las interuupciones
        set_timer1(65229);               /// se establece el valor del contador timmer1
para que el tiempo de la interrupción sea de 0.5ms
        while(rec == 1);                 /// espera hasta que la grabación se completada
}
// detector de voz
void detec()
{
   int tmp;
    tmp = read_adc();
    while( tmp > 100 && tmp < 154 )     /// espera hasta que el nivel de la voz
sobrepase un rango especificado
    {
        tmp = read_adc();            //// lee del conversor análogo digital...
    }
}

// paramerizador
void param(float *matriz_200, float *matriz_300 ,float *matriz_600)
{
   float Xz,Yz,Pm,sum,Xn,P;
   long i,j,pocicion;

   pocicion=0;
   Xn=0;Yz=0;sum=0;P=0;Pm=0;T[0]=0;T[1]=0;
   for(i=0;i<MUESTRAS;i++)
   {
        ptb.reconociendo = !ptb.reconociendo ;
```

```
  for(j=0;j<128;j++)
  {
    /// filtrado Digital para filtro de 200
    Xz=(float)buffer[i+j] - (float)127;
        Pm = Xz - T[0]*b1_200 - T[1]*b2_200;
        Yz = Pm*a0_200 + T[1]*a2_200;
    Rotar(Pm);
    /// la respuesta queda en Yz
    /// Calculo de potencia por muestra...
        ///*************************
        ///calculo de la potencia promedio
        ///                      N-1
        /// P=(1/N)( Sum  X²[n]) ;para una señal discreta
        ///                      n=0
            Xn=Yz;                      //Entrada para calculo
            Xn=Xn*Xn;
            sum=Xn+sum;
  }
  P = sum/128;  /// obtiene potencia promedió del vector de 128 muestras
  *(matriz_200 + posición++) = P;
     sum = 0;
  P = 0;
  i=i+j-1;
}

pocicion=0;
Xn=0;Yz=0;sum=0;P=0;Pm=0;T[0]=0;T[1]=0;
for(i=0;i<MUESTRAS;i++)
```

```
{
    ptb.reconociendo = !ptb.reconociendo ;

  for(j=0;j<128;j++)
  {
    /// filtrado Digital para filtro de 300
    Xz=(float)buffer[i+j] - (float)127;
        Pm = Xz - T[0]*b1_300 - T[1]*b2_300;
        Yz = Pm*a0_300 + T[1]*a2_300;
    Rotar(Pm);
    /// la respuesta queda en Yz
    /// Calculo de potencia por muestra...
        ///**************************
        ///calculo de la potencia promedio
        ///                        N-1
        /// P=(1/N)( Sum  X²[n]) ;para una señal discreta
        ///                         n=0
            Xn=Yz;                      //Entrada para calculo
            Xn=Xn*Xn;
            sum=Xn+sum;
  }

  P = sum/128;  /// obtiene potencia promedio del vector de 128 muestras
  *(matriz_300 + posición++) = P;
  sum = 0;
  P = 0;
  i=i+j-1;
}
```

```
pocicion=0;
Xn=0;Yz=0;sum=0;P=0;Pm=0;T[0]=0;T[1]=0;
for(i=0;i<MUESTRAS;i++)

{
    ptb.reconociendo = !ptb.reconociendo ;

  for(j=0;j<128;j++)

  {
    /// filtrado Digital para filtro de 600
    Xz=(float)buffer[i+j] - (float)127;
        Pm = Xz - T[0]*b1_600 - T[1]*b2_600;
        Yz = Pm*a0_600 + T[1]*a2_600;
    Rotar(Pm);          /// la respuesta queda en Yz

    /// Calculo de potencia por muestra...
        ///*************************
        ///calculo de la potencia promedio
        ///                        N-1
        /// P=(1/N)( Sum  X²[n]) ;para una señal discreta
        ///                          n=0
            Xn=Yz;                      //Entrada para calculo
            Xn=Xn*Xn;
            sum=Xn+sum;
  }
  P = sum/128;  /// obtiene potencia promedio del vector de 128 muestras
  *(matriz_600 + posición++) = P;
  P = 0;
```

```
      sum = 0;
      i=i+j-1;
    }
}

/// hace las rotaciones de memoria para el filtro digital....
void rotar(float Dato)
{
        T[1]=T[0];
        T[0]=Dato;
}

///  funcion que escribe en la eeprom i2c
void WriteEepromVector(int addr,int *vector){
        int i;
        for(i=0;i<28;i++)
                write_ext_eeprom((addr+i), *(vector+i));
                //write_eeprom((addr+i), *(vector+i));
}


///funcion que lee de la eeprom i2c
void ReadEepromVector(int addr,int *vector ) {
   int i;
   for (i = 0;i<28;i++)
         *(vector + i) = read_ext_eeprom(i + addr);
     //*(vector + i) = read_eeprom(i + addr);
}
```

```
/// compara las palabras grabadas en la memoria eeprom con las que se hablo en el micrófono
void comparador()
{
            /// compara palabra 1
           ReadEepromVector(lpPalabra1_200,palabra_mem_200 );  /// lee una palabra de la memoria
         ReadEepromVector(lpPalabra1_300,palabra_mem_300 );
            ReadEepromVector(lpPalabra1_600,palabra_mem_600 );

            distancia[0] = dtw(palabra_mem_200 , palabra_test_200 );   /// haya la distancia de la palabra anterior con la que se hablo en el micrófono
              distancia[0] = dtw(palabra_mem_300 , palabra_test_300 ) + distancia[0];
              distancia[0] = dtw(palabra_mem_600 , palabra_test_600 ) + distancia[0];

            distancia[0] = distancia[0] / 3;


            /// compara palabra 2
         ReadEepromVector(lpPalabra2_200,palabra_mem_200 );
         ReadEepromVector(lpPalabra2_300,palabra_mem_300 );
            ReadEepromVector(lpPalabra2_600,palabra_mem_600 );

            distancia[1] = dtw(palabra_mem_200 , palabra_test_200 );
              distancia[1] = dtw(palabra_mem_300 , palabra_test_300 ) + distancia[1];
```

```
            distancia[1] = dtw(palabra_mem_600 , palabra_test_600 ) +
distancia[1];
        distancia[1] = distancia[1] / 3;


           /// compara palabra 3
        ReadEepromVector(lpPalabra3_200,palabra_mem_200 );
        ReadEepromVector(lpPalabra3_300,palabra_mem_300 );
           ReadEepromVector(lpPalabra3_600,palabra_mem_600 );

           distancia[2] = dtw(palabra_mem_200 , palabra_test_200 );
            distancia[2] = dtw(palabra_mem_300 , palabra_test_300 ) +
distancia[2];
            distancia[2] = dtw(palabra_mem_600 , palabra_test_600 ) +
distancia[2];
        distancia[2] = distancia[2] / 3;


           /// compara palabra 4
        ReadEepromVector(lpPalabra4_200,palabra_mem_200 );
        ReadEepromVector(lpPalabra4_300,palabra_mem_300 );
           ReadEepromVector(lpPalabra4_600,palabra_mem_600 );

           distancia[3] = dtw(palabra_mem_200 , palabra_test_200 );
            distancia[3] = dtw(palabra_mem_300 , palabra_test_300 ) +
distancia[3];
            distancia[3] = dtw(palabra_mem_600 , palabra_test_600 ) +
distancia[3];
        distancia[3] = distancia[3] / 3;
```

```
            /// compara palabra 5
        ReadEepromVector(lpPalabra5_200,palabra_mem_200 );
        ReadEepromVector(lpPalabra5_300,palabra_mem_300 );
            ReadEepromVector(lpPalabra5_600,palabra_mem_600 );

            distancia[4] = dtw(palabra_mem_200 , palabra_test_200 );
            distancia[4] = dtw(palabra_mem_300 , palabra_test_300 ) +
distancia[4];
              distancia[4] = dtw(palabra_mem_600 , palabra_test_600 ) +
distancia[4];
        distancia[4] = distancia[4] / 3;
}
```

```
// Q = 1
/// filtro de 200 s.f.
//float const a0_200 = 0.310571741, a2_200 = -0.310571741;
//float const b1_200 = -0.598304596, b2_200 =0.378856517;
/// filtro de 300 Hz
//float const a0_300 = 0.332943705, a2_300 = -0.332943705;
//float const b1_300 = -0.078944387, b2_300 = 0.334112588;
/// filtro de 600 Hz
//float const a0_600 = 0.292785295, a2_600 = -0.292785295;
//float const b1_600 = 0.793119714, b2_600 = 0.414429408;
```

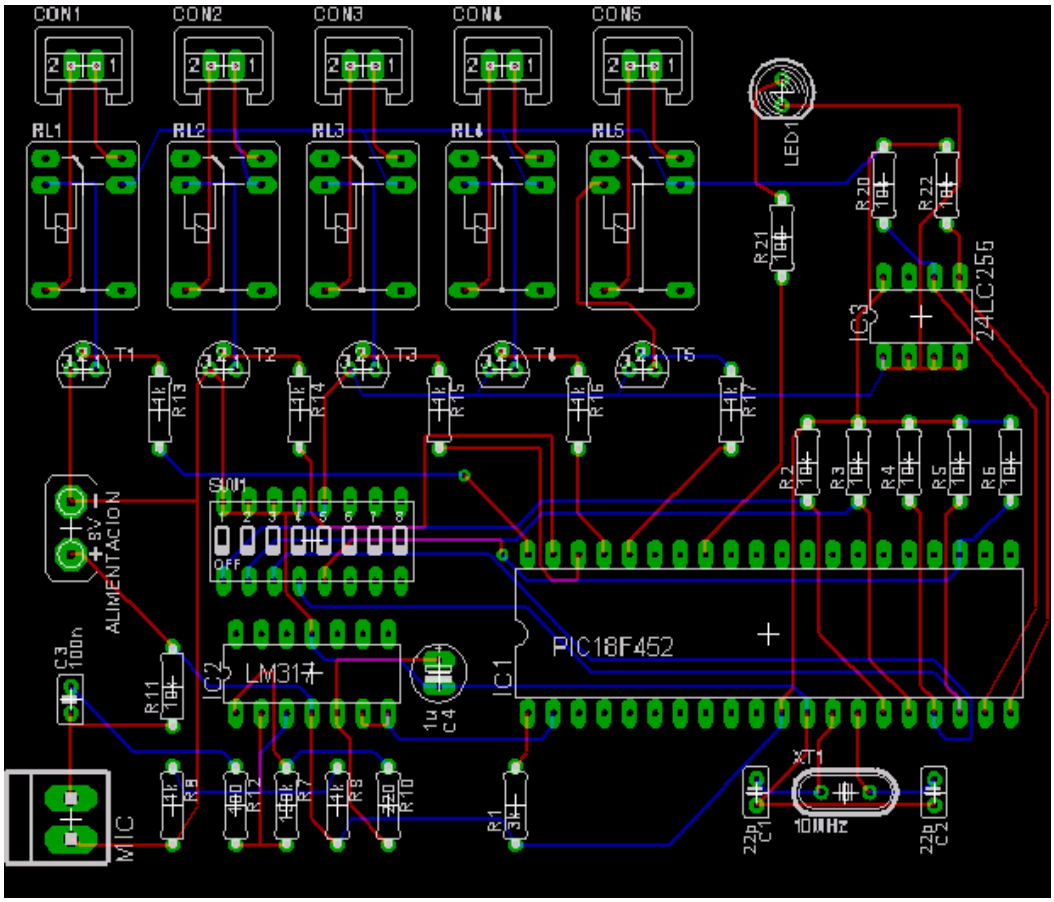
4.2 Diseño de la placa.

Para el diseño del hardware conectaremos al microcontrolador todo lo que necesita para funcionar y lo que le permite interaccionar con el mundo exterior.

4.2.1 Esquemático.

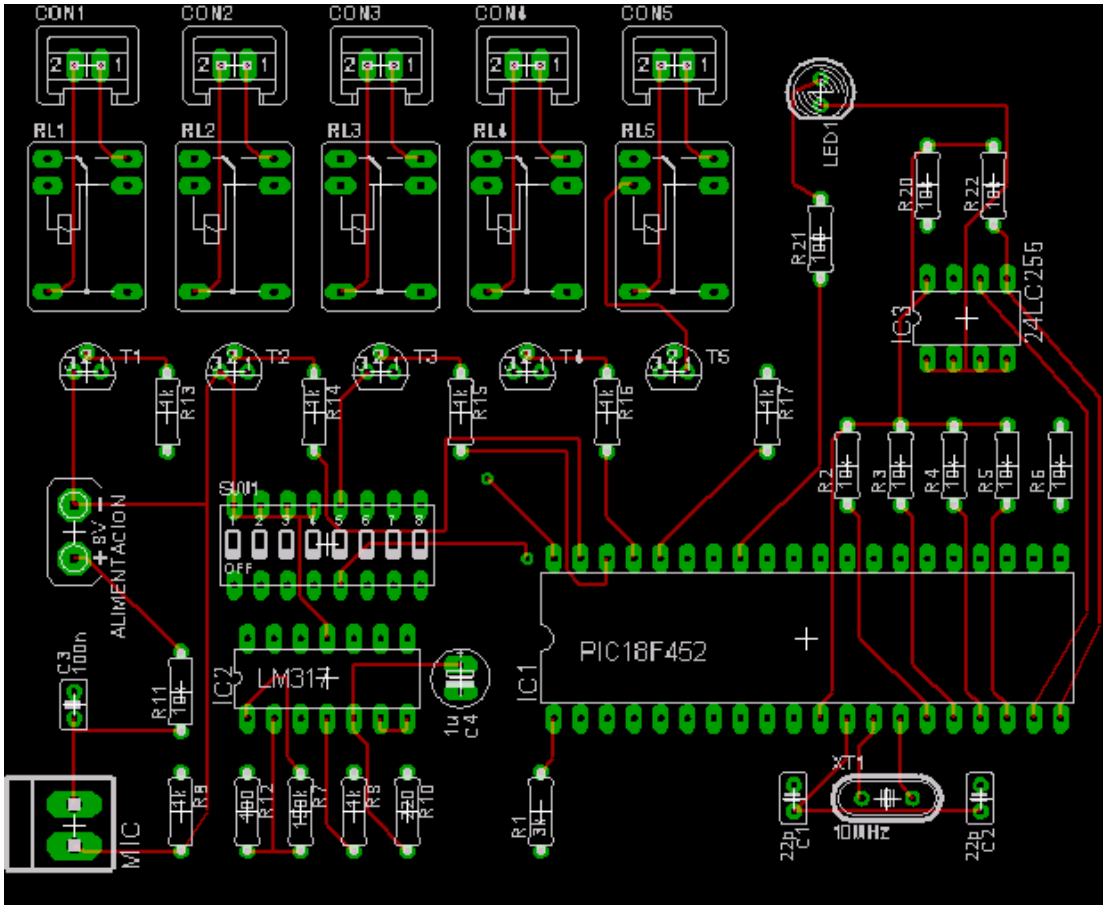
El esquemático fue realizado con ayuda de EAGLE Layout Editor 4.03.
En las figuras siguientes podemos ver el diseño completo en PCB de la placa las respectivas caras (TOP/BOTTOM) (Figura. 39, Figura. 40, Figura 41)

Figura 39. Diseño PCB.



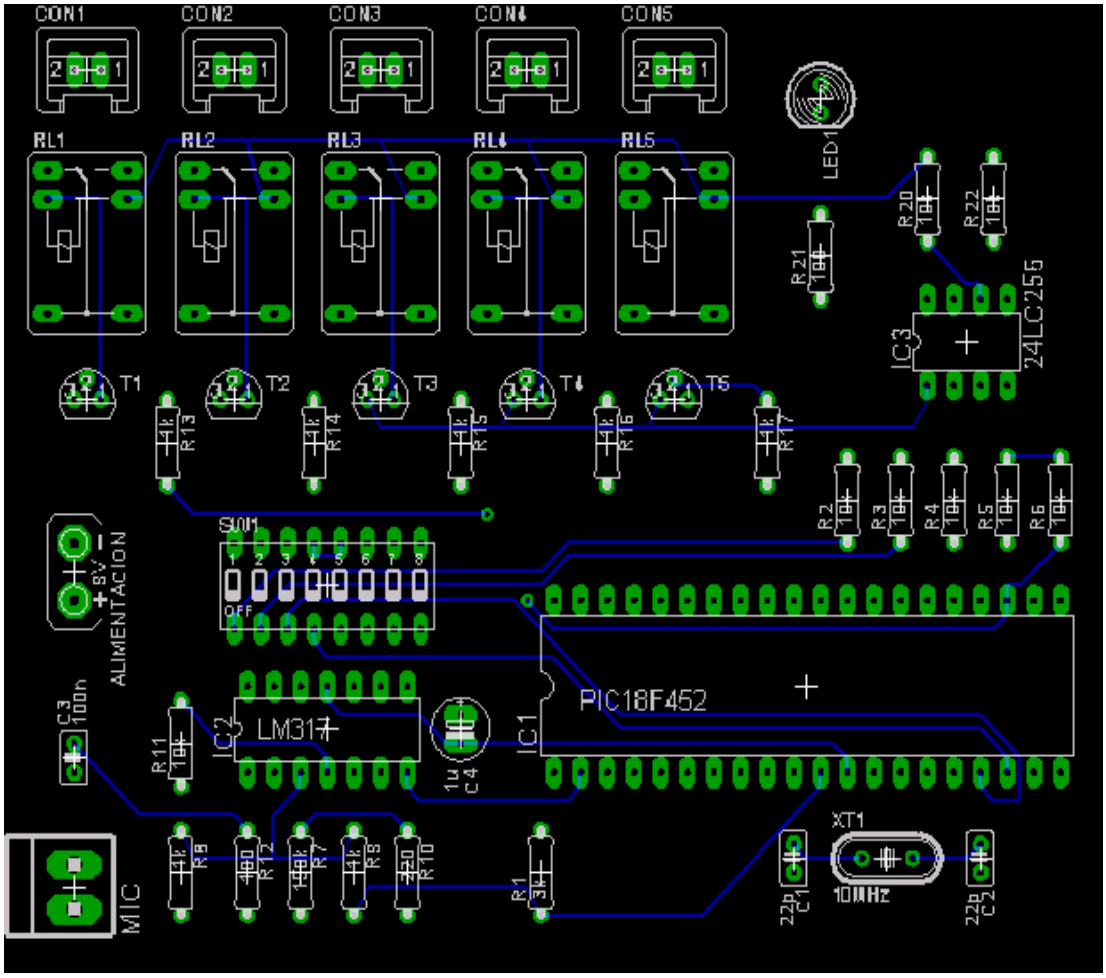
Fuente: EAGLE Layout Editor 4.03.

Figura 40. Diseño PCB (TOP).



Fuente: EAGLE Layout Editor 4.03.

Figura 41. Diseño PCB (BOTTOM)



Fuente: EAGLE Layout Editor 4.03.

4.3 Resultados.

Tabla 3. Resultado de pruebas.

PALABRA	EXACTITUD
On	20 de 20
Subir	18 de 20
Bajar	18 de 20
Aumentar	19 de 20
Disminuir	18 de 20
Total	**93 de 100**

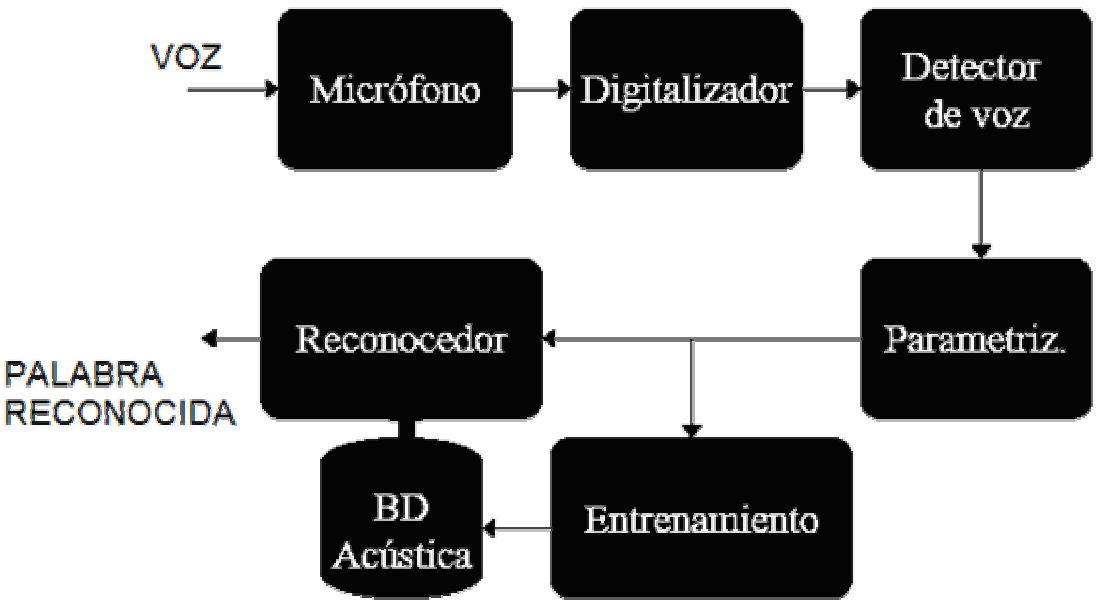
PALABRA	EXACTITUD
Power	19 de 20
Subir	18 de 20
Bajar	18 de 20
Derecha	18 de 20
Izquierda	19 de 20
Total	**92 de 100**

La Tabla 3. Fue obtenida usando una sola persona; la respuesta del sistema de reconocimiento, arrojó resultados del 92% al 93%, de exactitud.

5. DESARROLLO INGENIERIL.

5.1 Esquema General del Sistema de Reconocimiento Automático del Habla (RAH).

Figura 42. Esquema General de R.A.H. Implementado.

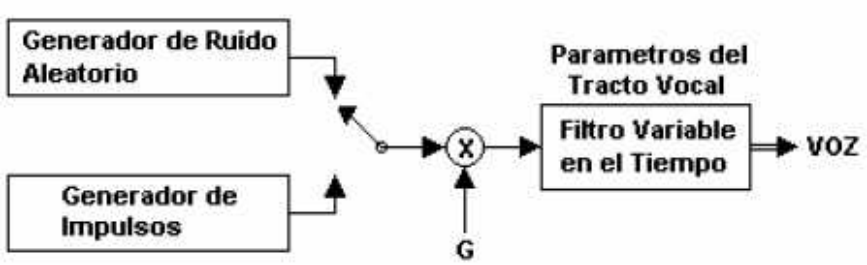


5.1.1 Modelo de Producción de la Voz. (MODELO SOURCE-FILTER)

El modelo clásico del tracto vocal se compone de un filtro variable en el tiempo, un generador de ruido aleatorio y de un generador de impulsos. Los parámetros del filtro varían en función de la acción consciente que se realiza al pronunciar una palabra.

El modelo tiene dos entradas, que dependen del tipo de señal. Para señales sonoras (vocales) la excitación es un tren de impulsos de frecuencia controlada, mientras que para las señales no sonoras (consonantes) la excitación es ruido aleatorio. La combinación de estas dos señales modelan el funcionamiento de la glotis. Es importante destacar que la mayor parte de la información del locutor esta en las cuerdas vocales (o excitación), mientras que la información de la palabra pronunciada está en la característica del filtro.

Figura 43. Modelo de producción de voz.



Fuente: http://laenciclopedialibre/La voz humana.htm. 11/10/2005, 16:44.

El espectro en frecuencias de la señal vocal se obtiene como el producto de la transformada de Fourier (FT) de la excitación por la respuesta en frecuencia del filtro, es decir:

$$S(w) = E(w) \bullet H(w)$$

Donde:

E: FT de la excitación.

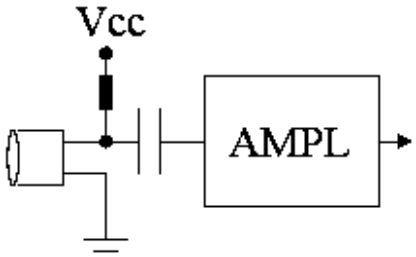
H: FT del Filtro.

S: FT de la voz.

5.1.2 Micrófono.

Es la entrada de señal del sistema. El transductor fue modelado como una resistencia variable. Dada la alimentación de 5V se eligió una resistencia lo más próxima posible a la del micrófono para así centrar la salida en ausencia de señal entorno a los 2.5V, buscando así maximizar su respuesta. A continuación se colocó un condensador, facilitando el trabajo a etapas posteriores, asegurando un tamaño lo suficientemente grande como para que se distorsionasen lo mínimo posible las bajas frecuencias.

Figura 44. Conexión del Micrófono.



La señal de voz básicamente está constituida por ondas de presión producidas por el aparato fonador humano. La manera obvia de capturar este tipo de señal se realiza mediante un micrófono, el cual se encargará de convertir la onda de presión sonora en una señal eléctrica.

La siguiente etapa será aquella que se encargue de amplificar las señales a niveles que sean manejables. A partir de la señal analógica obtenida se hace necesario convertir la señal a formato digital para poder procesarla en la computadora la cual se realiza mediante dos procesos: muestreo y cuantificación. Este proceso de dos etapas se conoce como Modulación por Código de Pulsos (PCM). La señal vocal tiene componentes frecuenciales que pueden llegar a los 10 khz., sin embargo la mayor parte de los sonidos vocales tiene energía espectral

significativa hasta los 5khz. Solamente los sonidos fricativos poseen componentes que pueden llegar a los 10 khz.

La frecuencia de muestreo dependerá del tipo de aplicación, para señales de voz se adopta un rango de 6 a 20 khz. Dependiendo de la resolución que se desee. Otra consideración que se debe tener en cuenta es la cuantificación de la señal, la cual involucra la conversión de la amplitud de los valores muestreados a forma digital usando un número determinado de bits. El número de bits usados afectará la calidad de la voz muestreada y determinará la cantidad de información a almacenar. Para cada instante de muestreo, el convertidor analógico digital compara la señal muestreada con una serie de niveles de cuantificación predefinidos. El número de niveles N a usar determina la precisión del análisis y por tanto el número de bits necesarios. Cada bit adicional que se agrega contribuye en mejorar la relación señal a ruido en aproximadamente 6 dB. La señal de voz exhibe un rango dinámico de unos 50 a 60 dB. por lo que resultaría suficiente una cuantificación de 8 a 9 bits para una buena calidad de voz. Sin embargo generalmente se usa de 11 a 20 bits en aplicaciones de procesamiento de señales de voz de alta calidad.

5.1.2.1 Amplificación.

Se escogió el LM324 como diseño de amplificador inversor, como el de la figura para lograr una respuesta de 5V pico a pico, máxima diferencia de amplitud reconocible por el conversor analógico-digital para su alimentación. La ganancia se escogió en base a medidas empíricas sobre la salida del micrófono y teniendo en cuenta los efectos de carga, y fue refinada para producir una respuesta que no se saturara al hablar al micrófono con una distancia y tono adecuados, por lo que se eligió finalmente una ganancia de G = -R1/R2 = -100.

Figura 45. Amplificador Inversor. LM324.

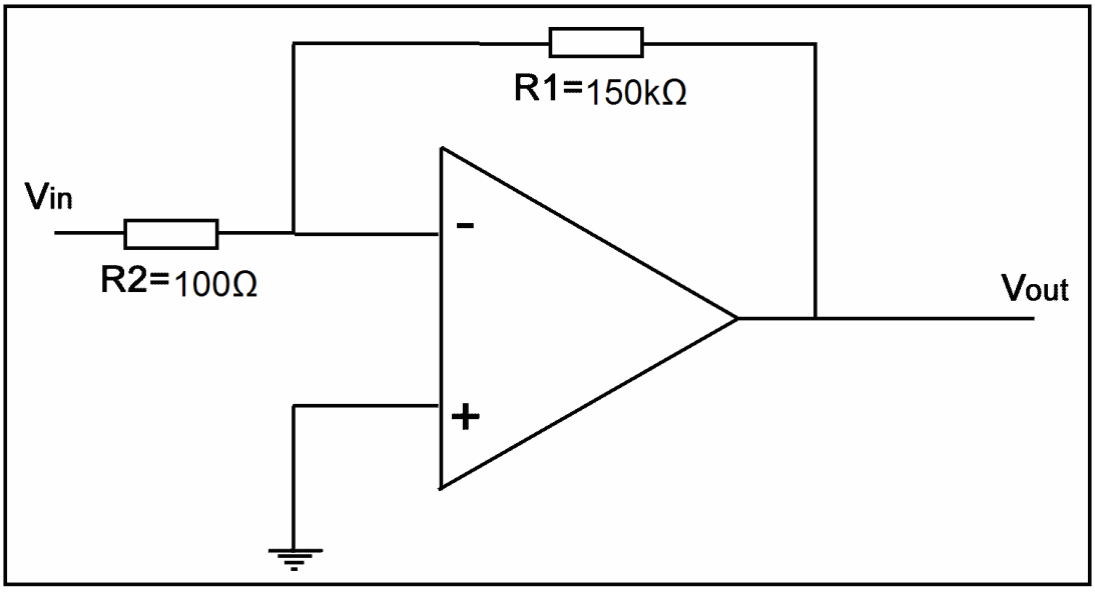


Figura 46. Diagrama de Conexión del LM324.

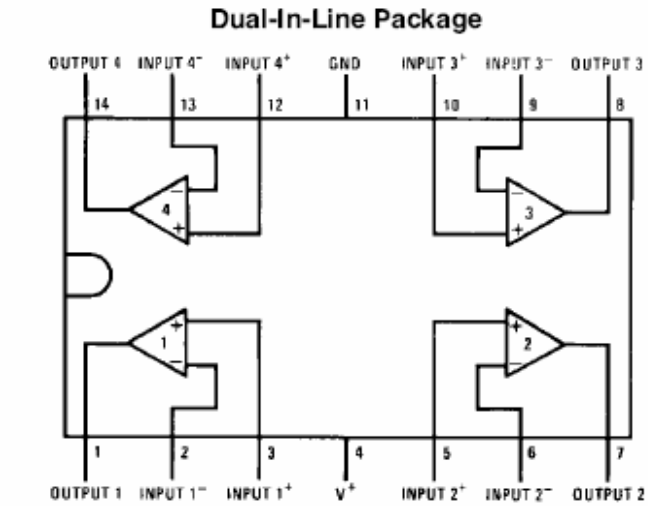
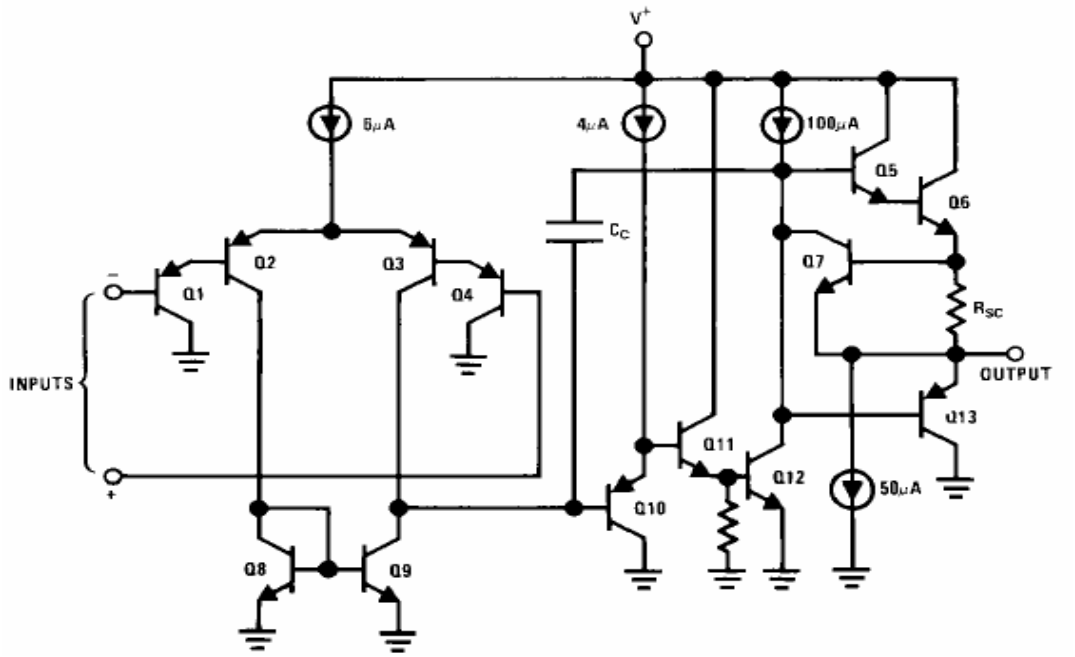


Figura 47. Diagrama esquemático del LM324.



Fuente. Hoja de especificaciones, datasheet LM324.

5.1.3 Digitalizador.

Está etapa adapta el valor medio nulo de la señal de entrada a los 2.5V que centran la señal respecto al margen de operación del conversor analógico digital, dando una señal que varía entre 0 y 5V. Se aprovecha un operacional en configuración de sumador inversor que atenúa a la mitad la continua sin modificar la amplitud de la señal. Para conseguir un margen dinámico adecuado a nuestro módulo de muestreo, y así poder aprovechar toda su precisión.

5.1.3.1 Conversión analógica-digital.

Es el módulo encargado de muestrear la señal de voz para su entrada al sistema. El conversor analógico-digital usado tiene una salida de 8 bits, está alimentado para capturar linealmente entre 0V y 5V y lo hace al ritmo que le dicta la placa entrenadora.

5.1.4 Detector de Voz.

5.1.4.1 Grabación de Voz.

El propósito de hacer una grabación de voz es almacenar una señal, en tiempo real, para luego poder tratarla; lo que un procesador de baja potencia no puede realizar por sí solo durante el proceso de captura. Por está razón habrá que realizar la grabación teniendo muy en cuenta los requisitos del procesamiento.

Sin embargo, considerar la grabación solo como un paso intermedio al procesamiento de voz sería trivializar su función, ya que es también suya la tarea de delimitar el comienzo y el fin de una grabación, etapa crucial en el proceso de reconocimiento ya que su fallo puede provocar carencia de información si no se graba lo suficiente, y largos tiempos de procesamiento o si se excede por mucho la duración de la señal.

El procesamiento de voz requiere dividir la señal en periodos donde las propiedades sean más o menos constantes para así poder caracterizar frecuencialmente dichas zonas. A dichos intervalos los llamaremos tramas y serán la unidad fundamental a la hora de tratar la señal. Experimentalmente se puede establecer que es correcto el tratado de la voz si se toman tramas de 64 ms. de duración y que es permisible extender dicha suposición a 78 ms. periodo que se barre tomando 128 muestras a una frecuencia de muestreo de 2 KHz. Dado que el

primero es un número fácilmente manejable para un microcontrolador y a que, por limitaciones del PIC, una frecuencia de muestreo tal como la segunda se acerca al límite de lo permisible éstos serán los valores de los que hará uso la grabación.

Por las limitaciones del procesador, las operaciones que se hagan en tiempo real han de ser sencillas, y por tanto no are una estimación de la energía para hacer las caracterizaciones. Desde este punto de vista, verificar si hay voz ha de reducirse a comprobar que la señal se mantenga por encima de un nivel de `voz presente´ un tiempo suficientemente largo como para diferenciar una subida espontánea en el tono de un comando intencionado.

Al igual que hay que distinguir una subida espontánea dentro de un intervalo indefinido de silencio, es necesario distinguir pausas dentro de un mismo comando. Para ello se puede hacer al contrario que en el caso opuesto: una vez sepamos que hay voz, vigilaremos que la señal no baje por debajo del umbral de `voz presente´, y si lo hace consideraremos que el comando puede haber terminado, pero no daremos dicha suposición como válida hasta que no haya pasado un tiempo mínimo.

Además de las dos anteriores consideraciones, hay una tercera a tener en cuenta: la cadencia del habla. Analizando la acentuación de las frases, es común que el comienzo tenga una lenta subida de volumen antes de llegar al tono natural. Los dos parámetros anteriores no tienen en cuenta esta situación, con lo que se puede perder valiosa información del comienzo de las frases. Por ello es conveniente considerar un segundo umbral de volumen, menor que el primero y que detecte posible presencia de voz. Cada vez que este umbral se supera se empezará a considerar la señal como posible voz, y sólo se descartará si se baja de dicho umbral sin que haya sido considerada como tal.

El razonamiento anterior lo aplicare al final de las palabras, que terminarán en aquel punto en el que la señal disminuya por debajo del umbral bajo.
Una vez establecidos los criterios de grabación sólo hace falta implementarlos sobre el hardware disponible. En este caso el tratamiento se hará desde una rutina de servicio a la que se accederá periódicamente. Para adaptarse al requisito del conversor A/D de recibir una señal, para prepararse para capturar y una segunda para realizar la captura, la frecuencia de proceso de está subrutina ha de ser el doble de la frecuencia de muestreo deseada, es decir, 4 KHz. Olvidando el aspecto secundario acerca de cómo llevar a cabo una llamada periódica a la subrutina, pasemos a describir su implementación.

Todo el proceso de detección y grabación de voz se lleva a cabo desde la rutina de servicio. Debido a la poca capacidad de procesamiento disponible durante el proceso de grabación de voz se han minimizado las subrutinas en la rutina de servicio, y se ha optimizado el uso de registros guardando en ellos las variables de acceso frecuente.

Tras un periodo de funcionamiento extraordinario del sistema en el que únicamente se dedica a tomar la referencia de ruido, la rutina de servicio entra en su funcionamiento normal. En este estado, podríamos distinguir dos partes diferenciadas en está rutina. La primera de ellas se encarga del muestreo de la señal a través del conversor A/D del PIC, a través de la división de todas las interrupciones en pares e impares, así como de la grabación y almacenamiento de muestras en memoria cuando esto sea necesario. La segunda parte de la rutina es la encargada de realizar el procesamiento cuando se ha terminado de capturar una trama completa.

Lo primero que se debe hacer es estimar en qué nivel relativo respecto a los dos umbrales de referencia se encuentra la energía de la trama. Este cálculo se

realiza en todos los casos, ya que es imprescindible para cualquier tratamiento. Tras esto, la rutina se bifurca dependiendo de si ya hay una presencia de voz confirmada o no.

En caso de la ausencia de voz confirmada, se salta a NO_VOZ (led apagado), estado en el que la tarea del sistema es esperar la presencia de voz, manteniendo el buffer de grabación vacío hasta que sea necesario grabar, y determinar si una posible presencia de voz lo es en efecto o no.

En el caso de que la presencia de voz ya haya sido confirmada, se deben grabar las muestras de voz en el buffer de grabación, y esperar una caída en el nivel de señal, es decir, esperar y comprobar el final de una grabación (led intermitente). Esto es lo que realiza la parte de la rutina a partir de VOZ.

5.1.4.2 Procesamiento de Voz.

El procesado de la señal, es el momento en el que se extrae la información útil de la señal, consiguiendo así una caracterización compacta que ahorra espacio de almacenamiento y facilita el tratamiento para la aplicación para la que está enfocado.

Al detallar el proceso de grabación se explicó cómo se iban a tratar la señal: como bloques de muestras llamados tramas que supondremos, frecuencialmente estacionarias. La forma en la que vamos a caracterizar la señal va a ser mediante un filtrado digital en distintas bandas frecuenciales. De está forma obtendremos tantas señales como filtros, con lo que el objetivo de ahorrar espacio se pierde y además se obtienen demasiados datos como para ser tratados mediante un

procesador de baja potencia. Por ello lo que se hará será calcular la energía resultante de cada filtrado, obteniendo por tantos valores de energía como filtros se usen.

Para este sistema se han escogido filtros de segundo orden de Infinita Respuesta caracterizables a partir de cuatro coeficientes A0, A2,B1, y B2; Así pues, el valor de una muestra procesada viene dado por

$$y[n] = \left[x[n] - B_1 \cdot y[n-1] - B_2 \cdot y[n-2]\right] \cdot A_0 + A_2\left[y\left[n-1\right]\right]$$

Siendo x[n] la señal a procesar e y[n] el resultado de filtrar.
El trabajo descrito anteriormente lo lleva a cabo la función Voz procesar, cuyo funcionamiento viene detallado a continuación.

Voz Procesar

Representa una señal a base de vectores de energía. Cada n elementos (de ahora en adelante cada trama) se le hace una serie de filtrados de los mismos a partir de 3 filtros de segundo orden y de ganancia a lo sumo la unidad, calculándose las energías de los resultados y guardándolas consecutivamente en estructuras de N palabras. La señal ha de estar compuesta de n*L bytes con signo correspondientes a muestras periódicas de una señal, dando lugar a n*N filtros de energía. Cada filtro está caracterizado mediante tres coeficientes A, B y C almacenados en palabras con signo contiguos que darán lugar a una señal filtrada mediante la fórmula:

$$y[n] = \left[x[n] - B_1 \cdot y[n-1] - B_2 \cdot y[n-2]\right] \cdot A_0 + A_2\left[y\left[n-1\right]\right]$$

Los tríos de coeficientes han de estar contiguos los unos a los otros.

Implementación: El algoritmo realiza los cálculos filtro por filtro. Al comienzo de cada uno, el puntero al destino se modifica para que apunte a la componente que corresponde, lo cual asegura que se pueda acceder fácilmente al resto de componentes, ya que el array de vectores de energía asegura que la distancia entre componentes equivalentes sea la misma. Una vez hecho esto hay que asegurar que los valores de y[n-1] e y[n-2] sean nulos, así como el de la energía parcial calculada. Una vez hecho esto se van cargando los valores de la señal y calculando las componentes filtradas y añadiéndolas a la energía (esto se hace simplemente sumando el valor absoluto de la componente calculada). Una comprobación asegura que una vez pasadas n tramas se guarde la energía parcial en su posición final y se prepare todo para la siguiente ronda: hay que poner la energía de trama a cero, mover el puntero de destino a su siguiente posición y asegurar la condición que avise del final de trama una vez se hayan procesado n muestras más. Adicionalmente hay que comprobar si se ha llegado al final de la señal, en cuyo caso hay que prepararlo todo para el siguiente filtro o salir.

Entradas: Hay que pasar los parámetros por pila en el siguiente orden: El número de elementos por trama como una palabra sin signo; el número de filtros como una palabra sin signo; el número de tramas de la señal como una palabra sin signo; un puntero al comienzo de los coeficientes como un long; un puntero a una zona de memoria par con n*3N palabras libres; y un puntero tamaño long al primer byte de la señal.

Salidas: A partir del puntero a la zona de memoria libre se habrán almacenado los vectores de energía de forma contigua, estando cada vector de energía constituido por las componentes de energía resultantes de los distintos filtrados, tal y como estaban ordenados los coeficientes de los filtros.

5.1.5 Parametrizador.

El objetivo de este módulo de parametrización es representar la señal de habla de la que partimos, previamente muestreada, mediante unos parámetros que contengan la información más relevante del mensaje comprendido en la onda acústica, y que eliminen la redundancia.
La señal de habla presenta una variación en el tiempo, causada por el movimiento de los órganos articulatorios, que es mucho más lenta que la frecuencia de análisis que se utiliza en los sistemas de proceso de habla. Debido a ello, puede considerarse que el proceso de articulación es continuo, con tendencias de estacionariedad en los núcleos de los sonidos.

Está lentitud en la variación es la que nos permite analizar la señal en tramas de duración corta y con solapamiento entre ellas.
El análisis localizado del habla se puede hacer en el dominio del tiempo o en el de la frecuencia.

5.1.5.1 Diseño y calculo de Filtros.

Los filtros desarrollados para la implementación del Reconocimiento Automático del Habla (RAH), en este proyecto son de Infinita Respuesta al Impulso (IIR) de 200Hz, 300Hz, y 600Hz; Explicados anterior mente en el enunciado 2.2.3.1.2.2.

$$H(s) = \frac{sw_n}{s^2 + sw_n / Q + w_n^2} \quad , \qquad \text{Q = 1(Factor de calidad)}$$

T = 1×10^{-3} (Tiempo de Respuesta)

$W_n = 2\pi f$ (Frecuencia De Resonancia)

$$s = \frac{2(z-1)}{T(z+1)}$$

$$H(z) = \frac{\left(\frac{2(z-1)}{T(z+1)}\right)w_n}{\left(\frac{2(z-1)}{T(z+1)}\right)^2 + \left(\frac{2(z-1)}{T(z+1)}\right)w_n + w_n^2}$$

$$H(z) = \frac{2(z-1)w_n}{\frac{\left[\left[2(z-1)\right]^2 + 2(z-1)\cdot\left[T(z+1)\right]w_n + w_n^2\left[T(z+1)^2\right]^2\right]\cdot T(z+1)}{\left[T(z+1)\right]^2}}$$

$$H(z) = \frac{2w_n(z-1)}{\frac{\left[\left[4(z^2-2z+1)\right] + 2Tw_n(z^2-1) + w_n^2T^2(z^2+2z+1)\right]}{T(z+1)}}$$

$$H(z) = \frac{2w_n(z-1)}{\frac{\left[4z^2-8z+4+2Tw_nz^2-2Tw_n+T^2{w_n}^2z^2+2T^2{w_n}^2z+w_n^2T^2\right]}{T(z+1)}}$$

$$H(z) = \frac{2w_n(z-1)\cdot T(z+1)}{(4+2Tw_n+T^2{w_n}^2)z^2+(-8+2T^2w_n^2)z+(4-2Tw_n+w_n^2T^2)}$$

$$H(z) = \frac{2w_nT(z^2-1)}{(T^2{w_n}^2+2Tw_n+4)z^2+(2T^2w_n^2-8)z+(w_n^2T^2-2Tw_n+4)}$$

1) $W_n = 2\pi 200$, $T_s = 1\times10^{-3}$, $W_n t = 1{,}2566370611$

$$H(z) = \frac{2{,}513274123(z^2-1)}{8{,}092410827z^2 - 4{,}841726592z + 3{,}06586258}$$

$$H(z) = \frac{0{,}310571741(z^2-1)}{z^2 - 0{,}598304596z + 0{,}378856517}$$

2) $W_n = 2\pi 300$, $T_s = 1\times10^{-3}$, $W_n t = 1{,}884955592$

$$H(z) = \frac{3{,}769911184(z^2-1)}{11{,}32246877z^2 - 0{,}893884832z + 3{,}7831464}$$

$$H(z) = \frac{0{,}332943705(z^2-1)}{z^2 - 0{,}078944387z + 0{,}334112588}$$

3) $W_n = 2\pi 600$, $T_s = 1\times10^{-3}$, $W_n t = 3{,}769911184$

$$H(z) = \frac{7{,}539822369(z^2-1)}{25{,}75205271z^2 - 20{,}42446068z + 10{,}67240797}$$

$$H(z) = \frac{0{,}292785295(z^2-1)}{z^2 - 0{,}793119711z + 0{,}414429408}$$

5.1.6 Entrenamiento.

Es la grabación de las palabras a reconocer, en la memoria I^2C **.** Está grabación se realiza abriendo el interruptor en el dispositivo, se pronuncia la palabra, y nuevamente cerrando el interruptor, mostrando el dispositivo una señal de reconocimiento y grabación (led enciende, durante la grabación y luego se apaga).

5.1.7 Reconocedor.

Ésta es la fase crucial para el reconocimiento de voz ya que es el proceso de elección. El procedimiento para establecer la distancia entre dos señales empleado se conoce como alineamiento temporal dinámico (DTW).

DTW

Estima la proximidad de dos señales. Para ello calcula la diferencia de energía entre dos secuencias (de tamaño arbitrario pero conocido) de vectores aplicando el algoritmo de alineamiento temporal dinámico. Descrito anteriormente en el apartado 2.1.2.

Para ello se requiere que las señales estén caracterizadas energéticamente a las frecuencias de interés. El algoritmo supondrá que se han caracterizado estudiando la señal a intervalos fijos o tramas, periodos durante los que la señal se pueda considerar estacionaria (respuesta frecuencial estable). Asimismo se considerará que cada trama se ha caracterizado en bandas frecuenciales (constante a definir en tiempo de compilación), cada una definida por una energía almacenadas en un

numero punto flotante. De está forma, una señal de n tramas estará definida por n vectores de energía consecutivos que ocuparán n números cada uno.

El algoritmo implementado supone que una señal puede estudiarse como secuencias de sonidos donde lo que más importa no es la duración de cada secuencia, sino el orden que siguen. Este criterio resulta apropiado para hacer comparaciones entre grabaciones de voz sin necesidad de hacer un análisis a nivel de fonemas ya que tiene en cuenta la variabilidad de la cadencia en el habla según el momento y la constancia en el orden de los sonidos que permite distinguir palabras.

Implementación.

Se basa en el recorrido de la matriz tal y como lo describe el apartado 2.2.4 arriba citado. Se basa en que el valor de cada nodo de la matriz viene dado por la energía diferencial de los vectores de energía que definen la posición del nodo más un valor elegido como el mínimo de las celda contiguas inferior e izquierda. Las excepciones las forman los nodos de la fila inferior, donde el valor que se suma sólo puede ser el del elemento izquierdo, los nodos de la columna izquierda, que sólo reciben aportación energética del elemento inferior, y el nodo inferior izquierdo (el nodo de partida), al que no se le suma ningún valor. Sin embargo, es posible tratar estos casos excepcionales desde el punto de vista de un nodo sin restricciones si añadimos de antemano una fila a la izquierda y una columna abajo con valores inicializados al máximo valor que puedan tener, salvo el nodo de unión, de valor cero. Hecho esto ya se puede empezar a rellenar desde el nodo de partida original tratándolo como un nodo cualquiera.

En está implementación se usa el método arriba descrito haciendo un barrido por filas de izquierda a derecha desde abajo hasta arriba, para encontrar el valor de

salida en el nodo superior derecho. Para ello se comienza por reservar una zona de memoria de tamaño suficiente para albergar los nodos de una fila horizontal (tantos espacios de tamaño para palabras como vectores de energía caractericen a la señal que se considere como horizontal). Durante el funcionamiento del algoritmo, está zona de memoria contendrá los valores de la fila inferior a la que se está tratando y se irán reemplazando con los nuevos valores conforme se vayan calculando.

Entradas: se introducen respectivamente el número de vectores energéticos de la señal horizontal y de la señal vertical. Se introducen punteros (tamaño long) al comienzo del vector inicial de cada una de las estructuras. Por consideraciones de eficiencia (conservación de memoria) lo óptimo es caracterizar la trama más corta mediante las distancias entre palabras.

Salidas: están en la distancia energética entre vectores, en tamaño long.

5.1.8 Palabra reconocida.

Es el resultado esperado de este proyecto de reconocimiento de voz, y donde obtendremos respuesta de las ordenes enviadas al dispositivo para realizar una operación del televisor (encendido, apagado, canal arriba o abajo y aumento y disminución del volumen).

6. CONCLUSIONES

1. El rendimiento del dispositivo de reconocimiento de caracteres vocálicos es bastante bueno aun que el sistema presenta problemas de cobertura léxica, no obstante el detector de voz debe ser mejorado para aumentar la precisión, y se está trabajando en esa dirección.

2. El sistema de reconocimiento fue sometido a prueba para un conjunto de 5 palabras. Se escogió un vocabulario compuesto por los comandos para operar el televisor desde el punto de vista de encender, apagar, canal arriba y abajo, y aumento y disminución de volumen (power, subir, bajar, derecha e izquierda).

   Los experimentos se llevaron a cabo pronunciando 20 veces cada una de las palabras y anotando los aciertos y desaciertos, obteniéndose una respuesta del 92% a 93% de clasificación correcta. Cabe destacar que las pruebas se realizaron por un solo locutor y en condiciones de ausencia de ruido de fondo.

   Los resultados para el sistema de reconocimiento no han sido del todo satisfactorios 93%. La taza de reconocimiento podría aumentarse mejorando las condiciones para la adquisición de las palabras de referencia, las cuales deberían ser lo más óptimas posible.

3. Un gran número de compañías con tecnología de punta, que se encuentran el mercado del R.A.H. realizan un considerable esfuerzo para el desarrollo de un software de reconocimiento del habla para poder ser comercializado con gran precisión; ya que dichos sistemas los encontramos implementados hoy en día en muchos electrodomésticos con los que contamos en hogares u oficios, celulares por mencionar uno, nos damos cuenta que presentan

problemas de cobertura léxica, dicho esto, compararlo con el dispositivo desarrollado en este proyecto nos da una respuesta muy buena y satisfactoria, teniendo en cuenta la tecnología con que se trabajo, al poco hardware empleado y a su bajo coste.

4. El algoritmo de alineamiento temporal dinámico (DTW), es una técnica que permite comparar dos palabras sin considerar la duración de las mismas. Las palabras a comparar deben tener un tiempo de duración cercano para que el algoritmo pueda ser aplicado.

7. RECOMENDACIONES

1. La transmisión de las señales de voz para la grabación debe realizarse en forma adecuada; es decir de forma continua, clara y sin pausas. De modo que cuando se ordene un comando para la ejecución del dispositivo hacia el televisor, el algoritmo de alineamiento temporal dinámico (DTW), logre comparar las palabras a reconocer con aquellas que pertenecen al vocabulario almacenado en la base de datos. El resultado de está operación es una medida de distancia entre la muestra a reconocer y la "más cercana" a la del vocabulario. Está palabra, la de menor distancia va hacer la palabra que reconoce y que ejecuta.

2. Para la grabación de las palabras a ejecutar, se ha programado una palabra de error para compararla con las del vocabulario almacenado, y así tener un mejor resultado de comparación con las muestras, la palabra de error debe ser una palabra común y de diferente acentuación con las almacenadas en este caso se han realizado pruebas con (uno y equis), dando un buen resultado. Está palabra se almacena subiendo los tres primeros interruptores del dipswish al mismo tiempo, se transmite la palabra, el sistema la reconoce y graba bajando de nuevo los tres primeros interruptores al tiempo.

3. La grabación de las palabras o comandos (5), para la ejecución de las órdenes se graban, subiendo uno de los primeros cinco interruptores del dipswish, se pronuncia la palabra a almacenar, en este momento enciende

el led de reconocimiento de voz, de forma continua durante la transmisión, indicando la grabación, luego se baja de nuevo el interruptor quedando la palabra almacenada en la base de datos de la memoria 24LC256, y así una por una, de las cinco ordenes a ejecutar.

Ya grabadas las palabras, solo queda ordenar comandos para que el dispositivo realicé la ejecución en el televisor.

4. El usuario siempre debe tratar de mantener un nivel acústico de transmisión semejante al vocabulario almacenado. Para así obtener un menor margen de error y el resultado esperado del comando ordenado para que el dispositivo ejecute.

5. Este dispositivo cuenta con la facilidad de ser implementado en diferentes aplicaciones tanto en la industria como en electrodomésticos, y así obtener una manipulación más sencilla de los sistemas, gracias a las técnicas desarrolladas en este proyecto (R.A.H y D.T.W).

BIBLIOGRAFÍA

1. ALBARDAR, Ashok; “Procesamiento de Señales Analógicas y Digitales”, Segunda Edición, México, Thomson Editores, 2002.

2. ANGULO, Usategui José María, ANGULO, Martínez Ignacio; “Microcontroladores PIC, Diseño practico de aplicaciones”, Segunda Edición, Montevideo, McGraw-Hill, 1999.

3. IRARRAZAVAL, Pablo; “Análisis de Señales”, Primera Edición, Santiago de Chile, McGraw-Hill, 1999.

4. OPPENEIN, Alan; “Señales y Sistemas”, Segunda Edición, México, Prentice Hall, 1997.

5. PERTENCE, Junior Antonio; “Amplificadores Operacionales y Filtros Activos”, McGraw-Hill, 1991.

6. PROAKIS, J., MANOLAKIS, Ch. D.G.; “Tratamiento Digital de Señales”, Tercera Edición, Madrid, Prentice - Hall, 1998.

7. FLORES, Espinaza Andrés; “Reconocimiento de Palabras aisladas”. Disponible en: http://www.alek.pucp.%20edu.pe/~dflores/INDEX .html

8. CARRILLO, Roberto, SAN MARTÍN, Cesar. Implementación de un reconocedor de palabras aisladas dependiente del locutor. Disponible en: http://www.scielo.cl/pdf/rfacing/v12n1/art02.pdf. 28/09/2005, 15:17.

9. COLÁS, Pasamontes José; E strategias de incorporación de conocimiento sintáctico y semántico en sistemas de comprensión de habla continua en español. Disponible en: http://elies.rediris.es/elies12. 14/10/2005, 11:22

10. http://es.wikipedia.org/wiki/Arquitectura_von_Neumann. 06/11/2005, 10:27.

11. http://es.wikipedia.org/wiki/Imagen:Filt_elect_pend.PNG. 18/11/2005. 14:29.

12. http://laenciclopedialibre/La voz humana.htm. 11/10/2005, 16:44.

13. http://laenciclopedialibre/Filtro digital.htm. 08/12/2005, 10:23

14. Microchip. [En línea]. Pagina Web, URL < http:// microchip.com >. 24/09/2005, 16:14.

ANEXOS

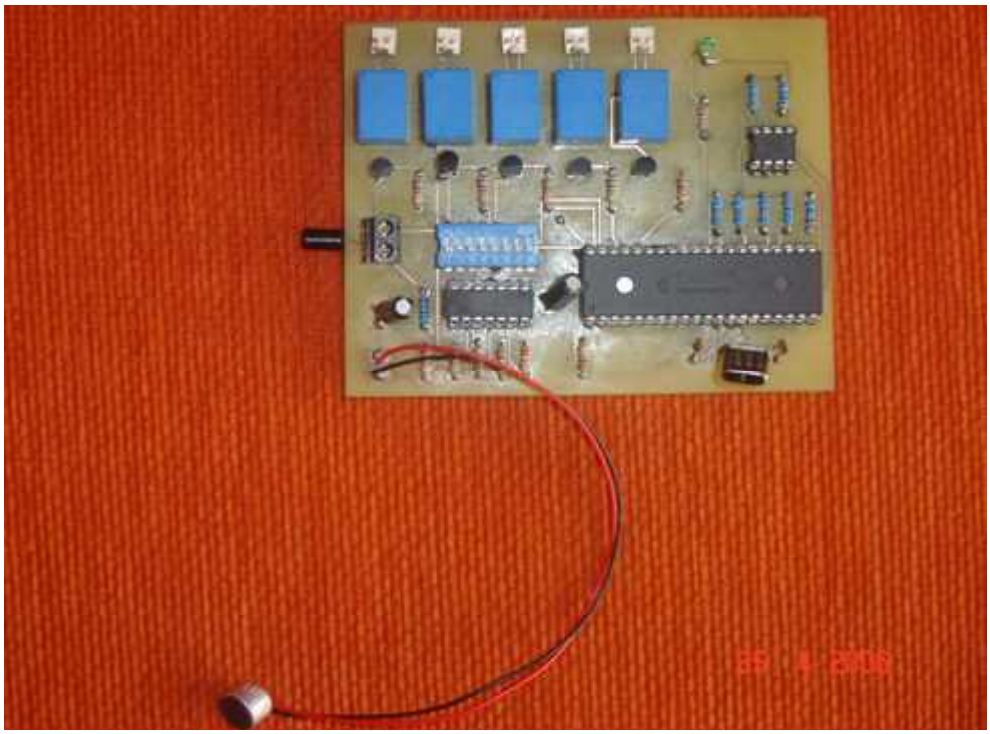
ANEXO 1. CRONOGRAMA DE ACTIVIDADES.

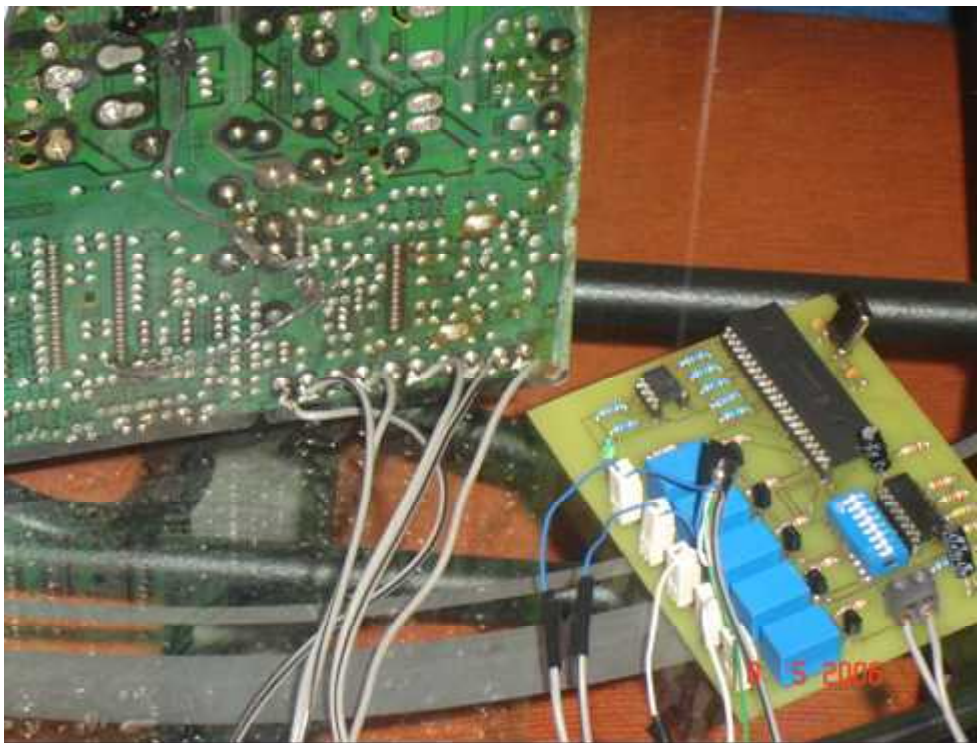
AÑO	2005						2006			
ACTIVIDADES	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE	ENERO	FEBRERO	MARZO	ABRIL	MAYO
RECOLECCIÓN DE INFORMACIÓN										
CLASIFICACIÓN DE LA INFORMACIÓN										
REALIZACIÓN DEL SOFTWARE (PIC)										
MONTAJE DEL DISPOSITIVO										
PRUEBAS										
REALIZACIÓN DEL DOCUMENTO										
ASESORÍA METODOLÓGICA										
PREPARACIÓN PARA SUSTENTACIÓN										

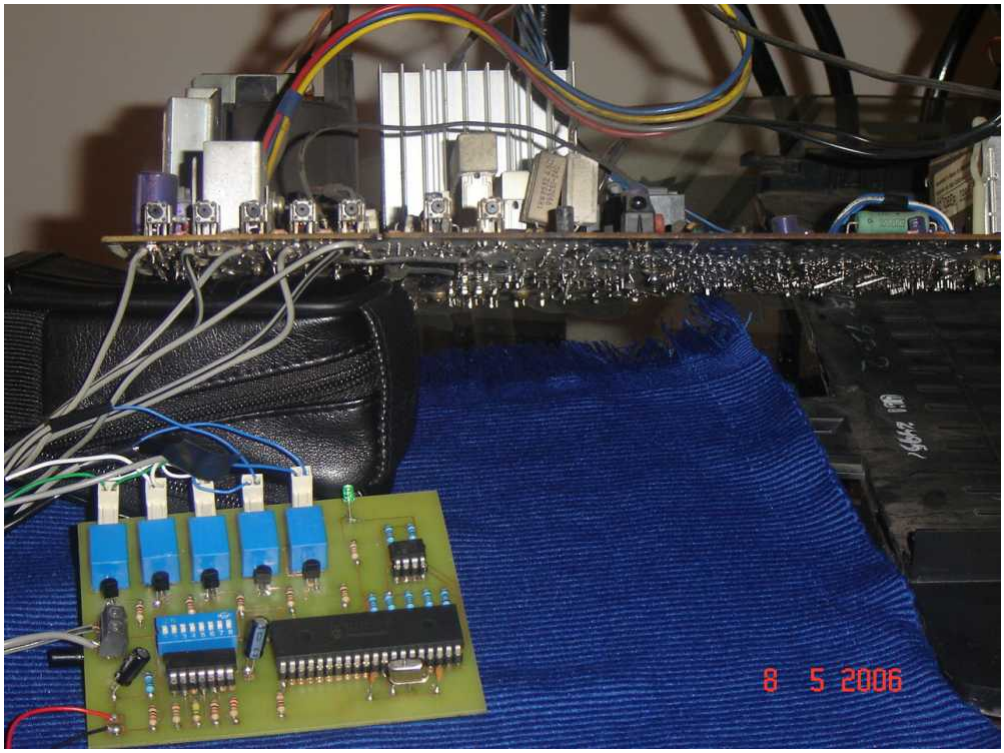
ANEXO 2. RECURSOS Y PRESUPUESTO.

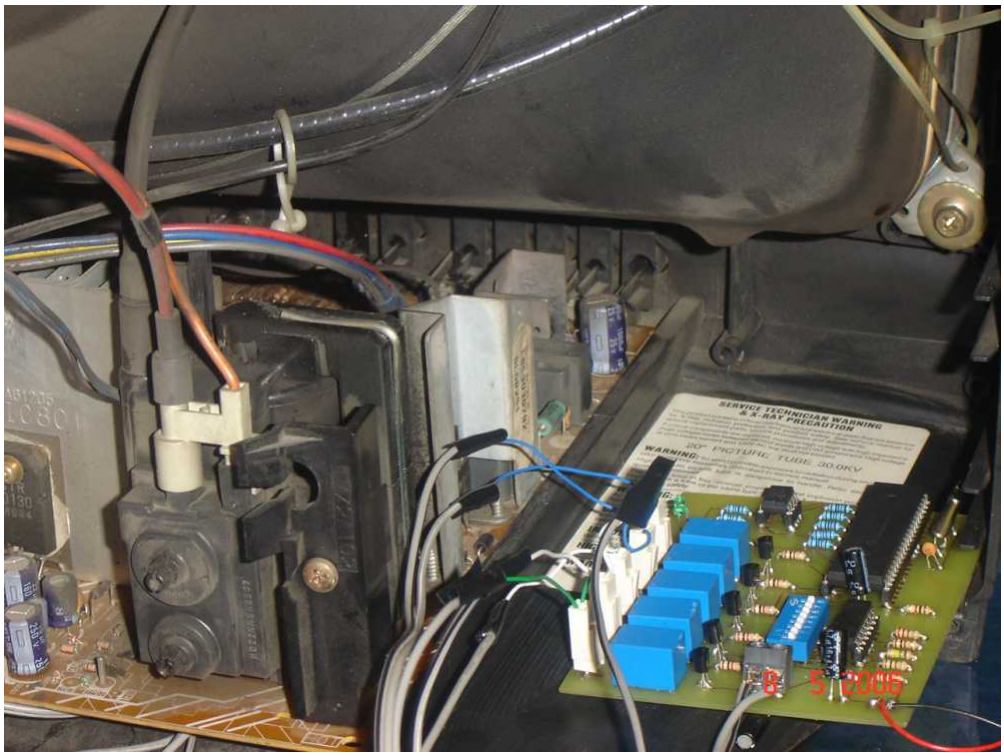
PRESUPUESTO				
PARTES ELECTRÓNICAS				
	Cantidad	v/r unitario	total	moneda
Amplificadores LM234	1	2300	2300	Pesos
Borneras	10	1000	10000	Pesos
Cable multípar (mt)	5	1000	5000	Pesos
Camillas o porta integrados	5	1000	5000	Pesos
Circuito impreso en fibra de vidrio	1	18000	18000	Pesos
Condensadores	5	100	500	Pesos
Conectores	10	350	3500	Pesos
cristales 20Mhz	2	4000	8000	Pesos
Dipswitch	1	2100	2100	Pesos
Leds	10	100	1000	Pesos
Memoria 24LC256	2	6500	13000	Pesos
Microcontrolador PIC 18F452	2	18000	36000	Pesos
Resistencias	30	50	1500	Pesos
subtotal parte electrónica			**105900**	**Pesos**
OTROS				
Fotocopias	180	50	9000	Pesos
varios(papelería-tinta)		100000	100000	Pesos
Televisor Panasonic 21¨	1	400000	400000	Pesos
Transporte (transmilenio)	30	1200	36000	Pesos
Internet	120 horas	1200	144000	Pesos
subtotal otros			**689000**	**Pesos**
Total coste del proyecto			**794900**	**Pesos**

ANEXO 3. PROTOTIPO Y CONEXIONES (FOTOS)






8 5 2006


8 5 2006


8 5 2006


8 5 2006