UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ingeniería

SISTEMA AUTOMATIZADO DE ILUMINACIÓN DE UNA CASA MEDIANTE COMANDOS DE VOZ

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO ELÉCTRICO ELECTRÓNICO

P R E S E N T A:

OSCAR FRANCISCO NAVARRETE TOLENTO

Director: Dr. José Abel Herrera Camacho

México, D.F. 2009

A mis Padres,
Oscar Armando Navarrete y Maya y
María Estela Tolento Maciel
Que me han brindado su apoyo en todo momento,
por su ejemplo y consejos a lo largo de mi vida.

A mis Hermanos,
Eduardo y Diana
Con entrañable aprecio y cariño.

A mis Abuelos,
Por el gran ejemplo que me han dado.

A mis Tíos,
Por su apoyo incondicional, por su ejemplo
y las enseñanzas que me han dado.

A mis Primos,
Con cariño.

A mis Amigos.
Que siempre me han alentado para seguir adelante.

A mi Director de Tesis,
Dr. José Abel Herrera Camacho
Por sus consejos y observaciones en la realización de este trabajo.

Al Dr. Pablo Roberto Pérez Alcázar,
Por su apoyo y comentarios.

Al Honorable Jurado,
Por sus valiosas aportaciones.

A la Universidad Nacional Autónoma de México,
Por la formación que en ella recibí.

A mis profesores,
Por haberme regalado parte de su sabiduría y experiencia.

Índice General

Introducción.....1

Las dificultades en el reconocimiento de voz.....2

Descripción general de la tesis.....3

I. Fundamentos sobre sonido y voz.....5

1.1. Fundamentos de Generación del sonido.....5

1.2. El sistema generador de la voz.....8

1.2.1. El tracto pulmonar.....9

1.2.2. La laringe.....10

1.2.3. El Tracto Vocal.....13

1.2.4. Algunas diferencias entre las voces del Hombre y la Mujer.....14

1.3. El modelo digital para la voz.....15

1.4. Fonética Articulatoria y Acústica.....17

1.4.1. Las vocales.....18

1.4.2. Diptongos.....18

1.4.3. Consonantes.....19

II. Aproximaciones al reconocimiento de voz.....23

2.1. Generalidades.....23

2.2. Sistema de reconocimiento de voz, de palabras aisladas y dependiente del locutor utilizando MSBC (basado en el modelo LPC).....26

2.2.1. Preprocesamiento.....27

2.2.2. Algoritmo de Detección de Inicio y Fin de palabra.....31

2.2.3. Entrenamiento.....34

2.2.4. El modelo LPC.....35

2.2.5. Distancias y medidas de distorsión.....40
2.2.6. Cuantización Vectorial (VQ).....46
2.2.7. Algoritmo de K-medias.....49
2.2.8. Reconocimiento.....50

III. Herramientas de Desarrollo: DSP, Microcontrolador y Software de Desarrollo.....53

3.1 Introducción.....53
3.2 Características Principales de la tarjeta de desarrollo del DSP….....53
3.3 Características del DSP TMS320C6711.....55
3.4 Descripción del CPU (núcleo del DSP).....57
3.5 interrupciones de DSP.....60
3.6 Periféricos del DSP.....61
3.7 Code Composer Studio.....65
3.7.1 Herramientas de generación de código.....66
3.7.2 El IDE del CSS.....68
3.7.3 DSP/BIOS Plug-ins y API.....70
3.7.4 Emulación de Hardware e intercambio de datos en tiempo real (RTDX) 72
3.8 El Microcontrolador.....73
3.8.1 Características principales del PIC 16F877A.....73
3.8.2 Mapa de Memoria.....75
3.8.3 Puertos.....79
3.8.4 Temporizadores (timers).....80
3.8.5 Interrupciones.....82
3.8.6 Modo de bajo consumo (Sleep).....83
3.8.7 Configuraciones del Oscilador.....84
3.9 MPLAB.....85

3.9.1 El IDE del MPLAB.....85
3.9.2 Compilación del programa y carga al PIC.....85

IV. Implementación del sistema de control de iluminación.....87

4.1 Introducción.....87
4.2 sistema de reconocimiento de comandos. 87
4.2.1 Preprocesamiento de las señales de voz.....88
4.2.2 Entrenamiento del sistema de reconocimiento de comandos….....90
4.2.3 Reconocimiento de los comandos.....92
4.3 Interfaz con el sistema eléctrico de iluminación.....95
4.3.1 Adecuación de la señal.....96
4.3.2 Algoritmo en el PIC.....101
4.4 Circuito eléctrico de iluminación.....104
4.4.1 El tiristor.....104
4.4.2 Tipos de tiristores.....105
4.4.3 Características del triac seleccionado.....108
4.4.4 Optoacopladores (MOC).....108
4.5 Circuito Final.....110
4.5.1 Preamplificador para la señal del micrófono.....110
4.5.2 Circuito Esquemático.....111

V. Resultados, Conclusiones y Trabajo a Futuro.....113

5.1. Porcentajes de Reconocimiento.....113
5.2. Conclusiones.....116
5.3. Trabajo a Futuro.....118

Bibliografía.....119

Introducción

La operación de las máquinas por medio de la voz ha sido una de las visiones que la mayoría de las personas ha tenido. Imaginemos por un momento que nos encontramos en el trabajo y ha terminado la jornada laboral. Qué bueno sería poder decirle a nuestro automóvil "llévame a casa"; el automóvil recibiría nuestra instrucción, se encaminaría a la casa y nosotros podríamos relajarnos en el camino. Esto día con día se está convirtiendo más y más en realidad gracias a los avances tecnológicos, especialmente en el área de la electrónica, las telecomunicaciones y la computación. La implementación de este tipo de tecnología involucra varias disciplinas como son: el procesamiento digital de voz, el control automático, las redes inalámbricas, etc. El objetivo de esta tesis fue realizar una interfaz de voz para controlar la iluminación de una casa mediante comandos de voz.

En la actualidad los procesadores tienen una gran velocidad de ejecución, lo que permite implementar sistemas cuyo funcionamiento se realiza prácticamente en tiempo real. El procesamiento digital de señales es, sin lugar a dudas, una de las áreas que más se ha beneficiado con estos avances, sobre todo con la creación de procesadores especializados, también conocidos como DSP's (Procesadores Digitales de Señales). En la tesis nos auxiliamos de un DSP para llevar a cabo la etapa de reconocimiento.

El propósito principal de esta tesis es efectuar un sistema capaz de reconocer 12 comandos de voz, correspondientes al encendido y apagado de la luz en cada una de las 6 habitaciones de una casa. El reconocimiento se debe realizar en tiempo real, con ayuda del DSP TMS320C6711, empleando las siguientes técnicas: análisis LPC, cuantización vectorial (con k-medias) y la distancia Itakura.

Las pruebas del sistema fueron hechas para dos casos: uno con condiciones normales de ruido y otro en el que se realizaron las pruebas en presencia de música como ruido ambiente.

Se pronunciaron 20 repeticiones de cada uno de los comandos y los resultados obtenidos fueron los siguientes: para el caso de condiciones normales de ruido, el 97.5% de las palabras fueron reconocidas al primer intento, el 2.5% de las palabras fueron reconocidas al segundo intento, de manera que no se detectó ninguna palabra desconocida; mientras que para el caso en presencia de ruido ambiente el 92.9% de las palabras fueron reconocidas al primer intento, el 4.6% de las palabras fueron reconocidas al segundo intento, mientras que el 2.5% de las palabras se detectaron como palabras desconocidas.

Las dificultades en el reconocimiento de voz

Uno de los aspectos más difíciles en el reconocimiento de voz es su naturaleza interdisciplinaria. Por lo tanto, para la implementación de un sistema satisfactorio se requiere del procesamiento de señales, la acústica, el reconocimiento de patrones, la lingüística, la fisiología, la teoría de la información, etc.

Las dificultades recaen principalmente en los siguientes factores:
Dependencia del locutor: el sistema será capaz de reconocer únicamente a un locutor (dependiente del locutor) o a múltiples locutores (independiente del locutor).
El tamaño del vocabulario: un vocabulario pequeño (de 1 a 99 palabras), mediano de (100 a 999 palabras) o grande (1000 palabras o más).
Forma de la pronunciación: se manejan unidades discretas, principalmente palabras, con pausas entre ellas (reconocimiento de palabras aisladas), o como una pronunciación continua (reconocimiento de palabras conectadas o reconocimiento continuo.

El nivel de ambigüedad: por ejemplo “caza” “casa”, así como las confusiones acústicas (por ejemplo: “casa”, “masa”, “gasa”).
La naturaleza del ruido ambiente: ambientes controlados o con ruido.

Descripción general de la tesis

El sistema automatizado de control de iluminación, que se ha implementado con auxilio de un DSP, permite reconocer los comandos de voz prácticamente en tiempo real y, al estar conectado a un circuito de potencia mediante un PIC, logra encender y apagar los focos (en este caso) correspondientes a cada una de las habitaciones de la casa. Este tipo de implementaciones no son aún comerciales y en los países de mayor desarrollo tecnológico están en etapa de desarrollo, siendo inminentes algunas de sus aplicaciones. Ojalá en nuestro país podamos desarrollar tecnología propia que logre este propósito en un futuro cercano.
El hardware empleado es principalmente el DSP Starter Kit TMS320C6711 de Texas Instruments y el microcontrolador PIC 16F877A, usando cuantización vectorial (VQ) y vectores de predicción lineal (lpc´s).

El capítulo 1 trata el proceso de producción y percepción de la voz, la clasificación de sonidos y la descripción de un modelo del tracto vocal. El capítulo 2 describe los fundamentos de procesamiento digital de voz empleados a lo largo de la tesis.

El capítulo 3 describe el hardware empleado para el desarrollo del proyecto, en este caso, la tarjeta de desarrollo DSK C6711 y el PIC 16F877A. De estos elementos se presentan sus principales características y sus entornos de desarrollo, es decir, el CCS (Code Composer Studio) para el caso del DSK y el MPLAB para el caso del PIC.

El capítulo 4 describe el proceso que se siguió para implementar el sistema de control de iluminación. Finalmente, el capítulo 5 describe los resultados obtenidos y presenta las conclusiones.

I. Fundamentos sobre sonido y voz

1.1. Fundamentos de Generación del sonido

Es necesario conocer algunos conceptos empleados en el estudio de las señales de voz, así como en el funcionamiento de los sistemas generadores de voz, de tal forma que se puedan establecer las características que sirvan para realizar un correcto reconocimiento de voz.

Naturaleza del sonido

El sonido es una perturbación en un medio elástico, que causa una alteración de la presión y un desplazamiento de las partículas en dicho medio, que puede ser detectada auditivamente.

Tono y Timbre

Estos son dos calificativos subjetivos. El tono se refiere a la frecuencia fundamental del sonido y el timbre a las armónicas presentes y sus amplitudes.

Intensidad del sonido

La intensidad de una onda sonora se define como la potencia media transportada por unidad de superficie. Como la potencia es igual al producto de la fuerza por la velocidad se puede llegar al valor medio de la intensidad en un periodo como:

$$I = \frac{P^2}{2\rho v}$$

Donde P es la amplitud de los cambios de presión. Por ejemplo, considerando valores comunes de $\rho = 1.22x10^{-3}\, g/cm^3$ y $v = 3.46x10^{4}\, cm/s$, con:

P=280 dinas/cm^2, entonces: $I = 94x10^{-6}\left[W/cm^2\right]$

P=3x10^{-4} dinas/cm^2, entonces: $I = 1x10^{-16}\left[W/cm^2\right]$

De tal manera que en una conversación normal, donde se tiene $1x10^{-9}[W/cm^2]$, considerando un área de $1[m^2]$, entonces se tiene una potencia media producida de $1x10^{-5}[W]$. Por lo tanto, un millón de personas producen $10[W]$ de potencia acústica. Como la potencia acústica es muy pequeña, es muy común utilizar una escala logarítmica llamada nivel de intensidad β, definida como:

$$\beta = 10 \log \frac{I}{I_o}$$

Donde I_o es el nivel de referencia.

Es usual que el nivel de referencia sea igual a $1x10^{-16}[W/cm^2]$. Tomando como referencia este valor se obtienen los niveles de intensidad mostrados en la tabla 1.1.

Tabla 1.1, Intensidad del sonido.

Fuente del sonido	Nivel de intensidad [dB]
Umbral o sensación desagradable	120
Maquina remachadora	95
Conversación ordinaria	65
Murmullo de las hojas	10
Umbral o sensación sonora	0

La sensación sonora equivalente a la intensidad es llamada sonoridad (no es proporcional a la intensidad). Los niveles de sonoridad dependen de la frecuencia de la onda. Fletcher desarrolló experimentalmente estas características, estableciendo las primeras curvas isofónicas. Las curvas isofónicas de la figura 1.1 muestran la relación que existe entre la frecuencia y la intensidad (en dB).

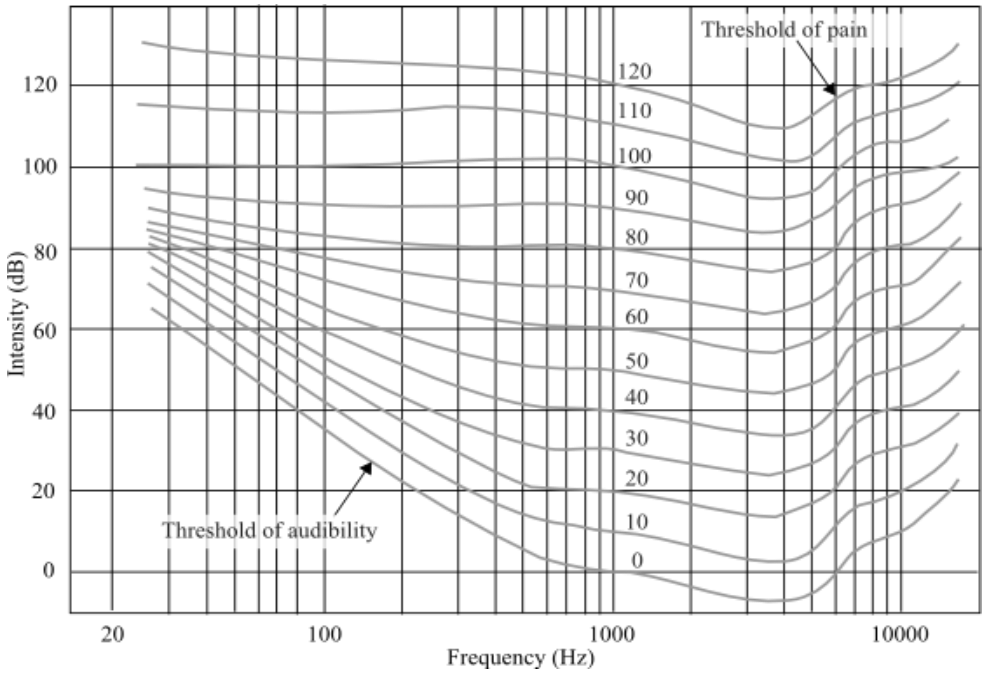


Figura 1.1, sonoridad.

1.2. El sistema generador de la voz

El proceso básico de generación de la voz es el mismo ya sea al hablar o al cantar. El cerebro envía señales a través del sistema nervioso a los músculos de la cabeza, cuello y torso de manera que se produzca la inhalación previa a la generación. Al final de la inhalación se efectúan varias acciones: el movimiento de los cartílagos aritenoides en la laringe acerca a las cuerdas vocales entre sí, el volumen de los pulmones disminuye para producir una presión de aire positiva en los mismos, y el aire comienza a fluir hacia la laringe.

La presión de aire resultante en las cuerdas vocales las mueve para que de esta forma comience el primer ciclo de movimiento. Después de que se mueven un poco, se moverán entonces en la dirección opuesta debido a su elasticidad y a las variaciones en la presión del aire. Si el aire sigue fluyendo y la presión se mantiene, las cuerdas vocales continuarán oscilando produciéndose los sonidos.

De una forma más detallada, los órganos productores de sonidos se pueden dividir en tres regiones:

1. Tracto pulmonar o respiratorio: Formado por los pulmones y la tráquea. Producen corrientes de aire.
2. Laringe: Área situada arriba de la tráquea y abajo de la faringe. Aquí se generan los sonidos.
3. Tracto vocal: Formado por la faringe y las cavidades bucal y nasal. Se modulan los sonidos provenientes de la laringe para producir los sonidos resultantes.

En la figura 1.2 podemos observar la ubicación de algunos de estos órganos.

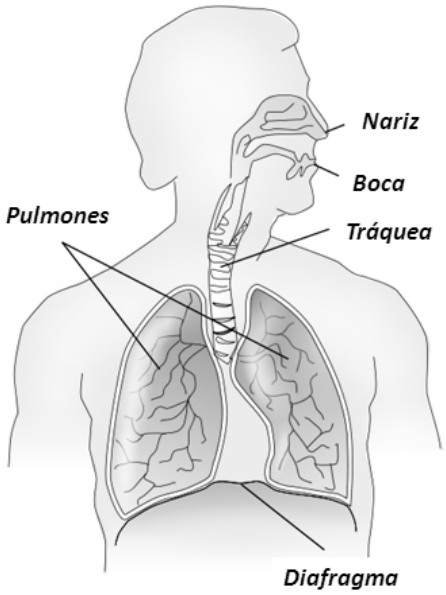


Figura 1.2, El Aparato Respiratorio

1.2.1. El tracto pulmonar

En la respiración los pulmones se llenan de aire, mismo que al ser expulsado es conducido por la tráquea hacia la laringe. Estos órganos controlan la amplitud de los sonidos, y la única contribución audible del tracto son los silencios ínter y entre palabras.

Los pulmones son una masa esponjosa que tiene un área grande. Su capacidad es de 4 a 5 litros en un adulto. Están contenidos en una cámara de aire, la pleura, la cual está contenida a su vez lateralmente por las costillas e inferiormente por el diafragma.

El diafragma es un músculo en forma de domo ubicado abajo de las costillas; cuando este músculo se contrae, el domo se extiende hacia afuera, el volumen de la pleura se incrementa y el aire entra a los pulmones. Cuando el diafragma se relaja, su extensión se contrae y el proceso es inverso. Por otro lado cuando se exhala involuntariamente, por ejemplo, al toser o gritar, se utiliza fuerza adicional del diafragma por los músculos abdominales. La producción de sonidos requiere de una

presión por parte de los pulmones del orden de 4cmH_2O, para sonidos muy suaves, hasta aproximadamente 20 cmH_2O, para sonidos muy fuertes y de altas frecuencias.

La tráquea es un conjunto de cartílagos en forma de anillos unidos por tejido. Es un tubo rígido pero transversalmente puede curvarse o torcerse en respuesta a movimientos de la cabeza. Tiene aproximadamente 12 cm de longitud y 2 cm de diámetro. En su parte inferior la tráquea se une con los bronquios, mismos que conducen al interior de los pulmones. En su parte superior, la tráquea desemboca en la laringe. La respiración consiste en inhalaciones y exhalaciones regulares de igual longitud. La generación de la voz consiste en inhalaciones largas o cortas; así como de exhalaciones controladas.

1.2.2. La laringe

Está formada por tres cartílagos (cricoides, tiroides y aritenoides), un conjunto de músculos y las cuerdas vocales. Los dos primeros contienen y controlan las cuerdas vocales, las cuales constituyen la fuente de generación de sonidos, cierran la tráquea para proteger al tracto pulmonar de objetos y permiten la formación de presión dentro del tórax y el abdomen.

El cartílago cricoides es el anillo más superior de la tráquea y tiene una altura mayor en la parte posterior. La tiroides está localizada al frente, a una altura semejante al cricoides. La forma de la tiroides le da la dureza para resistir el empuje de las cuerdas vocales. El cartílago aritenoides soporta la terminación posterior de las cuerdas y está conectado a la parte alta del cricoides. Estos cartílagos están controlados por un conjunto de músculos unidos al cartílago cricoides y pueden separar o unir la parte terminal de las cuerdas vocales.

Las cuerdas vocales son un tejido sólido con dobleces entre el frente y la parte posterior de la laringe. Cuando las partes terminales de las cuerdas están separadas, las cuerdas están abiertas, es la posición para la respiración. Cuando las partes terminales están juntas, las cuerdas están cerradas y proporcionan el sello al tracto

pulmonar para la deglución. Cuando las partes terminales se abren y cierran parcial o totalmente, de manera rápida y secuencial, se producen la exhalación y los sonidos.

Después de que los pulmones presionan el aire, la siguiente función es realizada por la laringe y es llamada excitación. Esta adquiere las formas siguientes: fonación, susurro, fricación, compresión y vibración.

Fonación

Este término se refiere a la oscilación de las cuerdas vocales por los movimientos de los cartílagos aritenoides. Cuando el aire es conducido a través de las cuerdas vocales, éstas vibran y sus oscilaciones son dirigidas por la masa y tensión de las cuerdas y el efecto Bernoulli.

La apertura y cierre de las cuerdas secciona el pulso de aire en pulsos cuasi-periódicos llamados pulsos glotales, con una frecuencia fundamental llamada tono. Las formas de onda son aproximadamente triangulares y tienen un ciclo de trabajo del orden de 0.3 a 0.7; como consecuencia a su forma, las altas frecuencias disminuyen su amplitud a 12 dB/octava. Su naturaleza paso-bajas proporciona un espectro con una fuerte fundamental y, progresivamente, armónicas más débiles.

Existen diferentes modos de vibración, llamados registros. Sonidos resultantes de la fonación se llaman *sonoros*, mientras que sonidos con ausencia de fonación se denominan *sordos*. Así, por ejemplo, las vocales son sonidos sonoros y las consonantes, como la f, s, p y k, son sonidos sordos.

Susurro

Los susurros son generados en la laringe. Las cuerdas vocales están juntas por el cartílago aritenoides, pero en lugar de sellar completamente la glotis existe una pequeña apertura triangular entre estos cartílagos. El aire que fluye a través de esta apertura genera turbulencias, que ocasionan ruido de banda ancha; el cual sirve como señal excitadora.

Los susurros son más débiles que las fonaciones ya que implican un menor volumen de aire, y tienen mayor energía en las altas frecuencias. En el caso del idioma inglés los únicos sonidos con susurro están asociados con la letra "h" como en el caso de la palabra "hello" o con "w" como en el caso de la palabra "where". Sin embargo; se producen sonidos inteligibles si todo un conjunto de sonidos con fonación son reemplazados por susurros de los mismos sonidos. Esto es común cuando se trata de hablar en voz muy baja.

Fricación

La fricación es similar al susurro en cuanto que el flujo de aire turbulento genera ruido de banda ancha; pero existe un lugar de articulación adicional en el tracto vocal. Por ejemplo, para la letra "f", en "finger", el lugar de articulación adicional es entre los dientes superiores y la lengua, o la "s", en "salt", en donde el lugar de articulación adicional es entre la base de la lengua y la arista alveolar.

Los sonidos producidos son llamados *fricativos*; y la fricación puede ocurrir con o sin fonación.

Dado que el lugar de articulación es cerca de los labios, sólo una pequeña parte de tracto vocal ésta entre la fuente de excitación y el aire de salida. Esto significa que la modulación producida por el tracto vocal está limitada en extensión y complejidad.

Al igual que el susurro, la fricación es más baja en amplitud que la fonación y tiene una proporción mucho más amplia de las altas frecuencias. Sin embargo, el filtrado muy limitado del tracto vocal, con sus pérdidas menores a bajas frecuencias, permite una mayor radiación de energía; por consiguiente los sonidos fricativos son más sonoros que los susurros. La fricación puede ocurrir durante interrupciones de fonaciones. Esto implica variaciones muy rápidas de amplitud y frecuencia, por lo que juega el papel de modulador.

Compresión

Cuando el tracto vocal está prácticamente cerrado y una persona sigue exhalando, la presión aumenta y resulta un pequeño transitorio. La combinación de un silencio pequeño seguido por una ráfaga de ruido crea una excitación aperiódica, en éste caso la onda de presión es una función escalón con un espectro inverso a la frecuencia. Si el transitorio es abrupto y limpio, el sonido es una *oclusiva* o plosiva como en el caso de "p" en "spin"; si es gradual y turbulento, éste se clasifica como un sonido muy parecido al fricativo, llamado *africativo*, como en el caso de "j" en la palabra "reject".

Vibración

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

1.2.3. El Tracto Vocal

Este término engloba a los órganos productores de voz situados arriba de las cuerdas vocales. Consiste de cinco elementos: faringe laringeal, faringe oral, faringe nasal, cavidad oral y cavidad nasal.

La parte superior o techo de la boca puede ser dividida en dos regiones. Al frente, el techo está formado por un hueso palatal que separa la boca de las cavidades nasales. Atrás del palatal, el techo está formado por un músculo y tejido conectivo llamado velo. El velo puede ser elevado por un músculo y presionado contra la pared trasera de la faringe para sellar los pasajes nasales del resto del tracto vocal. La úvula es un apéndice carnoso atrás del velo. Enfrente del paladar se encuentra la arista alveolar, formada por la parte gruesa del hueso, donde los dientes frontales están insertados.

La epiglotis es un cartílago en forma de plato por encima de las cuerdas vocales y atrás de la lengua y no tiene una función específica en la producción de voz.

En el adulto, el tracto vocal es de aproximadamente 17 cm. de longitud. Dado que las ondas acústicas pasan por él, su comportamiento espectral es modificado por sus resonancias, las cuales dependen de las formas que adopte el tracto. Moviendo la lengua se pueden modificar la estructura de la cavidad oral y de la faringe oral. Se puede desacoplar la cavidad nasal del sistema levantando el velo de manera que selle la cavidad.

1.2.4. Algunas diferencias entre las voces del Hombre y la Mujer

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

Durante la pubertad, el tamaño de las cuerdas se incrementa en promedio de 5-10 mm. en el caso de los hombres y de 3-5 mm. en el caso de las mujeres. Este aumento reduce el tono promedio a 120 Hz en el hombre y 200 Hz en la mujer. De manera que el tono más alto (casi una octava) de la mujer, comparado con el hombre, significa que las cuerdas vocales vibran casi el doble de veces por segundo que en los hombres.

1.3. El modelo digital para la voz

Un modelo de la voz que se considere completo debe incluir: los cambios en la señal de excitación, la respuesta del tracto vocal y los efectos de los labios en la radiación. Dicho modelo es el fuente-filtro que ha sido usado por todos los sistemas de procesamiento de voz. En este modelo los articuladores son modelados por filtros LTI usando el hecho de la independencia relativa entre la fuente y el tracto vocal y la estacionariedad en intervalos cortos (10 a 20 ms.) de los sonidos.

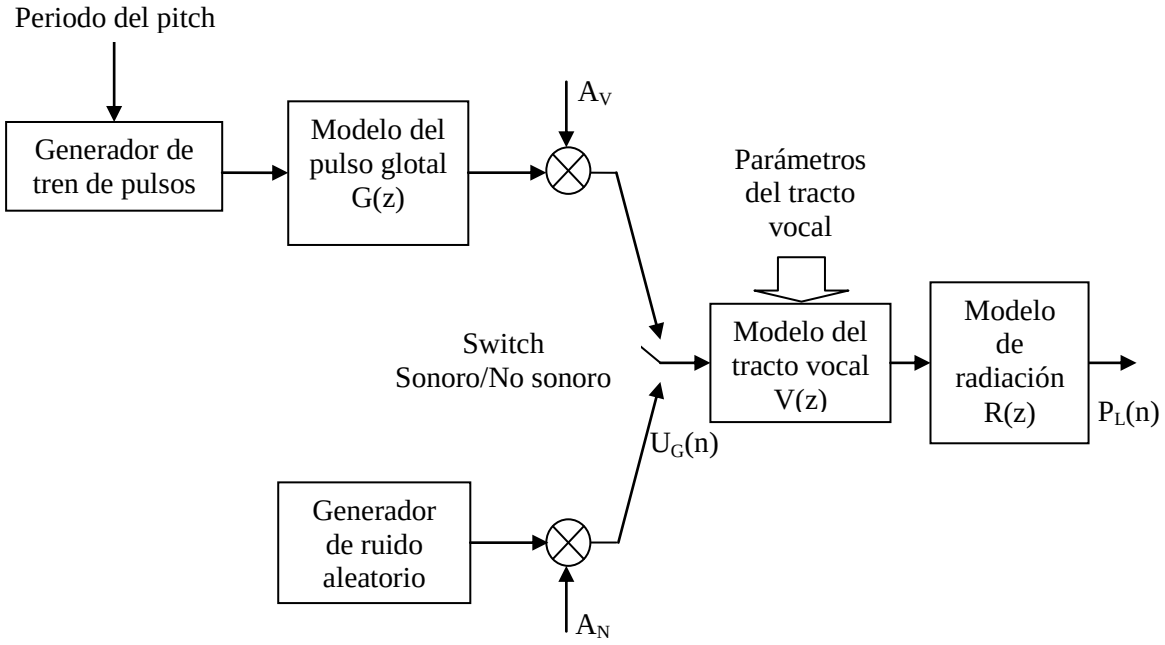


Figura 1.3, Modelo digital para la voz

En la figura 1.3. la excitación glotal es la entrada al filtro. Los cinco tipos de excitación son reducidos de manera general, a dos señales periódicas (sonoras) y ruido turbulento (sonidos sordos) con distribución gamma o laplaciana y espectro plano. Para los sonidos sonoros, se debe aplicar un modelo cambiante de la glotis. El espectro glotal es un tren de impulsos espaciado a frecuencias iguales a la frecuencia del tono fundamental. El efecto es, aproximadamente, una caída de 12 dB/ octava, alrededor de 0.8 – 0.1 khz.

El tracto vocal puede ser modelado aproximadamente por la función de transferencia:

$$H(z) = \frac{G}{1 - \sum_{i=1}^{N} \alpha_i z^{-i}}$$

Donde G representa al factor de ganancia total y α_i las ubicaciones de los polos. Los polos de H (z) corresponden a las resonancias o formantes de la voz.

La radiación de la voz tiene la propiedad de que a bajas frecuencias la presión del sonido es proporcional a la derivada de la velocidad volumétrica. Esto introduce un levantamiento de 6 dB/ octava en el espectro, que puede ser modelado por:

$$R(z) = 1 - z^{-1}$$

Los tres efectos pueden ser representados en una sola función de transferencia todo-polos llamada espectro de envolvente. En su forma refleja la información principal de la señal de voz y casi todos los sistemas de voz tratan de generarla o recuperarla.

1.4. Fonética Articulatoria y Acústica

Se revisaron algunos aspectos básicos de la fonética articulatoria y acústica para sonidos en el idioma español desde la perspectiva de un locutor, con la limitante de que en México, al igual que en muchos otros países, existe una gran diversidad de acentos.

El número de letras del alfabeto español difiere de acuerdo al autor o escuela. Sin embargo, podemos considerar el más grande de ellos, que incluye 30 letras, 5 vocales y 25 consonantes. Los fonemas asociados casi igualan al número de letras, ver tabla 1.2. Esta tabla incluye los símbolos de la IPA (International Phonetics Association).

Tabla 1.2, Alfabeto fonético del español.

Fonema	Ejemplos	Fonema	Ejemplos
[p]	pozo; topo; vas a pescar	[β̞]	bebé; ¡pobre bebé!; ¡viva!; vamos a ver; cuervo
[b]	bestia; ámbar; vaca; calvo; sin billete; sin vestido	[ð̞]	dado; Madrid; dad; comed; partido; arder
[t̪]	tamiz; átomo; para ti	[j]	yo ya voy; poyo; pollo; vas a llamar
[d̪]	dado; cuando; con dulzura	[ɯ̯]	trigo; Argos; jugar; trasgo; va a ganar; las garras
[ɟʝ]	ya; llamar; comen yogur	[r]	raro; perro
[k]	caña; laca; quisimos; de Córcega	[ɾ]	raro; pero; bravo; tronco; amor; comer
[g]	gato; lengua; guerra; dan golosinas	[ɻ]	caldo; alto; el duelo; mal trabajo
[tʃ]	chubasco; acechar	[l]	lino; calor; el sabio
[dʒ̟]	inyección; en llamas; sin yoyó	[ʲ]	colcha; salchicha; el chico; el ñu; el yoyó; el llavero
[f]	fase; café; para financia	[ʎ]	llave; pollo
[v]	Afganistán; Dafne; rosbif de ternera	[m]	madre; comer; voy a Málaga
[θ]	cereza; zorro; lazo; paz; voy a cerrar	[m̩]	anfibio; enfilar; un ánfora; un fanfarrón
[ð]	portazgo; paz ganada; haz ramos; hazlo; hazme el trabajo	[n̩]	pinza; ponzoña; sin zapatos; danzad; con cereales; son cerezas
[s]	saco; casa; puertas; de sobra	[n̪]	donde; cuanto; comen tiramisú; vienen de jugar
[z]	Israel; isla; es mía; sabes la última	[n]	nido; anillo; enroscar; consejo; cúrate en salud; sin radio; sin agua
[ʃ]	esos chinos; comemos chorizos; ¿vienes, chaval?	[nʲ]	concha; panchitos; allí pastan ñúes; vienen ya; anuncian lluvia; vienen chinos
[ʂ]	comes ñoquis; vemos ñúes; es ya la hora; ésas lloran; estas llaves	[ɲ]	ñoquis; cabaña
[x]	jamón; general; la gente, carcaj, boj	[ŋ]	cinco; venga; ángel; sobran jamones; traen cántaras; con ganas
[χ]	juntar; rugir; bruja; hijo		

1.4.1. Las vocales

Los fonemas vocálicos corresponden a las cinco vocales del alfabeto. Todos son articulaciones abiertas de corta duración, completamente sonoras, ya sea que estén acentuadas o no. Los tres primeros formantes vocálicos tienen aproximadamente las frecuencias que se muestran en la tabla 1.3. De esta tabla podemos notar también que entre mayor sea la sección del tracto vocal mayor es el formante F0; así también, entre más adelante y elevada esté la lengua, mayor es el formante F1.

Los fonemas varían levemente de acuerdo a las posiciones de las vocales en las palabras y al dialecto o región de quien habla. Estas variantes se llaman *alófonos*. El principal parámetro articulatorio (la posición de la lengua) se conserva para los alófonos de una vocal. Sin embargo, otros parámetros de menor importancia pueden variar, como el redondeo de los labios o la duración del sonido.

Tabla 1.3, Frecuencias de los formantes vocálicos

Vocal	F0 [Hz]	F1 [Hz]	F2 [Hz]
a	900	1300	2100
e	375	2200	2550
i	325	2300	3900
o	400	550	4300
u	325	425	4500

1.4.2. Diptongos

Este término se refiere al monosilábico que empieza en o cerca de la posición articulatoria de una vocal y se mueve a o hacia la posición de otra vocal. La vocal con la mayor abertura del tracto vocal se llama *núcleo silábico*, mientras que la vocal con la menor abertura se conoce como *sílaba marginal*.

Un diptongo puede ser creciente o decreciente. El primero existe cuando el núcleo silábico precede al margen silábico, y el segundo viceversa.

1.4.3. Consonantes

Las consonantes se clasifican de acuerdo a las formas de articulación. Estas categorías se refieren a los grados de constricción del punto de articulación y la manera en que se exhala para el siguiente sonido. Sin embargo, es importante describir todas las categorías para las consonantes:

- Africadas. Existe un cierre inicial del tracto vocal seguido de una expiración gradual que produce turbulencia.

- Aspiradas. El tracto vocal está cerrado inicialmente en el punto de articulación y se exhala aire antes del siguiente sonido.

- Fricativas. El tracto vocal está abierto parcialmente en el punto de articulación y el velo está cerrado. Se genera ruido en el punto de articulación.

- Laterales. El tracto vocal está cerrado en el punto de articulación pero abierto a los lados.

- Nasales. El tracto vocal está cerrado en el punto de articulación y el velo está abierto.

- Plosivas. El tracto vocal está cerrado en el punto de articulación, el pasaje nasal está cerrado, y existe una exhalación limpia y cortante.

- Semivocales. El tracto vocal está parcialmente abierto en el punto de articulación sin turbulencia.

- Vibrato. Existe una abertura y cerradura oscilatorias en el punto de articulación, seguida de una exhalación gradual que produce turbulencia.

Todos los fonemas que corresponden a las vocales, diptongos, semivocales y nasales, se conocen colectivamente como *sonorantes*. Los fonemas sonorantes implican sonidos sonoros y excitan al tracto vocal solamente con pulsos cuasi-periódicos originados por la vibración de las cuerdas vocales. En contraste, las restantes clases son excitadas fundamentalmente en punto de constricción del tracto vocal y se denominan *obstructivas*.

Si bien las formas de articulación dividen a los fonemas en categorías muy amplias, basadas en diferencias en la excitación, el lugar de articulación identifica las diferencias en el tracto vocal de acuerdo al punto máximo de constricción en el tracto vocal y permiten diferenciar más sutilmente a los fonemas que tienen la misma forma de articulación.

A lo largo del tracto vocal existen aproximadamente ocho regiones o puntos de articulación que se asocian con las consonantes; sin embargo, un idioma solo utiliza un número reducido de ellos.

- Alveolar. La punta de la lengua se acerca o toca la punta alveolar en el techo de la boca.
- Dental. La punta de la lengua hace contacto con la parte posterior de los dientes incisivos superiores.
- Glotal. Los dobleces de las cuerdas se cierran o constriñen.
- Labiales. Existe una constricción en los labios. Bilabial denota constricción en ambos labios, mientras que labiodental denota contacto del labio inferior con los dos dientes superiores.
- Palatal. El dorso de la lengua se constriñe con el paladar duro.
- Velar. El dorso de la lengua se aproxima al paladar suave.

Para algunos casos, el punto de articulación está afectado fuertemente por el ambiente fonético a su alrededor.

Plosivas

Existen seis fonemas plosivos: /b/, /d/, /g/, /k/, /p/, y /t/;

Los tres tipos de sonidos plosivos, de acuerdo al punto de articulación, tienen las siguientes características espectrales:

- /p/ y /b/ tienen concentraciones de energía en bajas frecuencias (esto es, 500 a 1500 Hz.) y un espectro débil.
- /t/ y /d/ tienen concentraciones de energía en bajas y altas frecuencias (esto es, arriba de 4000 Hz.) y un espectro fuerte.
- /k/ y /g/ tienen concentraciones de energía en frecuencias medias (esto es, 1500 a 4000 Hz.) y un espectro concentrado.

Fricativas

Existen cinco fricativas en el idioma español: /f/, /θ/, /s/, /ĵ/, /x/, y nueve alófonos fricativos.

Existen formantes para sonidos sordos no presentes en los sonidos sonoros. Los formantes principales se encuentran en:

- Para /f/ y alófonos, entre 0-400, 1400-2200, 2900-4000 y 6000- 8000 Hz.
- Para /θ/, /s/ y alófonos en 0-500, 2600-3600 y 5000-8000 Hz.
- Para /ĵ/ y sus alófonos en 0-600 y 2200-3000 Hz. En este caso predomina la intensidad en frecuencias centrales.
- Para /x/ y sus alófonos en 0-900 Hz.

Africativas

El único sonido africativo del español es / ʧ /. Este corresponde a la letra “ch”, y se trata de un fonema sordo palatal. Por ejemplo, en “muchacho” [mucáco].

Para la africativa /ĵ/ se tiene el alófono africativo [dz]. Este es sonoro palatal y se genera cuando /ĵ/ esta precedido por la lateral /l/ o la nasal /n/. Por ejemplo, para "cónyuge" [kóndzuxe].

Nasales

Los tres fonemas nasales /m/, /n/ y /ñ/ son sonoros. El fonema /m/ es bilabial y corresponde a la letra "m". El fonema /n/ es velar y corresponde a la letra "n", sin embargo, puede ser modificado por distintos alófonos. El fonema /ñ/ es bilabial y corresponde a la letra "ñ"; no tiene alófonos.

Semivocales

Existen los fonemas laterales /λ/ (palatal) y /l/ (alveolar) que corresponden a las letras "ll" y "l". El fonema /l/ tiene los alófonos [l] (interdental y [l] (dental). El primero, precede a fonemas fricativos interdentales sordos, por ejemplo en "dulce" [dulθe]. La segunda cuando precede a una consonante dental, por ejemplo en "el toro" [el toro].

También se tienen los fonemas vibrantes, /r/ que corresponde a la letra "r", y /ř/ que corresponde a la letra "r" (cuando está al comienzo de una palabra o sigue a "n" o a "l") o la letra "rr".

II. Aproximaciones al reconocimiento de voz

2.1. Generalidades

De manera general, existen tres aproximaciones para llevar a cabo el reconocimiento de voz, que son:

1. La aproximación fonético-acústica.
2. La aproximación mediante reconocimiento de patrones.
3. La aproximación mediante inteligencia artificial.

La aproximación fonético-acústica se basa en la teoría de que existe un número finito de unidades fonéticas distintivas en el lenguaje hablado, y que las unidades fonéticas son caracterizadas de manera general por un conjunto de propiedades que se manifiestan en las señales de voz, o en su espectro. A pesar de que las propiedades de las unidades fonéticas son altamente variables, ya sea debido al hablante, a la precedencia o a la consecuencia de otra unidad fonética, se asume que las reglas que rigen la variabilidad son bastante sencillas y por lo tanto pueden ser fácilmente aprendidas y aplicadas a un caso práctico.

De lo anterior se determina que el primer paso en la aproximación fonético-acústica sea la segmentación y el etiquetado, ya que involucra la división de la señal en regiones discretas que representan unidades fonéticas, y entonces se le asignan una o varias etiquetas a cada una de las regiones segmentadas, de acuerdo con sus propiedades acústicas. En la actualidad para llevar a cabo el reconocimiento se requiere de un segundo paso, que radica en determinar si la palabra (o cadena de palabras), es consistente con una serie de reglas que se pueden fijar para mejorar el reconocimiento.

La aproximación mediante reconocimiento de patrones es la que básicamente utiliza directamente los patrones de voz sin atender a las características acústicas. Como en

la mayoría de estos métodos, se requieren de dos etapas fundamentalmente, las cuales son:

- el entrenamiento de los patrones de voz.
- el reconocimiento de los patrones mediante la comparación.

El concepto radica en que si se tiene un número suficiente de repeticiones de los patrones que se desean reconocer y estas son empleadas para entrenar una serie de algoritmos, se logrará caracterizar a cada una de las series de patrones de tal forma que sean diferentes unas de otras. A este tipo de caracterización de patrones de voz mediante el entrenamiento se le conoce como clasificación de patrones, debido a que la máquina debe aprender cuales son las propiedades de cada una de las series y las tendrá almacenadas para posteriormente comparar los patrones que se desean reconocer, con los almacenados, arrojando un patrón reconocido.

Se escogió este método de reconocimiento debido fundamentalmente a tres razones:

1. Facilidad de uso. El método es fácil de entender y está muy bien fundamentado en cuanto a su matemática y en cuanto a la teoría de comunicación empleada en los procesos de codificación y decodificación.
2. Es Robusto e invariante a diferentes tipos de voz, usuarios, algoritmos de comparación y reglas de decisión. Esta propiedad convierte al algoritmo en apropiado para un amplio rango de unidades de voz (fonéticamente hablando: palabras, frases, y enunciados), vocabularios, regiones de hablantes, ambiente, condiciones de transmisión, etc.
3. Alto desempeño. Se notará que mediante el reconocimiento de patrones se obtiene consistentemente un alto desempeño de acuerdo con la tecnología empleada.

Antes de describir de manera general un procesador LPC para reconocimiento de voz, es conveniente revisar las razones por las cuales el modelo LPC es tan ampliamente usado. Dentro de estas razones se encuentran las siguientes:

1. El modelo LPC es una buena aproximación de la voz. Esto resulta especialmente cierto para las regiones de estado cuasi-estacionario de voz, en el cual el modelo todo polo de LPC provee una buena aproximación a la envoltura espectral del tracto vocal. Durante las regiones de voz sordas y transitorias, el modelo LPC es menos efectivo que para las regiones sonoras, pero aun muestra un desempeño aceptable para el propósito de reconocimiento de voz.

2. La forma en la que el modelo LPC se aplica al análisis de señales de voz nos lleva a una separación razonable de la fuente y tracto vocal. Como resultado, se vuelve posible una representación parsimoniosa de las características del tracto vocal.

3. El modelo es manejable analíticamente. El método LPC es matemáticamente preciso y es fácil de implementar tanto en software como en Hardware. Los cálculos involucrados en el procesamiento son considerablemente menores a los que se requieren para una implementación digital del modelo de banco de filtros.

4. El modelo trabaja bien en aplicaciones de reconocimiento. La experiencia muestra que el modelo LPC ha sido empleado en un gran número de reconocedores.

2.2. Sistema de reconocimiento de voz, de palabras aisladas y dependiente del locutor utilizando MSBC (basado en el modelo LPC)

A continuación se muestran los módulos necesarios para el diseño y construcción de un sistema de reconocimiento de voz para palabras aisladas y dependiente del locutor, usando MSBC (*Multisection Bookcode*).

Descripción General

El sistema para el reconocimiento de palabras aisladas utiliza las técnicas del procesamiento de voz más comunes.

Los diagramas de bloques que se muestran en las figuras 2.1 y 2.2 ilustran, de forma general, los procesos que se realizan en el entrenamiento y reconocimiento de palabras aisladas, respectivamente.

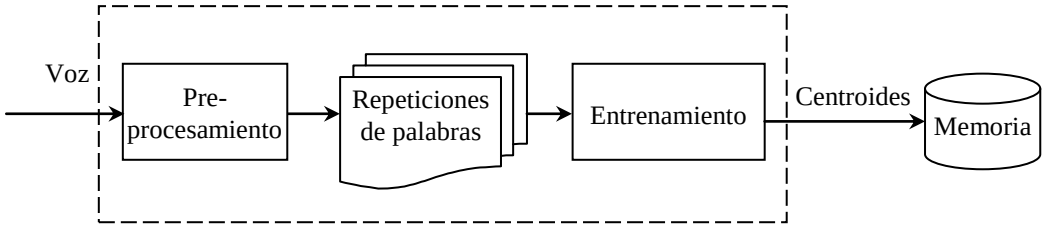


Figura 2.1, Etapa de Entrenamiento

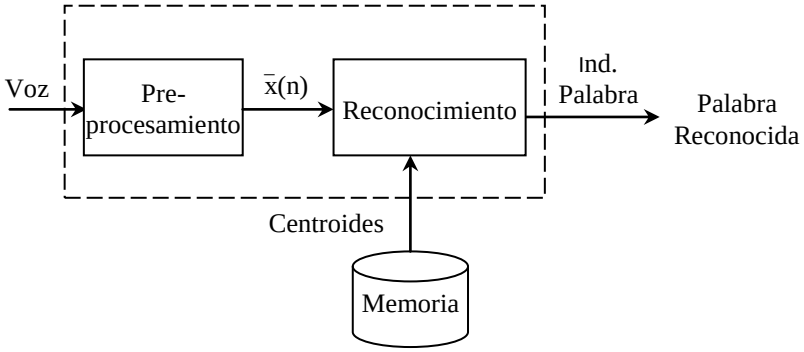


Figura 2.2, Etapa de Reconocimiento

Si observamos la figura 2.1 y la figura 2.2, existe un módulo común en ambas etapas; el preprocesamiento. Por esta razón, el preprocesamiento se manejará como un módulo independiente, para efecto del análisis.

Como se muestra en la figura 2.1, durante la etapa de entrenamiento se capturan diferentes señales de voz, a la que denominaremos *repeticiones de palabras*. A cada repetición se le aplica preprocesamiento para obtener una palabra delimitada. Posteriormente se agrupan todas las palabras recortadas y con ellas se obtienen sus centroides correspondientes. Estos centroides se guardan en memoria.

En la etapa de reconocimiento, como se observa en la figura 2.2, se captura la palabra que se desea reconocer. A esta palabra, también se le aplica preprocesamiento, para recortarla. Utilizando los centroides, calculados durante el entrenamiento y almacenados en la memoria, se realizan las comparaciones necesarias, para efectuar el reconocimiento. El éxito del evento dependerá de ciertos parámetros estadísticos obtenidos mediante el análisis de una población de resultados.

2.2.1. Preprocesamiento

En el preprocesamiento se recibe una señal de voz y se determinan sus límites. A la señal se le aplican dos tipos de filtros: paso bajas y de preénfasis, para eliminar ruidos de altas frecuencias y realzar las frecuencias altas presentes en la voz, respectivamente. Enseguida se divide la señal en tramas de 128 muestras y a cada trama se le aplica una ventana de tipo Hamming, para suavizar el espectro.

Es necesario aclarar que, cuando se inicializa el sistema, se determinan ciertos parámetros (*umbrales*) necesarios para la detección de inicio y fin de la palabra. Estos parámetros se calculan recolectando muestras del ruido presente en el

ambiente circundante. El módulo encargado de realizar este proceso se denomina ruido ambiental.

Finalmente, con las tramas provenientes del ventaneo y con los umbrales originados por el ruido ambiental, se determina el inicio y fin de cada palabra.

Como ya se mencionó, el preprocesamiento es un módulo presente en el entrenamiento y el reconocimiento. Sin embargo existe una diferencia sutil: durante el entrenamiento cada palabra recortada se almacena en un archivo independiente que posteriormente se utilizará para determinar los centroides representativos; mientras que en el reconocimiento la palabra recortada se utiliza directamente para hacer las comparaciones con los centroides. La figura 2.3 muestra el diagrama de bloques para este módulo.

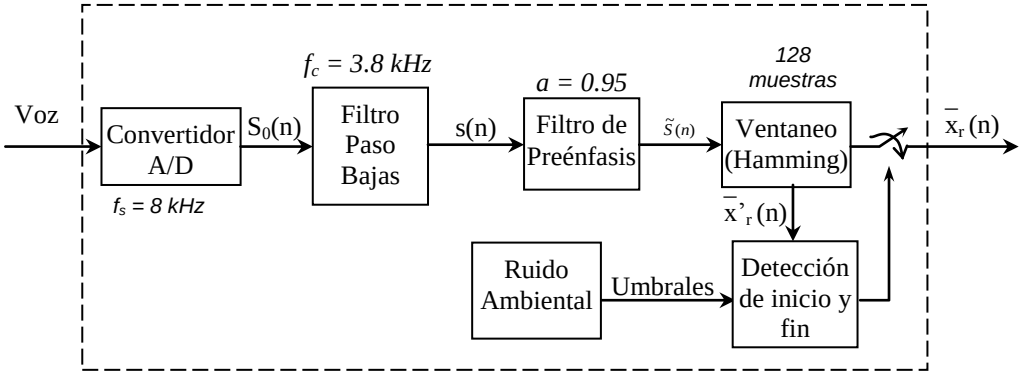


Figura 2.3, Diagrama de bloques para el módulo de preprocesamiento

El preprocesamiento consta de los siguientes módulos: filtro paso-bajas, filtro de preénfasis, ventaneo, detección de ruido ambiental y detección de inicio y fin de la palabra. Estos módulos se describen a continuación:

1. Preénfasis. La señal de voz digitalizada, $s(n)$, se hace pasar por un sistema digital de bajo orden (típicamente un filtro FIR de primer orden), para aplanar espectralmente a la señal y hacerla menos susceptible a efectos de precisión

finitos más tarde durante el proceso. El sistema digital que se usa en el preenfatizador puede ser fijo o lentamente adaptable. Quizá la red de preénfasis más ampliamente usada es el sistema fijo de primer orden:

$$H(z) = 1 - \tilde{a}z^{-1}, \qquad 0.9 \le a \le 1.0 \tag{2.1}$$

En este caso, la salida de la red de preénfasis, $\tilde{s}(n)$, está relacionada con la entrada de la red, $s(n)$, por medio de la ecuación en diferencias.

$$\tilde{s}(n) = s(n) - \tilde{a}s(n-1) \tag{2.2}$$

El valor más común para $\tilde{a}$ es alrededor de 0.95, para el caso fijo.

Un ejemplo de un preenfatizador adaptable de primer orden es la función de transferencia:

$$h(z) = 1 - \tilde{a}_n z^{-1} \tag{2.3}$$

Donde $\tilde{a}_n$ cambia con el tiempo (n) de acuerdo con el criterio de adaptación elegido.

2. Segmentación por bloques de 128 muestras. En esta etapa la señal de voz preenfatizada, $\tilde{s}(n)$, es dividida en bloques de N muestras. El primer segmento comprende las primeras N muestras de la voz. El segundo segmento comienza M muestras después de que inició el primer segmento, de manera similar, el tercer segmento comienza $2M$ muestras después de que inició el primero (o M muestras después de que inició el segundo). Este proceso continúa hasta que toda la señal de voz es incluida en al menos un segmento.

3. Ventaneo. El siguiente paso en el proceso es ventanear cada uno de los segmentos de manera individual para minimizar las discontinuidades de la señal al comienzo y al final de cada segmento. Si se define la ventana como $w(n)$, $0 \le n \le N-1$, entonces el resultado del ventaneo es la señal:

$$\tilde{x}_\ell(n) = x_\ell(n)w(n), \qquad 0 \le n \le N-1 \qquad (2.4)$$

Una ventana empleada típicamente, es la ventana de Hamming, la cual tiene la siguiente forma:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \le n \le N-1 \qquad (2.5)$$

4. Detección de inicio y fin.

El problema de detección de inicio y fin de una palabra en presencia de ruido ambiental es complicado. Para grabaciones en habitaciones a prueba de ruido se puede recurrir al uso de la energía en tiempo corto para la detección, sin embargo en ambientes con ruido es necesario tomar otras consideraciones.

El poder determinar cuál es el inicio y el final de una palabra proporciona ciertas ventajas en los sistemas de reconocimiento, como son:

- Procesar menor cantidad de información.
- Evitar confusiones a causa del ruido o señales de fondo.

Algunos de los problemas que se presentan en la detección:

- Presencia de espurias de ruido que se pueden confundir con la señal.
- Silencios contenidos dentro de las palabras que tienen fonemas plosivos (ej. /t/, /p/, /k/) los cuales pueden confundirse con un falso principio o fin.

- Los fonemas fricativos (ej. /f/, /th/, /h/, etc.), ya que tienen baja energía.
- Sonidos cortos (ej. /t/, /p/, /k/).
- Detección de fonemas nasales al final de la palabra (baja energía y cruces por cero).
- Respiraciones del locutor, que pueden confundirse por su duración.
- Los micrófonos tienen resonancia después de pronunciar una palabra (sobre todo en vocales).

Método para la detección de *inicio - fin* (*Rabiner–Sambur*)

Ante las dificultades para la detección del inicio y fin, se ha desarrollado un método que consiste en considerar las características de los sonidos:

Tabla 2.1, Sonidos Sonoros y No Sonoros.

Sonidos sonoros (*voiced*)	Tienen alto contenido en energía. Ocupan las frecuencias bajas del espectro de la voz humana.
Sonidos no sonoros (*unvoiced*)	Tienen bajo contenido de energía. Ocupan las frecuencias superiores del espectro de la voz humana.

De esta forma se puede implementar un detector que incluya las características de los sonidos sonoros y los no sonoros; análisis de energía, frecuencia y magnitud promedio (o energía en tiempo corto) y detección de cruces por ceros.

2.2.2. Algoritmo de Detección de Inicio y Fin de palabra:

Detección de inicio

i. Por cada trama de 128 muestras, calcular las funciones: magnitud promedio {M[n]} y cruce por ceros {*Z[n]*}; estas funciones se definen a continuación:

$$M_n = \sum_{m=0}^{N-1} \left| x[m] \right| \tag{2.6}$$

$$Z_n = \frac{\sum_{m=0}^{N-2} \left| sign(x[m+1]) - sign(x[m]) \right|}{2N} \tag{2.7}$$

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

$$Ms_n = \{M_1, M_2, \cdots, M_{10}\}$$
$$Zs_n = \{Z_1, Z_2, \cdots, Z_{10}\}$$

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

Umbral	***Nombre del umbral***	***Valor***
UmbSupEnrg	Umbral Superior de Energía	0.5*máx.{M_n}
UmbInfEnrg	Umbral Inferior de Energía	$\mu_{Ms} + 2*\sigma_{Ms}$
UmbCruCero	Umbral de cruces por cero	$\mu_{Zs} + 2*\sigma_{Zs}$

iv. Recorrer la función $\boldsymbol{M_n}$ incrementando en una unidad a ***n*** de 11 hasta que $\boldsymbol{M_n}$ > ***UmbSupEnrg***. En este punto estamos garantizando presencia de señal. A este punto lo marcaremos como ***In***.

v. Resulta lógico pensar que el inicio de la señal se encuentra en algún punto anterior a ***In***, por lo que ahora recorremos la función $\boldsymbol{M_n}$ desde ***n*** = ***In*** hasta que $\boldsymbol{M_n}$ < ***UmbInfEnrg***. Este punto lo marcaremos como ***Ie*** y lo reconocemos

tentativamente como el inicio de la señal, determinado por la función de magnitud.

vi. Ahora decrementamos ***n*** desde ***n*** = ***Ie*** hasta ***n*** = ***Ie – 25*** o en su defecto ***n*** = ***11***, verificando si se presenta alguna de las siguientes condiciones en la función de cruces por cero, ya que lo que ahora buscamos es la posibilidad de que un sonido *no sonoro* preceda a un sonido *sonoro*:

- Sí {$\boldsymbol{Z_n}$ < ***UmbCruCero***} significa que no encontramos alguna porción de la señal con aumento importante de frecuencia en 25 ventanas anteriores, por lo tanto el inicio es ***Ie***.

- Sí encontramos que {$\boldsymbol{Z_n}$ > ***UmbCruCero***} menos de tres veces seguidas significa que solo fue una espiga de ruido, el punto de inicio sigue siendo ***Ie***.

Si encontramos que {$\boldsymbol{Z_n}$ > ***UmbCruCero***} al menos tres veces seguidas hemos encontrado un sonido *no sonoro*, entonces buscamos el punto ***n*** para el cual {$\boldsymbol{Z_n}$ > ***UmbCruCero***} la primera de las más de tres veces, es decir, el punto para el cual la función $\boldsymbol{Z_n}$ sobrepasa el umbral, indicando el comienzo del sonido *no sonoro* y desplazamos el inicio de la palabra de ***Ie*** a ***Iz***.

Detección de fin

Para la detección de fin de la palabra hacemos lo mismo pero en sentido inverso a partir del punto (iv) de la sección anterior, como si detectáramos un inicio con la señal invertida en el tiempo.

2.2.3. Entrenamiento

Para el entrenamiento se deben capturar diversas palabras recortadas por medio del módulo de preprocesamiento, mismas que representan las diferentes *repeticiones*. A partir de estas repeticiones se obtiene un conjunto de centroides que son almacenados en la memoria.

Los archivos de palabras, previamente almacenados, son utilizados para aplicarles varios procesos a cada uno. Primero se determinan sus vectores de autocorrelación para obtener los coeficientes de predicción lineal. Enseguida, estos coeficientes son segmentados en los distintos fonemas. Se sigue el mismo procedimiento para todas las repeticiones de una misma palabra. Posteriormente, una vez que se tienen segmentadas todas las repeticiones, se agrupan en conjuntos del mismo segmento. Para cada conjunto se obtienen centroides, que representarán al segmento. Finalmente, estos centroides son almacenados en memoria. La figura 2.4 esquematiza el proceso completo del entrenamiento.

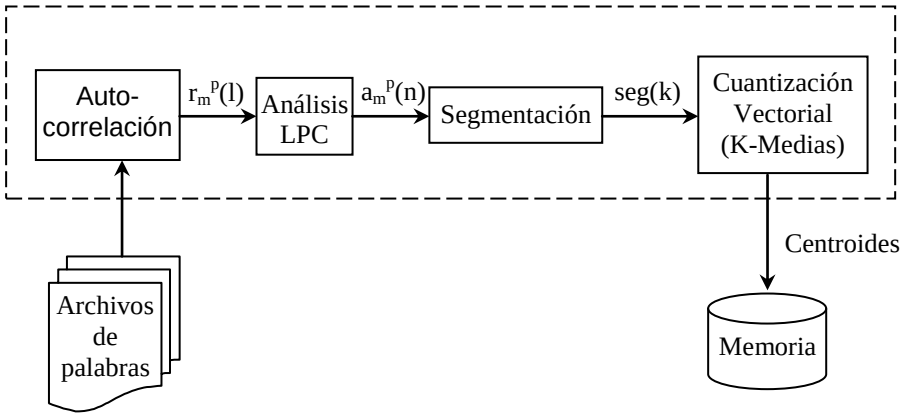


Figura 2.4, Módulo de Entrenamiento

El entrenamiento es la etapa que nos permite obtener los patrones de comparación para las señales de voz. Consta de los siguientes módulos: autocorrelación, análisis LPC, segmentación y cuantización vectorial.

El método de autocorrelación

5. Análisis de Autocorrelación. Cada uno de los segmentos ventaneados de la señal es autocorrelacionado para obtener:

$$r_\ell(m) = \sum_{n=0}^{N-1-m} \tilde{x}_\ell(n)\tilde{x}_\ell(n+m), \qquad m = 0,1,..., p, \qquad (2.8)$$

Donde el valor más alto de autocorrelación, p, es el orden del análisis LPC. Típicamente se emplean valores de p entre 8 y 16, siendo $p = 8$ el valor que más comúnmente se utiliza. Paralelamente, un beneficio que se obtiene del análisis de autocorrelación es que el término cero del vector de autocorrelación, $R_\ell(0)$, representa la energía del segmento, ℓ^{th}. El conocer la energía es importante para los sistemas de detección de voz que pueden emplearse.

2.2.4. El modelo LPC

La idea básica detrás del modelo es que una muestra dada de voz, en el tiempo n, s(n), se puede aproximar como una combinación lineal de las p muestras pasadas, tal como:

$$s(n) \approx a_1 s(n-1) + a_2 s(n-2) + \cdots + a_p s(n-p), \qquad (2.9)$$

Donde los coeficientes $a_1, a_2, ..., a_p$ se consideran constantes a lo largo del segmento de voz analizado. Para convertir a la ecuación (2.9) en una igualdad incluimos un término que representa la excitación, $Gu(n)$, dando como resultado:

$$s(n) = \sum_{i=1}^{p} a_i s(n-i) + Gu(n), \qquad (2.10)$$

Donde u(n) es una excitación normalizada y G es la ganancia de la excitación. Expresando la ecuación (2.10) en el dominio z, tenemos:

$$S(z) = \sum_{i=1}^{p} a_i z^{-i} S(z) + GU(z), \qquad (2.11)$$

Lo que nos lleva a obtener la función de transferencia:

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 - \sum_{i=1}^{p} a_i z^{-i}} = \frac{1}{A(z)}. \qquad (2.12)$$

La interpretación de la ecuación (2.12) es dada en la figura 2.5

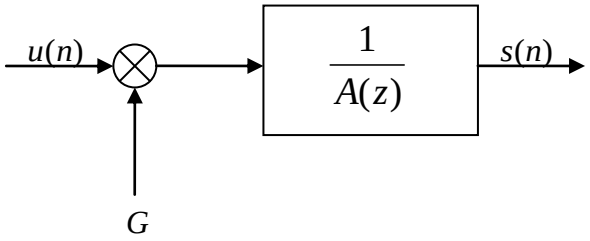


Figura 2.5, Modelo de predicción lineal

En ella se muestra la fuente de excitación normalizada, u(n), escalada por la ganancia G, y actuando como entrada de un sistema todo polo, $H(z) = \frac{1}{A(z)}$, para generar la señal de voz, s(n). Teniendo en cuenta que la función de excitación actual para voz es ya sea un tren de pulsos cuasi-periódicos (para sonidos sonoros) o una fuente de ruido aleatorio (en el caso de sonidos sordos), el modelo adecuado sería el que se muestra en la figura 2.6.

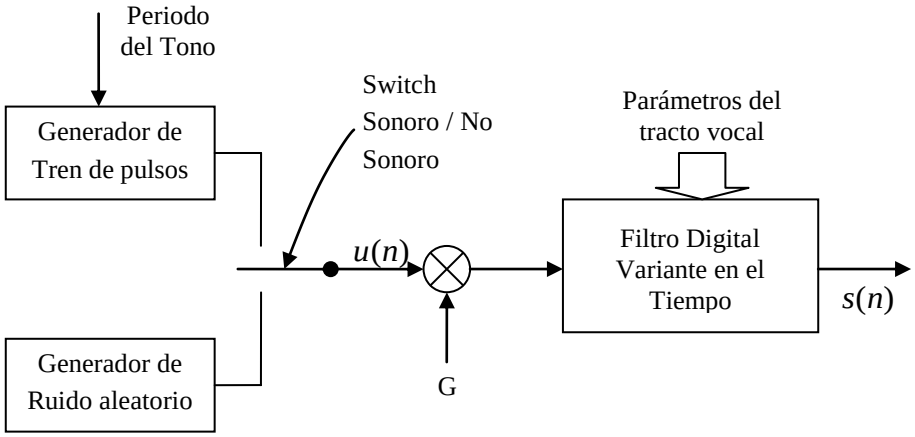


Figura 2.6, Modelo sintetizador de voz basado en el modelo de LPC

Ecuaciones de Análisis LPC

Basándonos en el modelo de la figura 2.5, la relación exacta entre s(n) y u(n) es:

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n) \tag{2.13}$$

Considerando la combinación lineal de las muestras pasadas de voz como la estimación $\tilde{s}(n)$, definida como:

$$\tilde{s}(n) = \sum_{k=1}^{p} a_k s(n-k) \tag{2.14}$$

Ahora conformamos el error de predicción, e(n), definido como:

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \tag{2.15}$$

Con la función de transferencia:

$$A(z) = \frac{E(z)}{S(z)} = 1 - \sum_{k=1}^{p} a_k z^{-k} \qquad (2.16)$$

Claramente, cuando s(n) es generado por un sistema lineal como el que se muestra en la figura 2.5, entonces el error de predicción, e(n), será igual a $Gu(n)$, la excitación escalada.

El problema del análisis de predicción lineal radica en determinar el conjunto de coeficientes de predicción, $\{a_k\}$, directamente de la señal de voz, de manera que las propiedades espectrales del filtro digital de la figura 2.6 correspondan a las formas de onda de la voz dentro de la ventana de análisis. Dado que las características espectrales de la voz varían en el tiempo, los coeficientes de predicción en un tiempo dado, n, deben ser estimados de un segmento corto de señal de voz ocurriendo alrededor del tiempo n.

Los coeficientes de predicción (a_k's) son determinados minimizando la suma de diferencias cuadradas sobre un intervalo finito (error de predicción promedio) entre las muestras actuales de voz y las predichas linealmente, esto es:

$$E_n = \sum_m e_n^2(m) = \sum_m (s_n(m) - \tilde{s}_n(m))^2 = \sum_m \left[s_n(m) - \sum_{k=1}^{P} a_k s_n(m-k) \right]^2 \qquad (2.17)$$

Donde, E_n es el error de predicción promedio en tiempo corto, $e_n(m)$ es error de predicción, $s_n(m)$ es la señal actual multiplicada por una ventana y $\tilde{s}_n(m)$ es la muestra predicha. Con fines de simplificar la notación, se ha usado $s_n(m)$ para la señal en tiempo corto, determinada por:

$$s_n(m) = \begin{cases} s(m+n) \cdot w(m), & 0 \le m \le N-1 \\ 0, & \textit{para otro caso} \end{cases} \qquad (2.18)$$

Donde *w(m)* es una ventana de longitud *N*.

Por otro lado, para encontrar los valores de a_k que minimizan a E_n en (2.17), se calcula:

$$\frac{\partial E_n}{\partial a_i} = 0, \quad para \quad i = 1,2,..., p \tag{2.19}$$

La solución de (2.19) se puede obtener por varios métodos, uno de los más usados es el método de la autocorrelación. La solución de este método da como resultado un sistema de ecuaciones lineales con *p* incógnitas, las cuales se pueden expresar de forma matricial de la manera siguiente:

$$\begin{bmatrix} R_n(0) & R_n(1) & . & . & R_n(p-1) \\ R_n(1) & R_n(0) & . & . & R_n(p-2) \\ . & . & . & . & . \\ . & . & . & . & . \\ R_n(p-1) & R_n(p-2) & . & . & R_n(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ . \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ . \\ . \\ R_n(p) \end{bmatrix}$$

Figura 2.7, Sistema de ecuaciones lineales con p incógnitas.

Donde R_n es la autocorrelación de la señal en tiempo corto y los α's son los coeficientes de predicción lineal (LPC's) que resuelven el sistema.

La matriz *p x p* con los valores de la autocorrelación es una matriz toeplitz (es decir, que es simétrica y con todos los elementos de la diagonal iguales) y por lo tanto puede resolverse eficientemente mediante el empleo de procedimientos bien conocidos, como el algoritmo de *Levinson-Durbin.*

6. Análisis LPC. El siguiente paso en el proceso es el análisis LPC, el cual convierte cada uno de los segmentos de $p+1$ autocorrelaciones en un conjunto de parámetros de LPC, en el cual el conjunto puede ser los coeficientes LPC. El método formal para convertir de coeficientes de autocorrelación a coeficiente LPC es el algoritmo conocido como método de Durbin y se puede obtener mediante la ejecución del siguiente algoritmo (por conveniencia se omitió el subíndice ℓ en $R_\ell(m)$):

$$E^{(0)} = r(0) \tag{2.20}$$

$$k_i = \left\{ r(i) - \sum_{j=1}^{L-1} \alpha_j^{(i-1)} r(|i-j|) \right\} \Big/ E^{(i-1)}, \qquad 1 \le i \le p \tag{2.21}$$

$$\alpha_i^{(i)} = k_i \tag{2.22}$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \tag{2.23}$$

$$E^{(i)} = (1-k_i^2) E^{(i-1)} \tag{2.24}$$

Donde la sumatoria en la ecuación 2.21se omite para i=1. El conjunto de ecuaciones son resueltas recursivamente para i=1,2,…,p, y la solución final se da como:

$$a_m = \text{Coeficientes LPC} = \alpha_m^{(p)}, \qquad 1 \le m \le p \tag{2.25}$$

2.2.5. Distancias y medidas de distorsión

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

Suponer que se tienen dos vectores característicos, $\boldsymbol{x}$ e $\boldsymbol{y}$, definidos en un espacio vectorialχ. Se define una *métrica* o *función de distancia, d,* en el espacio vectorialχ, como una función de valor real, sobre el producto cartesiano χ xχ, que cumpla las siguientes condiciones:

1. $0 \leq d(\boldsymbol{x}, \boldsymbol{y}) < \infty$, para $\boldsymbol{x}, \boldsymbol{y} \in \chi$ y $d(\boldsymbol{x},\boldsymbol{y}) = 0$ sí y solo sí $\boldsymbol{x} = \boldsymbol{y}$;
2. $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ para $\boldsymbol{x}, \boldsymbol{y} \in \chi$;
3. $d(\boldsymbol{x}, \boldsymbol{y}) \leq d(\boldsymbol{x}, \boldsymbol{z}) + d(\boldsymbol{z}, \boldsymbol{y})$ para $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \chi$.

Además, una función de distancia se denomina invariante si

4. $d(\boldsymbol{x} + \boldsymbol{z}, \boldsymbol{y} + \boldsymbol{z}) = d(\boldsymbol{x},\boldsymbol{y})$

Las primeras tres propiedades comúnmente son conocidas como positividad (no negatividad), simetría y desigualdad del triángulo, respectivamente. Una métrica que contenga estas propiedades, permite un alto grado de manejo matemático. Sí una medida de distancia, *d,* satisface solo la propiedad de positividad, se le denomina medida de distorsión, particularmente cuando los vectores son representaciones del espectro de la señal.

Para el procesamiento de voz es importante considerar que la definición (o elección), de la medida de distancia, es significativamente subjetiva. Una medida matemática de la distancia, para ser utilizada en el procesamiento de voz, debe tener una alta correlación entre su valor numérico y su distancia subjetiva aproximada, para evaluar una señal real de voz. Para el reconocimiento de voz, la consistencia psicofísica (los diferentes matices que se le pueden imprimir a una misma palabra o frase), que se desea medir con la distancia, obliga a que se encuentre una medida matemática ajustada por necesidad, a las características lingüísticas conocidas. Estos requisitos tan subjetivos no pueden ser satisfechos con medidas de distancia que proporcionen manejo matemático. Un ejemplo es la tradicional medida del Error Cuadrático, $d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^2$.

Dado que existe una enorme dificultad al querer cumplir simultáneamente ambos objetivos (subjetividad y manejo matemático), algún compromiso es inevitable. Por consiguiente, y dado que se necesita manipular matemáticamente las propiedades de esa medida de distancia, se necesita probar que estas propiedades subjetivas son lo suficientemente buenas como para lograr el reconocimiento de voz. Por otra parte se hablará de "medidas de distorsión" en vez de "métricas" debido a que se relajan las condiciones de simetría y desigualdad del triángulo. No se debe utilizar el término *distancia* en sentido estricto, acorde a la definición de arriba; por otro lado, se debe mantener la costumbre de la literatura de voz, donde el término *distancia* es análogo, a las medidas de distorsión [Rabiner, 1993].

Existen varios tipos de medidas de distorsión, cada una con sus características especiales. Entre ellas tenemos: *Distancia Euclidiana Cuadrática, Distorsión del Error Cuadrático Medio, Distorsión del Error Cuadrático Ponderado, Distancia de Itakura*, etc.

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

$$d(X_{1,}, X_2) = \left\| X_1 - X_2 \right\|^2 = \sum_{j=1}^{N} \left(X_{1j} - X_{2j} \right)^2 \qquad (2.26)$$

- ***Distorsión del Error Cuadrático Medio***. La distorsión del Error Cuadrático Medio (MSE) es otra de las medidas más utilizadas y se define como:

$$\mathrm{d}(\mathrm{X}_1, \mathrm{X}_2) = \frac{1}{\mathrm{N}} (\mathrm{X}_1 - \mathrm{X}_2)^{\mathrm{T}} (\mathrm{X}_1 - \mathrm{X}_2) = \frac{1}{\mathrm{N}} \sum_{\mathrm{j}=1}^{\mathrm{N}} (\mathrm{X}_{1\mathrm{j}} - \mathrm{X}_{2\mathrm{j}})^2 \qquad (2.27)$$

  En la cual la distorsión está definida por cada dimensión. La popularidad del MSE se basa en su simplicidad y seguimiento matemático.

- ***Distorsión del Error Cuadrático Medio Ponderado***. Otra medida de distorsión es el Error Cuadrático Medio Ponderado. En el MSE la medida asume que las distorsiones contribuyen cuantizando los diferentes parámetros $\{X_{1j}\}$ de igual forma. Y de manera general, se pueden introducir pesos diferentes con el fin de aportar ciertas contribuciones a la distorsión, dependiendo del parámetro. El MSE ponderado general se define como:

$$d(X_1, X_2) = (X_1 - X_2)^T W(X_1 - X_2) \tag{2.28}$$

Donde W es una matriz de ponderación definida, simétrica y positiva, y los vectores X_1 y X_2 son tratados como vectores columna.

Cada una de las medidas de distorsión mencionadas anteriormente, resultan simétricas en sus argumentos X_1 y X_2 y pueden ser aplicadas a las características derivadas del análisis de producción lineal de la voz; el uso de algunas presenta ciertas desventajas, tal es el caso de la distancia euclidiana que aunque resulte fácil de calcular, no todas sus características tienen el mismo significado perceptible.

Por lo anterior, en ciertos casos resulta conveniente y efectivo escoger una matriz de ponderación $W(X_1)$ que dependa explícitamente del vector X_1, para así obtener una medida de distorsión perceptiblemente motivada. En este caso, la distorsión:

$$d(X_1, X_2) = (X_1 - X_2)^T W(X_1)(X_1 - X_2) \tag{2.29}$$

es asimétrica.

- ***Distancia de Itakura***. En muchos casos del procesamiento de voz, es necesario tener otra medida de la distancia que existe entre dos vectores LPC. La distancia Euclidiana no es apropiada para medir los parámetros de dos LPC's individuales, en vectores que estén relacionados. Esto es debido a que los vectores LPC dependen del peso de la matriz de autocorrelación correspondiente a cada LPC.

  La medida de distancia más comúnmente utilizada para este propósito es la propuesta por *Itakura*. Esta distancia se deriva utilizando una interpretación intuitiva del rango de predicción en el error de la energía. Fue obtenida originalmente, utilizando la máxima probabilidad existente entre argumentos similares. La distancia de *Itakura* es, probablemente, la medida de distorsión más empleada para encontrar la similitud entre dos vectores LPC [Deller, 1987].

  Esta distancia se define de la siguiente forma: sean $R_y x$ y $R_y y$ las matrices de autocorrelación multiplicadas por las señales de voz de entrada y de comparación, respectivamente. Sean así mismo, $xR_y x^T$ la energía de salida del filtro inverso, tomado como referencia con la entrada, y $yR_y y^T$ la energía mínima posible de salida, del filtro LPC, con respecto a la entrada de la voz.

  Entonces tenemos que la distancia de *Itakura* se obtiene mediante la ecuación (2.30).

$$d(x,y) = \log\left(\frac{xR_y x^T}{yR_y y^T}\right) \tag{2.30}$$

  Donde:

$$yR_y y^T = \begin{bmatrix} -1 & a_1 & a_2 & \cdots & a_p \end{bmatrix} \begin{bmatrix} r(0) & r(1) & r(2) & \cdots & r(p) \\ r(1) & r(0) & r(1) & \cdots & r(p-1) \\ r(2) & r(1) & r(0) & \cdots & r(p-2) \\ \vdots & \vdots & \vdots & & \vdots \\ r(p) & r(p-1) & r(p-2) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} -1 \\ a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} \quad (2.31)$$

O lo que es lo mismo:

$$yR_y y^T = \sum_{i=0}^{P} a_i \sum_{n=0}^{P} r_n(|i-n|) a_n \quad (2.32)$$

Donde $y = \begin{bmatrix} -1 & a_1 & a_2 & \cdots & a_p \end{bmatrix}$

De esta forma, si ***y*** es el vector aumentado de coeficientes LPC's de "referencia" o de "plantilla" $\begin{bmatrix} -1 & a_1 & a_2 & \cdots & a_p \end{bmatrix}$, y sea ***x*** el vector aumentado de coeficientes LPC's "desconocido" u "observado" $\begin{bmatrix} -1 & a'_1 & a'_2 & \cdots & a'_p \end{bmatrix}$ entonces:

$xR_y x^T$ = Es la energía del filtro inverso formado con la señal de entrada de voz

$yR_y y^T$ = La energía de salida mínima posible para el filtro de predicción lineal con la entrada de voz

Por lo tanto (2.30) también se puede escribir como:

$$d(x,y) = \log\left(\frac{E_x}{E_y}\right) \quad (2.33)$$

2.2.6. Cuantización Vectorial (VQ).

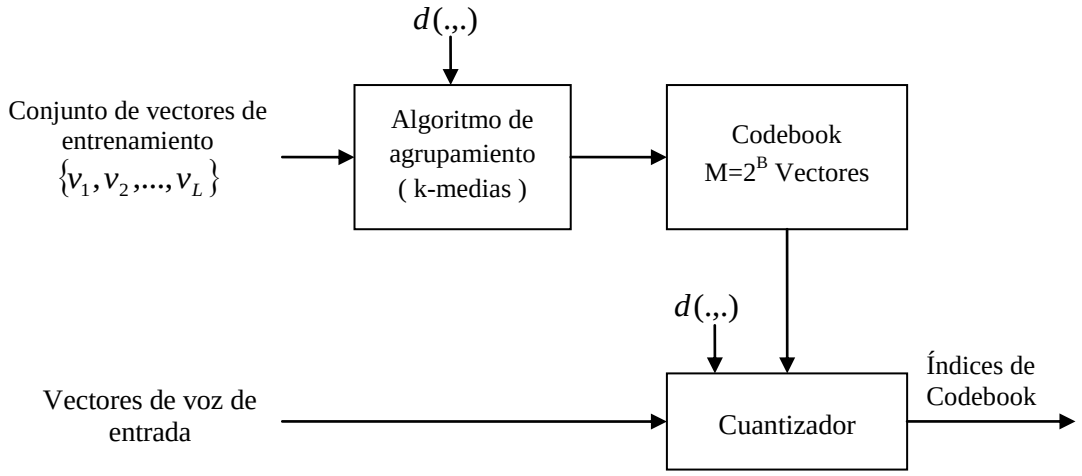


Figura 2.8, Diagrama de bloques de un VQ básico, entrenamiento y estructura de clasificación

Tomando en cuenta la figura 2.8, partimos de un conjunto de vectores que pertenecen a un espacio *K*-dimensional, asumiendo que ***x*** es un vector perteneciente a ese conjunto cuyos componentes

[$\mathbf{x}_i$, $1 \le i \le K$] son variables aleatorias reales y de amplitud continua. Un cuantizador vectorial Q, de dimensión *K* y tamaño *N*, es una transformación de un vector **x,** del espacio euclidiano de dimensión R^K, en un conjunto finito *C* que contiene *N* salidas o puntos de reproducción, llamados *code vectors* (vectores de código):

$$Q : \mathrm{R}^{\mathrm{K}} \rightarrow \mathrm{C}, \quad \mathrm{C} = \{\mathbf{y}_1,...,\mathbf{y}_{\mathrm{N}}\} \quad \mathbf{y}_{\mathrm{i}} \in \mathrm{R}^{\mathrm{K}} \quad \forall \quad \mathrm{i} \in \mathrm{I} \equiv \{0,1,...,\mathrm{N} \tag{2.34}$$

El conjunto *C* es llamado *code book* (libro de códigos) y tiene un tamaño *N*, esto significa que tiene *N* distintos elementos, cada uno de ellos dentro del espacio R^k.

Asociado a cada cuantizador vectorial de *N* puntos, existe una *partición* de R^k en *N* regiones o *celdas*, R_i para $i \in I$. La *i*-ésima celda está definida por:

$$R_i = \{ \mathbf{x} \in R^K : Q(\mathbf{x}) = \mathbf{y}_i \} \qquad (2.35)$$

Algunas veces llamada *imagen inversa* o *pre-imagen* de $\mathbf{y}_i$ dentro del mapeo Q y denotada de forma más consistente por $R_i = Q^{-1}(y_i)$.

De la definición de celdas, tenemos que:

$$\bigcup_i R_i = R^K \quad \text{y} \quad R_i \bigcap R_j = \varnothing \quad \text{para} \quad i \neq j \qquad (2.36)$$

Donde las celdas forman una partición de R^k.

La tarea de codificación de un cuantizador vectorial es examinar cada vector de entrada **x** e identificar a qué celda *k*-dimensional del espacio R^k pertenece. El codificador vectorial simplemente identifica el índice *i* de la región y el decodificador vectorial genera el vector del código $\mathbf{y}_i$ que representa a esta región [Gersho, 1997].

El conjunto **y** es conocido como diccionario de reconstrucción o simplemente diccionario, donde *N* es el tamaño del diccionario y $\{\mathbf{y_i}\}$ es el conjunto de vectores del código. Los vectores $\mathbf{y_i}$ son conocidos también en la literatura de reconocimiento de patrones, como los patrones de referencia o plantillas. El tamaño *N* del diccionario, también se conoce como número de niveles, término proveniente de la cuantización escalar. De esta forma, se puede hablar de un diccionario de *N* niveles. Al proceso de creación del diccionario, también se le conoce como entrenamiento o población del diccionario.

El modelo de operación de este codificador, se define de forma similar al caso escalar, la *función de selección, $S_i(\mathbf{x})$,* como *indicador* o *función miembro $1_{Ri}(\mathbf{x})$* para la celda R_i de la partición, esto es:

$$S_i(\mathbf{x}) = \begin{cases} 1 & \text{si} \quad \mathbf{x} \in R_i \\ 0 & \text{c.o.c.} \end{cases} \tag{2.37}$$

La operación de un cuantizador vectorial puede ser representada como:

$$Q(\mathbf{x}) = \sum_{i=1}^{N} \mathbf{y}_i S_i(\mathbf{x}) \tag{2.38}$$

Una descomposición estructural (como la mostrada anteriormente), es particularmente evaluable para encontrar un algoritmo efectivo, durante la implementación de la cuantización vectorial [Gersho, 1997].

Agrupamiento

El agrupamiento es la forma en que se realiza la cuantización vectorial; consiste en lo siguiente: a partir de un conjunto de *N* muestras $\chi = \{X_1, X_2, X_3, ..., X_N\}$, se intentan separar en *K* subconjuntos disjuntos $\chi_1, \chi_2, \chi_3, ..., \chi_K$. En donde cada subconjunto representa a un grupo (*clúster*) y en el cual, las muestras pertenecientes tienen una mayor similitud entre sí, en comparación a las muestras de cualquier otro grupo.

Existen varios algoritmos de agrupamiento, entre los que tenemos: *simple, distancia máxima,*
K-Medias, ISODATA y LBG; estos dos últimos son variantes del agrupamiento *K-Medias*, pero con mayor complejidad.

Definición del algoritmo de K-medias

Se describe a continuación el algoritmo de agrupamiento *K-Medias.* Su criterio de función es:

$$J_e = \sum_{j=1}^{K} \sum_{x\ en\ \chi_j} d(\mathbf{x}, \mathbf{z}_j) \qquad (2.39)$$

Donde:

K = número de grupos

z_j = centro del grupo (centroide) para el grupo j

χ_j = subconjunto de muestras asignadas al grupo j

$d(x, z_j)$ = Es la distancia de *Itakura* entre el vector x y el centroide z_j

2.2.7. Algoritmo de K-medias:

1) Escoger K centroides iníciales $\mathbf{z_1}(1)$, $\mathbf{z_2}(1)$, ..., $\mathbf{z_k}(1)$.

2) En la iteración l, asignar las muestras a los grupos:
Asignar:

$$\mathbf{x} \quad a \quad \chi_i(l) \quad si \quad d(\mathbf{x}, \mathbf{z}_i(l)) \quad \leq \quad d(\mathbf{x}, \mathbf{z}_j(l)) \quad j = 1,2,...,K \quad j \neq i$$

Donde $d(x, z_j(l))$ es la distancia (ó distorsión) de *Itakura*

3) Calcular los nuevos centros de grupo:

$$\mathbf{z}_i(l+1) = \frac{1}{N_i} \sum_{x \in \chi\chi_i(l)} \mathbf{x} \qquad i = 1,2,...,K$$

4) Donde N_i es el número de muestras asignadas a $\chi_j(l)$.

Si $\mathbf{z_i}(l+1) = \mathbf{z_i}(l)$ para $i = 1, 2, ..., K$, el algoritmo ha convergido y debe terminarse. En caso contrario, regresar al paso 2.

Una característica de este algoritmo es que los centroides o cuantizadores obtenidos, no son los óptimos globales; es decir, se obtienen cuantizadores óptimos locales. Estos dependen de varios factores, como son: asignación inicial de centroides (principalmente), orden de la toma de muestras, propiedades geométricas de los datos, medida de distorsión empleada, etc.

Una forma de poder alcanzar los óptimos globales, es probar con una gran variedad de centroides iníciales y seleccionar los cuantizadores finales que tengan la menor distorsión, con respecto a los vectores a los cuales representan. Solución poco factible porque existen un gran número de combinaciones para los cuantizadores iníciales. Otra forma es elegir los centroides iníciales de forma aleatoria para buscar una distribución homogénea [Deller, 1987].

2.2.8. Reconocimiento

El reconocimiento recibe las tramas de una señal de voz proveniente del preprocesamiento para efectuar comparaciones con los centroides y obtener un indicador a la palabra reconocida.

Cada trama recibida es utilizada para calcular su autocorrelación y sus coeficientes de predicción lineal. Este proceso se repite hasta que el preprocesamiento detecta el fin de la palabra. Una vez detectado el final, se segmentan los vectores LPC y los vectores de autocorrelación correspondientes, de forma lineal y con segmentos iguales al patrón de comparación. Con cada segmento se realiza una comparación, utilizando la distancia de *Itakura*, con los centroides que correspondan al mismo segmento.

El resultado de esta comparación, aunado con ciertos parámetros estadísticos, determina el éxito o fracaso del reconocimiento. La figura 2.9 muestra el diagrama de bloques del reconocimiento.

El reconocimiento permite identificar una señal de voz similar a ciertos patrones definidos previamente. Consta de los siguientes módulos: autocorrelación, análisis LPC, segmentación y comparación.

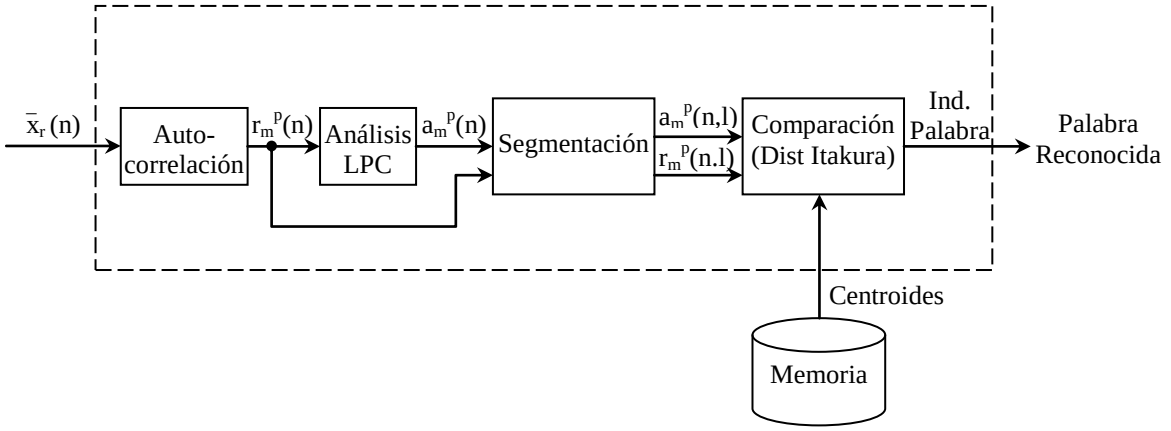


Figura 2.9 Módulo de Reconocimiento

Apéndice

Elementos de una Implementación de Cuantización Vectorial

Para construir un VQ codebook e implementar un procedimiento de análisis de VQ, necesitamos lo siguiente:

1. Un conjunto de vectores de análisis espectral, v1, v2,..., vL, mismos que conforman un conjunto de vectores de entrenamiento. El conjunto de entrenamiento es usado para crear un conjunto de codebook vectors óptimos para representar la variabilidad espectral observada en el conjunto de entrenamiento. Si denotamos el tamaño de VQ codebook como $M = 2^B$ vectores, entonces requeriremos $L >> M$ de manera que seamos capaces de encontrar el mejor conjunto de M codebook vectors en una forma robusta.

2. Una medida de similitud, o distancia, entre un par de vectores de análisis espectral, de tal manera que seamos capaces de agrupar los conjuntos de

vectores de entrenamiento tan bien como asociar o clasificar vectores espectrales arbitrarios en entradas de codebooks únicas. Denotamos la distancia espectral, como $d(v_i, v_j)$, entre dos vectores v_i y v_j como d_{ij}.

3. Un procedimiento para el cálculo del centroide. En la bases de la partición que clasifica a los L conjuntos de vectores en M grupos, escogemos los M conjuntos de vectores como centroides de cada uno de los M grupos.

4. Un procedimiento de clasificación para el análisis espectral de vectores de voz que escoja el codebook vector más cercano al vector de entrada y utilice el índice de codebook como la representación espectral resultante.

III. Herramientas de Desarrollo: DSP, Microcontrolador y Software de Desarrollo.

3.1 Introducción

En las últimas décadas hemos observado que las computadoras son muy capaces, principalmente en dos áreas: el manejo de datos y los cálculos matemáticos que se emplean en la ciencia, y particularmente en la ingeniería. Sin embargo aun resulta costoso y por lo tanto poco eficiente construir un procesador que emplee toda su capacidad en ambas tareas. Un DSP tiene una arquitectura tal que nos permite manejar un gran número de muestras y realizar una gran cantidad de operaciones con ellas en poco tiempo y con un bajo consumo de energía; por ello conviene utilizar un DSP para realizar el reconocimiento.

3.2 Características Principales de la tarjeta de desarrollo del DSP

Debido a las ventajas que representa utilizar un DSP para llevar a cabo el procesamiento, se empleó la tarjeta de desarrollo fabricada por Texas Instruments C6711 DSK (C6711 DSP Starter kit). En la figura 3.1 se observa la tarjeta de desarrollo DSK.

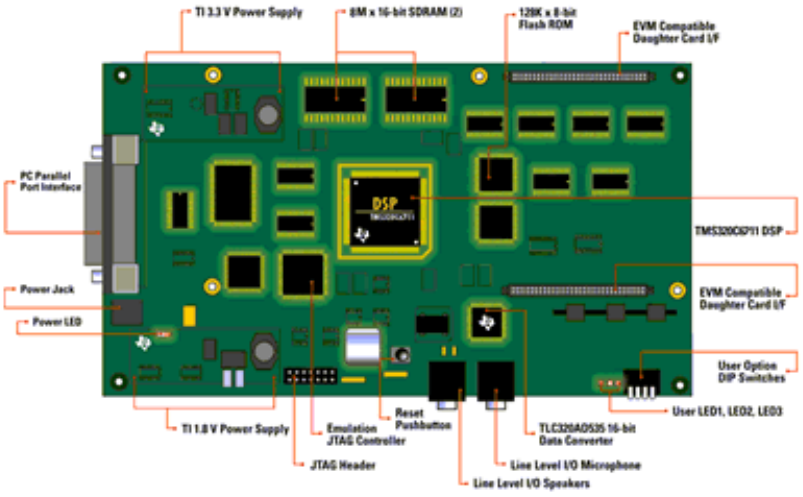


Figura 3.1, Tarjeta de desarrollo DSK.

El DSK C6711 tiene las siguientes características:

- Cuenta con el DSP TMS320C6711, que trabaja a una frecuencia de 150 MHz y es capaz de realizar 900 millones de operaciones de punto flotante por segundo, gracias a que soporta "pipeline".
- Soporta un reloj dual: 150 MHz en el CPU y 100 MHz en la interfaz de memoria externa.
- Cuenta con una interfaz de puerto paralelo a fin de conectarse con una PC.
- Cuenta con un banco de memoria RAM (SDRAM) de 4M x 32 bits de palabra, que opera a 100 MHZ.
- Tiene 128k bytes de memoria flash.
- Puerto de E/S mapeado en memoria de 8 bits.
- Emulación JTAG embebida por medio del puerto paralelo y soporte externo XDS510.
- La Interfaz del puerto Host es capaz de acceder a toda la memoria del DSP por medio del puerto paralelo.
- La tarjeta DSK cuenta con un códec de canal dual voz/datos TLC320AD535, mismo que está conectado con el McBSP del DSP. Este circuito de interfaz analógica tiene las siguientes características:
    - Procesamiento de señal de 16 bits.
    - Canales independientes de voz y datos (solo se utiliza el canal de voz).
    - Frecuencia máxima de muestreo de 11.025 kHz (por defecto 8 kHz).
    - Amplificadores de ganancia programable.
    - Manejador de audífonos de 60 ohms con amplificador de ganancia programable.
    - Soporte para micrófono (con bias).
- Cuenta con dos conectores que permiten acoplarle una tarjeta de expansión (daughtercard), misma que puede ser empleada para proveer alguna aplicación específica a la tarjeta DSK.

3.3 Características del DSP TMS320C6711

El TMS320C67x forma parte de la familia TMS320C6000 de DSP's de punto flotante. El C6711 está basado en el alto desempeño y en la arquitectura avanzada de palabra de instrucción muy larga desarrollada por Texas Instruments (TI). En la figura 3.2 podemos ver su diagrama de bloques.

Esta familia de dispositivos es capaz de ejecutar hasta 900 millones de operaciones de punto flotante (MFLOPS) a una velocidad de reloj de 150 MHz. La familia de DSP's C6711 posee la flexibilidad operacional de controles de alta velocidad así como la capacidad numérica de arreglos de procesadores. Este procesador cuenta con 32 registros de propósito general con tamaño de palabra de 32 bits y 8 unidades funcionales altamente independientes. Las ocho unidades proporcionan cuatro ALU's de punto (flotante/fijo), dos ALU's de punto fijo y dos multiplicadores de punto (fijo/flotante). El C6711 puede generar dos MACs por ciclo para un total de 300 MMACS.

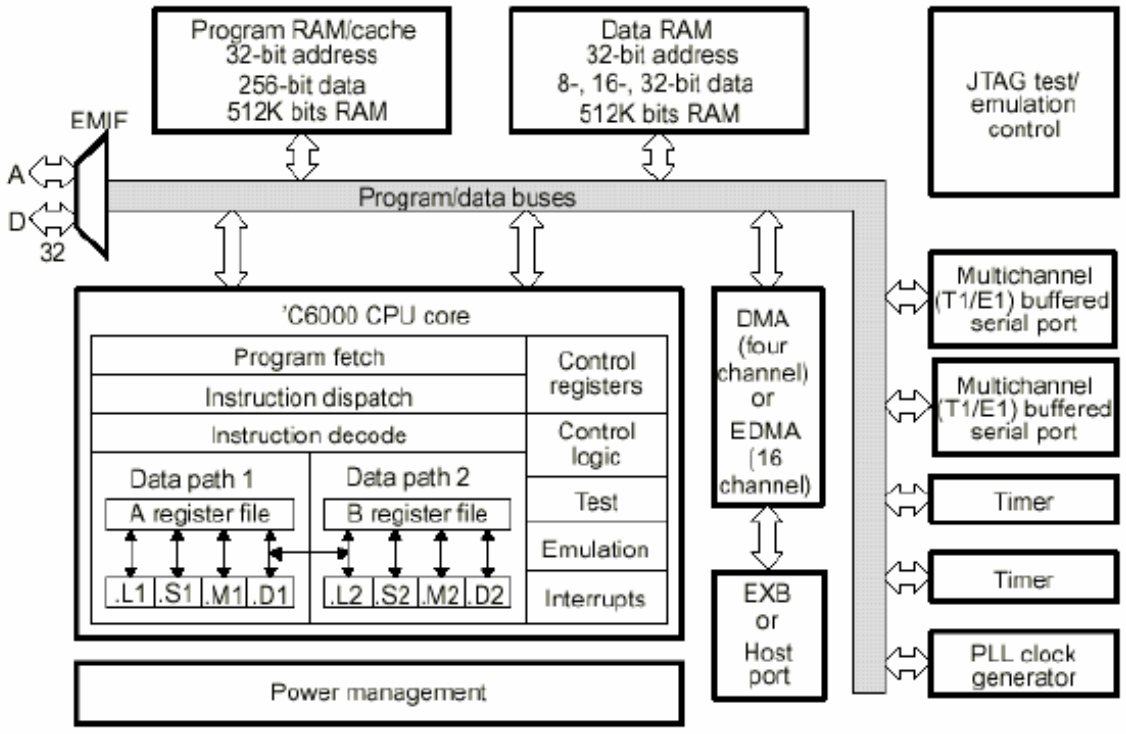


Figura 3.2, Diagrama de bloques del TMS320C62x/C67x

El DSP C6711 posee una arquitectura basada en un cache de dos niveles y tiene un poderoso y diverso conjunto de periféricos. El cache de programa nivel 1 (L1P) es un cache de 32 kbit de mapeo directo y el cache de datos nivel 1 (L1D) es un cache asociativo de 32 kbit de 2 vías.

La memoria/caché nivel 2 (L2) consiste en un espacio de 512 kbit de memoria que es compartida entre el espacio de programas y el de datos. L2 puede ser configurada como memoria mapeada, caché, o combinaciones de las dos. El conjunto de periféricos incluye dos puertos seriales con memoria intermedia multicanal (McBSPs), dos timers de propósito general, una interfaz de puerto-host (HPI), y una interfaz de memoria externa (EMIF), capaz de interconectarse con la SDRAM, SBSRAM y periféricos asíncronos.

El C6711 cuenta con un conjunto de herramientas de desarrollo entre las que se incluyen: un compilador C, un ensamblador simplificado para facilitar la programación y una interfaz de depuración para Windows de manera que se puede observar el código fuente en ejecución.

3.4 Descripción del CPU (núcleo del DSP)

El CPU maneja palabras de instrucción muy largas (VLIW) (de 256 bits de ancho) para suministrar hasta ocho instrucciones de 32 bits a las ocho unidades funcionales durante cada ciclo de reloj.

Como se observa en la figura 3.3, el CPU ofrece dos juegos de unidades funcionales. Cada juego contiene cuatro unidades y un archivo de registro; un juego contiene las unidades funcionales .L1, .S1, .M1 y .D1; y el otro juego contiene las unidades .D2, .M2, .S2 y .L2. Cada uno de los dos archivos de registro contiene 16 registros de 32 bits, para un total de 32 registros de propósito general. Los dos juegos de unidades funcionales, junto con los dos archivos de registro componen los lados A y B del CPU.

Las cuatro unidades funcionales de cada lado del CPU pueden compartir libremente los 16 registros que pertenecen a ese lado. Adicionalmente, cada lado ofrece un bus de datos sencillo conectado a todos los registros del otro lado, por medio del cual los dos juegos de unidades funcionales pueden acceder a datos contenidos en los registros del lado opuesto. El acceso de registro por medio de unidades funcionales del mismo lado del CPU, así como por el archivo de registro, puede dar servicio a todas las unidades en un ciclo sencillo de reloj, mientras que el acceso de registro usando el archivo de registro a través del CPU soporta una lectura y una escritura por ciclo.

El C67x CPU ejecuta todas las instrucciones del C62x. Además de las instrucciones de punto fijo del C62x, seis de las unidades funcionales (.L1, .S1, .M1, .M2, .S2, .L2) también ejecutan instrucciones de punto flotante. Las dos unidades restantes (.D1 y .D2) también ejecutan la instrucción LDDW que carga 64 bits por lado del CPU para un total de 128 bits por ciclo.

Otra característica clave de C67x CPU es la arquitectura de carga/almacenamiento, donde todas las instrucciones operan en registros (a diferencia de los datos en memoria). Dos juegos de unidades de direccionamiento de datos (.D1 y .D2) son responsables de todas las transferencias de datos entre el archivo de registro y el de memoria. La dirección de datos manejada por las unidades .D permite generar direcciones de datos de un archivo de registro para cargar o almacenar datos, hacia o desde otro archivo de registro. El C67x CPU soporta una gran variedad de modos de direccionamiento indirecto usando los modos de direccionamiento tanto lineal como circular con offset de 5 o 15 bits.

Todas las instrucciones son condicionales y la mayoría puede acceder a cualquiera de los 32 registros. Las dos unidades funcionales .M son dedicadas para la multiplicación. Las dos unidades funcionales .S y .L desempeñan un conjunto general de funciones aritméticas, lógicas y de salto; con resultados disponibles cada ciclo de reloj.

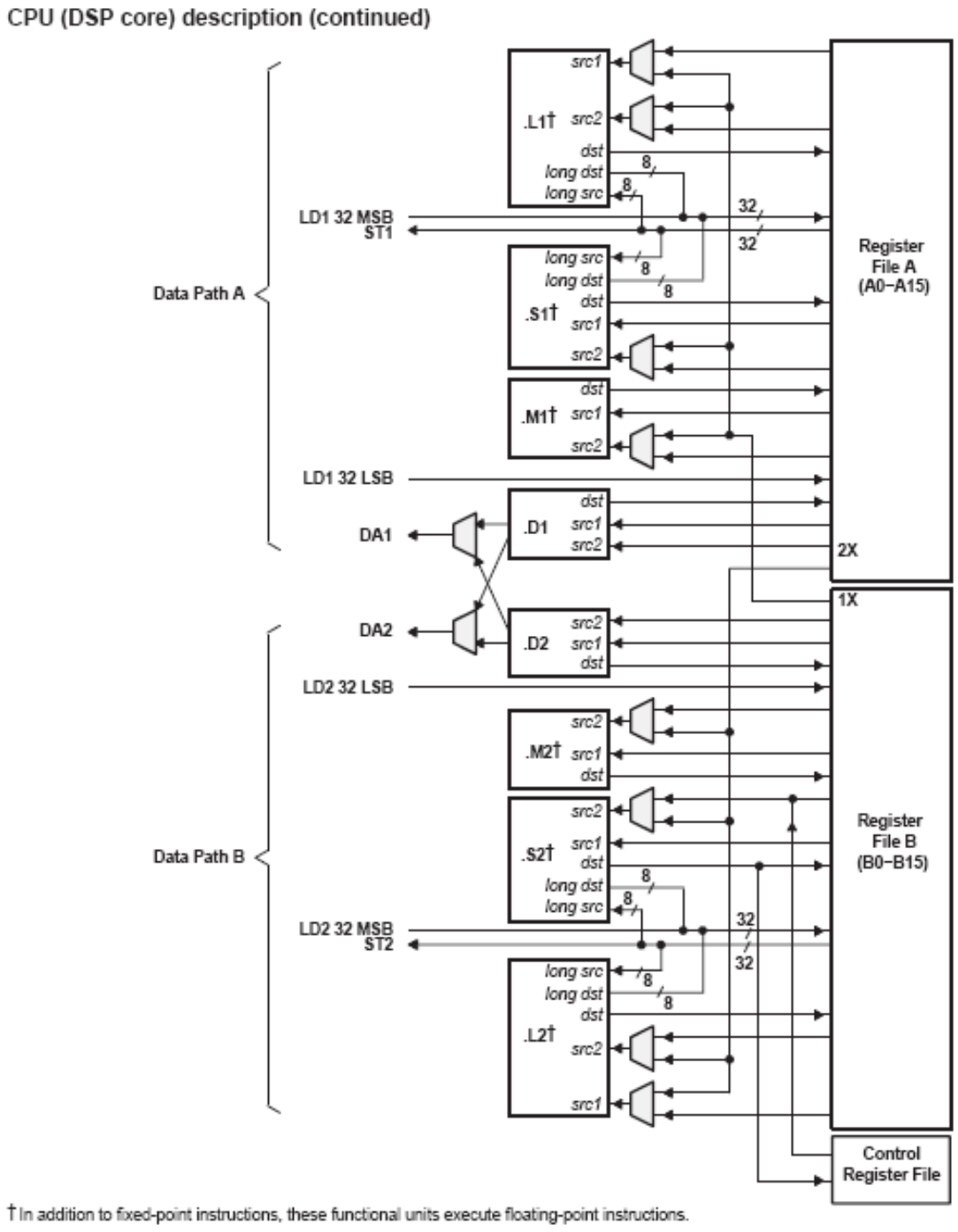


Figura 3.3, Trayectorias de los datos en el CPU TMS320C67x (núcleo del DSP)

3.5 interrupciones de DSP

Típicamente, los DSP's trabajan dentro de un entorno que contiene múltiples eventos externos asíncronos. Estos eventos requieren que el DSP ejecute tareas, cuando se presentan. Una interrupción es un evento que detiene el proceso actual en el CPU, de tal forma que pueda atender a la tarea requerida por el evento. Esta fuente de interrupción puede presentarse dentro del chip o externa a él, tal como temporizadores, convertidores analógico-digitales o periféricos.

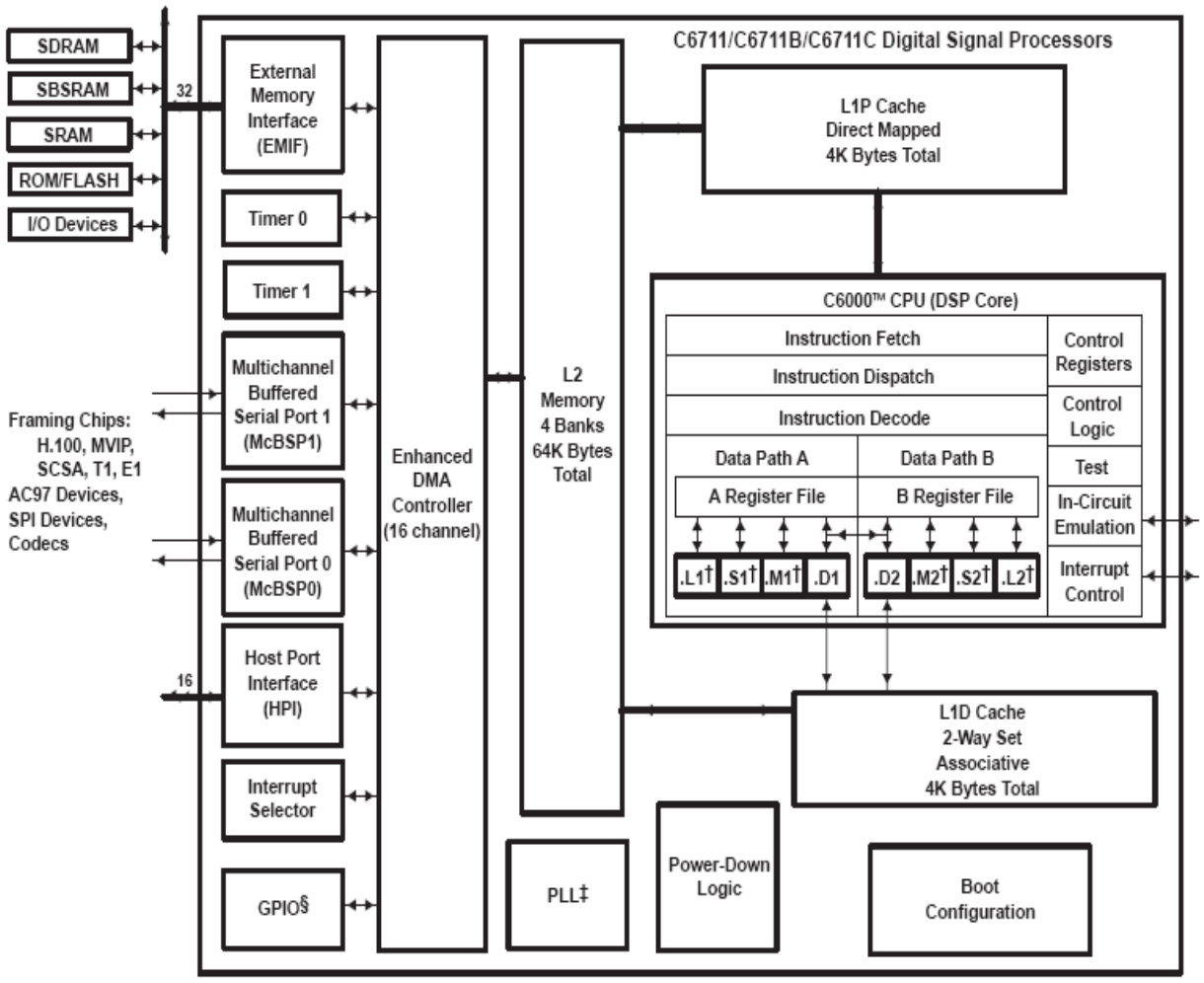
Tipos de interrupciones

Básicamente existen tres tipos de interrupciones en los CPUs de los DSP's de la familia C6000.

- Reset.
- Mascarables.
- No mascarables.

La diferencia entre los tres tipos radica en sus prioridades. La interrupción de reset tiene la más alta prioridad y corresponde a la señal RESET. La interrupción no mascarable tiene la siguiente prioridad y corresponde a la señal NMI. Las interrupciones de baja prioridad son las interrupciones 4-15 que corresponden a las señales INT4 – INT 15 que se encuentran mapeadas a los pines del C6000.

3.6 Periféricos del DSP



† In addition to fixed-point instructions, these functional units execute floating-point instructions.
‡ The C6711C device has a software-configurable PLL (with x4 through x25 multiplier and /1 through /32 divider) and a PLL Controller which is different from the hardware PLL peripheral on the C6711 and C6711B devices.
§ Applicable to the C6711C device only

Figura 3.4, Diagrama de bloques del DSP C67x.

En la figura 3.4 se observa un diagrama de bloques que representa los periféricos con los que cuentan los C67x. A continuación se describen algunos de ellos:

- Controlador de acceso directo a memoria mejorado (EDMA): El controlador EDMA se encarga de transferir datos, entre un rango de direcciones en el mapa de memoria, sin que actúe el CPU. Es el encargado de manejar todas

las transferencias de datos entre el controlador de memoria/caché de nivel 2 (L2) y los dispositivos periféricos. Cuenta con 16 canales independientes con prioridades programables y tiene la capacidad de ligar transferencias de datos. El EDMA permite movimientos de datos desde/hacia cualquier espacio de memoria direccionable, incluyendo la localidad de memoria interna (L2, SRAM), periféricos y memoria externa.

- Interfaz del puerto host (HPI): El HPI es un puerto paralelo de 16 bits, en el cual, un procesador host (huésped) puede acceder directamente a las localidades de memoria del CPU. El dispositivo host funciona como maestro para la interfaz, que facilita el incremento de acceso. El CPU y el host pueden intercambiar datos mediante la memoria interna o externa. Además, el host tiene acceso directo a los periféricos mapeados en memoria.

- Interfaz de memoria externa (EMIF): El EMIF maneja una interfaz "glueless" de 32 bits para varios dispositivos externos, incluyendo los siguientes:
    - SRAM síncrona de ráfaga (burst), (SBSRAM).
    - DRAM síncrona (SDRAM).
    - Dispositivos asíncronos, incluyendo SRAM, EPROM, ROM y FIFOs.
    - Un dispositivo externo de memoria compartida.

- Configuración de arranque: Los dispositivos de la familia C6000 ofrecen una variedad de configuraciones de arranque, las cuales determinan las acciones ejecutadas por el DSP acto seguido que el dispositivo se ha establecido y preparado para la inicialización. Este incluye la carga de código de una localidad de ROM externo en el EMIF y la carga de código a través del HPI de un host externo.

- McBSP: El puerto serial con memoria intermedia multicanal (McBSP) se basa en el estándar de interfaz de puerto serie que se encuentra en los dispositivos de las familias C2000 y C5000. Además, el puerto tiene la capacidad de almacenar muestras seriales en memoria de forma automática con la ayuda del controlador EDMA. También presenta la capacidad multicanal compatible con los estándares de redes T1, E1, SCSA y MVIP. Este puerto tiene las siguientes características:
    - Comunicación full-duplex.
    - Registros de datos de doble memoria intermedia, lo que permite un flujo de datos continuo.
    - Tramado independiente y sincronización por medio de reloj en la transmisión/ recepción.
    - Interfaz directa a codecs (codificador-decodificador) estándar, chips de interfaz analógica (AICs) y otros dispositivos análogo/digital (A/D) y digital/análogo (D/A) conectados en modo serial.
    - Interfaz directa a:
        - Tramas T1/E1
        - Dispositivos conforme a ST-BUS
        - Dispositivos conforme a IOM-2
        - Dispositivos conforme a AC97
        - Dispositivos conforme a IIS
        - Dispositivos SPI
    - Transmisión y recepción multicanal (hasta 128 canales).
    - Diferentes tamaños de datos (8,12,16,20,24 y 32 bits).
    - Compresión ley-μ y ley-A.

  - Transferencia de datos de 8 bits con LSB (bit menos significativo) y MSB (bit más significativo).
  - Polaridad programada de sincronización de tramas y reloj de datos.
  - Reloj interno y generador de tramas programables.

- Temporizador (Timer): Los dispositivos de la familia C6000, cuentan con temporizadores de propósito general de 32 bits que pueden ser empleados como:
  - Eventos de tiempo.
  - Eventos de conteo.
  - Generador de pulsos.
  - Interrupción del CPU.
  - Envío de eventos de sincronización para el controlador EDMA.

- Selector de interrupciones: El juego de periféricos de la familia C6000 cuenta hasta con 32 fuentes de interrupción. Sin embargo, el CPU dispone de 12 interrupciones. El selector de interrupciones permite al usuario seleccionar y priorizar, cuál de las 12 interrupciones de 32, se requieren en el sistema.

- Lógica para bajo consumo de energía: La lógica para bajo consumo de energía permite reducir la demanda de energía. La mayor parte de la potencia de operación de la lógica CMOS se disipa durante la conmutación de un estado lógico a otro.

3.7 Code Composer Studio

El Code Composer Studio (CCS) es un entorno de desarrollo integrado que permite generar código fuente para el TMS320x.

El CSS incluye los siguientes componentes:

- Herramientas de generación de código para el TMS320C6000.
- Entorno de Desarrollo Integrado del CSS (IDE).
- DSP/BIOS plug-ins y API.
- RTDX plug-in, interfaz host y API.

Estos componentes trabajan como se muestra en la figura 3.5.

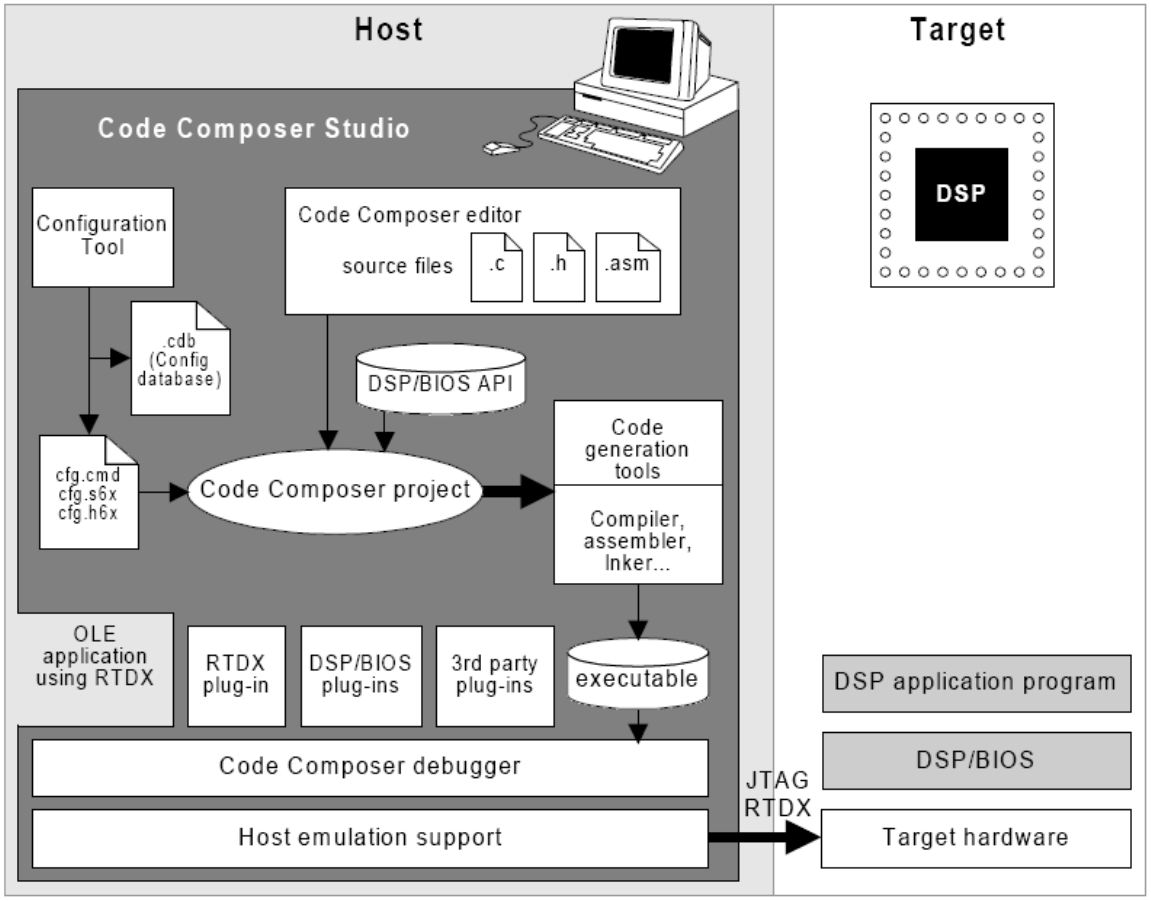


Figura 3.5, Componentes de trabajo del CCS

3.7.1 Herramientas de generación de código

Los TMS320C6000 soportan un conjunto de herramientas para desarrollo de software, que incluyen: un compilador C/C++ optimizado, un ensamblador optimizado, un ensamblador, un ligador. Así como las herramientas asociadas a ellos.

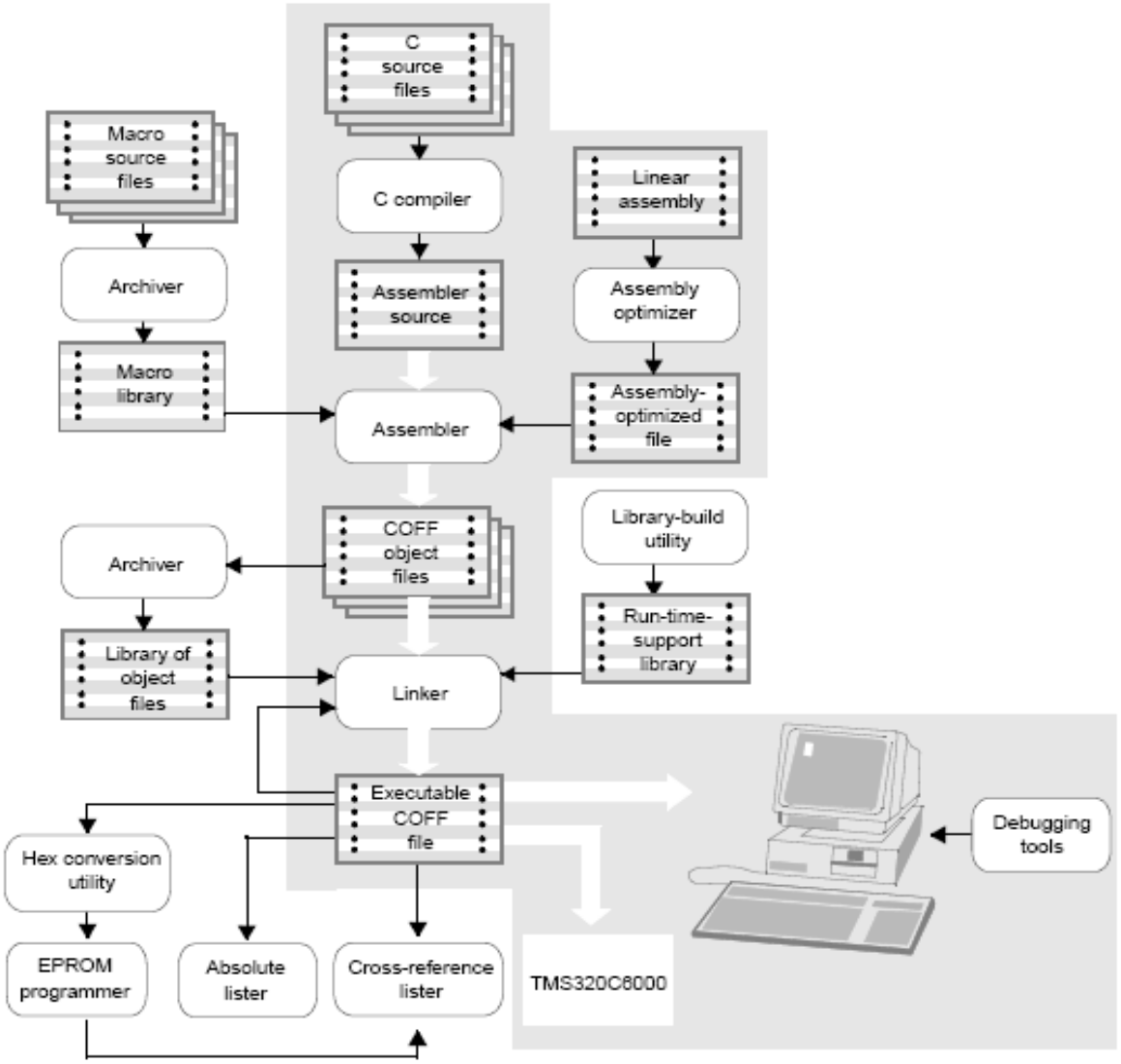


Figura 3.6, Flujo para el desarrollo de aplicaciones con los DSP C6000

La figura 3.6 muestra un flujo típico de desarrollo de software. El camino de desarrollo más común para programas en C esta sombreado. Las otras porciones son funciones periféricas que mejoran el proceso de desarrollo.

La siguiente lista describe las herramientas que se muestran en la figura 3.6:

- El C Compiler acepta código fuente C y produce código fuente en lenguaje ensamblador.
- El Assembler traduce los archivos fuente en lenguaje ensamblador en archivos objeto de lenguaje de máquina. El lenguaje de maquina está basado en el formato de archivo objeto común (COFF).
- El Assembly Optimizer permite escribir código en ensamblador lineal sin preocuparse por la estructura "pipeline" o por la asignación de registros. Asigna registros y utiliza optimización de bucle para convertir el código en ensamblador lineal a código ensamblador paralelo.
- El Linker combina archivos objeto en un solo archivo ejecutable para el DSP. También acepta archivos de librería y módulos de salida creados por una ejecución anterior del linker.
- El Archiver permite agrupar un grupo de archivos dentro de un solo archivo, llamado librería.
- Se puede utilizar la library-build utility para crear una librería de soporte propia en tiempo real personalizada.
- La Hex conversion utility convierte un archivo de objeto COFF en uno TI-Tagged, ASC-II hex, Intel, Motorola-s o Tektronix. Se puede bajar el archivo convertido a un programador de EPROM.
- El Cross-reference lister utiliza archivos objeto para generar símbolos de referencia cruzada, sus definiciones y sus referencias en los archivos fuente ligados.
- El Absolute lister acepta archivos objeto ligados como entrada y genera archivos .abs como salida. Se ensamblan los archivos .abs para producir un listado que contenga tanto las direcciones relativas como absolutas.

3.7.2 El IDE del CSS

El entorno de desarrollo integrado del CCS está diseñado para poder editar, generar y depurar programas del DSP.

Características del editor de código de programas

El CCS permite editar código fuente en lenguaje ensamblador y C. Del mismo modo se puede ver el código fuente C con la correspondiente instrucción en lenguaje ensamblador mostrada después de las sentencias de C. En la figura 3.7 se observa un ejemplo de una ventana de edición de codigo.

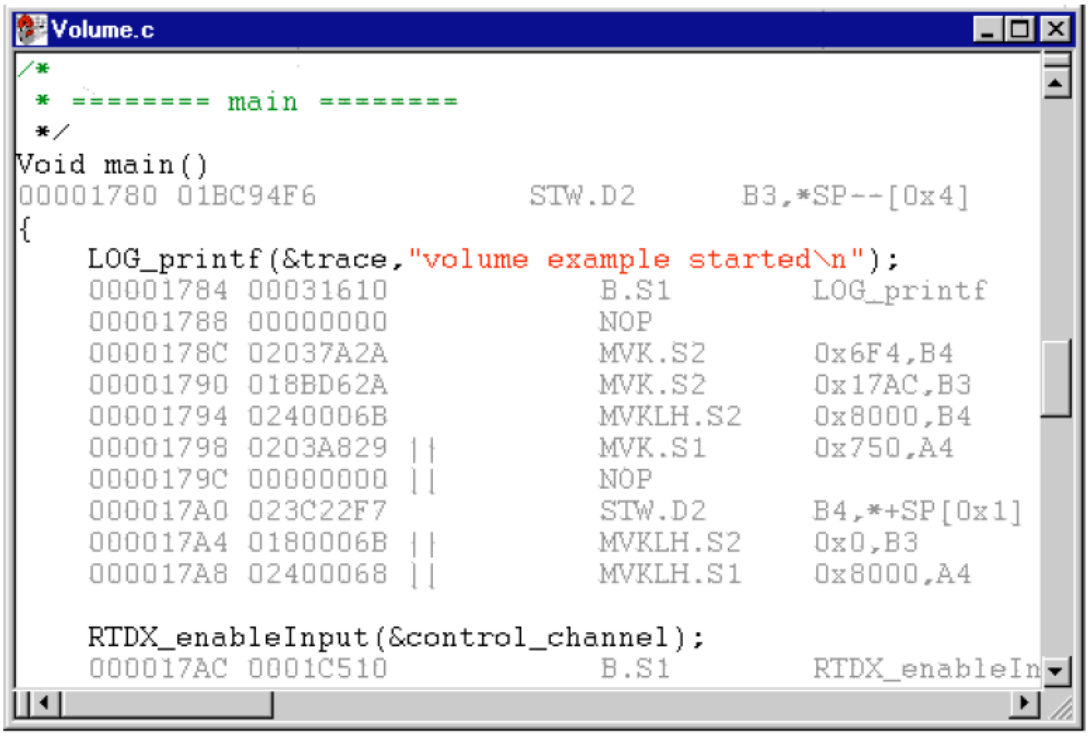


Figura 3.7, Ejemplo de una ventana de edición de código.

El editor integrado proporciona soporte para las siguientes actividades:

- Resalte de palabras clave, comentarios y cadenas en color.
- Marcar bloques de C en paréntesis y corchetes, encontrar el juego o el siguiente paréntesis o corchete.
- Incrementar o disminuir los niveles de sangría.

- Encontrar y reemplazar en uno o más archivos, encontrar el siguiente y el anterior, búsqueda rápida.
- Deshacer y rehacer múltiples acciones.
- Obtener ayuda sensible al contexto.
- Personalizar asignaciones de comandos al teclado.

Características de construcción de aplicaciones.

Dentro del Code Composer Studio, se puede crear una aplicación al agregar archivos a un proyecto. El archivo de proyecto es utilizado para construir la aplicación. Los archivos del proyecto pueden ser archivos de código fuente en C, archivos de código fuente en ensamblador, archivos objeto, librerías, archivos de comando del linker y archivos include. En la figura 3.8 podemos ver un ejemplo.

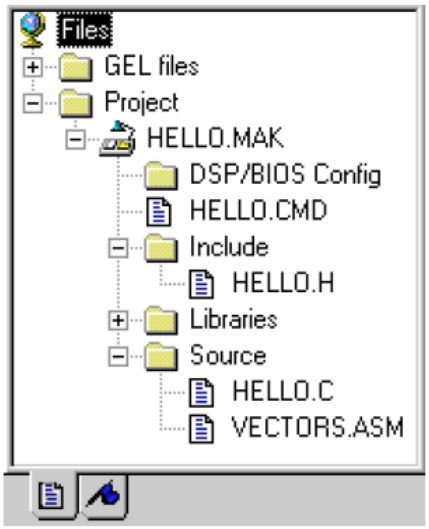


Figura 3.8, Ejemplo de un proyecto de trabajo.

Características de depuración de aplicaciones.

CCS provee soporte para las siguientes actividades de depuración:

- Colocar breakpoints.
- Actualizar automáticamente ventanas en los breakpoints.

- Observar variables.
- Ver y editar memoria y registros.
- Ver la pila de la llamada a funciones.
- Usar herramientas de puntos de prueba para flujo de datos, hacia y desde la tarjeta.
- Graficar señales de la tarjeta.
- Esbozar estadísticas de ejecución.
- Ver instrucciones desensambladas y en lenguaje C que se ejecutan en la tarjeta.

3.7.3 DSP/BIOS Plug-ins y API

Durante la fase de análisis de software del ciclo de desarrollo, las características de depuración tradicionales son ineficientes para diagnosticar problemas sutiles que surgen de las interacciones dependientes del tiempo.

Los Plug-ins del CCS proporcionados con el DSP/BIOS soportan ese análisis en tiempo real. Se pueden utilizar para sondar visualmente, trazar y monitorear una aplicación del DSP con el mínimo impacto para el desempeño en tiempo real.

Las DSP/BIOS APIs proporcionan las siguientes capacidades de análisis en tiempo real:

- Program Tracing. Despliega eventos escritos en registros seleccionados y refleja el flujo de control dinámico durante la ejecución de un programa.
- Performance Monitoring. Rastrea estadísticas que reflejan el uso de recursos seleccionados, tal como la carga del procesador y el tiempo de ejecución.
- File Streaming. Vincula objetos entrada/salida seleccionados/vecinos a archivos host.

Configuración del DSP/BIOS

Se pueden crear archivos de configuración en el entorno del CCS que definan objetos utilizados por el DSP/BIOS API. Este archivo también simplifica el mapeo de memoria y el vector de mapeo de hardware ISR, de tal forma que se puede utilizar a pesar de que no se utilice el DSP/BIOS API.

Cuando se abre un archivo de configuración del DSP/BIOS en el CCS este muestra un editor que permite crear y seleccionar propiedades para objetos de tiempo real que son utilizados en las llamadas de DSP/BIOS API's. Tal como se observa en la figura 3.9, entre estos objetos se incluye interrupciones por software, tuberías de I/O, eventos, registros, etc.

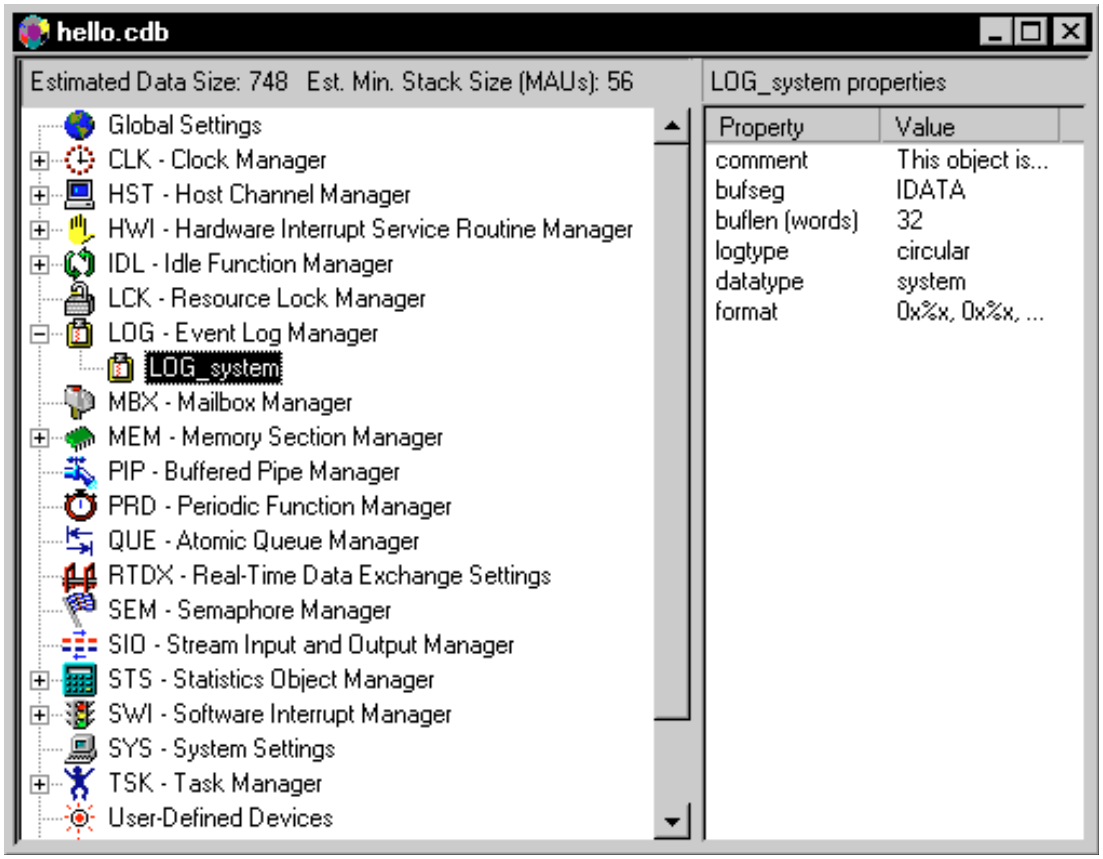


Figura 3.9, Ventana de configuración de los objetos del DSP/BIOS.

3.7.4 Emulación de Hardware e intercambio de datos en tiempo real (RTDX)

Los DSP's de TI proporcionan el soporte de emulación en el chip que habilita al CCS para controlar la ejecución de programas y monitorear actividades del programa en tiempo real.

El Hardware de emulación en el chip ofrece una variedad de capacidades, como:

- Inicio, detención o vuelta a cero del DSP.
- Carga de Código o datos en el DSP.
- Examinar los registros de memoria del DSP.
- Instrucciones de Hardware o breakpoints dependientes de datos.
- Intercambio de datos en tiempo real (RTDX) entre el host y el DSP.

3.8 El Microcontrolador

Un Microcontrolador es un circuito integrado (Chip) que contiene en su interior las tres unidades funcionales de una computadora: CPU, memoria y dispositivos de entrada/salida. A continuación se presentan varios aspectos del PIC 16F877A.

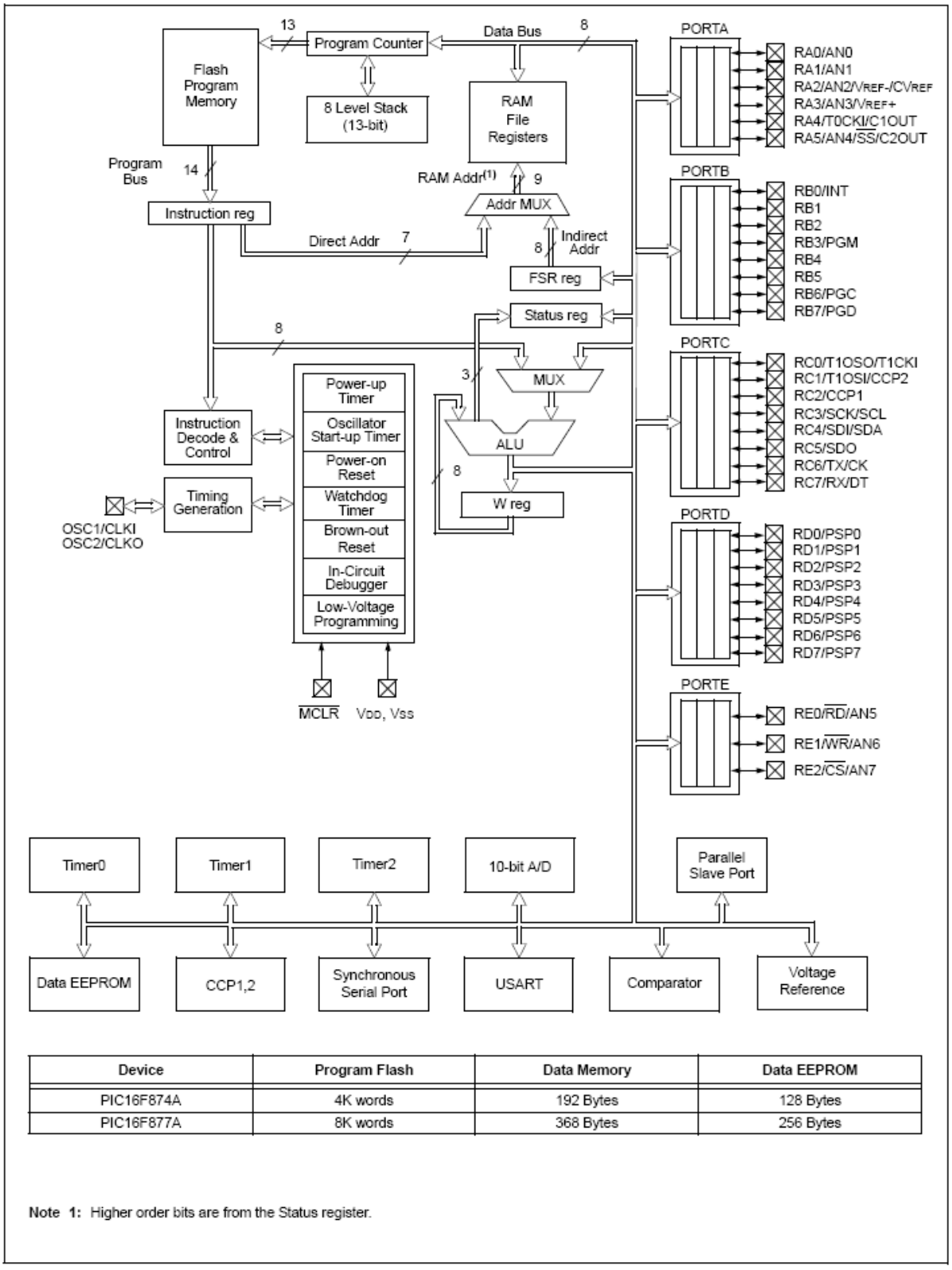
Los PIC's son una familia de microcontroladores tipo RISC fabricados por Microchip. El nombre actual no es un acrónimo. En realidad, el nombre completo es PICmicro, aunque generalmente se utiliza como Peripheral Interface Controller (Controlador de Interfaz Periférico).

3.8.1 Características principales del PIC 16F877A

En la tabla 3.1 se muestran las características principales del PIC 16F877A y en la figura 3.10 podemos ver su diagrama de bloques.

Tabla 3.1, Características Principales del PIC 16F877A.

Características Clave	**PIC 16F877A**
Frecuencia de operación	DC – 20 MHz
Reset y Retardos	POR, BOR (PWRT, OST)
Memoria de Programa Flash (Tamaño de palabra 14 bits)	8 k
Memoria de datos (bytes)	368
Memoria de datos EEPROM (bytes)	256
Interrupciones	15
Puertos de Entrada/Salida	Puertos A,B,C,D,E
Temporizadores (timers)	3
Módulos de Captura/Comparación y PWM	2
Comunicación Serial	MSSP, USART
Comunicación Paralelo	PSP
Modulo Análogo-Digital (10 bits)	8 Canales de entrada
Comparadores analógicos	2
Set de Instrucciones	35 Instrucciones



Device	Program Flash	Data Memory	Data EEPROM
PIC16F874A	4K words	192 Bytes	128 Bytes
PIC16F877A	8K words	368 Bytes	256 Bytes

Note 1: Higher order bits are from the Status register.

Figura 3.10, Diagrama de bloques del PIC 16F877A.

En el desarrollo de este prototipo solo se utilizaron algunas de las características de PIC, como:

- Memoria de programa
- Memoria de datos
- Set de instrucciones
- Puerto B como entrada/salida, de datos
- Puerto D como salida de datos
- Timer 0
- Modo de bajo consumo de energía (Sleep)
- Interrupción por evento externo
- Interrupción por desborde del timer 0

3.8.2 Mapa de Memoria

El PIC 16F877A cuenta con tres bloques de memoria. La memoria de programa y la memoria de datos tienen memoria intermedia (Buses) distintos, de tal forma que se puede acceder a ellos de manera simultánea. El bloque restante es el de la memoria EEPROM, mismo que no se utilizó en el proyecto.

Memoria de Programa

El PIC 16F877A tiene un contador de programa capaz de direccionar una palabra de 8k x 14 bits dentro de una localidad de memoria de programa. Acceder a una localidad por encima de las definidas causaría un sobre flujo, y por tanto, se generaría una acción no deseada. Como se observa en la figura 3.11, el vector de reset está en la localidad 0000h y el de interrupción en la 0004h.

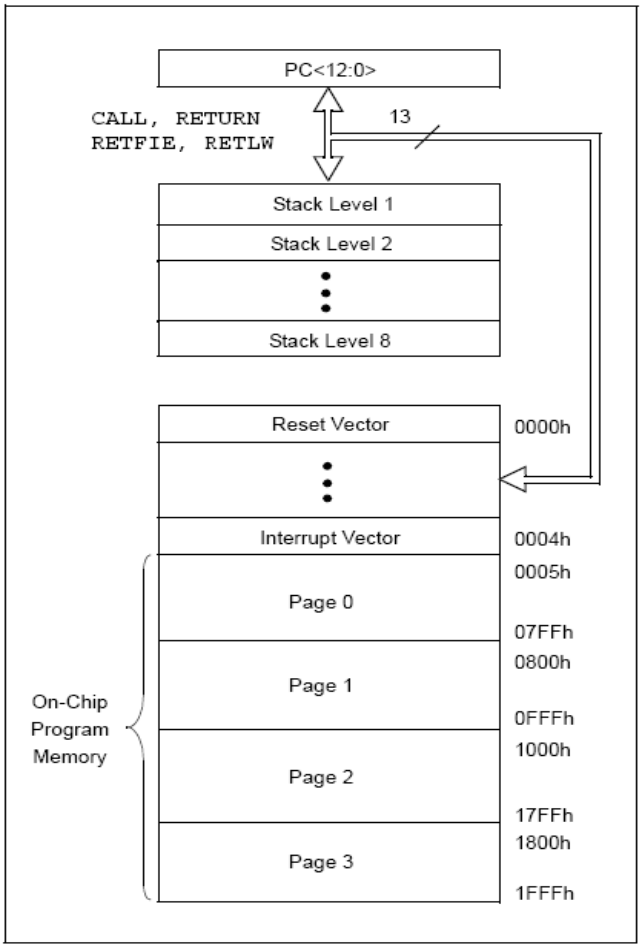


Figura 3.11, Mapa de memoria de programa y pila del PIC 16F877A.

Memoria de Datos

La memoria de datos está dividida en múltiples bancos que contienen los registros de propósito general y los registros de función especial. Los bits RP1 (Status<6>) y RP0 (Status <5>) son los que permiten seleccionar el banco; en la tabla 3.2 se observan los estados de los bits y los bancos asociados.

Tabla 3.2, Estados de los bits RP1 y RP0 y bancos asociados.

RP1:RP0	Bank
00	0
01	1
10	2
11	3

Cada banco se extiende hasta 7Fh (128 bytes). Las localidades más bajas de cada banco están reservadas para registros de función especial. Encima de los SFR, se encuentran los registros de propósito general implementados como memoria RAM estática.

Registros de propósito general

El registro puede ser accedido directa o indirectamente a través del registro de selección de archivo (FSR).

Registros de función especial

Los registros de función especial son registros utilizados por el CPU y los módulos periféricos para controlar la operación deseada del dispositivo. Estos registros están implementados como RAM estática.

En la figura 3.12 podemos ver el mapa del archivo de registro del PIC 16F877A.

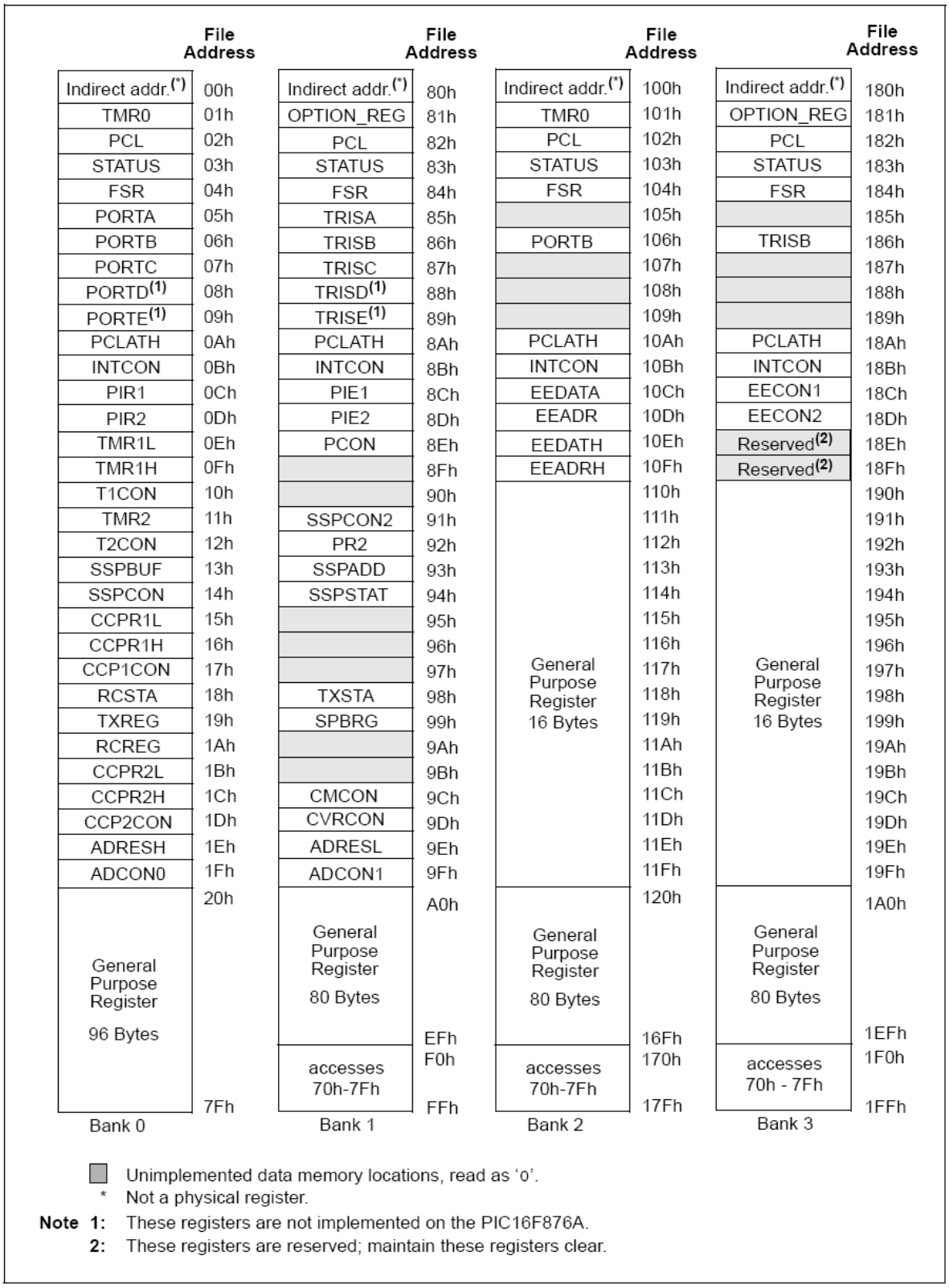


Figura 3.12, Mapa del archivo de registro del PIC 16F877A.

3.8.3 Puertos

Algunos pines para estos puertos de entrada/salida están multiplexados con un función alterna para las características periféricas del dispositivo. En general, cuando un periférico está habilitado, ese pin no debe ser utilizado como un pin de entrada/salida de propósito general.

Como ya se mencionó, para este prototipo solo se emplearon los puertos B y D, mismos que a continuación se describen.

Puerto B

El puerto B es un puerto Bidireccional de 8 bits. Como se observa en la tabla 3.3, la dirección de registro correspondiente es la TRISB. Fijar el bit TRISB (=1) hará el pin correspondiente del puerto B una entrada. Borrar el bit TRISB (=0) hará el pin correspondiente del puerto B una salida.

Tres pines del puerto B están multiplexados con las funciones "In-Circuit Debugger" y "Low-Voltaje Programming".

Cuatro de los pines del puerto B, pueden ser programados para tener una interrupción de cambio de estado. Esto solo puede ocurrir en aquellos que estén configurados como entrada.

RB0/INT es un pin de entrada de interrupción externa y es configurado usando el bit INTEDG.

Tabla 3.3, Registros asociados con el puerto B.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

Puerto D

El puerto D es un puerto de 8 bits con memoria intermedia de entrada de tipo Schmitt Trigger. Cada pin se puede configurar de manera individual como entrada/salida, ver tabla 3.4.

Tabla 3.4, Registros asociados con el puerto D.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

3.8.4 Temporizadores (timers)

El PIC 16F877A cuenta con un juego de temporizadores, que permiten manejar de manera eficiente todas las operaciones que involucran al tiempo y al conteo. Son tres y se denominan TMR0, TMR1 y TMR2. En este caso se utilizaron solamente los dos primeros. (TMR0 Y TMR1).

El timer0 (TMR0)

Las principales características de este temporizador son:

- Temporizador/contador de 8 bits.
- Legible y escribible.
- Preescalador programable de software de 8 bits.
- Selección de reloj interno o externo.
- Interrupción en sobre flujo de FFh a 00h.

En el proyecto, el TMR0 se utiliza para contar un segundo en el momento en que se genera una interrupción por evento externo, misma que despierta al microprocesador; al transcurrir ese segundo es posible determinar cuál fue la señal que se recibió. En la tabla 3.5 podemos ver los registros asociados con el timer0.

Tabla 3.5, Registros asociados con el timer0.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
01h,101h	TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
81h,181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

El timer1 (TMR1)

Las principales características de este temporizador son:

- Temporizador/contador de 16 bits.
- Legible y escribible.
- Preescalador programable de software de 8 bits.
- Selección de reloj interno o externo.
- Interrupción en sobre flujo de FFh a 00h.
- Reinicialización por medio del CCP1 y CCP2.

En la tabla 3.6 podemos ver los registros asociados con el timer1.

Tabla 3.6, Registros asociados con el timer1 como temporizador/contador.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.
Note 1: Bits PSPIE and PSPIF are reserved on the 28-pin devices; always maintain these bits clear.

En este proyecto, el TMR1 se utiliza para contar los pulsos recibidos.

3.8.5 Interrupciones

El PIC 16F877A cuenta con hasta 15 fuentes de interrupción. El registro de control de interrupciones (INTCON) almacena las peticiones de interrupción individuales en bits bandera. También tiene bits para habilitar interrupciones individuales y globales. Un bit de habilitación de interrupción global, GIE (INTCON<7>), habilita en el caso de ser fijado (=1) y deshabilita en el caso de ser borrado (=0).

Las banderas de interrupción son fijadas en 1 sin importar el estado en que se encuentre el bit correspondiente a la habilitación de cada una de las interrupciones o el estado del bit que habilita de manera global las interrupciones (GIE).

La instrucción "regreso de la interrupción" (RETFIE), abandona la rutina de interrupción y fija nuevamente a 1 el bit GIE, habilitando las interrupciones. Mientras que las interrupciones por evento externo, por cambio en el puerto B y por sobre flujo del timer0, están contenidas dentro del registro INTCON.

En el momento que se genera una interrupción, el bit GIE se deshabilita automáticamente para evitar que se efectúe cualquier otra interrupción; mientras que la dirección a la que debe regresar el programa cuando finaliza la ejecución de la rutina de interrupción es colocada en la pila (STACK) y el PC es fijado con 0004h.

Dentro de la rutina de interrupción se determina la causa de interrupción verificando el estado de las banderas de interrupción, mismas que deben ser puestas a 0 por medio del programa antes de habilitar nuevamente las interrupciones, evitando así las interrupciones cíclicas.

De las 15 interrupciones, se utilizan solamente dos:

- La interrupción por evento externo

Esta interrupción se habilita con el flanco de subida o bajada de una señal digital conectada al pin RB0/INT del microcontrolador. Si el bit INTEDG (OPTION_REG<6>) está fijado a 1, se activa por flanco de subida y si se encuentra fijado a 0 con flanco de bajada.

La bandera INTF debe ser fijada a 0 durante la rutina de interrupción, antes de habilitar nuevamente esta interrupción. Esta interrupción puede despertar al microprocesador del estado "Sleep", si el bit INTE fue fijado a 1 antes de que el microprocesador fuera mandado a dicho estado.

- La interrupción por sobre flujo del timer0

  Esta interrupción puede ser habilitada/deshabilitada fijando en 1/0 el bit de habilitación TMR0IE (INTCON<5>).

3.8.6 Modo de bajo consumo (Sleep)

Se accede a este modo al ejecutar la instrucción SLEEP. Los puertos de entrada y salida se mantienen en el mismo estado que se encontraban antes del acceso a dicho modo.

Para despertar al microprocesador debe ocurrir alguno de los siguientes eventos:

- Entrada de reset externo en el pin $\overline{MCLR}$.
- "Watchdog timer wake-up" (si está habilitado).
- Interrupción desde el pin INT, cambio en el puerto B o interrupción por algún periférico.

Para despertar al microprocesador mediante una interrupción es necesario que la misma esté habilitada. El despertar (wake-up) se presenta no importando el estado en que se encuentre el bit GIE. Si el bit GIE está fijado a 0, el dispositivo continúa la

ejecución a continuación de la instrucción SLEEP. Si este bit está fijado a 1 se ejecuta la instrucción después del SLEEP y avanza a la dirección del la localidad de interrupción (0004h). Cuando se presente el caso de que no sea requerida, se debe colocar un NOP después del SLEEP.

3.8.7 Configuraciones del Oscilador

El PIC 16F877A puede ser operado en cuatro diferentes modos de oscilación:

- LP: Low-Power Crystal.
- XT: Crystal/Resonator.
- HS: High-Speed Crystal/Resonator.
- RC: Resistor/Capacitor.

Esto se logra al programar dos bits de configuración (Fosc1 y Fosc0). En el caso de este proyecto, se usa el modo XT utilizado un cristal de cuarzo de 4 MHz, configuración que se ilustra en la figura 3.13.

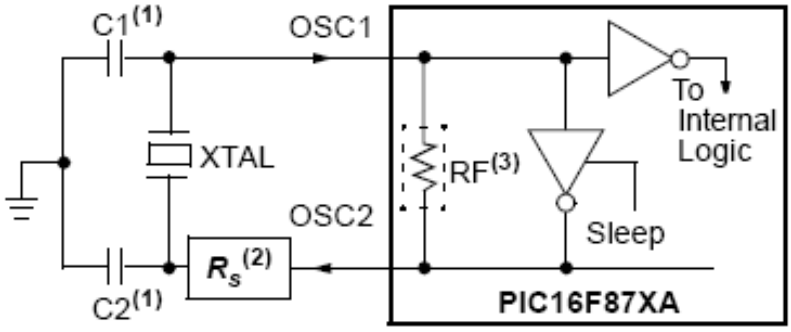


Figura 3.13, El PIC operado en el modo XT, utilizando un cristal de Cuarzo.

3.9 MPLAB

El MPLAB es un editor de entorno de desarrollo integrado (IDE), destinado a dispositivos de Microchip. Este editor es modular, permite seleccionar de entre diversos microprocesadores, además de permitir la grabación de estos circuitos integrados directamente al programador.

3.9.1 El IDE del MPLAB

Es un programa que corre bajo el sistema operativo Windows y como tal, presenta las clásicas barras de programa, de menú, de herramientas de estado, etc. El entorno de desarrollo del MPLAB posee un editor de texto, compilador y simulación (no en tiempo real). Para comenzar un programa desde cero y después grabarlo al microcontrolador se deben seguir los siguientes pasos:

- Crear un nuevo archivo con extensión .ASM.
- Crear un nuevo proyecto.
- Agregar el archivo .ASM como un archivo fuente (SOURCE FILE).
- Elegir el microcontrolador a utilizar desde (SELECT DEVICE) del menú CONFIGURE.

Una vez realizado esto, se está en condiciones de empezar a escribir el programa, respetando las directivas necesarias y la sintaxis, para luego compilarlo y grabarlo en el PIC.

3.9.2 Compilación del programa y carga al PIC.

Una vez escrito y depurado el programa, se procede a la compilación. Para esto, desde el menú PROYECT se elige la opción BUILD ALL, que si no existen errores, devolverá un mensaje BUILD SUCCESFULL. Los errores que muestra el compilador son de sintaxis, es decir que si se espera que se ponga un bit en 0 y esto

nunca pasa, se estaría entonces dentro de un ciclo infinito y el compilador trabajaría perfectamente porque no hay error de sintaxis.

También existen mensajes y advertencias (warnings); los mensajes pueden ser, por ejemplo, que se está trabajando en un banco de memoria que no es el banco 0, etc. Las advertencias tienen un poco más de peso, por ejemplo: el PIC seleccionado no es el mismo que está definido en el programa, etc. En ambos casos, mensajes y advertencias, la compilación termina satisfactoriamente pero hay que tenerlos en cuenta para prevenir errores.

Terminada la compilación el MPLAB nos genera un archivo de extensión .hex, el cual, es completamente legible para el PIC, es decir, solo hace falta grabarlo al PIC por medio de una interfaz, por ejemplo el programador Picstart Plus de microchip. Una vez terminado esto, se alimenta el PIC y el programa se estará ejecutando.

IV. Implementación del sistema de control de iluminación

4.1 Introducción.

En este capítulo se describe el proceso que se siguió para implementar el sistema de control de iluminación. El propósito del sistema es reconocer los comandos de encendido y apagado, para después accionar el interruptor correspondiente a cada luz.

Podemos dividir el sistema en tres etapas:

- Sistema de reconocimiento de comandos.
- Interfaz entre el sistema de reconocimiento de comandos y el circuito eléctrico.
- Circuito eléctrico de iluminación.

A continuación se detallarán las etapas, poniendo atención a las consideraciones tomadas, a los métodos empleados, así como a las herramientas utilizadas.

4.2 sistema de reconocimiento de comandos.

El sistema de reconocimiento de comandos utiliza las técnicas descritas en el capítulo 2. Recordemos que al tratarse de un sistema dependiente del locutor es necesario entrenar previamente el sistema de reconocimiento, en la figura 4.1 se muestran de manera general los pasos a seguir para reconocer los comandos.

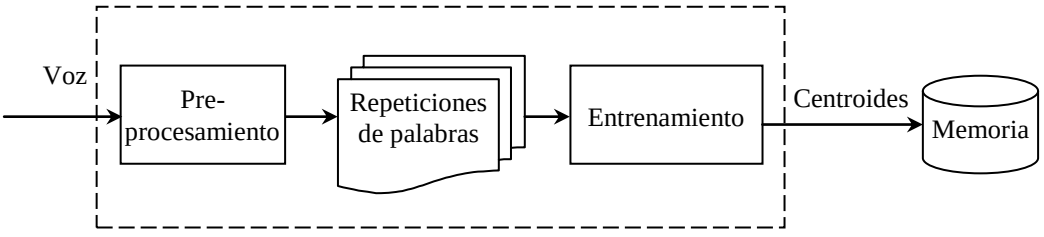


Figura 4.1 a), Etapa de entrenamiento del sistema de reconocimiento.

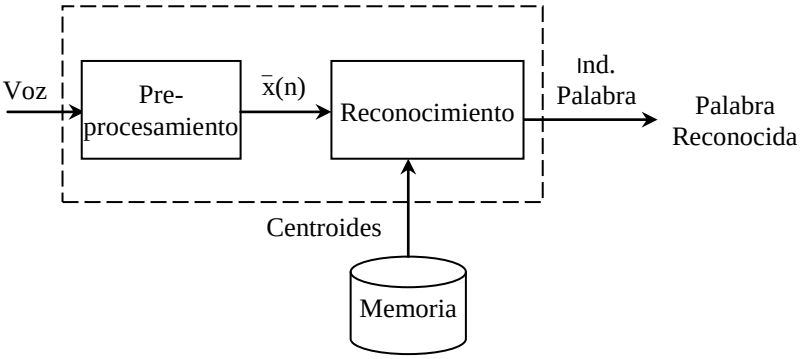


Figura 4.1 b), Etapa de reconocimiento.

4.2.1 Preprocesamiento de las señales de voz

Para poder realizar el reconocimiento de los comandos se utiliza un sistema basado en MSBC.

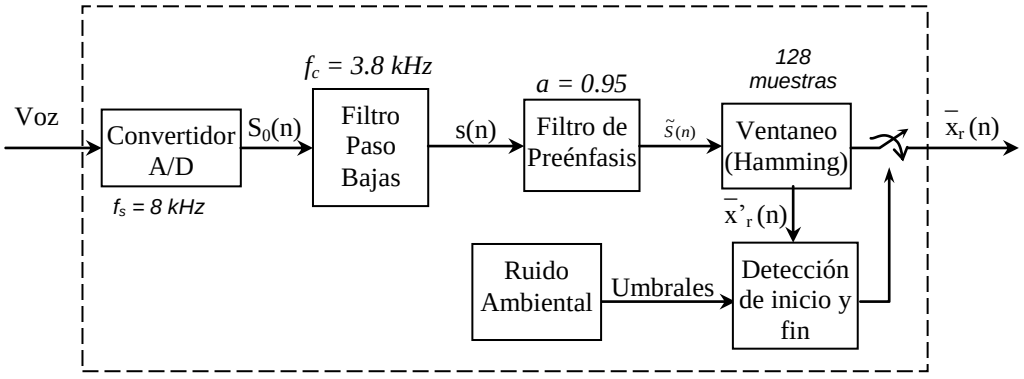


Figura 4.2, Diagrama de bloques para el módulo de preprocesamiento.

De acuerdo con el método propuesto en el capítulo 2, es conveniente adecuar las señales de voz a trabajar, de manera que faciliten su manejo y arrojen el resultado deseado. Es por eso que previo al entrenamiento se realizó una etapa de preprocesamiento.

Con ayuda de la tarjeta de desarrollo con que contamos DSK 6711 se grabaron 30 repeticiones de cada uno de los comandos que requeríamos.

En la figura 4.2 se muestran las etapas que siguieron las repeticiones de los comandos en el preprocesamiento, mismos que se describen a continuación:

- La señal de voz (señal analógica) es convertida en señal digital, con una frecuencia de muestreo de 8 kHz.
- Una vez que se tienen las muestras digitales, se pasan pon un filtro paso bajas, con una frecuencia de corte de 3.8 kHz. Hecho que nos permite concentrarnos en la porción de la señal donde está contenida en mayor porcentaje la voz (alrededor de los 4 kHz), eliminando gran parte del ruido que acompaña a la señal.
- Después de aplicar el filtro paso bajas, se envía la señal a un filtro preénfasis, con a=0.95, lo que nos permite elevar el nivel de las frecuencias altas de la señal, de tal manera que se modifiquen lo menos posible las características en frecuencia una vez que la señal se hizo pasar por un filtro paso bajas.

- A continuación se divide la señal en bloques de 128 muestras y se hace pasar por una ventana, para evitar discontinuidades al principio y al final de los bloques analizados. En este caso se utilizó una ventana de Hamming. En la figura 4.3 se muestra la forma de la ventana de Hamming y su respuesta en frecuencia.

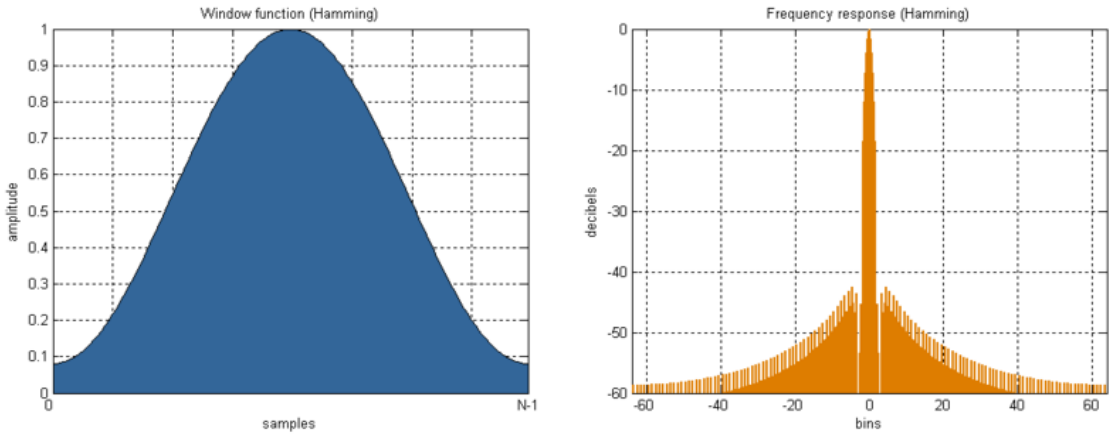


Figura 4.3, Forma de la ventana de Hamming y Respuesta en Frecuencia.

- Una vez que se tiene la señal muestreada, filtrada y ventaneada, se procede a detectar el inicio y el fin de los comandos. Esto se logra con ayuda del método Rabiner – Sambur, descrito previamente en el capítulo 2. Esto nos ayuda a reducir las muestras de la señal únicamente a aquellas en donde está contenida la señal de voz, eliminado aquellas en donde hay silencio.

Todos los algoritmos de los procesos descritos anteriormente fueron implementados en la tarjeta DSK C6711, misma que con ayuda de la computadora y el software (IDE, entorno de desarrollo) Code Composer Studio (CCS) nos permitió grabar las señales ya pre procesadas en la computadora; hecho que resultó indispensable para el siguiente paso del sistema de reconocimiento.

4.2.2 Entrenamiento del sistema de reconocimiento de comandos

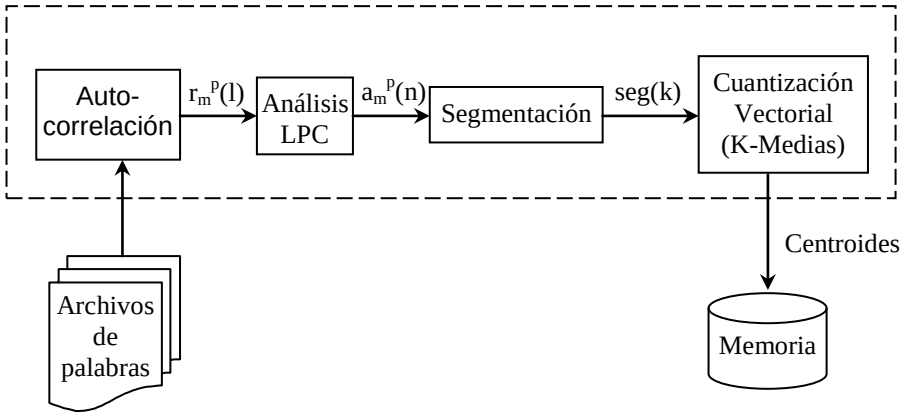


Figura 4.4, procesos requeridos para entrenar el sistema.

En la figura 4.4 se muestran los procesos requeridos para entrenar el sistema. Para facilidad en el análisis de las señales previo a la cuantización vectorial (obtención de los vectores característicos) se realizó este análisis en una computadora, implementando los algoritmos previamente descritos en MATLAB.

Estos son los pasos que se siguieron:

- A partir las grabaciones que se tenían se seleccionaron 24, al azar, ya que así se obtendría una muestra más representativa, que por lo tanto, nos permitiría caracterizar mejor a los vectores que representan los comandos.

- Primeramente, tomando tramas de cada una de las repeticiones de los comandos, respetando el número de muestras con que trabajamos anteriormente, 128 muestras. Aplicamos las siguientes técnicas, previamente descritas en el capítulo 2:
    - Autocorrelación de la señal.
    - Análisis LPC, con p= 16, que nos permitió obtener los coeficientes LPC.
    - Se segmentó la señal en cuatro partes (linealmente).
    - Para cada segmento se obtuvieron los centroides (vectores característicos), empleando los algoritmos descritos en el capítulo 2 para la distancia Itakura, como medida de distorsión y el método de K-medias, como método de cuantización vectorial.

- Una vez que ya se tenían los centroides de cada uno de los comandos, se generó un archivo de texto que fuera capaz de ser interpretado por el CCS, cuyo contenido son los centroides de todos los comandos de voz.

Es así como se logró tener almacenado en memoria los centroides (vectores característicos), de referencia para el reconocimiento de los comandos en tiempo real.

4.2.3 Reconocimiento de los comandos

En la figura 4.5 se muestran los pasos que se siguen para reconocer los comandos de voz. Como podemos notar en la figura los primeros tres pasos resultan idénticos a los realizados en el entrenamiento. Los algoritmos requeridos para estos pasos fueron implementados en el DSK C6711, lo que nos permite realizar el reconocimiento de los comandos con una gran rapidez, obteniéndose el comando reconocido prácticamente en tiempo real.

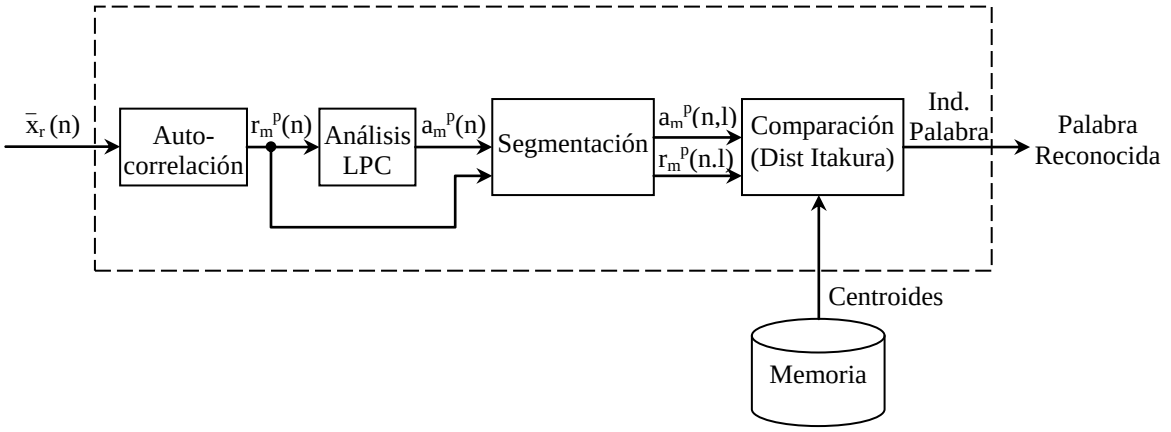


Figura 4.5, procesos requeridos para reconocer los comandos.

Los pasos a seguir son:

- Tomando una señal que fue previamente preprocesada, es decir, filtrada y recortada, se realizan las siguientes técnicas.
    - Por cada 128 muestras se realiza la autocorrelación de la señal. Se segmentan los resultados para cálculos posteriores.
    - A partir de los resultados obtenidos en la autocorrelación, se realiza el análisis LPC, obteniendo así los coeficientes LPC.
    - Una vez que se tienen los LPC se segmentan los resultados en 4 partes, tal y como se realizó durante el entrenamiento, lo que nos permite comparar los vectores con su segmento respectivo.

- Se comparan los coeficientes lpc con los centroides (vectores característicos), de todos los comandos y con ayuda de la distancia de Itakura (como medida de distorsión) se determina cuales son los vectores con los que se presenta la menor distorsión, indicándonos esto cual es el comando que se ha reconocido, ver figura 4.6.

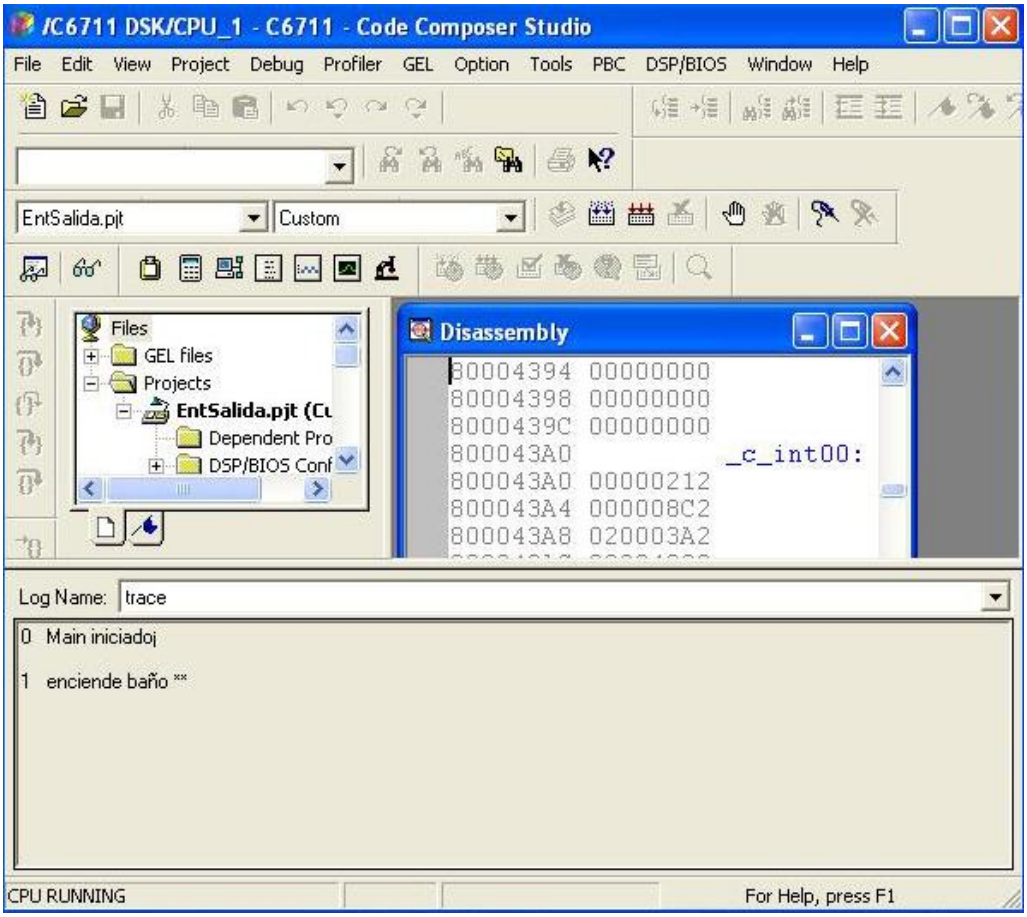


Figura 4.6, Mensaje desplegado para un comando reconocido.

Cabe destacar que se establecieron umbrales que nos permiten descartar comandos que no están caracterizados, a lo que el sistema responde con un mensaje de "palabra desconocida" (ver figura 4.7), por otro lado, si el sistema tiene duda en relación a dos comandos que tengan una distorsión muy similar el sistema pedirá repetir el comando con un mensaje de "repetir palabra" (ver figura 4.8), de tal

manera que se realice nuevamente el análisis, pero ahora únicamente considerando los vectores característicos de los comandos con distorsión muy similar en el análisis anterior.

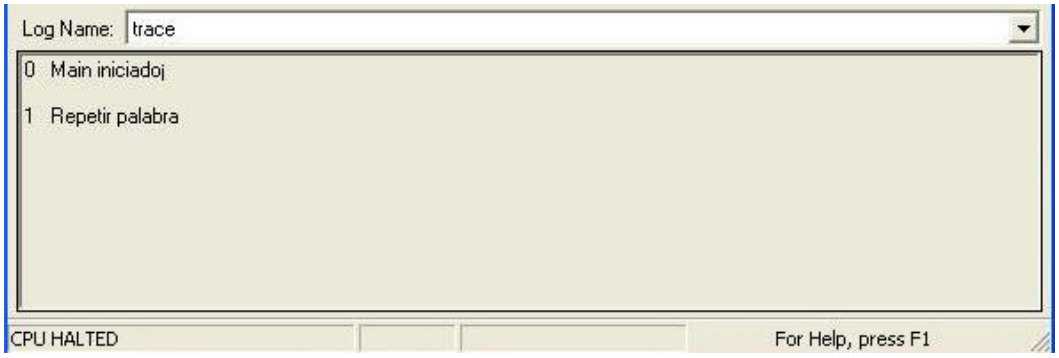


Figura 4.7, Mensaje desplegado para repetir palabra.

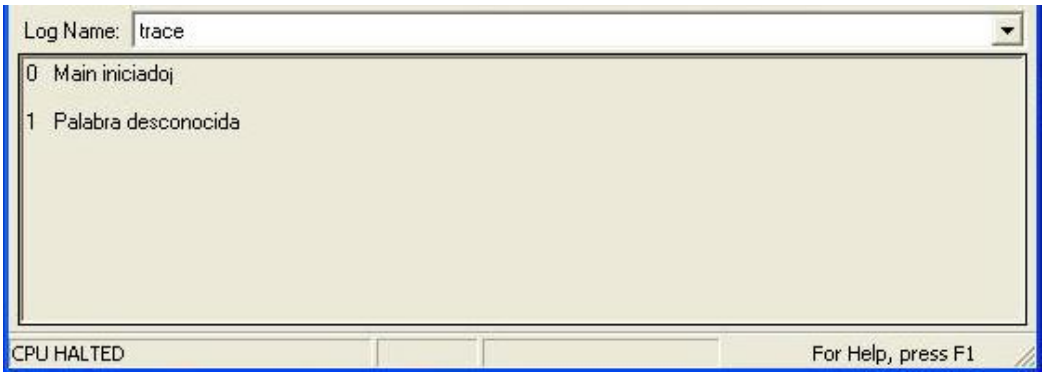


Figura 4.8, Mensaje desplegado para un comando desconocido.

Además, hay que notar que cuando se realiza la detección de inicio-fin del comando de voz durante la etapa de preprocesamiento se consideran las primeras 10 muestras de la señal sin recortar, para caracterizar al ruido, hecho que nos permite operar el sistema en condiciones diversas de ruido ambiente.

Recordando que los algoritmos están implementados en el IDE (Code Composer Studio) del DSK C6711, nos auxiliamos del mismo para desplegar los mensajes previamente mencionados y se implementó un algoritmo para enviar el comando reconocido al circuito que enciende y apaga la luz.

4.3 Interfaz con el sistema eléctrico de iluminación

Una vez que se consiguió reconocer los comandos exitosamente, se continuó con la siguiente etapa, el interconectar el sistema de reconocimiento de voz con el circuito eléctrico.

Para tal efecto se hicieron algunas adecuaciones al programa que se carga en el DSK, estas adecuaciones permitieron que el DSK, además de que desplegara un mensaje con el comando reconocido, generara una señal que pudiese ser enviada a otro circuito.

Inicialmente se consideró asignar un pulso constante cuya amplitud en voltaje variaba de acuerdo con el comando que se reconocía, sin embargo esto no resultó muy preciso ya que la tarjeta presenta variaciones en el voltaje de referencia y no permite que los valores generados acorde a la instrucción deseada sean siempre los mismos; de tal manera que ocasiona errores en la interpretación de los comandos.

Fue entonces que se implementó otra estrategia, esta consiste en generar una señal sinusoidal de amplitud constante en donde lo que se varía ahora es la frecuencia de la señal de acuerdo con el comando reconocido, haciendo el método inmune a las posibles variaciones en el voltaje de referencia que antes afectaba. En la tabla 4.1 se muestran las frecuencias de la señal sinusoidal de acuerdo al comando reconocido.

Para poder identificar la señal y posteriormente generar una instrucción que finalmente accione el interruptor en el circuito eléctrico se utiliza un microcontrolador PIC.

La señal debe ser reacondicionada, ya que por un lado es necesario eliminar el voltaje de “offset” y por otro lado se requiere convertirla en una señal cuadrada, en donde se mantenga la frecuencia de la señal sinusoidal. Esto nos permite manejar de mejor manera la señal en el microcontrolador.

Tabla 4.1, Frecuencias de las señales sinusoidales asociadas a los comandos reconocidos.

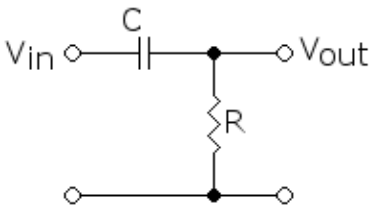
Comando	Frecuencia de la señal sinusoidal [Hz]
Apaga baño	300
Apaga cocina	500
Apaga comedor	700
Apaga jardín	900
Apaga recámara	1100
Apaga sala	1300
Enciende baño	1500
Enciende cocina	1700
Enciende comedor	1900
Enciende jardín	2100
Enciende recámara	2300
Enciende sala	2500
Señal desconocida	2700

4.3.1 Adecuación de la señal

Podemos dividirla en dos etapas:

- Eliminar el offset
- Convertir a la señal sinusoidal en pulsos cuadrados.

Al analizar la señal que originaba el offset, notamos que se trataba de una señal de baja frecuencia, por lo que para eliminarla se tomó la decisión de implementar un filtro. El filtro que cumplía con las características requeridas es un filtro paso altas, que es aquel que permite pasar únicamente a señales cuya frecuencia es mayor a la frecuencia de corte del filtro, f_0. El filtro implementado es un filtro de primer orden, mismo que se muestra en la figura 4.9 y cuya respuesta en frecuencia se observa en la figura 4.10.



Con: C= 220n

y R=10k

Figura 4.9, Filtro implementado para eliminar el offset de la señal.

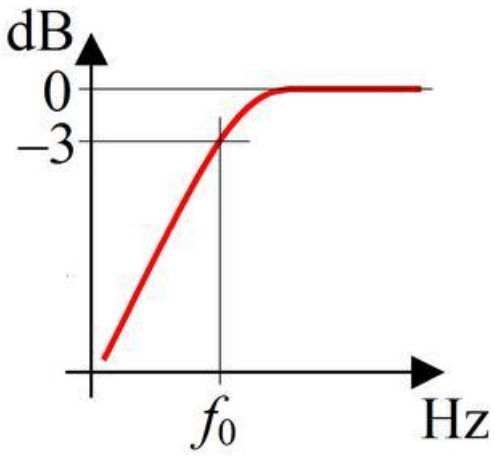


Figura 4.10, Respuesta en frecuencia del filtro para eliminar el offset.

Una vez que eliminamos el offset, una forma eficiente de convertir una señal sinusoidal en cuadrada es por medio de un comparador de voltaje. Un comparador es un circuito que observa una señal de voltaje en una de sus entradas en relación con el voltaje en otra de sus entradas, mismo que previamente se fijó como referencia.
Uno de los métodos más comunes para implementar un comparador, incluye la utilización de un amplificador operacional; aunque también existen circuitos que ya tienen implementado un comparador en su interior.

Si se presenta ruido en la señal, el amplificador operacional y el comparador funcionarán de manera inadecuada, por lo que una solución para resolver este problema radica en auxiliarnos de la configuración de realimentación positiva. Sin

embargo, hay que notar que la realimentación positiva no elimina el ruido, sino que consigue que el amplificador sea más inmune a él.

Realimentación Positiva

La realimentación positiva se consigue tomando una porción del voltaje en la salida V_0 y conectado esta, en la entrada (+). Como se observa en la figura 4.11 (a) el voltaje de salida V_0 se reparte entre R1 y R2. Una parte de V_0 se utiliza para realimentar a la entrada (+) y se genera de esta manera un voltaje de referencia que estará en función de V_0. A continuación se describe como se fijan los umbrales para convertir a la señal.

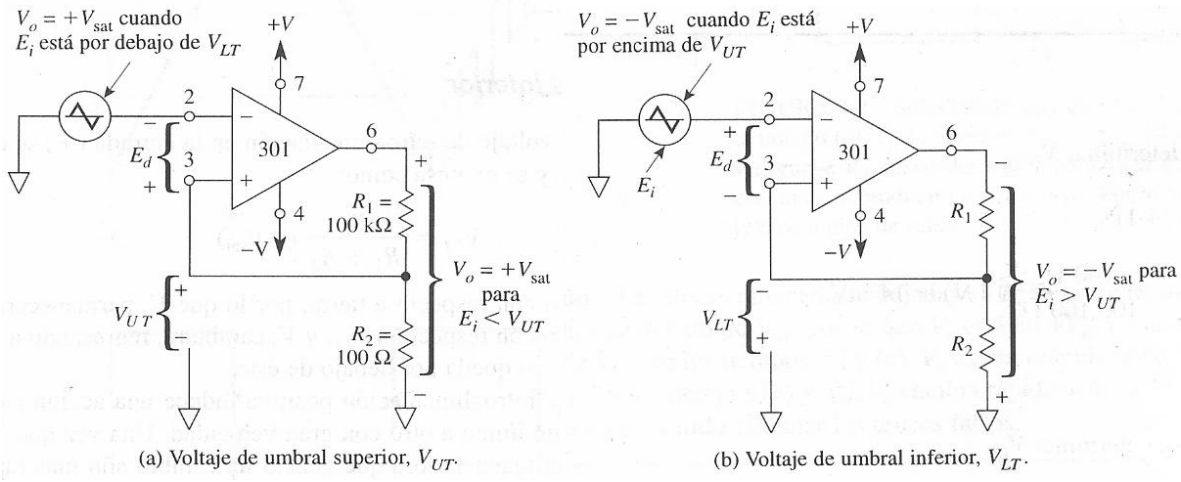


Figura 4.11, Configuraciones para la realimentación positiva.

Voltaje de umbral superior

Observado la figura 4.11 (a), recordamos que el voltaje de salida V_0 se reparte entre R1 y R2 donde una parte de V_0 se utiliza a manera de realimentación en la entrada (+). Cuando V_0=+Vsat el voltaje realimentado se conoce como umbral superior de voltaje, V_{UT}. Este voltaje se expresa en función de un divisor de voltaje, como se muestra a continuación:

$$V_{UT} = \frac{R_2}{R_1 + R_2}(+V_{SAT})$$

Para valores de E_i inferiores a V_{UT}, el voltaje en la entrada (+) es mayor que el voltaje en la entrada (-), por lo que, V_0 se establece como $+V_{SAT}$. Si E_i se vuelve más positivo que V_{UT}, la polaridad de E_d, tal como se observa, se invierte y entonces V_0 toma el valor de $-V_{SAT}$; de este modo el circuito es estable, según lo que se observa en la fig. 4.11 (b).

Voltaje de umbral inferior

Cuando V_0 se encuentra en $-V_{SAT}$ el voltaje de realimentación en la entrada (+, positiva) se conoce como umbral inferior de voltaje, V_{LT}, y se define como:

$$V_{LT} = \frac{R_2}{R_1 + R_2}(-V_{SAT})$$

Podemos notar que V_{LT} es negativo con respecto a tierra, por lo que V_0 se mantendrá en $-V_{SAT}$ mientras que E_i sea mayor o tenga un valor positivo con respecto a V_{LT} y por otro lado cambiará, regresando a $+V_{SAT}$, si E_i tiene un valor más negativo que V_{LT}.

Podemos concluir que la realimentación positiva produce una acción casi instantánea, lo que permite cambiar V_0 de un límite a otro con gran velocidad. Si los voltajes de umbral son mayores que los voltajes pico de ruido, la realimentación positiva logrará eliminar las transiciones falsas de salida.

Comparador implementado

En la figura 4.12 se muestra el comparador implementado.

Los valores de sus componentes se obtuvieron gracias a:

$$V_{UT} = \frac{1k\Omega}{10k\Omega + 1k\Omega}(5) = 0.4545V$$

$$V_{UT} = \frac{1k\Omega}{10k\Omega + 1k\Omega}(0) = 0.V$$

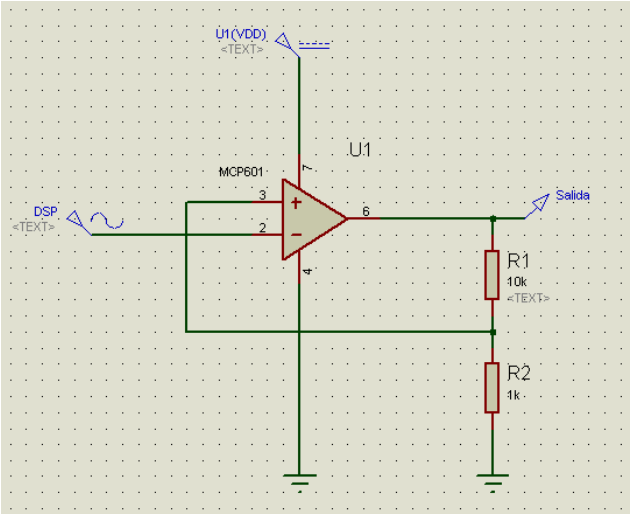


Figura 4.12, Comparador implementado.

Se utiliza un circuito que contiene un amplificador operacional (MCP601), ya que este circuito opera con 5V, mismo voltaje con el que operan los demás circuitos del proyecto; lográndose así una mejor integración.

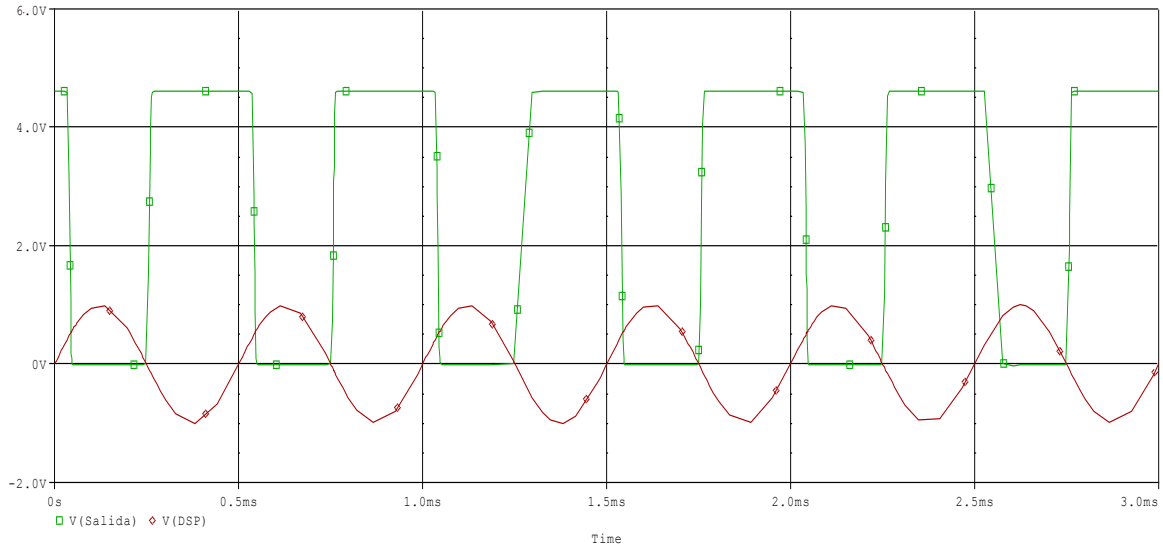


Figura 4.13, Modificación de la forma de onda de la señal.

En la figura 4.13 se observa cómo se modifica la forma de onda de la señal y se monta sobre un nivel de referencia de 0V, lo que facilita su interconexión con el PIC.

4.3.2 Algoritmo en el PIC

Se programó al PIC en modo sleep. Cuando el microcontrolador es despertado por el primer tren de pulsos, comienza la cuenta durante un segundo del número de pulsos, con lo que determina de qué instrucción se trata y activa o desactiva el pin correspondiente, del puerto D, un vez hecho esto regresa al modo sleep, aguardando a la siguiente instrucción para ser despertado. En las figuras 4.14 podemos ver el diagrama de flujo del programa.

Como se mencionó en el capítulo anterior se hace uso de interrupciones, contadores con preescaladores y de los puertos B y D. Cabe recordar que para el programa implementado se tomaron las consideraciones para utilizar un cristal del cuarzo de 4MHz.
Un led (en este caso de color verde) es encendido mientras el microcontrolador está recibiendo información del DSK C6711 y está en proceso de conteo.

Un led (en este caso de color rojo) es encendido cuando la instrucción enviada por el DSK, es la de un comando no reconocido o cuando la frecuencia de la señal sinusoidal esta fuera de las frecuencias permitidas.

Los bits de configuración para seleccionar el modo de oscilación, así como para apagar el Watchdog Timer son configurados por medio del software empleado para programar el microcontrolador PIC, en este caso el MPLAB, como se mencionó en el capítulo anterior, es por esto que no es necesario generar el código para establecer el estado de los bits correspondientes a las funciones mencionadas.

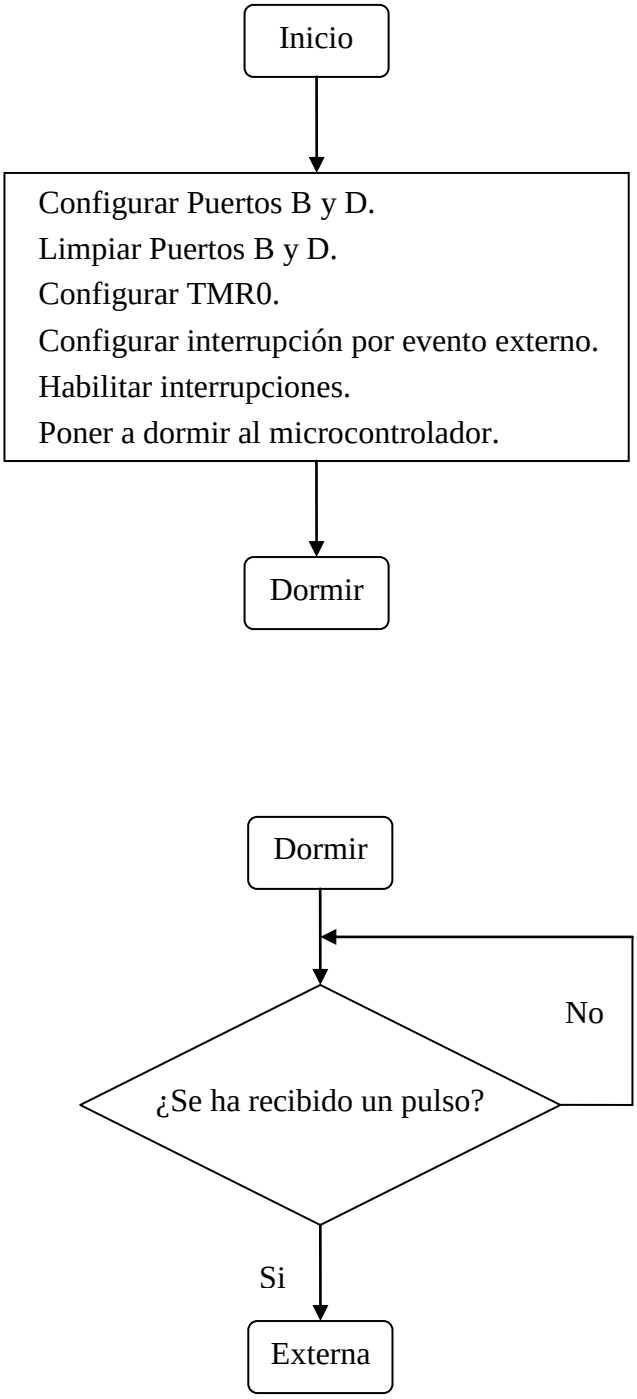


Figura 4.14 a), Diagrama de flujo del programa del PIC.

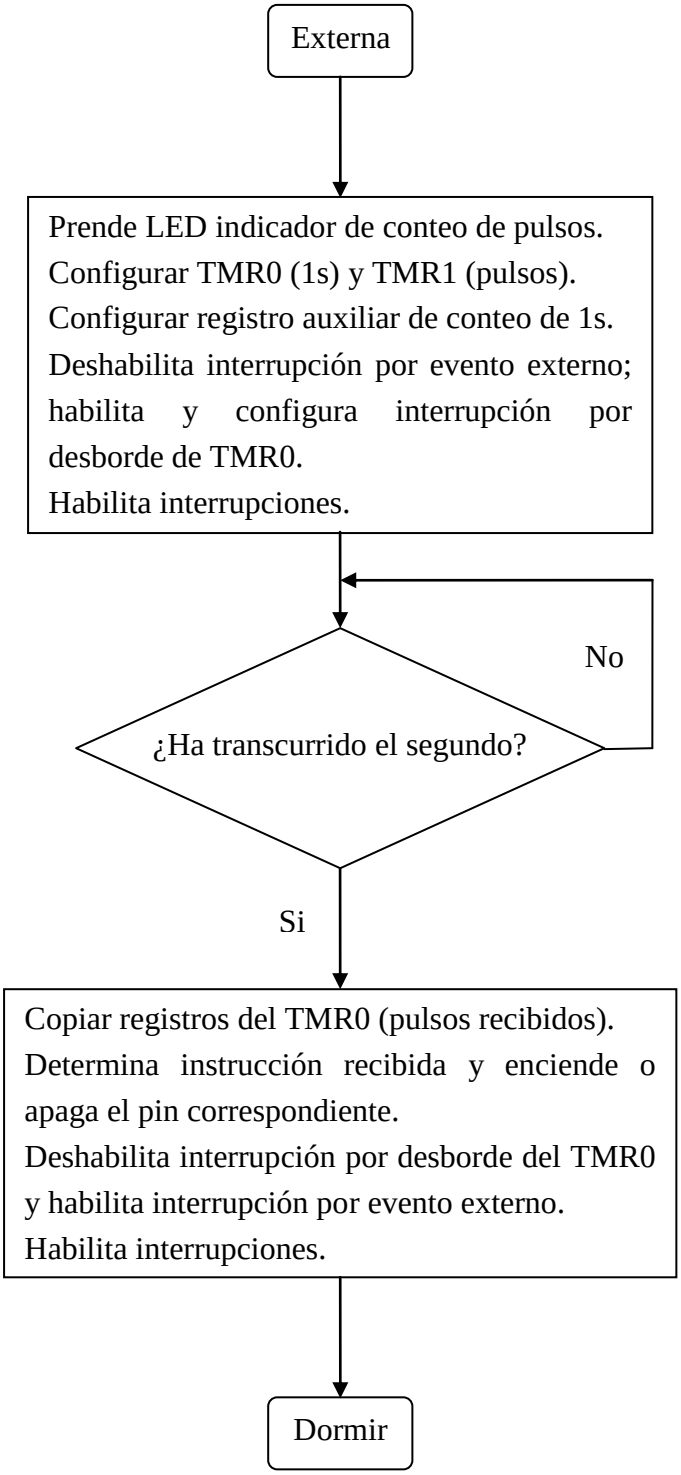


Figura 4.14 b), Continuacion del diagrama de flujo del programa del PIC.

4.4 Circuito eléctrico de iluminación

Para implementar el circuito eléctrico se tomaron las siguientes consideraciones:

- Circuito capaz de soportar 127 volts en corriente alterna.
- Circuitos de protección para la tarjeta DSK C6711

Dado que requeríamos controlar corriente alterna y un voltaje de 127 volts, es decir la tensión nominal de la línea, se dispuso la utilización de Triacs, que son un tipo de tiristores.

4.4.1 El tiristor

Un tiristor es un componente electrónico construido con elementos semiconductores que utiliza realimentación interna para producir una conmutación; en la figura 4.15 podemos ver su símbolo. Los materiales de los que se compone son de tipo semiconductor, es decir, dependiendo de la temperatura a la que se encuentren pueden funcionar como aislantes o como conductores. Es un dispositivo unidireccional, ya que únicamente trasmite la corriente en una dirección. Se emplea generalmente para el control de potencia eléctrica.

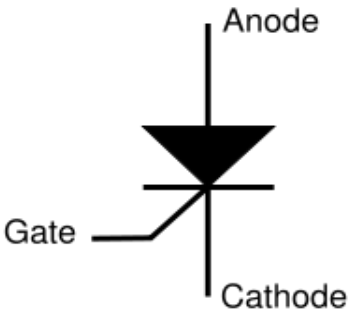


Figura 4.15, Simbolo del Tiristor.

El tiristor es un conmutador biestable, es decir, es el equivalente electrónico de los interruptores mecánicos; por lo tanto, es capaz de dejar pasar plenamente o bloquear por completo el paso de la corriente.

El diseño del tiristor permite que este pase rápidamente a encendido al recibir un pulso de corriente en su terminal de control, denominada "gate", cuando hay una tensión positiva entre ánodo y cátodo, es decir la tensión en el ánodo es mayor que en el cátodo. Solo puede ser apagado con la interrupción de la fuente de voltaje, abriendo el circuito o bien haciendo pasar una corriente en sentido inverso por el dispositivo.

Se deben tener en cuenta los siguientes puntos para diseñar el circuito de control de compuerta (gate):

- La señal de compuerta debe retirarse después de que haya encendido el tiristor. Una señal de control continua aumentaría la pérdida de potencia en la unión de la compuerta.
- A pesar de que el tiristor esta polarizado en sentido inverso, no debe haber señal de compuerta, ya que de lo contrario se puede ocasionar una falla a causa de un aumento en la corriente de fuga.
- El ancho de pulso en la compuerta tg puede ser mayor que el tiempo necesario para que la corriente anódica aumente hasta el valor de la corriente de retención IH. En la práctica, el ancho TG del pulso se hace, en operación normal, mayor que el tiempo de activación tON del tiristor.

4.4.2 Tipos de tiristores

Se encuentran en el mercado diversos tipos de tiristores, de acuerdo con su construcción y el comportamiento durante el encendido y el apagado, se pueden clasificar de la siguiente manera:

- Tiristores controlados por fase (SCR, por sus siglas en inglés "silicon controlled rectifier").

- Tiristores bidireccionales controlados por fase (BCT, por sus siglas en ingles "bidirectional phase-controlled").
- Tiristores de conmutación rápida (ASCR).
- Rectificadores controlados de silicio fotoactivados (LASCR, por sus siglas en ingles "light-activated SCR").
- Tiristores de tríodo bidireccional (TRIAC).
- Tiristores de conducción en sentido inverso (RCT, "reverse-conducting thyristor").
- Tiristores apagados por compuerta (GTO).
- Tiristores controlados por FET (FET-CTH, "FET-controlled thyristor").
- Tiristores de apagado por MOS (MTO," MOS turn off").
- Tiristores de apagado por emisor (ETO, "emitter turn off").
- Tiristores conmutados por compuerta integrada (IGCT, "integrated gate-conmutated thyristor").
- Tiristores controlados por MOS (MCT, "MOS-controlled thyristor").
- Tiristores de inducción estática (SITH, "static induction thyristor").

Tiristores de tríodo bidireccional (TRIACS)

Un TRIAC puede conducir en ambas direcciones, y se utiliza normalmente para efectuar control por fase, como en el caso de controladores de AC. Se puede considerar como dos SCR conectados en antiparalelo en conexión de compuerta común, tal como se observa en la figura 4.16 (a). Las características v-i se observan en la figura 4.16 (c).

Debido a que un TRIAC es un dispositivo bidireccional, no se puede decir que sus terminales sean ánodo y cátodo. Si la terminal MT2 es positiva con respecto a la terminal MT1, el TRIAC se podrá encender aplicando una señal positiva entre la compuerta G y la terminal MT1. Si la terminal MT2 es negativa con respecto a la terminal MT1, se logrará encender aplicando una señal negativa entre la compuerta G y la terminal MT1. No es indispensable tener las dos polaridades de señal de compuerta, y un TRIAC se puede encender con una señal de compuerta ya sea positiva o negativa.

En la práctica la sensibilidad varia de un cuadrante a otro, y los TRIACS se suelen operar en el primer cuadrante (voltaje de compuerta y corriente de compuerta positivos), o en el tercer cuadrante (voltaje y corriente de compuerta negativos).

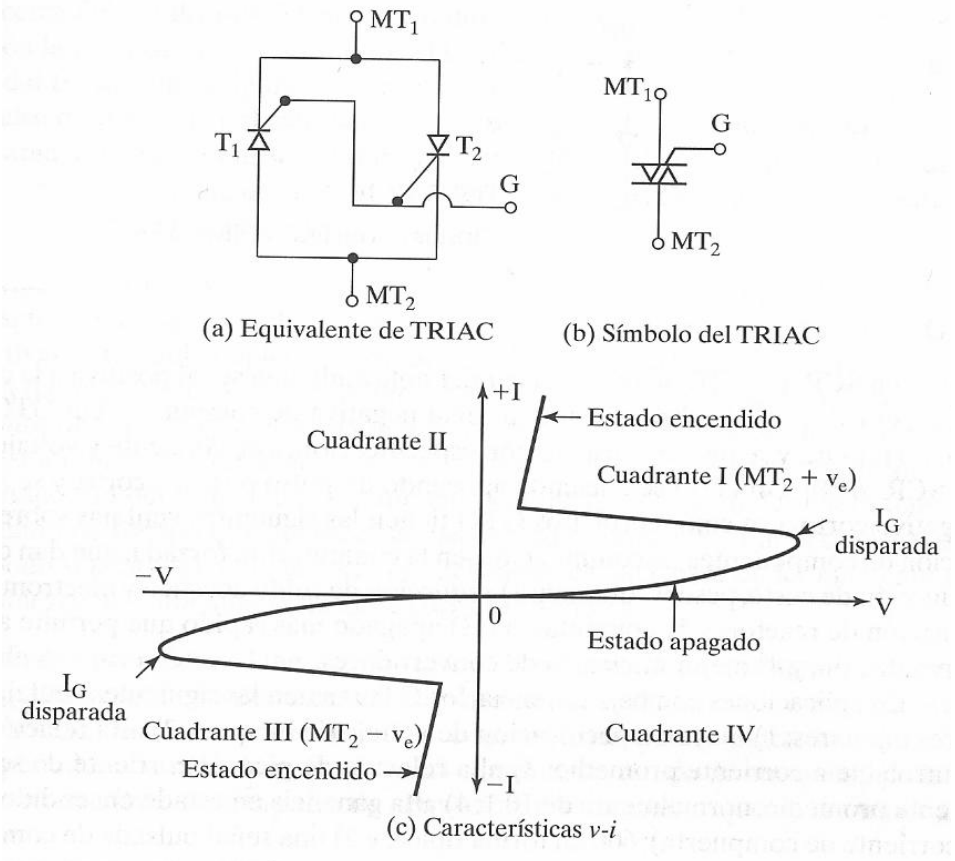


Figura 4.16, Comportamiento del triac en los diferentes cuadrantes.

4.4.3 Características del triac seleccionado

Para implementar el proyecto se seleccionó el TRIAC 2N6071A, cuyas terminales se observan en la figura 4.17.

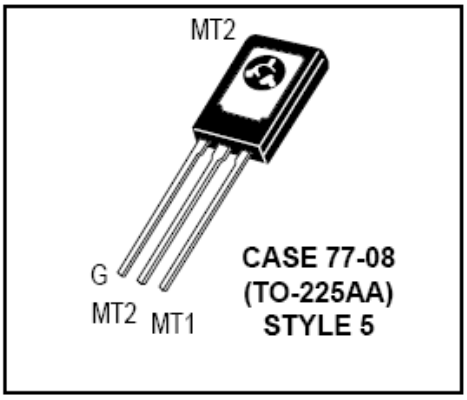


Figura 4.17, Terminales del triac.

En la tabla 4.2 se muestran los valores máximos que soporta:

Tabla 4.2, Valores máximos de operación del Triac 2NA6071A.

MAXIMUM RATINGS: (T_J=25°C unless otherwise noted)		**2N6071 2N6071A 2N6071B**	**2N6073 2N6073A 2N6073B**	**2N6075 2N6075A 2N6075B**	
	SYMBOL				**UNITS**
Peak Repetitive Off-State Voltage	V_{DRM}, V_{RRM}	200	400	600	V
RMS On-State Current (T_C=85°C)	$I_{T(RMS)}$		4.0		A
Peak One Cycle Surge (60Hz, T_J=110°C)	I_{TSM}		30		A
I^2t Value for Fusing (t=8.3ms)	I^2t		3.7		A^2s
Peak Gate Power (T_C=85°C)	P_{GM}		10		W
Average Gate Power (t=8.3ms, T_C=85°C)	$P_{G(AV)}$		0.5		W
Peak Gate Voltage (T_C=85°C)	V_{GM}		5.0		V
Storage Temperature	T_{stg}		-40 to +150		°C
Junction Temperature	T_J		-40 to +110		°C
Thermal Resistance	Θ_{JC}		3.5		°C/W
Thermal Resistance	Θ_{JA}		75		°C/W
Maximum Lead Temperature	T_L		260		°C

4.4.4 Optoacopladores (MOC)

Se trata de un circuito integrado que contiene en su interior un LED infrarrojo y un foto sensor, el cual puede ser un diodo de silicio, un transistor par Darlington o un SCR. Estos dispositivos operan con tiempos de respuesta tan pequeños que se pueden utilizar para transmitir datos en el rango de los MHz.

La utilización de un dispositivo como este nos proporciona un aislamiento eléctrico entre el circuito eléctrico y el circuito electrónico de control. Por otro lado, este aislamiento es útil para reducir el efecto que tiene la línea eléctrica sobre las transmisiones de señales en el sistema, así como para reducir el riesgo de una posible descarga en el circuito electrónico.

En este proyecto se utilizan optoacopladores MOC3010, que tienen salida a TRIAC, ya que se buscaba aislar el circuito de reconocimiento de comandos y el PIC del circuito de potencia que contiene los focos, y que por lo tanto maneja voltajes elevados. En la figura 4.18 se observa el diagrama de conexiones utilizado.

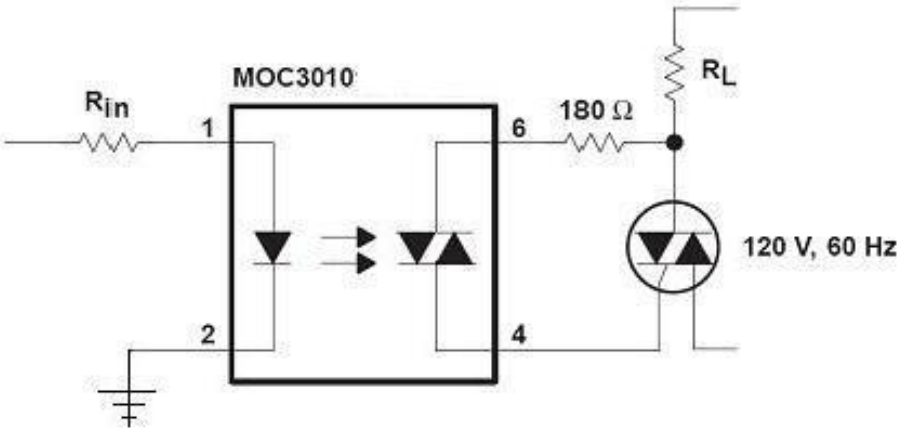


Figura 4.18, Diagrama de conexiones utilizado para el optoacoplador.

En la terminal 1 de cada optoacoplador está conectada una resistencia y esta a su vez va conectada al pin correspondiente del microcontrolador, la terminal 2 está conectada a tierra y en las terminales 4 y 6 va conectado el TRIAC antes descrito, por el que circula la corriente que permite el encendido del foco, R_L, en este caso con tensión nominal de 127 V y 100 W.

4.5 Circuito Final

En la figura 4.20 se muestra el esquemático del circuito empleado tanto para la interfaz (decodificación de las señales provenientes del DSK C6711) como para habilitar los dispositivos que controlan encendido de los focos dentro del circuito eléctrico de potencia. Cabe destacar que en este circuito se incorporó un preamplificador para la señal de un micrófono, que está alimentado con la misma fuente y que se conecta en la entrada de audio de la tarjeta del DSK C6711.

4.5.1 Preamplificador para la señal del micrófono.

Esta diseñado auxiliándonos del circuito integrado LM386; el cual es un amplificador de potencia, para audio y de bajo voltaje. En la figura 4.19 se muestra el diseño que se tomó como base para implementar el preamplificador.

Las principales características por las que se seleccionó son:

- Operación con voltajes bajos (en este caso de 5 V, lo que permite una buena integración con el circuito de interfaz-decodificación).
- Opera dentro del rango de frecuencias de la voz, ya que es un amplificador de audio.
- Bajo consumo de corriente.
- Requiere de mínimos componentes adicionales para su operación.

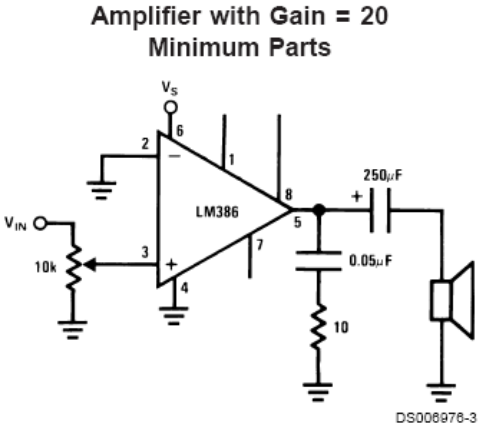


Figura 4.19, Diseño de un preamplificador para audio.

4.5.2 Circuito Esquemático

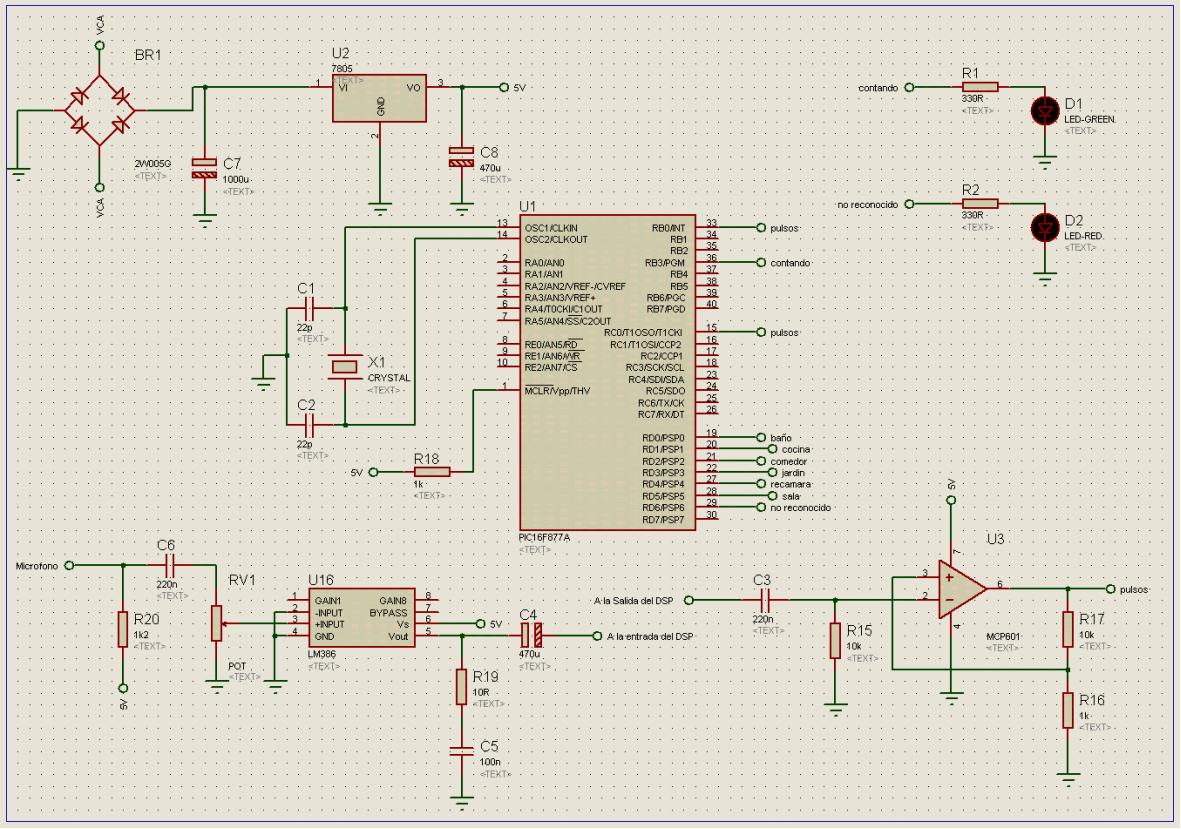


Figura 4.20, Esquemático del circuito electrónico de decodificación y control.

En la figura 4.21 se observa el esquemático del circuito eléctrico y electrónico de potencia.

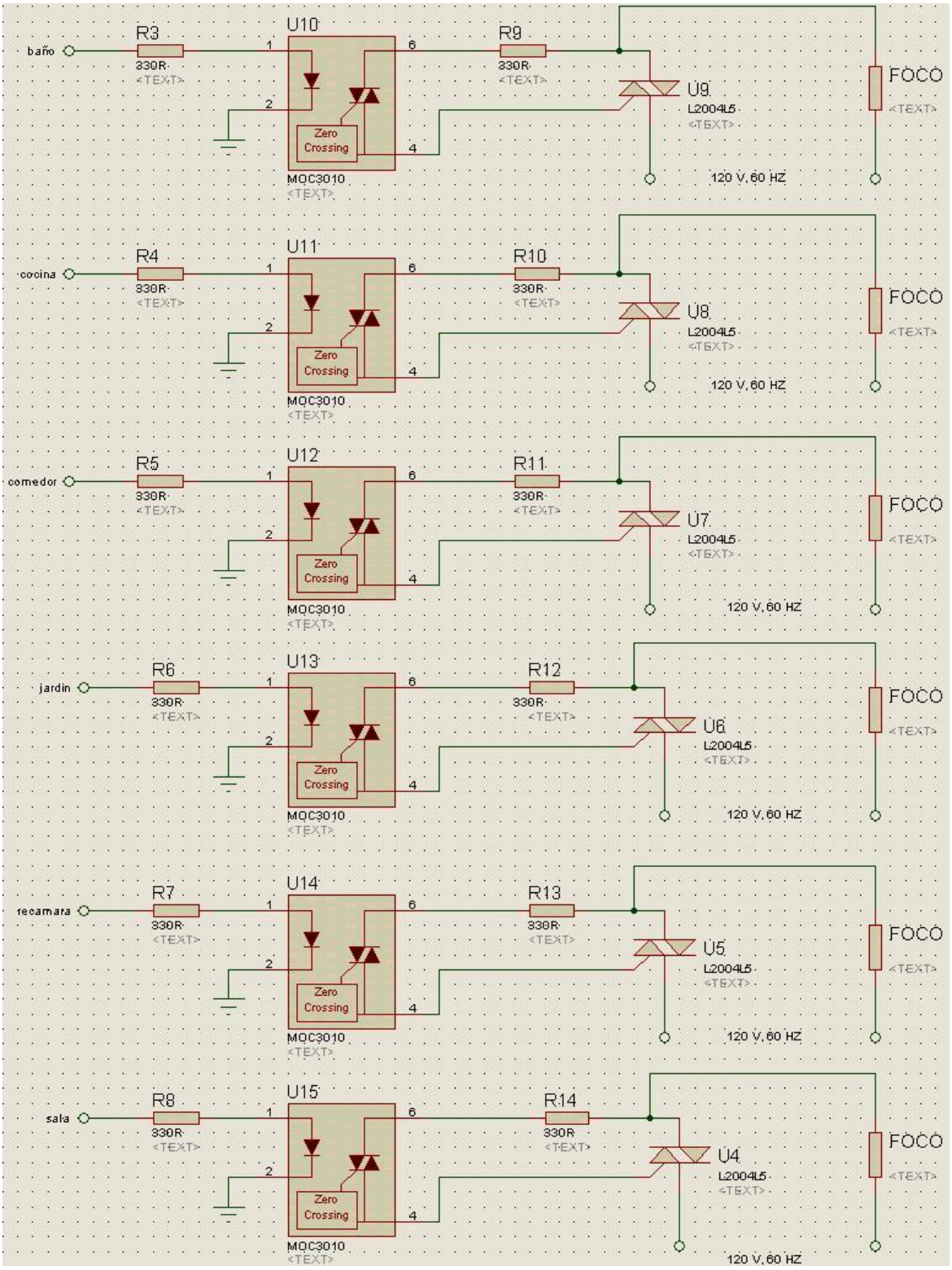


Figura 4.21, Esquemático del circuito electrónico de potencia.

V. Resultados, Conclusiones y Trabajo a Futuro

La etapa de reconocimiento de palabras aisladas en tiempo real, mediante el DSK 6711, se diseñó para identificar 12 palabras diferentes de un mismo locutor, con el uso de vectores LPC y la técnica de K-medias para la VQ.

En la fase de entrenamiento se utilizaron 24 repeticiones de cada palabra. Se dividieron las repeticiones de cada palabra en 4 segmentos de manera lineal. Cada segmento fue agrupado en 16 centroides por medio del algoritmo de K-medias.

5.1. Porcentajes de Reconocimiento

El sistema se probó con 20 pronunciaciones de cada una de las palabras previamente entrenadas, dentro de un entorno de ruido normal en el Laboratorio de Procesamiento de Voz.

Se realizaron dos tipos de pruebas: una sin alterar las condiciones de ruido ambiente y una segunda alterando el ruido ambiente mediante reproducir música a un volumen medio-alto.

La tabla 5.1 muestra los porcentajes de reconocimiento obtenidos al realizar la prueba sin alterar las condiciones de ruido ambiente: el 97.5% de las palabras fueron reconocidas al primer intento, el 2.5% de las palabras fueron reconocidas al segundo intento; finalmente, no se detecto ninguna palabra desconocida.

El segundo experimento, cuyos resultados se observan en la tabla 5.2, arrojó buenos resultados considerando que se alteraron las condiciones de ruido ambiente: el 92.9% de las palabras fueron reconocidas al primer intento, el 4.6% de las palabras fueron reconocidas al segundo intento y el 2.5% de las palabras se detectaron como palabras desconocidas.

Tabla 5.1, Resultados obtenidos en condiciones normales.

Palabra	Palabra reconocida al 1er intento	Palabra reconocida al 2do intento (Repetir Palabra)	Palabra desconocida
Apaga Baño	20	0	0
Apaga Cocina	19	1	0
Apaga Comedor	20	0	0
Apaga Jardín	19	1	0
Apaga Recamara	20	0	0
Apaga Sala	19	1	0
Enciende Baño	19	1	0
Enciende Cocina	19	1	0
Enciende Comedor	20	0	0
Enciende Jardín	20	0	0
Enciende Recamara	19	1	0
Enciende Sala	20	0	0
Total	234	6	0
Porcentajes	97.5%	2.5%	0.0%

Tabla 5.2, Resultados obtenidos con música como ruido de fondo.

Palabra	Palabra reconocida al 1er intento	Palabra reconocida al 2do intento (Repetir Palabra)	Palabra desconocida
Apaga Baño	20	0	0
Apaga Cocina	16	2	2
Apaga Comedor	19	1	0
Apaga Jardín	16	2	2
Apaga Recamara	20	0	0
Apaga Sala	18	1	1
Enciende Baño	19	1	0
Enciende Cocina	19	1	0
Enciende Comedor	20	0	0
Enciende Jardín	20	0	0
Enciende Recamara	18	2	0
Enciende Sala	18	1	1
Total	223	11	6
Porcentajes	92.9%	4.6%	2.5%

Al analizar los resultados pudimos notar que en el segundo caso, si la canción no cambia mucho de ritmo no se presentan mayores problemas, sin embargo, cuando hay mas presencia de voz en relación a melodía se debe hablar un poco más fuerte, para evitar confusiones en el sistema. De manera similar, cuando hay menos presencia de voz no se debe hablar tan fuerte para no saturar el amplificador del micrófono.

5.2. Conclusiones

Con el desarrollo de este proyecto se logró implementar el prototipo de un sistema de iluminación controlado por medio de comandos de voz. Esto se logró con ayuda de un DSK 6711 (DSP Starter Kit de Texas Instruments), en donde se efectúa el reconocimiento de los comandos en tiempo real, y circuitos adicionales para lograr la interfaz entre esta tarjeta de desarrollo y el circuito eléctrico de iluminación.

Con el prototipo que se tiene es posible conectar al sistema un micrófono alámbrico o inalámbrico para dar los comandos, lo que permite al usuario tener libertad de movimiento dentro del rango de recepción de la señal del micrófono (en el caso inalámbrico). En la fase de entrenamiento se utilizaron 24 repeticiones de cada palabra. Durante la etapa de extracción de características se utilizaron: el análisis LPC, la técnica de K-medias y la distancia Itakura, para la cuantización vectorial, VQ. Se utiliza un sistema VQ, ya muy antiguo, porque para comandos nos proporciona una velocidad del doble de un sistema DHMM, y 10 veces más rápido que un sistema DTW.

El sistema se probó con varias repeticiones de los comandos, lográndose un porcentaje de reconocimiento cercano al 98%, no alcanzándose el 100% porque se tomó como éxito el que el sistema reconociera el comando desde el primer intento y no cuando el sistema pide al usuario repetir el comando. Lo anterior debido a que por seguridad se fijó un umbral para detectar los casos que queden fuera de los comandos, o casos en que el comando reconocido y un segundo comando estén muy cercanos en sus distancias de clasificación.

Para la interfaz entre el DSK (en donde se lleva a cabo el reconocimiento del comando) y el circuito de iluminación, se utilizó un PIC, mismo que nos permitió decodificar las instrucciones y por otro lado enviar la señal que enciende o apaga el

foco dentro del circuito de iluminación. Con la utilización del PIC se logró implementar un circuito muy eficiente, que resultó ser económico y de tamaño pequeño; además, deja abierta la posibilidad de implementar otras opciones en el futuro gracias a las capacidades del PIC.

Finalmente, cabe destacar que se cubrieron los requerimientos de potencia requeridos para manejar niveles de voltaje nominales (127 V), poniendo atención a la protección de los circuitos electrónicos. Para tal efecto se utilizaron optoacopladores, los cuales permiten aislar el circuito eléctrico del electrónico.

5.3. Trabajo a Futuro

A pesar de que este sistema cumple con los objetivos trazados, es importante destacar que se pueden implementar mejoras en varias etapas. Por ejemplo, una siguiente fase del proyecto es poder lograr la independencia del sistema de una computadora, necesaria para poder ejecutar los programas dentro del DSK. Una posibilidad es migrar los algoritmos que se tienen a un dsPIC. Otra fase es el lograr que el sistema sea independiente del locutor, o que el entrenamiento requerido para identificar diversos usuarios sea mínimo.

Otra posible mejora radica en colocar un mezclador a la entrada del micrófono, lo que nos permitiría colocar micrófonos en distintos puntos de la casa y no sólo en un punto de control central como es el caso.

Dada la lógica empleada para el desarrollo del sistema, se puede ampliar el rango de aplicación del mismo realizando pequeñas modificaciones, tal como sería un eventual caso en que en lugar de encender un foco de 100 W se encendiera un aparato electrodoméstico.

En caso de lograr la independencia de la computadora y migrar los algoritmos a un dsPIC, se tendría un circuito de tamaño pequeño que nos permitiría implementarlo en gran número de aplicaciones.

Finalmente quiero mencionar que me resultó muy gratificante el poder aplicar los conocimientos adquiridos a lo largo de mi carrera en un proyecto de aplicación real. Obteniendo un buen sistema de reconocimiento, ya que, si bien existen otros sistemas en operación estos no son desarrollados en México y prácticamente en su totalidad no son para el español de México. Espero que este sistema sea tomado como base para futuros desarrollos.

Bibliografía

R. L. Boylestad, L. Nashelsky, "Electrónica: teoría de circuitos y dispositivos electrónicos", Prentice Hall, octava edición.

Deller, Proakis y Hansen, "Discrete-Time Processing of Speech Signals", Prentice Hall, tercera edición, 1987.

A. Gersho, R. M. Gray, "Vector Quantization and Signal Compression", Kluwer Academic Publishers", sexta edición, 1997.

J. A. Herrera, "Apuntes de la materia de Procesamiento de Voz". FI, UNAM.

O. Nieto, V. López, "Reconocimiento de comandos verbales en DSP's", Tesis, Facultad de Ingeniería, UNAM, 2002.

L. Rabiner, B. H. Juang, "Fundamentals of Speech Recognition", Prentice Hall, 1993.

A. S. Sedra, K. C. Smith, "Circuitos microelectrónicos", McGraw-Hill, quinta edición.

G. J. Tortora, N.P. Anagnostakos, "Principios de anatomía y fisiología", Oxford University Press, novena edición, 2006.

"Code Composer Studio Tutorial", Texas Instruments, SPRU301C.

"TMS320 DSP/BIOS User's Guide", Texas Instruments, SPRU423.

"TMS320c6711x Floating Point DSP's, Texas Instruments, SPRS0880.

"TMS320C67x DSP CPU and Instruction Set Reference Guide", Texas Instruments, SPRU733.

"PIC16F87X Data Sheet", Microchip.

"LM386 Low Voltage Audio Power Amplifier Data Sheet", National Semiconductor.

"MOC3010 Optoisolator Triac Driver Output Data Sheet", Motorola Semiconductors.

"2N6071A Sensitive Gate Triac Data Sheet", Motorola Semiconductors".