UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

“Decodificador de Tonos Duales Multifrecuencia mediante un microcontrolador embebido en un adaptador USB”

PRESENTADO POR:

IVAN STANLEY CARMONA PERAZA

PARA OPTAR AL TITULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, ENERO DE 2009.

UNIVERSIDAD DE EL SALVADOR

RECTOR :

MSc. RUFINO ANTONIO QUEZADA SÁNCHEZ

SECRETARIO GENERAL :

LIC. DOUGLAS VLADIMIR ALFARO CHÁVEZ

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO :

ING. MARIO ROBERTO NIETO LOVO

SECRETARIO :

ING. OSCAR EDUARDO MARROQUÍN HERNÁNDEZ

ESCUELA DE INGENIERÍA ELÉCTRICA

DIRECTOR :

ING. JOSÉ WILBER CALDERÓN URRUTIA

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título :

"Decodificador de Tonos Duales Multifrecuencia mediante un microcontrolador embebido en un adaptador USB"

Presentado por :

IVAN STANLEY CARMONA PERAZA

Trabajo de Graduación Aprobado por :

Docente Director :

Dr. CARLOS EUGENIO MARTÍNEZ CRUZ

San Salvador, Enero de 2009

Trabajo de Graduación Aprobado por:

Docente Director :

Dr. CARLOS EUGENIO MARTÍNEZ CRUZ

AGRADECIMIENTOS.

Principalmente se lo he dedicado a Dios por brindarme la sabiduría para hacerle frente a este trabajo, al Maestro Jesús, a su madre la Virgen María, a la Virgen de Santa Rosa de Lima por su intercesión.

A mis padres y hermana por haberme apoyado a lo largo de mi carrera.

A todos mis amigos y conocidos también por brindarme ánimos en el transcurso de mi tesis.

RESUMEN

En el año 2002 se empezó dentro de la Escuela de Ingeniería Eléctrica, el desarrollo de instrumentos para el campo de la telefonía. Durante estos últimos años se han realizado importantes avances tanto en la construcción de hardware como en el desarrollo de software.
El objetivo principal del presente trabajo es el desarrollo de un decodificador de tonos duales multifrecuencia (DTMF) que cumpla con el estándar de la Unión Internacional de las Telecomunicaciones (ITU) Q.24. Para dar cumplimiento a este objetivo, el trabajo se ha desarrollado con la siguiente estructura: Información básica sobre la generación de los tonos DTMF, requisitos que debe cumplir el decodificador DTMF según el estándar Q.24 de la ITU, simulación del decodificador DTMF usando MATLAB e implementación del decodificador DTMF en un microcontrolador.

El primer capítulo es una recopilación de los aspectos teóricos sobre la codificación y decodificación DTMF y de los diversos métodos de decodificación de los tonos duales multifrecuencia DTMF ya existentes; también se da una explicación de los filtros de ranura y del algoritmo de cruce por cero para la estimación de frecuencia.

En el segundo capítulo se describe las diversas etapas que componen el decodificador DTMF y el rol que tiene cada etapa en el funcionamiento del decodificador DTMF.

En el tercer capítulo se explica el proceso de como se implementa el decodificador DTMF en un microcontrolador embebido en un adaptador USB.

Finalmente en el capítulo cuarto se presentan los resultados obtenidos de las diversas pruebas efectuadas al decodificador y propuestas de mejora.

TABLA DE CONTENIDO

Pagina.

LISTA DE FIGURAS

Figura *Página*

LISTA DE TABLA

Tabla *Página*

CAPITULO I

GENERALIDADES E INFORMACION BASICA

En este capítulo se presenta la teoría básica y especificaciones técnicas sobre los tonos DTMF. El capítulo I está formado por 5 partes: primero, se detalla los campos de aplicación y la forma de como se genera los tonos DTMF.
En la siguiente sección se detalla los requisitos que debe de cumplir el decodificador DTMF; en la tercera sección se detalla los diversos métodos de decodificación DTMF empleados.
En la cuarta sección damos una breve explicación de los filtros tipo ranura.
El texto para la sección 1 y 2 se basa principalmente en las recomendaciones de la Telecomunicaciones de la Unión Internacional de las Telecomunicaciones (UIT-T, antes CCITT), a las que se apegan los sistemas de comunicaciones a nivel mundial.

GENERACION DE TONOS DUALES MULTIFRECUENCIA .

Los tonos duales multifrecuencia (DTMF) son usados en el marcado telefónico, en maquinas contestadoras digital, en sistemas de banca interactiva, etc.
En el caso del marcado telefónico, cada tecla del teclado al presionarla le indica al teléfono que genere una señal DTMF, como a continuación se describe.

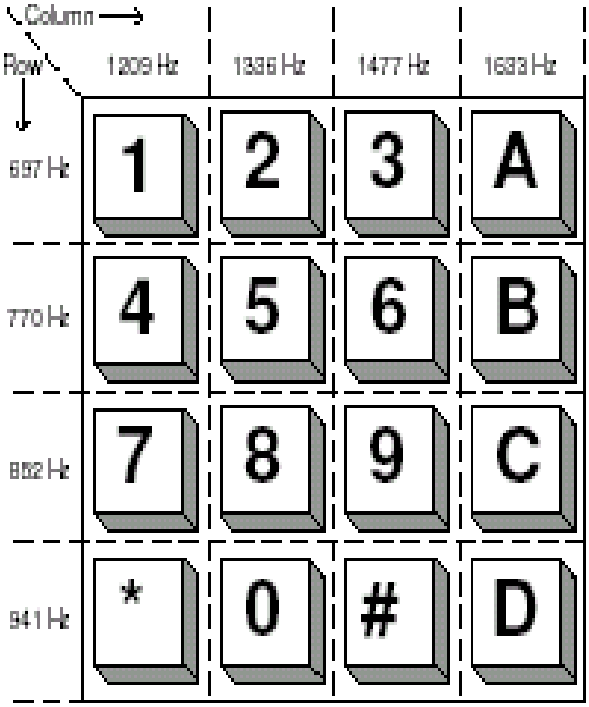


Figura 1. Teclado de un teléfono

Como observamos en la figura 1 el teclado esta compuesto por 4 filas y 4 columnas, cada fila tiene su frecuencia especifica las cuales se les conoce como grupo de frecuencias bajas y cada columna tiene su frecuencia especifica que también se le conoce como grupo de frecuencias altas.
Al presionar como ejemplo la tecla 5 el circuito del teléfono nos genera 2 ondas senoidales

una con una frecuencia de 770 Hz y la otra con una frecuencia de 1336 Hz, que posteriormente son sumadas para así generar el tono DTMF correspondiente a la tecla 5, tal como se muestra en la figura 2 a continuación:

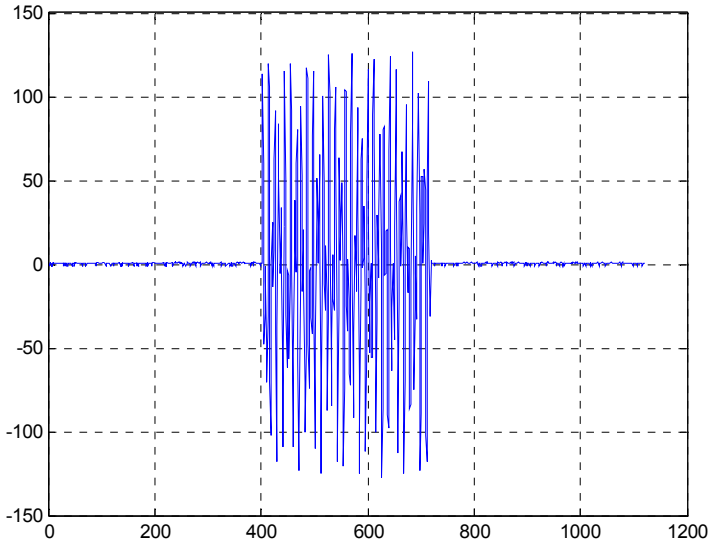


Figura 2. Tono DTMF para la tecla 5.

Las teclas A, B, C y D no son usadas en forma comercial sino que son teclas usadas para uso militar y aplicaciones de radio.
La tasa máxima de marcado es de 12.5 símbolos/ segundo.

DECODIFICACION DE TONOS DTMF.

Un decodificador de tonos DTMF debe de cumplir con el reglamento de la ITU Q.24 en la cual se describe los requisitos que debe de reunir el decodificador DTMF, en las cuales se detallan en la siguiente tabla.

ESPECIFICACIONES ITU PARA DETECCION DTMF.

Tolerancia en frecuencia	Un tono DTMF válido deberá tener un margen de desviación en frecuencia de un 1.5 % con respecto a su frecuencia nominal. Un tono con un margen de frecuencia de 3.5% podría no ser detectado.
Duración de la señal	Un tono válido DTMF con una duración de 40 ms. deberá ser considerado como válido. Tonos con una duración menor a 23 ms. podrían ser rechazado.
Interrupción de la señal	Una señal DTMF válida interrumpida por 10 ms o menos podría no ser detectada como 2 tonos DTMF distintos.
Pausa de la señal	Una señal DTMF válida separada por un tiempo de pausa de 40ms deberá ser detectada como 2 tonos DTMF .
Potencia de la señal	El decodificador podría trabajar en el peor caso de la relación SNR de 15 dB a una potencia de la señal de –26dBm (atenuación de 26 dB para 1 mW de potencia transmitida).
Twist	El decodificador deberá operar con un máximo de 8 dB en twist normal y a 4 dB de twist en reversa. Twist es definido como la diferencia en decibeles en las amplitudes de los 2 tonos fundamentales de la señal DTMF
Talk-off	El decodificador podría operar en la presencia de voz sin identificar incorrectamente la señal de voz como un tono DTMF válido.

Tabla1. Especificaciones ITU para decodificación DTMF.

METODOS DE DECODIFICACION.

En el mercado de productos telefónicos hay una gama de circuitos integrados que hacen la tarea de decodificar tonos DTMF y el método que ellos emplean es el de 8 resonadores implementado con tecnología de capacitores conmutados seguido por un procesador digital en el cual se encarga de medir la duración del tono y provee una salida digital codificada.

Otro método digital popular está basado en el uso de DFT que también usan 8 filtros IIR pasabanda resonantes . Para una DFT de longitud M, la M salida de cada filtro digital es el coeficiente DFT calculado por medio del algoritmo de Goertzel.

El algoritmo de Goertzel usa una palabra en ROM y 2 palabras de RAM y requiere de M+2 multiplicaciones y 2M+2 sumas por cada M muestras. El valor absoluto del filtro de salida es elevado al cuadrado para medir la energía de la señal presente a la frecuencia DTMF. Un par de tono DTMF es considerado válido si tiene suficiente energía en la frecuencia DTMF y que dicha energía exceda a las otras energías de los otros tonos DTMF.

Incrementado la longitud de DFT habrá un incremento en la resolución en frecuencia pero se decrementa la resolución con respecto al tiempo. Ciertas longitudes DFT sin embargo pueden hacer coincidir simultáneamente los requisitos de la ITU Q.24 con respecto al tiempo y a la frecuencia. Un símbolo DTMF es válido si este es válido durante 2 o más ventanas consecutivas.

La DFT no uniforme (NDFT) podría ser usada para calcular la energía de la señal a la frecuencia DTMF exacta. La aplicación directa de la NDFT en el decodificador no hará que el decodificador cumpla con los requisitos de frecuencia y de tiempo, pero usando dos NDFT de longitudes diferentes hará que si cumpla con los requisitos de la ITU para la decodificación pero con un costo de implementación de 15 multiplicaciones por muestra.

Las técnicas de descomposición sub-espacial tales como la clasificación de señales múltiples han sido aplicadas a la decodificación DTMF. Aunque las técnicas de

subespacio podrían cumplir con los requisitos de la ITU , su implementación con respecto al cálculo y de uso de memoria son enormes, generalmente requiere de aritmética de punto flotante y es dificultoso al implementarlo en tiempo real sobre microcontroladores embebidos.

FILTROS NOTCH (DE RANURA).

Es un filtro digital de tipo FIR que en su respuesta en frecuencia se comporta como un filtro pasatodo pero en una porción del espectro de frecuencia presenta una ranura es decir que a una frecuencia específica el filtro atenúa un cierto pequeño rango de frecuencia tal como se ilustra en la Figura 3.

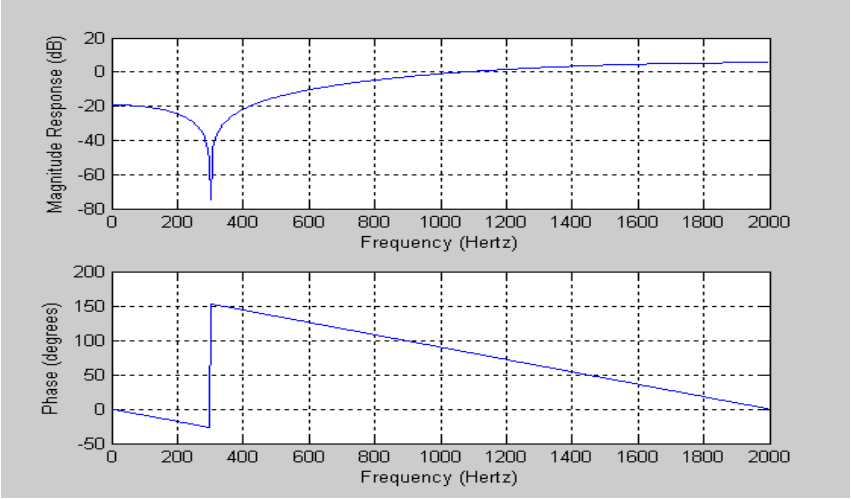


Figura 3.Respuesta en frecuencia del filtro notch

CAPITULO II

DESCRIPCION DEL DECODIFICADOR.

La siguiente etapa del trabajo consiste en la descripción del funcionamiento del decodificador DTMF usando como software de apoyo MATLAB.
En esta sección se muestra el diagrama en bloque que conforma el decodificador y a la vez se brinda información acerca del funcionamiento de cada etapa que conforma el decodificador .

DIAGRAMA EN BLOQUE DEL DECODIFICADOR DTMF

A continuación el diagrama en bloque del decodificador DTMF en la Figura 4.

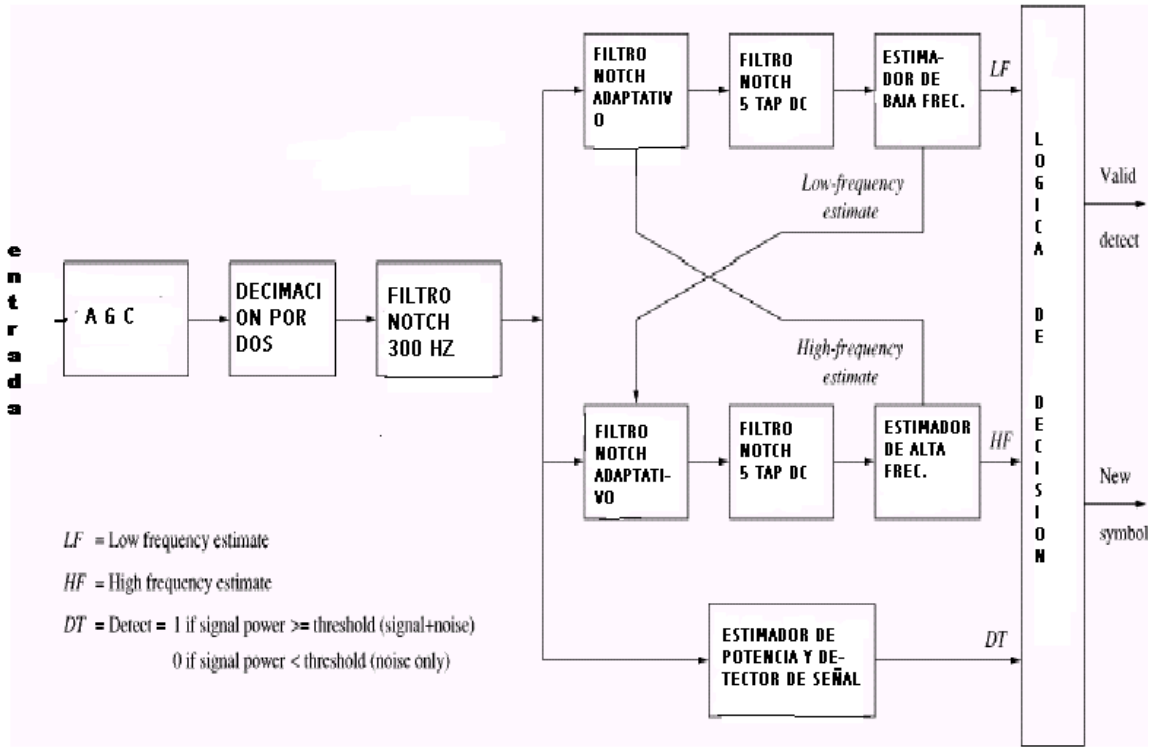


Figura 4. Diagrama en bloque del decodificador DTMF.

Una observación importante con respecto a este decodificador es que la señal DTMF tiene que haber sido muestreada a 8 KHz y la muestra es representada a 8 bits.

El rango del valor decimal de las muestras es de -127 a 128 y expresado en notación hexadecimal seria:

De 0 hasta 128 : 0h hasta 80h.

De –1 hasta –127: FFh hasta 81h.

ETAPA DE CONTROL AUTOMATICO DE GANANCIA (AGC).

Su rol es el de incrementar el rango dinámico efectivo, dicha etapa eleva el nivel de la señal de entrada por un factor de 4 (12 decibeles) si éste cae debajo de –24 decibeles y por un factor de 2 (6 dB) si dicha señal cae debajo de –18 dB ; en otras palabras si el valor de la muestra de entrada cae en el rango de –16 a 16 entonces a esas muestra se le aplica el AGC.

DIEZMADO POR DOS.

Uno de los objetivos del diezmador por dos es el de reducir el total de muestras a procesar es decir de 8000 muestras a procesar el decimador solo dejará pasar 4000 muestras, en otras palabras reduce la tasa de muestreo de 8000Hz a 4000 Hz y cada muestra estará presente en los registros destinados a esta etapa por el microcontrolador un tiempo de 250 useg.

El diezmado por dos nos ayuda a reducir la varianza del ruido en un 70 % lo cual nos hace incrementar el rango dinámico dado que en dicha etapa también hace uso de un filtro promediado por dos.

FILTRO NOTCH 300 HZ.

La función del filtro notch 300 Hz es el de suprimir el tono DTMF de marcado el cual esta formado por las frecuencias de 350 Hz y 440 Hz. Los supresores de tonos es común en los decodificadores DTMF pero no es requerido en el estándar de decodificación DTMF ITU Q.24.

El dato resultante de esta etapa es dirigido a tres ramas que son: la ruta de alta frecuencia, la ruta de baja frecuencia y la ruta estimador de potencia.

FILTRO NOTCH ADAPTATIVO.

Son filtros FIR de segundo orden. Dicho filtro tiene un par de cero conjugado complejo sobre el círculo unitario a $z = \exp[\pm j\omega_0]$, donde ω_0 es la frecuencia notch. La función de transferencia del filtro notch es :

$$H(z) = b_0(1 - 2\cos\omega_0 z^{-1} + z^{-2})$$

Seleccionamos $b_0 = 0.5$ como un factor de escala para prevenir el sobreflujo de la muestra. El filtro notch FIR tiene banda de rechazo centrado en ω_0 , por consiguiente este filtro atenúa otros componentes de frecuencia cerca del nulo, usamos esta propiedad para suprimir una componente de frecuencia del tono DTMF y solamente tendremos una componente de frecuencia. Parte de filtrado adaptativo se debe a que el filtro recibe una instrucción proveniente de la etapa del estimador de frecuencia que se encuentra localizado en la otra ruta para que coloque un nulo a cierta frecuencia dictada por el estimador de frecuencia. Los filtros notch adaptativo tienen un margen de frecuencia de 3.5 % de la frecuencia nominal.

Por ejemplo tomemos la tecla 2 la cual esta formada por las frecuencias de 697 Hz y de 1336 Hz, al entrar la señal en el filtro notch adaptativo localizado en la ruta de baja frecuencia, dicho filtro recibe la siguiente instrucción proveniente del estimador de frecuencia localizado en la ruta de alta frecuencia : "Coloca una ranura a 1336 Hz" es decir que dicho filtro atenúa la frecuencia de 1336Hz y por consiguiente el tono DTMF deja de existir y solamente tenemos una componente de frecuencia de 697 Hz que se dirige hacia el estimador de frecuencia.

FILTRO NOTCH DC DE 5 COEFICIENTES.

Una desventaja de los filtros notch adaptivos es que introducen a la señal de salida componente de dc y es necesario suprimir esa componente de dc para el correcto funcionamiento del estimador de frecuencia, el rol del filtro notch DC es suprimir la componente de DC.

El filtro notch DC en la ruta de datos de baja frecuencia presenta una alta ganancia en el espectro DTMF de baja frecuencia (debajo de 1000 Hz) y los coeficientes de dicho filtro son {0.25, 0.25, 0, -0.25, -0.25}.

El filtro notch DC en la ruta de datos de alta frecuencia presenta una alta ganancia en el espectro DTMF de alta frecuencia (arriba de 1000 Hz) y los coeficientes de dicho filtro son {0.25, -0.25, 0, 0.25, -0.25}.

La figura 5 muestra la respuesta de frecuencia para la ruta de baja frecuencia después que la tecla 1 (697 Hz y 1209 Hz) fue introducida y la frecuencia estimada se ha estabilizado.

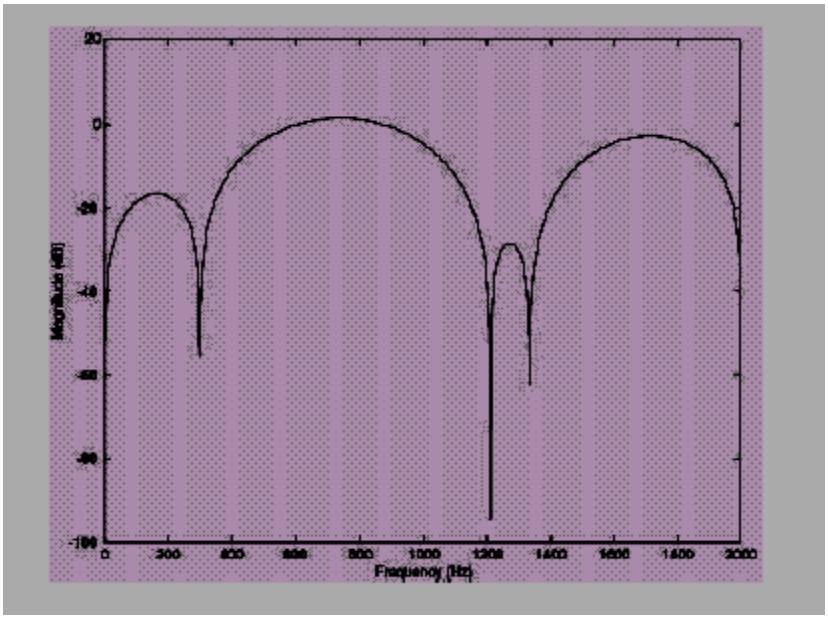


Figura 5. Respuesta en frecuencia de la rama de baja frecuencia.

ESTIMADOR DE FRECUENCIA.

En dicha etapa se analiza si la señal proveniente del filtro notch dc cumple con los requerimientos con respecto a la frecuencia según contemplados en el estándar de la ITU Q.24, dicha etapa se compone de tres subetapas.

El estimador de frecuencia está basado en un algoritmo de alta precisión y está basado en el cruce por cero de las muestras, para comprender mejor esta etapa se dará una explicación de cómo funciona este algoritmo en las siguientes subsecciones.

ALGORITMO DE CRUCE POR CERO.

Como se muestra en la figura 6, consideremos el cruce por cero de una muestra negativa hacia una muestra positiva.

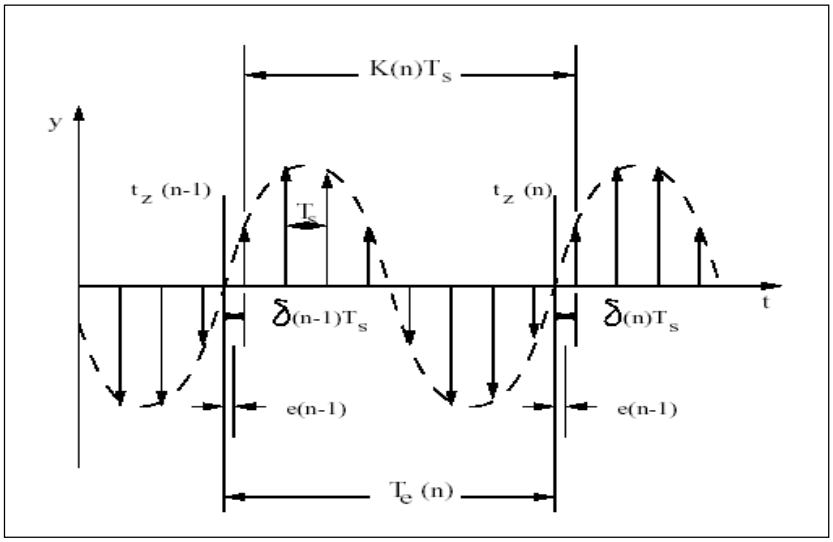


Figura 6. Estimación del periodo de una onda seno

A la llegada de la primera muestra positiva siguiente al cruce por cero, el periodo de la onda senoidal Te(n) es determinado por:

$$Te(n) = [K(n) - \delta(n) - \delta(n-1)]Ts$$

Donde:

Ts = Periodo de muestreo.

Tsin = periodo de la onda seno.

K(n) = Es el número de intervalos de muestras entre las muestras positivas siguientes a *(n-1)* y *(n)*.

$\delta(n)Ts$ y $\delta(n-1)Ts$: Son los intervalos de tiempo entre los cruces por cero y la siguiente muestra positiva.

Los intervalos de tiempo son determinados al aproximar la sinusoide con un segmento de línea comprendida entre las muestras de un lado del cruce por cero, tal como se muestra en la figura 7.

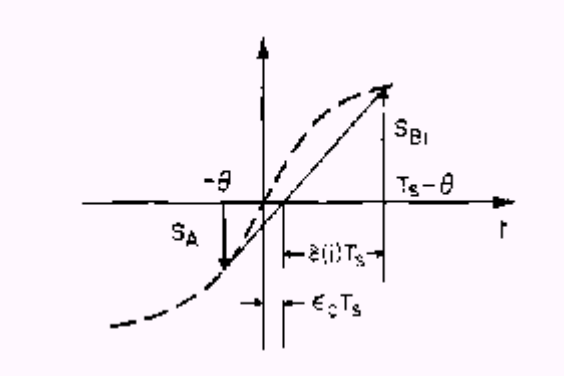


Figura 7. Cálculo del cruce por cero por interpolación lineal.

Donde δ(i) es determinado por:

$$\delta(i) = \frac{S_{Bi}}{S_{Bi} - S_{Ai}}$$

Donde S_{Ai} y S_{Bi} son las magnitudes de las muestras precedentes y siguientes al cruce por cero.

La estimación del periodo de la sinusoide es obtenido al pasar las muestras *Te(n)* a través de un filtro pasabajo.

Si ε(n) es el error obtenido en el cálculo del enésimo cruce por cero $t_z(n)$ en la figura 7.

$$t_z(n) = nT_{sin}$$

Entonces la estimación del periodo de la sinusoide Te es igual a :

$$Te(n) = t_z(n) - t_z(n-1) = T_{sin} + \varepsilon(n) - \varepsilon(n-1)$$

La transformada z de *Te(n)* y el espectro $|Te(f)|^2$ están dados por:

$$Te(z) = T_{sin} + \varepsilon(z)\,(1-z^{-1})$$

$$|Te(f)|^2 = T_{sin}^2\,\delta(f) + |\varepsilon(f)|^2\,4\sin^2(\pi f / f_{sin})$$

El espectro de *Te(f)* se muestra en la figura 8.

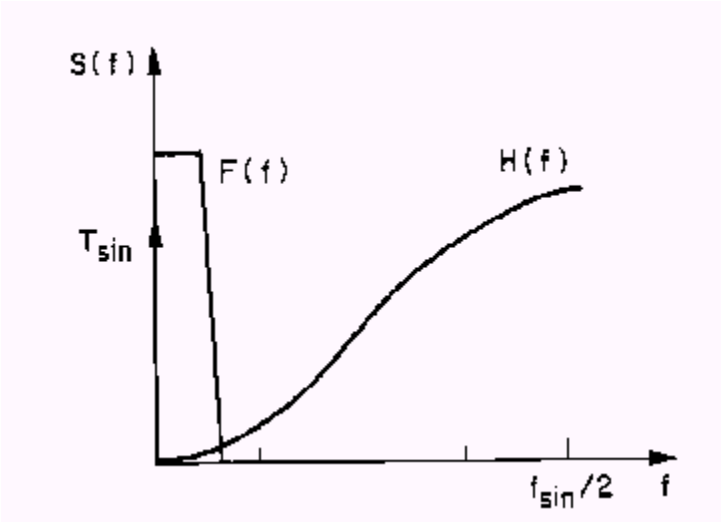


Figura 8 . Espectro de la señal de salida.

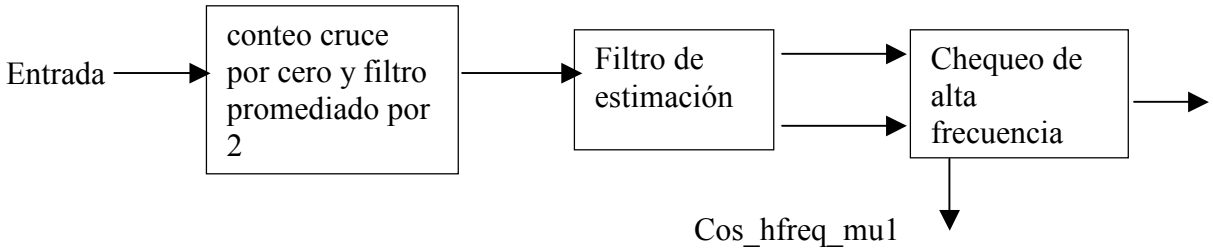
Al observar la figura 8 vemos que el espectro de Te (f) contiene una componente de DC y que es igual al periodo de la señal sinusoidal y al espectro de la señal de error que está conformada por:

$$|H(f)| = 4\sin^2(\pi f / f_{sin})$$

Debido a que la contribución del error en dc es cero teóricamente, usamos un filtro pasabajo apropiado F (f) mostrado en la Figura 8 para remover el ruido en la región de alta frecuencia y por consiguiente el periodo puede ser calculado con un alto grado de precisión

DIAGRAMA EN BLOQUE DEL ESTIMADOR DE FRECUENCIA.

A continuación mostramos el diagrama en bloque del estimador de frecuencia en la ruta de alta frecuencia.



En el bloque del conteo de cruce por cero, esta etapa cuenta los cruces por cero de una muestra de un valor negativo hacia una muestra con valor positivo. De la explicación del algoritmo del cruce por cero se nos dice que el periodo estimado debe pasarse a través de un filtro pasabajo lo cual en el decodificador se implementa en dos etapas, una que es el filtro promediado por 2 y posteriormente pasa a la etapa del filtro de estimación y que es un filtro pasabajo IIR de primer orden cuya función de transferencia es :

$$F(n) = \alpha F(n-1) + (1 + \alpha)F(n).$$

Donde F (n) es el estimado del cruce por cero y seleccionamos α con el valor de 0.875 dado que con este valor nos da una gran precisión en la determinación de la estimación del periodo.

Como necesitamos una gran precisión en la determinación del periodo estimado, el valor que nos arroja sobre el periodo estimado estará conformado por un valor entero y por un

valor decimal o fraccionario, a continuación daremos un ejemplo de cómo está conformado el valor del periodo estimado.

Tomemos como ejemplo la tecla A y observamos que la frecuencia del grupo de alta frecuencia es de 1633 Hz, a continuación mostramos como es representada la frecuencia por medio del periodo estimado:

i- Determinamos los valores limites de la estimación de frecuencia (1633 ± 3.5 %) y nos arroja los valores de 1576 Hz y 1690 Hz.

ii- Determinamos el periodo para dichas frecuencias limites y nos da el valor de 6.35E-04 (para 1576 Hz) y 5.92E-04 (para 1690 Hz).

iii- Dividimos el periodo calculado en ii por 2 veces el periodo de muestreo (es debido a que la señal ha sido diezmada anteriormente) es decir dividir entre 250 useg. (2*125 useg.) y nos da el valor de 2.538 (para 1576 Hz) y de 2.3668 (para 1690 Hz).

iv- Observamos que el dato 2.538 representa a la frecuencia (1633 – 3.5 %)Hz y notamos que la parte entera tiene el valor de 2 y la parte fraccionaria es 0.538 ; el dato 2.3668 representa la frecuencia de (1633 + 3.5%), la parte entera es 2 y la parte fraccionaria es 0.3668.

v- Como vamos a trabajar con datos a 8 bits , expresamos la parte fraccionaria a datos de 8 bits multiplicando la parte fraccionaria por 256 y nos da el valor de 138 para la frecuencia (1633 – 3.5%) y de 94 para la frecuencia de (1690 + 3.5%), es decir que la parte fraccionaria representada en dato a 8 bits estará variando en el rango de 94 a 138, tal como se muestra

en las figuras 9, 10 y 11 obtenidas en la simulación del decodificador en matlab.

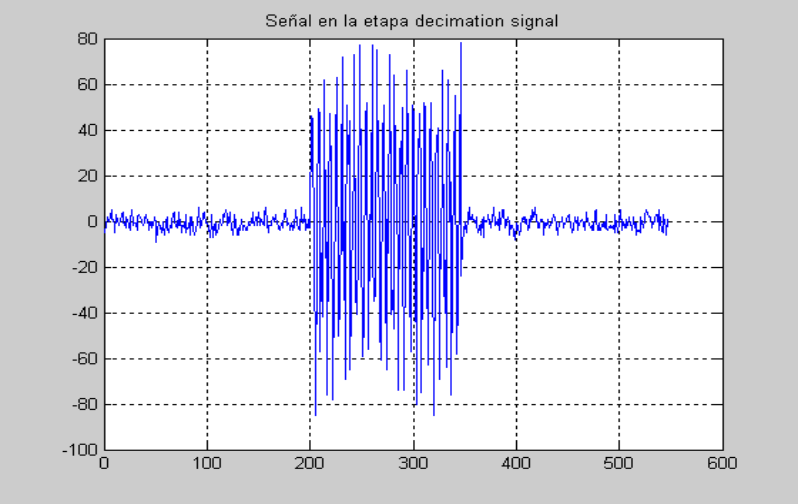


Figura 9 . Tono DTMF de la tecla A con SNR de 30.

Como vemos la parte activa del tono A ocurre desde la muestra 200 hasta la muestra 350.

En la figura 11 mostramos la parte entera del periodo estimado por el estimador de frecuencia.

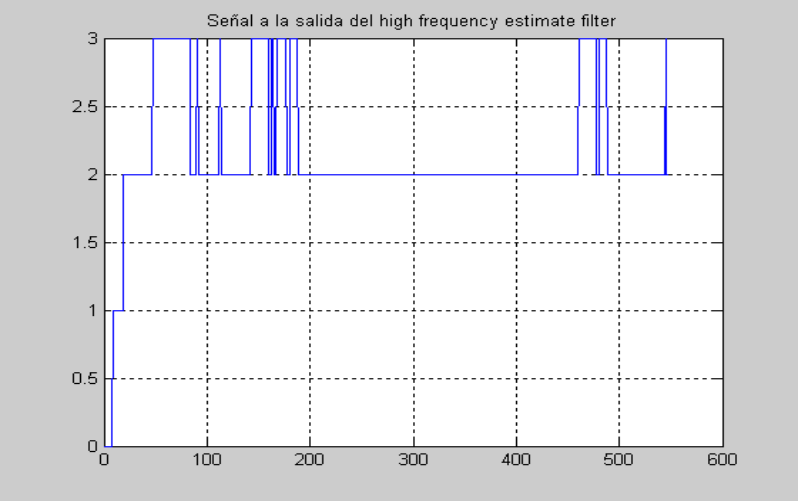


Figura 10 . Parte entera del periodo estimado para el tono A

Como observará en el rango de muestras de 200 a 350 vemos que la parte entera del periodo estimado en la ruta de alta frecuencia tiene un valor constante de 2 y que cumple con lo plasmado en el inciso iv.

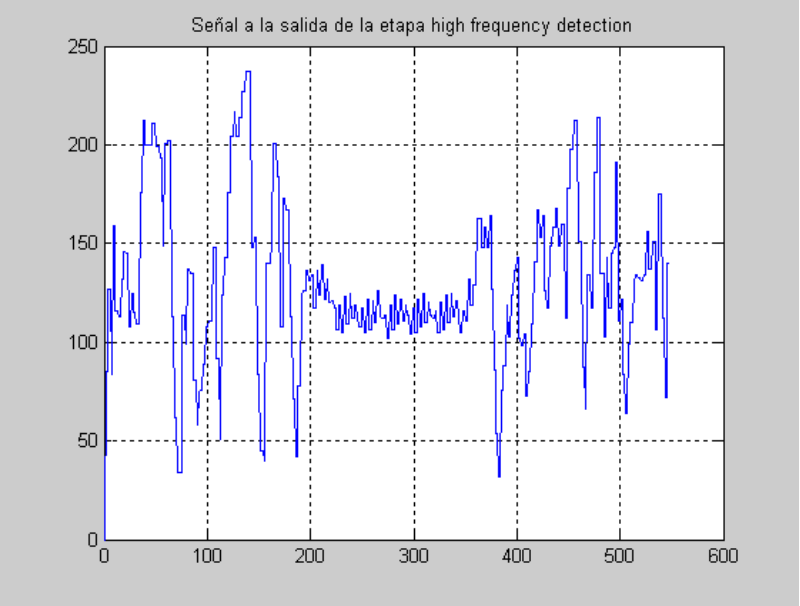


Figura 11. Parte fraccionaria del periodo estimado para el tono A.

Si observamos la figura 11 los valores que adquiere en el rango de muestras de 200 a 350 muestras notamos que la parte fraccionaria varia de 100 a 125 y se determina que la parte entera y fraccionaria si están dentro de los márgenes permitidos para la estimación de la frecuencia de 1633 Hz.

Los datos que representa la parte entera y fraccionaria para la frecuencia estimada (1633 Hz) entran al bloque de chequeo de alta frecuencia, como su nombre lo indica, en este etapa se analiza si la frecuencia estimada esta dentro de los márgenes permitidos (3.5 % de desviación permitido) y si está dentro de los márgenes se asigna la variable Column (columna) con uno de los datos a continuación:

Column = 4 ; si la frecuencia cae dentro de 1633 ± 3.5 %.

Column = 3 ; si la frecuencia cae dentro de 1477 ± 3.5 %.

Column = 2 ; si la frecuencia cae dentro de 1336 ± 3.5 %.

Column = 1 ; si la frecuencia cae dentro de 1209 ± 3.5 %.

Posteriormente dicha variable es enviada a la etapa de "Chequeo de baja frecuencia", y a la vez esta etapa (Chequeo de alta frecuencia) nos hace un ajuste de control por medio de la variable cos_hfreq_mu1 lo cual va a controlar al filtro adaptativo notch localizado en la ruta de baja frecuencia (le va ordenar a dicho filtro a que frecuencia va a poner la ranura), que en este caso sería a la frecuencia de 1633 Hz .

El estimador de frecuencia en la ruta de baja frecuencia tiene el mismo diagrama en bloque que el de la ruta de alta frecuencia con la diferencia en que el sub-bloque de Chequeo de baja frecuencia recibe la parte fraccionaria y la parte entera del periodo estimado y analiza si dicha frecuencia está en los márgenes permitidos , si dicha señal está en los márgenes permitidos , este asigna a la variable row (fila) con unos de los datos a continuación :

Row = 12 ; si la frecuencia cae dentro de 941 ± 3.5 %.

Row = 8 ; si la frecuencia cae dentro de 852 ± 3.5 %.

Row = 4 ; si la frecuencia cae dentro de 770 ± 3.5 %.

Row = 0 ; si la frecuencia cae dentro de 697 ± 3.5 %.

A la vez en esta etapa se recibe la variable column y una de las salidas de esta etapa es la variable NS (Nuevo símbolo) y dicha salida es el resultado de sumar la variable row más la variable column; la variable NS va a ser un equivalente de cada tecla del teclado que a continuación se muestra en el cuadro 2.

NS	TONO DTMF
1	1
2	2
3	3
4	A
5	4
6	5
7	6
8	B
9	7
10	8
11	9
12	C
13	*
14	0
15	#
16	D

Tabla 2. Representacion de los tonos.

Ahora la variable NS va a ser analizada por la lógica de decisión.

ESTIMADOR DE POTENCIA

El estimador de potencia analiza la potencia de la señal DTMF , dado que la señal DTMF ocurre en forma de ráfaga (burst), la potencia de la señal puede ser seguida como un término promedio de la potencia. El estimador de potencia usa un filtro IIR de primer orden de la forma:

$$P(n) = \beta P(n-1) + (1 - \beta) |s(n)|.$$

Donde $P(n)$ es la potencia actual y $s(n)$ es la muestra de la señal actual. El factor β es cercano a 1, para que el estimador de potencia sea rápido al analizar la señal, seleccionamos $\beta = 0.95$.

La potencia de la señal es comparada con el ruido para detectar la presencia de la componente de señal dentro del ruido, tal como se muestra en la figura 12.

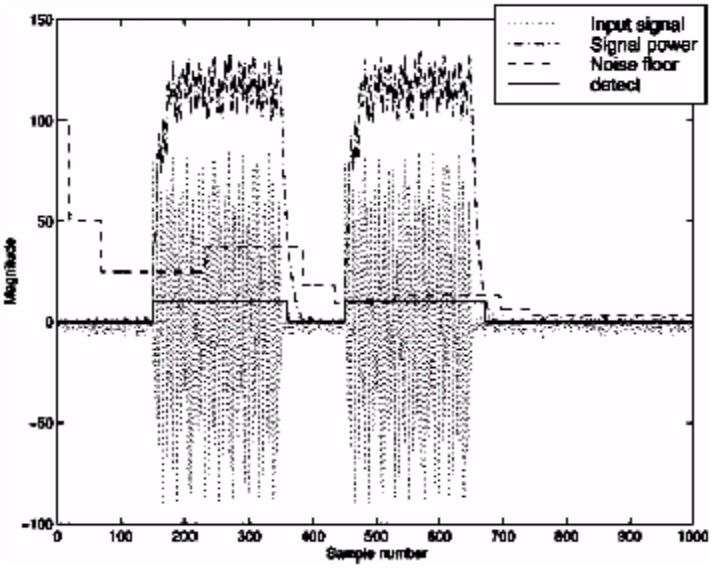


Figura 12. Estimador de potencia .

Siempre y cuando la potencia de la señal es menor que el ruido, el ruido es decrementado en forma exponencial con respecto al tiempo. Sin embargo, el ruido es incrementado exponencialmente siempre y cuando la potencia de la señal es mayor que el ruido. La señal detect (DT) es 1 lógico si la potencia de la señal es mayor que el ruido y es cero si la potencia de la señal es menor al ruido; el estimador de potencia detecta de una forma robusta la presencia de la componente de señal dentro de una señal con ruido y el detector tiene un excelente desempeño al analizar el talk-off.

LOGICA DE DECISION.

En esta etapa es la que se analiza los estatutos de la ITU Q.24 con respecto al tiempo, tal como se observa en la figura 13 , en el flujograma observamos las variables LF (estimación de baja frecuencia) y HF (estimación de alta frecuencia) las cuales son procesadas en la etapa de chequeo de baja frecuencia y como resultado final tenemos la variable NS (símbolo nuevo), también el detector recibe la información del estimador de potencia y lo hace por medio de la variable DT.

En esta etapa también se le aplica un diezmado por 2 y las muestras ya no van a estar presentes por 250 useg sino que van a estar presentes por una duración de 500 useg. (0.5 ms) en esta etapa, es decir que la lógica de decisión funcionará a 2 Khz.

Para analizar los requerimientos con respecto al tiempo usamos tres contadores .

El contador TC es usado para determinar si la última interrupción del tono fue por más de 10 ms., este es inicializado a 20 muestras (10 ms. a 2 KHz).

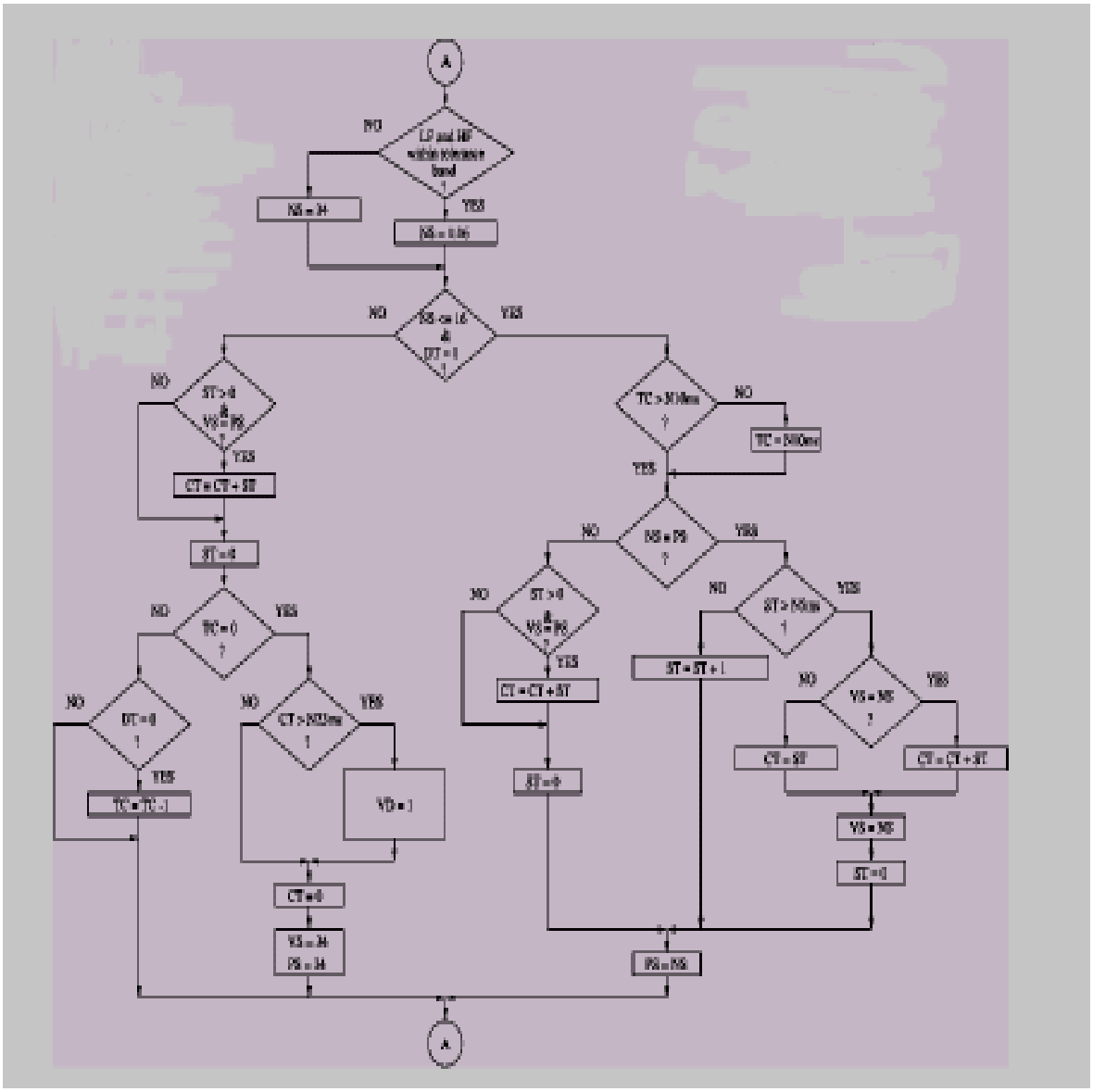


Figura 13 Flujograma de la Lógica de Decisión.

El contador de señal (ST) es un contador ascendente cíclico de 0 a 10 muestras (5 ms a 2 KHz) ; éste es incrementado en uno siempre y cuando los dos símbolos DTMF detectados son el mismo y se resetea si no lo son.

El contador de la duración de la señal (CT) es el indicador del número total de muestras

para lo cual el símbolo DTMF está presente. La estimación de tiempo tiene una precisión de 0.5 ms a una tasa de muestreo de 2 KHz.

La etapa de chequeo de baja frecuencia asigna a NS en el rango de 1 hasta 16 si cumple con los requerimientos de decodificación con respecto a la frecuencia, si no cumple con esto se le asigna a NS con un valor de 34 .

Si NS está en el intervalo de 1 a 16 y DT es igual a 1, entonces el contador TC es reinicializado, y el símbolo previo (PS) es comparado con NS. Si PS no es igual a NS, entonces el contador de duración de la señal CT es incrementado por ST, y ST es reseteado a cero. Si PS es igual a NS y ST es menor que 10 muestras, entonces CT es actualizado y ST es puesto a cero. La primera vez que ST es igual a 10 muestras significa que la frecuencia estimada para NS se considera una frecuencia estabilizada.

Cuando una pausa después de la ráfaga DTMF exceda de 20 muestras (10 ms a 2KHz), la lógica de decisión examina CT contra 46 muestras (23 ms a 2 KHz) , Una detección válida es señalizada si el símbolo DTMF representado por NS cumple con los requisitos de la ITU con respecto a la decodificación, antes de salir , la lógica de decisión asigna NS a PS.

SIMULACION DEL DECODIFICADOR DTMF.

Para realizar la simulación del decodificador en MATLAB, se procedió a crear una función en MATLAB que nos genere tonos DTMF puro o contaminado por ruido, dicha función se llama dtmfgen cuya sintaxis como ejemplo es:

a = dtmfgen('A ', 0.04, 0.05, 1/8000, 1, 15);

A la función le estamos indicando que nos genere la señal DTMF correspondiente a la tecla A con una duración de tono de 40 ms, y con un tiempo de pausa de 50 ms, y dicho tono va a tener una componente de ruido con SNR de 15 dB.

La función que nos va a hacer la decodificación del tono A se llama : dtmfdetector(a,256) ; donde el valor de 256 nos indica el valor de la amplitud que va a tener dentro del procesado.

A continuación se presentan ciertas gráficas en las cuales se muestra el procesado de la señal en cada etapa del decodificador.

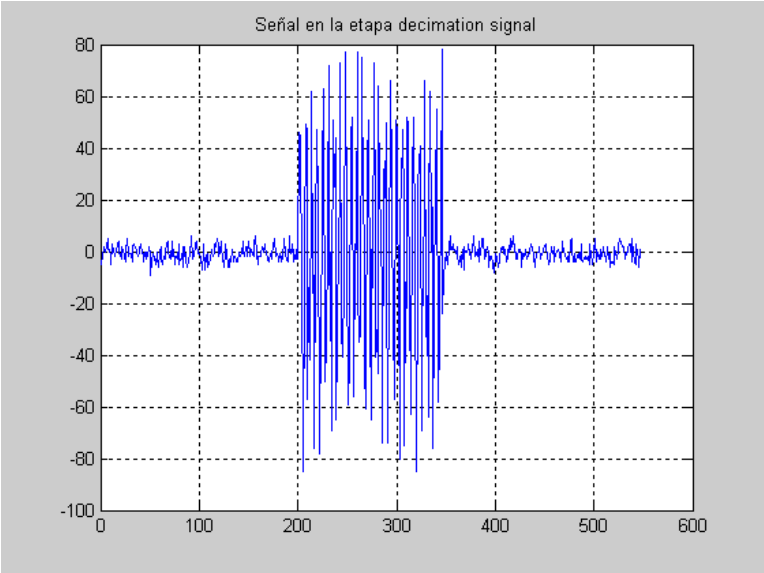


Figura. 14 . Señal presente en la etapa del diezmado por 2 (tecla A).

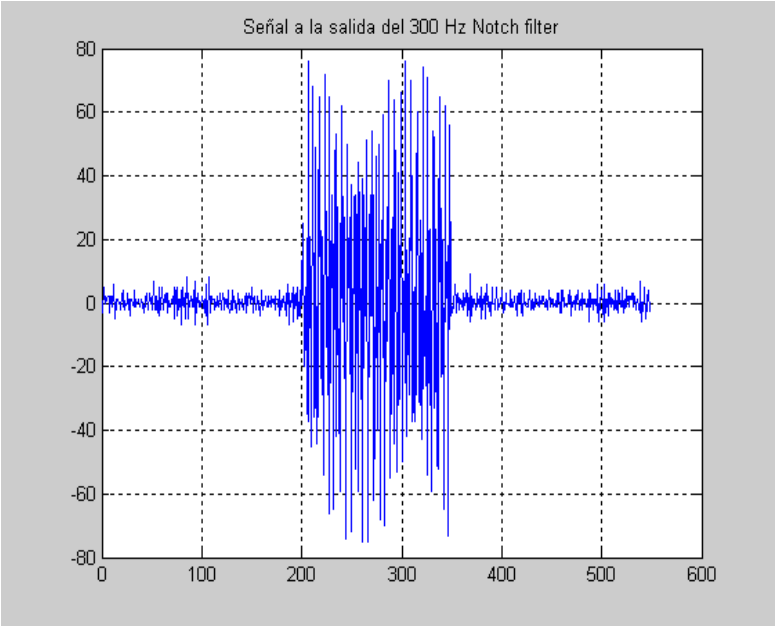


Figura 15. Señal presente a la salida del filtro notch 300 Hz.

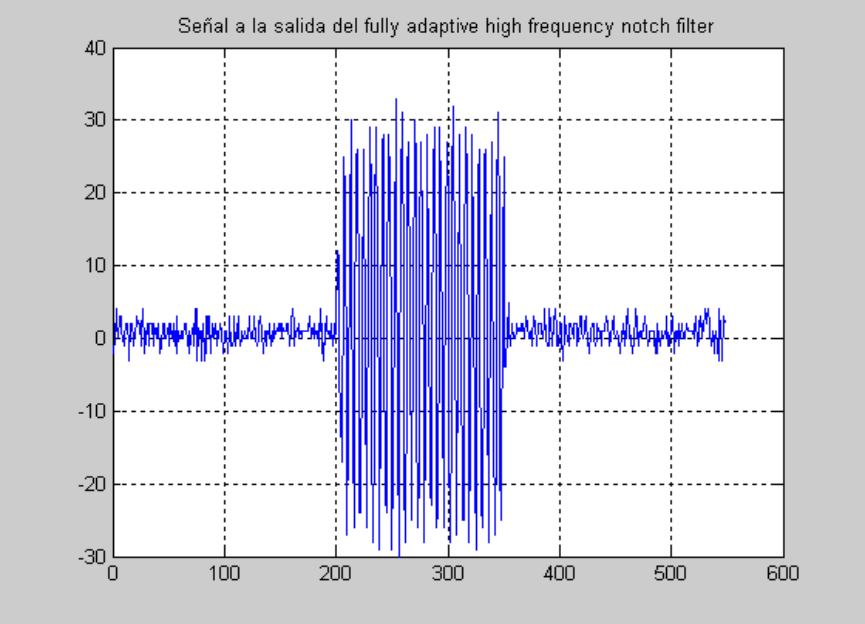


Figura 16. Señal a la salida del filtro notch adaptativo rama alta

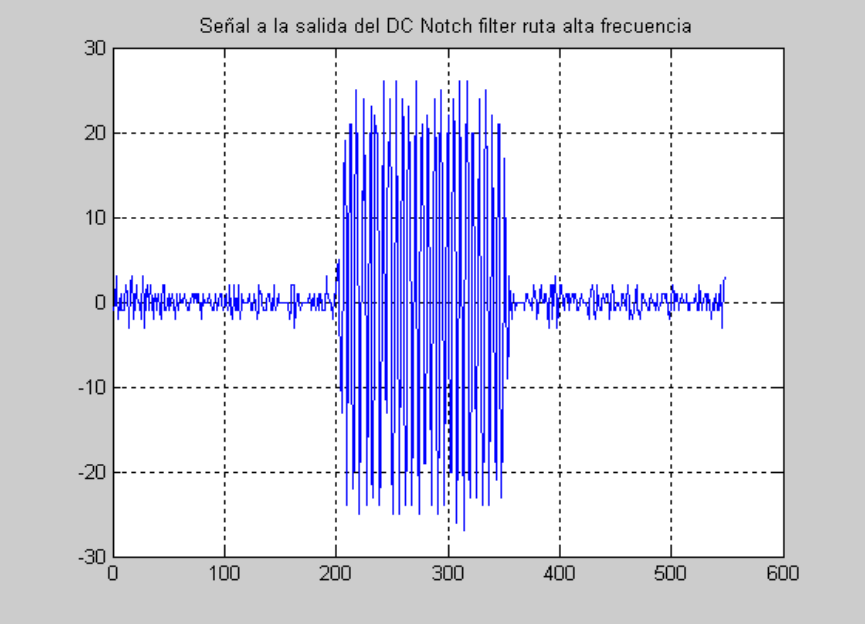


Figura 17. Señal a la salida del DC Notch filter rama alta frecuencia.

Figura 18. Señal en el estimador de frecuencia (parte fraccionaria).

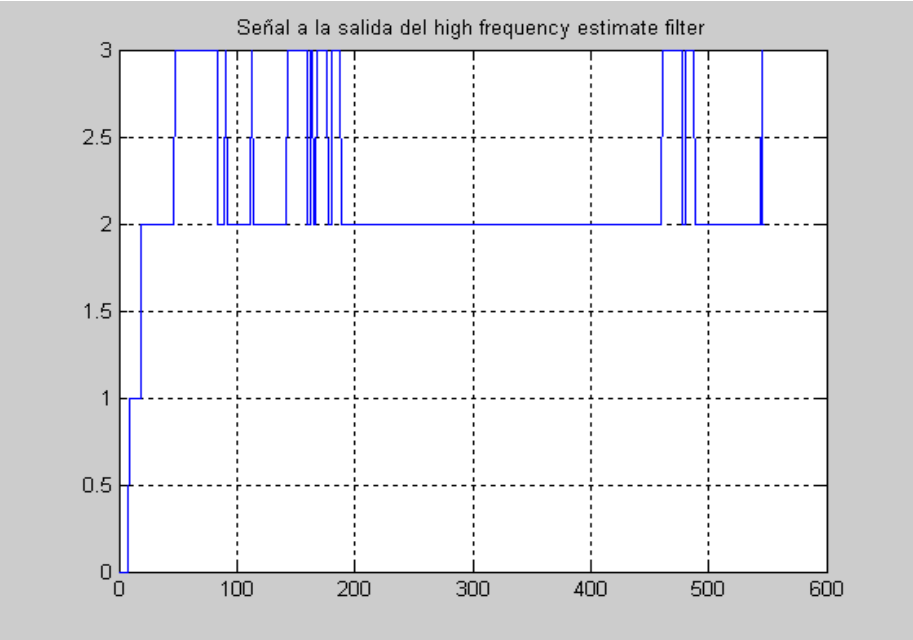


Figura 19.Señal en el estimador de frecuencia .

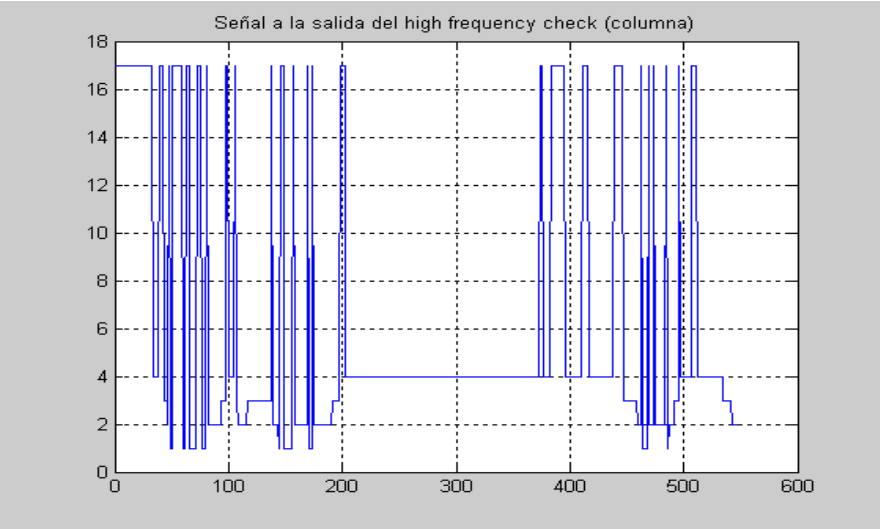


Figura 20. Señal en el estimador de frecuencia (columna)

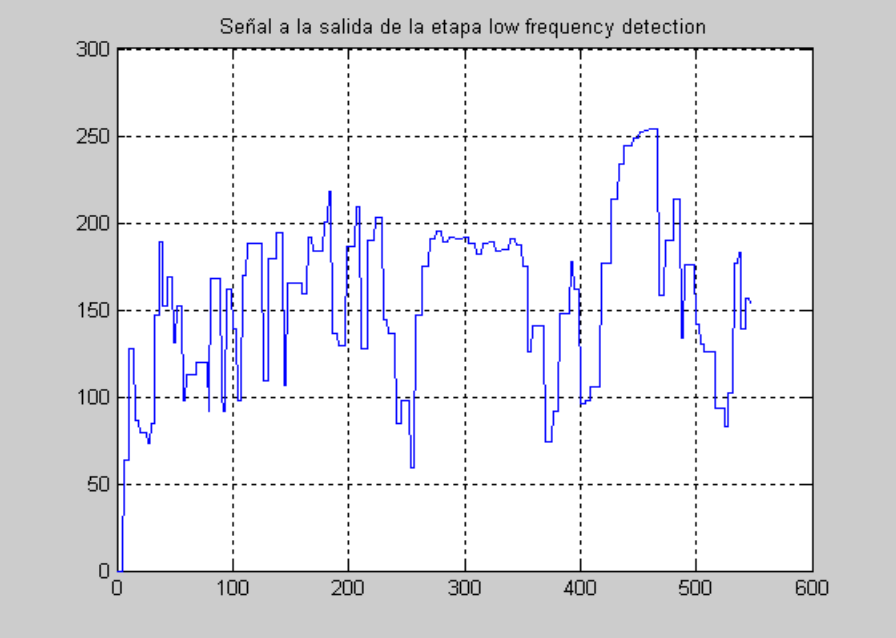


Figura 21.. Señal en el estimador de frecuencia (parte fraccionaria). rama de baja frecuencia

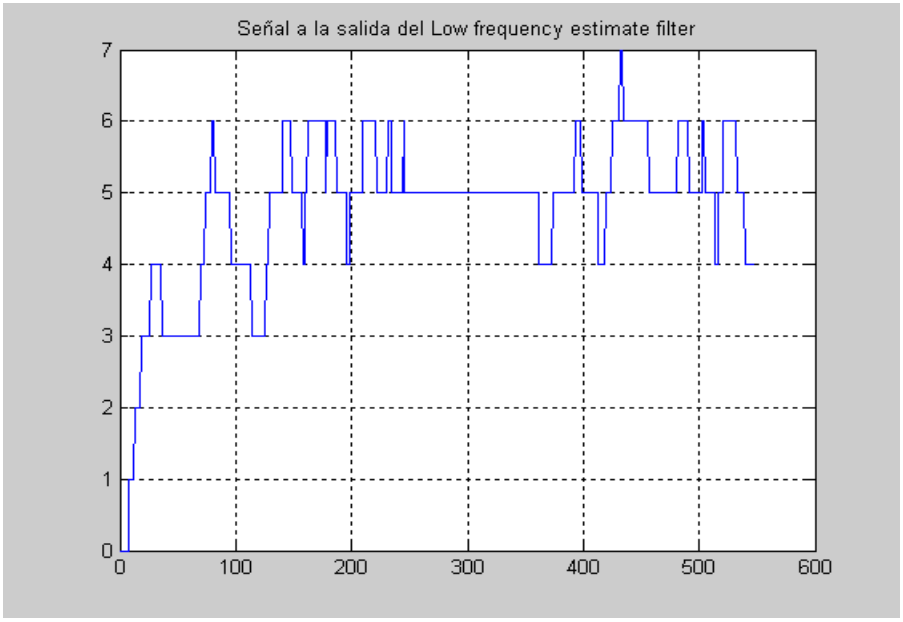


Figura 22.Señal en el estimador de frecuencia (parte entera), rama baja

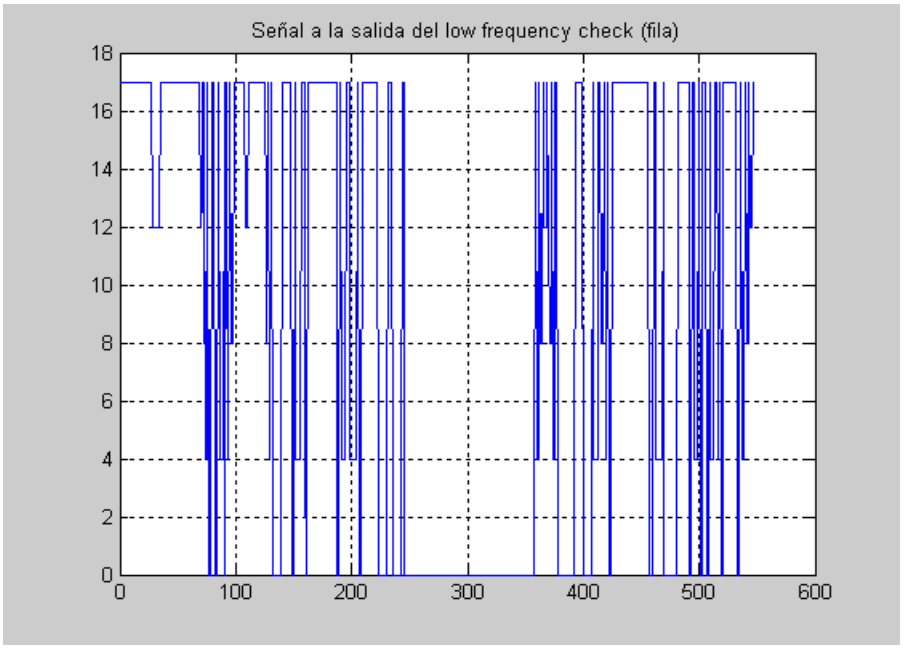


Figura 23. Señal en el estimador de frecuencia (fila) rama baja.

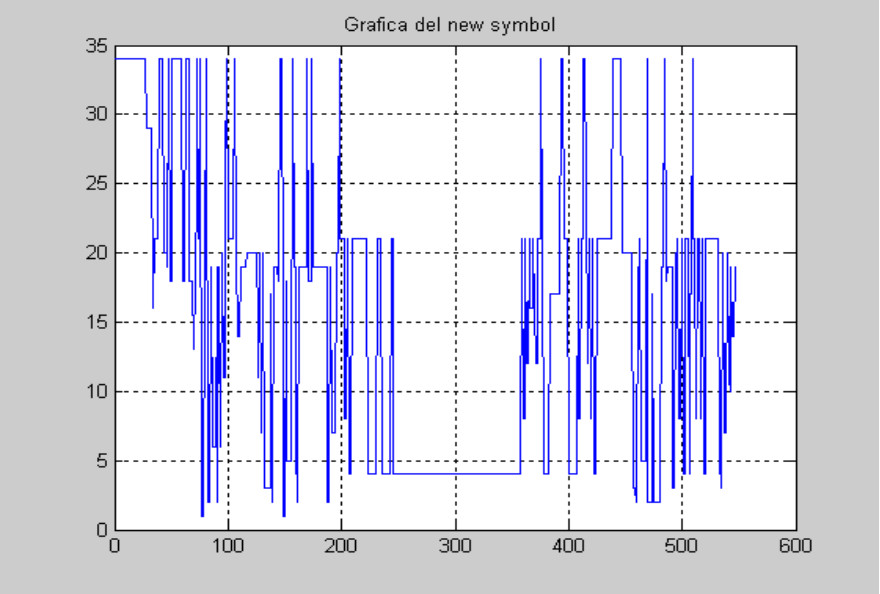


Figura 24. Señal de nuevo simbolo (valor de 4).

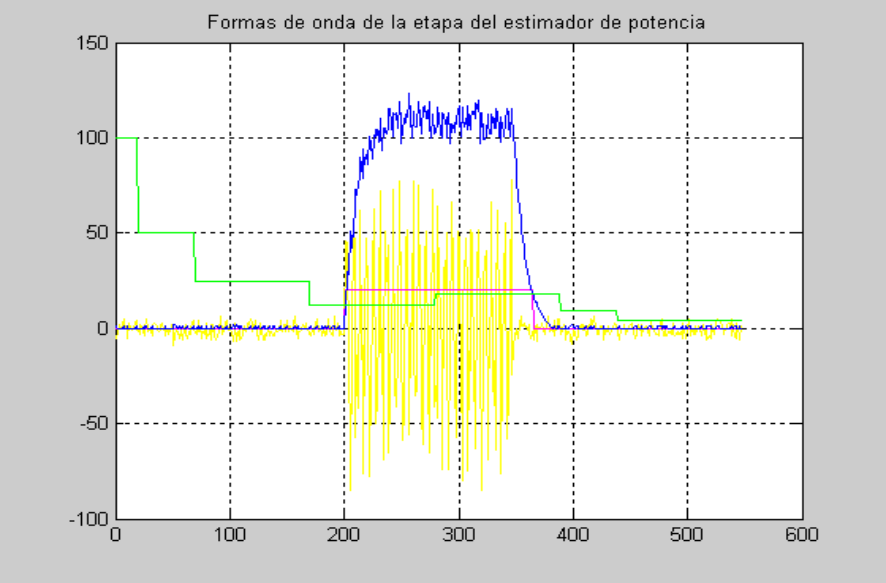


Figura 25 . Señales en la etapa del estimador de potencia.

verde: ruido ; lila : señal DT

amarillo : señal DTMF ; azul: potencia de la señal.

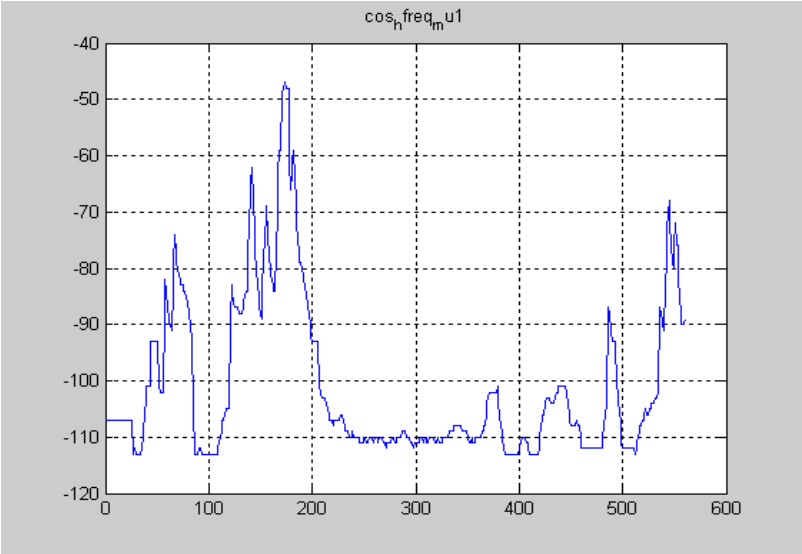


Figura 26. Señal de control de estimador de baja frecuencia .

Como vemos el valor que toma esa señal de control es de -110 (intervalo de muestra de 250 a 350), usando matlab para ver la respuesta en frecuencia del filtro notch adaptativo con esa nueva orden tenemos:

A=1 ; B= [0.5 110/128   0.5] ; freqz (B , A , 512, 4000) ;

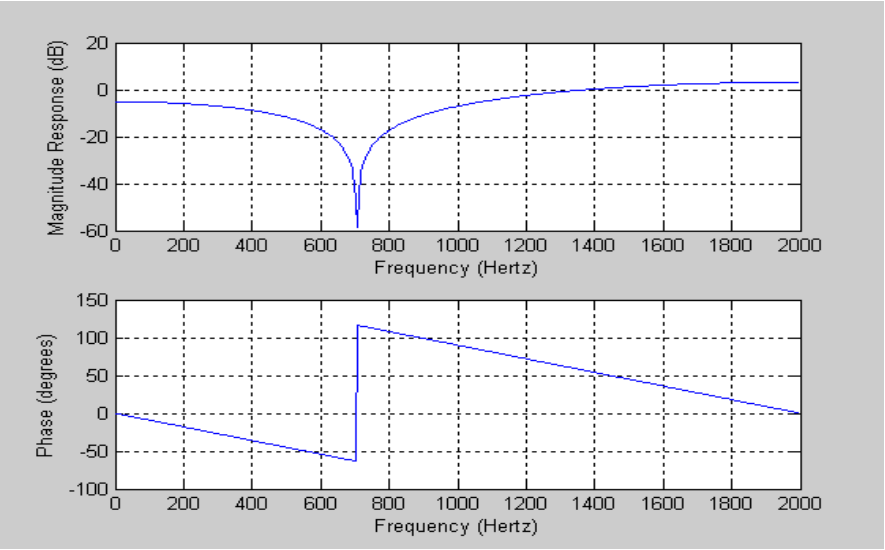


Figura 27. Respuesta del filtro notch adaptativo rama alta.

Como observamos en la figura 28, la señal de control proveniente del estimador de baja frecuencia le ordena al filtro notch adaptativo de alta frecuencia que coloque una ranura a la frecuencia de 697 Hz., el filtro atenuará las frecuencias muy cercanas a 697 Hz.

Lo mismo sucede para el estimador de alta frecuencia según las figuras 29 y 30.

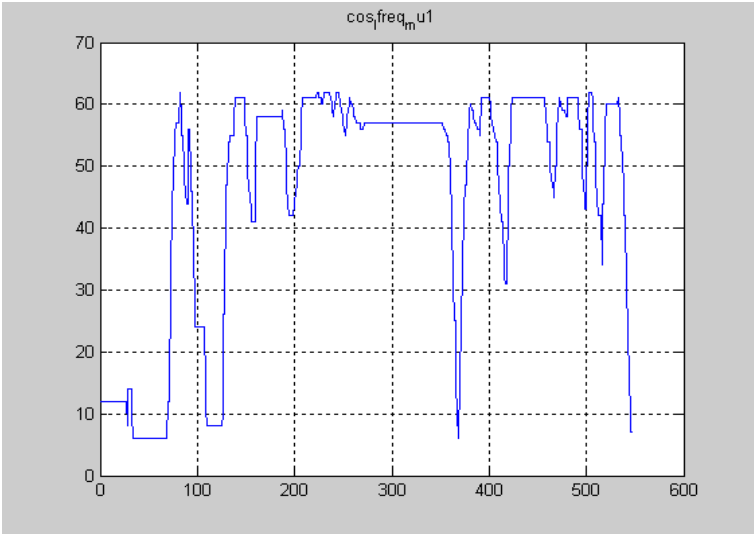


Figura 28.Señal de control de estimador de alta frecuencia.

Como vemos el valor que toma esa señal de control es de 57 (intervalo de muestra de 250 a 350), usando matlab para ver la respuesta en frecuencia del filtro notch adaptativo con esa nueva orden tenemos:

a = 1 ; b = [0.5 -57/128 0.5] ; freqz (b, a, 512, 4000) ;

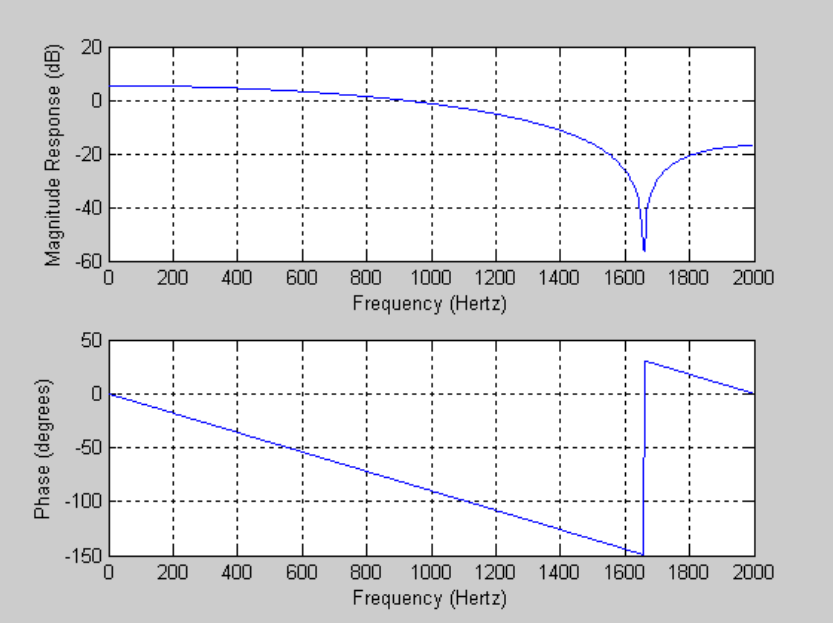


Figura 29. Respuesta del filtro notch adaptativo

Como observamos en la figura 30, la señal de control proveniente del estimador de alta frecuencia le ordena al filtro notch adaptativo de baja frecuencia que coloque una ranura a la frecuencia de 1633 Hz., el filtro atenuará las frecuencias muy cercanas a 1633 Hz.

CAPITULO III

IMPLEMENTACION DEL DECODIFICADOR EN UNA TARJETA EMBEBIDA USB.

En este capítulo se trata la forma de cómo es implementada el decodificador DTMF en una tarjeta de desarrollo conformada por un microcontrolador y de un adaptador USB.
Empezamos con una sección en la cual nos explica que es un sistema embebido, la arquitectura del microcontrolador y una descripción de cómo esta conformada la tarjeta de desarrollo en la cual se implementa el decodificador.
En la siguiente sección explicamos el uso del MPLAB y del compilador CCS para la programación de la tarjeta.
Posteriormente en la otra sección se explica como esta conformado el programa en lenguaje C y de cómo implementamos el filtro notch de 300 Hz y el filtro adaptativo.
En la última sección hablaremos del programa DLP test y la forma en que lo utilizamos.

SISTEMAS EMBEBIDOS.

Son sistemas diseñados para desarrollar una tarea especifica; en las aplicaciones embebidas, el hardware es único para el sistema embebido, y el software específicamente ha sido escrito para hacer un uso óptimo de los recursos disponibles del sistema; ejemplos de sistema embebidos los encontramos en los CD players, teléfonos celulares, lectores de tarjetas, modems, cámaras digitales, calculadoras, sintetizadores musicales, etc.
Una parte primordial de los sistemas embebidos son los microcontroladores los cuales se les puede programar para realizar diferentes tareas tales como administración de recursos, procesado de datos y control de periféricos.

MICROCONTROLADORES.

Un controlador es un dispositivo que es usado para controlar algún proceso o aspecto de un entorno. Un microcontrolador es un chip programable altamente integrado lo cual lo hace ideal para aplicaciones de control, ellos son usados en sistemas embebidos y contienen varios periféricos tales como timers, convertidores A/D , drivers LCD, etc.
Hoy en día los microcontroladores vienen en varias obleas de silicón y tienen algunas o todas de las características siguientes:

1- Arquitectura: Hay dos tipos de arquitectura que son Princeton y Harvard.
   i- Arquitectura Priceton : usa un bus de datos para accesar a las instrucciones y datos. Además es estructurado para una ejecución secuencial y almacena a ambos instrucciones y operandos en la misma memoria , esto potencialmente hace lento la ejecución de los programas; ejemplo de esta arquitectura lo encontramos en los procesadores motorola

MC68HC11 y en el 8051 de Intel.

ii- Arquitectura Harvard: es orientada hacia la arquitectura paralela; este tiene buses de dato separado para las instrucciones y los operandos. Además este tiene memoria solo para datos y memoria para el programa. Este tipo de arquitectura acelera la ejecución de los programas pero requiere de más silicón. Los microcontroladores PIC de Microchip usa este tipo de arquitectura.

2- Conjunto de Instrucciones: El conjunto de instrucciones puede ser clasificado como CISC y RISC.

i- CISC (Complex Instruction Set Computers) : es la organización mas usada por los microcontroladores , tienen un número grande de instrucciones para realizar una tarea específica de control y tiene modos de direccionamiento muy variado. Una ventaja de la organización CISC es que muchas instrucciones son usadas en macros, permitiendo que una instrucción sea usada para varias tareas, una ventaja clave es el empaquetado de bits dentro de los opcodes para reducir la longitud de la palabra de instrucción lo cual reduce la longitud del programa en bytes y lleva una reducción en el consumo de energía.

ii- RISC (Reduced Instruction Set Computers) : Un procesador con esta organización de instrucciones provee pocas instrucciones , las operaciones sin embargo son de registro a registro, el tamaño de las instrucciones son uniformes y es apropiado para aplicaciones tipo pipelining. La ventaja de este tipo de organización es que éste reduce el tamaño del chip y el número de terminales del chip, RISC es ideal para la tecnología

de compilador y para aplicaciones que usa frecuencia alta de reloj.

3- Memoria: Casi todos los microcontroladores tiene una memoria on-chip(una combinación del programa y de datos), los tipos de memoria pueden ser: EEPROM, EPROM, FLASH PROM y OTP (one time programmable).
4- Administración de energía : los microcontroladores son procesadores de baja potencia debido a que su reloj son bajos y la longitud de palabra de instrucción es corta. Algunos microcontroladores tiene una protección contra los UVL (under voltaje level) lo cual se resetea el dispositivo cuando el voltaje de operación (Vcc) es bajo con respecto a un voltaje de umbral. Opcionalmente algunos de ellos tienen un modo SLEEP/WAKE UP, este modo es útil en la reducción del consumo de energía cuando el microcontrolador está ocioso.
5- Interface I/O : Todos los microcontroladores tienen unos o más puertos bidireccionales para el interface con periféricos. Ocasionalmente ellos podrían tener un UART (Universal Asynchronous Receiver Transmitter) o USART (Universal Synchronous Receiver Transmitter .
6- Interface Análoga : Algunas versiones más caras de microcontroladores tienen convertidores A/D y convertidores D/A , esto reduce grandemente el tiempo de acceso en el periférico. Algunos microcontroladores tienen un modulador PWM para la generación de formas de ondas.
7- Timers : Casi todos los microcontroladores por lo menos tiene un timer y un watchdog timer, el watchdog timer es un timer programable, lo cual cuando es activado , resetea el microcontrolador después de un intervalo específico.
8- Características especiales : una característica especial es el manejo de interrupciones , protección de código por medio de bits e instrucción de manipulación de bits.

TARJETA DE DESARROLLO DLP 245 PL-G.

Es una tarjeta de desarrollo para prototipo de productos en las cuales se involucran los microcontroladores, dicha tarjeta está conformada por el microcontrolador 18LF8722 y de un chip que se encarga de la transferencia de datos usando el protocolo USB.

Figura 30. Tarjeta en la cual se implementa el decodificador DTMF

El oscilador con cristal de cuarzo del microcontrolador trabaja a 6 Mhz, pero activando la etapa interna del PLL llevamos la frecuencia de reloj a 24Mhz.

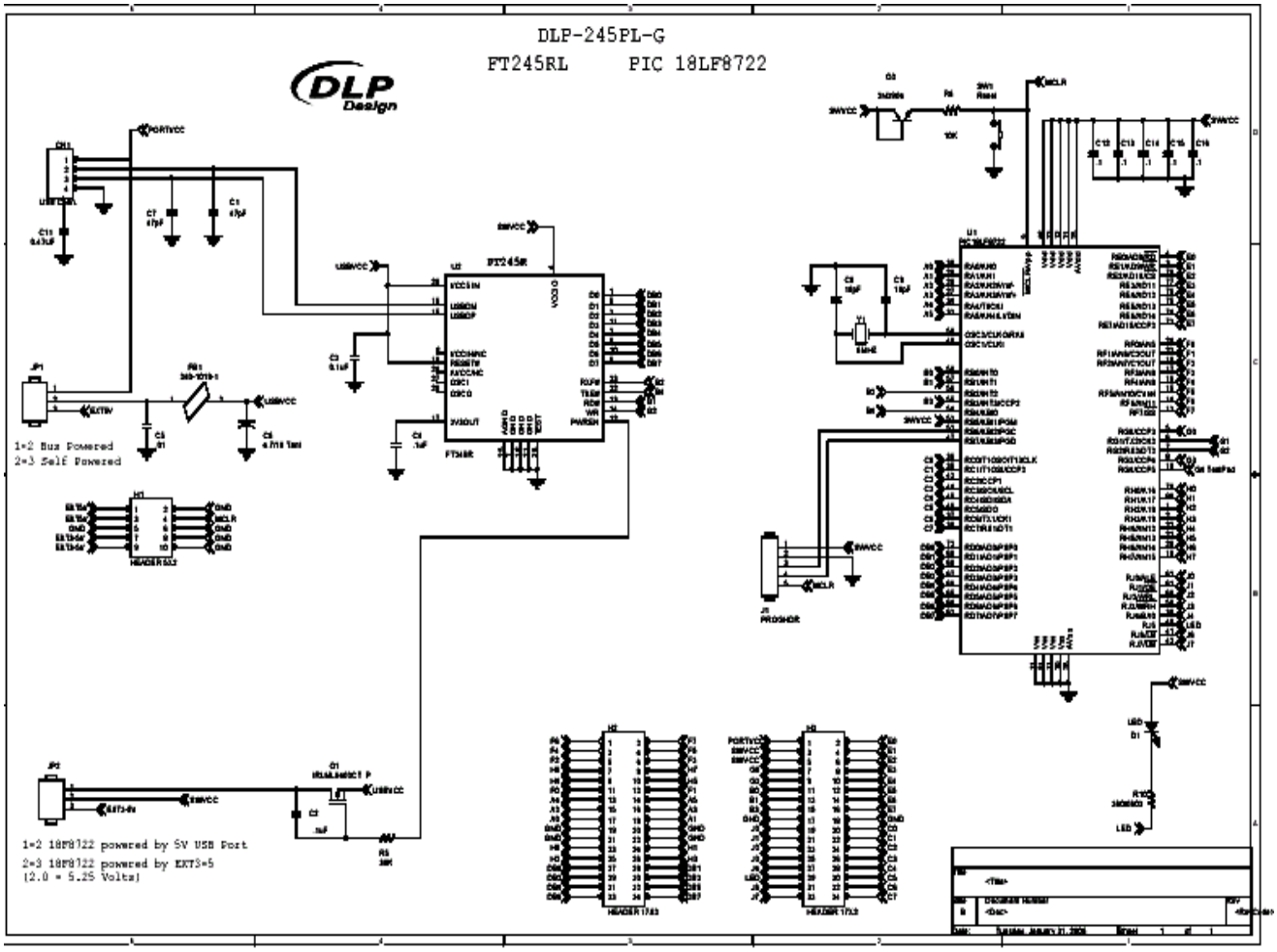


Figura 31 Diagrama eléctrico de la tarjeta de desarrollo.

Para enviar datos desde el periférico hacia la computadora, el microcontrolador simplemente escribe el dato de tamaño de 1 byte hacia el chip FT245RL cuando la linea TXE# es bajo. Si el bufer transmisor del FT245RL está llenándose o está ocupado almacenando el byte previamente escrito, este pondría a la línea TXE# en alto para así detener que mas datos sean escritos y a la vez transferir los datos sobre el host de la PC

por medio del protocolo USB.
Cuando el host envía datos hacia el periférico por medio del USB, el chip FT245RL hará que la línea RXF# sea bajo para permitir que el microcontrolador conozca que el byte de dato está disponible. El microcontrolador entonces lee el dato hasta que la línea RXF# se ponga en alto indicando que no hay más datos disponibles para leer.
La tarjeta tiene disponible un conjunto de orificios dedicados en el cual se puede colocar un conector de 6 pines de tipo RJ para conectar el ICD2 de Microchip para la programación y depuración del microcontrolador y también la tarjeta se puede configurar para que sea alimentada por el cable USB (media vez el consumo de corriente sea menor a 500 mA); o sino por una fuente de voltaje externa.

MPLAB y CCS.

El CCS es un compilador en la cual el código del programa escrito en lenguaje C para un microcontrolador específico (seleccionado de una base de datos solo de microcontroladores) lo convierte en lenguaje ensamblador compuesta de instrucciones RISC propias del microcontrolador.
La ventaja de este compilador es la reducción de líneas de código escrita en C en comparación de la escritura de código en ensamblador del microcontrolador específico.
El MPLAB es un entorno de desarrollo para los microcontroladores de Microchip.
El MPLAB trabaja a base de proyectos en las cuales posee un asistente de proyectos en las cuales nos pregunta que microcontrolador vamos a usar, como se va a llamar el proyecto, el lenguaje de programación que usaremos y la selección de los archivos que se van a utilizar para la generación del código a programar en el microcontrolador.
En el caso de la tarjeta, escribimos el programa del decodificador en lenguaje C en los cuales el código fuente invoca dos archivos también escrito en C, dichos archivos son prototypedtmf.h en el cual se encuentra las definiciones de constantes y variables; el archivo funciones.c el cual se definen las funciones que utiliza el archivo tipo fuente del decodificador.

Creamos un proyecto en MPLAB usando el asistente del proyecto y especificamos el microcontrolador a usar y en la parte de opción del lenguaje a utilizar seleccionamos CCS Compiler y posteriormente nos pide los archivos que vamos a insertar en el proyecto.
Al momento de compilar, el MPLAB invoca (por medio de un plug-in) a CCS y el CCS empieza a compilar, y si la compilación es exitosa el CSS nos genera una serie de archivos con diversas extensiones tales como .HEX, .ERR, .SYM. , COF , PJT , .TRE., .STA.
Ya tenemos el archivo .HEX que vamos a programar en el microcontrolador, posteriormente en el MPLAB hay un menú en el cual podemos seleccionar que programador vamos a usar, es este caso usamos el ICD2 como programador de la tarjeta y posteriormente programamos la tarjeta.

IMPLEMENTACION DEL FILTRO NOTCH EN LENGUAJE C.

Usando la función de transferencia del filtro notch:

$$H(z) = b_0(1 - 2\cos\omega_0 z^{-1} + z^{-2})$$

El factor $2\cos\omega_0$ nos dictará a que frecuencia vamos a colocar la ranura.

Para una ranura de 300 Hz el valor de $\cos\omega_0$ es de 0.89 y para expresar este dato a nivel de 8 bits lo multiplicamos por 128 y nos da el valor aproximado de 114, este valor lo guardamos en una variable denominada notch300_mu1.
El primer término de la función de transferencia (0.5) va a multiplicar a la muestra presente x[n], el segundo término de la función de transferencia va a multiplicar a la muestra anterior x[n-1] y el último término va a multiplicar a la muestra x[n-2].
A continuación presentamos un extracto del programa en C que implementa el filtro notch de 300 Hz.

```
Sea x[n] = decimated_signal[0] ;  x[n-1] = decimated_signal[1],
      x[n-2] =decimated_signal[2];

    a=mymul(notch300_mu1,decimated_signal[1]);
```

```
a=2*a;
d=mysub(div2(decimated_signal[0] ), a);
e = myplus (d, div2( decimated_signal [2] ) );
notch300[ 0 ] = e;
```

el dato resultante del filtro notch 300 lo guardamos en la variable notch300[0].
Las funciones mymul, myplus, mysub, div2 opera con datos enteros con signo (rango de –127 a 128).
En el CD se encuentra el programa del decodificador y los archivos auxiliares.

PROGRAMA DLPTEST.

Este programa lo ocupamos para monitorear la transmisión y recepción de datos de la PC hacia la tarjeta ; tiene la modalidad de enviar por medio de un archivo las muestras digitalizadas de los tonos DTMF, la desventaja de este programa es que los bytes de datos a enviar deben de llevar la terminación d(de notación decimal) o h (si es notación hexadecimal) para el envío de los datos.

Figura 32. Vista en pantalla del programa DLPTEST.

CAPITULO IV

RESULTADOS OBTENIDOS

El presente capítulo muestra los resultados obtenidos al implementar el decodificador DTMF en la tarjeta.

Previo a los resultados finales se hace una descripción de la forma en que se realizaron dichas pruebas.

Por último se da unas propuestas de mejora al decodificador DTMF.

DESCRIPCION DE LAS PRUEBAS.

La forma en que se desarrolla es la siguiente:

i-Se ha creado la función dtmfgen de MATLAB para generar el tono DTMF mezclado con ruido

ii- El tono generado es representado con números de notación decimal y lo convertimos a una notación hexadecimal.

iii- Los datos en notación hexadecimal hay que agregarle la letra h para sea entendida como un dato hexadecimal y posteriormente es almacenada en un archivo de formato .dat; la razón de agregarle la letra h es por la sintaxis que usa el programa DLPTEST para la transmisión de datos hacia la tarjeta.

iv- Usando el software DLPTEST abrimos el archivo .dat y lo enviamos a la tarjeta por medio del USB.

v- El decodificador procesa los datos y después nos envía el valor correspondiente al tono.

Con respecto a los tonos se le ha dado un tiempo de separación de tono de 50 mseg.; la duración de los tonos la hemos ido variando de 40 a 36 mseg en pasos de 1 mseg ; con respecto a la relación señal ruido se ha ido variando el SNR de 40 a 15 en pasos de 5 con resultados de decodificación correctos.

Para examinar la respuesta del decodificador con respecto a la frecuencia se ha modificado el archivo generador de matlab con respecto a las frecuencias nominales variándola a los limites de 3.5 % del valor de la frecuencia nominal y posteriormente se hizo el proceso antes mencionado para enviar los datos a la tarjeta , con resultados correctos en el proceso de la decodificación del tono, por último variamos las frecuencias

nominales hacia una desviación del 4% y se tuvo como resultado tanto en la simulación de matlab como en la tarjeta de una decodificación errónea.

Posteriormente se realizó la prueba de enviarle a la tarjeta datos correspondiente a un tono contaminado por voz obteniendo resultados satisfactorios.

TABLA DE RESULTADOS

A continuación presentamos una tabla que nos resume las diferentes pruebas realizadas.

Simbolo DTMF	Pausa mseg.	Tiempo On mseg.	SNR	Desviación Frec nominal %	Resultado
1	50	36	15	0	OK
2	50	37	30	0	OK
3	50	38	30	0	OK
A	50	39	30	0	OK
4	50	40	30	3.5	OK
5	50	40	20	3.5	OK
6	50	38	25	3.5	OK
B	50	36	15	3.5	OK
7	50	39	40	3.5	OK
8	50	40	15	3.5	OK
9	50	37	30	3.5	OK
C	50	37	18	3.5	OK
*	50	40	30	3.5	OK
0	50	36	15	3.5	OK
#	50	38	35	3.5	OK
D	50	40	30	3.5	OK
1	50	40	25	4	NO OK

TABLA 3. RESUMEN DE LAS PRUEBAS REALIZADAS AL DECODER DTMF.

PROPUESTA DE MEJORA.

Se propone el diseño de una aplicación de software (interfaz gráfica con generación de reporte) para la transmisión y recepción de datos entre la PC y la tarjeta y también del diseño de un circuito acoplador (acomodador) de señal para así enviarle una señal DTMF análoga a la entrada del circuito y que a la salida de éste sea conectado a un convertidor análogo digital interno del microcontrolador de la tarjeta para el posterior proceso de decodificación.

CONCLUSIONES

1- El decodificador DTMF implementado en la tarjeta DLP 245 PL G si cumple con los requerimientos de la ITU Q. 24 con respecto al tiempo, frecuencia, SNR y talk-off..
2- El comportamiento del decodificador implementado en la tarjeta si cumple con las expectativas que nos mostraba el software MATLAB con respecto a la simulación.
3- Se demuestra el comportamiento de los filtros notch adaptativos tanto en la simulación en MATLAB como en la implementación en la tarjeta.
4- Se verifica el correcto funcionamiento de los estimadores de frecuencia en el análisis de los requerimientos de frecuencia del decodificador en sí.
5- Se verifica en la etapa de la lógica de decisión los requerimientos de decodificación con respecto al tiempo.
6- Se verifica la robustez del decodificador ante señales DTMF contaminadas por diversos niveles SNR de ruido y de voz.
7- Se demuestra el uso de diversos softwares (MPLAB y CCS) para la programación del microcontrolador de la tarjeta.

ANEXOS

TABLA DE TIEMPOS Y DE FRECUENCIAS USADAS EN EL ESTIMADOR DE FRECUENCIA

GRUPO BAJO	-3.5% Fdtmf	Fdtmf	3.5% Fdtmf
Freq.	**673**	**697**	**721**
Tiempo	1,486E-006	1,435E-006	1,387E-006
Tiempo/2 Ts	5.94	5.74	5.55
Period_integer	5	5	5
Period_fraction	0.94	0.74	0.55
Period_fraction*256	242	189	140
Freq.	**743**	**770**	**797**
Tiempo	1,350E-006	1,299E-006	1,250E-006
Tiempo/2 Ts	5.38	5.19	5.02
Period_integer	5	5	5
Period_fraction	0.38	0.19	0.02
Period_fraction*256	98	50	5
Freq.	**822**	**852**	**882**
Tiempo	1,220E-006	1,170E-006	1,130E-006
Tiempo/2 Ts	4.87	4.69	4.54
Period_integer	4	4	4
Period_fraction	0.87	0.69	0.54
Period_fraction*256	222	178	137
Freq.	**908**	**941**	**974**
Tiempo	1,100E-006	1,060E-006	1,030E-006
Tiempo/2 Ts	4.41	4.25	4.11
Period_integer	4	4	4
Period_fraction	0.41	0.25	0.11
Period_fraction*256	104	64	27

TABLA DE TIEMPOS Y DE FRECUENCIAS USADAS EN EL ESTIMADOR DE FRECUENCIA

GRUPO ALTO	-3.5% Fdtmf	Fdtmf	3.5% Fdtmf
Freq.	**1167**	**1209**	**1251**
Tiempo	8,570E-007	8,270E-007	7,990E-007
Tiempo/2 Ts	3.43	3.31	3.2
Period_integer	3	3	3
Period_fraction	0.43	0.31	0.2
Period_fraction*256	109	79	51
Freq.	**1289**	**1336**	**1382**
Tiempo	7,760E-007	7,490E-007	7,240E-007
Tiempo/2 Ts	3.1	2.99	2.89
Period_integer	2	2	2
Period_fraction	0.1	0.99	0.89
Period_fraction*256	26	254	229
Freq.	**1425**	**1477**	**1528**
Tiempo	7,020E-007	6,770E-007	6,540E-007
Tiempo/2 Ts	2.81	2.71	2.62
Period_integer	2	2	2
Period_fraction	0.81	0.71	0.62
Period_fraction*256	207	181	158
Freq.	**1576**	**1633**	**1690**
Tiempo	6,350E-007	6,120E-007	5,920E-007
Tiempo/2 Ts	2.54	2.45	2.37
Period_integer	2	2	2
Period_fraction	0.54	0.45	0.37
Period_fraction*256	138	115	94

BIBLIOGRAFIA.

-Recommendation Q.24: Multi-Frequency Push-Button Signal Reception: ITU Blue Book, 1989

-V. Friedman. "A zero crossing algorithm for the estimation of the frequency of a single sinusoid in white noise " *IEEE Trans. Signal Processing , vol 42, pp 1565-1569, June 1994.*

-Leon W. Cough II. " Sistemas de Comunicación digitales y Analógicos " Quinta Edición, Pearson Educación, 2001.

-Alan V. Oppenheim, "Tratamiento de señales en tiempo discreto ", Segunda Edición, Prentice Hall ,2000.

-www.ccsinfo.com.

-www.microchip.com

-www.mathworks.com.

-www.dlpdesign.com.