Modulación diferencial adaptativa por codificación de pulsos (ADPCM) empleando el microcontrolador MC68HC11

Rafael Boloix Tortosa
rboloix@viento.us.es

Antonio Luque Estepa
aluque@zipi.us.es

9 de junio de 2001

1 Introducción

El algoritmo ADPCM de compresión aprovecha la alta correlación existente entre muestras consecutivas de señales vocales para predecir muestras futuras a partir de las pasadas. En lugar de codificar cada una de las muestras de la señal, se puede codificar la diferencia entre la muestra y la predicción de dicha muestra, consiguiendo una reducción del número de bits por muestra manteniendo una adecuada calidad de la señal.

Este algoritmo se utiliza en una gran variedad de dispositivos, yendo sus aplicaciones desde los formatos de audio comprimido para ordenadores domésticos hasta el almacenamiento de pequeñas muestras de voz en microcontroladores industriales.

En esta memoria se describe la implementación del algoritmo ADPCM realizada sobre un microcontrolador modelo de la serie 68HC11 de Motorola [1]. Estos microcontroladores están realizados en tecnología HCMOS y han gozado desde su lanzamiento de una gran aceptación en el mercado, lo que ha propiciado que existan multitud de herramientas para desarrollar código y gran cantidad de subrutinas disponibles.

La versión concreta del ADPCM usada está basada en las Prácticas Recomendadas para Realzar la Compatibilidad de Audio Digital en Sistemas Multimedia, revisión 3.00, de la Asociación Multimedia Interactiva (IMA) [2].

2 Algoritmo ADPCM

En esta sección se describe el funcionamiento del algoritmo empleado. La versión implementada en el HC11 es una adaptación de una versión para microcontroladores PIC16/17, que es la que se describe aquí [3].

2.1 Compresión

El algoritmo de compresión toma una muestra de 16 bits de la señal a comprimir, con un valor entre -32768 y 32767 y devuelve un número de 8 bits que contiene el código ADPCM en signo y magnitud. Dicho código se obtiene cuantificando de forma adaptativa la diferencia entre cada muestra y la predicción almacenada, calculada con la muestra anterior. El compresor

incluye un descompresor, que toma el código ADPCM calculado, lo descuantifica, y produce una predicción de la siguiente muestra. La figura 1 muestra un esquema del codificador descrito.

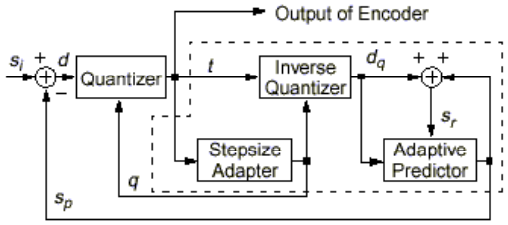

Figura 1: Diagrama de bloques del codificador

2.2 Descompresión

El descompresor toma como entrada un valor del mismo tamaño que la salida del compresor: 8 bits que contienen un valor de 4 bits donde sealmacena el código ADPCM. El valor de salida es un entero de 16 bits. Este descompresor es idéntico al que se encontraba en el interior del compresor (ver figura 1), y se puede ver en la figura 2.

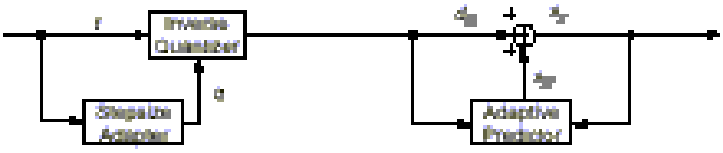

Figura 2: Diagrama de bloques del decodificador

3 Implementación en C

Antes de iniciar la programación del microcontrolador 68HC11 se ha decidido implementar el algoritmo en lenguaje C en un ordenador personal, para tener un marco de referencia probado con el que contrastar el funcionamiento del ADPCM en el HC11. Se ha utilizado el compilador Borland C++ Builder 3.0 para diseñar un sencillo programa que lee un fichero que contiene las muestras de 16 bits de la señal a codoficar y genera otro fichero con los códigos ADPCM de 8 bits de cada una de las muestras. Además permite visualizar en pantalla los códigos que se van calculando (ver figura 3).

También se ha realizado el programa descompresor que a partir del fichero generado por el programa anterior, es capaz de regenerar la señal original. En la figura 4 se muestra una captura de pantalla del programa descompresor.

En la sección 5 se discuten los resultados obtenidos con ambos programas, así como con la implementación en el microcontrolador.

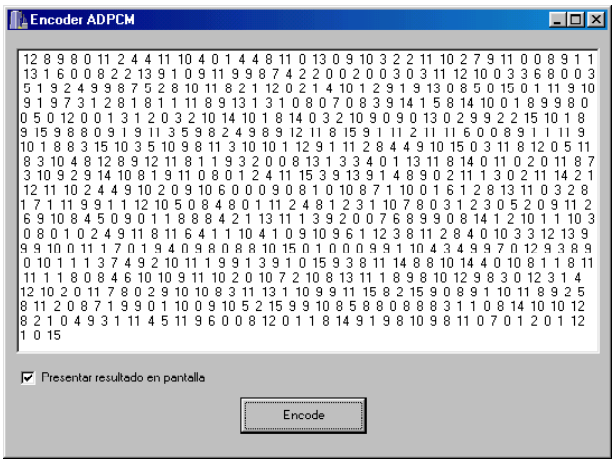

Figura 3: Programa codificador en C

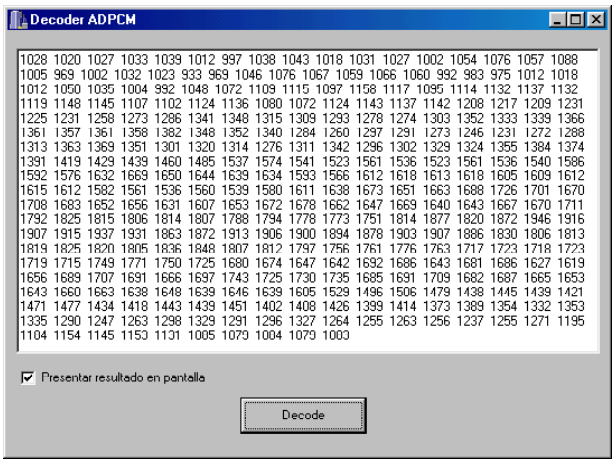

Figura 4: Programa decodificador en C

4 Implementación en HC11

En una aplicación real, el codificador debe obtener las muestras de la señal mediante un proceso de conversión analógica/digital de una señal obtenida del entorno. Así, una aplicación típica consistiría en la captura de datos desde un micrófono cuya salida estuviese conectada a una de las entradas analógicas del HC11. El programa del microcontrolador debería encargarse de programar el convertidor analógico digital para que muestrease a la frecuencia deseada. La depuración de un programa de estas características es muy dificultosa, por lo que una solución de compromiso consiste en introducir durante la fase de pruebas la señal ya muestreada en una zona de memoria, de donde el compresor lee los datos. Posteriormente, la subrutina encargada de leer los datos se podría cambiar por otra que los obtuviese del convertidor A/D.

Otro problema que surge con el uso del convertidor del HC11 es que este convertidor proporciona una señal analógica de 8 bits, mientras que el algoritmo de compresión espera un dato de entrada de 16 bits. La solución a emplear en este caso consiste en considerar los 8 bits de salida del convertidor como los más significativos de un dato de 16 que tuviese los otros 8 bits a 0.

A continuación se presenta el listado del programa compresor para el microcontrolador HC11.

```
1   ;PROGRAMA ENCODER ADPCM

    ;CONSTANTES

5   TOC5    EQU     $101E
    TFLG1   EQU     $1023
    TMSK1   EQU     $1022
    ADCTL   EQU     $1030
    OPTION  EQU     $1039
10  ADR1    EQU     $1031

    ;VARIABLES GLOBALES
            ORG     $0033
    index   RMB     1               ;STEP SIZE INDEX
15  previo  RMB     2               ;Sp
    code    RMB     1               ;CODIGO
    dif     RMB     2               ;DIFERENCIA
    step    RMB     2               ;PASO DE CUANTIZACION
    tstep   RMB     2               ;MODIFICAR PASO DE CUANTIZACION
20  difq    RMB     2               ;DIFERENCIA PRED.
    cuen    RMB     1
    TABLA4  RMB     100

    ;VECTORES DE INTERRUPCION
25          ORG     $FFFE
            FDB     inicio          ;RESET

    ;MODULO DE INICIO
            ORG     $E000
30  inicio  LDS     #$00FF          ;INICIO DE PILA
            CLR     index
            CLR     cuen
            CLR     code
            CLRA
35          CLRB
            STD     previo
            STD     dif
            STD     step
            STD     tstep
40          STD     difq


            ;EJECUCION DEL PROGRAMA

45  A1      LDY     #TABLA3
            LDAB    cuen
            ADDB    cuen
            ABY
            LDD     $00,Y
50          SUBD    previo          ;D=DIF=MUESTRA-PREVIO
```

```
            STD     dif
            BLT     C1
            CLRB                            ;SI DIF>=0, CODE=0
            BRA     C2
55  C1      LDD     #0
            SUBD    dif
            STD     dif
            LDAB    #8              ;SI DIF<0, CODE=8
    C2      STAB    code
60
            LDX     #TABLA
            LDAB    index
            ADDB    index
            ABX
65          LDD     $00,X           ;B=STEP=TABLA[INDEX]
            STD     step
            STD     tstep

            LDD     dif
70          CPD     tstep           ;COMPARO DIF Y TSTEP
            BLT     E1
            LDAB    code
            ORAB    #4
            STAB    code            ;CODE=CODE OR 00000100
75          LDD     dif
            SUBD    tstep
            STD     dif             ;DIF=DIF-STEP
    E1      LDD     tstep
            LSRD
80          STD     tstep           ;tstep=tstep>>1

            LDD     dif
            CPD     tstep           ;COMPARO DIF Y TSTEP
            BLT     E2
85          LDAB    code
            ORAB    #2
            STAB    code            ;CODE=CODE OR 00000010
            LDD     dif
            SUBD    tstep
90          STD     dif             ;DIF=DIF-STEP
    E2      LDD     tstep
            LSRD
            STD     tstep           ;tstep=tstep>>1

95          LDD     dif
            CPD     tstep           ;COMPARO DIF Y TSTEP
            BLT     E3
            LDAB    code
            ORAB    #1
100         STAB    code            ;CODE=CODE OR 00000001

    E3      LDD     step
            LSRD
```

```
            LSRD
105         LSRD                    ;dfiq=step>>3
            STD     difq
            LDAB    code
            ANDB    #4
            BEQ     E4              ;if(code & 4), diffq+=step
110         LDD     difq
            ADDD    step
            STD     difq
      E4    LDAB    code
            ANDB    #2
115         BEQ     E5              ;if(code & 2)
            LDD     step
            LSRD                    ;dfiq=step>>1
            ADDD    difq
            STD     difq
120   E5    LDAB    code
            ANDB    #1
            BEQ     E6              ;if(code & 6)
            LDD     step
            LSRD
125         LSRD                    ;dfiq=step>>2
            ADDD    difq
            STD     difq

      E6    LDAB    code
130         ANDB    #8
            BEQ     D1              ;if(code & 8)
            LDD     previo
            SUBD    difq            ;previo-=diffq
            BRA     D2              ;else
135   D1    LDD     previo
            ADDD    difq            ;previo+=diffq
      D2    STD     previo

            LDX     #TABLA2
140         LDAB    code
            ABX
            LDAA    $00,X
            LDAB    index
            ABA                     ;INDEX+=INDEXTABLE[CODE]
145         CMPA    #$00
            BGE     D5              ;if(index<0)
            CLRA                    ;index=0;
            BRA     D6
      D5    LDAB    #88
150         CBA
            BLE     D6              ;if(index>88)
            TBA                     ;index=88;
      D6    STAA    index

155         LDAA    code
            LDX     #TABLA4
```

```
            LDAB    cuen
            ABX
            STAA    $00,X
160
            INCB
            STAB    cuen
            CMPB    #100
            BEQ     A2
165         JMP     A1

            ;FIN EJECUCION DEL PROGRAMA
    A2      NOP
    princ   BRA     princ
170

    TABLA   FDB     7,8,9,10,11,12,13,14,16,17,19,21,23,25,28,31,34,37,41,45,
                    50,55,60,66,73,80,88,97,107,118,130,143,157,173,190,209,230,
                    253,279,307,337,371,408,449,494,544,598,658,724,796,876,963,
175                 1060,1166,1282,1411,1552,1707,1878,2066,2272,2499,2749,3024,
                    3327,3660,4026,4428,4871,5358,5894,6484,7132,7845,8630,9493,
                    10442,11487,12635,13899,15289,16818,18500,20350,22385,24623,
                    27086,29794,32767
    TABLA2  FCB     $FF,$FF,$FF,$FF,$02,$04,$06,$08,$FF,$FF,$FF,$FF,$02,$04,
180                 $06,$08
    TABLA3  FDB     0,1492,2978,4453,5910,7344,8749,10119,11449,12734,13969,
                    15149,16269,17325,18312,19227,20067,20827,21505,22097,22603,
                    23020,23346,23579,23720,23767,23720,23579,23346,23020,22603,
                    22097,21505,20827,20067,19227,18312,17325,16269,15149,13969,
185                 12734,11449,10119,8749,7344,5910,4453,2978,1492,-1,-1493,
                    -2979,-4454,-5911,-7345,-8750,-10120,-11450,-12735,-13970,
                    -15150,-16270,-17326,-18313,-19228,-20068,-20828,-21506,
                    -22098,-22604,-23021,-23347,-23580,-23721,-23767,-23721,
                    -23580,-23347,-23021,-22604,-22098,-21506,-20828,-20068,
190                 -19228,-18313,-17326,-16270,-15150,-13970,-12735,-11450,
                    -10120,-8750,-7345,-5911,-4454,-2979,-1493
```

Las líneas de definición de tablas deben introducirse en una sola línea. Aquí se han separado en varias sólo por claridad.

A la hora de obtener los datos de `TABLA3` hay que tener cuidado de no leer el mismo byte dos veces. Como los datos son de 16 bits, se debe incrementar el puntero que se usa para leer el dato dos veces antes de acceder el siguiente dato.

Para tratar con algo más parecido a datos de 8 bits se podría poner a 0 el acumulador `B` antes de empezar el algoritmo, pero en este programa no se ha hecho.

El siguiente listado es el programa descompresor.

```
1   ;PROGRAMA DECODER ADPCM

    ;CONSTANTES

5   TOC5    EQU     $101E
```

```
TFLG1   EQU     $1023
TMSK1   EQU     $1022
ADCTL   EQU     $1030
OPTION  EQU     $1039
ADR1    EQU     $1031

;VARIABLES GLOBALES
        ORG     $0033
index   RMB     1               ;STEP SIZE INDEX
previo  RMB     2               ;Sp
code    RMB     1               ;CODIGO
step    RMB     2               ;PASO DE CUANTIZACION
difq    RMB     2               ;DIFERENCIA PRED.
cuen    RMB     1
TABLA4  RMB     200

;VECTORES DE INTERRUPCION
        ORG     $FFFE
        FDB     inicio          ;RESET

;MODULO DE INICIO
        ORG     $E000
inicio  LDS     #$00FF          ;INICIO DE PILA
        CLR     index
        CLR     cuen
        CLR     code
        CLRA
        CLRB
        STD     previo
        STD     step
        STD     difq


        ;EJECUCION DEL PROGRAMA

A1      LDY     #TABLA3
        LDAB    cuen
        ABY
        LDAA    $00,Y
        STAA    code

        LDX     #TABLA
        LDAB    index
        ADDB    index
        ABX
        LDD     $00,X           ;D=STEP=TABLA[INDEX]
        STD     step
        LSRD
        LSRD
        LSRD                    ;dfiq=step>>3
        STD     difq
        LDAB    code
        ANDB    #4
```

```
        BEQ     E4              ;if(code & 4), diffq+=step
        LDD     difq
        ADDD    step
        STD     difq
E4      LDAB    code
        ANDB    #2
        BEQ     E5              ;if(code & 2)
        LDD     step
        LSRD                    ;dfiq=step>>1
        ADDD    difq
        STD     difq
E5      LDAB    code
        ANDB    #1
        BEQ     E6              ;if(code & 6)
        LDD     step
        LSRD
        LSRD                    ;dfiq=step>>2
        ADDD    difq
        STD     difq

E6      LDAB    code
        ANDB    #8
        BEQ     D1              ;if(code & 8)
        LDD     previo
        SUBD    difq            ;previo-=diffq
        BRA     D2              ;else
D1      LDD     previo
        ADDD    difq            ;previo+=diffq
D2      STD     previo

        LDX     #TABLA2
        LDAB    code
        ABX
        LDAA    $00,X
        LDAB    index
        ABA                     ;INDEX+=INDEXTABLE[CODE]
        CMPA    #$00
        BGE     D5              ;if(index<0)
        CLRA                    ;index=0;
        BRA     D6
D5      LDAB    #88
        CBA
        BLE     D6              ;if(index>88)
        TBA                     ;index=88;
D6      STAA    index

        LDX     #TABLA4
        LDAB    cuen
        ADDB    cuen
        ABX
        LDD     previo
        STD     $00,X
```

```
            LDAB    cuen
            INCB
            STAB    cuen
115         CMPB    #100
            BEQ     A2
            JMP     A1

            ;FIN EJECUCION DEL PROGRAMA
120 A2      NOP
    princ   BRA     princ


    TABLA   FDB     7,8,9,10,11,12,13,14,16,17,19,21,23,25,28,31,34,37,41,45,50,
125                 55,60,66,73,80,88,97,107,118,130,143,157,173,190,209,230,253,
                    279,307,337,371,408,449,494,544,598,658,724,796,876,963,1060,
                    1166,1282,1411,1552,1707,1878,2066,2272,2499,2749,3024,3327,
                    3660,4026,4428,4871,5358,5894,6484,7132,7845,8630,9493,10442,
                    11487,12635,13899,15289,16818,18500,20350,22385,24623,27086,
130                 29794,32767
    TABLA2  FCB     $FF,$FF,$FF,$FF,$02,$04,$06,$08,$FF,$FF,$FF,$FF,$02,$04,
                    $06,$08
    TABLA3  FCB     0,7,7,7,7,7,7,7,7,7,1,0,0,1,0,0,1,0,0,0,1,0,0,0,0,8,0,8,9,
                    8,9,10,10,11,12,11,12,12,11,12,11,12,11,12,11,11,12,11,12,
135                 11,11,12,11,11,12,11,11,12,10,11,11,11,12,10,11,10,11,11,10,
                    11,10,10,10,9,9,8,0,1,3,3,6,4,3,5,3,4,4,3,4,3,4,4,3,3,3,5,3,
                    3,4,3
```

5 Resultados obtenidos

Tanto el programa en C como el realizado en ensamblador para el HC11 han sido probados codificando y luego decodificando una onda senoidal, y comparando la señal recuperada con la original.

En la figura 5 se muestra el resultado obtenido con el programa. En esta figura se ve la onda original, perfectamente senoidal, y la onda recuperada después de ser codificada y decodificada. Se aprecia que la onda de salida no coincide con la original al principio, pero llega un punto en el que ambas son prácticamente idénticas.

EL resultado de aplicar la misma prueba al programa para HC11 se representa en la figura 6

Se puede ver como el resultado es igual de bueno que con el programa en C para ordenador personal. El número de muestras representado es menor en el caso del HC11 por lo tedioso de simular el programa, teniendo que introducir los valores a mano y uno a uno en el programa usado para simular el microcontrolador, como ya se mencionó en la sección 4

Hay que mencionar la importancia de que las variables usadas tengan el tamaño adecuado. A la hora de realizar el programa en C es importante saber qué tamaño asigna el compilador a cada tipo de variable. Si se utiliza un compilador equivocado, el resultado difiere mucho del esperado.

En la figura 7 se presenta el resultado obtenido con un programa de estas características en el que el compilador asignó tamaños de 32 bits a variables que debían ser de 16. Como se

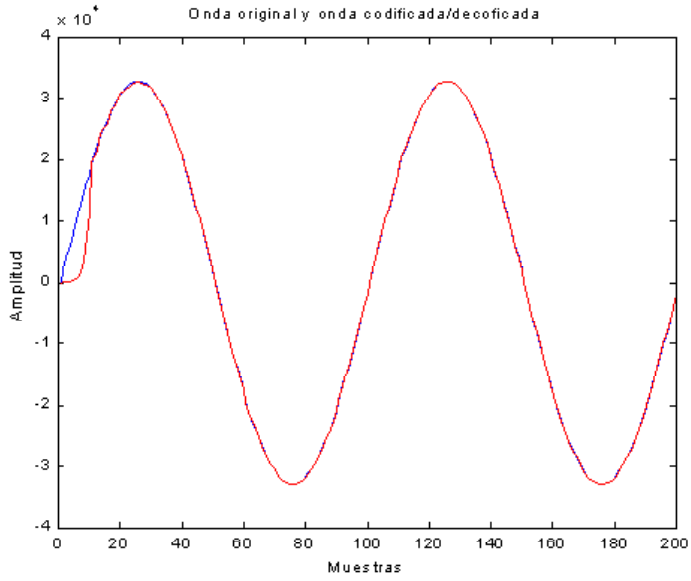

Figura 5: Resultado del programa en C

puede ver, los errores son muchos mayores que en el programa correspondiente con variables de 16 bits.

6 Conclusiones

El algoritmo ADPCM está especialmente indicado para aplicaciones de almacenamiento de voz, en las que la correlación entre muestras consecutivas es muy alta y no es importante que haya una pequeña diferencia entre la señal recuperada y la original.

Esta diferencia es manifiestamente apreciable en las figuras 5 y 6 en las que las ondas original y recuperada no coinciden al principio por la falta de suficientos muestras para que el predictor adaptativo sea capaz de estimar las muestras futuras.

En la mayoría de las aplicaciones de voz este error no es importante. En las figuras 8 y 9 se muestran dos ondas correspondientes a la voz humana, una original y otra comprimida y desomprimida. Apenas existen diferencias entre ambas, y el oído humano reconoce ambas igualmente.

Referencias

[1] Motorola Semiconductors, Inc. *M68HC11 Reference Manual*, 1996

[2] Interactive Multimedia Association, *Recommended Practices for Enhancing Digital Audio Compatibility in Multimedia Systems*, Revision 3.00, 21 Octubre 1992.

[3] Rodger Richey, Advanced Microcontroller and Technology Division, Microchip Technology Inc., *Adaptive Differential Pulse Code Modulation using PIC16/17 microcontrollers*, Application Note AN643, 1996.

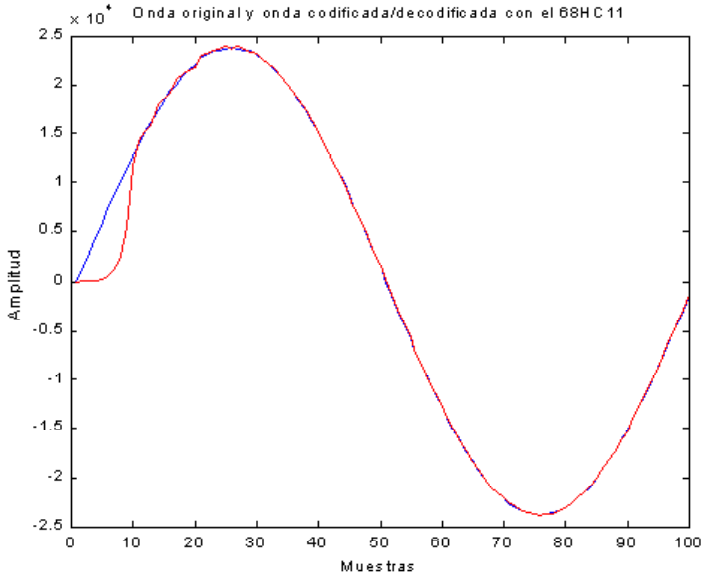

Figura 6: Resultado del programa para el HC11

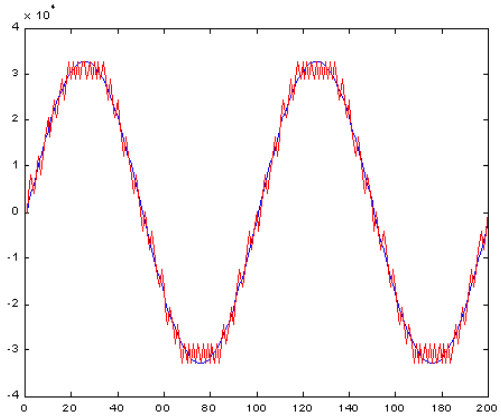

Figura 7: Resultado obtenido con un programa en C con tamaños incorrectos de las variables

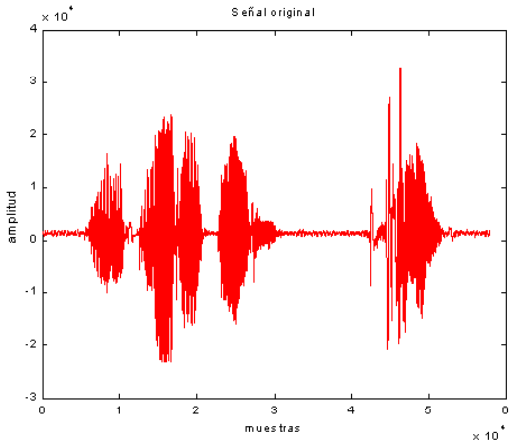

Figura 8: Onda de voz original

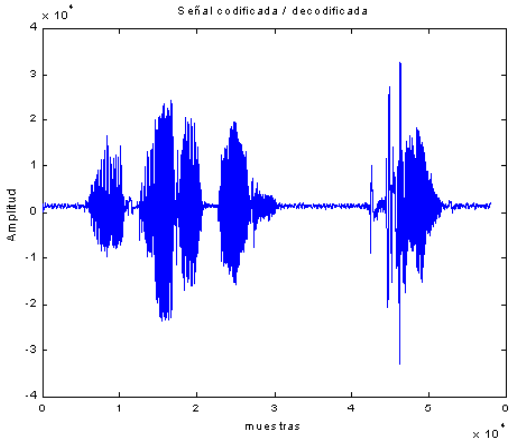

Figura 9: Onda de voz recuperada