Ediciones pensando el futuro
Editions thinking the future
Informática 2013
XV CONVENCIÓN Y FERIA INTERNACIONAL
15TH INTERNATIONAL CONVENTION AND FAIR
www.informaticahabana.com
www.informaticahabana.cu
ISBN: 978-959-7213-02-4

# SLD104 IMPLEMENTACIÓN EN TIEMPO REAL DE ALGORITMO DE COMPRESIÓN DE SEÑALES EEG BASADO EN CODIFICACIÓN DE SUBBANDAS

## SLD104 REAL TIME IMPLEMENTATION OF EEG SIGNAL COMPRESSION ALGORITHM BASED ON SUBBAND CODING

Carlos Bazán-Prieto[1], Jorge Sánchez-Pérez[2] Julián Cárdenas-Barrera[3]

1 Universidad Central "Marta Abreu" de Las Villas, Cuba, cabazan@uclv.edu.cu Carretera a Camajuaní km 5½
2 Universidad Central "Marta Abreu" de Las Villas, Cuba
3 Universidad Central "Marta Abreu" de Las Villas, Cuba, julian@uclv.edu.cu

**RESUMEN:** *En este trabajo se analiza la implementación en tiempo real de un algoritmo de compresión con pérdidas, para señales EEG desarrollado por los autores. El algoritmo, basado en descomposición en subbandas y codificación de Golomb, posee un buen desempeño en cuanto a tasa de compresión y calidad de la señal reconstruida. Para evaluar la aplicabilidad del algoritmo en entornos de variados recursos computacionales y particularmente en equipamiento portátil, se analiza el costo computacional de las etapas que lo conforman y se realiza la implementación en varias plataformas hardware incluyendo un dispositivo móvil. Como parámetro de evaluación se establece el tiempo de ejecución del algoritmo en comparación con el tiempo necesario para la ejecución en tiempo real. En todos los casos analizados se logra superar el tiempo real de ejecución. Los tiempos de ejecución alcanzados toman valores desde 172 veces menor que el tiempo real en la más rápida de las plataformas hardware utilizadas, hasta poco más de tres veces en la plataforma de teléfono móvil. Con estos resultados se demuestra la factibilidad de utilizar el algoritmo seleccionado en dispositivos de hardware de bajo perfil para aplicaciones portátiles y ambulatorias. El mayor costo computacional del algoritmo reside en la operación de filtrado, por lo que el empleo de realizaciones rápidas existentes de esta, puede contribuir a mejorar la operación en tiempo real del método particularmente en plataformas móviles.*

***Palabras Clave:*** Compresión de señales EEG, Codificación en subbandas, Codificación de Golomb, Implementación en tiempo real

**ABSTRACT***: This paper analyses the real time implementation of a lossy compression algorithm for EEG signals designed by the authors. The algorithm, based on subband decomposition and Golomb coding, has obtained good results in terms of compression ratio and reconstructed signal quality. In order to assess the applicability of the algorithm using different hardware platforms, particularly in portable devices, the computational cost of the stages of the algorithm analyzed and its implementation in several hardware platforms including a mobile device are presented. The ratio between the execution time of the algorithm to the time needed for real time execution is used as performance criterion. In all the cases studied, the algorithm executed in real time. The execution times ranged from 172 times shorter than real time, using the fastest platform; to three times in the mobile phone platform. These results demonstrate the possibility of using this lossy compression algorithm for EEG signals in low profile hardware devices for portables and ambulatory applications. The main computational burden of the algorithm resides in the filtering operation and hence, existing fast implementations of this stage can improve the real time operation of the algorithm particularly with mobile devices.*

***KeyWords:*** EEG signal compression, Subband coding, Golomb coding, real time implementation

## 1. INTRODUCCIÓN

El EEG tradicional es un estudio no invasivo y no doloroso que brinda gran cantidad de información relacionada con varias enfermedades. Actualmente se emplea de manera creciente en múltiples aplicaciones [1]. Entre estas aplicaciones las más usadas están relacionadas con los estudios de la epilepsia y los desórdenes del sueño. Estos estudios requieren generalmente la adquisición de la señal EEG por un tiempo prolongado, para su posterior análisis.

En Neurofisiología Clínica se impone cada vez más el uso de equipos muy pequeños, compactos y portátiles. Estos equipos deben ser capaces de registrar un número importante de variables fisiológicas durante largos períodos de tiempo, almacenarlas, procesarlas y trasmitirlas generalmente de forma inalámbrica para su futuro procesamiento por parte de los especialistas. Actualmente se desarrollan investigaciones para aplicar técnicas ambulatorias y portables para la adquisición de señales EEG de larga duración [2, 3].

En cada aplicación [1] el empleo de la señal EEG no requiere de igual forma la información contenida en la señal EEG digital. Por ejemplo, en [4] se concluye que se puede aceptar una distorsión hasta un 30 % de "*Percent Root mean square Difference"* (PRD), para aplicaciones de detección automática de enfermedades. La gran cantidad de aplicaciones existentes y en desarrollo, se convierten en escenarios propicios para el empleo de algoritmos eficientes de compresión que faciliten el procesamiento de las señales EEG y reduzcan los costos de las implementaciones. En especial, los algoritmos de compresión de señales EEG se han utilizado en equipos ambulatorios [5, 6].

Los algoritmos de compresión de señales EEG se pueden clasificar en dos categorías: compresión sin pérdida, y compresión con pérdidas. La compresión sin pérdidas logra una reconstrucción de la señal EEG sin distorsión, sin embargo, solo se pueden alcanzar modestas tasas de compresión con valores inferiores a 3 [7-9], [10]. Por otra parte, los algoritmos de compresión con pérdidas introducen una distorsión en la señal recuperada, pero logran tasas de compresión mayores a los algoritmos anteriores, llegan a alcanzar valores superiores a 10 [11-15].

El objetivo de este trabajo es evaluar el desempeño de un algoritmo de compresión con pérdidas para señales EEG en un dispositivo móvil. El algoritmo elegido reporta elevadas tasa de compresión y calidad de la señal reconstruida [16], las distintas etapas del algoritmo se describen brevemente. Previo a la aplicación en el dispositivo móvil, se evalúan diferentes plataformas de hardware, para poder-comparar los resultados y la viabilidad del empleo del hardware de bajo perfil.

## 2. APLICACIÓN DEL ALGORITMO DE COMPRESIÓN

El algoritmo elegido para la aplicación, realiza la compresión de señales EEG basado en la descomposición en subbandas y codificación de Golomb [16]. Primeramente se realiza una breve descripción de las etapas y del orden de complejidad. A continuación se presentan las plataformas de hardware donde se aplica el algoritmo. Finalmente se presentan los resultados, resaltando el desempeño de la aplicación en un dispositivo móvil.

### 2.1 Algoritmo de compresión

La codificación de Golom-Rice [17] es una variante del algoritmo de codificación de Golomb [18] que es óptimo o casi óptimo para codificar datos que tienen una distribución geométrica [19], como es el caso de la señal EEG con distribución Gaussiana generalizada uniformemente cuantificada con zona muerta [20]. La codificación de Golomb genera un código de longitud variable, que está basado en un simple modelo de la probabilidad de los valores, donde los de poca amplitud son codificados más eficientemente que los de mayor amplitud. La precisa relación entre el tamaño y la probabilidad, es capturada en un parámetro: el divisor. Para codificar un número con este método, se encuentra el cociente y el resto de la división con este divisor. Se escribe el cociente en notación unaria y el resto en notación binaria truncada. En la práctica es necesario utilizar un bit separador entre ambos. La longitud del resto estará dada por el divisor. La codificación de Golomb-Rice es una variante de la codificación de Golomb donde el divisor es una potencia de 2, permitiendo una realización práctica más eficiente y menos costosa.

Las etapas del algoritmo de compresión se muestran en el diagrama de bloques de la Figura 1 y se describen brevemente a continuación, conjuntamente con el orden de complejidad en la notación *O(f(n))*:

Etapa 1: La señal EEG es segmentada en bloques de *N* muestras, *x*[*n*]. El tamaño *N* del segmento debe ser pequeño para garantizar que la señal EEG sea estacionaria, a pesar de que segmentos de gran tamaño puedan beneficiar el desempeño del algoritmo de compresión. Hay que tomar en consideración que el tamaño del segmento define también la demora introducida en el proceso.

Cada segmento x[n] es descompuesto en subbandas, y[n], empleando un banco de filtros coseno modulado de reconstrucción aproximada, "*Nearly-*

*Perfect Reconstruction Cosine Modulated Filter Bank*" (N-PR CMFB). El banco de filtros descompone el espectro de frecuencia de la señal EEG en 32 bandas (*M*) con el objetivo de obtener bandas de frecuencia de 4 Hz, aproximadamente en correspondencia con el espectro de frecuencia de los ritmos EEG [21, 22]. El orden de complejidad de esta etapa es *O(N*M*P)* donde *P* es el orden del filtro, para la implementación directa [23].

Etapa 2: El valor umbral (**ε**) se obtiene basado en la técnica de la energía retenida [24] en la señal descompuesta en subbandas ($y_i$), según el *PRD* deseado. Las muestras ($y_i$) son umbralizadas y cuantificadas utilizando un cuantificador uniforme con zona muerta (USDZQ). USDZQ, puede ser visto como una operación de umbralización seguido por una cuantificación uniforme escalar [25]. La ecuación (1), describe el funcionamiento de este cuantificador, donde los valores de $y_i$ son las muestras subbandas, $\hat{y}_i$ las muestras cuantificadas y ε el valor umbral definido a priori. Los valores $y_i$ que se encuentran en el intervalo ± **ε**, la llamada zona muerta, se cuantifican como 0 (equivalente al proceso de umbralización) donde no existirán valores significativos. La mitad del paso de cuantificación es δ, siendo δ < **ε** < 2δ, en la práctica se tomó δ = 0.6 **ε** en base a los experimentos realizados. Otro caso particular ocurre en el intervalo de decisión $\varepsilon \le |y_i| \le 3\delta$ donde los valores $y_i$ se cuantifican con valor 1. El resto de los intervalos son uniformemente cuantificados con un paso de cuantificación de 2δ.

$$\hat{y}_i = \begin{cases} -1, & -3\delta \le y_i \le -\varepsilon \\ 0, & -\varepsilon < y_i < \varepsilon \\ 1, & \varepsilon \le y_i \le 3\delta \\ h, & (2h-1)\delta < y_i \le (2h+1)\delta, \\ & h = \pm 2, \pm 3, \ldots \end{cases} \quad (1)$$

Los niveles de reconstrucción correspondientes, $\hat{y}_{iR}$, son:

$$\hat{y}_{iR} = \begin{cases} 0, & y_i = 0 \\ 2\hat{y}_i\delta, & y_i \neq 0 \end{cases} \quad (2)$$

Como resultado de esta etapa se obtienen las muestras cuantificadas $(\hat{y}_i)$, donde los ceros representan a las muestras originales con valor inferior al umbral (**ε**) y los valores distintos de cero son las muestras significativas cuantificadas. El orden de complejidad de esta etapa es *O(Nlog$_2$(N)),* dominado por el ordenamiento de las muestras; sin embargo debe considerarse además que previo a la determinación del umbral es preciso el cálculo de la energía retenida. Este proceso tiene un orden de complejidad *N* que se añade a la complejidad del ordenamiento. Luego del ordenamiento también se requiere de la determinación de la energía retenida en función del PRD deseado que tiene una complejidad inferior a N. Por tal razón la complejidad de la determinación del umbral se selecciona del orden de *O(N + Nlog$_2$(N)).*

Etapa 3: Dado $\hat{y}_i$ que contiene las muestras subbandas codificadas, formado por las *M* subbandas concatenadas, cada una con *M/N* valores, donde *N* es el tamaño del segmento. La codificación de las muestras cuantificadas $\hat{y}_i$, se realiza por separado, por una parte los valores $\hat{y}_i \neq 0$, y por otra parte los valores $\hat{y}_i = 0$:

Para los valores $\hat{y}_i \neq 0$, primeramente se codifica el signo ya que Golomb-Rice solo codifica enteros positivos, de forma que los valores $\hat{y}_i \neq 0$ se convierten en valores positivos $y_{ci}$, de la siguiente forma:

$$y_{ci} = \begin{cases} 2|\hat{y}_i| - 1, & \hat{y}_i < 0, \\ 2|\hat{y}_i| - 2, & \hat{y}_i > 0. \end{cases} \quad (3)$$

Las muestras subbandas codificadas con signo codificado $y_{ci}$, son codificadas con Golomb-Rice, donde el parámetro ajustable "divisor" (*m*) es potencia de 2. Dado el divisor, como un parámetro entero positivo $m = 2^k$, la palabra de código para $y_{ci}$ se construye a partir de la representación unaria de $\lfloor y_{ci}/m \rfloor$, seguido por un '0' como separador, y los *k* bits menos significativos (resto, $r = y_{ci} \bmod m$). La longitud del código generado es $\lfloor y_{ci}/m \rfloor + 1 + k$. Para cada distribución de esta forma, hay un valor del parámetro *k* tal que alcance la longitud promedio más corta posible. La estimación del parámetro *k* se realiza de la siguiente forma:

$$k = \left\{ j \middle| 2^j \ge \frac{A}{N} \right\}, \quad (4)$$

donde *N* es el conteo de valores $\hat{y}_i$ y *A* la suma acumulada. Después de codificar una muestra $y_{ci}$, *N* es incrementada por 1 y *A* es incrementada por $|\hat{y}_i| - 1/2$. Cuando *N* alcance su valor máximo (*Nmax* = 8, empíricamente seleccionado), *N* y *A* son reajustados, haciendo $N = \lfloor N/2 \rfloor$ y $A = \lfloor A/2 \rfloor$; finalmente se calcula el nuevo valor de $k = \lceil \log_2(A/N) \rceil$.

La estructura del código es la siguiente:

$$\overbrace{1 \ \ldots \ 1}^{\left\lfloor \frac{y_{ci}}{m} \right\rfloor} \ 0 \ \overset{k}{r} \qquad (5)$$

Para los valores $\hat{y}_i = 0$ se utiliza la codificación de Golomb-Exp, la cual es casi óptima y robusta para codificar datos exponenciales con una razón de cai-

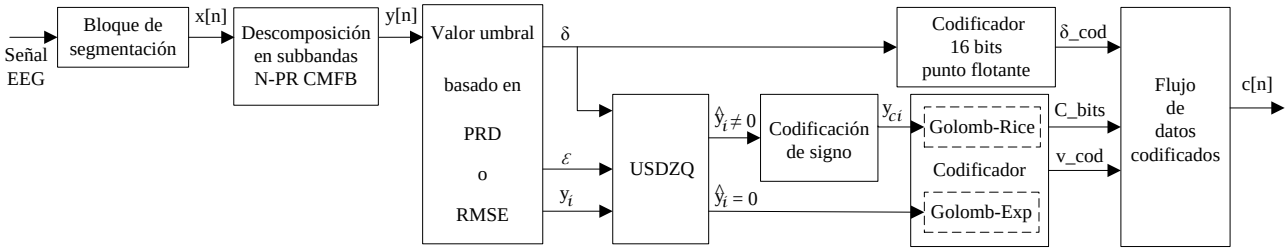


**Figura 1: Diagrama de bloques del algoritmo de compresión**

da desconocida, de forma que la longitud de las corridas de ceros son codificadas eficientemente. La palabra de código resultante de la codificación de las corridas de ceros, está formada por *k* '0's, seguidos de un bit separador '1', finalizando con la representación binaria (*k* bits) del valor $RL + 1 - 2^k$, como se muestra a continuación:

$$\overbrace{0 \ \ldots \ 0}^{k} \ 1 \ \overbrace{x_{k-1} \ldots \ x_0}^{k}, \qquad (6)$$

y la representación binaria

$$x_{k-1} \ldots \ x_0 = RL + 1 - 2^k, \qquad (7)$$

donde *RL* es la cantidad de ceros consecutivos y *k* es calculado como:

$$k = \lfloor \log_2(RL + 1) \rfloor. \qquad (8)$$

Cada palabra de código tiene la longitud *2k +1* bits. Los códigos resultantes son concatenados, según el orden original, como valores codificados (v_cod). Finalmente se obtiene un flujo de bits c[n], con una estructura para cada segmento formada por: a) valor δ codificado en 16 bits en punto flotante (δ_cod), b) cantidad de bits codificados (C_bits) en (v_cod) representados con [log2(longitud del segmento) + 1] bits, y c) valores codificados (v_cod). El orden de complejidad de esta etapa es *O(N)*. La Tabla I presenta un resumen del estimado del orden de complejidad de las diferentes etapas del algoritmo a implementar cuando el bloque de datos a transformar es de 2048 muestras y e banco de filtros de orden 287 y 32 bandas.

**Tabla I: Orden de complejidad del algoritmo**

| Etapa | MOPS | % Total |
|---|---|---|
| **Banco de Filtros** | 18,874368 | 99,69% |
| **Determinación Umbral** | 0,024576 | 0,13% |
| **Umbralización** | 0,002048 | 0,01% |
| **Codificación Golomb** | 0,032768 | 0,17% |
| **Total** | 18,933760 | 100,00% |

Como puede observarse, el algoritmo necesita de poco menos de 20 millones de operaciones por segundo (MOPS) lo que lo hace factible de implementar en tiempo real en la mayoría de las plataformas hardware disponibles hoy en día. La etapa de filtrado es precisamente la que emplea más del 99 % de las operaciones necesarias para realizar la compresión. Tomando en cuenta que la mayoría de las plataformas hardware actuales poseen una microarquitectura superescalar en la que dinámicamente se asignan los recursos para ejecutar un proceso dado y que simultáneamente coexisten varios procesos en ejecución en el procesador, es necesario identificar el tiempo requerido por una arquitectura de computador para ejecutar el algoritmo. Conociendo el tiempo de ejecución del algoritmo podremos inferir que su empleo en una plataforma dada puede ofrecer o no ejecución en tiempo real y cuántos canales EEG pueden ser comprimidos simultáneamente o si se pueden realizar otros procesos además del proceso de compresión.

## 2.2 Desarrollo de la aplicación

Uno de los principales objetivos de la compresión de señales EEG, es facilitar los estudios de larga duración realizados con equipos portátiles e inalámbricos. Así, la presente investigación evalúa el desempeño del algoritmo en diferentes plataformas de computadoras incluyendo la implementación en un dispositivo móvil. Se han seleccionado, como se muestra en la Tabla II, computadoras personales con diferentes características y un dispositivo móvil HTC Desire Z con un procesador ARM de 800 MHz y 512 MB de memoria, con sistema operativo Android 2.2.1. Este sistema operativo fue seleccionado por ser de código abierto basado en Linux y brindar facilidades para el desarrollo de aplicaciones. También ha sido importante, la disponibilidad de herramientas que permiten desarrollar aplicaciones en la plataforma Android utilizando el lenguaje de pro-

gramación Java. Para realizar la implementación en Java del Algoritmo de compresión se utiliza Eclipse, un entorno de desarrollo integrado (IDE por sus siglas en inglés) multiplataforma y de código abierto que cuenta además con módulos para desarrollar aplicaciones Android.

Luego de desarrollar la programación Java del algoritmo de compresión elegido, se realiza un experimento para identificar la capacidad de ejecutarse en tiempo real en diversas plataformas. El objetivo del experimento es identificar, en primer lugar las demandas de recursos de cómputo que realiza el algoritmo. En segundo lugar, si es capaz de ejecutar la compresión-decompresión en tiempo real, de manera que quede tiempo suficiente para ejecutar otras aplicaciones por parte del sistema de cómputo que lo ejecute. Por último, identificar aquellos procesos del algoritmo que demandan mayor número de recursos para sugerir implementaciones más eficientes de los mismos que conduzcan a mejoras de los tiempos de ejecución.

Los experimentos propuestos permiten identificar el tiempo total de la compresión (*TejecT*) y el porciento que este tiempo representa del tiempo necesario para su ejecución en tiempo real (*xRT*), dado por la duración de la señal:

$$xRT = \frac{TejecT}{Duración\ de\ la\ señal} 100\%. \quad (9)$$

También, se determina el porciento del tiempo de procesamiento ($TejecE_x$) consumido por cada etapa ($x$) del algoritmo de compresión y el tiempo total de la compresión (*TejecT*): $T_{P1}$ (tiempo de procesamiento de la etapa 1, filtrado), $T_{P2}$ (tiempo de procesamiento de la etapa 2, cuantificación) y $T_{P3}$ (tiempo de procesamiento de la etapa 3, codificación). La expresión general es:

$$T_{Px} = \frac{TejecE_x}{TejecT} 100\%. \quad (10)$$

## 2.3 Resultados

El algoritmo propuesto constituye un códec prácticamente simétrico dado que en la descompresión sólo se excluyen las operaciones de determinación del umbral y de elección de los parámetros de la codificación Golomb empleada. Estas operaciones representan un porciento muy pequeño de la carga computacional como se observa en la Tabla I. De esta manera podemos considerar que la complejidad computacional de la etapa de codificación (compresión) es similar a la de la etapa de decodificación (descompresión). Por esta razón en los experimentos, solo se incluyen los resultados del proceso de compresión ya que estos resultarían similares en el proceso de descompresión. La principal demanda de recursos de cómputo del algoritmo está relacionada con la potencia de cálculo. Las demandas de memoria no son significativas dadas la frecuencia de muestreo relativamente bajas de las señales EEG (256Hz en la base de datos empleada [26]) y los métodos empleados por el algoritmo. Las principales demandas de memoria están referidas al almacenamiento de los segmentos de datos en el dominio del tiempo. Las muestras de las subbandas de frecuencia y los valores codificados que en total no superan los 512KB de memoria cuando se trabajan, por ejemplo, segmentos de 8 segundos (2048 muestras) de duración. A partir de las consideraciones anteriores, se ha decidido identificar las demandas de cómputo medidas a partir de los tiempos de ejecución.

El tamaño de los segmentos, en que se divide la señal EEG, es un importante elemento a tomar en

**Tabla II: Arquitectura de computadoras empleadas**

| PC | Tipo | Procesador | Memoria | Sistema Operativo |
|---|---|---|---|---|
| 1 | Escritorio | Intel(R) Core(TM) i3 3.0 GHz | 2GB DDR23 | Microsoft Windows XP SP 3 |
| 2 | Portátil | AMD Turion(tm) 2.0 GHz | 2.5GB DR2 | Microsoft Windows 7 (6.1) |
| 3 | Portátil | Intel(R) Celeron(R) 2.0 GHz | 2 GBDR2 | Microsoft Windows 7 (6.1) |
| 4 | Portátil | Intel(R) Pentium(R) 1.6 GHz | 256MB DDR | Microsoft Windows XP SP 3 |
| 5 | Escritorio | Intel(R) Celeron(R) 2.0 GHz | 256MBDDR2 | Microsoft Windows XP SP 3 |
| 6 | Tel. Móvil | Qualcomm MSM 800 MHz | 512 MB | Android 2.2.1 |

**Tabla III: Resultados de variar el tamaño de los segmentos**

| PC | Segmentos | xRT | TejecE$_1$ ($T_{P1}$) | TejecE$_2$ ($T_{P2}$) | TejecE$_2$ ($T_{P2}$) | CR |
|---|---|---|---|---|---|---|
| **1** | 1024 | 0.58 % | 20745 ms (99.06 %) | 282 µs (0.54 %) | 116 µs (0.40 %) | 6.17 |
| | 2048 | 0.58 % | 20928 ms (99.04 %) | 168 µs (0.56 %) | 539 ns (0.39 %) | 6.33 |
| | 4096 | 0.58 % | 20893 ms (99.01 %) | 922 µs (0.60 %) | 355 ns (0.39 %) | 6.46 |
| **2** | 1024 | 1.33 % | 47723 ms (99.16 %) | 311 µs (0.43 %) | 80 ns (0.41 %) | 6.17 |
| | 2048 | 1.32 % | 47565 ms (99.17 %) | 253 µs (0.43 %) | 720 ns (0.40 %) | 6.33 |
| | 4096 | 1.34 % | 48128 ms (99.13 %) | 656 µs (0.45 %) | 960 ns (0.41 %) | 6.46 |
| **3** | 1024 | 0.98 % | 35177 ms (99.02 %) | 219 µs (0.54 %) | 699 ns (0.44 %) | 6.17 |
| | 2048 | 1.15 % | 41245 ms (99.02 %) | 560 µs (0.53 %) | 294 ns (0.45 %) | 6.33 |
| | 4096 | 1.24 % | 44513 ms (99.09 %) | 426 µs (0.52 %) | 7 ns (0.39 %) | 6.46 |
| **4** | 1024 | 1.86 % | 1 min 6824 ms (99.31 %) | 74 µs (0.38 %) | 9 ns (0.32 %) | 6.17 |
| | 2048 | 1.87 % | 1 min 7169 ms (99.22 %) | 546 µs (0.47 %) | 968 ns (0.31 %) | 6.33 |
| | 4096 | 1.86 % | 1 min 7052 ms (99.28 %) | 935 µs (0.42 %) | 778 ns (0.30 %) | 6.46 |
| **5** | 1024 | 1.63 % | 58624 ms (99.01%) | 41 µs (0.57 %) | 323 ns (0.41 %) | 6.17 |
| | 2048 | 1.63 % | 58619 ms (98.97%) | 662 µs (0.62 %) | 555 ns (0.41 %) | 6.33 |
| | 4096 | 1.63 % | 58667 ms (99.00%) | 519 µs (0.59 %) | 754 ns (0.40 %) | 6.46 |

**Tabla IV: Resultados de variar el valor PRD esperado**

| PC | PRD | xRT | TejecE$_1$ ($T_{P1}$) | TejecE$_2$ ($T_{P2}$) | TejecE$_2$ ($T_{P2}$) | CR |
|---|---|---|---|---|---|---|
| **1** | 2 | 0.58 % | 20725 ms | 635 µs | 547 µs | 4.24 |
| | 6 | 0.58 % | 20731 ms | 701 µs | 577 ns | 7.33 |
| | 9 | 0.58 % | 20747 ms | 225 µs | 145 ns | 9.62 |
| **2** | 2 | 1.32 % | 47454 ms | 587 µs | 75 ns | 4.24 |
| | 6 | 1.32 % | 47487 ms | 178 µs | 360 ns | 7.33 |
| | 9 | 1.32 % | 47570 ms | 86 µs | 960 ns | 9.62 |
| **3** | 2 | 0.99 % | 35585 ms | 37 µs | 708 ns | 4.24 |
| | 6 | 0.96 % | 34533 ms | 171 µs | 720 ns | 7.33 |
| | 9 | 1.16 % | 41575 ms | 697 µs | 629 ns | 9.62 |
| **4** | 2 | 1.86 % | 1 min 7146 ms | 848 µs | 552 ns | 4.24 |
| | 6 | 1.87 % | 1 min 7007 ms | 530 µs | 295 ns | 7.33 |
| | 9 | 1.86 % | 1 min 7329 ms | 537 µs | 134 ns | 9.62 |
| **5** | 2 | 1.63 % | 58910 ms | 984 µs | 268 ns | 4.24 |
| | 6 | 1.62 % | 58496 ms | 860 µs | 355 ns | 7.33 |
| | 9 | 1.64 % | 58910 ms | 483 µs | 875 ns | 9.62 |
| **6** | 2 | 30.04% | 17 min 53278 ms (99.26 %) | 2504 ms (0.23 %) | 5492 ms (0.51 %) | 4.24 |
| | 6 | 23.50% | 14 min 1458 ms (99.45 %) | 1937 ms (0.23 %) | 2712 ms ( 0.32%) | 7.33 |
| | 9 | 23.47% | 14 min 616 ms (99.49 %) | 2148 ms (0.25 %) | 2118 ms (0.25 %) | 9.62 |

consideración. En la Tabla III se muestran los resultados de ejecutar el algoritmo de compresión en las distintas plataformas de hardware, variando el tamaño de los segmentos. El incremento del tamaño de los segmentos de datos, conduce a un ligero incremento de la tasa de compresión en la primera cifra decimal significativo y no resulta dependiente de la plataforma elegida dado que la aritmética de todos los procesadores es la misma. El tiempo de la ejecución del algoritmo completo permanece invariante. Aunque, evidentemente, los tiempos de procesamiento de cada trama o segmento de datos se incrementan de manera proporcional a la longitud del segmento de datos. Es conveniente elegir duraciones de segmento que posean un buen comportamiento respecto a la compresión y que no representen grandes retardos, fundamentalmente ante la transmisión de señales en tiempo real. Un segmento de 2048 muestras, es la solución de compromiso más adecuada.

De la Tabla III resulta evidente que de los procesadores utilizados, el de mejor desempeño corresponde al procesador Core i3 que logra realizar la compresión en 0,58% del tiempo de duración de la señal procesada. Esta cifra en porciento indica que aproximadamente el programa realiza su procesamiento 172 veces en el requerido para la operación en tiempo real. La configuración con menores prestaciones logra realizar su procesamiento 53 veces en el tiempo requerido para la ejecución en tiempo real. En general, al analizar los tiempos de ejecución del método de compresión en las distintas arquitecturas de computadoras personales se puede concluir que superan el tiempo real con valores entre 53 y 172 veces aproximadamente variando de una arquitectura a otra debido a las diferentes capacidades de procesamiento.

Es relevante destacar que la operación que más tiempo demandó en todos los casos fue la descomposición en subbandas, en concordancia con el orden de complejidad de cada etapa. Esto se debe a que la implementación los filtros FIR coseno modu-

lados, se realizó en su forma directa, al realizar la convolución de las muestras a filtrar con la respuesta impulsiva de cada uno de los 32 filtros del banco de filtros. Este resultado permite concluir que una vía para reducir aún más los tiempos de procesamiento debe intentar minimizar el tiempo demandado por las operaciones de filtrado empleando estructuras eficientes y métodos óptimos para este tipo de estructura de bancos de filtros como se describe en [23].

La Tabla IV muestra los efectos de variar el PRD sobre los tiempos de ejecución del algoritmo en las distintas plataformas, incluyendo la implementación en el teléfono móvil. El segmento se ha tomado de 2048 muestras. A medida que el PRD se hace menor, los tiempos de ejecución debieran ser mayores debido a que mayor número de muestras subbandas serán diferentes de cero, y deberán ser cuantificadas y codificadas. No obstante, este efecto se encuentra ensombrecido en el tiempo de ejecución total del algoritmo, por el hecho de que la operación de filtrado ocupa más del 99% del tiempo de ejecución del algoritmo de compresión, y tal operación es independiente del PRD elegido. Si detallamos los tiempos de las operaciones de cuantificación y codificación, sí podemos inferir los incrementos del tiempo de estas etapas al emplear valores de PRD menores. No obstante, dado que los tiempos son muy pequeños, estos pueden verse afectado por los tiempos que demandan otros procesos que el sistema operativo mantiene en ejecución concurrente con la operación de compresión.

La Figura 2 muestra una instantánea como ejemplo de los resultados obtenidos cuando el programa se ejecuta en el dispositivo móvil. Al igual que aparece en la Tabla IV, la ejecución en el móvil HTC resultó ser la de peor desempeño al compararla con el resto de las plataformas elegidas. Este resultado era de esperar, dadas las características de estos procesadores móviles en comparación con las otras plataformas de computadoras. Los tiempos de ejecución suben significativamente a valores entre 14 y 18 minutos para diferentes valores de PRD. Estos tiempos de ejecución significan que, el procesamiento se puede realizar entre 3.32 y 4.26 veces aproximadamente el tiempo requerido para la operación en tiempo real. Resalta el mismo comportamiento con respecto al porciento del tiempo dedicado a cada etapa del algoritmo, siendo el banco de filtros quien más tiempo consume y que la codificación consume un tiempo mayor a medida que se intenta reducir el PRD de la señal reconstruida. Es importante señalar que en todos los casos, al algoritmo de compresión se le asigna un hilo de ejecución propio que compite con el resto de las aplicaciones del móvil, por lo que los tiempos presentados pueden incluir los tiempos de otros procesos que se ejecutan concurrentemente.

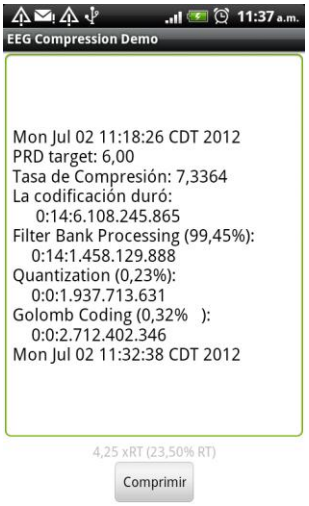


**Figura 2: Ejemplo de ejecución del algoritmo en el teléfono móvil**

## 3. CONCLUSIONES

A partir de los resultados obtenidos de los experimentos realizados, resulta conveniente destacar que el algoritmo seleccionado posee un costo computacional que permite ser ejecutado en tiempo real en plataformas de cálculos comunes en la actualidad incluyendo dispositivos portátiles de bajos recursos computacionales como los teléfonos móviles. Estos resultados permiten inferir que en otros dispositivos de hardware de bajo perfil que tengan características similares, el algoritmo podría ser empleado. En todos los casos analizados se logra superar el tiempo real de ejecución. Los tiempos de ejecución alcanzados toman valores desde 172 veces el tiempo real en la mejor de las plataformas de hardware utilizadas, hasta poco más de tres veces en la plataforma de teléfono móvil. La ejecución en la plataforma de hardware de bajo perfil presenta un desempeño adecuado, y es susceptible de mejorar si se minimizan los tiempos requeridos por la operación de filtrado. No obstante, si se pretenden comprimir múltiples canales simultáneamente con el algoritmo propuesto, este dispositivo no ofrece la posibilidad de comprimir más de tres canales sin afectar la ejecución en tiempo real.

## 4. REFERENCIAS BIBLIOGRÁFICAS

1. **Teplan, M.**: "Fundamentals of EEG measurements," *Measurement science review,* vol. 2, 2002.
2. **Casson, A. J., Yates, D. C., Smith, S. J. M., Duncan, J. S., and Rodriguez-Villegas, E.**: "Wearable Electroencephalography What Is It, Why Is It Needed, and What Does It Entail?," *Ieee Engineering in Medicine and Biology Magazine,* vol. 29, pp. 44-56, May-Jun 2010.

3. **Farajidavar, A., Seifert, J. L., Bell, J. E. S., Seo, Y. S., Delgado, M. R., Sparagana, S., Romero, M. I., and Chiao, J. C.**: "A Wireless System for Monitoring Transcranial Motor Evoked Potentials," *Annals of Biomedical Engineering,* vol. 39, pp. 517-523, Jan 2011.
4. **Garry Higgins, Stephen Faul, Robert P. McEvoy, Brian McGinley, Martin Glavin, William P. Marnane, and Jones, E.**: "EEG Compression Using JPEG2000: How Much Loss Is Too Much?," in *32nd Annual International Conference of the IEEE EMBS*, Buenos Aires, Argentina, 2010.
5. **A.Avila, Santoyo, R., and Martinez, S. O.**: "Hardware/software implementation of the EEG signal compression module for an ambulatory monitoring subsystem," in *Proceedings of the 6th International Caribbean Conference on Devices, Circuits and Systems*, Mexico, 2006.
6. **Francisco Martín González and Reyes, E. V.**: "Diseño de un Holter para Electroencefalografía y Polisomnografía," *Revista de Ingeniería Electrónica, Automática y Comunicaciones,* vol. 30, 2009.
7. **Antoniol, G. and Tonella, P.**: "EEG data compression techniques," *Ieee Transactions on Biomedical Engineering,* vol. 44, pp. 105-114, Feb 1997.
8. **Srinivasan, K. and Reddy, M. R.**: "Efficient preprocessing technique for real-time lossless EEG compression," *Electronics Letters,* vol. 46, pp. 26-U43, Jan 7 2010.
9. **Memon, N., Kong, X., and Cinkler, J.**: "Context-based lossless and near-lossless compression of EEG signals," *IEEE Trans Inf Technol Biomed,* vol. 3, pp. 231-8, Sep 1999.
10. **Sriraam, N.**: "A High-Performance Lossless Compression Scheme for EEG Signals Using Wavelet Transform and Neural Network Predictors," *International Journal of Telemedicine and Applications,* vol. 2012, 2012.
11. **Gopikrishna, D. and Makur, A.**: "A high performance scheme for EEG compression using a multichannel model," in *High Performance Computing - Hipc 2002, Proceedings*, pp. 443-451, 2002.
12. **Cárdenas-Barrera, J. L., Lorenzo-Ginori, J. V., and Rodríguez-Valdivia, E.**: "A wavelet-packets based algorithm for EEG signal compression," *Medical Informatics and the Internet in Medicine,* vol. 29, pp. 15-27, Mar 2004.
13. **Nielsen, M., Kamavuako, E. N., Andersen, M. M., Lucas, M. F., and Farina, D.**: "Optimal wavelets for biomedical signal compression," *Medical & Biological Engineering & Computing,* vol. 44, pp. 561-568, Jul 2006.
14. **Gurkan, H., Guz, U., and Yarman, B. S.**: "EEG signal compression based on classified signature and envelope vector sets," *International Journal of Circuit Theory and Applications,* vol. 37, pp. 351-363, Mar 2009.
15. **Poh, K. K. and Marziliano, P.**: "Compressive Sampling of EEG Signals with Finite Rate of Innovation," *Eurasip Journal on Advances in Signal Processing,* 2010.
16. **Bazán-Prieto, C., Blanco-Velasco, M., Cárdenas-Barrera, J., and Cruz-Roldán, F.**: "Compresión de Señales EEG Basada en Descomposición en Subbandas y Codificación de Golomb," in *V Latin American Congress on Biomedical Engineering CLAIB 2011*, Habana, Cuba, 2011.
17. **Rice, R.**: "Adaptive Variable-Length Coding for Efficient Compression of Spacecraft Television Data," *IEEE Transactions on Communication Technology,* vol. 19, pp. 889 - 897, 1971.
18. **Golomb, S. W.**: "Run-length encodings," *IEEE Transactions on Information Theory,* vol. 12, pp. 399-401, 1966.
19. **Seroussi, G. and Weinberger, M. J.**: "Optimal prefix codes for sources with two-sided geometric distributions," *IIEEE Transactions on nformation Theory,* vol. 46, pp. 121-135, 2000.
20. **Malvar, H. S.**: "Adaptive Run-Length / Golomb-Rice Encoding of Quantized Generalized Gaussian Sources with Unknown Statistics," in *Data Compression Conference, 2006. DCC 2006. Proceedings* pp. 23-32, 2006.
21. **Sanei, S. and Chambers, J.**: *EEG signal processing*. Chichester, England ; Hoboken, NJ: John Wiley & Sons, 2007.
22. **Sijertit, Z. and Agarwd, G.**: "Tree Structured Filter Bank for Time-Frequency Decomposition of EEG signals," in *IE13E-EMBC and CMBEC Theme 41: Signal Processing*, pp. 991-992, 1995.
23. **Fernando Cruz-Roldán and Monteagudo-Prim, M.**: "Efficient Implementation of Nearly Perfect Reconstruction FIR Cosine-Modulated Filterbanks," *IEEE TRANSACTIONS ON SIGNAL PROCESSING,* vol. 52, 2004.
24. **Bazán-Prieto, C., Blanco-Velasco, M., Cárdenas-Barrera, J., and Cruz-Roldán, F.**: "Retained energy-based coding for EEG signals," *Medical Engineering & Physics,* vol. 34, pp. 892-899, 2012.
25. **Sullivan, G. J. and Sun, S.**: "On Dead-Zone Plus Uniform Threshold Scalar Quantization," *Visual Communications and Image Processing,* vol. 5960, 2005.
26. **Goldberger, A. L., Amaral, L. A. N., Glass, L.,**

**Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C. K., and Stanley, H. E.**: "PhysioBank, PhysioToolkit, and PhysioNet - Components of a new research resource for complex physiologic signals," *Circulation,* vol. 101, pp. E215-E220, Jun 13 2000.

## 5. SÍNTESIS CURRICULARES DE LOS AUTORES

**Carlos Alberto Bazán Prieto:** Ingeniero Electrónico. Máster en Electrónica. Profesor Auxiliar. Departamento de Electrónica y Telecomunicaciones. Facultad de Ingeniería Eléctrica. Universidad Central ¨Marta Abreu¨ de Las Villas. Santa Clara, Cuba. cabazan@uclv.edu.cu

**Julián Luciano Cárdenas Barrera:** Ingeniero Electrónico. Doctor en Ciencias Técnicas. Profesor Titular. Centro de Estudios de Electrónica y Tecnologías de la Información (CEETI). Facultad de Ingeniería Eléctrica. Universidad Central ¨Marta Abreu¨ de Las Villas. Santa Clara, Cuba.