# Implementación en sistemas embebidos de la transformada discreta de Hartley

Juan S. Botero-Valencia[1],
Edilson Delgado-Trejos[2]

**Resumen:**

Las transformaciones ortogonales han sido de gran utilidad en la caracterización y procesamiento de señales. En particular la Transformada de Hartley permite obtener representaciones tiempo-frecuencia y viceversa. En este trabajo se presenta un algoritmo para el cálculo de la Transformada Discreta de Hartley en sistemas embebidos con el objetivo de minimizar la carga computacional y la capacidad de almacenamiento necesaria. Se aprovecha la similitud con la Transformada Discreta de Fourier para usar un algoritmo de cálculo rápido y se reduce el número de funciones trigonométricas calculadas usando los factores de giro (twiddle factors). En general la implementación permite aumentar el tamaño de la ventana de transformación y aumentar la velocidad de cálculo respecto al cálculo directo.

**Palabras clave:**

Transformada Discreta de Hartley, sistemas embebidos, decimación en frecuencia, factores de giro.

1 **Instituto Tecnológico Metropolitano**, Centro de Investigación, juanbotero@itm.edu.co

2 **Instituto Tecnológico Metropolitano**, Centro de Investigación, edilsondelgado@itm.edu.co

Fecha de recepción: 03 de abril de 2010
Fecha de aceptación: 25 de junio de 2010

**Abstract:**

Orthogonal transformations have been very useful in the characterization and signal processing. In particular Hartley Transform allows for time-frequency representations and vice versa. This paper presents an algorithm for calculating the Discrete Hartley Transform in embedded systems with the objective of minimizing the computational load and storage capacity required. It exploits the similarity with the Discrete Fourier Transform to use a fast algorithm reduces the number of trigonometric functions calculated using the rotation factors (Twiddle factors). In general, the implementation can increase the size of the processing window and increase computational speed compared to direct calculation.

**Keywords:**

Discrete Hartley Transform, embedded systems, decimation in frequency, twiddle factors.

## 1. Introducción

Las transformaciones tiempo frecuencia son de gran utilidad en al análisis y procesamiento de señales (Manolakis & Proakis, 1998), en la compresión de datos, el análisis de sistemas de control (Kuo, 1996) y en el reconocimiento de patrones usando las transformaciones como extractor de características. En particular, la Transformada de Fourier ha sido usada por muchos años con gran éxito dado el esfuerzo realizado para mejorar la velocidad de cálculo con algoritmos de decimación y su efectividad para representar señales multidimensionales en el dominio de la frecuencia. Sin embargo, la Transformada de Fourier considera la posibilidad de que el espacio de entrada sea complejo al igual que el espacio de salida, lo que conlleva el uso de tipos especiales de variables cuando se realizan cálculos.

La Transformada de Hartley descrita en (Bracewell, 1985; Grigoryan, 2004), presenta una forma de obtener representaciones tiempo frecuencia de señales reales en un espacio real, es decir, es una trasformación ortogonal de R->R. Este tipo de representación facilita los cálculos, y permite reducir la capacidad de almacenamiento necesaria en el cálculo. En el presente documento se presenta la implementación de un algoritmo de optimización de cálculo de la Transformada Discreta de Hartley (Chu & George, 2000; Shah & Rathore, 2009) con el objetivo de ser implementada en sistemas embebidos para obtener características espectrales de señales biológicas (Prutchi & Norris, 2005). Se compara la efectividad del uso del algoritmo de decimación en frecuencia sobre la implementación directa de la transformada, y algunos ejemplos sobre la efectividad de la transformada para obtener representaciones espectrales.

## 2. Metodología

De acuerdo a la definición presentada en (Bracewell, 1985) de una secuencia de datos dada se puede obtener un vector

transformado usando la TDH (Transformada Discreta de Hartley), la transformación se calcula usando (1).

$$y_k(N,x) = \sum_{n=0}^{N-1} x_n \left[\cos\left(\frac{2\pi kn}{N}\right) + \sin\left(\frac{2\pi kn}{N}\right)\right], \qquad k = 0, \dots, N-1 \quad (1)$$

Expresada de forma matricial, se puede ver en (2).

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} cas(0) & cas(0) & \dots & cas(0) \\ cas(0) & cas\left(\frac{2\pi}{N}\right) & \dots & cas\left(\frac{2\pi(N-1)}{N}\right) \\ \vdots & \vdots & \ddots & \vdots \\ cas(0) & cas\left(\frac{2\pi(N-1)}{N}\right) & \dots & cas\left(\frac{2\pi(N-1)(N-1)}{N}\right) \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} \quad (2)$$

$$cas(x) = sin(x) + cos(x)$$

$$cas(0) = 1$$

## 2.1 Cálculo directo de la TDH

En (2) se observa que los vectores fila de la matriz de transformación se pueden generar a partir de la segunda fila. Esto con el objetivo de realizar el cálculo de la función veces y no veces que serían necesarias si se realiza el cálculo de la matriz de forma directa. En (3) se muestra el vector almacenado para computar la TDH usando el algoritmo propuesto.

$$V = \left[cas(0) \quad cas\left(\frac{2\pi}{N}\right) \quad \dots \quad cas\left(\frac{2\pi n}{N}\right)\right] \quad (3)$$

Si se enumeran las posiciones en (3) entre , se puede obtener todos los valores que componen la matriz conociendo la variación de frecuencia escalar que se presenta entre filas. En (4) se muestra la construcción de la matriz de transformación a partir del vector fundamental, nótese que los valores en la matriz son posiciones del vector descrito en (3).

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} V(0) & V(0) & \dots & V(0) \\ V(0) & V(1) & \dots & V(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ V(0) & V\left(res\left(\frac{N-1}{N}\right)\right) & \dots & V\left(res\left(\frac{k(N-1)}{N}\right)\right) \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} \quad (4)$$

Usando (4) se puede calcular la TDH de forma directa como se muestra en el Algoritmo 1. La expresión en (4) indica la operación residuo del cociente indicado.

Se inicializa de acuerdo a la ecuación presentada en (3).

```
begin
for K=0:N-1
for I=0:N-1
    y[k]=V(res(K*I,N))*x[i]+y[k]
end
```

Algoritmo 1. Cálculo directo de la TDH

## 2.2 Cálculo rápido de la TDH

Dada la similitud entre el TDH y la TDF (Transformada Discreta de Fourier) se puede analizar su relación, la forma de obtener la TDH a partir de la TDF se presenta en (5)

$$y_k(N,x) = RE(z_k(N,x)) - IM(z_k(N,x))$$
$$z_k(N,x) = \sum_{n=0}^{N-1} x_n \left[\cos\left(\frac{2\pi kn}{N}\right) + \mathrm{j}\sin\left(\frac{2\pi kn}{N}\right)\right], \qquad k = 0, \dots, N-1 \quad (5)$$

Usando el algoritmo que se presenta en (Chu & George, 2000), se puede obtener la TDF, se presenta en el Algoritmo 2. El Algoritmo 2 calcula la FFT (Fast Fourier Transform, por sus siglas en Ingles), nótese que se usan los factores de giro calculados en (3) y que los vectores son estructuras para simular las operaciones internas de la FFT.

```
Se inicializa  de acuerdo a la ecuación presentada en (3).
begin
NumOfProblems := 1
ProblemSize := N
while ProblemSize > 1 do
HalfSize := ProblemSize/2
for K := 0 to NumOfProblems − 1 do
JFirst := K * ProblemSize
JLast := JFirst + HalfSize − 1
Jtwiddle := 0
for J := JFirst to JLast do
W := w[Jtwiddle]
Temp := a[J]
a[J] := Temp + a[J + HalfSize]
a[J + HalfSize] := W (Temp − a[J + HalfSize])
Jtwiddle := Jtwiddle + NumOfProblems
end for
end for
NumOfProblems := 2*NumOfProblems
ProblemSize := HalfSize
end while
for J := 0 to N/2 do
        a[J]= Re(a[J])-Im(a[J])
end for
end
```

Algoritmo 2. Cálculo rápido de la TDH

## 3. Resultados y discusión

Para probar el desempeño de los Algoritmos en las diferentes arquitecturas de la familia Microchip, se inyecto una señal senoidal pura de 4 Vpp con un offset de 2 V (el conversor de los microcontroladores no soporta voltajes negativos) se eliminó el offset internamente y se vario la frecuencia. Se uso un bit como disparo para generar un pulso cuadrado y así medir el tiempo total de procesamiento de los algoritmos. Los datos se exportan al finalizar el algoritmo en un vector enviado por el puerto serial.

Para fines de representación las figuras presentadas se adecuaron usando la operación mostrada en (6).

$$S = \left[\frac{1}{N}\left[y_0 \quad y_1 \quad \cdots \quad y_{\frac{N}{2}}\right]\right]^2, \qquad k = 0, \ldots, N-1 \tag{6}$$

En la Tabla 1, se presenta el tiempo de cálculo empleado para obtener la Transformada Discreta de Hartley usando el a Algoritmo 1 (A1), y el Algoritmo 2 (A2), es clara la reducción en el tiempo de cálculo al usar el algoritmo de decimación en frecuencia. Si se tiene en cuenta una ventana típica con  para el tratamiento de señales SEMG (Electromiografía de Superficie), este cálculo se podría realizar en menos de 82 mS (teniendo en cuenta el uso del Algoritmo 2 (A2) y la ventana con ) lo que permitiría implementar en sistemas embebidos algoritmos de reconocimiento como (Kuruganti et al., 1995). La Tabla 2 muestra los resultados para ventanas de mayor tamaño.

Tabla 1. Costo en tiempo (s)

| | N | | | | | |
|---|---|---|---|---|---|---|
| **Micro** | **4** | | **8** | | **16** | |
| | **A1** | **A2** | **A1** | **A2** | **A1** | **A2** |
| PIC16 20MHz | 0,00891 | 0,00812 | 0,02241 | 0,01822 | 0,06101 | 0,04062 |
| PIC18 20MHz | 0,00915 | 0,00907 | 0,02345 | 0,01917 | 0,06401 | 0,04115 |
| PIC18 48MHz | 0,00442 | 0,00397 | 0,01165 | 0,00941 | 0,03059 | 0,02040 |

Tabla 2. Costo en tiempo (s)

| | N | | | | | |
|---|---|---|---|---|---|---|
| **Micro** | **32** | | **64** | | **128** | |
| | **A1** | **A2** | **A1** | **A2** | **A1** | **A2** |
| PIC16 20MHz | 0,1620 | 0,08521 | 0,4925 | 0,18622 | 1,9226 | 0,4256 |
| PIC18 20MHz | 0,1715 | 0,09112 | 0,5181 | 0,19215 | 2,1221 | 0,4415 |
| PIC18 48MHz | 0,08396 | 0,04342 | 0,24889 | 0,08150 | 0,94536 | 0,1889 |

En la Fig. 1 se presenta la salida obtenida de la trasformada de Hartley a una señal de entrada senoidal pura con , para la Fig. 2 la señal senoidal es de , es claro que al duplicar la frecuencia de muestreo, mejora la resolución de la transformada.

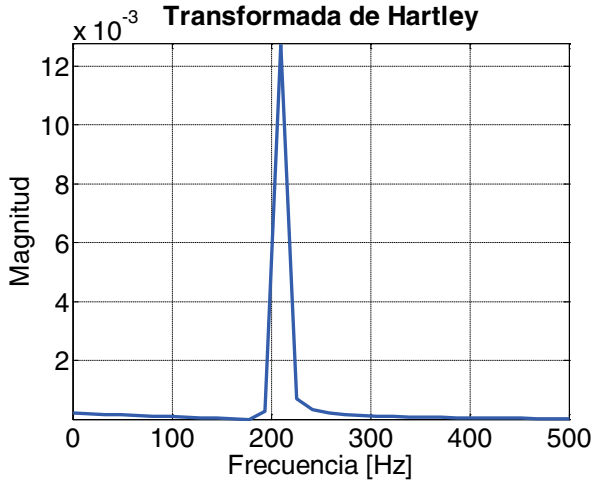


FIG. 1. $F_s$ = 1000 Hz, $F$ = 200 Hz, $N$ = 64

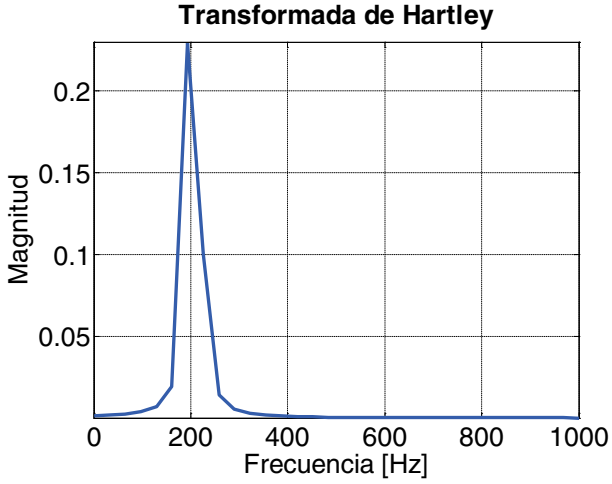


FIG. 2. $F_s$ = 2000 Hz, $F$ = 200 Hz, $N$ = 64

En la Fig. 3 se presenta la salida obtenida de la transformada de Hartley a una señal de entrada senoidal pura con , para la Fig. 4 la señal senoidal es de , es claro que al cuadruplicar la ventana de análisis, mejora la representación espectral de la transformada.

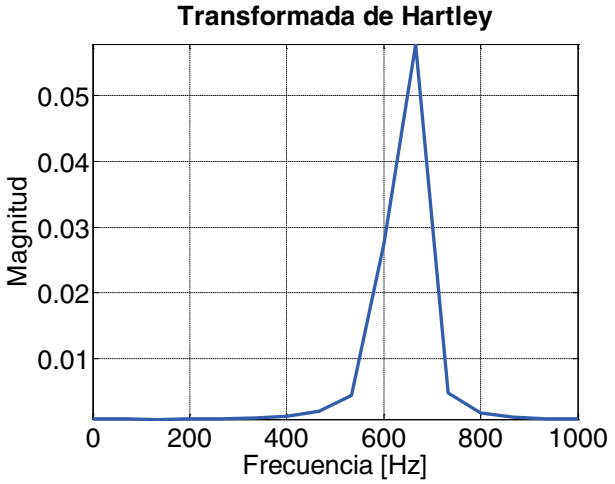


FIG. 3. $F_s$ = 2000 Hz, $F$ = 600 Hz, $N$ = 32

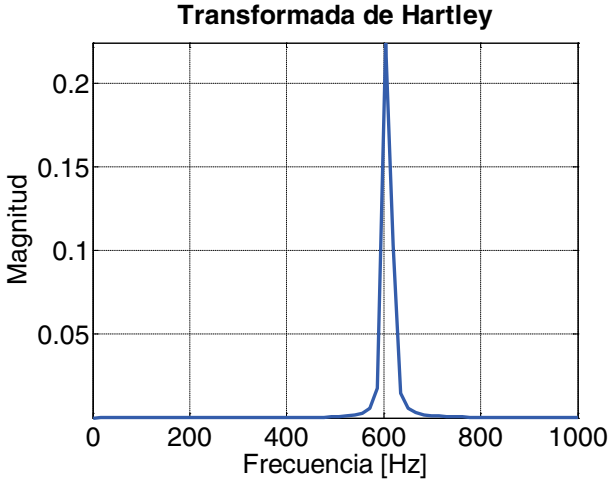


FIG. 4. $F_s$ = 2000 Hz, $F$ = 600 Hz, $N$ = 128

## 4. Conclusiones

La Transformada Discreta de Hartley, puede transformar con efectividad un espacio temporal en uno frecuencial con eficiencia usando algoritmos de decimación en frecuencia sobre plataformas de bajo procesamiento. Así, el método estudiado expone ventajas en cuanto a la velocidad de cálculo, además de permitir aumentar el tamaño de la ventana de análisis por requerir menor uso de memoria de datos. Dado que la arquitectura del algoritmo es fácilmente adaptable se pueden desarrollar aplicaciones de reconocimiento de patrones o de filtrado que se adecuen en línea sin comprometer el costo computacional asociado a las rutinas del procedimiento. La Transformación puede ser implementada en aplicaciones reales, sobre plataformas de baja exigencia de procesamiento.

## 5. Agradecimientos

Este trabajo está enmarcado en labores de investigación del Grupo MIRP, del Centro de Investigación del **INSTITUTO TECNOLÓGICO METROPOLITANO**, se desarrolla como parte de la Tesis "Representación ortogonal de información tiempo-frecuencia de señales SEMG orientado a la detección de flexión de los dedos para HMI".

## 6. Referencias

Avendaño, L.E., (2007); Sistemas Electrónicos Analógicos Un Enfoque Matricial (Primera Edición ed.). Pereira, Colombia: Universidad Tecnológica de Pereira.

Betancourt, G.A., Giraldo, E., Franco, J.F., (2004); Reconocimiento de patrones de movimiento a partir de señales electromiográficas. Scientia et Technica, 53-58.

Botero-Valencia, J.S., Sánchez-Giraldo, L.G., Delgado-Trejos, E., (2009); Clasificador no lineal basado en redes neuronales con funciones de base radial para implementación en sistemas de punto fijo, Tecnológicas, (22), 11-28.

Bracewell, R.N., (1985); The Hartley Transform. Oxford University Press.

Caballero, K., Duque, L.M., Ceballos, S., Ramírez, J.C., Peláez, A., (2002); Conceptos básicos para el análisis electromiográfico. Revista CES Odontología, 41-50.

Chu, E., George, A., (2000); Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform. CRC Press.

García, E., (2008); Compilador C CCS y simulador PROTEUS para microcontroladores PIC (Primera edición ed.). México, México: Alfaomega.

Grigoryan, A.M., (2004); A Novel Algorithm for Computing the 1-D Discrete Hartley Transform. IEEE Signal Processing Letters, 156-159.

Kuo, B.C., (1996); Sistemas de Control Automático (Septima edición ed.). México, México: Prentice Hall.

Kuruganti, U., Hudgins, B., Scott, R.N., (1995); Two-Channel Enhancement of a Multifunction Control System. IEEE Transactions On Biomedical Engineering, 109-111.

Lopez-Rincon, A., Vazquez-Gonzalez, J., Aguilera-Fernández, A., Navarro-Martinez, J., Escudero, A.Z., (2008); Fast Fourier Transform Adjusted into 8 bit Format for Instrumentation Purposes. 18th International Conference on Electronics, 95-98.

Manolakis, D.G., Proakis, J., (1998); Digital Signal Processing. Prentice Hall.

Nait-Ali, A., (2009); Advanced Biosignal Processing. Berlín: Springer.

Prutchi, D., Norris, M., (2005); Design and development of medical electronic instrumentation. New Jersey: John Wiley & Sons.

Shah, G.A., Rathore, T.S., (2009); A New Fast Radix-2 Decimation-In-Frequency Algorithm for Computing the Discrete Hartley Transform. First International Conference on Computational Intelligence (págs. 363-368). Indore: IEEE.