# DISEÑO DE UN FILTRO DIGITAL PASA BAJAS DE PRIMER Y SEGUNDO ORDEN A PARTIR DE CIRCUITO RC

***Armando S. Chipule Pérez***
Tecnológico Nacional de México, Instituto Tecnológico de Orizaba
*satielo.3000@gmail.com*

***Gerardo Águila Rodríguez***
Tecnológico Nacional de México, Instituto Tecnológico de Orizaba
*gerardo_aguila03@yahoo.com.mx*

***Rubén Posada Gómez***
Tecnológico Nacional de México, Instituto Tecnológico de Orizaba
*rposada@itorizaba.edu.mx*

## Resumen

En este artículo se presenta una metodología para realizar modelos matemáticos de filtros pasa baja de primero y segundo orden en tiempo discreto partiendo de la función transferencia de un circuito simple RC. Entre las ventajas de utilizar técnicas de procesamiento digital de señales resaltan la inmunidad a factores intrínsecos tales como temperatura ambiente, ruido electromagnético y acoplamiento entre componentes. De esta forma los sistemas de procesamiento de señales involucrados en lazos de instrumentación incrementan la estabilidad y robustez. En concreto este trabajo se centra en el diseño de filtros digitales pasa bajas que emulen el comportamiento de un circuito RC, así también se presenta la forma de implementar el C++ dicho filtro, estableciendo como parámetros principales la frecuencia de corte y el periodo de muestreo para su posterior aplicación en lazos de instrumentación.

**Palabra(s) Clave(s):** Digital, filtro, microcontroladores, pasa-bajas.

## 1. Introducción

Un filtro por definición es un procedimiento utilizado para seleccionar cierta información y descartar otra, en procesamiento de señales se utilizan normalmente para, dependiendo de sus parámetros, llevar a cabo un proceso de discriminación en la señal de entrada, obteniendo una señal de salida con la información que ha dejado pasar [1], usados normalmente para eliminar ruido eléctrico, oscilaciones en la señal, o bien niveles de DC que no se requieren.

Los filtros digitales han ganado popularidad debido al amplio uso que se le ha dado a los microcontroladores, ya que el procesamiento digital de señales es una parte importante de todo sistema embebido. Existen dos grandes divisiones en cuanto a filtros digitales se refiere, los filtros de respuesta infinita al impulso (IIR por sus siglas en inglés) y los filtros de respuesta finita al impulso (FIR por sus siglas en inglés). Los filtros IIR en general son, más rápidos que los FIR y con menos elementos pero, ya que presentan retroalimentaciones, pueden llegar a ser inestables por lo que deben ser sometidos a análisis de estabilidad, los FIR por el contrario no presentan retroalimentación y por lo tanto son estables por definición [2], por el contrario presentan mayor complejidad computacionalmente hablando de los IIR por lo que existen una gran cantidad de métodos propuestos para reducir su complejidad [3].

Diferentes métodos son utilizados para obtener los modelos discretizados de funciones conocidas en tiempo continuo, como en [4], que plantea una metodología basada en utilizar la transformada Z bilineal para determinar ciertos coeficientes, y determinar la matriz de pascal de acuerdo al orden del filtro, este método es fácilmente programable, pero ya que implica utilización de operaciones matriciales resulta de difícil implementación en microcontroladores, ya que se deberían desarrollar todas las funciones para resolver operaciones matriciales o importar librerías que lo hagan; así mismo el procedimiento resulta bastante extenso para ser resuelto a mano.

La transformada Z es una herramienta útil en el procesamiento de señales digitales, ya que es posible realizar modelos matemáticos que representen el comportamiento de una señal que se encuentra en tiempo continuo, pero

discretizada, así mismo es posible realizar, mediante el análisis correcto, modelados de funciones transferencia que pueden ser implementadas mediante un microcontrolador, teniendo así un resultado muy similar al que se tendría con el mismo sistema hecho con dispositivos analógicos.

## 2. Desarrollo

### Filtro de Primer Orden

Dependiendo el tipo de señal que se tenga y lo que se desee tener, es como se elige el tipo de filtro a utilizar, ya sea un filtro pasa bajas, pasa altas, pasa banda, entre otros. En el caso de señales en las cuales existen componentes harmónicas de determinada frecuencia que se desean eliminar para tener solo la componente de DC (tratándose de una señal en voltaje), lo ideal es hacerla pasar por un filtro pasa bajas.

Dentro de la gran variedad de configuraciones existentes para crear un filtro pasa bajas con dispositivos analógicos para una señal en voltaje, el más elemental es el circuito RC que se muestra (figura 1), se puede obtener un modelo matemático de este circuito bajo el concepto de impedancias complejas, obteniendo una función transferencia que da información acerca de la respuesta en frecuencia de dicho circuito, esto en el dominio de Laplace; sin embargo cuando se trata de una señal digital la estrategia a utilizar es diferente.

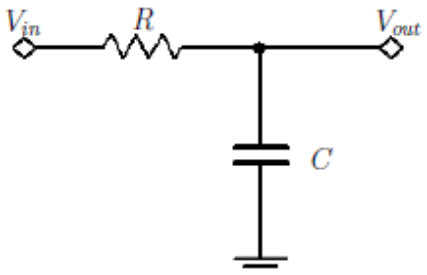



Figura 1 Filtro pasa bajas de primer orden analógico.

Existe una gran variedad de técnicas utilizadas para procesar señales digitales que no se estabilizan en ningún momento del tiempo, es decir que se encuentran oscilando constantemente, inclusive a muy baja frecuencia, en ese caso un simple

promediado de las muestras en dicha señal no es suficiente, a menos que se tomen suficientes muestras como para abarcar más de un ciclo de la señal, lo cual haría que el programa que se está ejecutando se vuelva muy lento, todo esto cambia si se utiliza un filtro de tiempo discreto por transformada Z.

Se puede realizar un modelado en el dominio Z partiendo de la función transferencia en Laplace de cualquier circuito analógico, como ya se dijo anteriormente, la transformada Z representa señales discretas, por lo que para poder aplicar la transformación primero se debe obtener una señal relacionada con dicho sistema, en este caso su respuesta a una entrada conocida, como puede ser una entrada de tipo escalón. Una vez aplicando la transformada Z a la señal de respuesta se puede obtener un cociente $\frac{V_o(z)}{V_i(z)}$, es decir una función transferencia, que en este caso, al estar en el plano Z, se denomina función transferencia pulso.

La función transferencia en el dominio de Laplace de un circuito pasa bajas de primer orden se muestra en la ecuación 1.

$$\frac{V_o(Z)}{V_i(Z)} = \frac{1}{RCs+1} = \frac{\frac{1}{RC}}{s+\frac{1}{RC}} = \frac{\omega}{s+\omega} \tag{1}$$

Ahora para partir del modelo en Laplace al modelo en Z se debe, primeramente, obtener una ecuación de respuesta del sistema a una entrada conocida, como puede ser una entrada de tipo escalón unitario, por lo que la función de respuesta en Laplace queda ecuación 2.

$$V_o(S) = \left(\frac{\omega}{s+\omega}\right)\left(\frac{1}{s}\right) = \frac{1}{s} - \frac{1}{s+\omega} \tag{2}$$

La ecuación 2 es una función que presenta la respuesta del filtro de primer orden analógico, a una entrada de tipo escalón, sobre la cual se puede aplicar el proceso de transformada Z mediante la propiedad de linealidad. La transformada Z de dicha función queda expresada como ecuación 3.

$$V_o(Z) = \mathcal{Z}\left\{\frac{1}{s}\right\} - \mathcal{Z}\left\{\frac{1}{s+\omega}\right\} = \frac{z}{z-1} - \frac{z}{z-e^{-\omega T}} \tag{3}$$

Para obtener el cociente $\frac{V_o(z)}{V_i(z)}$ se factoriza de la ecuación 3, la transformada Z del escalón unitario usado como entrada, por lo tanto el cociente queda expresado como ecuación 4.

$$\frac{V_o(Z)}{V_i(Z)} = \frac{1 - e^{-\omega T}}{z - e^{-\omega T}} \tag{4}$$

La ecuación 4 es la función transferencia pulso, correspondiente al filtro pasa bajas de primer orden, pero en este caso es de tipo digital. Por comodidad para el proceso siguiente, se expresará en forma de filtro digital, es decir, con exponentes negativos en la variable , ecuación 5..

$$\frac{V_o(Z)}{V_i(Z)} = \frac{(1 - e^{-\omega T})z^{-1}}{1 - e^{-\omega T}z^{-1}} \tag{5}$$

Es bien sabido que en la notación de transformada Z, $Z^{-n}$ indica retrasos en el tiempo equivalentes a Tn en donde T es el periodo de muestreo, por esta razón se expresa la ecuación 5, ya que de esta manera es más fácil observar su comportamiento en el tiempo, figura 2.

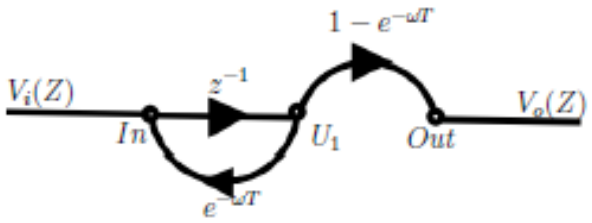



Figura 2 Gráfico de flujo de señal para el filtro pasa bajas de primer orden modelado.

Teniendo la función transferencia pulso del filtro se procede a aplicar un proceso de transformada Z inversa. Existen muchos métodos que se utilizan dependiendo la aplicación. El método utiliza la ecuación de Mason, que normalmente sirve para calcular la función transferencia de diagramas a bloques, representándolo como diagrama de flujo de señal, pero en este caso se realizará un proceso inverso ya que se parte de la función transferencia para llegar al diagrama de flujo de señal.

Ya que la función transferencia se encuentra en función de la variable Z y que, como se mencionó anteriormente, el exponente de esta indica retrasos en el

tiempo iguales a n periodos de muestreo, el diagrama de flujo de señal representará esos retrasos en el tiempo, así también con base en la ecuación de Mason[5] se puede determinar que existe una trayectoria directa entre la entrada y la salida: $(1-e^{-\omega T})z^{-1}$, así como también un lazo cerrado: $e^{-\omega T}z^{-1}$. Con estos datos se puede proceder a crear un diagrama de flujo de señal que represente el comportamiento en el tiempo del sistema diseñado.

En el diagrama de flujo de señal correspondiente al filtro diseñado (figura 2), cada lazo tiene una ganancia definida por la función transferencia pulso, además se indican un conjunto de nodos que servirán para determinar ecuaciones necesarias en la implementación del sistema.

Se deben determinar ecuaciones para los nodos de entrada y salida, tomándolos como puntos de suma, y solamente los lazos que llegan a estos y el nodo del que parten dichos lazos.

El nodo de entrada queda expresado por ecuación 6.

$$In = V_i(Z) + e^{-\omega T}U_1 \tag{6}$$

Mientras que el nodo de salida se expresa como ecuación 7.

$$Out = (1-e^{-\omega T})U_1 \tag{7}$$

Para la implementación de las ecuaciones 6 y 7 se debe tomar en cuenta que cada trazo marcado con una $z^{-1}$ indica un retraso en el tiempo de un periodo de muestreo, por lo cual la señal se ira desplazando entre los nodos unidos por un $z^{-1}$, en otras palabras, en cada iteración se calcula un nuevo valor para el nodo de entrada con la señal muestreada ($V_i(Z)$), se calcula el valor para la salida, se realiza el corrimiento de valores entre los nodos y se vuelve a iniciar el proceso en el siguiente muestreo.

**Filtro de Segundo Orden**

Para el diseño de un filtro de segundo orden hay varias alternativas en cuanto a la metodología a seguir, las cuales pueden ser:

- Realizar todo el proceso descrito para el filtro de primer orden, sobre la función transferencia de un filtro de segundo orden.

- Tomar la función transferencia de un filtro de primer orden y colocar dos iguales en cascada, obtener la función transferencia resultante y realizar el proceso ya descrito.
- Tomar la función transferencia pulso del filtro de primer orden, colocar dos en cascada y realizar el proceso de transformada Z inversa sobre la función transferencia pulso resultante, con el diagrama de flujo de señal.
- Repetir las ecuaciones de entrada/salida y hacer que la salida del primer par sea la entrada del segundo.

Como es bien sabido, tanto en el dominio de Laplace como en el dominio Z, se puede poner sistemas en cascada, lo cual matemáticamente resulta en la multiplicación de sus funciones transferencia, todo esto es válido si ambos sistemas se encuentran modelados en el mismo dominio, es decir se pueden poner varias funciones transferencia en cascada, así como también varias funciones transferencia pulso en cascada, siempre y cuando se respete el dominio en el que se encuentran. Con base en lo anterior y sabiendo que ya se modeló anteriormente la función transferencia pulso de un filtro de primer orden, se opta por tomar dicha función, colocar dos en cascada y proseguir con la metodología descrita a partir de ahí.

Partiendo de la ecuación 5, se procede a colocar dos iguales en cascada, es decir, multiplicando entre si, o bien, elevando al cuadrado dicha ecuación.

$$\frac{V_o(Z)}{V_i(Z)} = \left(\frac{(1-e^{-\omega T})z^{-1}}{1-e^{-\omega T}z^{-1}}\right)\left(\frac{(1-e^{-\omega T})z^{-1}}{1-e^{-\omega T}z^{-1}}\right) \tag{8}$$

De esta manera se vuelve más sencillo llegar a la función transferencia pulso del filtro de segundo orden, resolviendo la ecuación 8m quedando como se muestra en la ecuación 9.

$$\frac{V_o(Z)}{V_i(Z)} = \frac{(1-2e^{-\omega T}+e^{-2\omega T})z^{-2}}{1-2e^{-\omega T}z^{-1}+e^{-2\omega T}z^{-2}} \tag{9}$$

Como resultado de este proceso se tiene la función transferencia pulso para un filtro de segundo orden, y se procede a dibujar su diagrama de flujo de señal (figura 3).

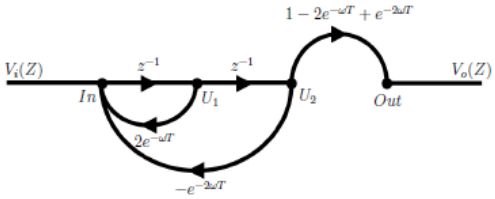



Figura 3 Diagrama de flujo de señal para el filtro pasa bajas de segundo orden.

Teniendo el diagrama de flujo de señal se pueden obtener las respectivas ecuaciones en entrada/salida para este nuevo sistema.

La entrada queda expresada por ecuación 10.

$$In = 2e^{-\omega T}U_1 - e^{-2\omega T}U_2 \tag{10}$$

Para la salida se tiene ecuación 11.

$$Out = (1 - 2e^{-\omega T} + e^{-2\omega T})U_2 \tag{11}$$

La implementación es la misma que la descrita en el caso anterior, en cada iteración se deberán hacer corrimientos consecutivos de los valores del nodo In, el nodo U1 y el nodo U2 de modo que se comporte de la misma manera que como se describe en el diagrama de flujo de señal, existiendo un periodo de muestreo en atraso entre cada uno de dichos nodos.

## 3. Resultados

### Simulación

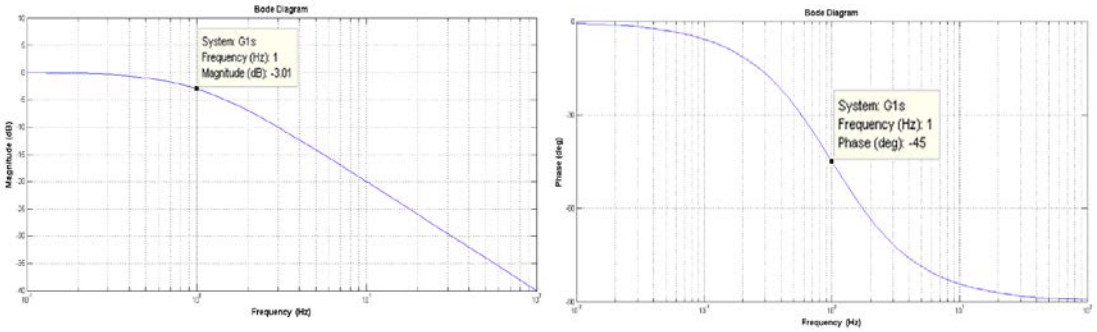
Una forma de comprobar que el filtro funcionará correctamente puede ser observando su respuesta en frecuencia, a través de un diagrama de bode, y comparar el diagrama correspondiente a la función transferencia, con el correspondiente a la función transferencia pulso, para esto primeramente habría que proponer valores de frecuencia de corte, y en el caso del filtro de tiempo discreto también habría que incluir un valor para el periodo de muestreo (T), primeramente proponiendo valores para $\omega$ y T.

Definiendo f=1hz y T=0.001s, se puede calcular el valor correspondiente de $\omega = 6.2832$ los cuales pueden ser sustituidos en la función transferencia y función transferencia pulso quedando ambas mediante ecuaciones 12 y 13.

$$G1(s) = \frac{6.283}{s + 6.283} \tag{12}$$

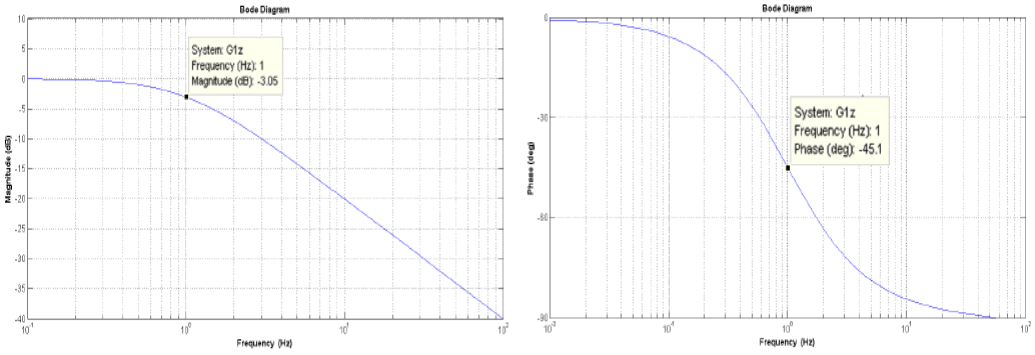
$$G1(z) = \frac{0.006263z^{-1}}{1 - 0.9937z^{-1}} \tag{13}$$

Todo filtro pasa-bajas de primer orden en la frecuencia de corte presenta una magnitud de -3dB (figura 4a) y una fase de -45° (figura 4b); además una pendiente de atenuación de -20dB/Dec debida al polo simple existente en la función transferencia como se aprecia en la ecuación 12, de la misma manera para la función transferencia pulso equivalente mostrada en la ecuación 13, en la frecuencia de corte se presenta la magnitud antes mencionada (figura 5a) con la fase antes mencionada (figura 5b), que en el mismo sistema pero en tiempo continuo.




(a) Magnitud del filtro. (b) Fase del filtro.

Figura 4 Bode del filtro de primer orden en tiempo continuo con frecuencia de corte f=1hz.




(a) Magnitud de la función transferencia. (b) Fase de la función transferencia..

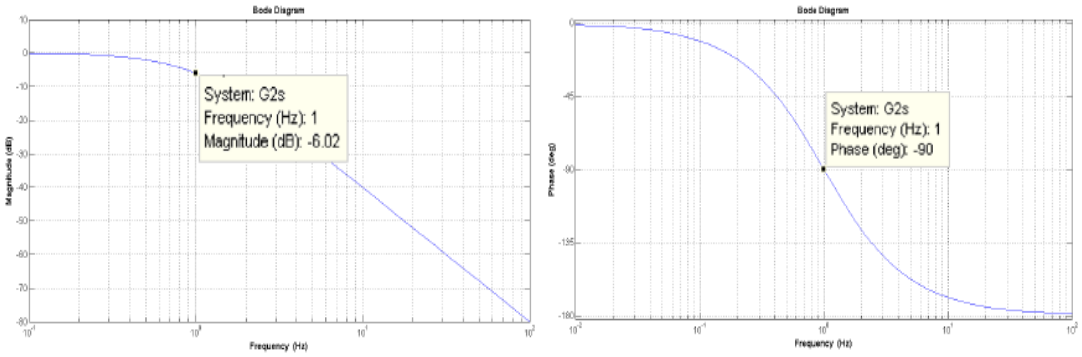
Figura 5 Bode del filtro de primer orden en tiempo discreto con frecuencia de corte f=1hz.

La misma prueba se puede realizar para el filtro de segundo orden, sabiendo que se tienen dos frecuencias naturales en el sistema, si se ajustan ambas al mismo valor, con la misma frecuencia de corte f=1Hz y periodo de muestreo T=0.001s, la función transferencia queda como se muestra en la ecuación 14 y función transferencia pulso queda expresada en la ecuación 15.

$$G2(s) = \frac{1}{0.02533s^2 + 0.3103s + 1} \tag{14}$$

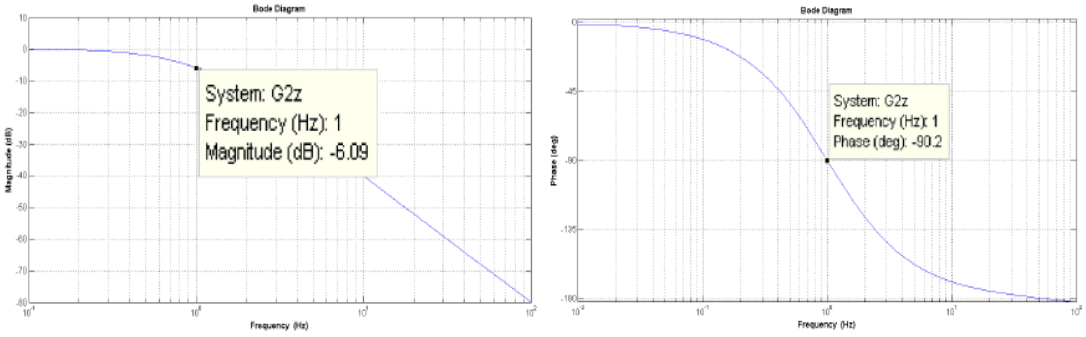
$$G2(z) = \frac{3.945 \times 10^{-7} z^{-2}}{1 - 1.999z^{-1} + 0.9987z^{-2}} \tag{15}$$

En la frecuencia de corte ahora se tendrán -6dB (figura 6a), y una fase de -90° (ver figura 6b), y ya que ambas frecuencias naturales, y por consiguiente ambos polos del sistema, cuentan con el mismo valor, se presenta una pendiente de atenuación de -40dB/dec. Así también en el modelo de tiempo discreto se presenta la magnitud mostrada en el modelo de tiempo continuo (figura 7a), la misma fase (figura 7b) y la misma pendiente de atenuación.




a) Magnitud del filtro en tiempo continuo. b) Fase) del filtro en tiempo continuo

Figura 6 Bode del filtro de segundo orden en tiempo continuo.




a) Magntud del filtro. b) Fase del filtro.

Figura 7 Bode del filtro de segundo orden en tiempo discreto función transferencia pulso.

## Implementación

La implementación se realizará mediante un microcontrolador en c++, en este caso se trata de un Arduino Due (pudiendo ser cualquier otro) y una interfaz gráfica desarrollada en LabView para crear gráficas de los datos provenientes de la tarjeta Arduino, con la finalidad de mostrar que el filtro diseñado es igual o más efectivo que algunas técnicas utilizadas para filtrar señales, como puede ser un promedio o un valor medio de datos, la única "desventaja" podría ser la carga matemática que conlleva por parte del diseñador, en especial si se quiere diseñar desde cero algún filtrado de este tipo, pero con distintas características. La señal que se procesará proviene de un cooler adaptado para funcionar de manera opuesta, es decir, genera un voltaje a partir del movimiento de sus aspas a manera de un sensor de flujo. Se realiza un acoplamiento entre el sensor descrito anteriormente y un cooler sin alterar, de esta manera se asegura que el mismo flujo circule a través de ambos.

El movimiento generado por las aspas del sensor produce una diferencia de potencial en las terminales que normalmente se utilizarían para alimentar el cooler, esta señal puede ser primeramente filtrada analógicamente para reducir el nivel de ruido que pudiera tener, pero en este caso se conectará directamente al convertidor analógico-digital con el que cuenta la tarjeta Arduino, con lo que se observa una señal que se encuentra oscilando constantemente y sin periodicidad; tiene una gran amplitud (tomando en cuenta los niveles minimos y máximos de la señal comparando con los máximos y mínimos del convertidor) oscila a una frecuencia muy baja, del orden de unos cuantos Hertz (figura 8).

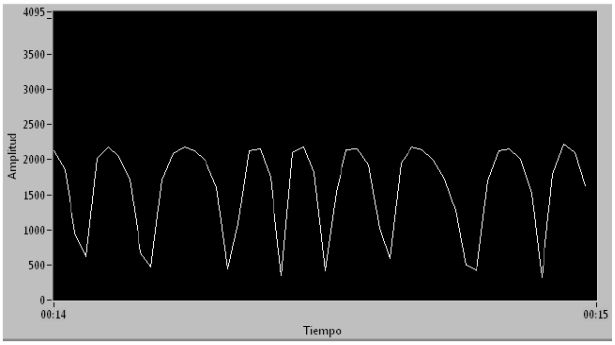



Figura 8 Señal sin filtrar.

Posteriormente la señal es procesada con el filtrado de primer orden, en esta ocasión se ajustó a una frecuencia de corte de 0.1Hz para poder reducir la mayor cantidad de componentes en frecuencia existentes en la señal. Dentro del programa de Arduino se creó una función que como parámetro recibe el valor de la señal y devuelve el valor de la salida del filtro, en este caso el valor calculado para el nodo de salida con los valores existentes en las variables, y siguiendo el algoritmo mencionado anteriormente, realizando los debidos corrimientos entre el nodo In y el nodo U1.

Con esto se consigue una señal más estable pero en la cual aún sigue existiendo una oscilación (figura 9), a partir de este punto es posible utilizar un promediado, o cualquier otra técnica, pero para este caso se implementaron las ecuaciones del filtro de segundo orden en vez de las de primer orden, aunque también se puede ingresar la salida obtenida en el proceso anterior a otro filtro de primer orden.

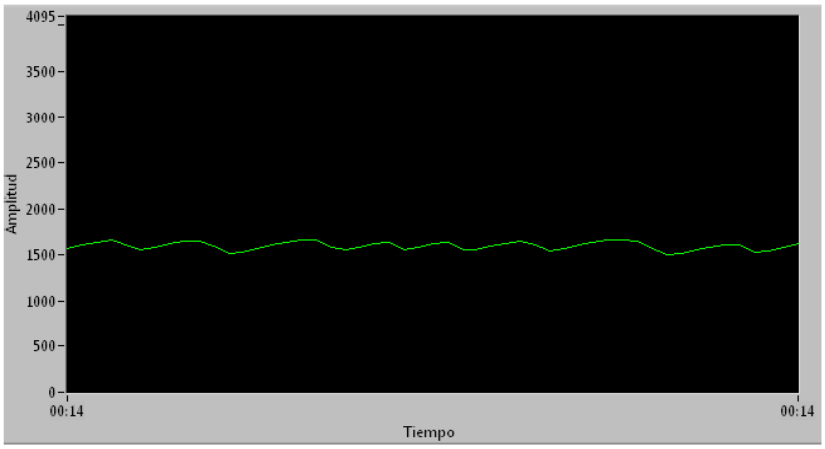



Figura 9 Señal procesada con filtro de primer orden y una frecuencia de corte f=0.1hz.

Para utilizar el filtro de segundo orden la estrategia es la misma, con la única diferencia de que se realizan los corrimientos al final de los cálculos, es decir, el valor de In pasa a u1 mientras que el valor de U1 pasa a U2.

Con este segundo sistema se consigue una señal mucho más estable ya que el nivel de oscilación es mínimo comparado con el filtrado de primer orden (figura 10), mucho más si se compara con la señal original, a pesar de que la frecuencia de corte utilizada es la misma que en el caso anterior, la diferencia radica en el

orden del filtrado, ya que un filtro de segundo orden presenta una pendiente de atenuación de -40dB/dec, mientras que uno de primer orden presenta una pendiente de -20dB/dec, es decir la pendiente del filtro de segundo orden es mas pronunciada que la de primer orden (figura 11), esto hace que las frecuencias superiores a la frecuencia de corte se vean más atenuadas.

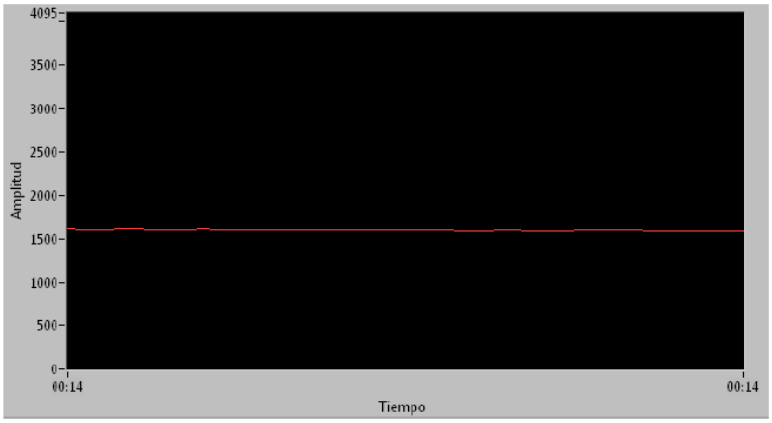



Figura 10 Señal procesada con filtro de segundo orden y una frecuencia de corte f=0.1hz.

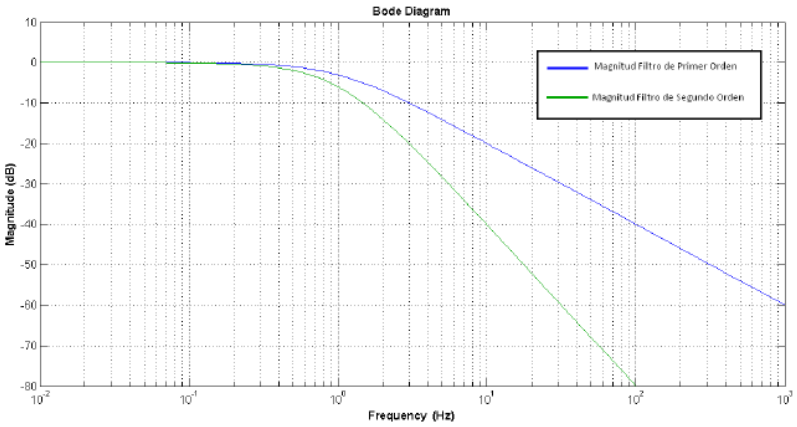



Figura 11 Pendientes de atenuación del filtro de primer orden y el filtro de segundo orden.

## 4. Discusión

Los dos filtros diseñados son de bastante utilidad pero presentan un inconveniente, en cuanto a código se refiere, ya que habría que declarar una función por cada señal a filtrar, o hacer un arreglo matricial e ir indexando los filtros. Con la finalidad de resolver dicha problemática se planteó programar la clase LPFilter, la cual contiene los métodos correspondientes a los filtros de

primero y segundo orden, de esta manera se puede utilizar un mismo código para filtrar n señales, simplemente declarando cada filtro con su respectiva frecuencia de corte y periodo de muestreo, posteriormente utilizando el método deseado (primer o segundo orden).

## 5. Conclusiones

Primeramente cabe señalarse que el procedimiento descrito en este artículo permite quitar carga matemática a la computadora ya que es el diseñador quien lleva a cabo el cálculo de las ecuaciones que son utilizadas, de manera que el programa solo tiene dos ecuaciones para resolver. Las dos ecuaciones proporcionadas al sistema están en función de la frecuencia de corte y el periodo de muestreo, por lo que es de fácil reconfiguración para ajustar una amplia gama de valores para las frecuencias de corte y el periodo de muestreo sin necesidad de que la computadora tenga que volver a determinar ecuaciones, sino coeficientes de las ecuaciones que a su vez solo se determinan con una función de exponenciación. Esto se vuelve de utilidad para procesar señales con oscilaciones que no son periódicas, ya que sería complicado determinar su frecuencia base y se tendrían que probar distintas frecuencias de corte.

## 6. Referencias

[1] J. Antonio, Á. Cedillo, K. Michael, L. Bos, G. M. Romero, “Implementación de Filtros Digitales tipo FIR en FPGA”. No.37. 2008. Pp.83-87.

[2] M. Mirghani, Implementation of Matched Filters Using Microcontrollers. 2016. Pp. 128–132.

[3] G. J. Dolecek, S. K. Mitra, “Computationally Efficient Multiplier-Free Fir Filter Design”. Vol. 10. No. 3. 2007. Pp. 251–267.

[4] A. H.-C. B. Psenicka, F. García-Ugalde, “The Bilinear Z Transform by Pascal Matrix and Its Application in the Design of Digital Filters”. Vol. 9. No. 11. 2002. Pp. 368–370.

[5] B. C. Kuo, Sistemas de control automático. 7 ed. 1996.

## 7. Autores

Ing. Armando Satiel Chipule Pérez obtuvo la licenciatura en Ingeniería Electrónica en el Instituto Tecnológico de Orizaba, actualmente es estudiante de la Maestría en Ingeniería Electrónica del mismo instituto, sus líneas de investigación son Teoría de Control Clásico y Programación de Microcontroladores.

Dr. Gerardo Águila Rodriguez, obtuvo el grado de Doctor en ciencias de la Ingeniería Eléctrica en la SEES, CINVESTAV – I.P.N. en el año 2008.

Dr. Rubén Posada Gómez es Dr. En Automatización y procesamiento de señales del Institut National Politechnique de Lorraine.