MANUAL DE REFERENCIA

TOOLBOX DE PROCESAMIENTO DE SEÑALES

DSPIC33FJ32MC204

(**NOTA:** Los logos aquí presentados son solo para fines académicos, que sirven como ilustración de los programas o servicios de empresas, usados para el desarrollo del presente trabajo. Ya que estas son marcas registradas).

TABLA DE CONTENIDOS

Objetivo del manual

El presente manual tiene como objetivo dar una breve descripción de cada una de las librerías desarrolladas para el Toolbox de procesamiento de señales. Donde se menciona algunas de las consideraciones de diseño, limites y recomendaciones del Toolbox.

Hardware de aplicación

Para poder visualizar las señales generadas por el dsPIC se implementó un circuito en el software de simulación de circuitos PROTEUS. Se compone del DSPIC33FJ32MC204, un convertidor digital análogo (DAC1208) el cual esta alimentado con una tensión mínima de 0V y máxima de 10V (este aspecto es de vital importancia ya que limita la amplitud máxima de las señales que se pueden generar. Mas adelante se brinda una explicación detallada). También se tienen algunos amplificadores operacionales y un filtro RC que ayuda a suavizar un poco la señal generada.

Como instrumentos de medida, para poder verificar las señales generadas se implementaron graficas estáticas y osciloscopios. También se incluyo una terminal virtual, que permite leer los datos correspondientes a cada unas de las muestras de la señal generada. Esta herramienta sirve para la depuración del código, dado que estos valores se pueden tomar y graficar mediante un programa adicional (por ejemplo, EXCEL o el graficador DESMOS) y así poder encontrar fallas al realizar un análisis grafico o numérico. Se incluyeron 5 botones lógicos que permiten seleccionar la señal que se desea generar.

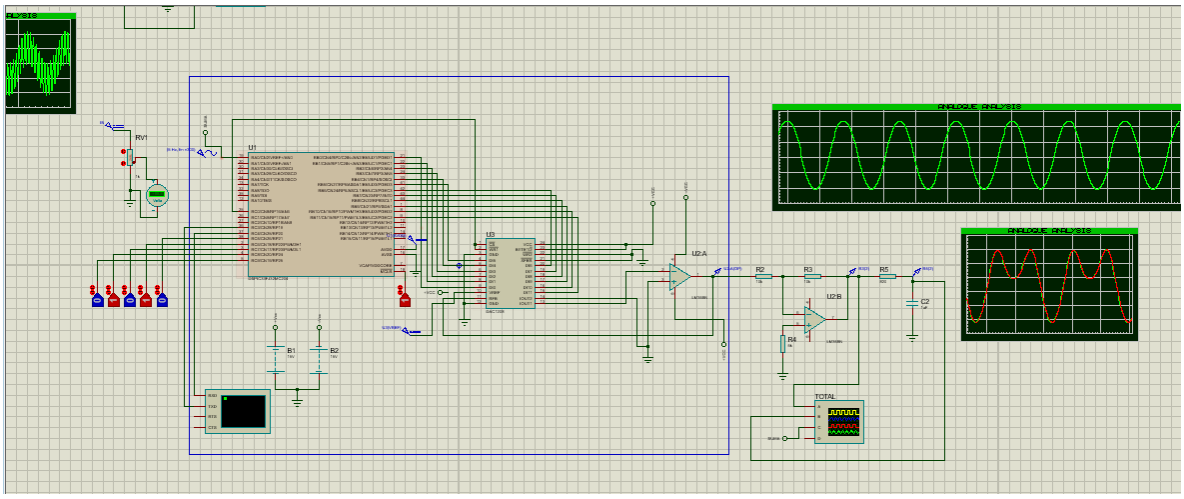


Figura 1. Hardware del toolbox.

1. Generación de señales básicas.

El toolbox fue diseñado con el editor del MPLABX y permite generar señales senoidales, cuadradas, triangulares y diente de sierra. Cada señal es generada mediante una cabecera que recibe los siguientes parámetros:

Amplitud de la señal (A), frecuencia de la señal (f), frecuencia de muestreo (fm), ángulo de desfase (fa).

Adicional a los parámetros de entrada, cada cabecera entrega el dato correspondiente al numero de muestras necesarias para construir la señal en base a los parámetros establecidos (mue).

Los parámetros (A, f, fm, fa) son datos del tipo (float) en cambio la cantidad de muestras es un dato del tipo (int).

1.1 Consideraciones de diseño:

Para la implementación de la cabecera de las señales, primero se creó en un solo archivo principal una función que recibía como parámetros la amplitud (A), frecuencia de la señal (f), ángulo de desfase (fa) y la frecuencia de muestreo (fm). Esta función aparte de entregar cada uno de los valores de la señal también entrega el numero de la cantidad de muestras (m).

Como la señal deseada, se va a generar en el dominio discreto y no continuo, se debe normalizar la frecuencia de la señal para que esta quede en función de las muestras. Lo anterior se define como la frecuencia normalizada (fn) dada por la siguiente ecuación:

$$fn = \frac{f}{fm}$$

A parte de la frecuencia normalizada se construirán cada unas de las señales. Para el calculo de la cantidad de muestras de define la siguiente ecuación.

$$m = \frac{1}{fn}$$

También es importante aclarar que el valor de pi (π) para cada una de las señales tiene un valor de (3.14).

1.2 Ejemplos

RECOMENDACIONES

Respecto a la amplitud máxima permitida, la señal generada se debe compensar con un offset, porque los 16 bits del puerto del DSPIC por los cuales se generan la señal; son un entero sin signo y por consiguiente no se pueden generar valores negativos. Con lo anterior dado que la alimentación del convertidor análogo digital (DAC1208) es alimentado con una tensión de 10V la amplitud máxima permitida son 4V para cuando se agregue el offset esta quedara ubicada entre 0V y 8V. Como la alimentación del DAC es de 10V la señal no se saturaría. Es importante considerar que el voltaje de saturación por l general se alcanza al 90% de la señal, por lo tanto, no se aconseja trabajar al tope o muy cerca a los límites permitidos.

Teorema del muestreo:

Adicionalmente se recomienda tener cuidado con el teorema del muestreo de Nyquist-Shannon, donde se establece que la frecuencia de muestreo debe ser dos veces o más la frecuencia de la señal. Pero lo anterior no es del todo cierto.

Por ejemplo, si establecemos una frecuencia de 70 Hz:

$$70\ Hz * 3\ veces = 210$$

Como se puede observar en la ecuación anterior, si se establece una frecuencia de muestreo de 300 Hz no se estaría pasando por alto el teorema del muestreo de Nyquist-Shannon. Pero si se calcula el número de muestras se obtiene el siguiente resultado.

$$fn = \frac{f}{fm} = \frac{70}{300} = 0.233333333$$

$$m = \frac{1}{fn} = \frac{1}{fn} = \frac{1}{0.233333333} = 4\ muestras\ aproximadamente$$

Con 4 muestras no se podrían construir una buena aproximación de la señal. Por lo tanto, es importante saber que valores de frecuencia son seleccionados y en función de esta y la frecuencia de muestreo realizar el calculo de la cantidad de muestras. Dependiendo de sus requerimientos usted establece si esa cantidad de muestras es suficiente.

Se concluye que a pesar de que se aplican muchos conceptos teóricos para la construcción de las señales, al momento de integrar el software con el hardware se experimentan ciertas dificultades. Por lo tanto, es indispensable contar con una metodología para la depuración del proyecto.

Frecuencia de la señal generada:

Para el Toolbox se estableció una frecuencia de muestreo (fm) por defecto de 300 Hz (La cual no se puede modificar. Pero para una segunda versión de este toolbox se podrá modificar). Como esta frecuencia es muy baja ya que permite generar pocas muestras (aunque a su vez dependa de la frecuencia (f) de la señal), por lo tanto, se incluye un error en la frecuencia de la señal generada. Conforme el número de muestras disminuya, el error en la frecuencia aumenta. Con lo anterior así la frecuencia de la señal sea baja, se conserva el error.

NOTA: Se opto por establecer un valor de frecuencia de muestreo por defecto, para así no exceder con la cantidad de muestras y por ende el tamaño de memoria requerido por el Toolbox, ya que, por tratarse de un equipo de procesamiento, el dspic cuenta con una memoria limitada. En la sección de los filtros se brinda una explicación de como se gestiona la cantidad de memoria.

Visualización de las señales generadas:

Para poder visualizar las señales se implementó una cabecera llamada imprimir (señal,mue,tipo), la cual permite imprimir la señal generada mediante el puerto B del dspic.

Esta cabecera recibe tres parámetros:

Señal = Nombre del arreglo que almacena los valores de la señal generada.

Mue = Cantidad de muestras necesarias para construir la señal (Generalmente este valor es estregado por las funciones de cada señal).

Tipo = Seleccione el tipo de impresión. Dependiendo del tipo de señal y si se quiere visualizar de manera indefinida la señal generada o un único periodo se pone un numero especifico (hay 4 tipos por lo tanto los números están en el rango de 1 a 4). A continuación, se explica cada tipo. Ver tabla 1.

Numero	Descripción
1	Imprime de forma indefinida las señales seno, cuadrada, triangular, diente de sierra, la derivada e integral del seno, derivada e integral de la triangular, convolución, filtro IIR, filtro FIR
2	Imprime de forma indefinida la derivada e integral de la cuadrada
3	Permite imprimir un solo periodo de la señal
4	Permite imprimir la señal de salida de los filtros IIR y FIR para señales ingresadas por el ADC del dsPIC

Tabla 1. Tipos de impresión, para visualizar las señales generadas por el dsPIC.

Señal senoidal

La función de la señal seno está dada por la siguiente ecuación, la cual se encuentra en función de las muestras:

$$f[n] = A.sen(2 * 3.14 * fn * n + fa)$$

Nombre de la cabecera: float * funSeno(float A, float f, float Fa, float fm, int *m).

Ejemplo:

Se procedió a generar una función con las siguientes características.

```
s=funSeno(1,5,0,300,&mue);
imprimir(s,mue,1);
serial(s,mue);
```

Resultados obtenidos

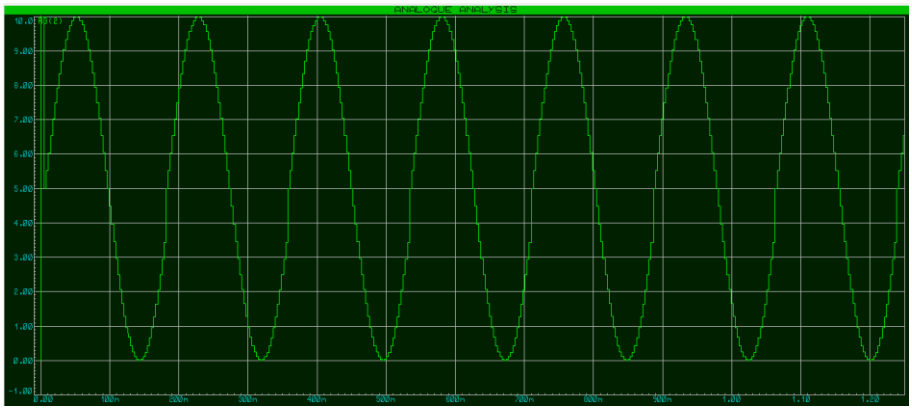


Figura 2. Señal senoidal.

Si desea imprimir los valores que construyen un periodo de la señal mediante el terminal virtual (independiente del tipo de señal, debe seleccionar en la cabecera imprimir () el numero 3 para así poder imprimir un solo periodo de la señal y de esta forma la cabecera serial () podrá entregarle la cantidad de muestras correspondientes a ese periodo. Si no imprime un único periodo no podrá visualizar las muestras. A continuación, se muestra un ejemplo.

```
s=funSeno(1,5,0,300,&mue);
imprimir(s,mue,3);
serial(s,mue);
```

Figura 3. Sección de código.

A continuación, se presenta un único periodo de la señal.

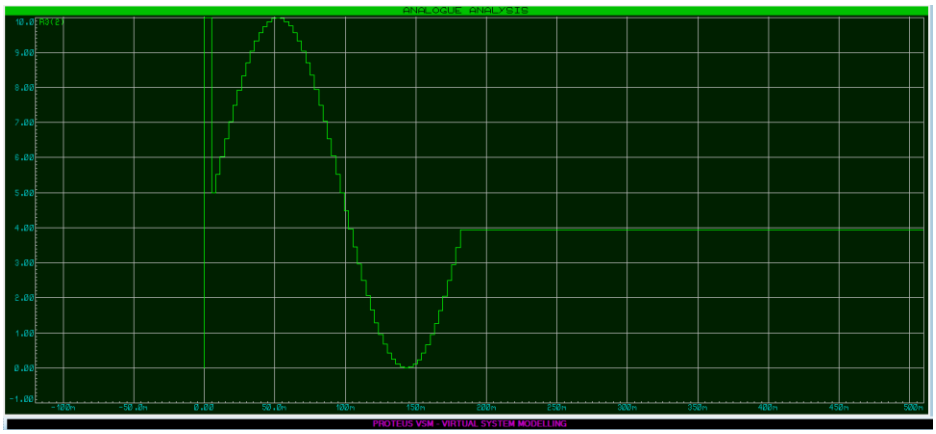


Figura 4. Un solo periodo de la señal senoidal.

Para obtener los valores de la señal se debe correr la simulación y abrir la consola del terminal virtual.

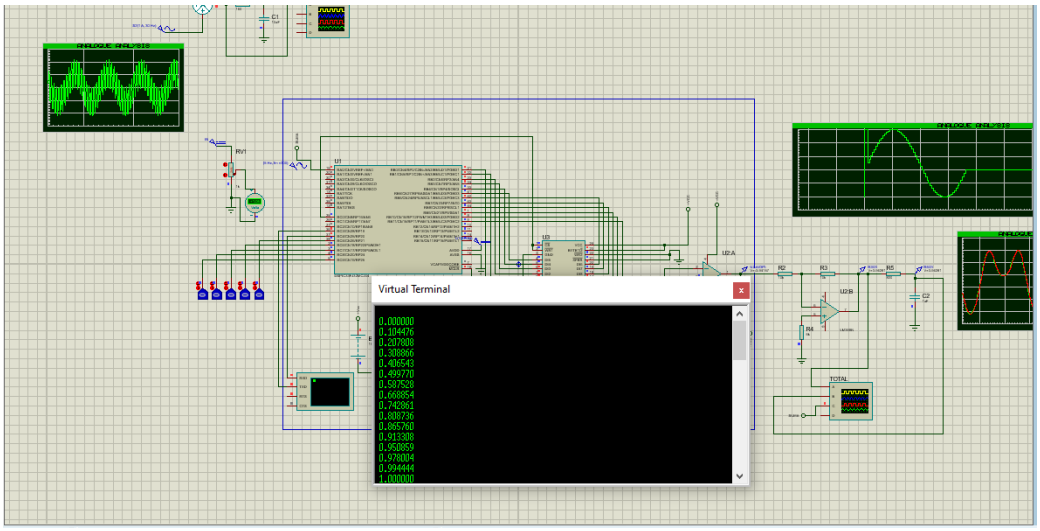
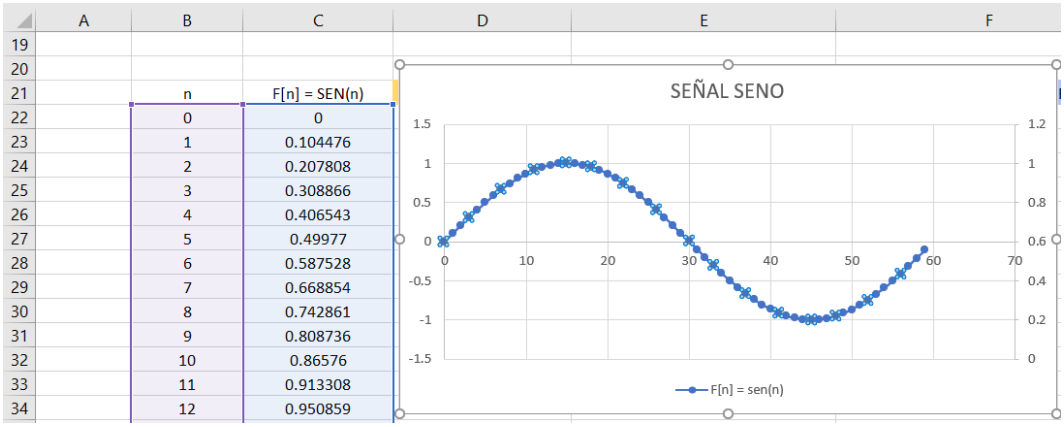


Figura 5. Visualización de la consola del terminal virtual.

Con los valores entregados por la consola del terminal virtual los puede copiar y llevar a Excel u otro programa de análisis numérico de su elección y graficarlos. Este proceso le sirve como herramienta de depuración, en caso de que la señal visualizada en el osciloscopio o grafica estática no coincidan con la señal esperada. En la siguiente figura se muestra como ejemplo la grafica obtenida a partir de los datos de la figura anterior.



n	F[n] = SEN(n)
0	0
1	0.104476
2	0.207808
3	0.308866
4	0.406543
5	0.49977
6	0.587528
7	0.668854
8	0.742861
9	0.808736
10	0.86576
11	0.913308
12	0.950859

Figura 6. Grafica de la señal senoidal en Excel.

Para las demás señales se repite el mismo proceso de la señal seno (). Teniendo en cuenta las recomendaciones previas.

Señal cuadrada

```
s=funCuadrada(1,5,300,&mue);
imprimir(s,mue,1);
serial(s,mue);
```

Figura 7. Sección de código.

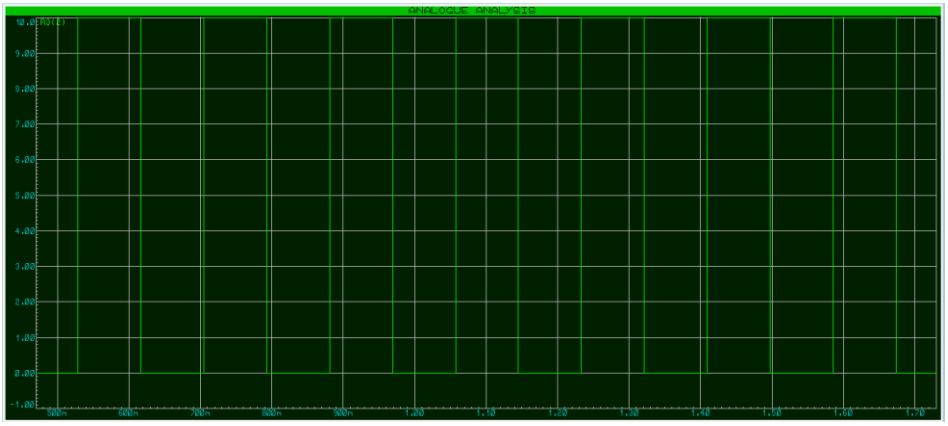


Figura 8. Señal cuadrada.

Señal triangular

```
s=funTriangular(1,5,300,&mue);
imprimir(s,mue,3);
serial(s,mue);
```

Figura 9. Sección de código.

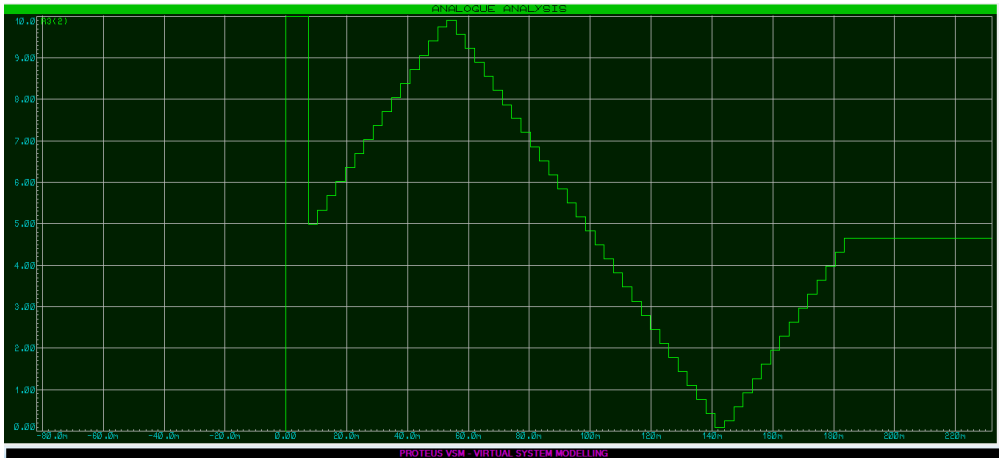


Figura 10. Un solo periodo de la señal triangular.

Señal diente de sierra

```
s=funDiente(1,5,300,&mue);
imprimir(s,mue,1);
serial(s,mue);
```

Figura 11. Sección de código.

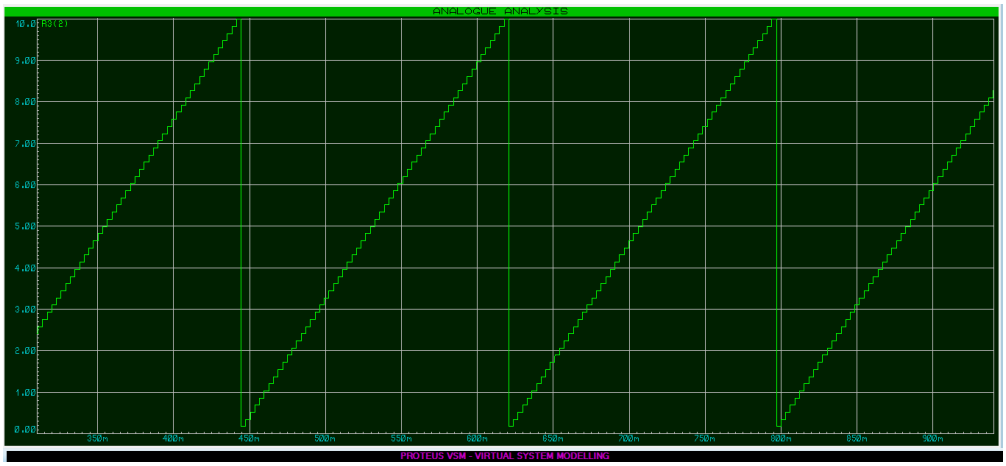


Figura 12. Señal diente de sierra.

2. Derivadas de las señales básicas.

Para generar la derivada de las señales básicas se implementó el operador derivada, mediante el método de diferencias hacia adelante, dado por la siguiente ecuación.

$$f'_{(x)} = \frac{f_{x_{i+1}} - f_{x_i}}{x_{i+1} - x_i}$$

Para el calculo de la derivada se implementó la siguiente cabecera:

float * derivada(float *pY, int tx)

Recibe los siguientes parámetros:

float *pY = Posición de memoria del primer valor, del arreglo que almacena los valores de la señal que se desea derivar.

int tx = Cantidad de muestras que construyen la señal que se desea derivar. Aquí se puede reemplazar por la variable “mue”, la cual es retornada por las funciones que generan las señales seno, cuadrada, triangular y diente de sierra. Mue, corresponde a la cantidad de muestras.

Observaciones:

Tenga en cuenta que dependiendo de la amplitud de la señal que desea derivar, la amplitud de la derivada no va a ser la misma. Con lo anterior si la derivada en amplitud excede los límites del convertidor digital - análogo, la señal de salida se saturará. Adicionalmente debe considerar el offset que se añade a la señal generada (Ver página 6, sección “Recomendaciones”).

Adicionalmente si al momento de generar la señal que desea derivar, no tiene en cuenta las recomendaciones previas de las secciones anteriores, la derivada no será la correcta.

Para efectos prácticos se recomienda realizar el cálculo teórico de la derivada para así poder determinar la amplitud y por ende si la cabecera le será útil para tal aplicación.

2.1 Ejemplos

Derivada de la señal seno

```
s=funSeno(1,5,0,300,&mue);
ph=derivada(&s[0],mue);
imprimir(ph,mue,1);

serial(ph,mue);
```

Figura 13. Sección de código.

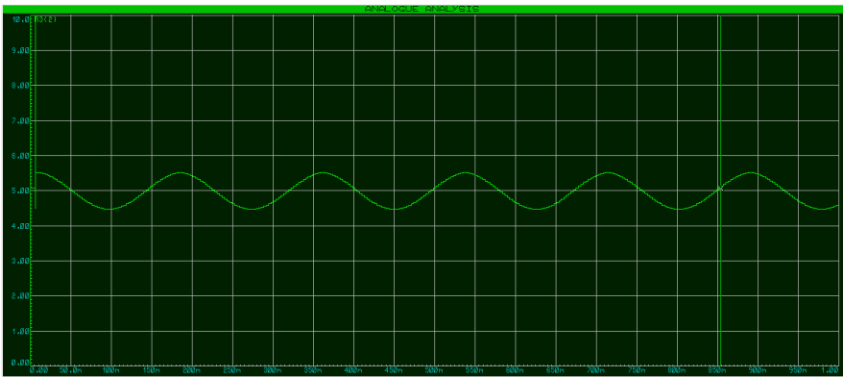


Figura 14. Derivada de la señal seno.

Derivada de la señal cuadrada

```
s=funCuadrada(1,5,300,&mue);
ph=derivada(&s[0],(mue+1));
imprimir(ph,mue,2);

serial(ph,mue);
```

Figura 15. Sección de código.

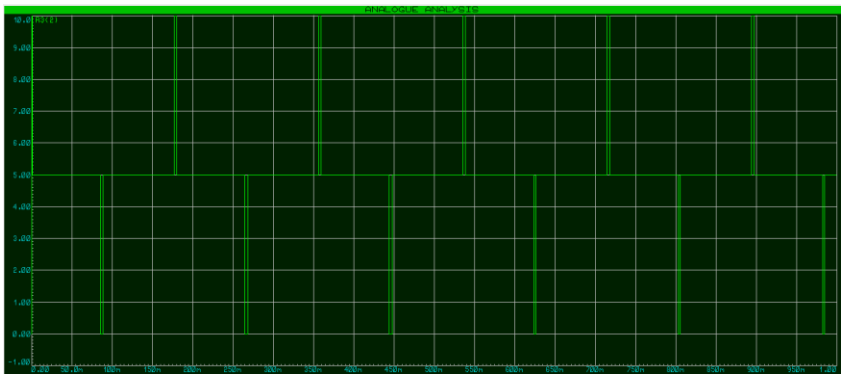


Figura 16. Derivada de la señal cuadrada.

Derivada de la señal triangular

```
s=funTriangular(1,5,300,&mue);
ph=derivada(&s[0],mue);
imprimir(ph,mue,1);

serial(ph,mue);
```

Figura 17. Sección de código.

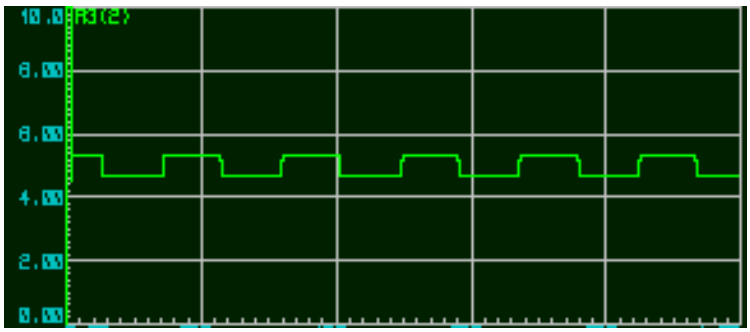


Figura 18. Derivada de la señal triangular.

3. Integrales de las señales básicas.

Para el calculo de la integral se implemento el operador integral, mediante el método de diferencias hacia atrás, definido por la siguiente ecuación.

$$y_{i[k]} = h * y_{[k]} + y_{i[K-1]}$$

Donde h corresponde al tiempo de muestreo (tm), el cual se obtiene a partir de la frecuencia de muestreo.

$$fm = 300\ Hz$$

$$tm = \frac{1}{300} = (0.003\ s)\ aprox.$$

La cabecera para el calculo de la integral tiene la siguiente estructura:

float * integral(float *pG, int tx, float st)

float *pG = Posición de memoria del primer valor, del arreglo que almacena los valores de la señal que se desea integrar.

int tx = Cantidad de muestras que construyen la señal que se desea derivar. Aquí se puede reemplazar por la variable “mue”, la cual es retornada por las funciones que generan las señales seno, cuadrada, triangular y diente de sierra. Mue, corresponde a la cantidad de muestras.

float st = Tiempo de muestreo (tm).

Observaciones:

Al igual que en la derivada, con la integración debe tener cuidado con la amplitud de la señal generada para no exceder los límites del convertidor digital – análogo y así evitar que la señal de salida se sature. Debe tener en cuenta las recomendaciones previas de las secciones anteriores o la integral no será la correcta.

3.1 Ejemplos

Integral de la señal seno

```
s=funSeno(1,5,0,300,&mue);
ph=integral(&s[0],mue,0.003);
imprimir(ph,mue,1);
```

Figura 19. Sección de código.

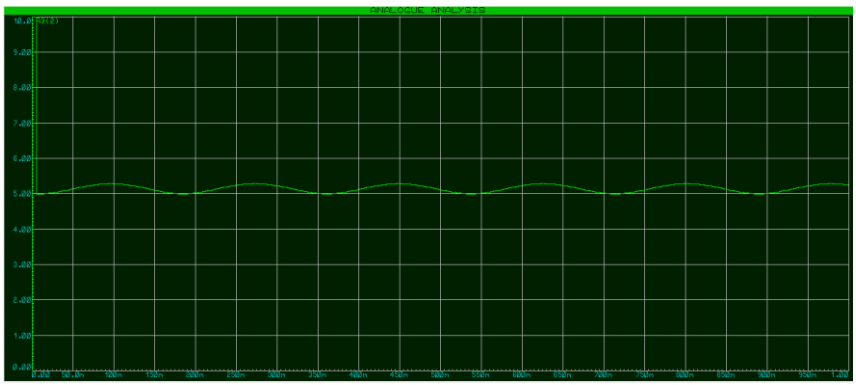


Figura 20. Integral de la señal seno.

Integral de la señal cuadrada

```
s=funCuadrada(1,5,300,&mue);
ph=integral(&s[0],mue,0.003);
imprimir(ph,mue,2);
```

Figura 21. Sección de código.

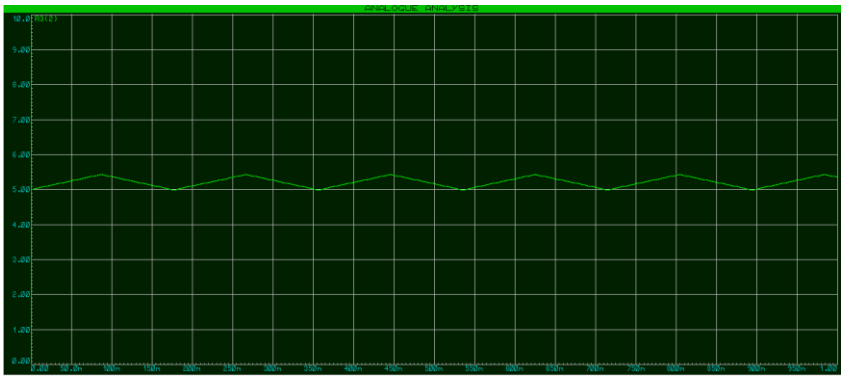


Figura 22. Integral de la señal cuadrada.

Integral de la señal triangular

```
s=funTriangular(1,5,300,&mue);
ph=integral(&s[0],mue,0.003);
imprimir(ph,mue,1);
```

Figura 23. Sección de código.

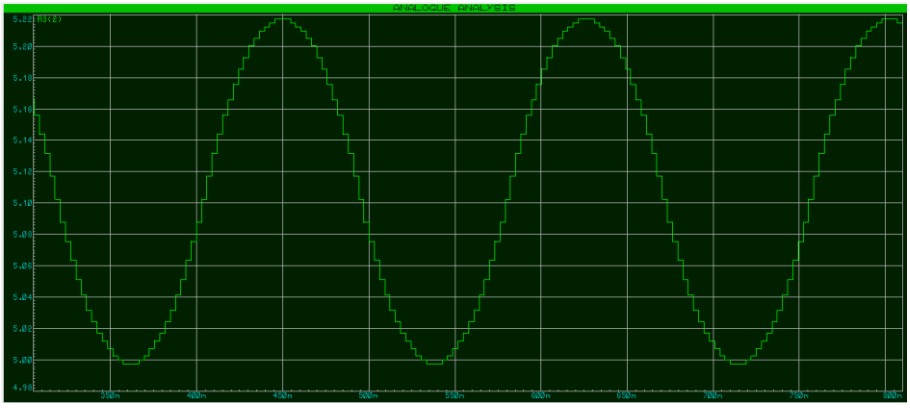


Figura 24. Integral de la señal triangular.

4. Convolución.

CONVOLUCION A PARTIR DE SEÑALES GENERADAS POR EL dsPIC

Se implemento la convolución discreta dada por la siguiente ecuación.

$$y_{[n]} = \sum_{k=-\infty}^{\infty} x_{[k]} \cdot h_{[n-k]}$$

La cabecera de la convolución se definió de la siguiente forma:

float * conv2(float *pX, float *pH, int th, int tx, int *salida)

float *pX = Posición de memoria del primer valor, del arreglo que almacena los valores de la primera señal.

float *pH = Posición de memoria del primer valor, del arreglo que almacena los valores de la segunda señal.

int th = Numero de muestras de la segunda señal.

int tx = Numero de muestras de la primera señal.

int *salida = Tamaño del arreglo que almacena los valores de la convolución.

Observaciones:

Con la convolución también se recomienda tener mucho cuidado con las amplitudes de las señales con las que se va a realizar la convolución, para que la amplitud de la señal resultante no exceda el voltaje máximo del convertidor digital – análogo. No olvidar el offset que se añade a la señal.

Con lo anterior se recomiendan amplitudes máximas en orden de los mV. Cualquier valor por fuera de este rango causa que la señal de salida se sature.

4.1 Ejemplo

```
h=funSeno(0.05,5,0,300,&mue1);

X=funSeno(0.05,10,0,300,&mue2);

pY=conv2(&X[0],&h[0],60,30,&mue);

imprimir(pY,mue,1);
```

Figura 25. Sección de código.

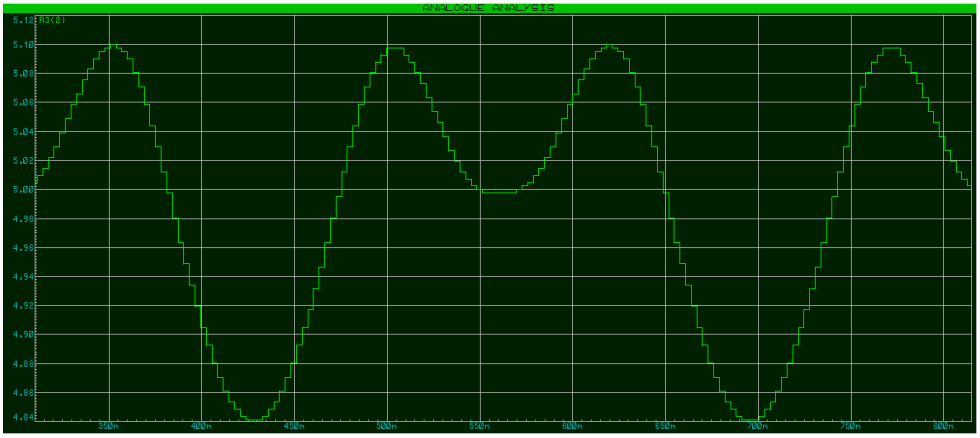


Figura 26. Convolución entre dos señales.

CONVOLUCION A PARTIR DE SEÑALES OBTENIDAS CON EL CONVERTIDOR ANALOGO – DIGITAL (ADC)

Consideraciones de diseño

Dado que se desean muestrear señales mediante el ADC, se estableció que la frecuencia de muestreo (fm=300 Hz) con la que se trabajó en los numerales anteriores es muy baja, limitando mucho el rango de frecuencias de las señales (f) que se desean muestrear y por ende la cantidad de muestras capturadas.

Por lo tanto, se amplió la frecuencia de muestro (fm) a **4000 Hz**. Con esta frecuencia dado que se va a realizar una captura para un número limitado de datos, cada vez que se modifique la frecuencia de las señales, se debe calcular el numero de muestras necesario para construir cada señal y se debe modificar el código principal.

Recuerde que para el calculo del número de muestras (m), debe emplear las siguientes ecuaciones:

$$fn = \frac{f}{fm}$$

$$m = \frac{1}{fn}$$

OBSERVACIONES

Para esta aplicación se estableció que la frecuencia máxima que se puede ingresar en las señales es de 250 Hz. Esto se debe a que cumpliendo el teorema del muestreo de Nyquist-shannon, el cual establece que la frecuencia deber ser el doble o más de la frecuencia de la señal. En este diseño se estableció que seria 16 veces la máxima frecuencia permitida.

Para ingresar ambas señales al dsPIC se habilitaron los pines A0 y A1.

Ejemplo

Se ingresaron dos señales (S1 y S2) de 100 Hz y 100 mV cada una.

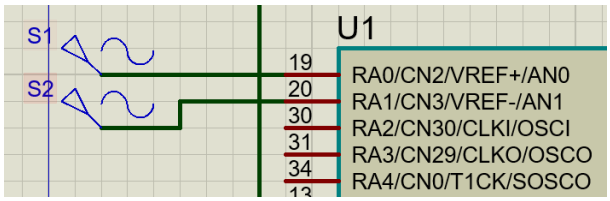


Figura 27. Conexión de señales análogas a puertos de ADC.

Después se ingresó la cantidad de muestras en el código principal.

```
int taml = 40; // numero de meustras para S1
int tam2 = 40; // numero de meustras para S2
```

Figura 28. Sección de código.

En la siguiente figura se presenta el resultado obtenido.

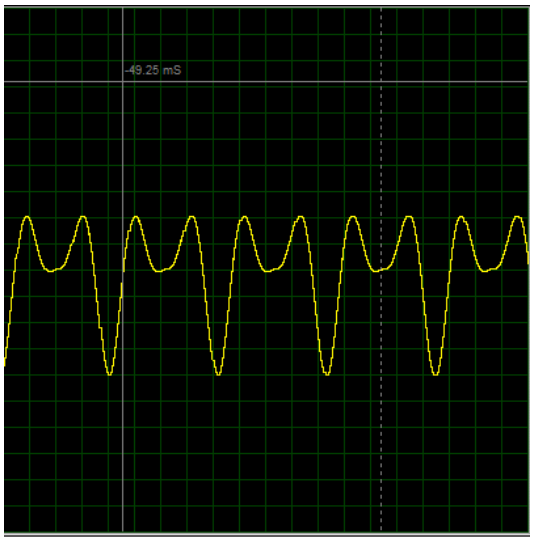


Figura 29. Convolución entre dos señales.

5. Filtro de respuesta finita al impulso (FIR)

Para la implementación del filtro FIR se usó la misma cabecera de la convolución dado que presenta la misma ecuación de recurrencia.

$$y_{[n]} = \sum_{k=-\infty}^{\infty} x_{[k]} \cdot h_{[n-k]}$$

Donde (x) o (h) se cambian por el arreglo que guarda los coeficientes (b) del filtro diseñado.

Para la implementación de este filtro se recomienda seguir todas las recomendaciones anteriores del numeral de la convolución.

Ejemplo

Se ingresaron dos señales de diferentes frecuencias (400 Hz y 30 Hz) con la misma amplitud igual a 1V. Después se sumaron de forma análoga y se ingresaron a dsPIC para implementar un filtro pasa baja con frecuencia de corte de 50 Hz.

A continuación, se presenta la suma de ambas señales.

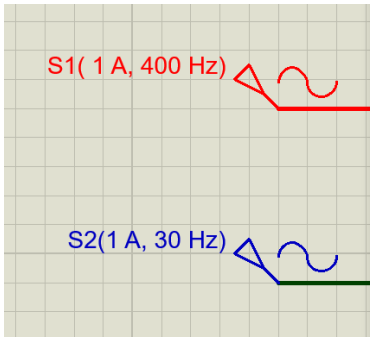


Figura 30. Parámetros de señales análogas.

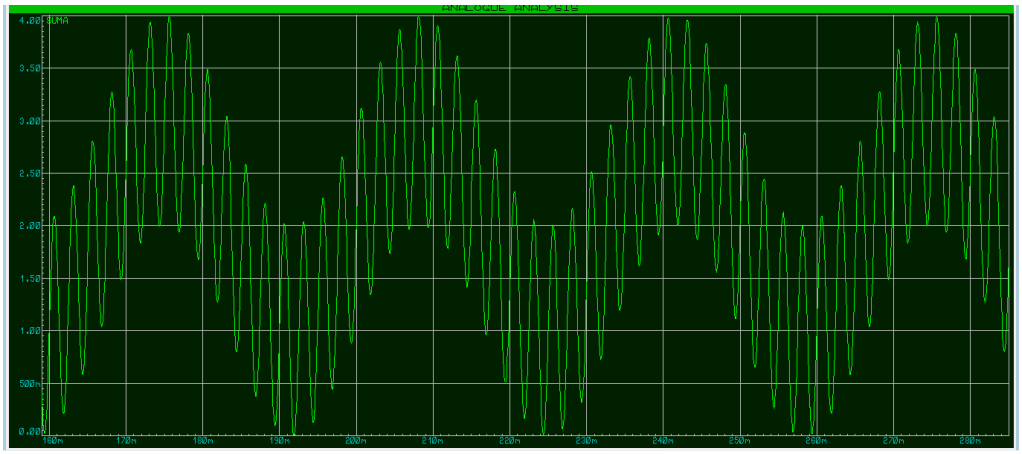


Figura 31. Suma de señales.

Resultado obtenido:

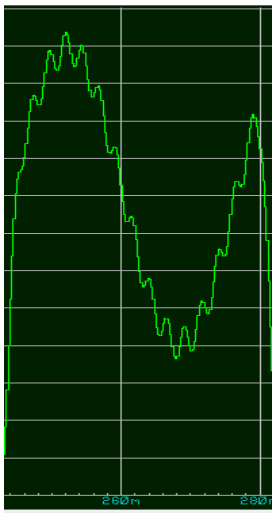


Figura 32. Señal filtrada.

6. Filtro de respuesta infinita al impulso (IIR)

Para la implementación del filtro IIR se creo una nueva cabecera, la cual integra la que ya se tenía de la convolución.

$$y_{[n]} = \sum_{k=0}^{N} b_{[k]} \cdot h_{[n-k]} + \sum_{k=1}^{N} a_{[k]} \cdot y_{[n-k]}$$

La estructura de la cabecera tiene la siguiente forma:

float * conv4(float *sr, float *fcA, float *fcB, int tam)

float *sr = Posición de memoria del primer valor, del arreglo que almacena la señal que se desea filtrar.

float *fcA = Posición de memoria del primer valor, del arreglo que almacena los coeficientes del denominador del filtro.

float *fcB = Posición de memoria del primer valor, del arreglo que almacena los coeficientes del numerador del filtro.

int tam = Cantidad de muestras que se tomen de la señal que se desee filtrar.

7. correlación cruzada.

Para la correlación cruzada también se uso la cabecera de la convolución (referirse a la sección 4) y por ende también incluye las observaciones y consideraciones de la convolución en cuanto a la frecuencias y amplitudes máximas permitidas.

La único que cambia es que una de las dos señales que se va a convolucionar para el cálculo de la correlación cruzada, se invierte mediante una cabecera. La cual tiene la siguiente forma.

float * INVER(float *s, int m)

float *s = Posición de memoria del primer valor, del arreglo que almacena la señal que se desea invertir.

int m = Cantidad de muestras de la señal que se desea invertir.

Aquí se presenta el resultado obtenido para la correlación cruzada entre dos señales de igual frecuencia.

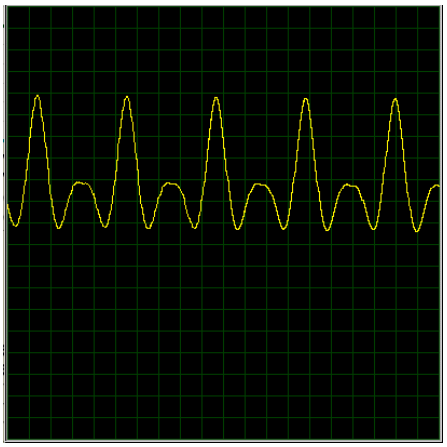


Figura 33. Correlación cruzada.

8. Autocorrelación.

Para el cálculo de la autocorrelación se emplea la misma cabecera y metodología de la correlación cruzada.

CONCLUSIONES

1. Se logro implementar cada una de las operaciones para el procesamiento de señales.

2. Debido a ciertas dificultades técnicas y teóricas, el rendimiento del toolbox no es del todo eficiente ya que requiere de más pruebas de depuración para corregir fallas y mejorar sus alcances.

3. Todos estos componentes que deben ser mejorados se podrían incluir en una segunda versión.