PROYECTO EN ELECTRONICA II

PRÁCTICA filtros FIR – IIR. Parte 1.

Implementación

INTEGRANTES	PROFESOR
Daniel Guillermo Morantes Salguero.	Ing. Pablo Rodríguez Ferro.
Joan Ferney Muñoz Tarazona.	**GRUPO:** 4-1.
Mauricio David Cuello Alzate.	**FECHA DE ENTREGA:** 02 de Mayo de 2021.

Desarrollo de la práctica:

Esta práctica estaba dividida en dos partes, para la primera parte se debía realizar un código para convertir de analógico a digital dos canales de entrada y convertir las dos señales digitalizadas a analógicas utilizando el DAC de la tarjeta de desarrollo, y para la segunda parte de debía implementar 7 filtros digitales a través de uno de los dos canales desarrollados en la primera parte y comprobar su funcionamiento.

La primera parte se comenzó desarrollándose para un solo canal, con el fin de que todos los ítems funcionaran correctamente para finalmente colocar los dos canales multiplexados a la entrada del ADC, durante el proceso se presentaron problemas con los bits de configuración de la transmisión por SPI, además de problemas en la recepción de la UART, sin embargo, los problemas que mayor relevancia tuvieron fueron: el valor del timer y la transmisión por UART.

Al iniciar el programa, se inicializan todas las variables y registros necesarios para el funcionamiento requerido de este laboratorio, en este caso se configuró el oscilador del sistema a 64 MHz y se configuró el timer 0 para que genere una interrupción en overflow y además para ser atendida en alta prioridad junto con el ADC, dando como resultado un timer a 4 us. El algoritmo implementado realiza una pregunta en la interrupción si se llegó al valor de “count_to” de la estructura “states”, que al iniciar es 12, esto con el fin de iniciar la conversión del ADC en aproximadamente 50 us. Posteriormente, se ejecuta el programa principal, sin embargo en la primera conversión el programa principal no realiza ninguna acción relevante. Al terminar la conversión del ADC, se guarda el canal del cual proviene el dato, se cambia al otro canal y se pone en ‘1’ la bandera de conversión.

Luego de haber leído un dato, este se convierte en voltaje y se toman cada uno de los dígitos en una variable, una para el que corresponde al entero y otras dos para los decimales. Se ponen en ‘1’ las banderas respectivas, luego se procede a transmitir estos tres valores por UART, y puesto que se tienen dos canales entonces la cantidad de valores a transmitir se convierten en seis, esto más dos valores adicionales para separar cada valor de voltaje y para determinar el final de cada dato de voltaje.

Los dos valores de ADC son preparados en el programa principal para ser transmitidos por SPI, debido a que el SPI tiene un buffer de transmisión de 8 bits por lo que la variable entera que almacena los 12 bits de la lectura de ADC debe ser tomada como dos de 8 bits, que para este caso

se llaman “MSB_spi” y “LSB_spi”, a los cuales se les agrego los parámetros necesarios para el uso del DAC. El módulo de SPI realiza la transmisión en una interrupción debido a que es necesario transmitir los 16 bits juntos, es decir, uno después del otro de manera consecutiva y lo más rápido posible para que el circuito del DAC no tenga problemas al interpretar la trama enviada.

Por último, si se presenta el caso, se reciben los datos de cambio de frecuencia de muestreo mediante una interrupción de recepción de la UART. Debido a que los datos enviados pueden superar el máximo valor que puede dar un número de 8 bits se envían dos bytes. En consecuencia, hay dos interrupciones y el programa principal se encarga de reconstruir el número enviado que será procesado (multiplicado) para que dé como resultado el tiempo de muestreo requerido.

Resultados:

🎬 Medida de voltaje DC a la entrada vs la salida DAC vs salida serial por UART:

Tabla 1. Medición de voltaje DC a la entrada vs Salida DAC vs salida serial por UART.

DVM		Serial		DAC		Error porcentual Canal A (DVM vs Serial)	Error porcentual Canal B (DVM vs Serial)	Error porcentual Canal A (DVM vs DAC)	Error porcentual Canal B (DVM vs DAC)
Canal A	Canal B	Canal A	Canal B	Canal A	Canal B				
3.89	2.00	3.89	2.00	3.88	2.01	0 %	0 %	0.26%	0.5%
0.86	4.89	0.85	4.89	0.83	4.92	1.16 %	0 %	3.4 %	0.62 %
1.21	4.28	1.20	4.27	1.15	4.34	0.82 %	0.23 %	4.95 %	1.4 %
1.78	3.88	1.77	3.87	1.68	3.97	0.56 %	0.26 %	5.61 %	2.32 %
2.08	3.38	2.07	3.38	2.14	3.33	0.48 %	0 %	2.88 %	1.48 %

Medición tiempo de conversión ADC por canal

Tiempo de conversión del ADC para el canal A: 97.6 uS.

Tiempo de conversión del ADC para el canal B: 97.6 uS.

Medición tiempo de transmisión SPI por canal

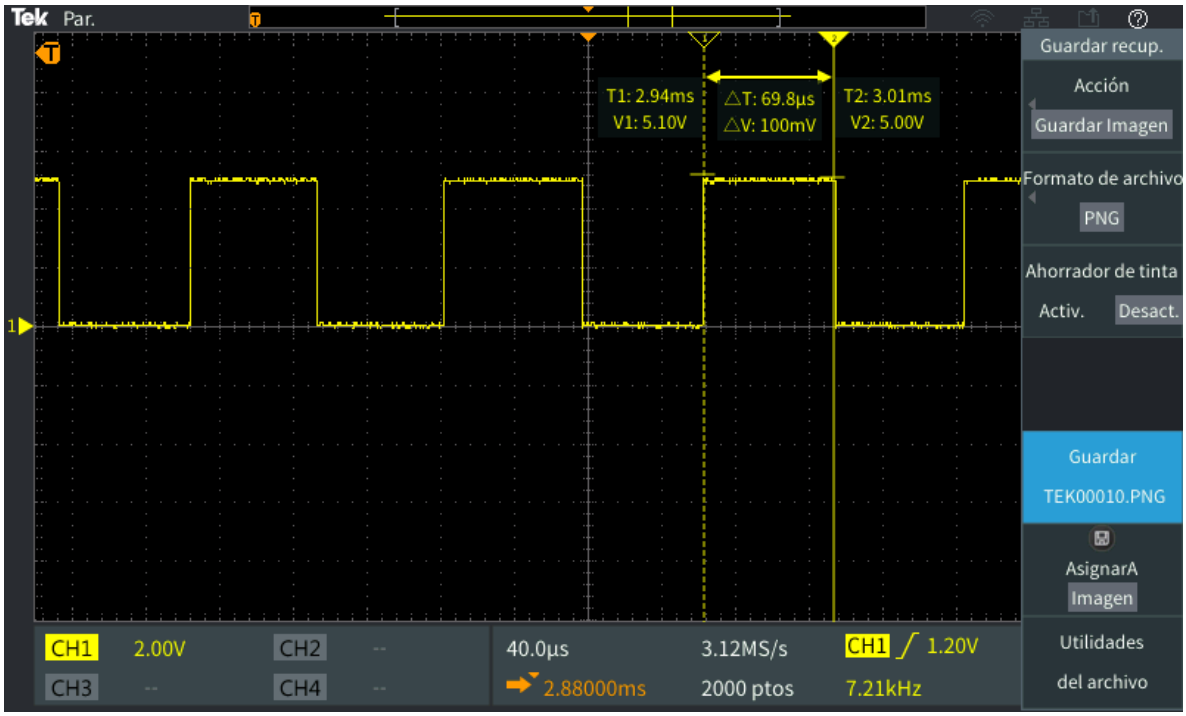


Imagen [1]. Tiempo de transmisión SPI.

Medición del tiempo de transmisión SPI para el canal A: 69.8 uS.

Medición del tiempo de transmisión SPI para el canal B: 69.8 uS.

Medición tiempo de transmisión UART por canal

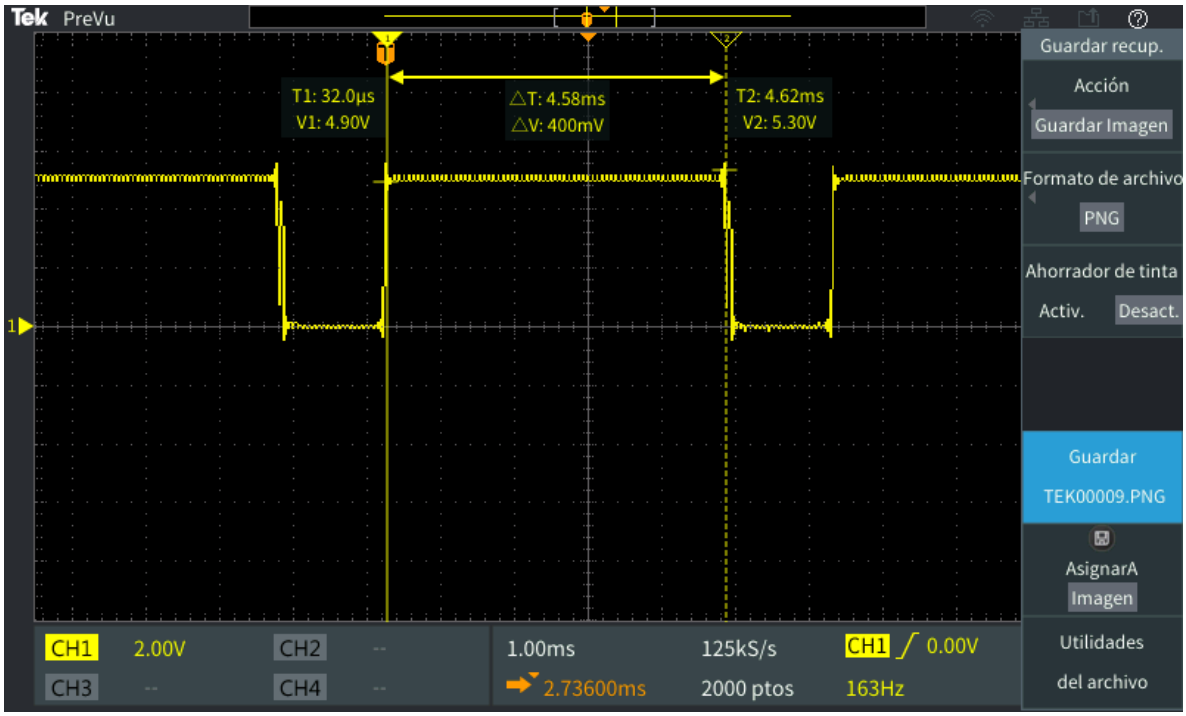


Imagen [2]. Tiempo de transmisión UART.

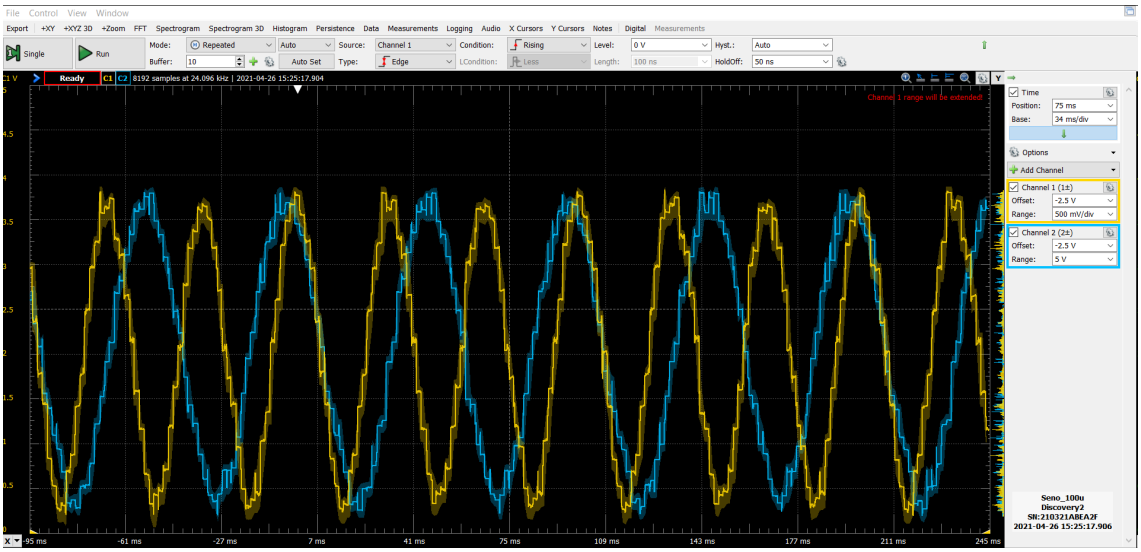
Medición del tiempo de transmisión UART para el canal A: 4.88 mS.

Medición del tiempo de transmisión UART para el canal B: 4.88 mS.

🎬 Con una señal sinusoidal de entrada por canal (dos frecuencias similares) que cumplan con criterio de Nyquist en el menor tiempo de muestreo (100us), escoger 10 tiempos de muestreo escritos por UART entre el mínimo y el máximo, y mostrar la señal reconstruida a la salida del DAC y en la gráfica del archivo CSV que se adquiere por el puerto serial.

Para un tiempo de muestreo de 100 us se obtuvieron las siguientes reconstrucciones:

a.



b.

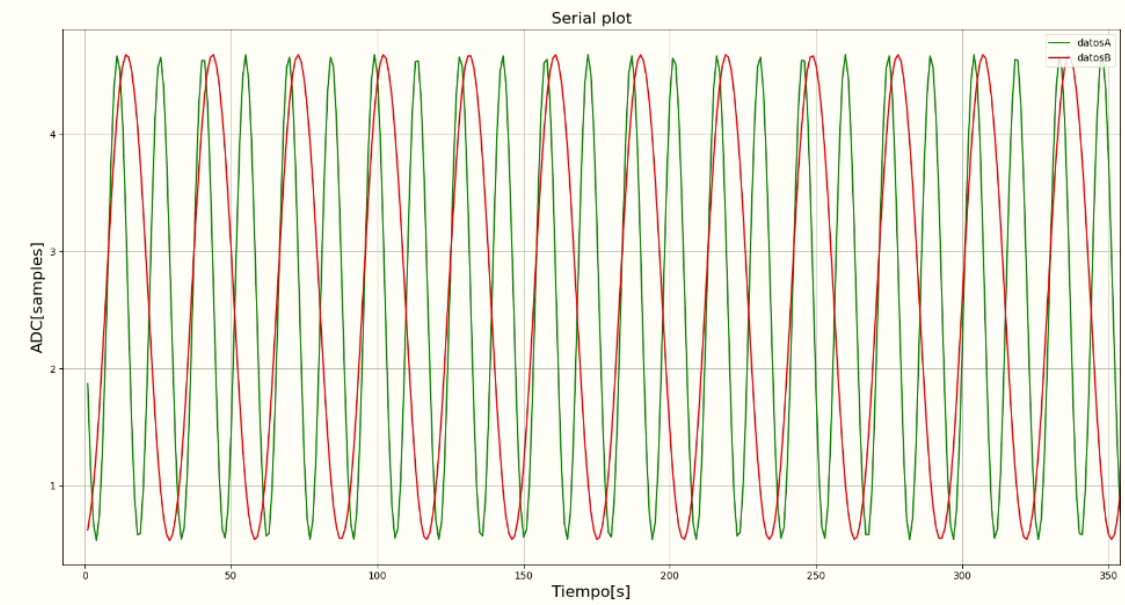
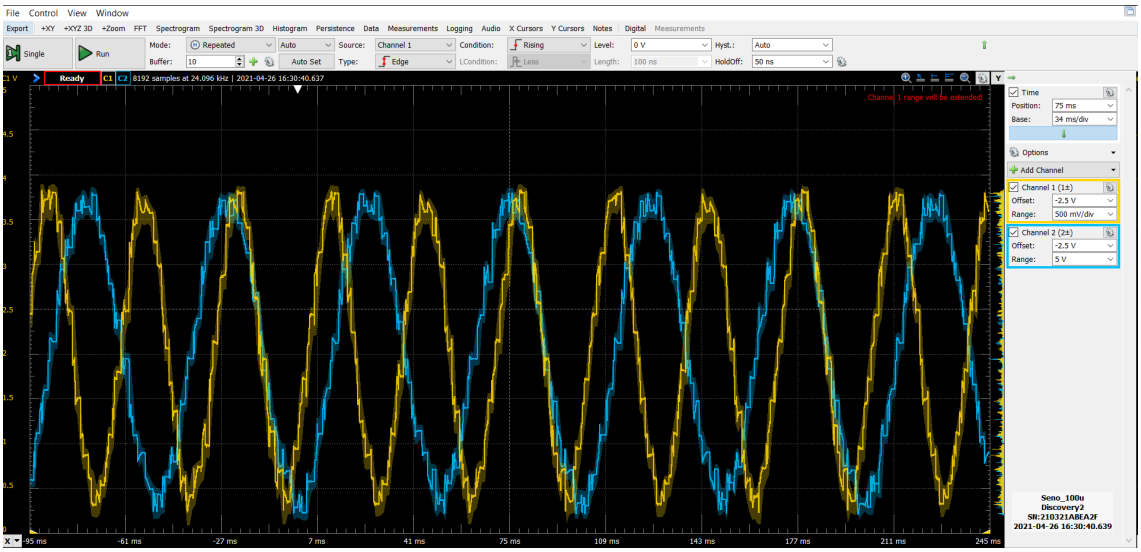


Imagen [3]. a. Reconstrucción a la salida del DAC de señales con un tiempo de muestreo de 100 us con tiempo de muestreo efectivo de 100 us. b. Reconstrucción por el serial de UART de señales con un tiempo de 100 us.

a.



b.

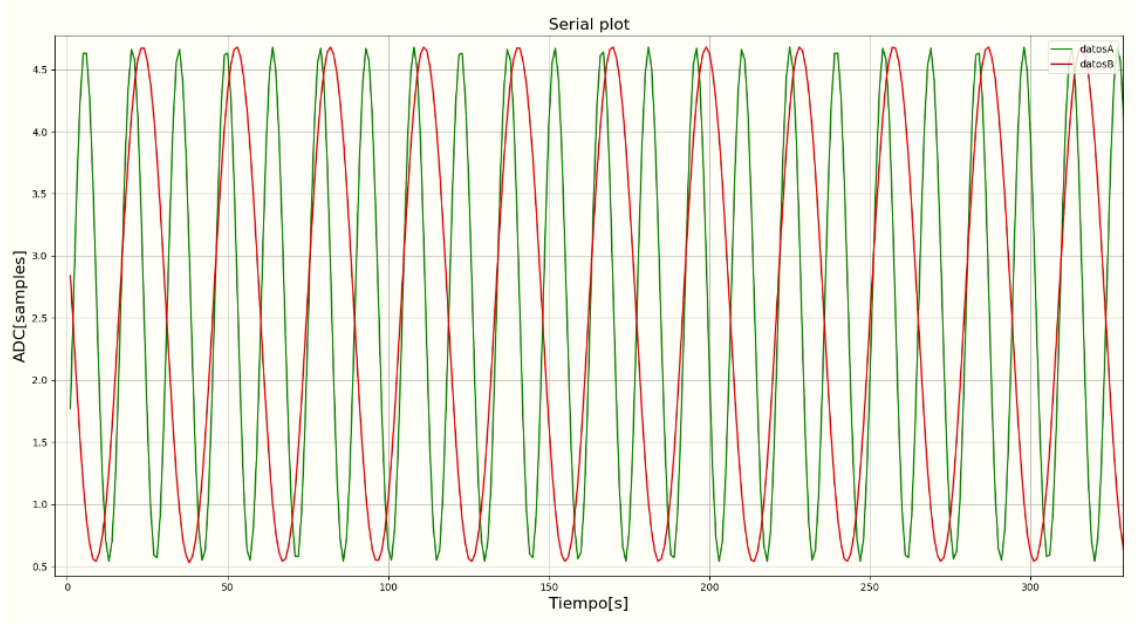
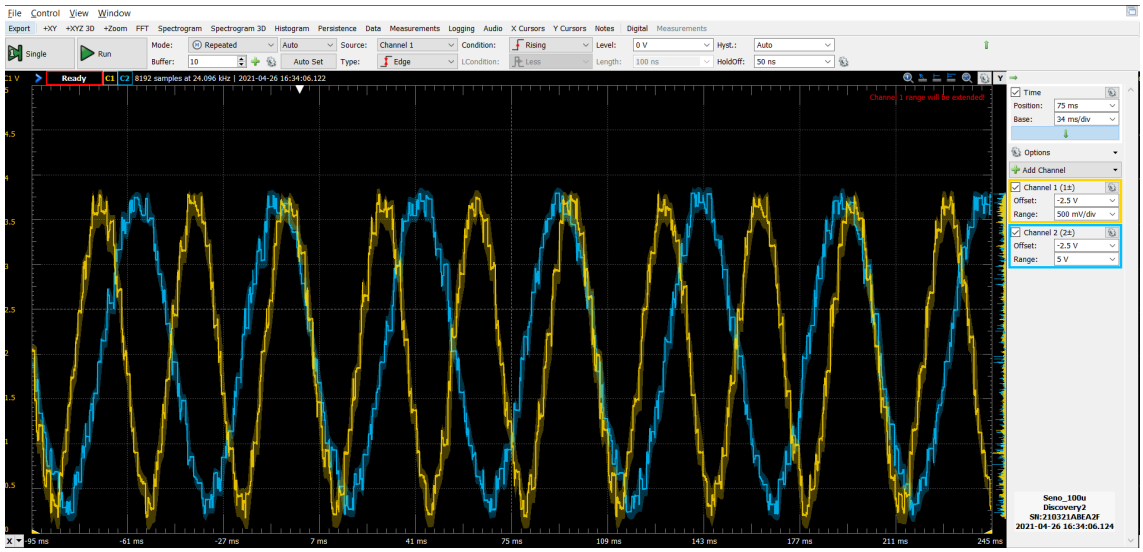


Imagen [4]. a. Reconstrucción a la salida del DAC de señales con un tiempo de muestreo de 300 us con tiempo de muestreo efectivo de 296 us. b. Reconstrucción por el serial de UART de señales con un tiempo de 300 us.

a.



b.

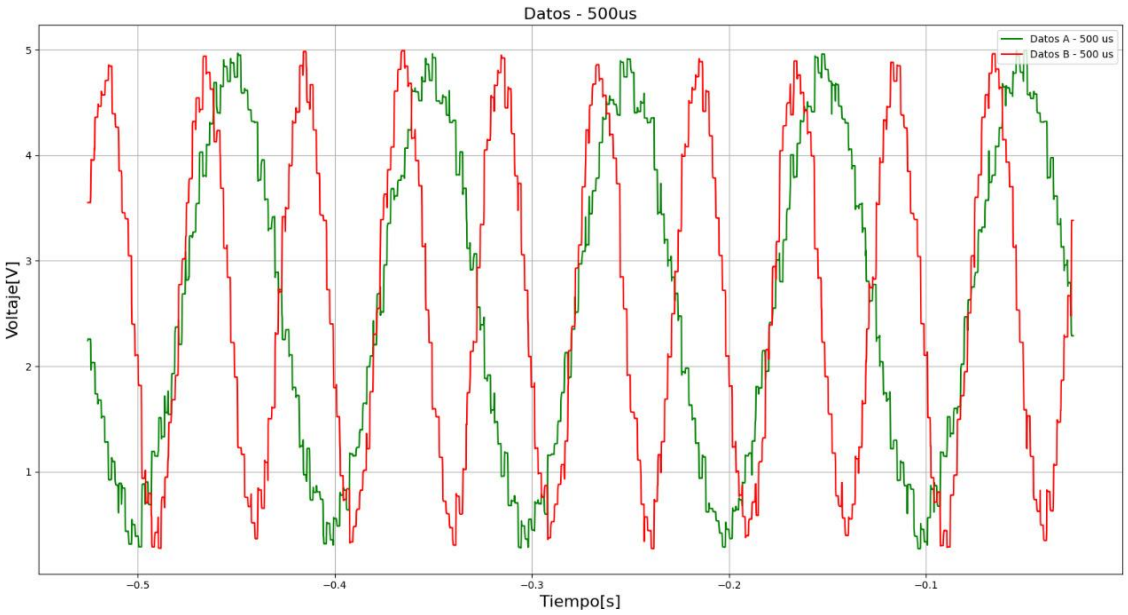
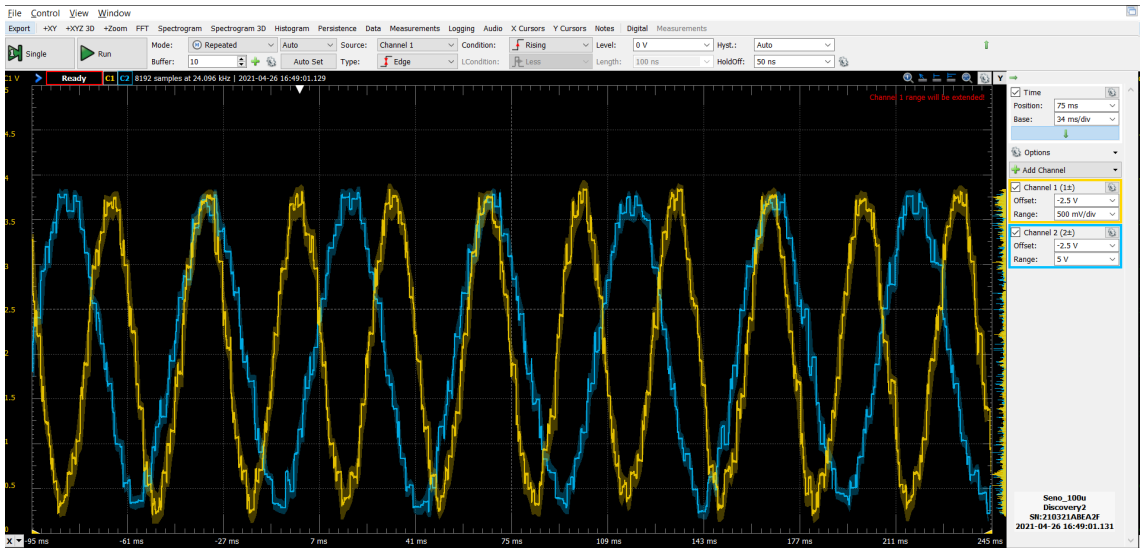


Imagen [5]. a. Reconstrucción a la salida del DAC de señales con un tiempo de muestreo de 500 us con tiempo de muestreo efectivo de 494 us. b. Reconstrucción por el serial de UART de señales con un tiempo de 500 us.

a.



b.

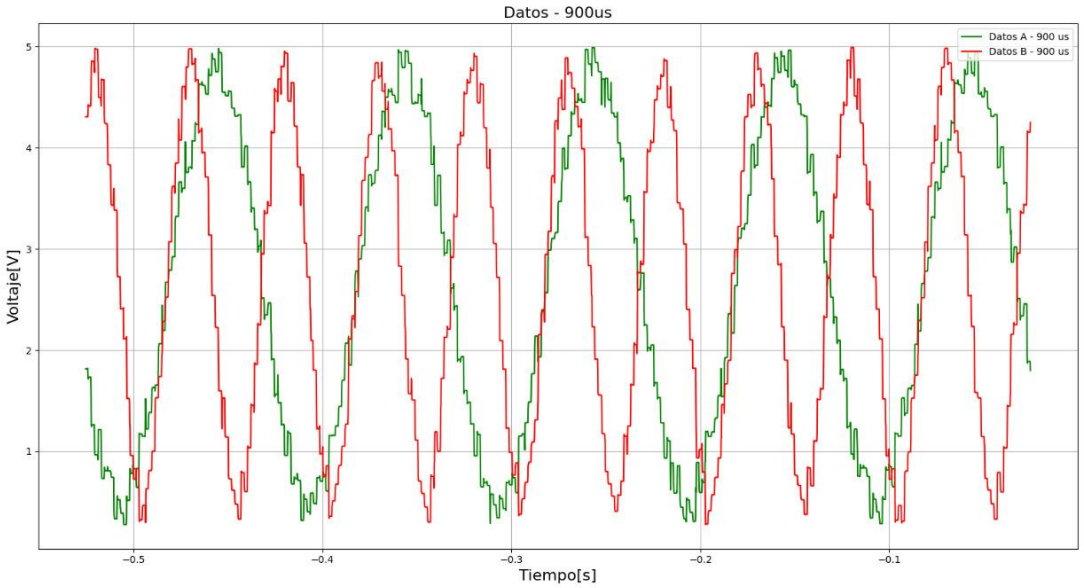
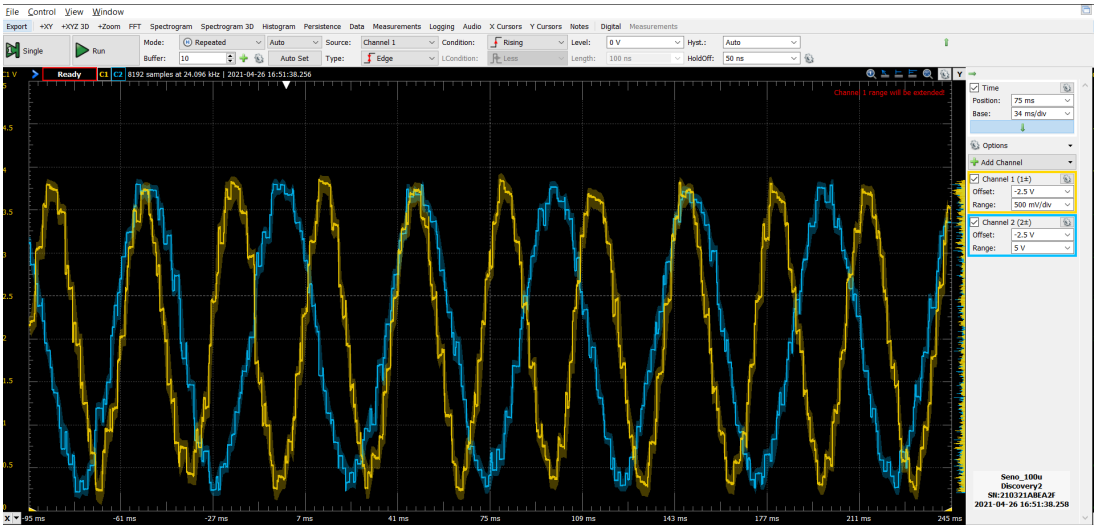
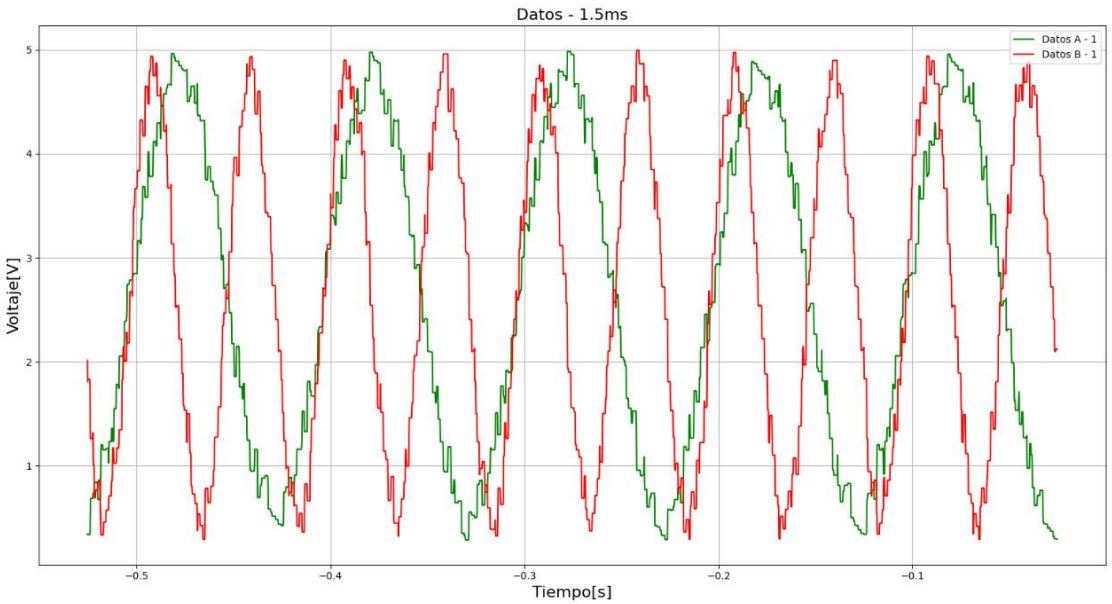


Imagen [6]. a. Reconstrucción a la salida del DAC de señales con un tiempo de muestreo de 900 us con tiempo de muestreo efectivo de 892 us. b. Reconstrucción por el serial de UART de señales con un tiempo de 900 us.

a.

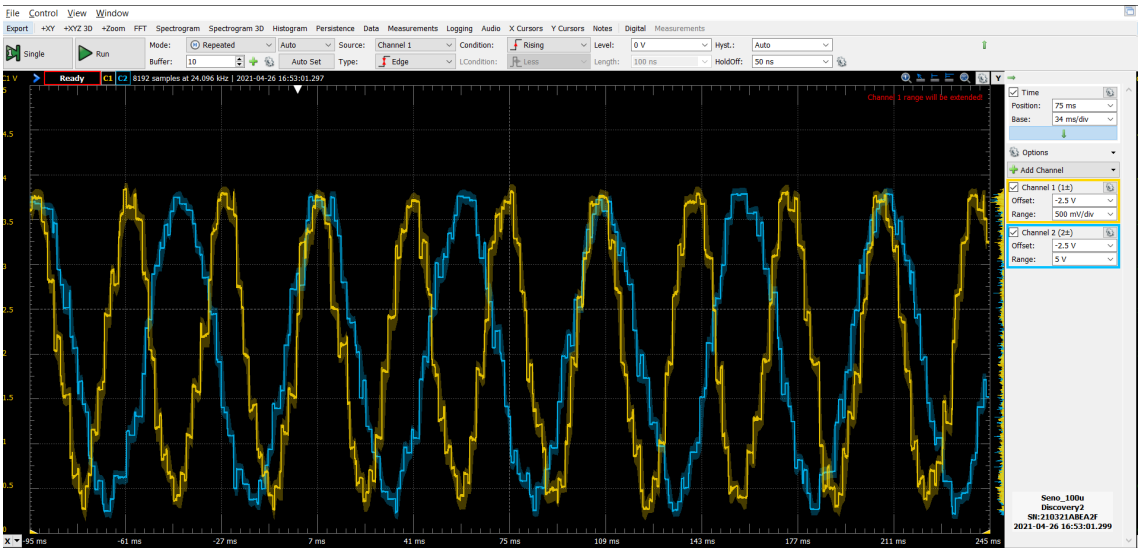




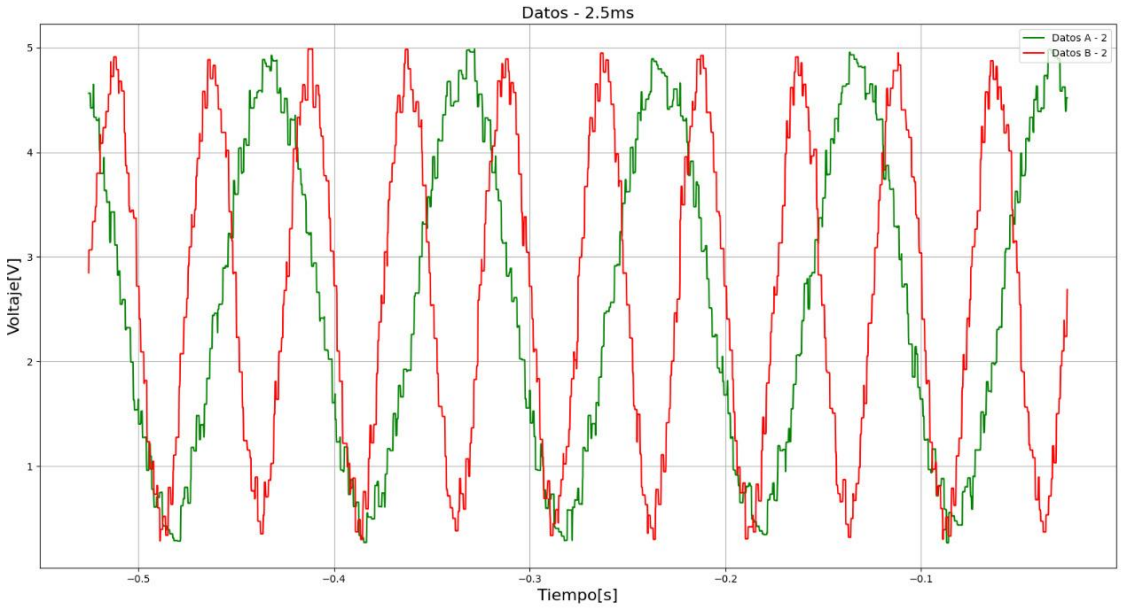
b.

Imagen[7]. a. Reconstrucción a la salida del DAC de señales con un tiempo de muestreo de 1.5 ms con tiempo de muestreo efectivo de 1.488 ms. b. Reconstrucción por el serial de UART de señales con un tiempo de 1.5 ms.

a.

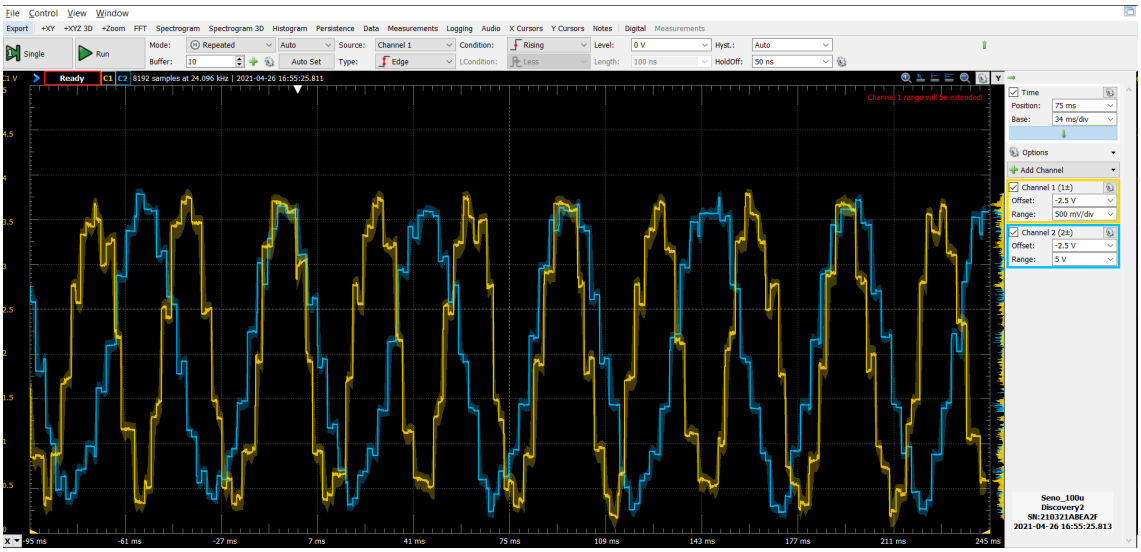


b.



Imagen[8]. a. Reconstrucción a la salida del DAC de señales con un tiempo de muestreo de 2.5 ms con tiempo de muestreo efectivo de 2.482 ms. b. Reconstrucción por el serial de UART de señales con un tiempo de 2.5 ms.

a.



b.

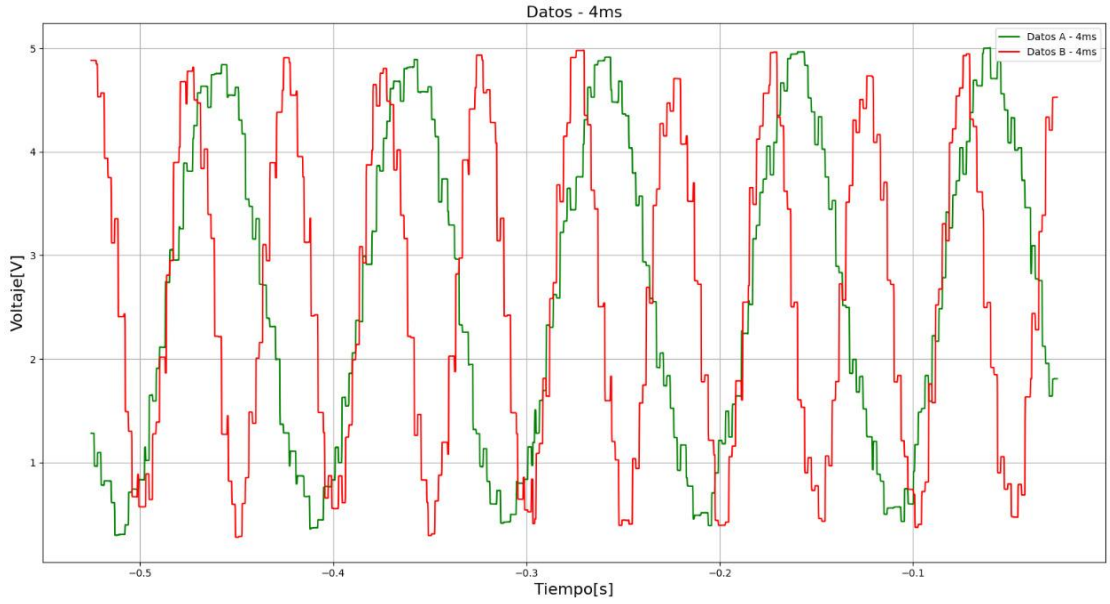
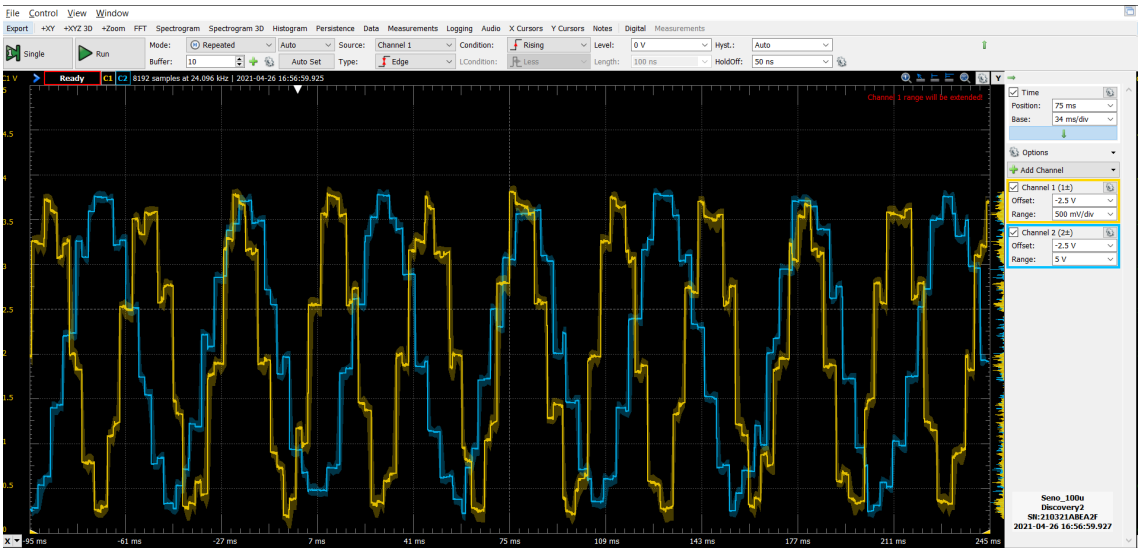


Imagen [9]. a. Reconstrucción a la salida del DAC de señales con un tiempo de muestreo de 4 ms con tiempo de muestreo efectivo de 3.972 ms. b. Reconstrucción por el serial de UART de señales con un tiempo de 4 ms.

a.



b.

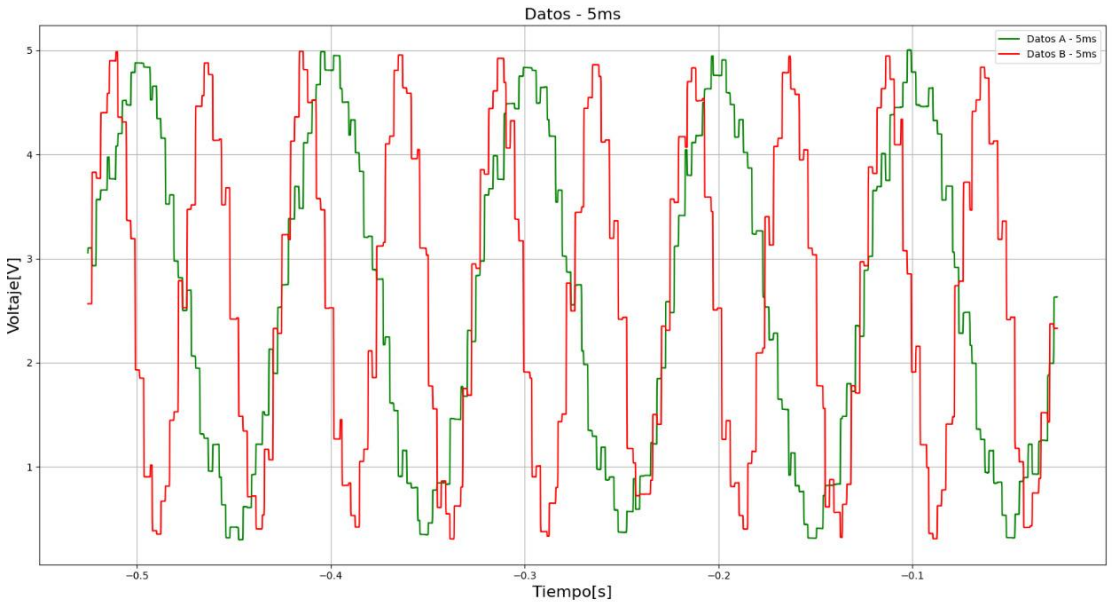
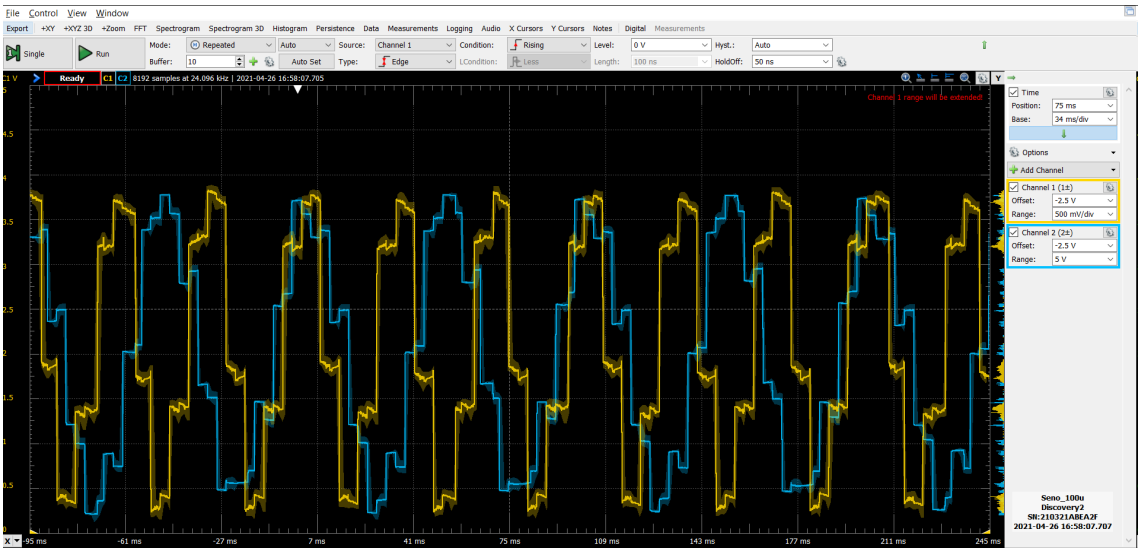


Imagen [10]. a. Reconstrucción a la salida del DAC de señales con un tiempo de muestreo de 5 ms con tiempo de muestreo efectivo de 4.968 ms. b. b. Reconstrucción por el serial de UART de señales con un tiempo de 5 ms.

a.



b.

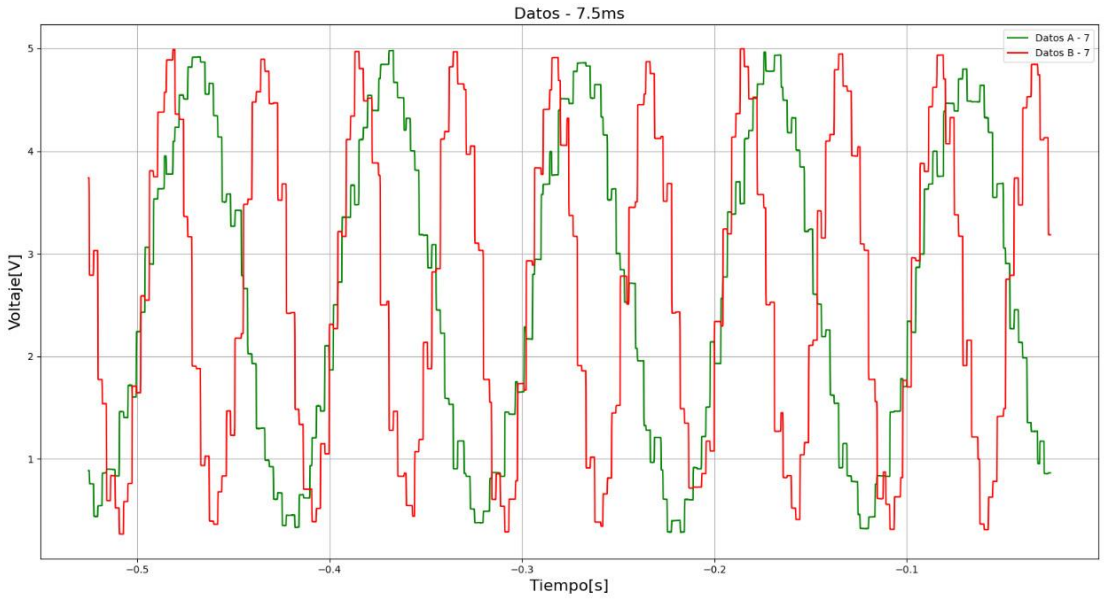
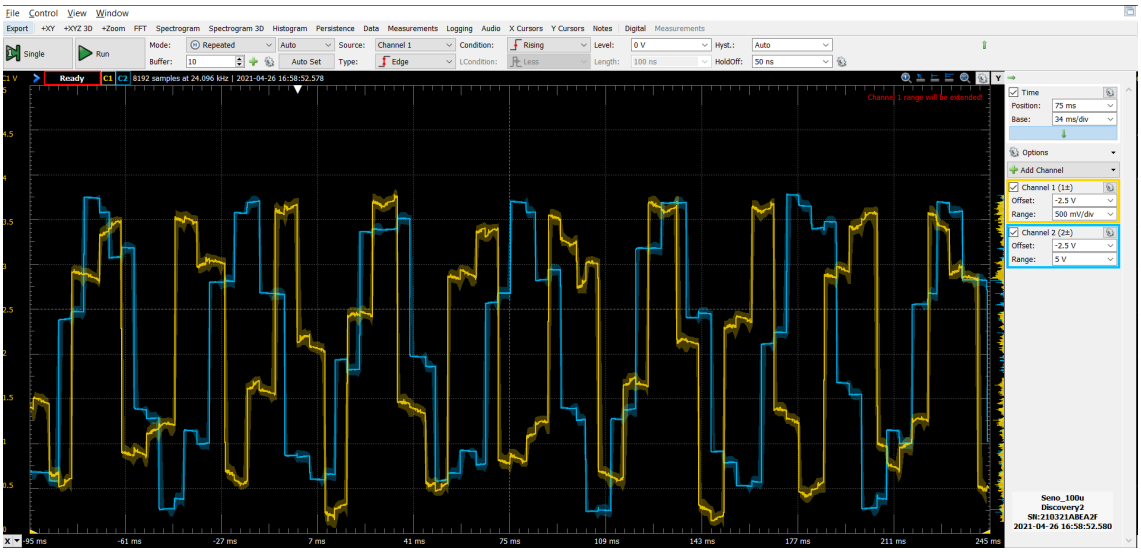


Imagen [11]. a. Reconstrucción a la salida del DAC de señales con un tiempo de muestreo de 7.5 ms con tiempo de muestreo efectivo de 7.432 ms. b. Reconstrucción por el serial de UART de señales con un tiempo de 7.5 ms.

a.



b.

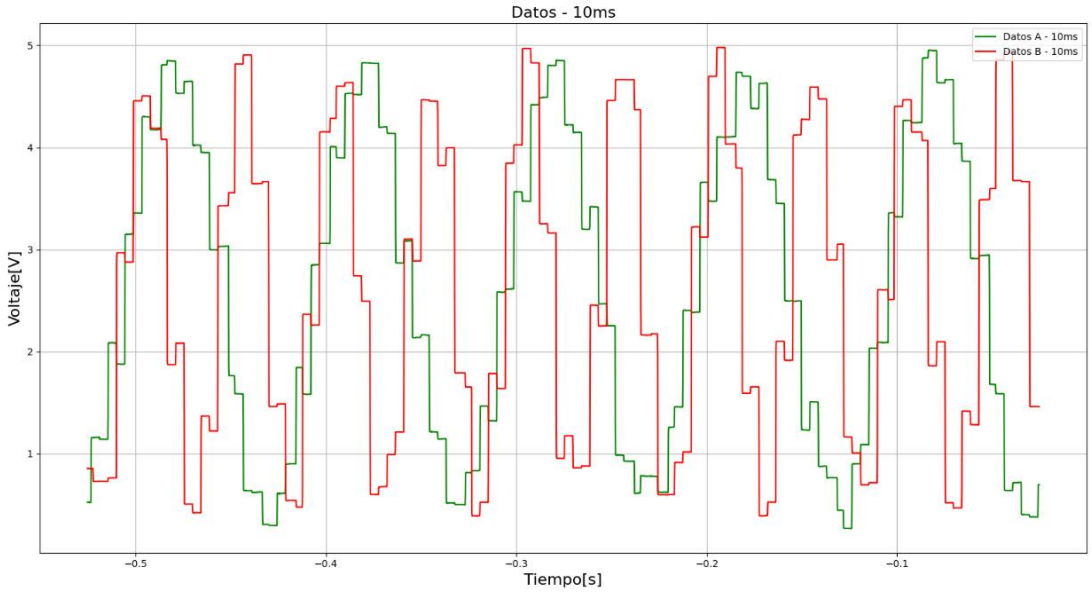
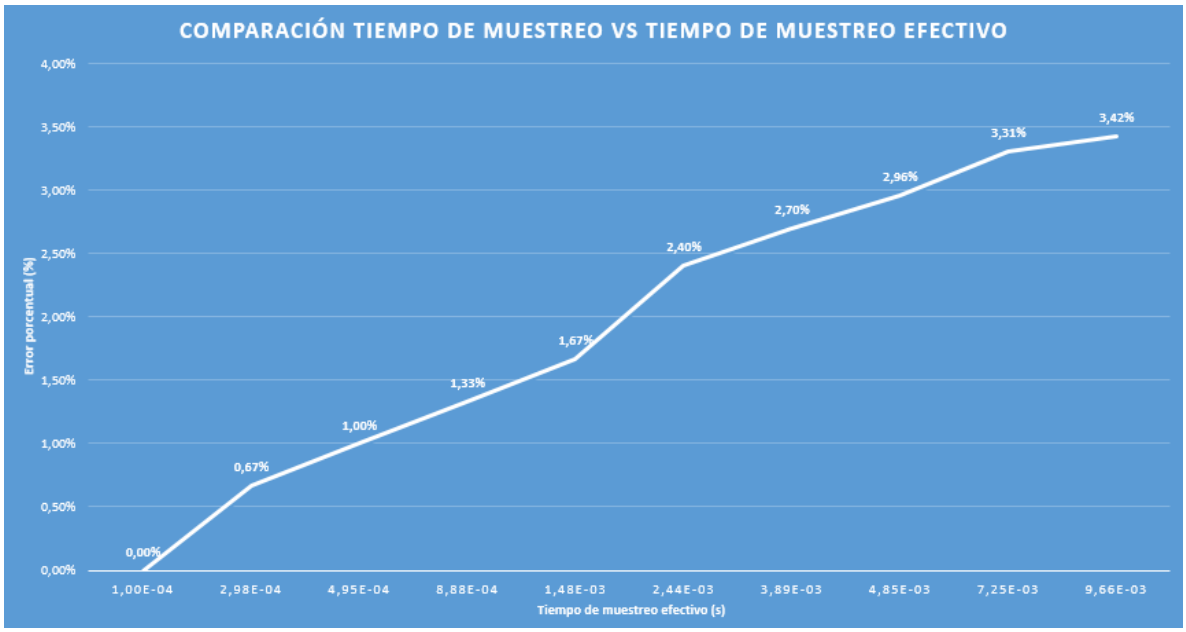


Imagen [12]. a. Reconstrucción a la salida del DAC de señales con un tiempo de muestreo de 10 ms con tiempo de muestreo efectivo de 9.968 ms. b. Reconstrucción por el serial de UART de señales con un tiempo de 10 ms.



Muestreo teórico	Muestreo efectivo	Diferencia	Error porcentual
1,00E-04	1,00E-04	0,00E+00	0,00%
3,00E-04	2,98E-04	2,00E-06	0,67%
5,00E-04	4,95E-04	5,00E-06	1,00%
9,00E-04	8,88E-04	1,20E-05	1,33%
1,50E-03	1,48E-03	2,50E-05	1,67%
2,50E-03	2,44E-03	6,00E-05	2,40%
4,00E-03	3,89E-03	1,08E-04	2,70%
5,00E-03	4,85E-03	1,48E-04	2,96%
7,50E-03	7,25E-03	2,48E-04	3,31%
1,00E-02	9,66E-03	3,42E-04	3,42%

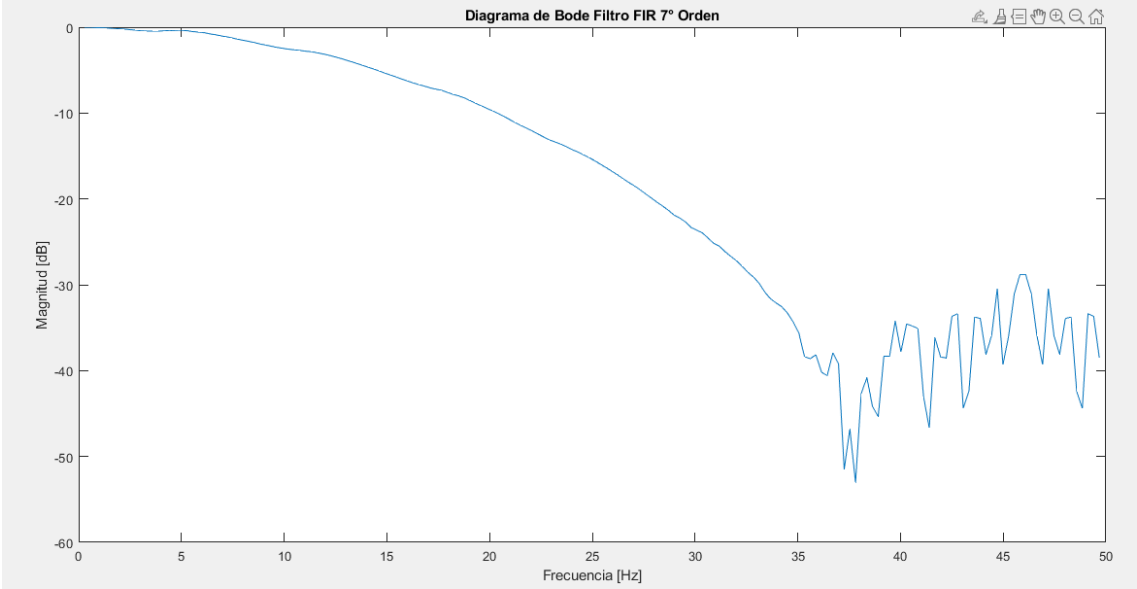
Imagen [13]. Comparación error porcentual vs tiempo de muestreo efectivo con su tabla de datos respectiva.

Para la sección de los filtros de la práctica 2B:

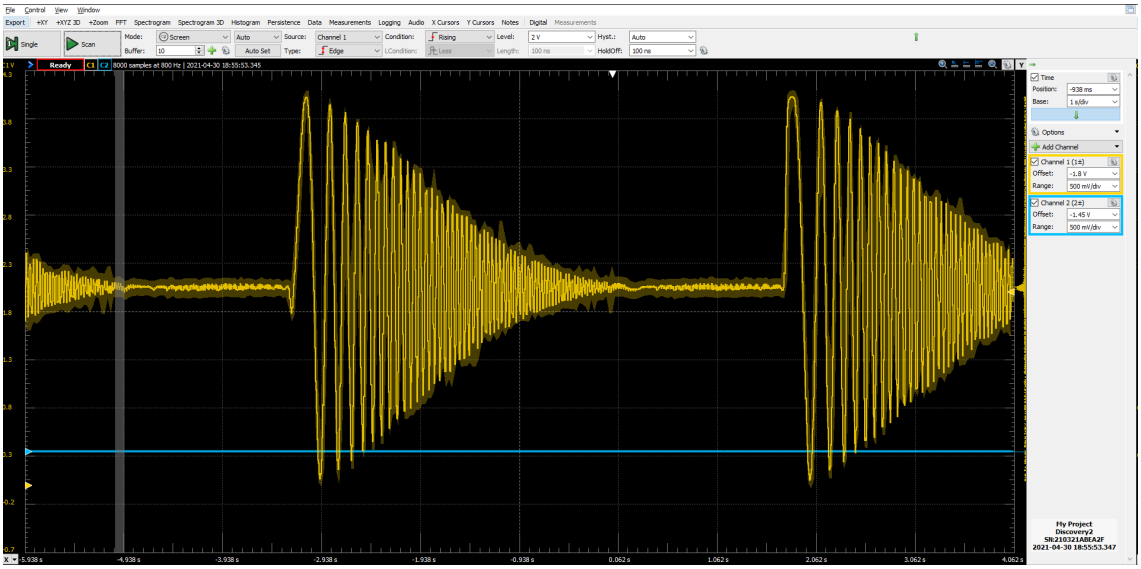
Respuesta en tiempo y diagrama de bode de los filtros.

Filtros FIR:

a.



b.



c.

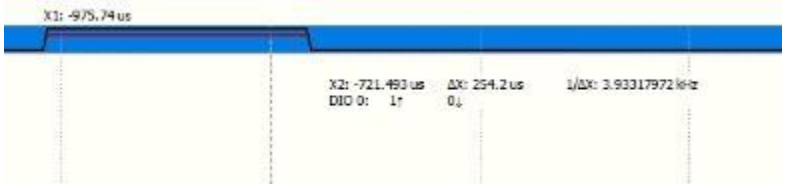
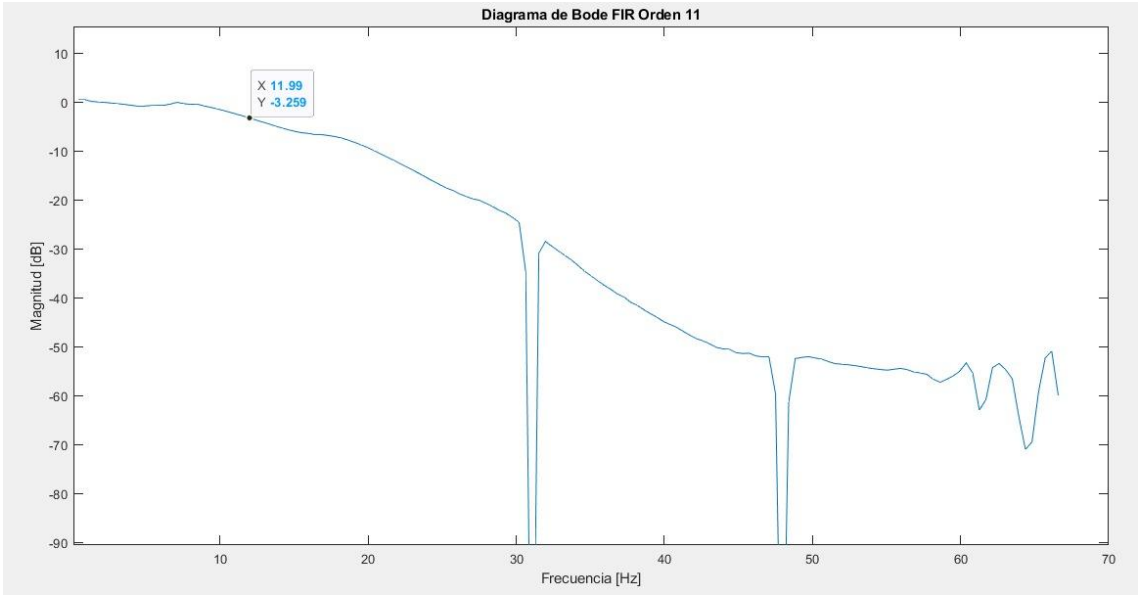
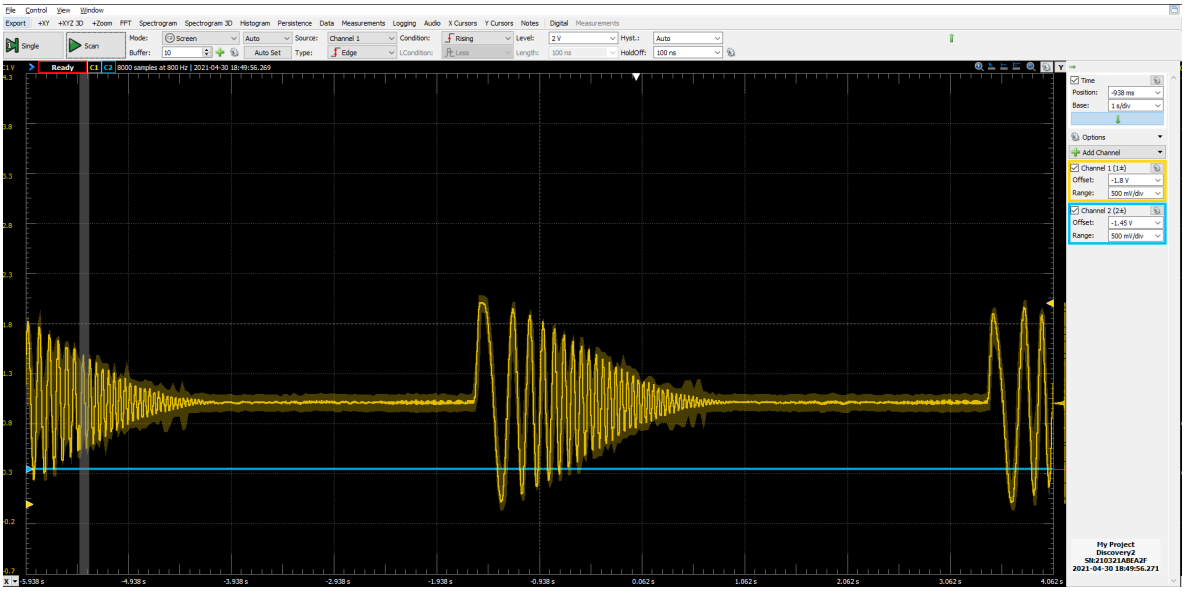


Imagen [14]. Filtro FIR Orden 7: a. Diagrama de Bode realizado por UART. b. Respuesta en tiempo transmitida por el DAC. c. Tiempo de ejecución del filtro.

a.



b.



c.

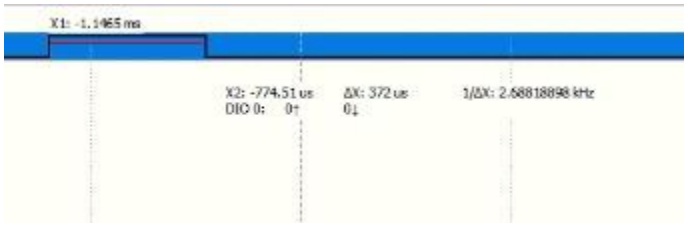
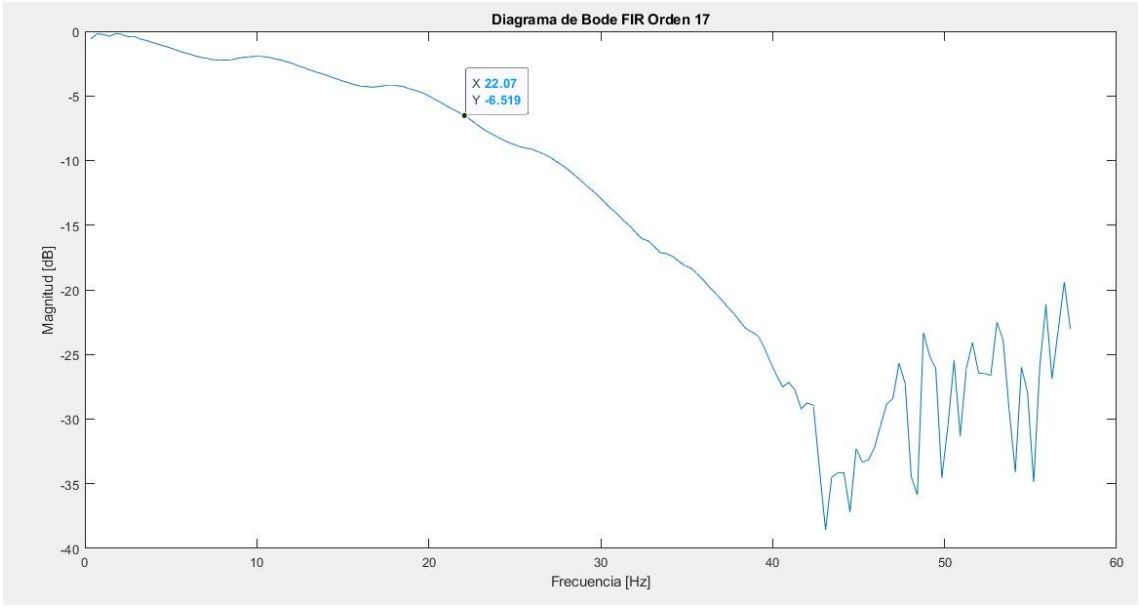
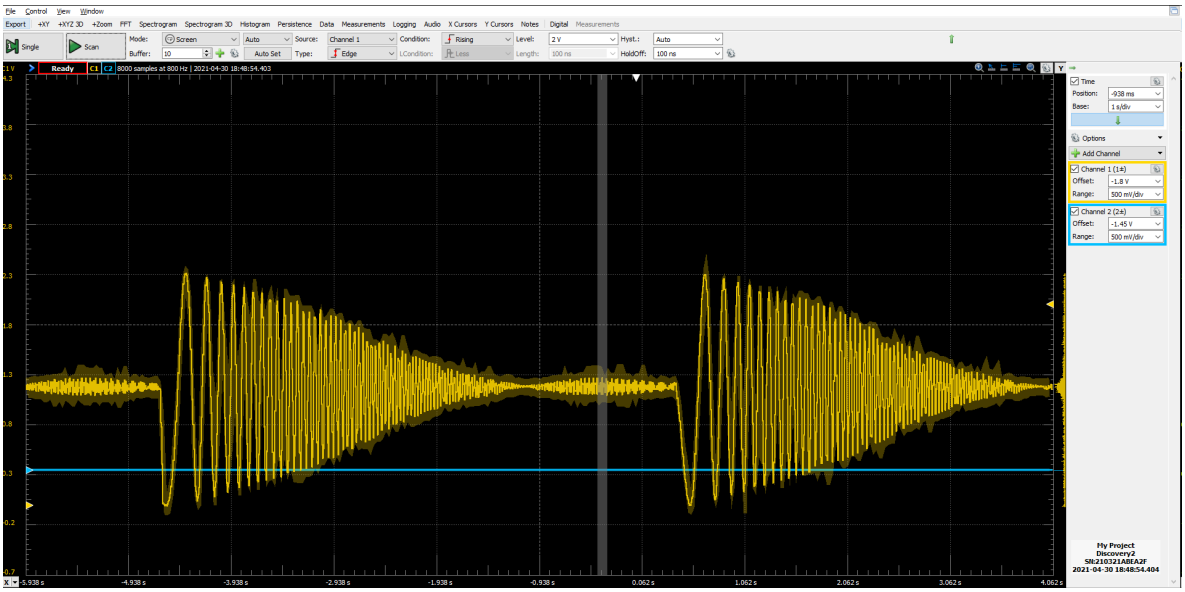


Imagen [15]. Filtro FIR Orden 11: a. Diagrama de Bode realizado por UART. b. Respuesta en tiempo transmitida por el DAC. c. Tiempo de ejecución del filtro.

a.



b.



c.

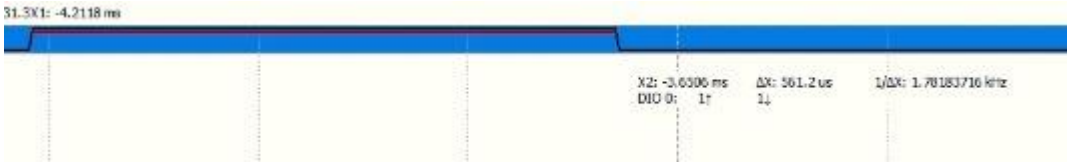
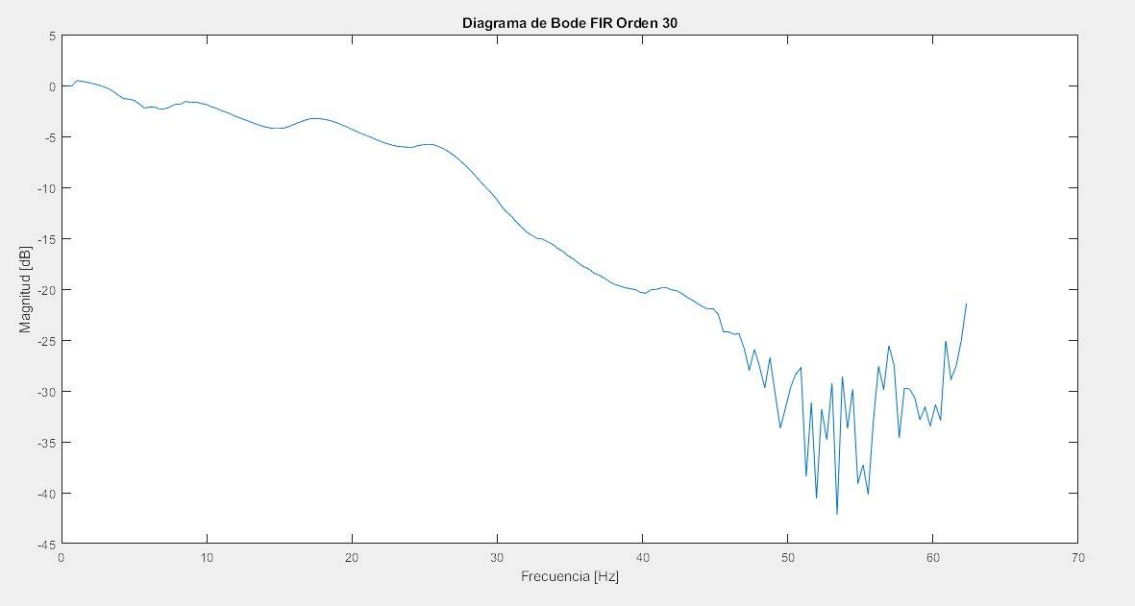
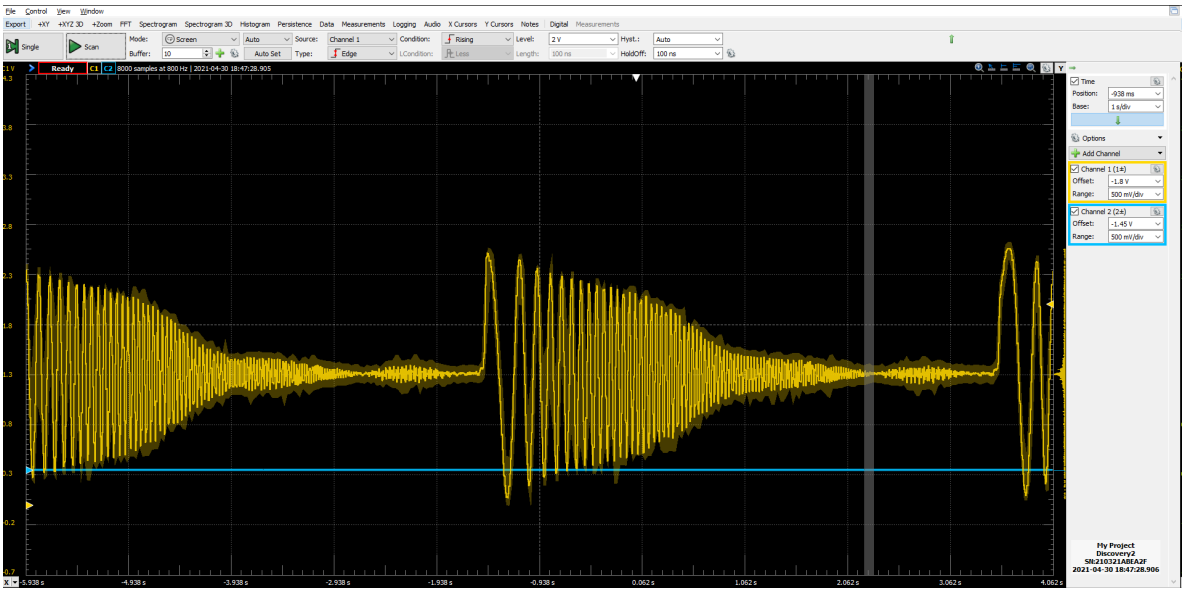


Imagen [16]. Filtro FIR Orden 17: a. Diagrama de Bode realizado por UART. b. Respuesta en tiempo transmitida por el DAC. c. Tiempo de ejecución del filtro.

a.



b.



c.

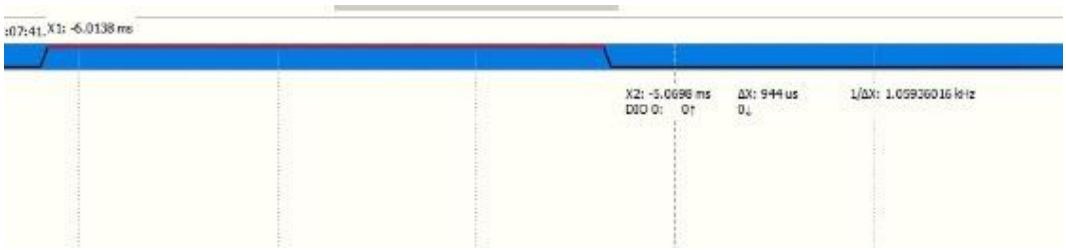
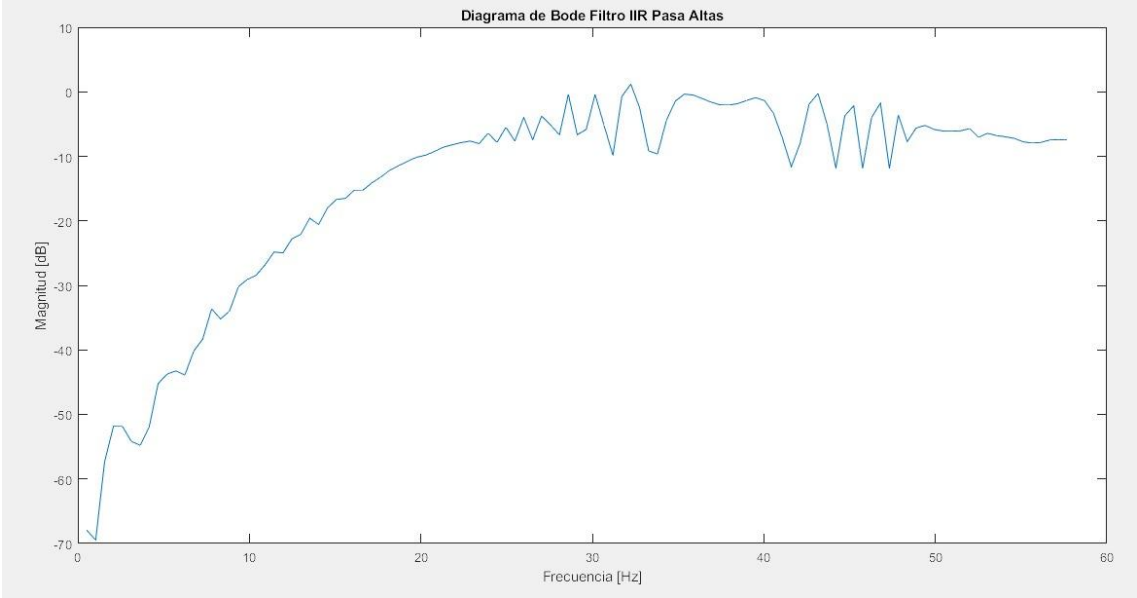


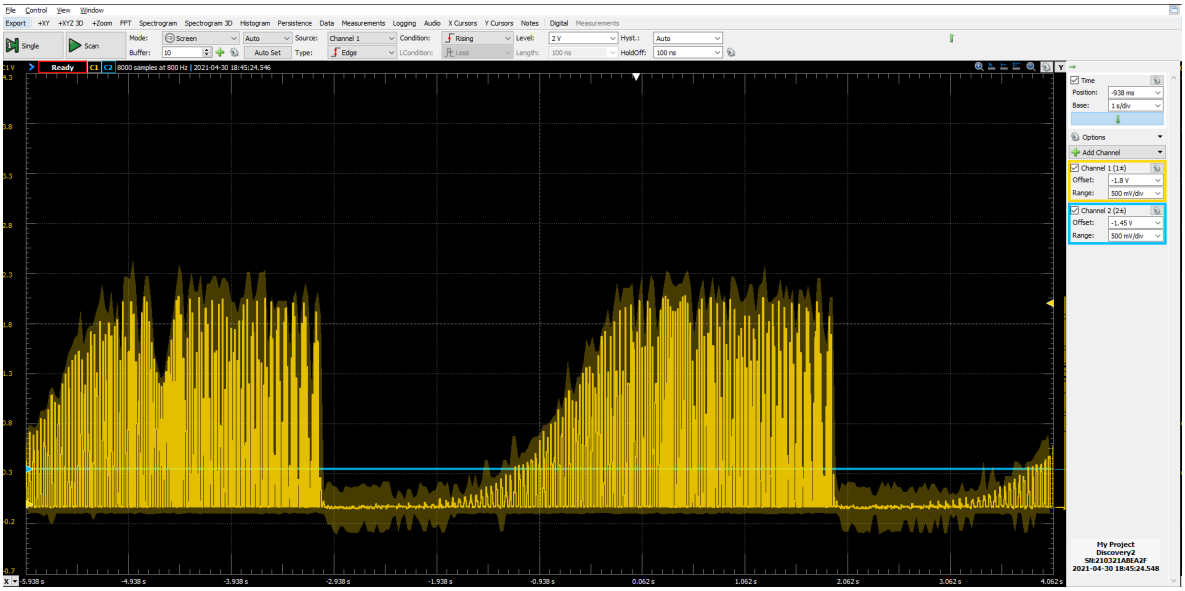
Imagen [17]. Filtro FIR Orden 30: a. Diagrama de Bode realizado por UART. b. Respuesta en tiempo transmitida por el DAC. c. Tiempo de ejecución del filtro.

a.



Filtros IIR:

b.



c.

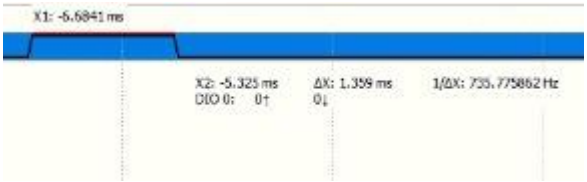
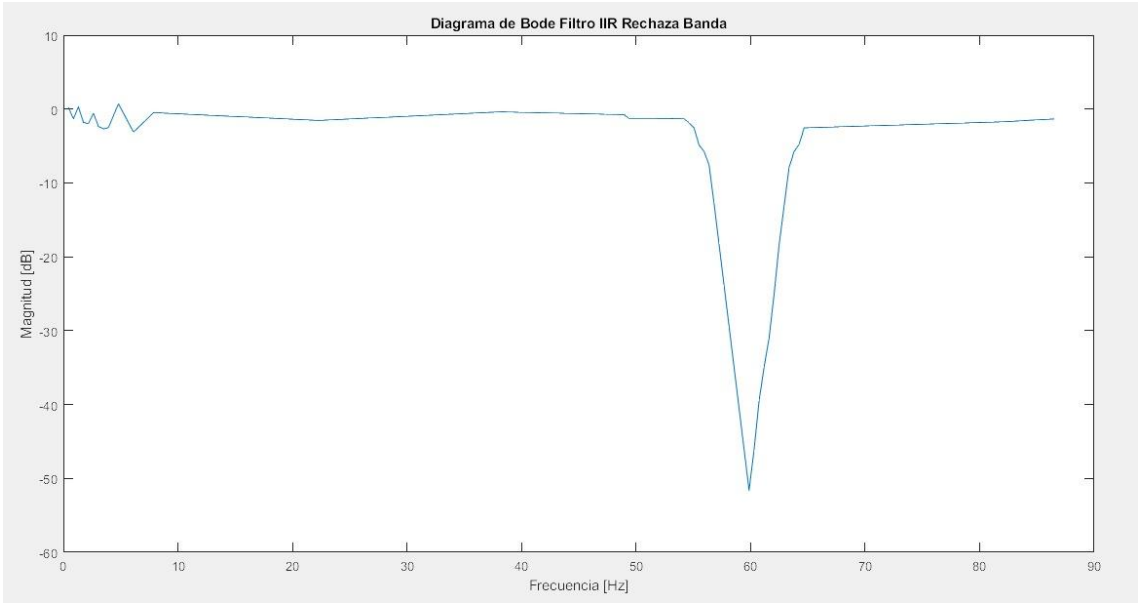
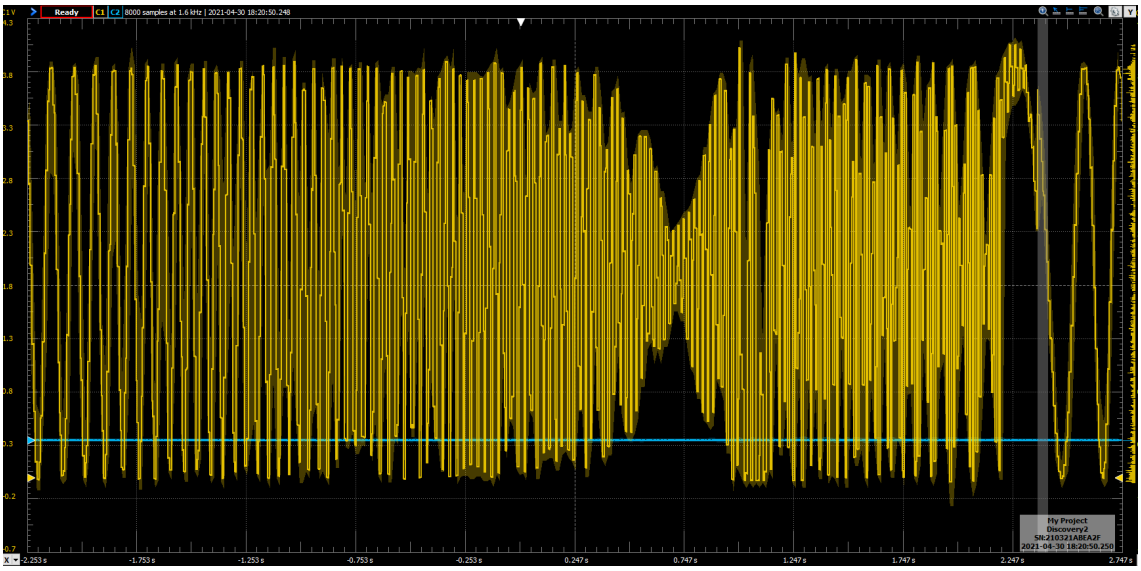


Imagen [18]. Filtro IIR Pasa altas: a. Diagrama de Bode realizado por UART. b. Respuesta en tiempo transmitida por el DAC. c. Tiempo de ejecución del filtro.

a.



b.



c.

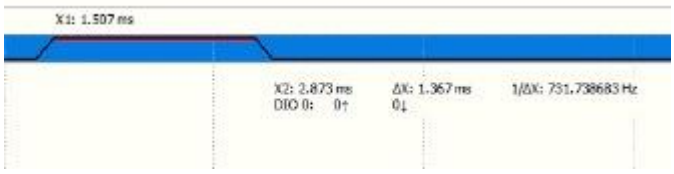
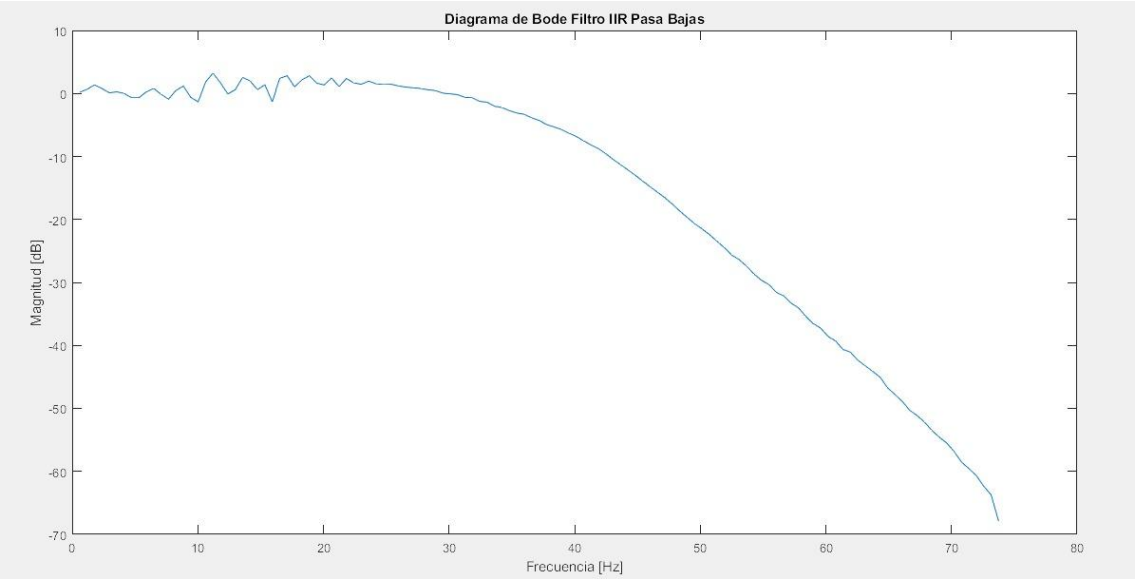
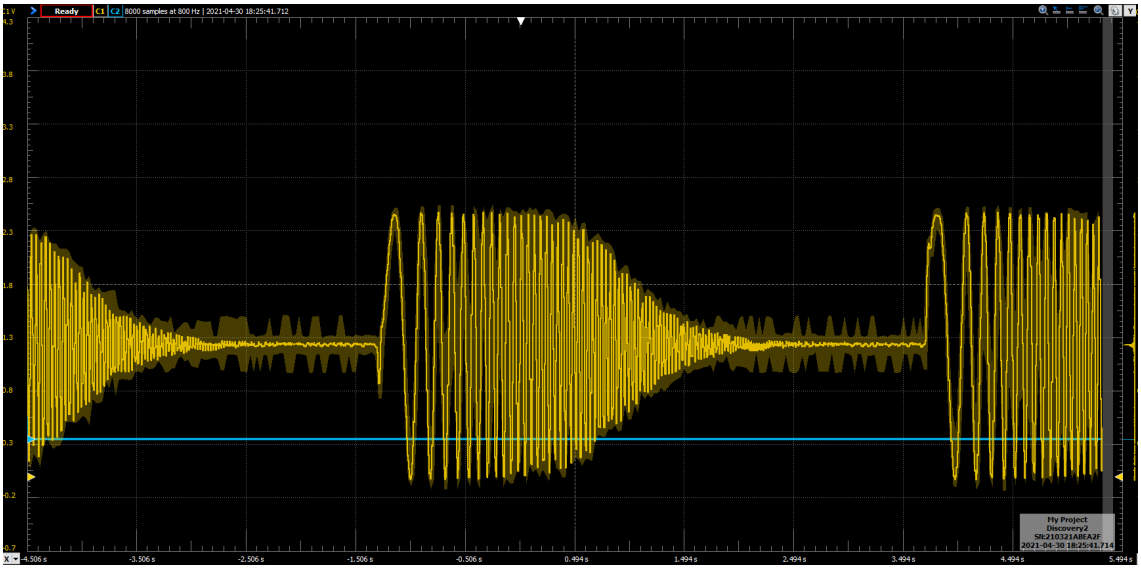


Imagen [19]. Filtro IIR Rechaza banda: a. Diagrama de Bode realizado por UART. b. Respuesta en tiempo transmitida por el DAC. c. Tiempo de ejecución del filtro.

a.



b.



c.

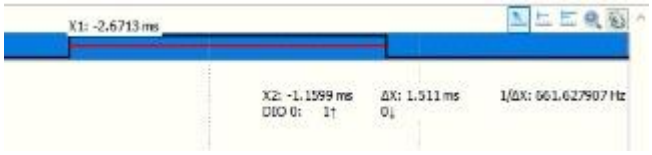


Imagen [20]. Filtro IIR Pasa bajas: a. Diagrama de Bode realizado por UART. b. Respuesta en tiempo transmitida por el DAC. c. Tiempo de ejecución del filtro.

Análisis de resultados:

A lo largo del desarrollo del laboratorio fue posible evidenciar que el algoritmo pensado inicialmente se fue transformado debido a los errores encontrados durante el desarrollo de la práctica y es por esto que se analizarán los cambios hechos, los errores que aún persisten y las posibles soluciones a estos problemas:

Timer y tiempo de muestreo:

Al realizar la toma de medidas se comprobaron dos errores con respecto al timer y a el tiempo de muestreo: Para el caso del timer, la manera en que se implementó la interrupción se genera a 4 us que no es un divisor de 50 us, lo cual genera que desde el inicio se tenga un error de ± 2us. Para solucionar esto se debe seleccionar un reloj y un prescaler que juntos dan como resultado un divisor de 50us.

Para el caso del tiempo de muestreo, al aumentar los tiempos de muestreo debido a el valor que proviene de la UART, es posible evidenciar un error que poco a poco se va haciendo más grande entre mayor sea el valor que se quiere obtener, como se puede evidenciar en la gráfica de la imagen 13. Este error se debe a una acumulacion de pequeños errores entre conteos, los cuales son cada vez más grandes, debido al aumento en el valor de "count_to", esta es la variable que define el tiempo de muestreo. Cada conteo del timer hasta el overflow puede tener errores provenientes de varias fuentes. Esta acumulación se le conoce como error cuadrático medio. En la implementación se decidió priorizar el tiempo de ejecución sobre la corrección del error puesto que ya se tenían varias tareas que consumen mucho tiempo en el programa principal.

Transmisión UART:

En las mediciones de la señal seno por UART fue posible evidenciar que debido a la tasa de baudios utilizada (19200) y a la manera en la cual se realiza la transmisión, el tiempo que consume esta es demasiado alta, lo que provocaba que no se puedan enviar señales con periodo mayor al tiempo que se demora la UART en enviar toda la trama que corresponde a 4.58ms, en consecuencia a esto, no se pueden enviar por la UART señales que tengan una frecuencia mayor a aproximadamente 100Hz. El tiempo que demora la UART en enviar, depende de la cantidad de números a enviar y cuanto se demoran estos números en estar listos para ser enviados. El algoritmo implementado como se explicó previamente, convierte los valores de ADC en valores de voltaje y además se toman los dígitos por aparte, este último proceso requiere de varias multiplicaciones que aumentan el tiempo en que el resultado está listo para ser enviado. Como solución para este problema se puede evitar la conversión del número de ADC y además solo transmitir cuando sea necesario.

Transmisión SPI:

Debido a que la transmisión de SPI se hace por interrupción, y esta interrupción llega cada vez que el buffer de transmisión está vacío, el módulo de SPI siempre está transmitiendo sin importar si llegó un nuevo dato del ADC, esto hace que el tiempo dedicado al programa principal disminuya de manera significativa. Para corregir este problema existen dos posibles soluciones: la primera es apagar el módulo SPI si no es necesario realizar el envío de datos, esto hace que intrínsecamente se apague la interrupción, la segunda opción es usar el contador del SPI que decrementa al enviar un byte, y realizar la transmisión por polling, para esto hay que tener en cuenta que el contador no se puede leer por lo que no es posible saber si decrementa haciendo debugging.

Respecto de los resultados obtenidos: Como se puede observar en la tabla 1 los errores que se presentan entre las 3 mediciones son pequeños, ya que oscilan entre el 0 y el 4 %, lo cual nos señala que los códigos que se implementaron para dar solución a la problemática planteada, sin embargo, se puede evidenciar que la salida serial que se ve por la UART tiene menor precisión que la que se ve por el DAC que se conectó a través del SPI.

Para la reconstrucción por el DAC se observa en azul una señal de 20 Hz y en amarillo una de 30 Hz, se ve con un poco de error de cuantización, ya que esta reconstrucción de realizó con un módulo Analog Discovery 2 y tenemos ciertas restricciones al ver la señal que se lee. Aparte de esta se puede ver por los dos canales la lectura que se tiene por UART, y se puede decir que la señal que se recibe por UART se ve mejor que la que se ve por el DAC, por ejemplo, en la imagen [] donde está la señal con un tiempo de muestreo de 10 ms se ve que en UART las muestras tienen una señal sinusoidal más clara que por el DAC.

Como se puede ver resumido en la imagen [13], y a lo largo de todas las imágenes anteriores, el tiempo de muestreo efectivo respecto del tiempo de muestreo teórico no presenta una variación muy amplia, y como se puede apreciar en la misma imagen mencionada anteriormente esta diferencia presenta un error porcentual con un comportamiento exponencial, ya que este va subiendo poco a poco, del 0 al 3.42%, lo cual nos indica que no las muestras que se están perdiendo van aumentando con el tiempo de muestreo que se coloque.

Parte 2B:

Algunos de los errores anteriormente mencionados fueron encontrados al momento de realizar y analizar la parte 2B. En vista de que el tiempo de muestreo para los filtros era considerablemente alto (8.4ms para FIR y 5.2 ms para IIR) se evidenció que había más tiempo para la ejecución del programa principal, pero en el caso de que los filtros necesitaran de una frecuencia mayor, el programa empezaría a tener fallas como consecuencia de la demora en la ejecución de la función de filtrado, especialmente con filtros de orden superior. La solución de este problema radica en el módulo SPI, evitando que transmita todo el tiempo.

Respecto a la parte B del laboratorio, en el código se estaba perdiendo mucha resolución, dado que en ambos tipos de filtros se convertían los datos a entero y se perdían varios bits, en el caso de los FIR se perdían 20 bits y en el caso de los IIR se perdían 8 bits, por lo cual se modificó el código para evitar esta pérdida de resolución y como se ve en las imágenes de la 14 a la 20 la respuesta tanto en tiempo, como en frecuencia mejora mucho aunque en algunas la frecuencia de corte se

corre un poco, respecto de los diseños originales, mejoran mucho con respecto a las respuestas que se mostraron en la sustentación.

Las siguientes imágenes de la 21 a la 29, representan los cambios que se hicieron al código durante el proceso de implementación del laboratorio:

Para la parte A:

Como se aprecia en los diagramas de flujo siguientes existen cambios sustanciales en cuanto al flujo que se tuvo para la ejecución de cada uno de los casos que se presentaron.

Diagramas de flujo:

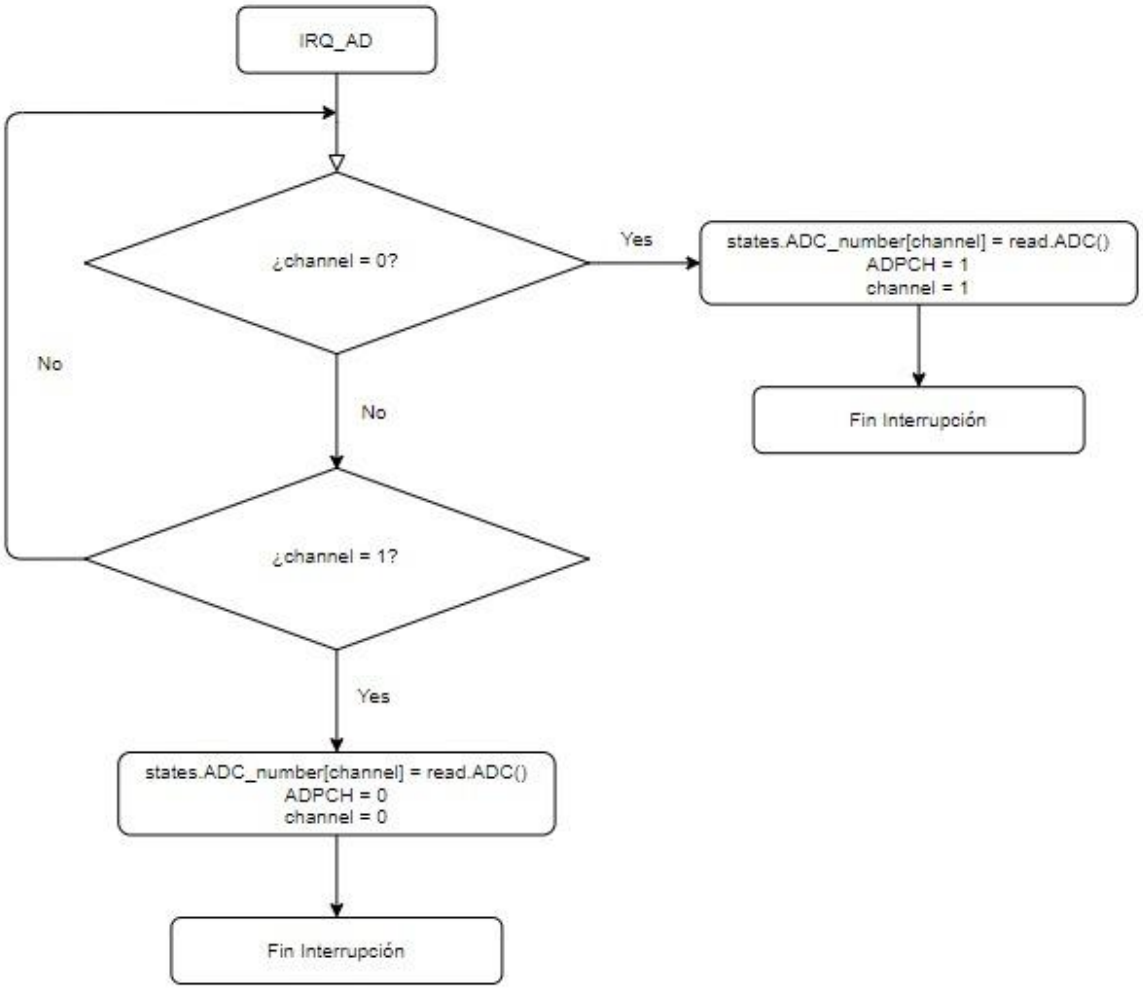


Imagen [21]. Rutina de interrupción para la conversión del ADC.

La imagen anterior nos muestra la ISR del ADC, en este caso su principal función es la multiplexación de los canales en los que ingresaban las dos señales para ser procesadas y de igual manera tomar el dato de salida del ADC y guardarlo en una variable que posteriormente será usada. La modificación con el diseño original prevalece en los condicionales y en el cambio de

canal por medio de software.

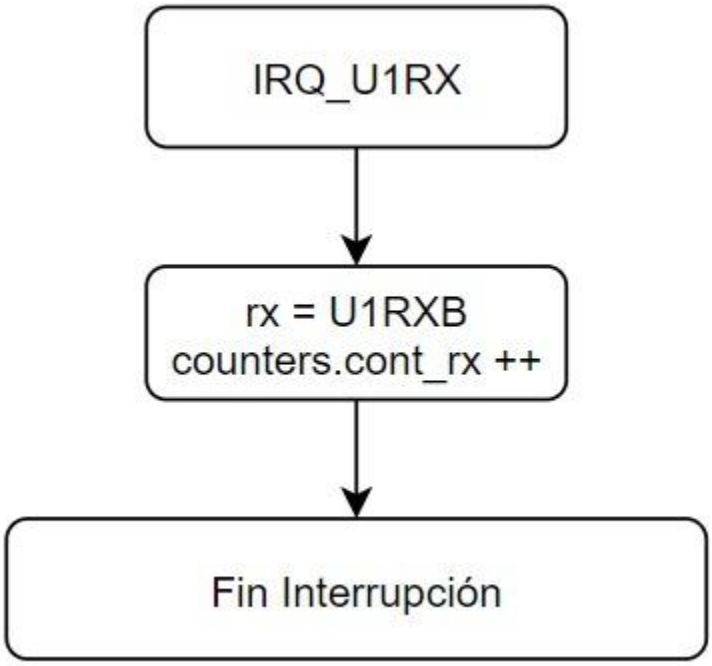


Imagen [22]. Rutina de interrupción para la recepción de UART.

Al observar la imagen anterior, vemos que el cambio principal efectuado durante la ejecución de la práctica está en que el proceso de cambio de parámetros no está dentro de la ISR de Recepción por UART como se planteó inicialmente, esto hace que la rutina sea lo más corta posible y dentro de ella se guarda en variable el dato recibido y el aumento de un contador que cuenta con la función de habilitador dentro del proceso de cambio de parámetro que se hace en el Polling debido a que se reciben dos bytes de datos.

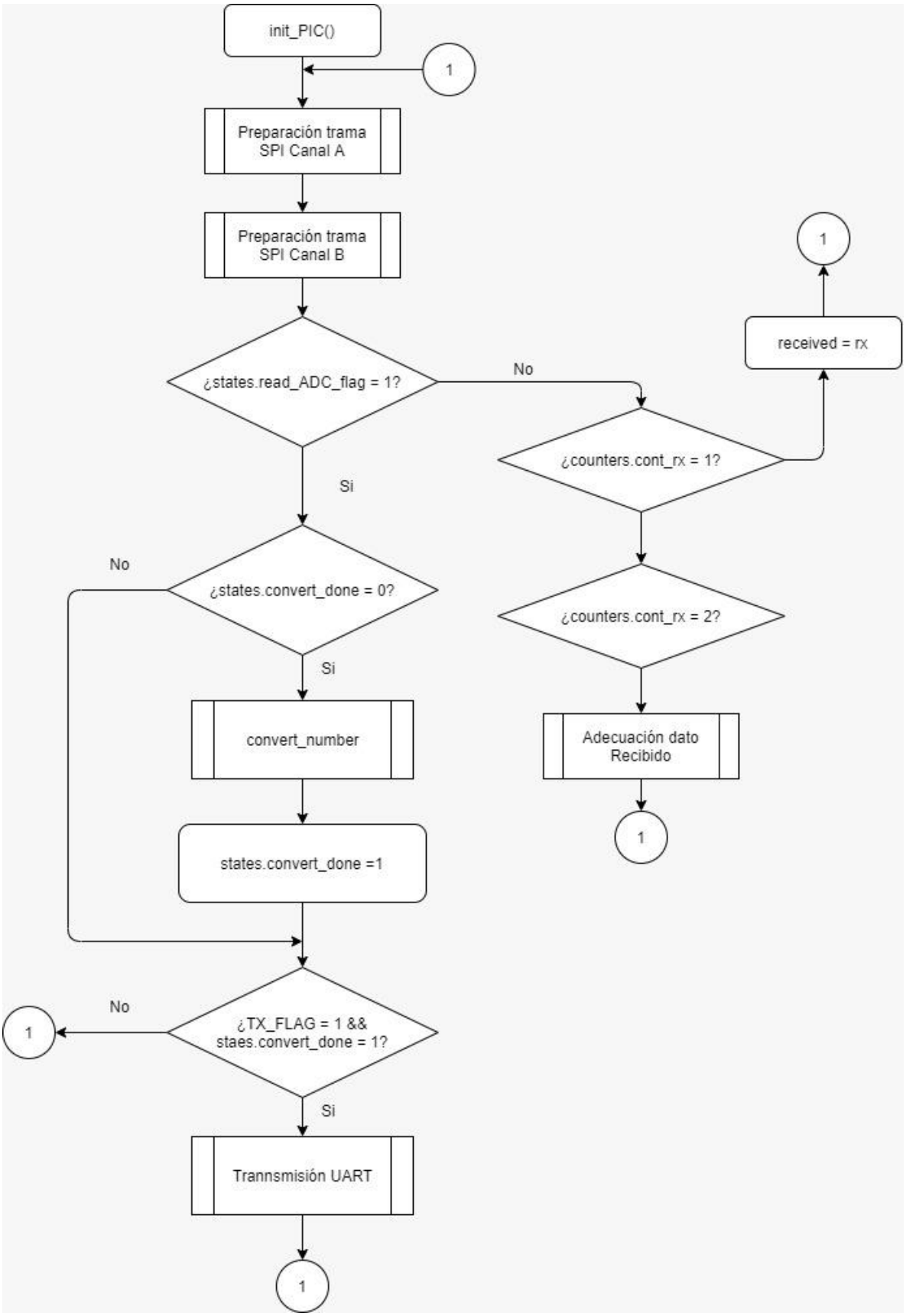

init_PIC()
1
Preparación trama SPI Canal A
Preparación trama SPI Canal B
¿states.read_ADC_flag = 1?
No
Si
¿states.convert_done = 0?
No
Si
convert_number
states.convert_done =1
¿TX_FLAG = 1 && staes.convert_done = 1?
No
1
Si
Trannsmisión UART
1
¿counters.cont_rx = 1?
¿counters.cont_rx = 2?
Adecuación dato Recibido
1
received = rx
1

Imagen [23]. Flowchart que describe el código del main.

En el Polling básicamente se realizan 4 procesos, los cuales son: Transmisión por SPI, la conversión de la salida del ADC a un conjunto de datos en formato ASCII, la transmisión por UART de los datos anteriormente convertidos y por último el proceso dependiente de la interrupción de recepción.

El primer proceso que encontramos es el encargado de añadir los bits de configuración necesarios para que la trama enviada haga que el DAC funcione; además, se hace la separación de los bits para poder enviar dos bytes por cada trama.

En el proceso de transmisión por UART, la adecuación que se realiza es que el resultado de cada canal se tiene guardado en dos posiciones diferentes por lo que es necesario hacer la iteración de estas para el envío y de igual manera la separación de dígitos y la separación de número para poder distinguir en el proceso de reconstrucción de las señales con el archivo .csv

Los cambios presentes respecto al original se basan en el flujo de trabajo y los procesos internos por cada condición cumplida.

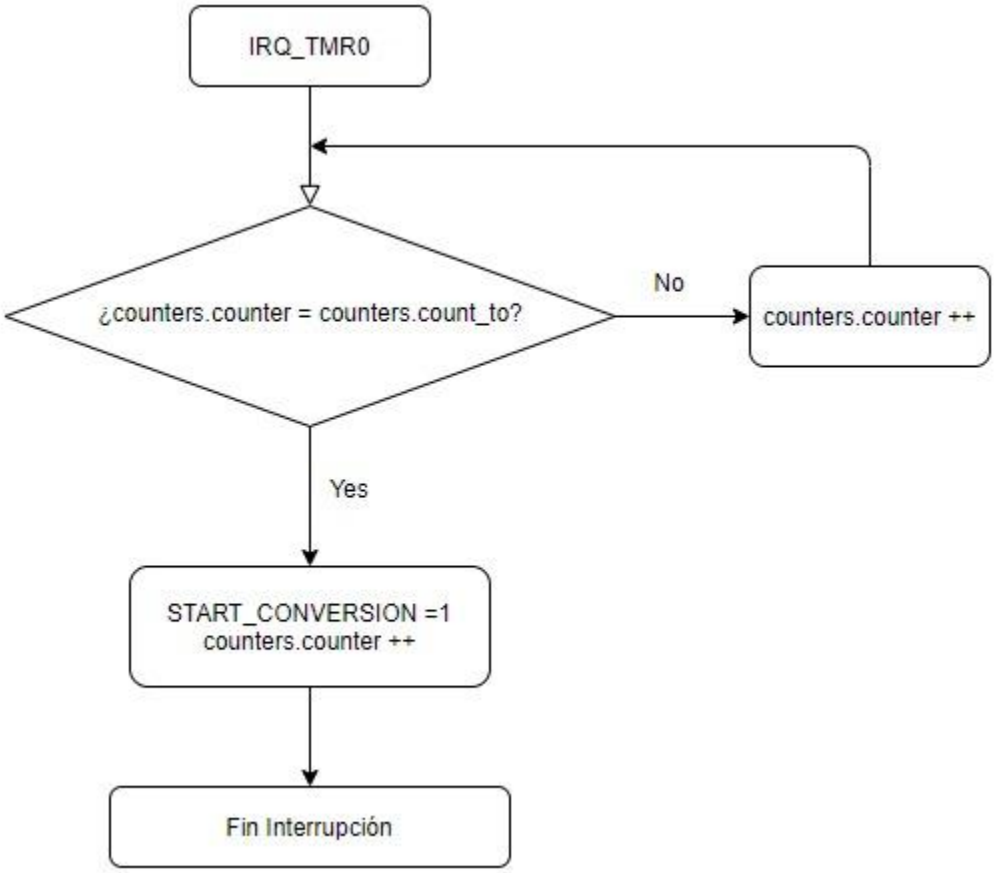


Imagen [24]. Rutina de interrupción para el timer 0.

En la ISR del timer esencialmente se controla el tiempo en el cual se habilita la conversión del ADC, se tiene una base de 4 us y a partir de ahí se realiza un contador por software con el fin inicialmente propuesto. Respecto al diseño original se tiene que el proceso de habilitación de ADC se realiza en este proceso.

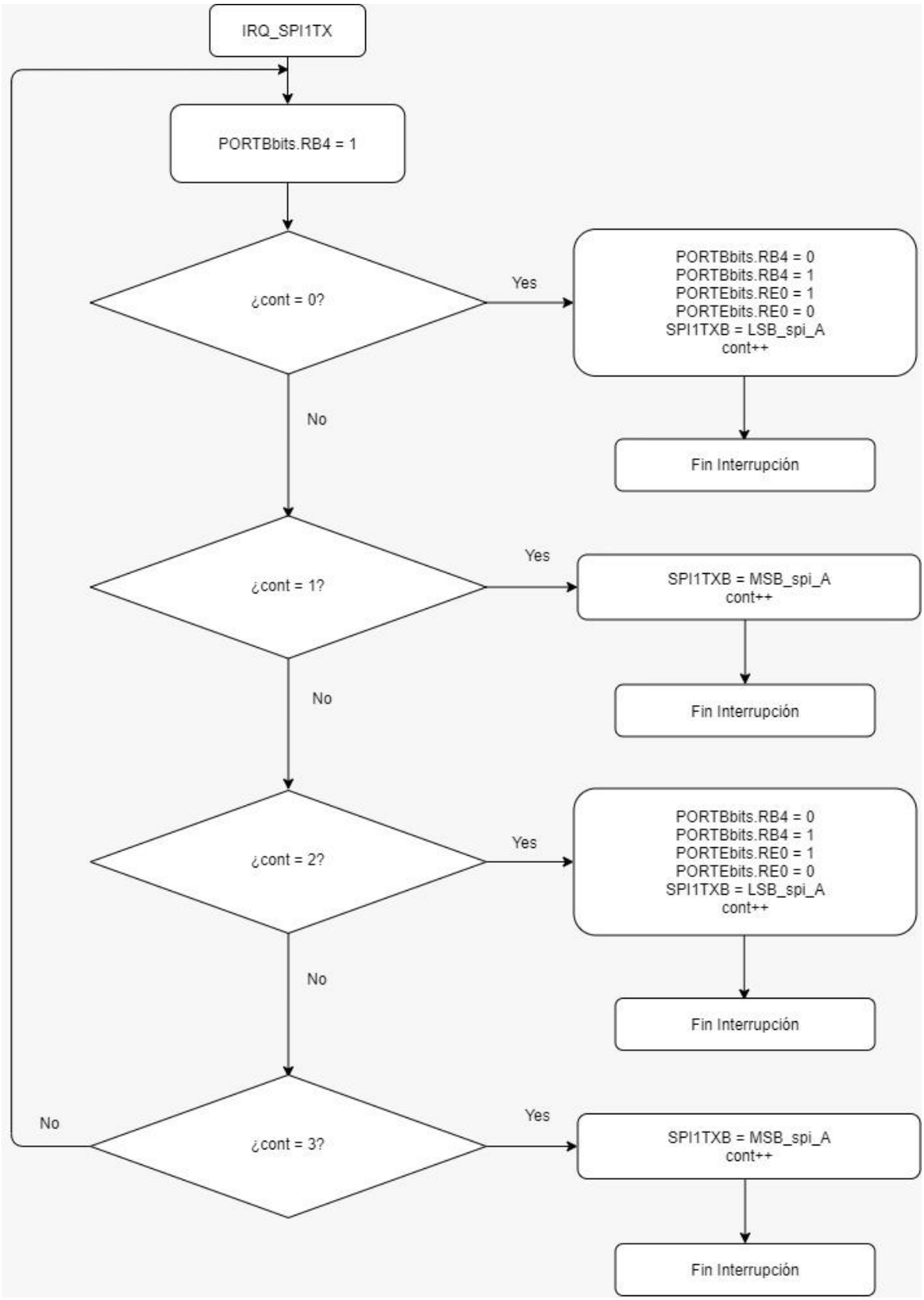


Imagen [25]. Rutina de interrupción para la transmisión por SPI.

Dentro de las rutinas que encontramos esta es la de mayor extensión debido a que de esta forma aseguramos que las tramas que queremos sean enviadas, lleguen de manera correcta y no se dupliquen o se presenten posibles errores que afecten la información tanto de configuración como de información de datos. En este caso de tener que mostrar a doble canal la salida del DAC, debemos tener en cuenta que son dos las tramas a enviar y como el registro en el cual se cargan las mismas es de 8 bits es el motivo por el cual en esta rutina se realiza 4 procesos condicionados.

También es posible ver que se activan algunas señales digitales que cumplen la función de activación de las entradas y salidas del periférico.

Respecto al diseño original, en este flujo se presentan variaciones sensibles todo esto con el fin de asegurar una transmisión efectiva, cosa que con el diseño original no se pudo lograr.

Para la parte B:

Los cambios existentes en esta sección son similares a los descritos anteriormente, sin embargo existen algunos procesos que no encontramos antes.

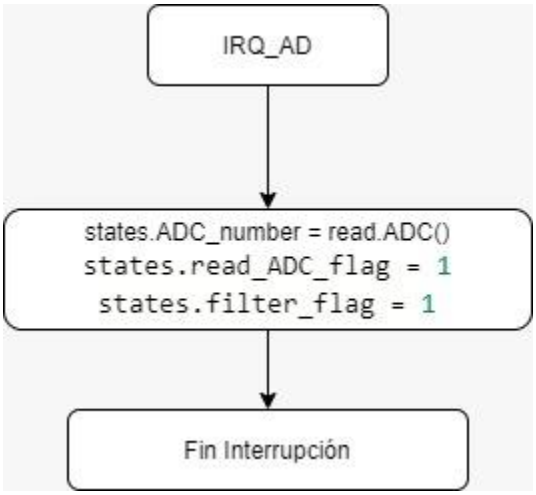


Imagen [26]. Rutina de interrupción para la conversión del ADC.

En esta simple ISR, al no tener que realizar la acción de multiplexación de canal se realiza un proceso breve que consta de asignar la salida del ADC a una variable para ser usada posteriormente. De igual manera habilitamos algunas banderas que sirven de condicionales dentro de procesos que hay en el polling.

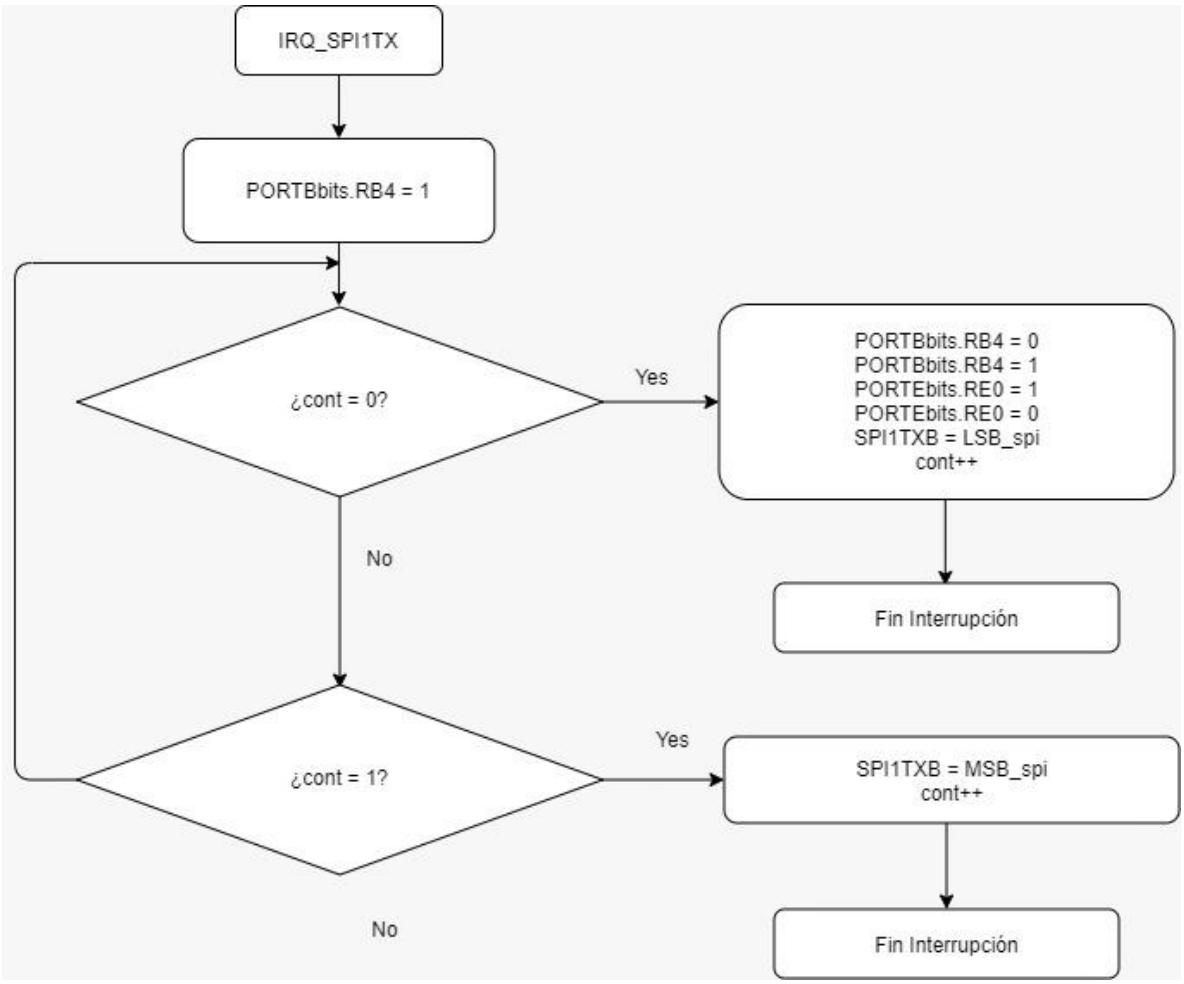


Imagen [27]. Rutina de interrupción para la transmisión por SPI.

El proceso que se realiza dentro de esta ISR, tenemos el un comportamiento similar al descrito en la ISR de SPI de la parte 2A con la diferencia de que en este caso es necesario enviar una única trama al periférico DAC; por este motivo, encontramos dos condicionales que nos permiten enviar dos bytes que contienen la configuración para el periférico y la información que se debe reconstruir, además de las señales digitales necesarias para la habilitación de las entradas y salidas del DAC

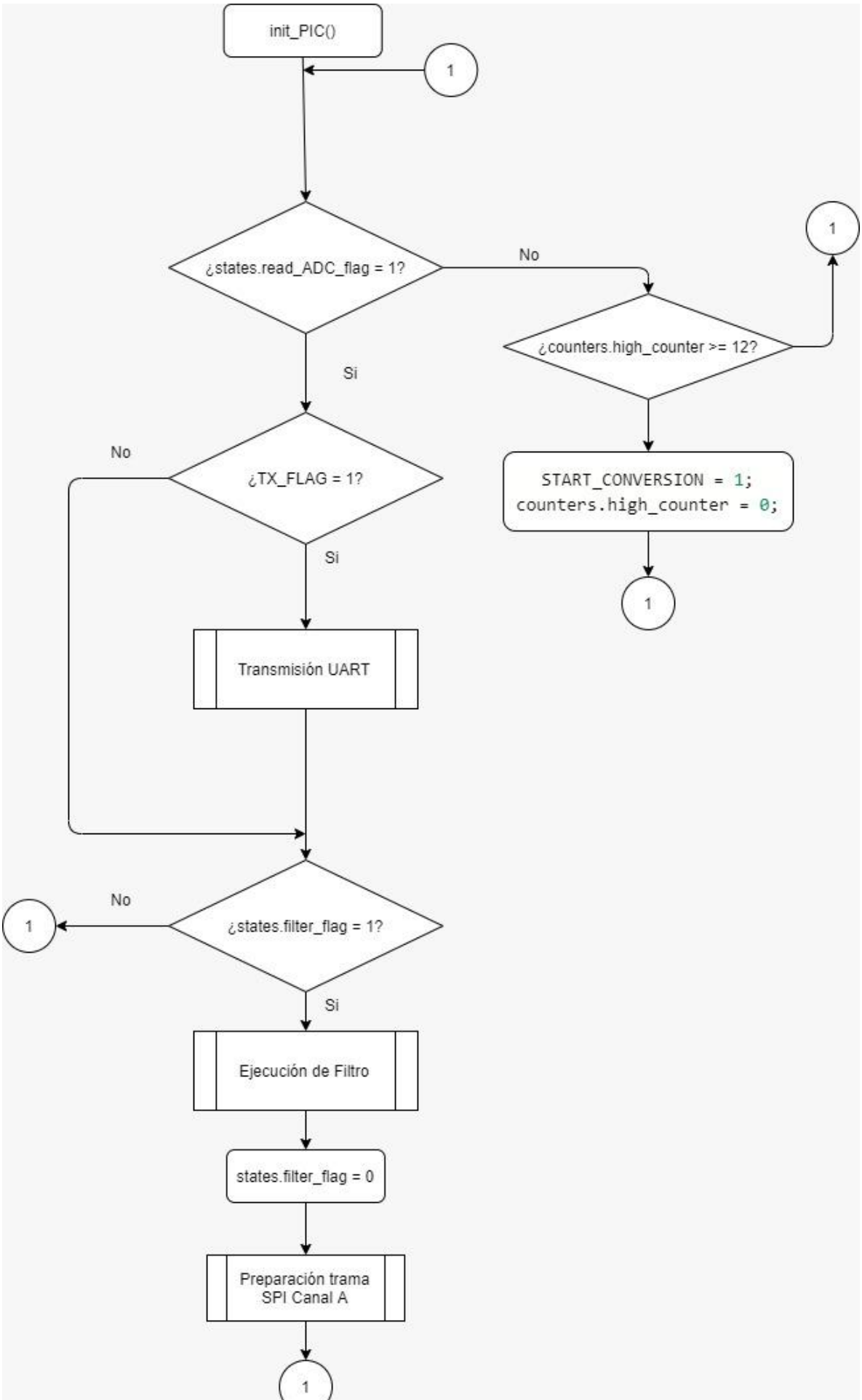

init_PIC()
1
¿states.read_ADC_flag = 1?
No
1
¿counters.high_counter >= 12?
Si
No
¿TX_FLAG = 1?
START_CONVERSION = 1;
counters.high_counter = 0;
Si
1
Transmisión UART
No
1
¿states.filter_flag = 1?
Si
Ejecución de Filtro
states.filter_flag = 0
Preparación trama
SPI Canal A
1

Imagen [28]. Flowchart que describe el código del main..

En este Polling, encontramos principalmente la ejecución condicionada de 3 procesos: Contador tiempo muestreo, Transmisión UART, Ejecución filtro y transmisión SPI.

El proceso de contador es el encargado de activar la conversión del ADC justo cuando una variable cumple una condición específica (Tiempo varía dependiendo del filtro que se quiera implementar) y de reiniciar la cuenta del mismo.

Para el proceso de ejecución del filtro encontramos el llamado de la función encargada de realizar el filtro el cual se activa en la ISR del ADC, de este modo aseguramos que únicamente se ejecute este proceso cuando exista una nueva muestra; seguido a esto se hace la adecuación de la trama que se debe enviar por medio de SPI.

Por último, en el proceso de transmisión por UART, estamos enviando la salida del ADC y la salida del filtro con el fin de ser almacenadas y procesadas por un PC; dicha transmisión es en forma decimal, es decir, que a diferencia de la transmisión realizada en la parte 2A no se hace ningún tipo de conversión o modificación a los datos que se van a enviar.

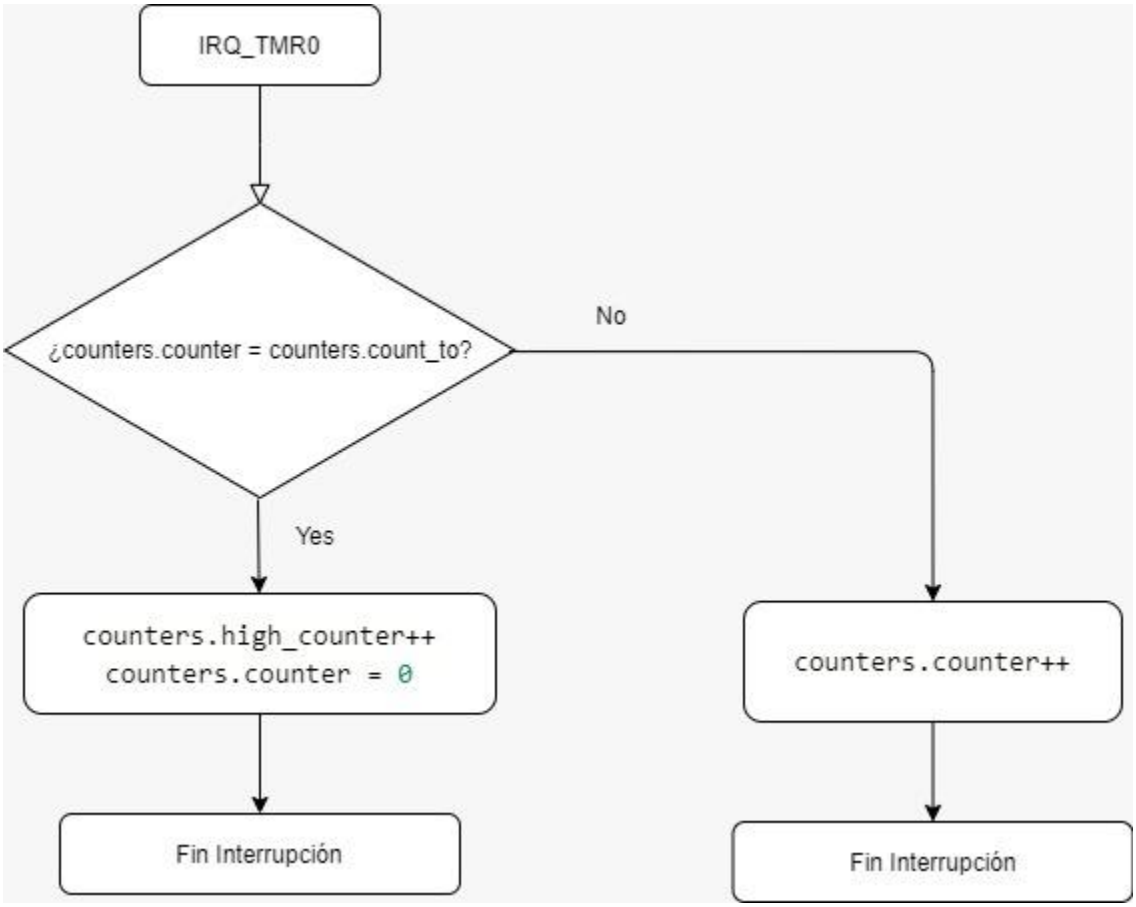


Imagen [29]. Rutina de interrupción del timer 0.

En la ISR del timer se lleva a cabo el proceso de hacer el contador que se usa para tener el tiempo de muestreo necesario dependiendo el filtro que se quiera ejecutar. Partiendo de la base de que el esta interrupción se genera cada 4 us

Conclusiones:

1. Se pudo observar que efectivamente hay diferencias sensibles entre lo que en un principio se planteó y lo que se ejecutó para cumplir los objetivos, desde el flujo del programa hasta la cantidad de procesos que se desarrollaron, sin dejar atrás las implementaciones de cada una de las ISR. Consideramos que estas diferencias se deben a la falta de experiencia en un primer momento con los periféricos que se utilizaron, además de no tener un plano dimensionado de cada tema específico que cada uno de estos trae y sus diferentes modos de operación.
2. En este laboratorio se desarrollaron habilidades para implementar rutinas de conversión para el ADC del procesador, rutinas para mostrar datos digitales en un conversor DAC usando SPI y también manejar la interacción entre las rutinas del ADC, DAC y timers, como se explica a lo largo de este documento se cometieron una serie de errores que ayudarán a mejorar el desarrollo de todo esto en futuros trabajos.
3. Se aprendieron técnicas para implementar varios filtros FIR de diferentes órdenes de magnitud e IIR de diferente tipos de filtros, del mismo modo se aprendió la manera de realizar la verificación del funcionamiento de los mismos y de compararlos con los diseños originales, lo cual se convierte en una herramienta muy importante para el desarrollo del proyecto final de la asignatura