Universidad Internacional de La Rioja (UNIR)

ESIT

Máster Universitario en Inteligencia Artificial

Procesamiento de señales electromiográficas utilizando algoritmos de inteligencia artificial 'on-the-edge'

Trabajo Fin de Máster

Presentado por: Proaño Guevara, Daniel David

Director/a: Blanco Valencia, Xiomara Patricia

Ciudad: Cuenca, Ecuador
Fecha: 19 de julio de 2020

Resumen

En los últimos años, la inteligencia artificial se ha posicionado en todos los campos tecnológicos, principalmente ejecutando cálculos en sistemas distribuidos o en la nube, pero en aplicaciones como el procesamiento de señales electromiográficas, se ha observado la necesidad de procesar en el dispositivo de adquisición, utilizando una metodología 'on-the-edge'. Este documento compara un sistema DSP tradicional, con un sistema informático distribuido y uno 'en el borde', para este propósito se utiliza un sistema embebido PYNQ-Z1, que evalúa las tres metodologías de procesamiento en escenarios similares. Los resultados muestran que el sistema de procesamiento 'en el borde' es más efectivo que los demás, ya que tiene una latencia aceptable, mayor seguridad en el manejo de la información y, dado el estado de la tecnología actual, su uso es justificable y recomendado.

Palabras Clave: DSP, Edge Computing, EMG, Filtro RLS, Inteligencia Artificial, PYNQ-Z1

Abstract

In recent years, artificial intelligence has positioned itself in all technological fields, mainly executing calculations in distributed systems or in the cloud, but in applications such as electromyographic signal processing, the need to process in the acquisition device has been observed. , using an 'on-the-edge' methodology. This document compares a traditional DSP system, with a distributed and an 'on-the-edge' computer system, for this purpose a PYNQ-Z1 embedded system is used, which evaluates the three processing methodologies in similar scenarios. The results show that the 'on-the-edge' processing system is more effective than the others since it has an acceptable latency, greater security in the handling of information and, given the current state of technology, its use is justifiable and recommended instead of distributed computing systems

Keywords: Artificial Intelligence, DSP, Edge Computing, EMG, PYNQ-Z1, RLS Filtering

Índice de contenidos

Índice de tablas

Índice de figuras

1. Introducción

Una señal biomédica es un fenómeno que transporta información relativa a uno o más sistemas biológicos involucrados. Debido a la complejidad de los sistemas biológicos y sus interrelaciones, el procesamiento avanzado de señales solo ve una sombra de la realidad y deben ser procesadas de manera no trivial para ser realmente informativas. Es posible decir que el impacto de la tecnología en el procesamiento de señales en medicina está mejorando debido al creciente número de nuevos procedimientos y su confiabilidad, lo que permite un mejor análisis de situaciones conocidas (Cerutti & Marchesi, 2011).

Tradicionalmente el procesamiento de señales biomédicas se ha limitado a filtrar el ruido de interferencia de las líneas de potencia y el análisis espectral para entender las características frecuenciales de estas señales (Rangayyan, 2015), y recientemente se ha introducido el procesamiento y filtrado adaptativo de estas señales utilizando algoritmos inteligentes. Para esto se emplea un paradigma de procesamiento sensor-computador-actuador, que centraliza todo el procesamiento relacionado a la señal en un computador con el cual se comunica el sistema de adquisición de señales. Este proceso produce latencias y potenciales fallas de seguridad sobre la información adquirida. Este trabajo presenta una comparativa entre las técnicas de procesamiento tradicional, inteligente e introduce el procesamiento inteligente en el borde (AI on-the-edge) buscando evaluar las características de filtrado, latencias y finalmente establecer qué sistema es el más adecuado para el procesamiento de señales electromiográficas.

1.1 Motivación

En años recientes, la inteligencia artificial (AI) se encuentra en todos los campos de la tecnología, pero dado que muchos procesos de aprendizaje automático y procesamiento requieren una inmensa capacidad de cómputo, se ha realizado cálculos en servidores en la nube. Recientemente se ha hecho evidente la necesidad de nuevas soluciones alternativas para solucionar los problemas de cómputo, como las unidades de procesamiento tensorial de Google (TPU) o las arquitecturas dedicadas para AI que NVidia está implementando en sus GPU. El ecosistema basado en la nube ha demostrado ser una plataforma práctica para solucionar varias aplicaciones de AI, pero debemos tomar en cuenta que hay sectores con necesidades de procesamiento donde la latencia de datos es crítica, el dispositivo puede encontrarse en zonas sin cobertura de internet, o trabajar con datos críticos e información

sensible sobre el usuario. Estos problemas llevan a la necesidad de que el cómputo de AI se realice en el borde (AI on-the-edge) (Lee, Tsung, & Wu, 2018; Zhou et al., 2019).

El procesamiento de señales electromiográficas es un campo importante dentro del desarrollo de dispositivos protésicos, ya que estas señales son utilizadas para comandar todo el sistema robótico, por lo cual es necesario que las señales de entrada al sistema controlador sean robustas a la interferencia y contengan sólo la información necesaria para la ejecución de acciones (Cerutti & Marchesi, 2011).

1.2 Planteamiento del trabajo

Plantear un sistema de procesamiento de señales inteligente en el borde disminuye la inseguridad debida a la transmisión de información a un HUB de cómputo, ya que la señal puede ser interferida exponiendo información sensible del usuario, elimina la posible corrupción de los datos debido al proceso de comunicación, reduce el tiempo de latencia asociada al cómputo de procesamiento y permite la escalabilidad a futuros procesos dentro del mismo dispositivo de hardware.

Este trabajo presenta una comparativa entre las técnicas de procesamiento tradicional, inteligente e introduce el procesamiento inteligente en el borde (AI on-the-edge) buscando evaluar las características de filtrado, latencias y finalmente establecer qué sistema es el más adecuado para el procesamiento de señales electromiográficas.

Las preguntas de investigación consideradas para el siguiente trabajo son:

- ¿Es más efectivo un procesamiento inteligente ‘on-the-edge’ de las señales electromiográficas que el procesamiento tradicional en hardware?
- ¿Se justifica la aplicación de técnicas de inteligencia artificial ‘on-the-edge’ para el procesamiento de señales biométricas?

1.3 Estructura de la memoria

En el capítulo 1. Introducción se presenta la motivación y planteamiento del presente trabajo

En el capítulo 2. Contexto y estado del arte se realiza una revisión teórica de los sistemas de procesamiento disponibles y las tecnologías que se están utilizando actualmente para el procesamiento inteligente de señales biométricas

En el capítulo 3. Objetivos y metodología de trabajo, se presenta el objetivo genera, objetivos específicos, la metodología del trabajo y las delimitaciones con las que se realizó el presente trabajo de fin de máster

En el capítulo 4. Descripción detallada del experimento, se presenta el diseño en hardware del sistema de procesamiento y se desarrolla en varias secciones la implementación y resultados del sistema de DSP tradicional, el algoritmo implementado en el dispositivo en el borde, y la implementación en un sistema distribuido de computación, cubriendo de esta manera las tres aproximaciones más comunes: DSP en el periférico, inteligencia artificial 'on-the-edge' y cómputo distribuido, como una aproximación al cloud computing.

En el capítulo 5. Descripción de los resultados, se presentan los gráficos resultantes del escenario de pruebas propuesto para la comparación de resultados y se finaliza con un análisis del perfil de Harris en el que se evalúa por fortalezas o debilidades cada una de las propuestas

En el capítulo 6. Discusión, se comenta sobre los resultados obtenidos en el capítulo 5, y se hacen acotaciones a las anomalías

El capítulo 7. Conclusiones y trabajo futuro, muestra el aporte realizado por el presente trabajo

El capítulo 8. Bibliografía, contiene las citas y referencias que darán sustento teórico al presente trabajo

En Anexos, se presentan documentos, códigos y simulaciones que complementan la información expuesta en el presente trabajo y finaliza con un artículo de investigación

2. Contexto y estado del arte

La electromiografía es una técnica experimental dedicada al desarrollo, registro y análisis de las señales mioeléctricas. Las señales mioeléctricas, se construyen por variaciones fisiológicas en el estado de las membranas de las fibras musculares (Konrad, 2005). Esta técnica prueba la integridad del sistema motor entero, que consiste en las motoneuronas superiores e inferiores, unión neuromuscular y el músculo (Kimura, 2013).

Uno de los principales problemas con las señales SEMG es que son una de las señales electrofisiológicas más fáciles de medir, pero también es una de las más complejas de interpretar cuantitativamente por lo tanto (Stegeman & Hermens, 2007) expresa que “para su detrimento, la electromiografía es muy fácil de usar y consecuentemente muy fácil de abusar”, lo que implica que hay que tener especial cuidado en las técnicas de adquisición y procesamiento de dichas señales.

El procesamiento digital de señales (DSP por sus siglas en inglés) se diferencia de otras áreas de las ciencias de la computación por el único tipo de datos que utiliza: señales, que por lo general son adquiridas del mundo real, como ondas sonoras, sísmicas, biométricas, etc. El procesamiento digital de señales consiste en las matemáticas, algoritmos y las técnicas utilizadas para manipular, transformar y representar estas señales después que hayan sido digitalizadas (Oppenheim, Schafer, & Buck, 1998; Smith, 1997).

A diferencia de las técnicas de electrónica analógica de procesamiento de señales, con la utilización de técnicas de DSP se garantiza la reproductibilidad y precisión, siendo reconocida como superior y más fiable que su contraparte analógica (Woods, McAllister, Lightbody, & Yi, 2008). El uso de DSP en aplicaciones clínicas proporciona información sobre el rendimiento de los sistemas que constituyen organismos vivos. La eficacia y, por lo tanto, la utilidad del análisis de datos en su conjunto depende de la calidad de la tecnología y su desarrollo. Es posible decir que el impacto de la tecnología en el procesamiento de señales en medicina está mejorando debido al creciente número de nuevos procedimientos y su confiabilidad, lo que permite un mejor análisis de situaciones conocidas (Cerutti & Marchesi, 2011).

El desarrollo del procesamiento digital de señales ha estado ampliamente ligado al desarrollo de los dispositivos de silicio. El desarrollo de chips dedicados para DSP y System-on-chip (SoC) es extremadamente costoso y dado que su arquitectura interna es fija, lo que impide la escalabilidad de estos dispositivos y su adaptabilidad a las nuevas técnicas de procesamiento. De esta forma los FPGA han surgido como una propuesta adecuada para tareas de DSP, ya que, por su flexibilidad, paralelismo y alta velocidad permiten realizar las tareas de DSP con

la precisión y velocidad que los procesos biomédicos requieren (Hsueh, Yin, & Chen, 2015; Toledo-Pérez, Martínez-Prado, Rodríguez-Reséndiz, Tovar Arriaga, & Márquez-Gutiérrez, 2017; Woods et al., 2008).

La evolución de los diferentes enfoques empleados en el procesamiento de señales biomédicas, dificultan la adecuada elección de un método que se utilizará en tareas de filtrado de señales, extracción de información de validez clínica, clasificación diagnóstica, monitoreo de pacientes, etc. (Cerutti & Marchesi, 2011).

Tomando como cierto que las señales biológicas llevan información relativa al sistema o sistemas que lo generan, se establece que el procesamiento de la señal debe seguir los siguientes objetivos:

1) Resaltar la información útil de la señal original

2) Interpretar los resultados y validar los parámetros obtenidos de la subsecuente etapa de decisión

3) Producir innovación en la mejora del conocimiento fisiológico, la producción de nuevos equipos y dispositivos “inteligentes” y la definición de nuevos protocolos para la prevención, diagnóstico y terapia.

La señal EMG sin procesamiento ofrece muy poca información, por lo que para el análisis inicial se pueden considerar análisis frecuencial, de amplitud, o ambos. Las mediciones de amplitud están relacionadas a la fuerza, torque y activación muscular, mientras que el análisis frecuencial ofrece información sobre la fatiga muscular, algunas de las técnicas más comunes de procesamiento tradicional son la rectificación de la onda, sea calculando el valor absoluto o cuadrática, elevando los valores de las muestras al cuadrado y se realiza un promediado temporal, móvil o ponderado. Esto ofrece valores, en el caso lineal del valor rectificado promedio (ARV) y en el caso cuadrático, como la media cuadrática (RMS). En el análisis frecuencial, se obtienen parámetros del cálculo del espectro de potencia en épocas (epochs) temporales con una duración promedio de 0.25 a 1 segundos. De aquí se obtiene la frecuencia media (MNF) y mediana (MDF); también para el análisis y descomposición de las unidades de disparo de las motoneuronas se analiza la razón de cruce por cero de la señal (ZCR). Los filtros frecuenciales que se implementan de tipo FIR e IIR, comparten la eliminación de banda por debajo de los 20Hz hasta los 1000Hz (Hägg, Melin, & Kadefors, 2005; Hsueh et al., 2015; Merletti & Farina, 2016; Rangayyan, 2015; Roland, Amsuess, Russold, & Baumgartner, 2019; Rozaqi, Nugroho, Sanjaya, & Ivonita Simbolon, 2019; Sahu & Sahu, 2018; Tankisi et al., 2020).

(Li, Li, Jiang, Chen, & Liu, 2018; Roland et al., 2019) proponen la utilización de filtros tipo peine (comb filter) para eliminar la interferencia que genera la línea de potencia con un proceso aditivo-sustractivo de la señal a eliminar.

En el caso de la detección de patrones y eventos tempo-frecuenciales, es comúnmente utilizado el procesamiento multiresolución a través de wavelets o la transformada de Fourier en intervalos temporales, así mismo, se utiliza la descomposición por wavelets para realizar la reducción de ruido por umbrales wavelet (Cerutti & Marchesi, 2011; Li et al., 2018; Onay & Mert, 2020; Prasad, Naganjaneyulu, & Prasad, 2018; Restrepo-Agudelo et al., 2017).

(Roland et al., 2019) establece que para la implementación de los procesos de DSP es necesario que se cumplan con las siguientes características:

- Señal de alta calidad, conseguir una relación señal-ruido (SNR) elevada
- Estabilidad, que la señal resultante no se vea contaminada por artefactos de movimiento u otras fuentes de ruido
- Cálculo eficiente, el procesamiento digital de señales debe estar implementado para operaciones de tiempo real de forma que minimice los recursos de cálculo
- Corta respuesta temporal, los retardos de procesamiento no sean mayores a 100ms
- Requerimientos optimizados de memoria, de forma que permita la implementación de otros algoritmos que se necesiten en etapas subsecuentes de procesamiento o decisión

El procesamiento inteligente de señales (ISP por sus siglas en inglés) utiliza aprendizaje y otras técnicas ‘inteligentes’ para extraer tanta información como sea posible de los datos de la señal que se está recibiendo. El procesamiento de señales clásico ha trabajado ampliamente con modelos matemáticos que son lineales, estacionarios, gaussianos y por esto son ampliamente utilizados, pero los sistemas reales son no lineales, con estructuras estadísticas erráticas o impulsivas que pueden variar en el tiempo. Cambios mínimos en la señal o en la estructura del ruido pueden llevar a cambios cualitativos en cómo los sistemas de procesamiento clásico filtran el ruido o mantienen la estabilidad (Kosko & Haykin, 2001).

En el campo del control de prótesis de miembro superior se utilizan ampliamente las señales EMG de superficie, de esta se extraen los coeficientes de un modelo temporal de promediado móvil autorregresivo (ARMA) que serán los parámetros que se ingresan al controlador de movimientos de la prótesis (Parker, Englehart, & Hudgins, 2005).

Los filtros adaptativos utilizan técnicas que se adecúan a las condiciones de la señal de entrada manteniendo estable la respuesta del sistema, así se han constituido en uno de los métodos más eficientes para la adquisición de señales fisiológicas (Salamea Palacios & Luna Romero, 2011).

Una aproximación para la eliminación de ruido es la cancelación adaptativa de ruido de las señales EMG utilizando una fuente de ruido externa que esté vagamente relacionada con el ruido implícito en la señal EMG. Para esta tarea se han implementado algoritmos de filtrado como el Filtro de Kalman, LMS, RLS, Wiener, UFIR, Gaussiano, algoritmos de colonia de abejas, Bayesianos, entre otros (Ghalyan, Abouelenin, & Kapila, 2019; Jamal, Lee, & Hyun, 2019; Márquez-Figueroa, Shmaliy, & Ibarra-Manzano, 2020; Menegaldo, 2017; Okoniewski, Kocon, & Piskorowski, 2018; Verma, Singh, & Gupta, 2018; Yu, Akhmadeev, Le Carpentier, Aoustin, & Farina, 2019).

Se han encontrado escasos trabajos relacionados al procesamiento inteligente de señales EMG en el borde, (Kasaeyan Naeini et al., 2019; Tam, Boukadoum, Campeau-Lecours, & Gosselin, 2020) realizan el proceso de adquisición y filtrado digital tradicional utilizando como dispositivo común una GPU de propósito general ‘NVidia Jetson’, sobre la cual implementan algoritmos de aprendizaje automático y toma de decisiones, pero no se evidencia un procesamiento inteligente de la señal per se.

En el presente trabajo se utilizará la arquitectura propuesta por (Roland et al., 2019) para el procesamiento digital tradicional de las señales, el cual está compuesto por filtros paso bajo y paso alto, una etapa de rectificación y suavizado. Para el procesamiento inteligente se utilizará un filtro RLS con una fuente de ruido blanco que será implementado en un computador y en un dispositivo en el borde

3. Objetivos y metodología de trabajo

3.1. Objetivo general

Desarrollar sistemas de procesamiento de señales electromiográficas que permitan comparar la aplicación de algoritmos inteligentes 'on-the-edge', con un sistema distribuido de cómputo, y técnicas de DSP tradicionales para identificar las ventajas y desventajas de cada sistema y planificar a futuro la incorporación de sistemas inteligentes sobre hardware para la adquisición de señales EMG enfocadas al control de dispositivos protésicos de miembro superior.

3.2. Objetivos específicos

- Conocer las técnicas de procesamiento digital de señales para señales electromiográficas.
- Diseñar un escenario de pruebas para la comparativa de tres sistemas de procesamiento de señales que sigan paradigmas diferentes de implementación.
- Implementar algoritmos de procesamiento de señales utilizando técnicas clásicas, técnicas inteligentes utilizando un sistema discreto de cálculo y técnicas inteligentes sobre un dispositivo en el borde
- Evaluar los sistemas de procesamiento de señales

3.3. Metodología del trabajo

Para la implementación de los sistemas de procesamiento se utilizará el sistema embebido PYNQ-Z1, que utiliza un SoC ZYNQ-7000 de XILINX compuesto por un ARM Cortex-A9 y celdas de lógica programable de la familia Artix-7

De cara a la evaluación de tres sistemas de procesamiento se establece un escenario de pruebas con la estructura presentada en la **¡Error! No se encuentra el origen de la r eferencia.**, el cual está dividido en los siguientes pasos:

1. Diseñar una arquitectura de filtrado *tradicional* compuesta por:
    a. Filtro paso alto con frecuencia de corte en 20Hz
    b. Filtro paso bajo con frecuencia de corte en 500Hz

   c. Rectificación obteniendo el valor absoluto de la muestra
   d. Suavizado utilizando un filtro de promediado móvil (MA)

2. Diseñar una arquitectura de filtrado inteligente que utilice filtrado RLS
3. Implementar el filtro tradicional en el sistema embebido
4. Utilizar el sistema embebido para captura, envío de datos al PC, y reconstrucción de la señal analógica, procesando la señal en la arquitectura de filtrado inteligente
5. Implementar la arquitectura inteligente en el sistema embebido

Figura 1. *Escenario de pruebas para la comparativa de los sistemas de procesamiento* (Fuente Propia)

Las métricas de evaluación para los escenarios de prueba se basarán en la comparación de medias y raíz media cuadrática de la señal de entrada y salida, el desfase temporal asociado al tiempo de procesamiento, y un análisis del espectrograma de las señales.

Finalmente se comparan las métricas de evaluación buscando de esta manera determinar la arquitectura más adecuada para las tareas de filtrado de señales EMG.

3.4. Delimitaciones

Las señales EMG serán recreadas utilizando un generador de funciones con información obtenida de bases de datos públicas obtenidas desde (Fang, Zhou, Li, & Liu, 2018).

Los análisis frecuenciales y mediciones se realizarán en un osciloscopio Analog Discovery 2 de Digilent.

4. Descripción detallada del experimento

El dispositivo en el borde utilizado es un PYNQ-Z1 (presentado en la Figura 2), que utiliza un SoC ZYNQ-7000 de XILINX compuesto por un ARM Cortex-A9 y celdas de lógica programable de la familia Artix-7, cargada con un sistema operativo basado en Linux que ejecuta como medio de interfaz notebooks de Python

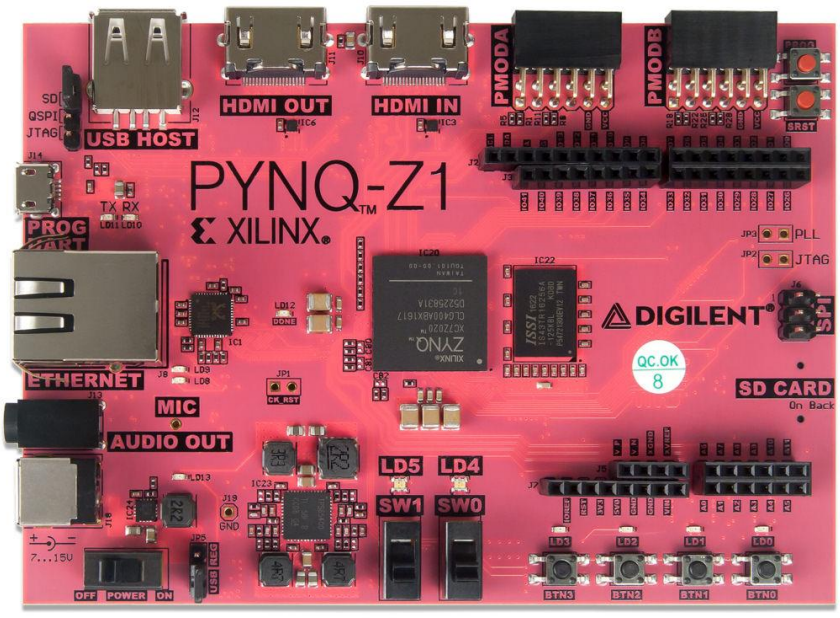


Figura 2. *Sistema Embebido PYNQ Z-1* (Xilinx Inc., 2017)

Como sistema de lectura de señales, se utiliza un módulo PMOD AD1 como se puede ver en la Figura 3, que incorpora un convertidor de analógico a digital (ADC) AD7476, este convertidor tiene una tasa de salida máxima de 1MSPS, a una resolución de 12 bits.

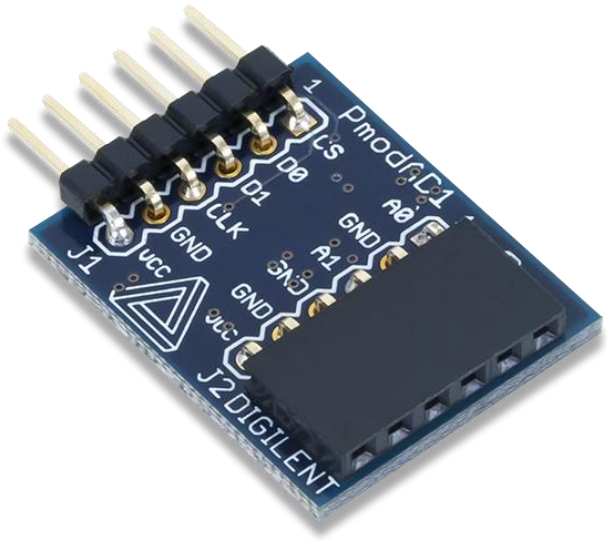


Figura 3. *Módulo convertidor de analógico a digital* (DIGILENT, 2016a)

Ya que el sistema de comunicación se basa en un diagrama de tiempos como se puede ver en la Figura 4, es necesario decodificar este protocolo e implementarlo en HDL (Lenguaje de descripción de hardware).

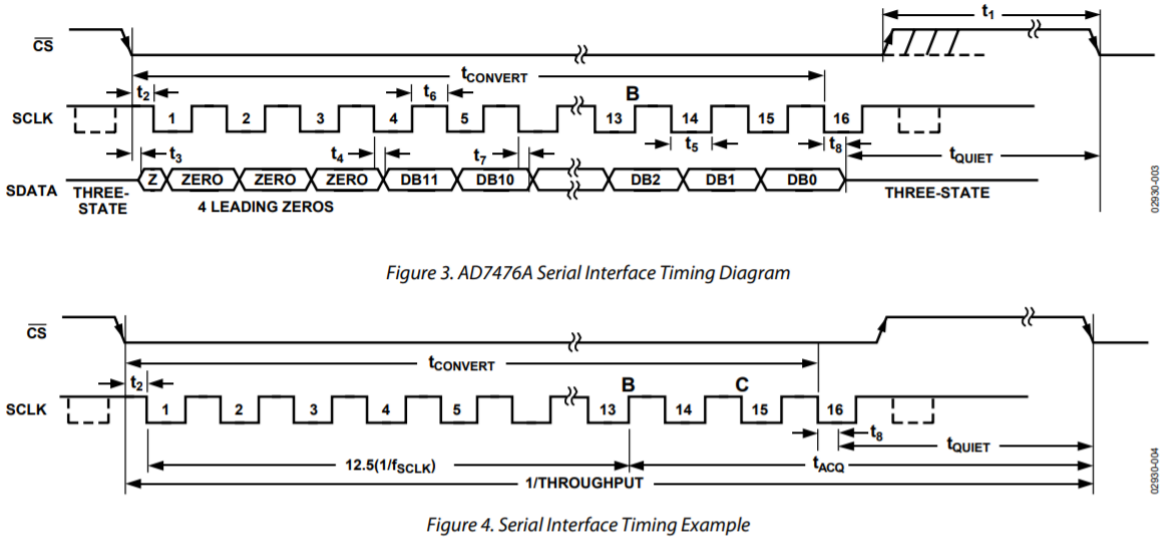


Figura 4. *Diagrama de tiempos y reloj para el protocolo de comunicación del ADC* (Analog Devices, 2011)

Para efectuar la lectura de las señales que ofrece este ADC, se diseña un diagrama de caja negra como se puede ver en la Figura 5, que describe las entradas y salidas que tendrá el

sistema de procesamiento, en este caso se cuenta con una señal de entrada de reloj proporcionada por el sistema, una señal de reset, señal de fin de lectura, vector de datos leídos, señal de activación de periférico (CS), señal de reloj de periférico (spi_clk) y la lectura de la señal (miso)

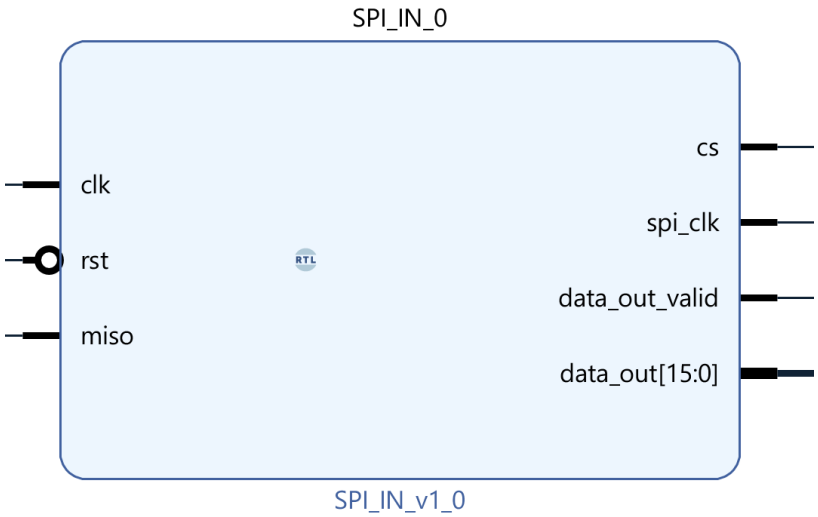


Figura 5. *Modelo de caja negra de protocolo de lectura por SPI modificado* (Fuente Propia)

Posteriormente se diseña una máquina de estados finitos como se puede ver en la Figura 6 que gobernará la ejecución del bloque de hardware

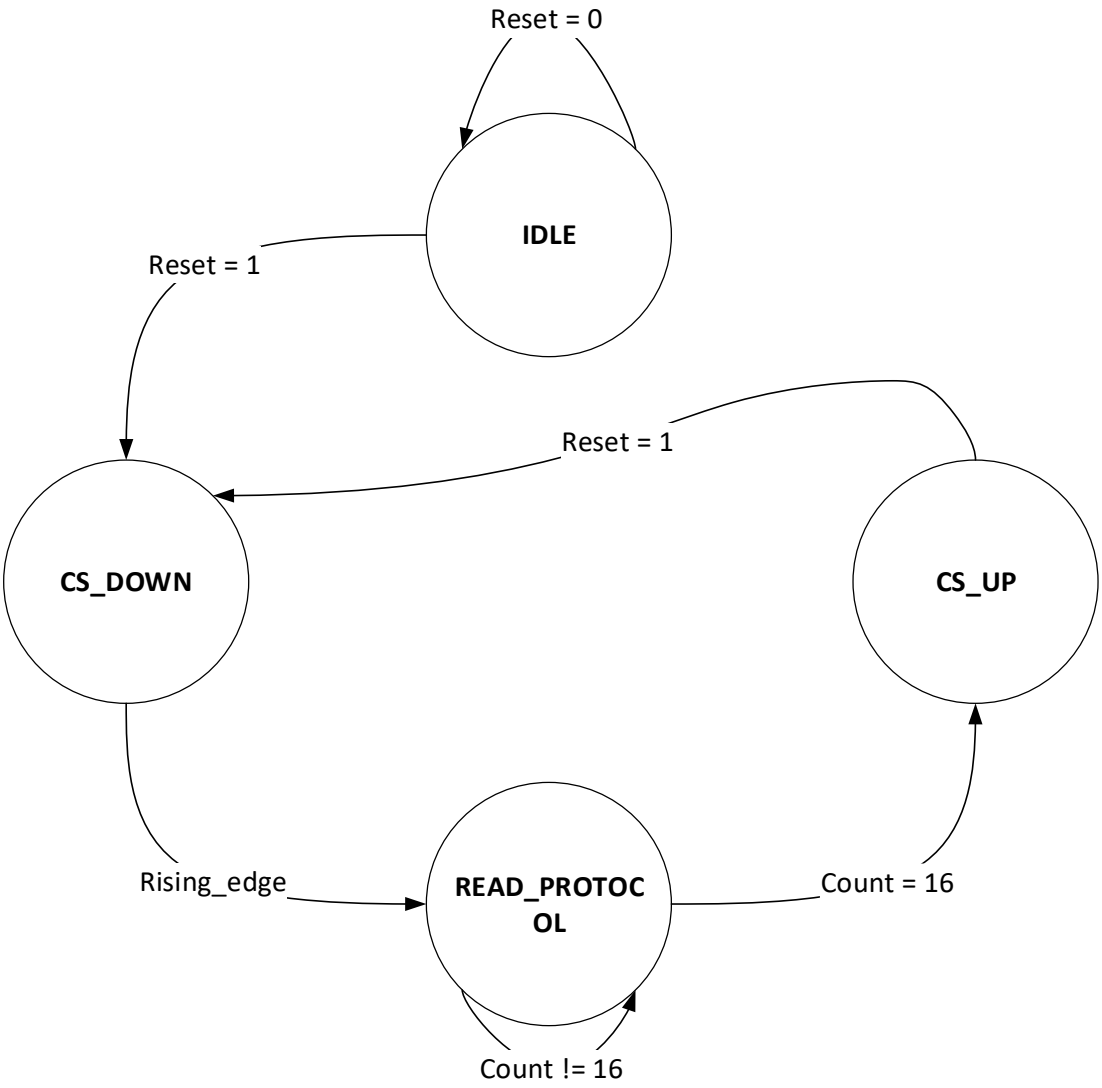


Figura 6. *Máquina de estados finitos del protocolo de adquisición* (Fuente Propia)

En el Anexo I. Código VHDL para lectura del protocolo de señales del convertidor analógico-digital, se adjunta el código de VHDL que describe esta máquina de estados

Como sistema de reconstrucción de señales, se utiliza un módulo PMOD DA2 como se puede ver en la Figura 7, que incorpora un convertidor digital a analógico (DAC) DAC121S101, este convertidor tiene una tasa de salida máxima de 1MSPS, a una resolución de 12 bits.

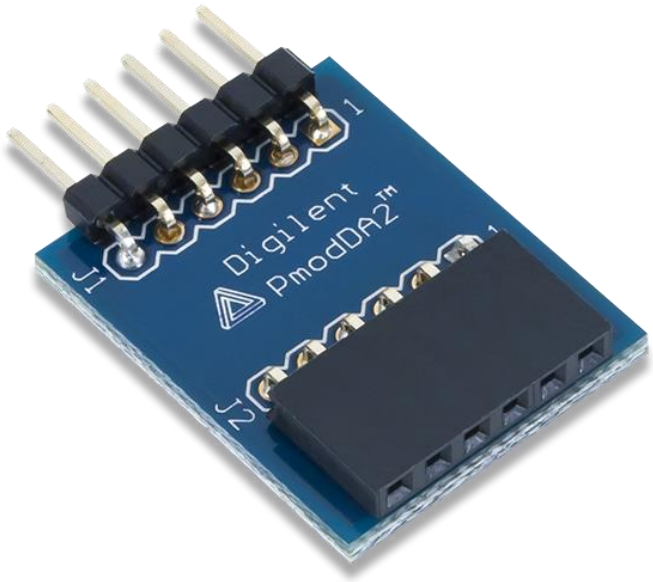


Figura 7. *Módulo convertidor de digital a analógico* (DIGILENT, 2016b)

Ya que el sistema de comunicación se basa en un diagrama de tiempos como se puede ver en la Figura 8, es necesario decodificar este protocolo e implementarlo en HDL (Lenguaje de descripción de hardware).

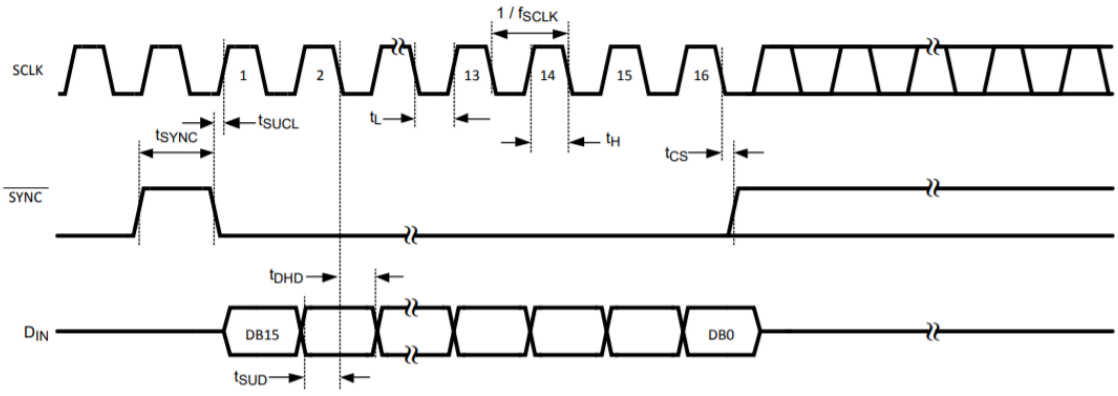


Figure 2. DAC121S101 Timing

Figura 8. *Diagrama de tiempos y reloj para el protocolo de comunicación del DAC* (Texas Instruments, 2005)

Para escribir en el registro de este DAC, se diseña un diagrama de caja negra como se puede ver en la Figura 9, que consta de una señal de entrada de reloj proporcionada por el sistema, una señal de reset, señal de fin de lectura, vector de datos de entrada, señal de activación de periférico (CS), señal de reloj de periférico (spi_clk) y la salida de señal (mosi)

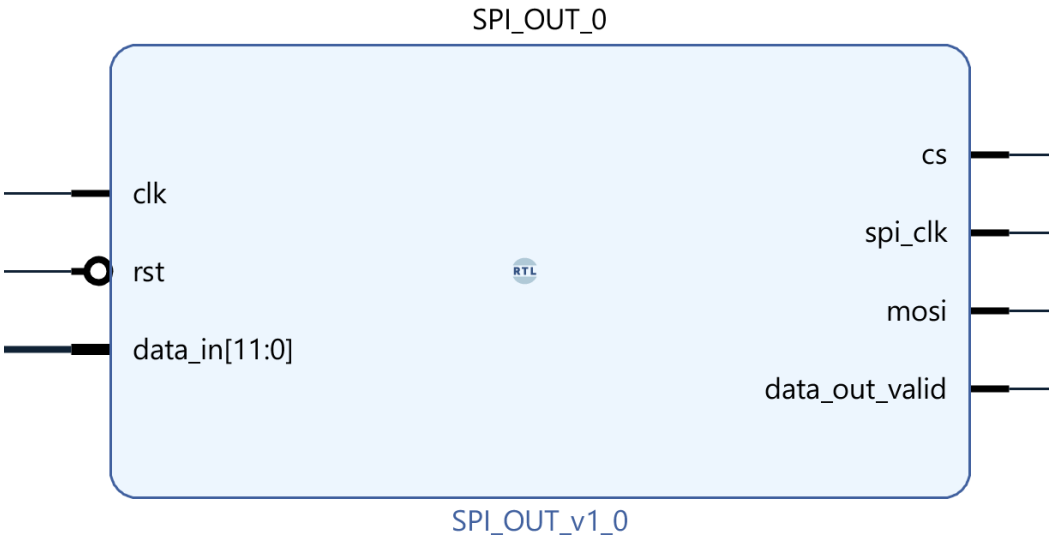


Figura 9. *Modelo de caja negra de protocolo de escritura por SPI modificado* (Fuente Propia)

Posteriormente se diseña una máquina de estados finitos como se puede ver en la Figura 10 que gobernará la ejecución del bloque de hardware

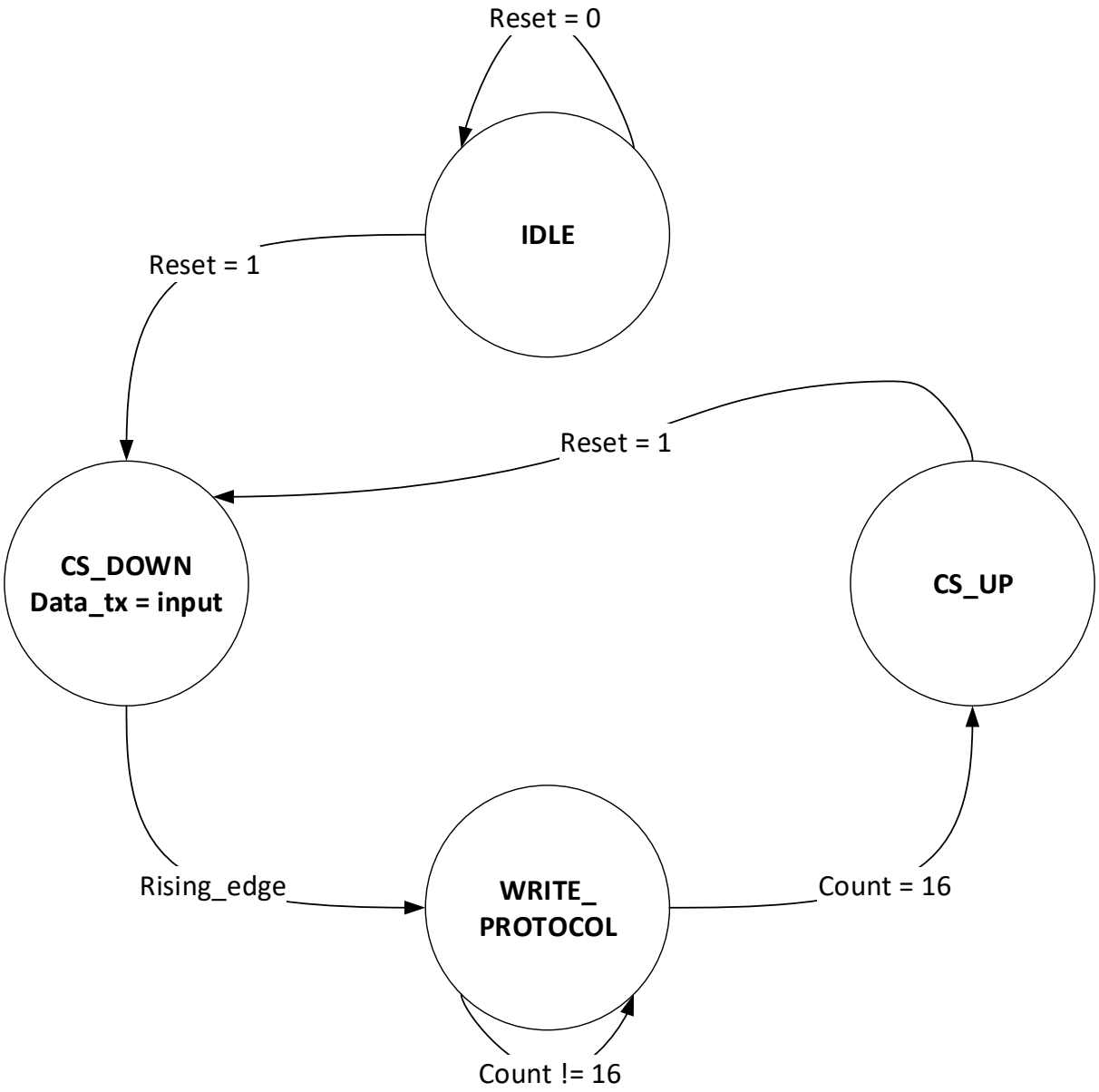


Figura 10. *Máquina de estados finitos del protocolo de escritura* (Fuente Propia)

En el Anexo II. Código VHDL para escritura del protocolo de señales del convertidor digital-analógico, se adjunta el código de VHDL que describe esta máquina de estados

Se juntan todos los bloques de hardware y se los interconecta con el procesador del ARM, resultando en un sistema como se muestra en la Figura 11

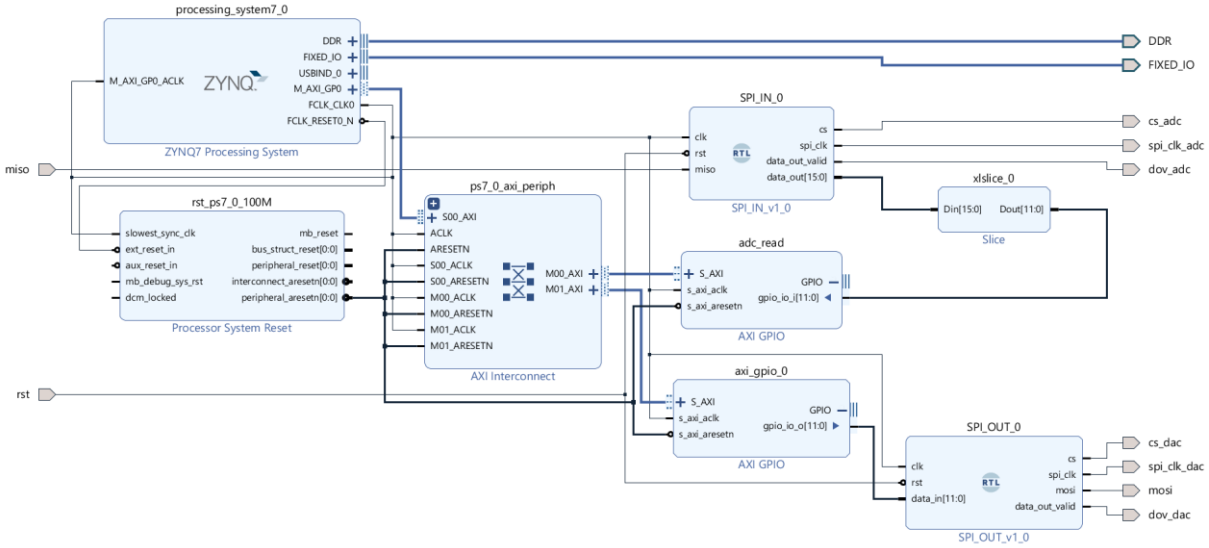


Figura 11. *Diagrama de bloques que definen el comportamiento del sistema de adquisición* (Fuente Propia)

4.1. Diseño e implementación de sistema DSP tradicional

El sistema de procesamiento tradicional consta de un filtro paso banda entre 20 y 500 Hz en estructura FIR, un filtro de promediados con una ventana de 7 muestras. Los filtros fueron diseñados originalmente en Python utilizando vectores de simulación y se implementan en el sistema PYNQ.

Los filtros FIR (Respuesta Infinita al Impulso), consisten en la convolución de una ventana basada en la función sinc: $h[i] = \frac{\sin(2\pi f_c i)}{i\pi}$ con la señal que se desea filtrar. De manera similar el filtro de promediado (Moving Average) se puede considerar la convolución de una ventana de tamaño M llena de elementos 1/M (Smith, 1997).

La estructura del filtro que se va a implementar, para que sea capaz de procesar y operar en tiempo real, es la conocida como estructura horizontal del filtro, presentada en la Figura 12, en la cual se llena una serie de elementos con un retraso definido por Z^{-1} que multiplica a los coeficientes de filtrado representados por W_n y todos los elementos son sumados para generar una salida (Mahmoud & Zhang, 2013), la ecuación resumida de operación se puede expresar como $\sum W * X$ que recuerda en gran medida a la operación hacia delante de un perceptrón simple. Consta de 61 coeficientes de filtrado que modelan un filtro paso-banda entre 20 y 500Hz

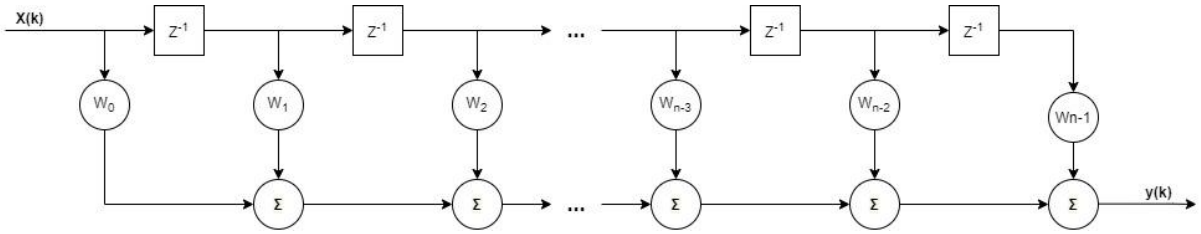


Figura 12: *Filtro FIR en estructura horizontal* (Fuente Propia)

Posteriormente los resultados del filtro FIR pasan a un filtro de promediado (moving average) con una ventana de 20 elementos, de los cuales previamente se obtiene su valor absoluto a manera de rectificación.

El código del diseño y pruebas de los filtros se encuentra en el Anexo III. Diseño de filtros tradicionales.

La implementación sobre el sistema embebido se muestra a continuación:

1. Se importan las librerías y estructura de hardware utilizada por el dispositivo.

```
from pynq import Overlay
overlay = Overlay('./PYNQ_read.bit')
ADC = overlay.adc_read
DAC = overlay.axi_gpio_0
import numpy as np
import matplotlib.pyplot as plt
```

2. Importar coeficientes (Anexo IV. Coeficientes de filtrado que utilizará el sistema FIR).

Estos coeficientes se obtienen desde fiiir.com estableciendo una frecuencia de muestreo de 2kHz, y bandas de levantamiento y asentamiento de 60 Hz. Se obtienen curvas de filtrado como las que se ven en la Figura 13.

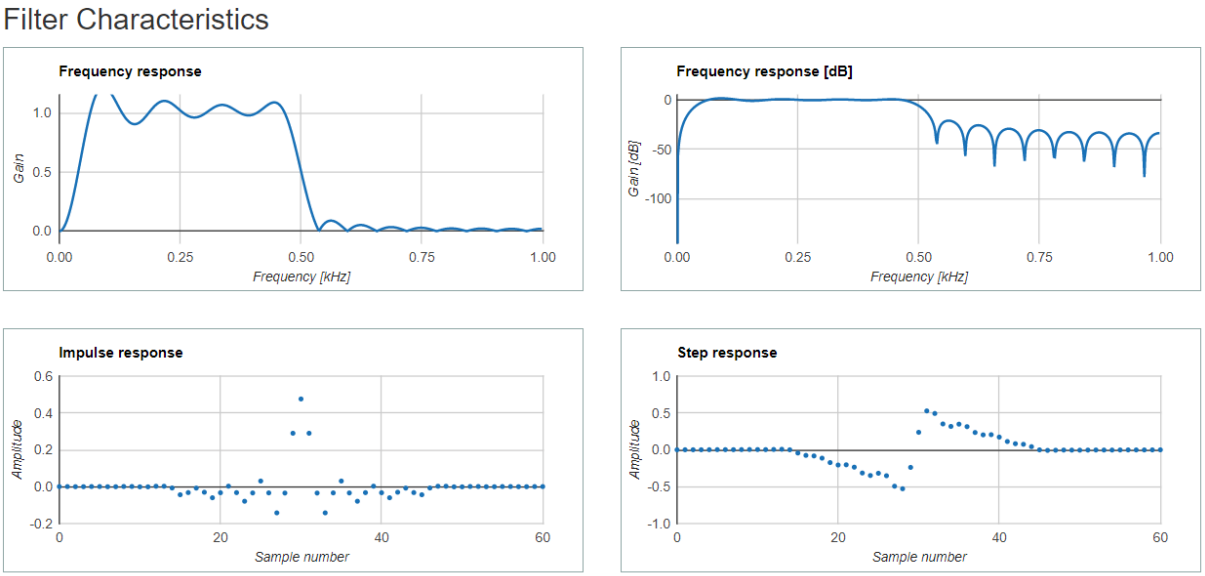


Figura 13. *Respuestas al impulso y escalón del filtro diseñado* (Fuente Propia)

3. Definir tamaños de los vectores de procesamiento.

```
N_fir = len(taps)
y = np.zeros(N_fir)
i1 = 0
i2 = 0
#3 tamaño de ventana para moving average
M = 20
muestras = np.zeros(M)
```

4. Implementación de los filtros de procesamiento en tiempo real.

```
while(True):
    #Lectura a tiempo real de los valores
    lect = ADC.read()
    #Llenado del primer vector de filtrado
    if i1 < N_fir:
        y[i1] = lect
        i1 = i1 + 1
    else:
        #shift register
        for k in range(N_fir-1):
            y[k] = y[k+1]
        y[N_fir-1] = lect

        #Convolución del vector de filtrado con los coeficientes
        filtered_FIR = 0
        for j in range(N_fir-1):
            filtered_FIR += y[j] * taps[j]

        #rectificación por valor absoluto
        filtered_FIR = abs(filtered_FIR)

        #Llenado del vector de filtrado por Moving Average

        if i2 < M:
            muestras[i2] = filtered_FIR
            i2 = i2 + 1

        else:
            #shift register
            for k in range(M-1):
                muestras[k] = muestras[k+1]
            muestras[M-1] = filtered_FIR

            filtered_MA = 0
            for j in range(M-1):
                filtered_MA += muestras[j]
            filtered_MA /= M
            #amplificación
            filtered_MA *= 6
            DAC.write(0x0000,int(filtered_MA))
```

Para evitar retardos de propagación y procesamiento en los vectores no se utilizó ninguna librería especial de Python para realizar las convoluciones o los procesos de filtrado ya que se implementaron todos los algoritmos tomando en consideración su comportamiento en tiempo real.

4.2. Resultados de DSP tradicional

La evaluación de funcionamiento se analiza inicialmente con un análisis temporal y visual, utilizando un osciloscopio para determinar el comportamiento de las señales.

4.2.1. Frecuencias bajo la frecuencia de corte

En la Figura 14 se muestra la respuesta del filtro de DSP tradicional trabajando debajo de la frecuencia de corte diseñada, a 1Hz se aprecia el rechazo de banda de este filtro.

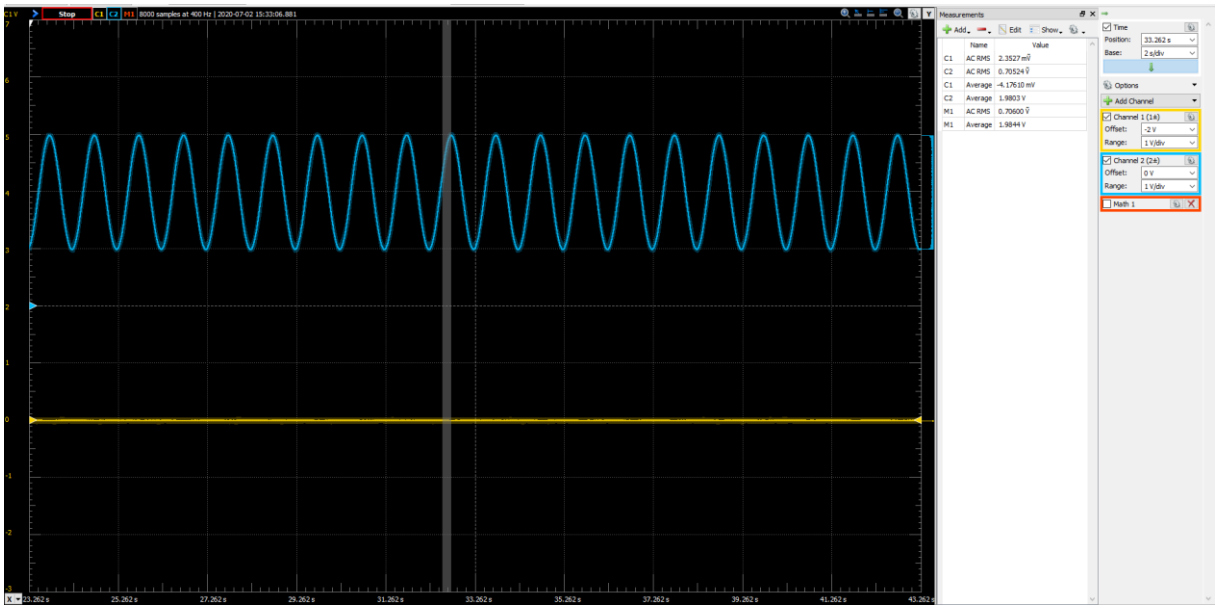


Figura 14: *Señal Generada a 1Hz [azul] y salida del filtro [amarillo]* (Fuente Propia)

4.2.2. Frecuencias sobre la frecuencia de corte baja

En la Figura 15 se aprecia la respuesta del filtro DSP operando sobre la frecuencia de corte baja, 20Hz, y se aprecia una atenuación mínima, y la rectificación de media onda de la señal.

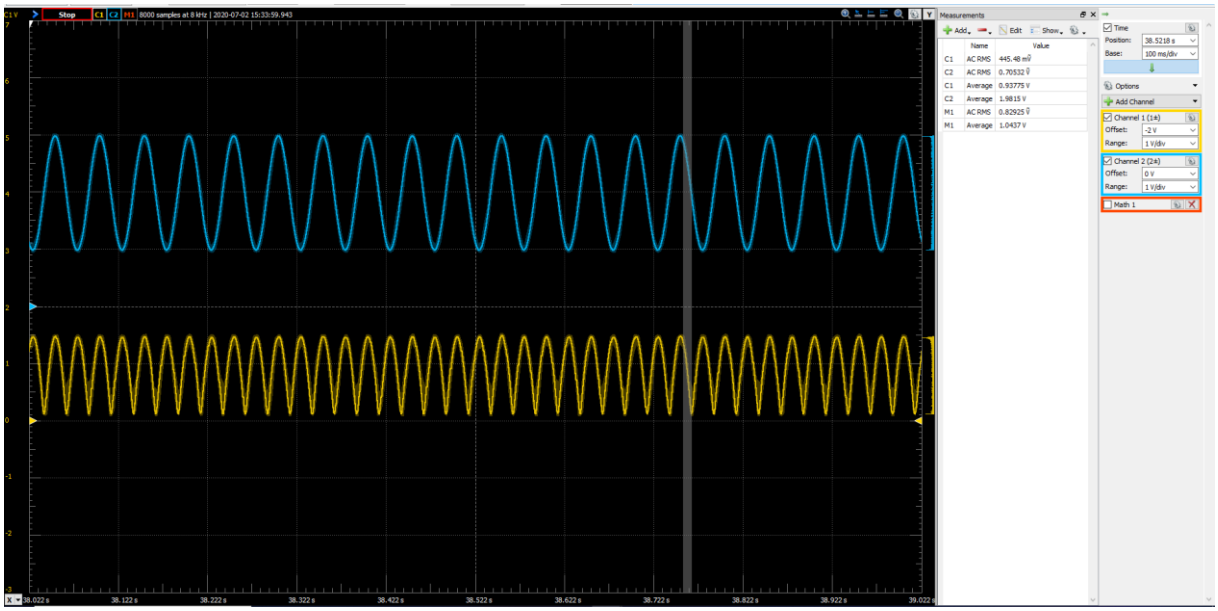


Figura 15: *Señal Generada a 20Hz [azul] y salida del filtro [amarillo]* (Fuente Propia)

4.2.3. Frecuencias en banda de paso

En la Figura 16 se puede observar la operación del filtro DSP en la banda de paso, analizando una señal de 100Hz. En estas señales ya es posible apreciar los errores de cuantización y retardos de procesamiento causados por el sistema DSP.

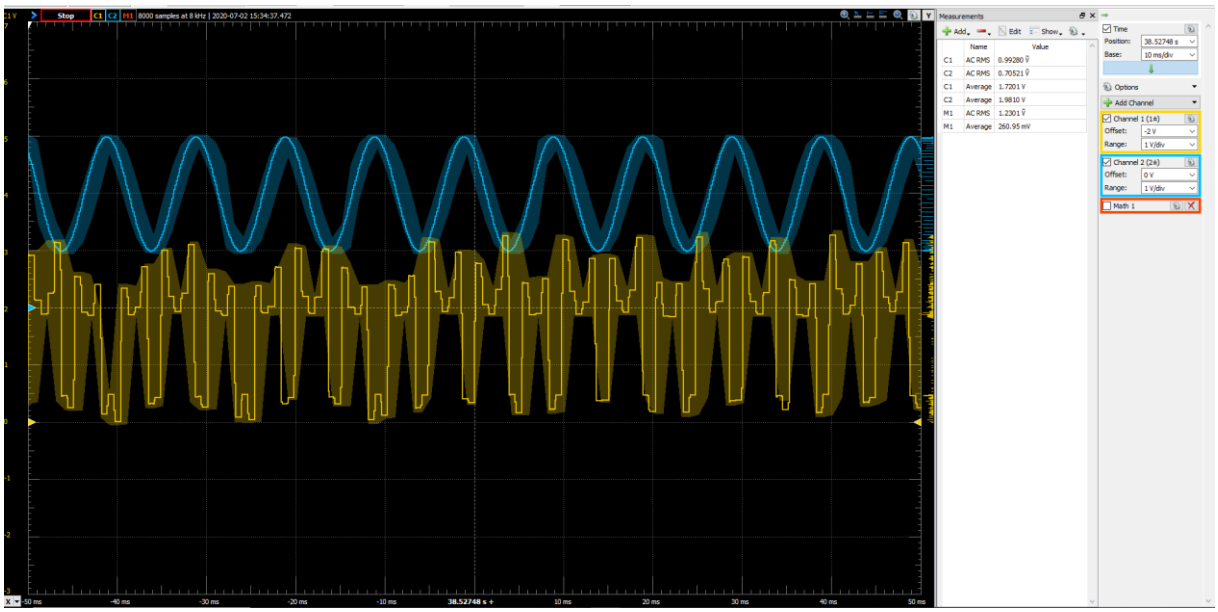


Figura 16: *Señal generada a 100Hz [azul] y salida del filtro [amarillo]* (Fuente Propia)

4.2.4. Frecuencias en la frecuencia de corte superior

En La Figura 17 se observa la respuesta del filtro DSP operando con una señal de 500Hz, en este caso la señal ya se ve cortada casi por completo por la banda superior, considerando los errores de cuantización y los retardos de procesamiento que causa el sistema DSP.

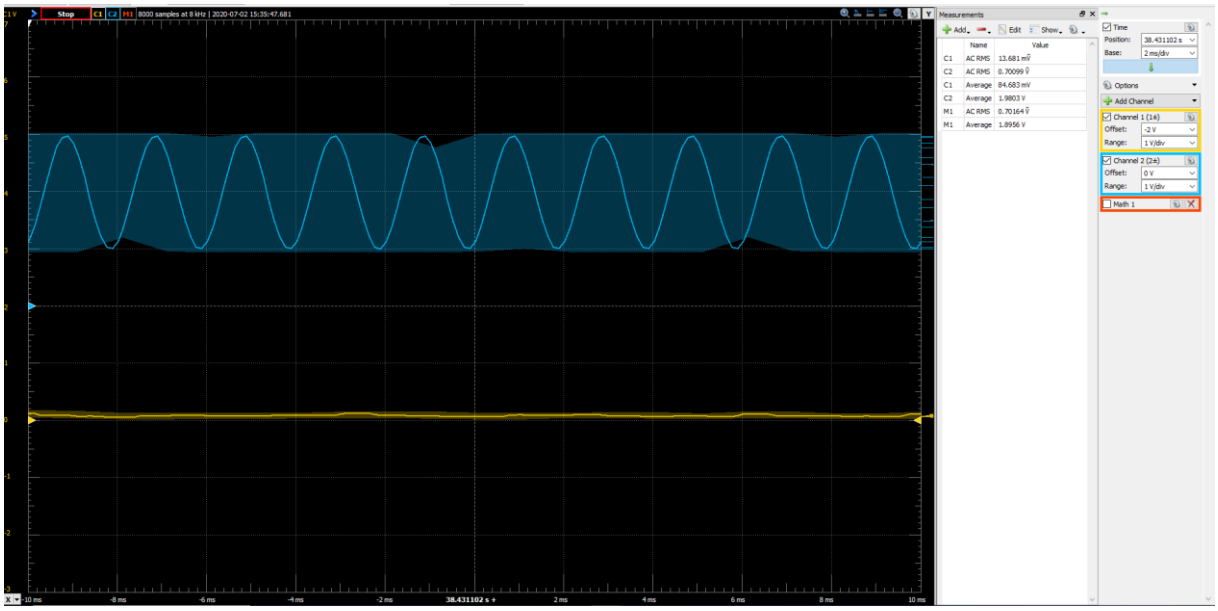


Figura 17: *Señal generada a 500Hz [azul] y la salida del filtro [amarillo]* (Fuente Propia)

4.2.5. Pruebas con señal EMG obtenida de base de datos

En la Figura 18 se muestra la generación de una señal analógica que se reconstruye desde una base de datos de EMG cruda. Se evidencia el filtrado y rectificación de la señal, elimina parcialmente la información indeseada de la señal y amplifica los picos que se pueden considerar de interés, pero aumenta considerablemente el ruido debido al promediado.

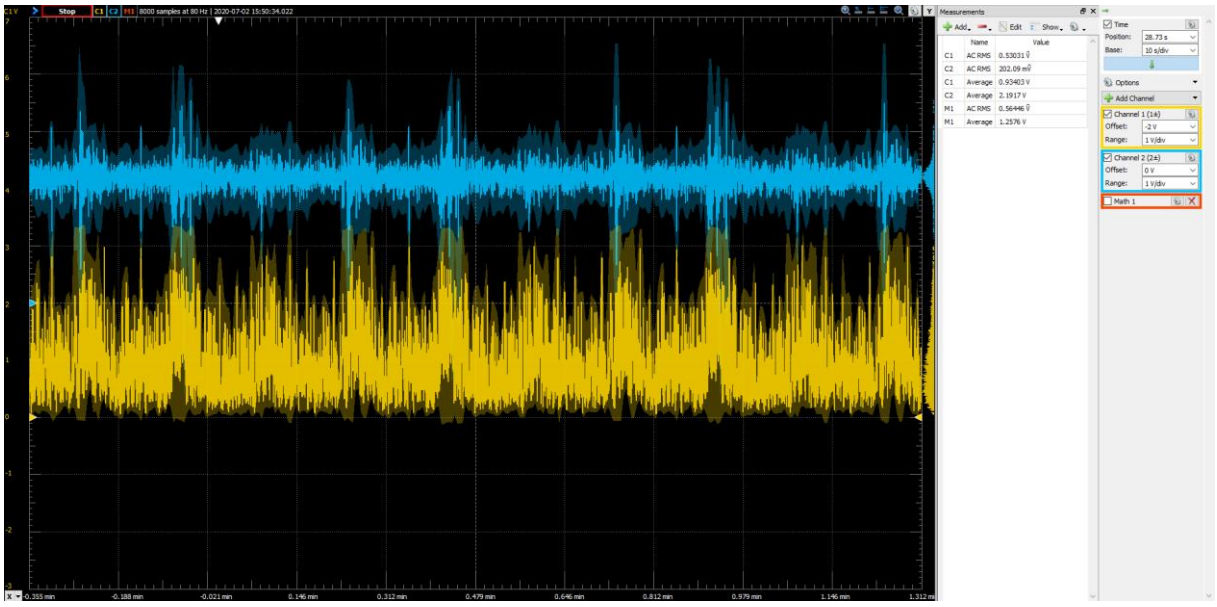


Figura 18: *Señal sEMG reconstruida desde base de datos [azul] y respuesta del filtro [amarillo]* (Fuente Propia)

4.3. Diseño e implementación de Filtro RLS 'on-the-edge'

El filtrado adaptativo es una técnica de *'Online Learning'* ya que entrena sus parámetros mientras adquiere la información a diferencia del aprendizaje automático que por lo general se entrena con todo el conjunto de datos o por lo menos con un mini lote de datos, pero el filtrado adaptativo se puede considerar una técnica de Inteligencia Artificial ya que busca minimizar una señal de error utilizando el descenso estocástico del gradiente (SGD). En la Figura 19 se muestra la estructura general de un algoritmo de filtrado adaptativo (Diniz, 2008).

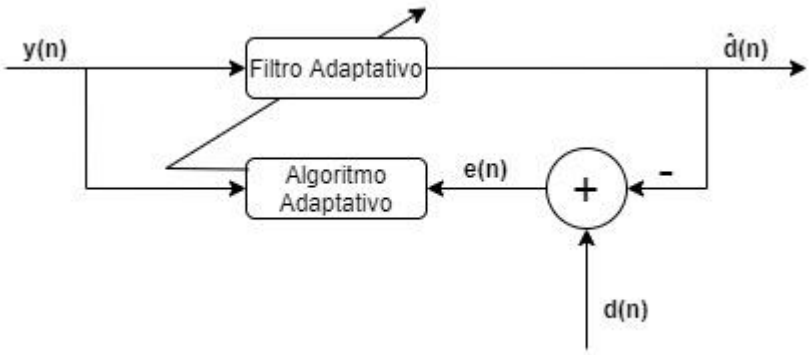


Figura 19: *Estructura general de un filtro adaptativo* (Fuente Propia)

La diferencia más importante entre un filtro adaptativo y un filtro tradicional es que el filtro adaptativo cambia sus coeficientes y forma de comportarse en el tiempo. Los coeficientes se calculan cuando el filtro se implementa y se ajustan durante su fase de aprendizaje.

El algoritmo de filtrado RLS (Recursive Least Square) busca la minimización de la suma de los cuadrados de las diferencias entre la señal deseada y la salida del filtro, actualizándose de manera iterativa conforme adquiera nueva información. Este algoritmo resuelve la estimación de mínimos cuadrados de manera recursiva (Diniz, 2008; Limem, Hamdi, & Maaref, 2016; Mugdha, Rawnaque, & Ahmed, 2015).

Las ecuaciones que se utilizarán en este filtro están basada en los trabajos de (Diniz, 2008; Limem et al., 2016; Mugdha et al., 2015; Singh, Bhole, & Sharma, 2017):

1. Inicialización

   1.1. Determinar el tamaño de ventana del filtro **M**

   1.2. Determinar el factor de olvido $0 \ll \lambda < 1$

   1.3. Determinar el factor de regularización δ que es la inversa de la potencia estimada de entrada

   1.4. Creación de la matriz de autocorrelación inversa

$$\boldsymbol{P}(-1) = \delta^{-1} * I$$

   1.5. Creación de matrices de pesos

$$\boldsymbol{W}(-1) = \boldsymbol{X}(-1) = [0\ 0\ 0\ \ldots M_{-1}]^T$$

2. Algoritmo de filtrado (repetir)

   2.1. Lectura de entrada y valor estimado, X y d respectivamente

   2.2. Cálculo de la salida del filtro

$$y(n) = \boldsymbol{W}(n) \cdot \boldsymbol{X}(n)$$

   2.3. Cálculo del error

$$e = d - y$$

   2.4. Actualización de la matriz de autocorrelación inversa

$$\boldsymbol{P}(n) = \left(\frac{1}{\lambda}\right) * \left(\boldsymbol{P}(n-1) - \frac{\boldsymbol{P}(n-1) \cdot \boldsymbol{X}(n) \cdot \boldsymbol{X}^T(n) \cdot \boldsymbol{P}(n-1)}{\lambda + \boldsymbol{X^T}(n) \cdot \boldsymbol{P}(n-1) \cdot \boldsymbol{X}(n)}\right)$$

   2.5. Actualización de pesos por descenso del gradiente (SGD)

$$dw = \boldsymbol{P}(n) \cdot \boldsymbol{X^T}(n) * e$$

$$\boldsymbol{W}(n) = \boldsymbol{W}(n) + dw$$

Para esta aplicación se utiliza una estructura lineal de filtro FIR y RLS, lo que permitirá un aprendizaje en tiempo real de las variaciones respecto al ruido. La estructura de cancelación adaptativa de ruido se toma de (Gerardo, C., & Velazquez, 2011), presentado en la Figura 20.

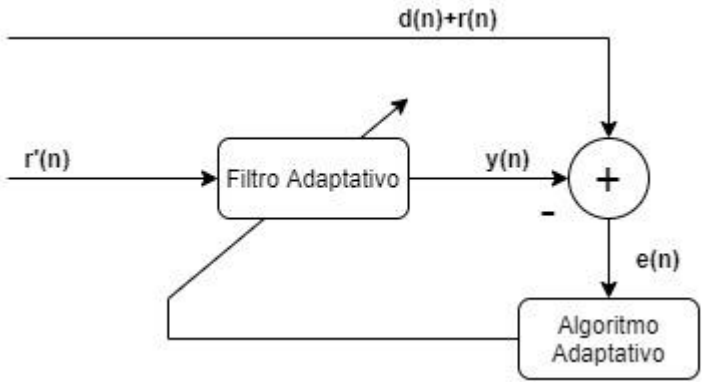


Figura 20: *Estructura de cancelación de ruido utilizando filtros adaptativos* (Fuente Propia)

En esta arquitectura la entrada al filtro es la señal modelada o medida de ruido $r'(n)$, la señal deseada es la señal adquirida (que se entiende que es la señal deseada añadida con ruido) $d(n) + r(n)$, por lo tanto, la salida $y(n)$ del filtro sería el modelo del ruido, y la salida $e(n)$ corresponde a la señal filtrada.

El modelo de la señal a eliminar se considera como ruido blanco ya que la simulación de una línea base de EMG es computacionalmente excesivo, como se presenta en el Anexo V. Simulación de señales SEMG.

En la Figura 21 se presenta la arquitectura de un filtro RLS horizontal, el cual puede procesar en tiempo real los datos adquiridos. Esta arquitectura recuerda en gran manera a una unidad neuronal de un perceptrón, ya que en el paso hacia adelante realiza la sumatoria de las multiplicaciones de las entradas por sus pesos $\sum_k X(k) * W(k)$, lo compara con la señal deseada $d(k)$ y genera una señal de error $\epsilon(k)$ que se retro propaga por la estructura con un algoritmo basado en el descenso estocástico del gradiente (SGD).

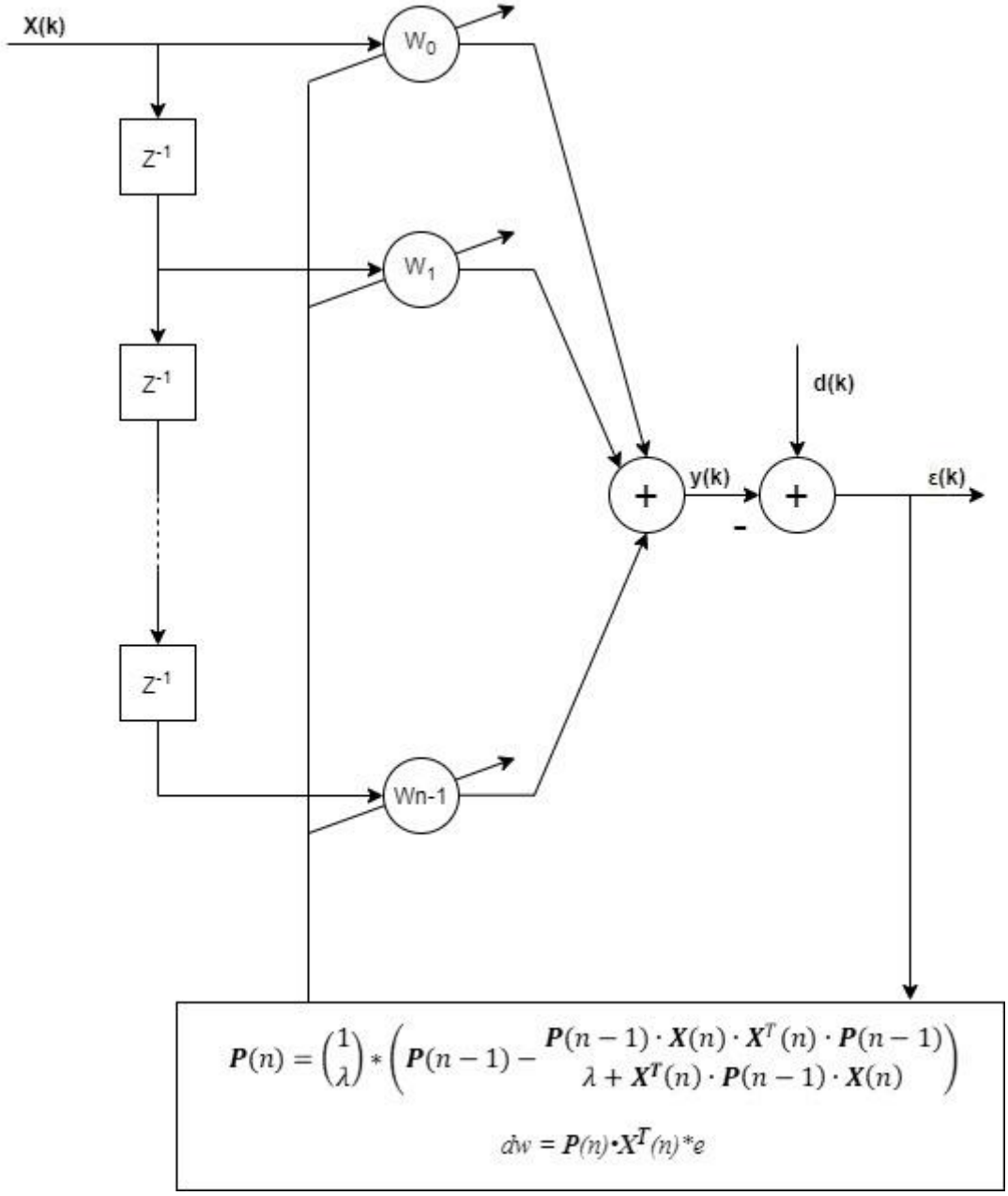


Figura 21: *Filtro RLS en arquitectura horizontal* (Fuente Propia)

El código del diseño y pruebas de los filtros se encuentra en el Anexo VI. Diseño de filtro RLS, aquí se puede apreciar la generación estimada del error del algoritmo y el proceso de filtrado en 'tiempo real' simulado de la señal.

El código de implementación en el sistema embebido se presenta a continuación:

1. Inicializar variables

```
from pynq import Overlay
overlay = Overlay('./PYNQ_read.bit')
ADC = overlay.adc_read
DAC = overlay.axi_gpio_0
```

```
import numpy as np
import matplotlib.pyplot as plt
#Generación del ruido correlacionado al que interfiere con la señal deseada
noise = np.random.normal(0,50,10000)
```

2. Algoritmo RLS

2.2. Hiperparámetros del filtro

```
#Tamaño de ventana del filtro
M = 100
#Factor de regularización
delta = 0.01
#Factor de olvido
lbda = 0.99
```

2.3. Inicialización de vectores filtro

```
#Vector de autocorrelación inversa
P = np.identity(M)
P = P*delta
#Matrices de entrada y pesos
W = np.zeros(M)
X = np.zeros(M)
Y = np.zeros(M)
D = np.zeros(M)
#variables auxiliares
i = 0
j = 0
```

2.4. Implementación de algoritmo en tiempo real

```
while(True):
    des = ADC.read()
    if i < 10000:
        inp = noise[i]
    else:
        i = 0
    i += 1

    if j < M:
        X[j] = inp
        j += 1
    else:
        #shift register
        for k in range(M-1):
            X[k]= X[k+1]
        X[M-1] = inp

        #cálculo del vector de salida
        y = np.dot(W, X)
        e = des - y
        P1 = np.dot(np.dot(np.dot(P, X), X.T), P)
        P2 = lbda + np.dot(np.dot(X, P), X.T)
        P = (1/lbda)*(P - P1/P2)
        dw = np.dot(P, X.T) * e
        W += dw

        DAC.write(0x0000,int(e))
```

4.4. Resultados de Filtrado RLS ‘on-the-edge’

La Evaluación de funcionamiento se analiza inicialmente con un análisis temporal y visual, utilizando un osciloscopio para determinar el comportamiento de las señales y el espectrograma para el análisis temporofrecuencial.

4.4.1. Pruebas de filtrado con EMG reconstruido

En la Figura 22 se muestra la eliminación de ruido blanco en la señal original y comparando el espectrograma de la Figura 23 y el de la Figura 24 se puede apreciar una variación mínima en el espectro de frecuencia. La forma de las señales mantiene su estructura general, pero se aprecian variaciones mínimas.

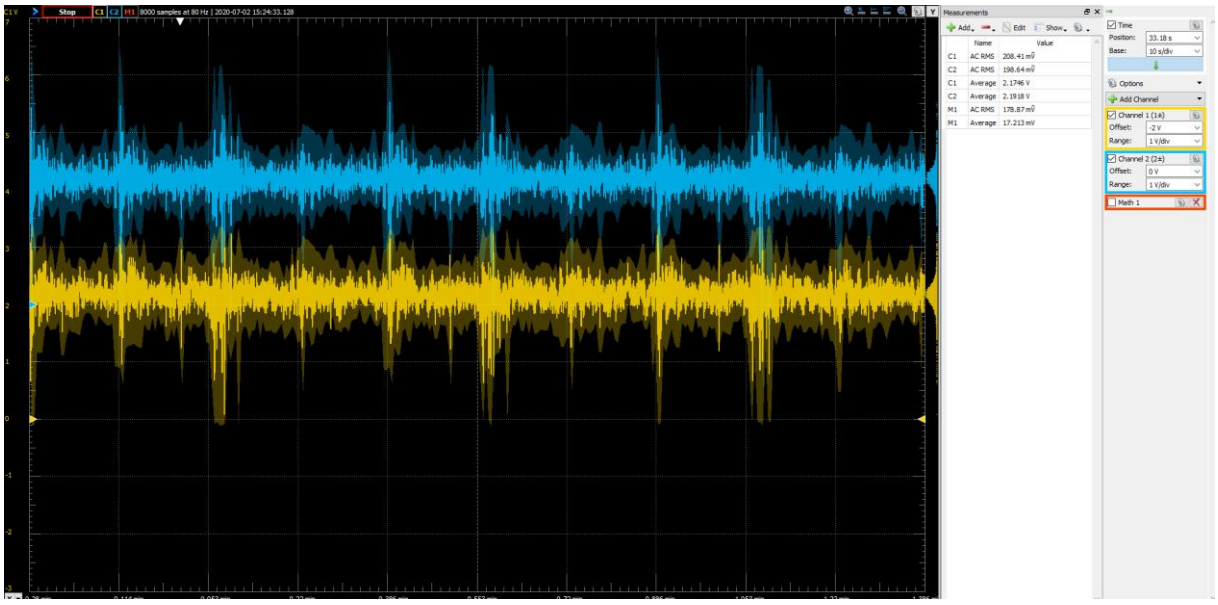


Figura 22: *Prueba de filtrado con EMG reconstruido [azul] y la salida de filtrado RLS [amarillo]* (Fuente Propia)

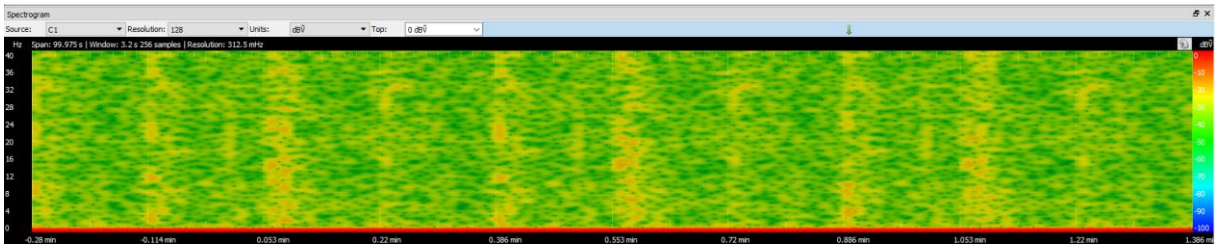


Figura 23: *Espectrograma de señal filtrada RLS* (Fuente Propia)

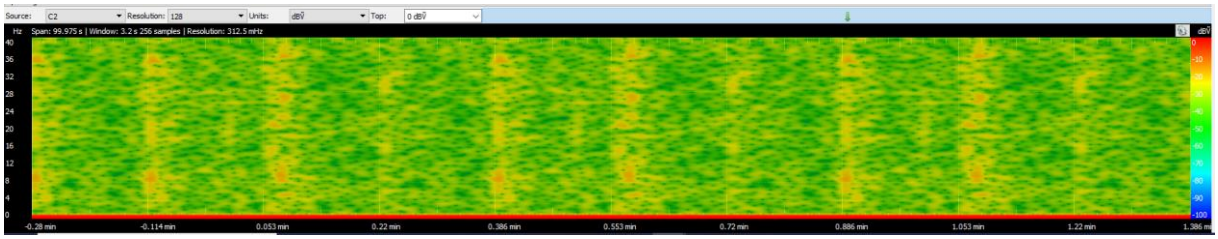


Figura 24: *Espectrograma de señal original* (Fuente Propia)

Para apreciar los efectos del filtrado adaptativo se ha tomado una ventana amplificada, presentado en la Figura 25, donde se aprecia el cambio sobre todo en la línea base del EMG que elimina el ruido blanco y refuerza el comportamiento impulsivo de una señal sEMG

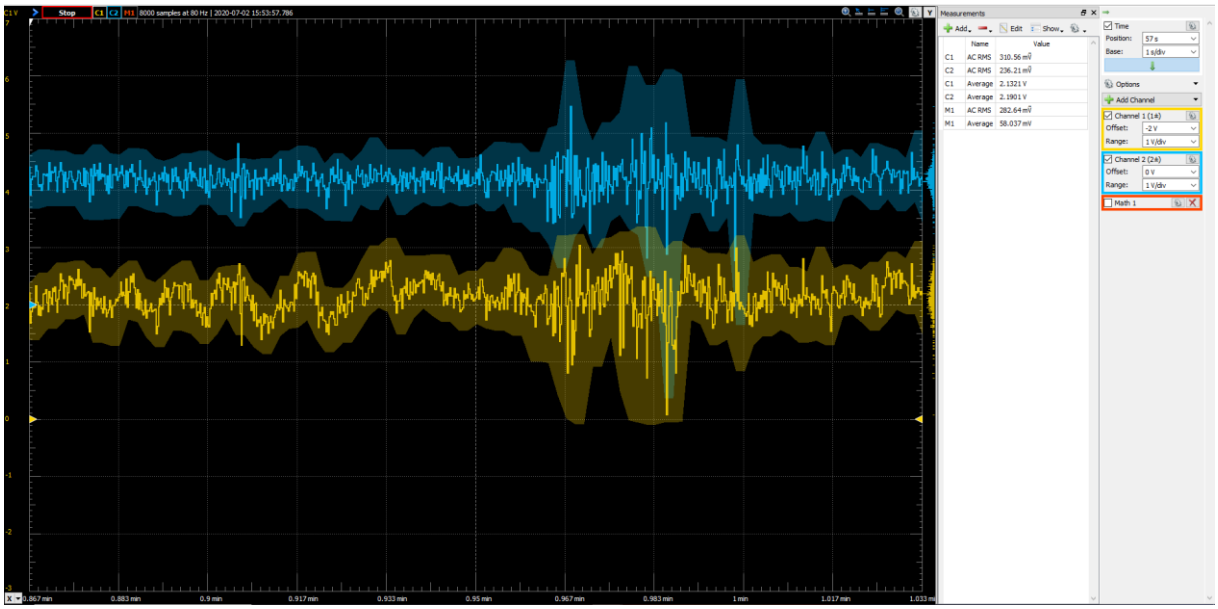


Figura 25: *Ampliación de señales de EMG reconstruido [azul] y señal filtrada RLS [amarillo]* (Fuente Propia)

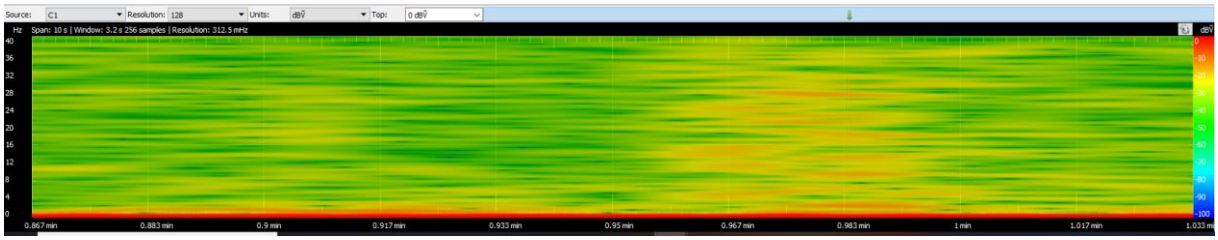


Figura 26: *Ampliación de espectrograma de señal filtrada RLS* (Fuente Propia)

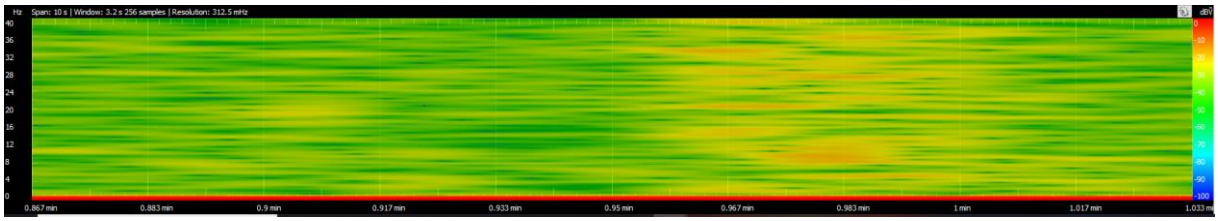


Figura 27: *Ampliación de espectrograma de señal original* (Fuente Propia)

El espectrograma de la señal filtrada, presentado en la Figura 26 muestra una distribución en frecuencia más uniforme y reforzada del momento de activación muscular que, en comparación con el espectrograma de la señal original presentado en la Figura 27, donde la señal filtrada ofrece más información útil para la utilización futura de la señal.

4.4.2. Análisis con señales no contaminadas

En el análisis con señales puras no contaminadas presentado en la Figura 28, se observa el efecto de agregación de ruido a la señal de salida, ya que las señales de ruido de ambas señales no mantienen correlación, esto demuestra la importancia de modelar adecuadamente el ruido de filtrado.

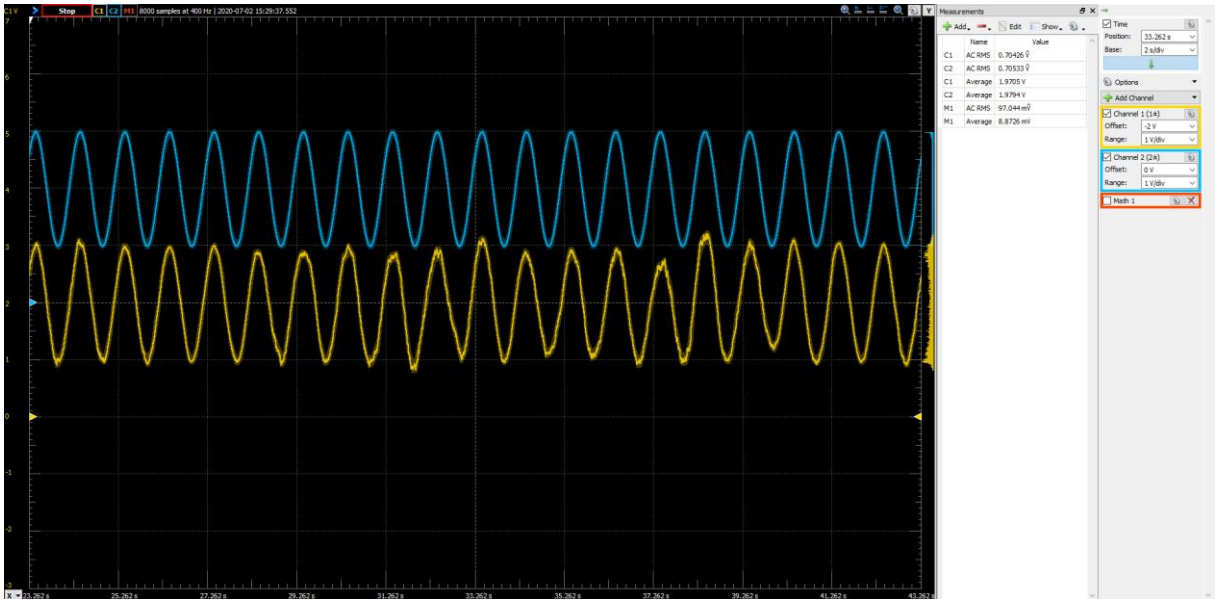


Figura 28: *Prueba de filtrado con señal pura de 1Hz [azul] y salida de filtrado RLS [amarillo]* (Fuente Propia)

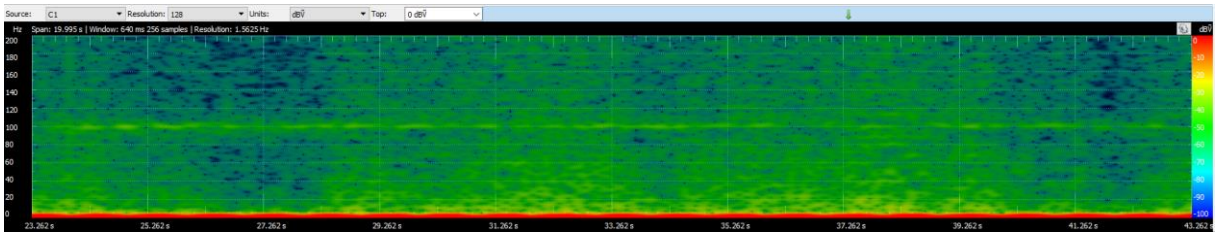


Figura 29: *Espectrograma de señal pura filtrada con RLS* (Fuente Propia)

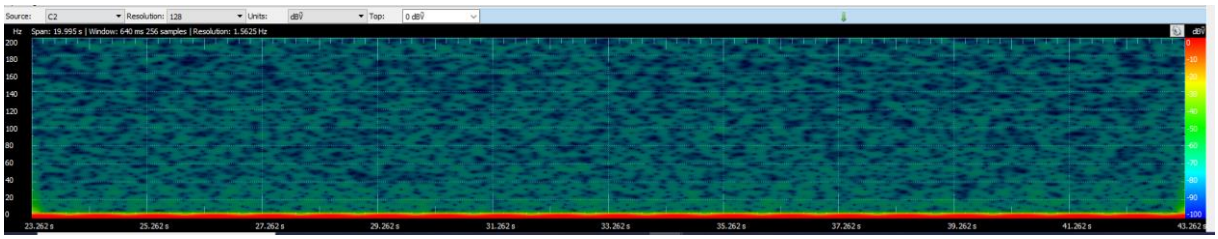


Figura 30: *Espectrograma de señal pura de 1Hz* (Fuente Propia)

El espectrograma de la señal filtrada presentado en la Figura 29 muestra la agregación de componentes frecuenciales no deseados por la no-correlación entre los ruidos en comparación al espectrograma de la señal pura presentado en la Figura 30.

4.5. Sistema distribuido de computación

Para establecer el sistema distribuido de computación, es decir que se utilicen elementos discretos especializados para diferentes tareas, la estructura de red utilizada se presenta en la Figura 31. Al sistema embebido se le agrega una tarjeta de red inalámbrica tp-link TL-WN823N, con una velocidad de transmisión a 300Mbps, que se comunicará por sockets utilizando protocolo TCP IPv4 con un servidor de cálculo establecido en un computador Lenovo con procesador Intel Xeon CPU E3-1535M v6 de 3.1 GHz, y memoria RAM de 64 Gb.

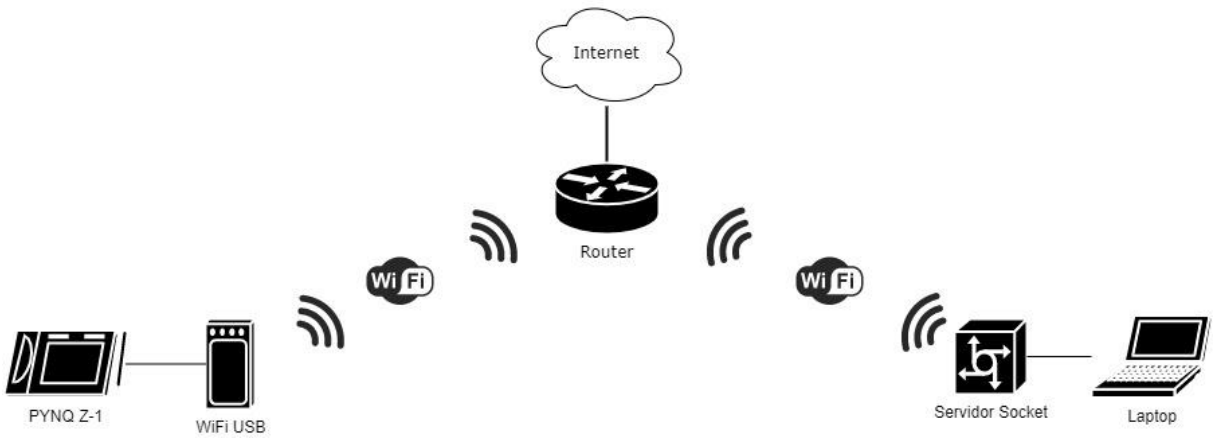


Figura 31: *Estructura de red para comunicación con servidor de cálculo* (Fuente Propia)

En ambientes de computación distribuida, una aplicación cliente-servidor puede ser diseñada utilizando programación por sockets. Un Socket es un enlace de comunicación bidireccional entre un programa de cliente y servidor que se ejecutan en un ambiente de red (Corcoran, 1998; Maata, Cordova, Sudramurthy, & Halibas, 2018). A continuación, se describe únicamente lo relevante al cliente y servidor socket implementados en python ya que el algoritmo de filtrado es el mismo presentado en Diseño e implementación de Filtro RLS.

4.5.1. Servidor Socket

El programa completo se encuentra en el Anexo VII. Programa de Servidor Socket. En este código, se establece un Socket de tipo servidor operando en IPv4 en protocolo TCP-IP, que una vez que establezca conexión con un cliente pude seguir ejecutando el resto del programa

```
import socket
import numpy as np

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #(IPV4 TCP-IP)
s.bind(('192.168.100.8', 1234)) #IP del servidor, puerto de conexión
s.listen(5) #Cola de 5 conexiones posibles que aceptará
#s.close para cerrar servidor
clientsocket, address = s.accept()
print(f"Se ha establecido una conexión desde {address}!")
```

Para recibir datos, es necesario establecer la cantidad de bytes que se van a recibir y decodificarlos para que puedan ser utilizados en el contexto del programa

```
#Recibir lectura desde SoC
ans = clientsocket.recv(8)
des = int.from_bytes(ans, byteorder = 'little')
```

Para enviar los datos, de igual manera es necesario codificarlos en una cantidad de bytes determinada

```
clientsocket.send((int(abs(e))).to_bytes(8, 'little'))
```

4.5.2. Cliente Socket

El programa completo se encuentra en el Anexo VIII. Programa de cliente Socket. En este código se establece la conexión de un cliente con el servidor Socket, determinando que el protocolo de comunicación sea TCP-IP, en IPv4

```
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('192.168.100.8', 1234)) #IP del servidor y puerto de
conexión
```

A continuación, se muestra el código de envío y recepción de datos establecido en el cliente, considerando que la cantidad de bytes sea la misma que envía y recibe el servidor

```
while True:
    #Lectura del ADC y envío al servidor
    lect = ADC.read()
    client.send((lect).to_bytes(8, 'little'))
    #Espera una respuesta del servidor y lo escribe en la DAC
    recv = client.recv(8) #Recibir bufer de 8 bytes
    escrt = int.from_bytes(recv, byteorder = 'little')
    DAC.write(0x0000,escrt)
```

4.6. Pruebas con sistema distribuido

La Evaluación de funcionamiento se analiza inicialmente con un análisis temporal y visual, utilizando un osciloscopio para determinar el comportamiento de las señales y el espectrograma para el análisis temporofrecuencial.

4.6.1. Pruebas de filtrado con EMG reconstruido

En la Figura *32* se muestra la eliminación de ruido blanco en la señal original utilizando un sistema de computación distribuido y comparando el espectrograma de la Figura *33* y el de la Figura *34* se puede apreciar una variación mínima en el espectro de frecuencia. La forma de las señales mantiene su estructura general, pero se observan errores en la comunicación de datos que se reflejan tanto en la forma de la onda o como vacíos de información en el espectrograma, esto pone en riesgo la integridad del proceso de filtrado ya que son datos críticos que no deberían perderse.

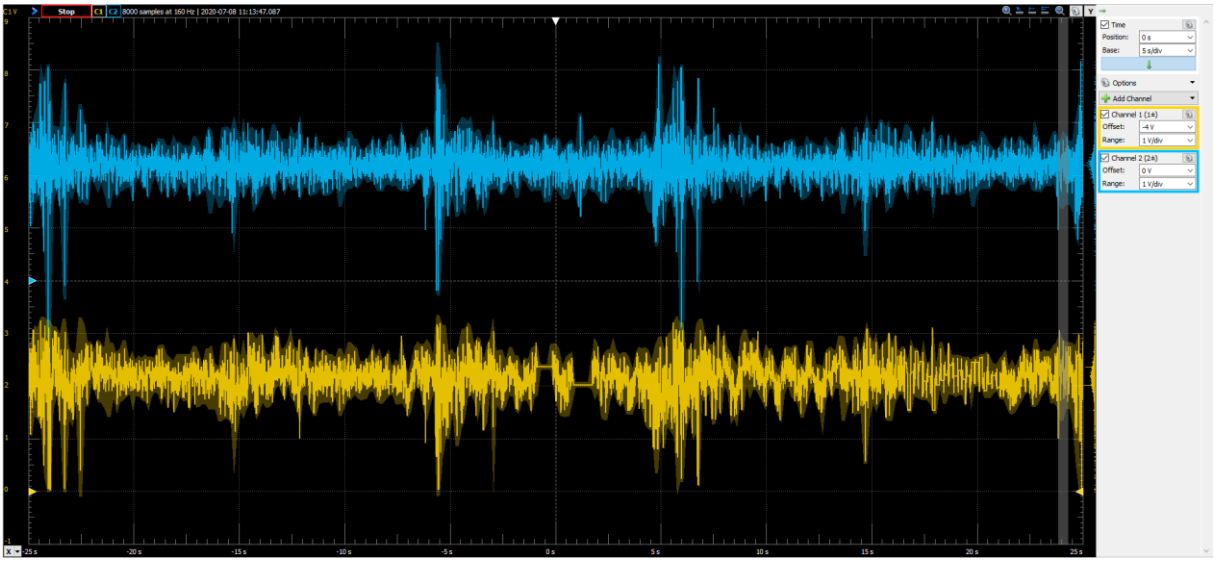


Figura 32: *Prueba de filtrado con EMG reconstruido [azul] y la salida de filtrado RLS por sistema distribuido [amarillo]* (Fuente Propia)

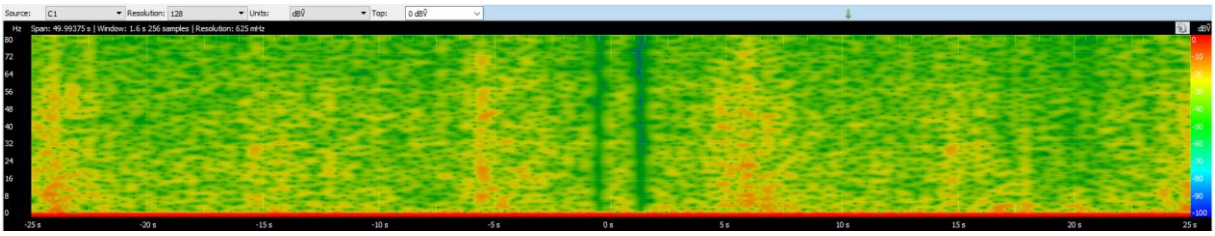


Figura 33: *Espectrograma de la señal filtrada RLS por sistema distribuido* (Fuente Propia)

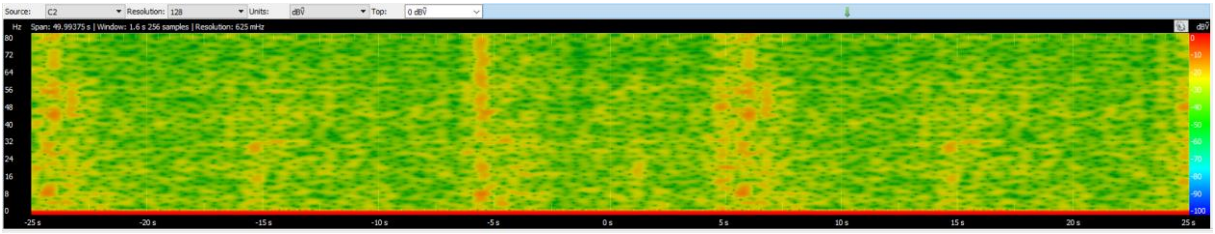


Figura 34: *Espectrograma de la señal original en el sistema distribuido* (Fuente Propia)

4.6.2. Análisis con señales no contaminadas

En la Figura *35* se aprecia que el algoritmo de filtrado está bien entrenado, aunque agrega ruido blanco a la señal ya que los ruidos de las dos señales no están correlacionados, lo interesante de este episodio captado son los múltiples errores de transmisión que representan una pérdida significativa de los datos.

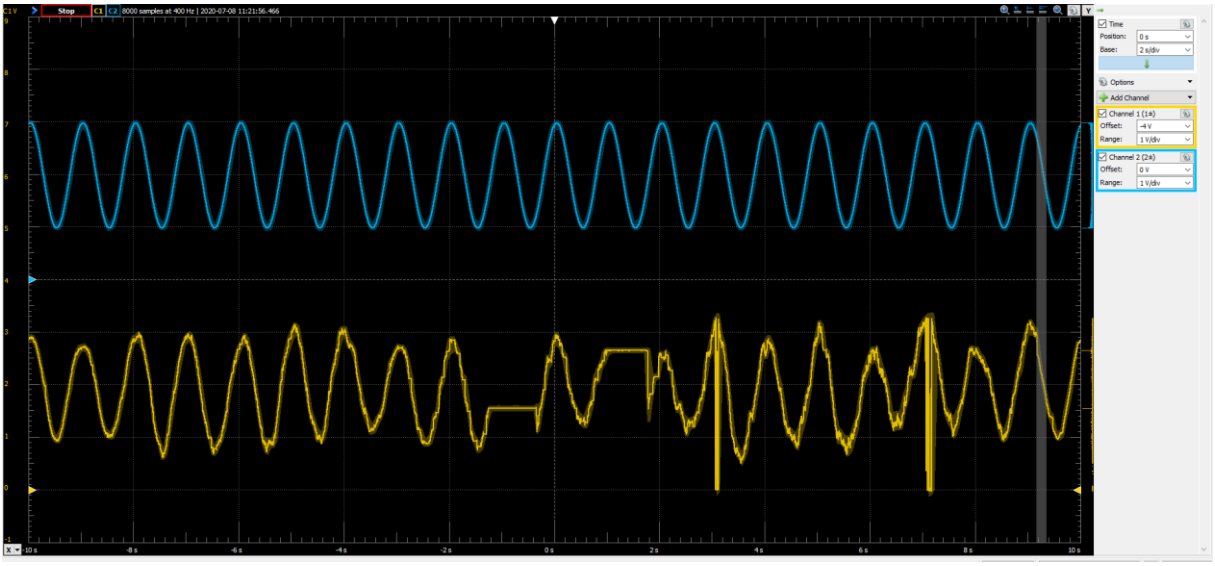


Figura 35: *Señal senoidal pura generada a 1Hz [Azul] y señal filtrada con algoritmo RLS en sistema distribuido [amarillo]* (Fuente Propia)

En los espectrogramas presentados en la Figura *36* y Figura *37* se puede observar la agregación de ruido en todas las bandas de frecuencia y eventos impulsivos de ausencia de información y de agregación masiva de información.

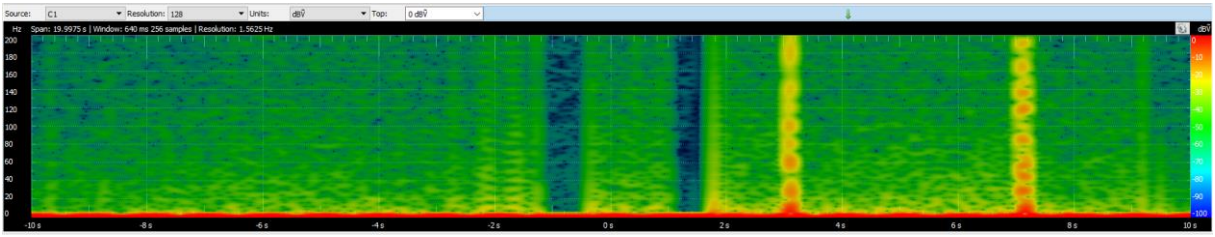


Figura 36: *Espectrograma de la señal filtrada por algoritmo RLS en sistema distribuido* (Fuente Propia)

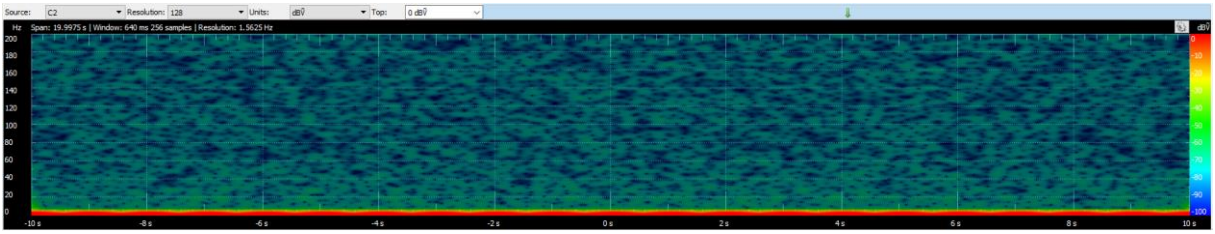


Figura 37: *Espectrograma de la señal senoidal pura a 1 Hz* (Fuente Propia)

5. Descripción de los resultados

Para la descripción de resultados se presentarán las pruebas y mediciones relacionadas, tomando en cuenta las señales en un mismo marco contextual, sus espectrogramas, señales de AC RMS y un análisis de retardo de propagación analizado con señales senoidales puras.

5.1. DSP Tradicional

En la Figura *38* y Figura *39* se muestra que el algoritmo comienza a funcionar desde que es iniciado, sin necesitar tiempos de espera

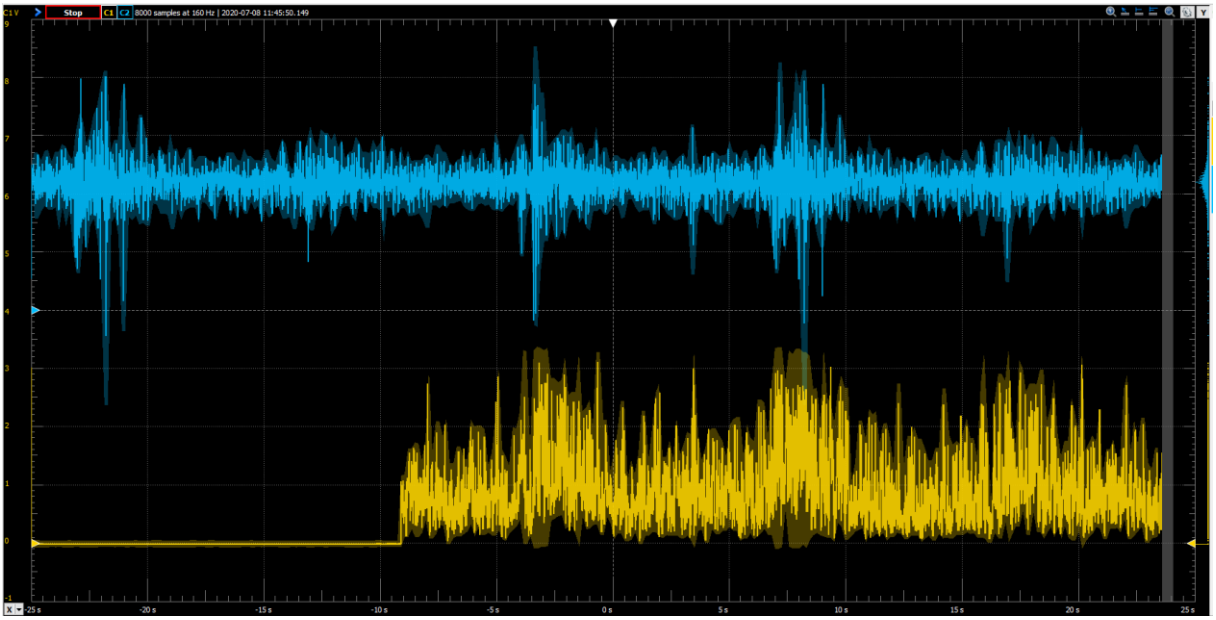


Figura 38: *Inicio de adquisición con DSP tradicional, señal original [azul] y señal filtrada [amarillo]* (Fuente Propia)

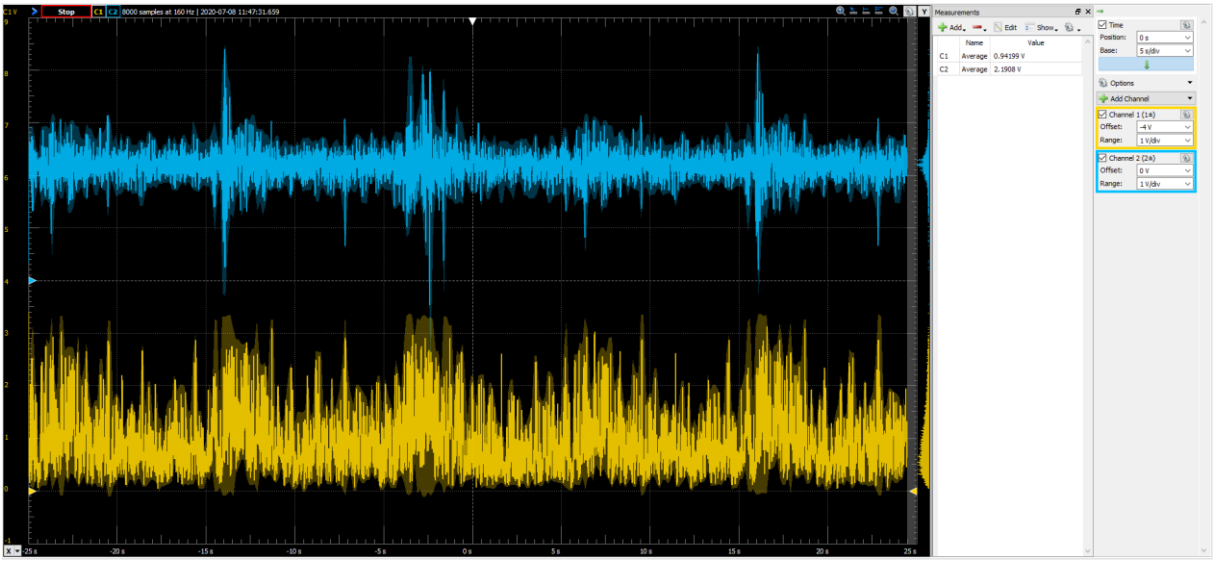


Figura 39: *Funcionamiento normal de DSP tradicional, señal original [azul] y señal filtrada [amarillo]* (Fuente Propia)

El análisis visual de las señales muestra un cambio radical en la forma de onda, rectificando los impulsos y llevándola a oscilar desde la línea 0, en vez de una componente aditiva que mueve la línea de oscilación como es el caso de la señal original, la media de la señal original es de 2.1908V y la señal filtrada es de 0.9419V, lo que comprueba lo descrito anteriormente.

En cuanto al análisis de espectros, se pueden observar en la Figura *40* y Figura *41*

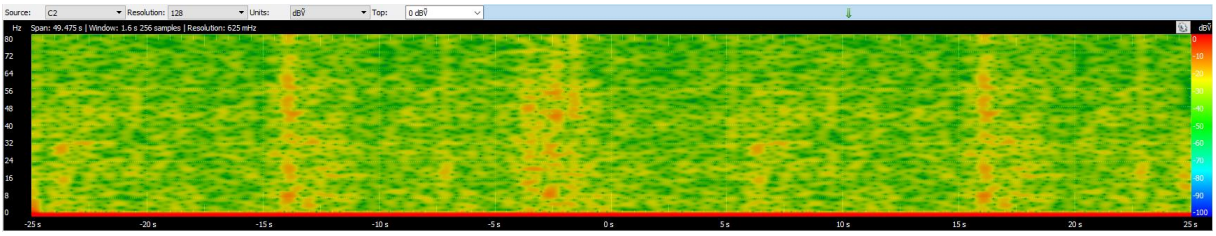


Figura 40: *Espectrograma de señal original en contexto de DSP tradicional* (Fuente Propia)

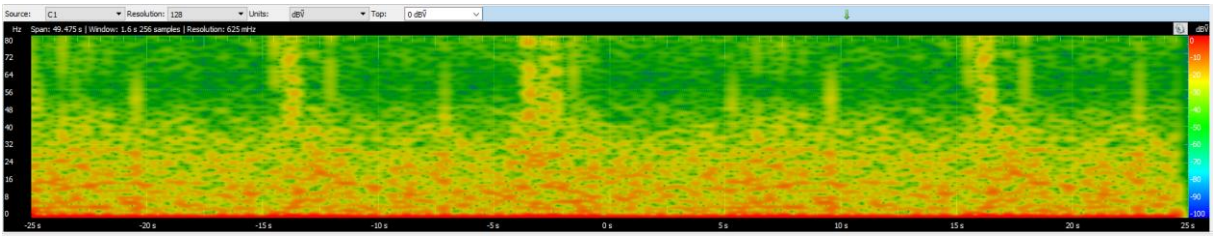


Figura 41: *Espectrograma de señal filtrada en contexto de DSP tradicional* (Fuente Propia)

En la comparativa de espectrogramas se aprecia que en el sistema filtrado se elimina buena parte de la información de alta frecuencia mientras que se contamina la señal de baja frecuencia.

El análisis visual de AC RMS mostrado en la Figura *42* muestra que ambas señales mantienen en gran manera la correlación entre sus señales, pero la señal filtrada, presentada en morado, se sobrepone a la señal original como si estuviese tratando de ser su señal portadora.

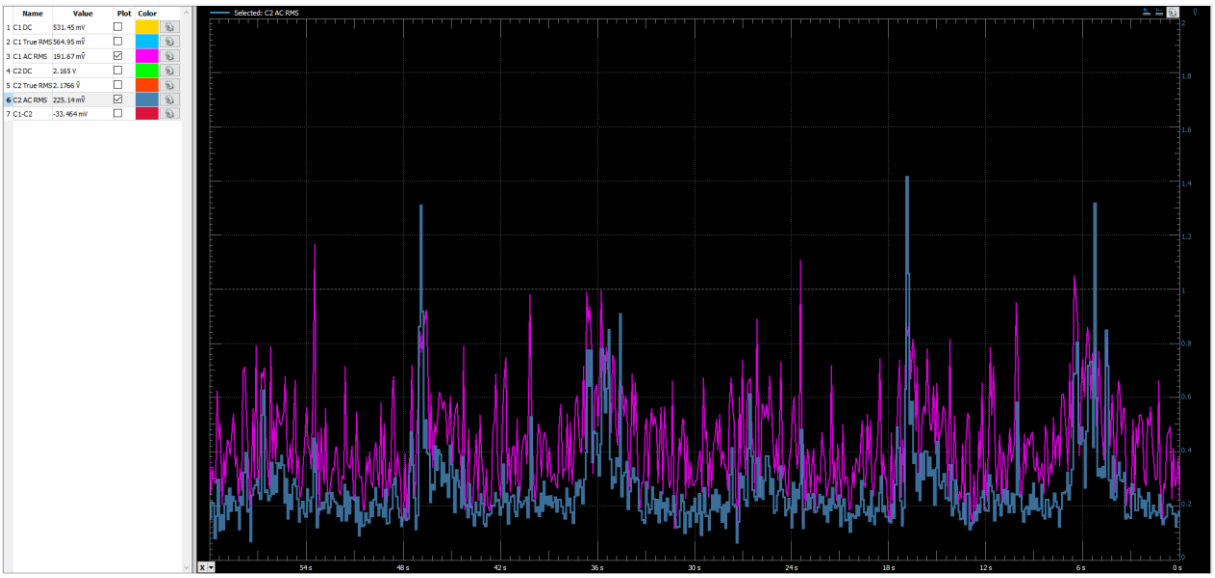


Figura 42: *AC RMS de las señales original [azul] y señal filtrada [morado] en contexto de DSP tradicional* (Fuente Propia)

En el análisis comparativo de la diferencia de la señal original con la señal filtrada que se presenta en la Figura *43* con su espectrograma en la Figura *44* es una composición aditiva entre ambas señales, lo que impide identificar adecuadamente la relación de señal a ruido. La media de dicha señal se encuentra en -1.2488V.

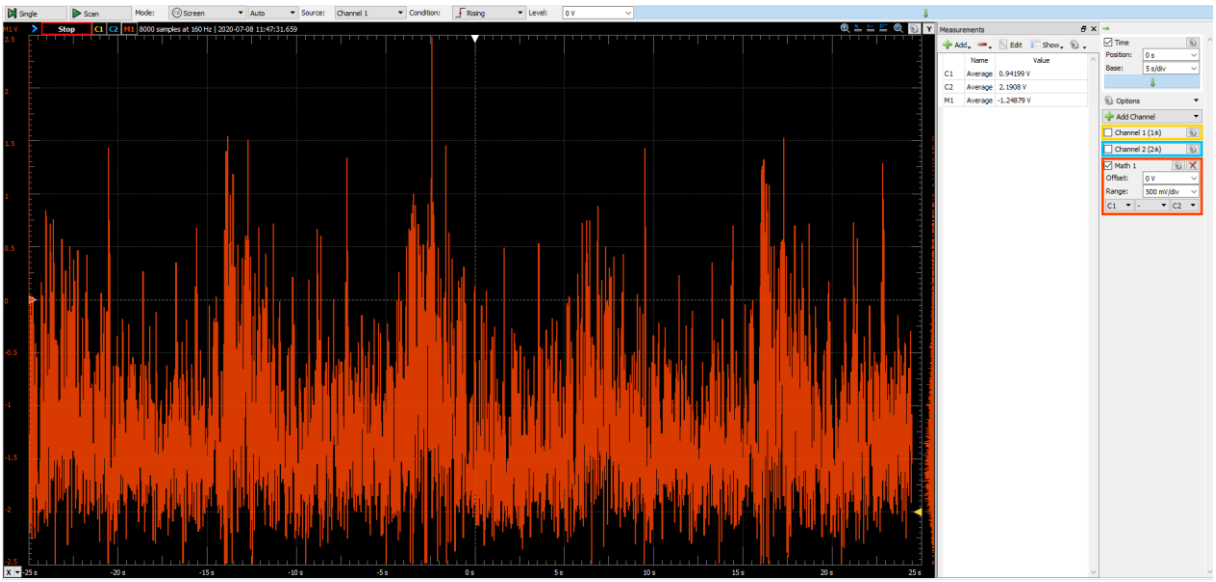


Figura 43: *Resta entre señal original y filtrada en el contexto de DSP tradicional* (Fuente Propia)

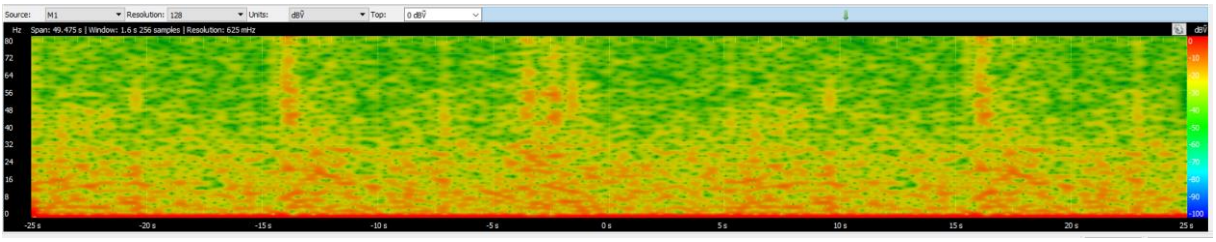


Figura 44: *Espectrograma de la señal resta entre señal original y filtrada en el contexto de DSP tradicional* (Fuente Propia)

Para el análisis de retardo temporal de las señales se analiza la medición en dos puntos diferentes, presentados en la Figura *45* y en la Figura *46*

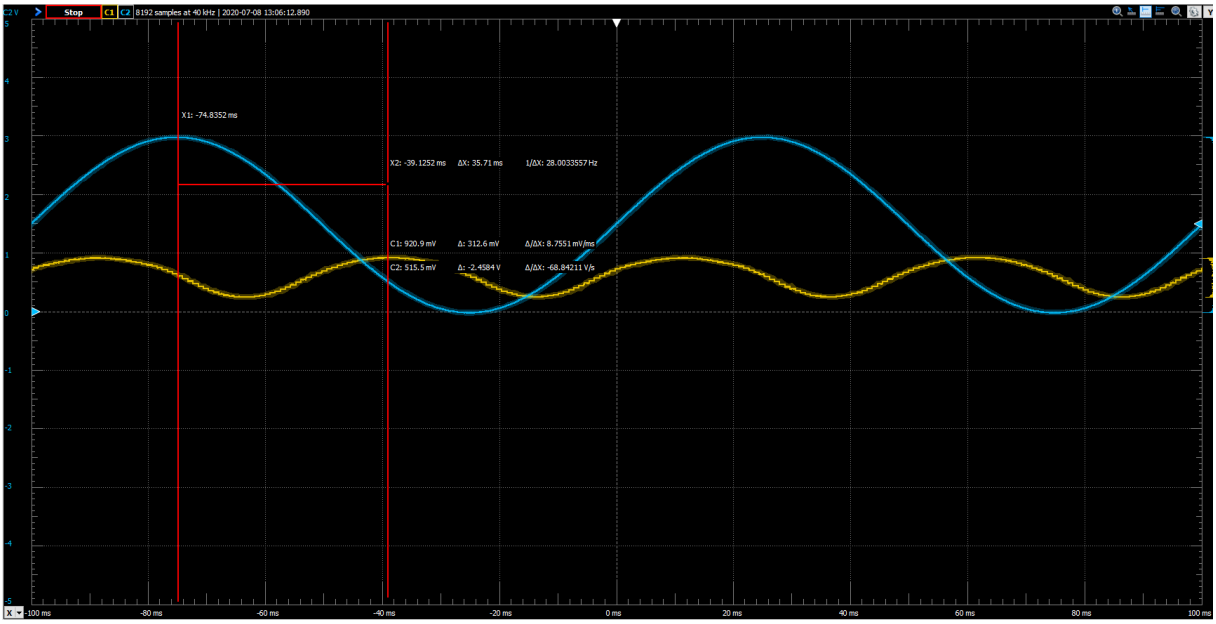


Figura 45: *Medición 1 de desfase temporal entre señal original [azul] y señal filtrada [amarillo] en el contexto de DSP tradicional* (Fuente Propia)

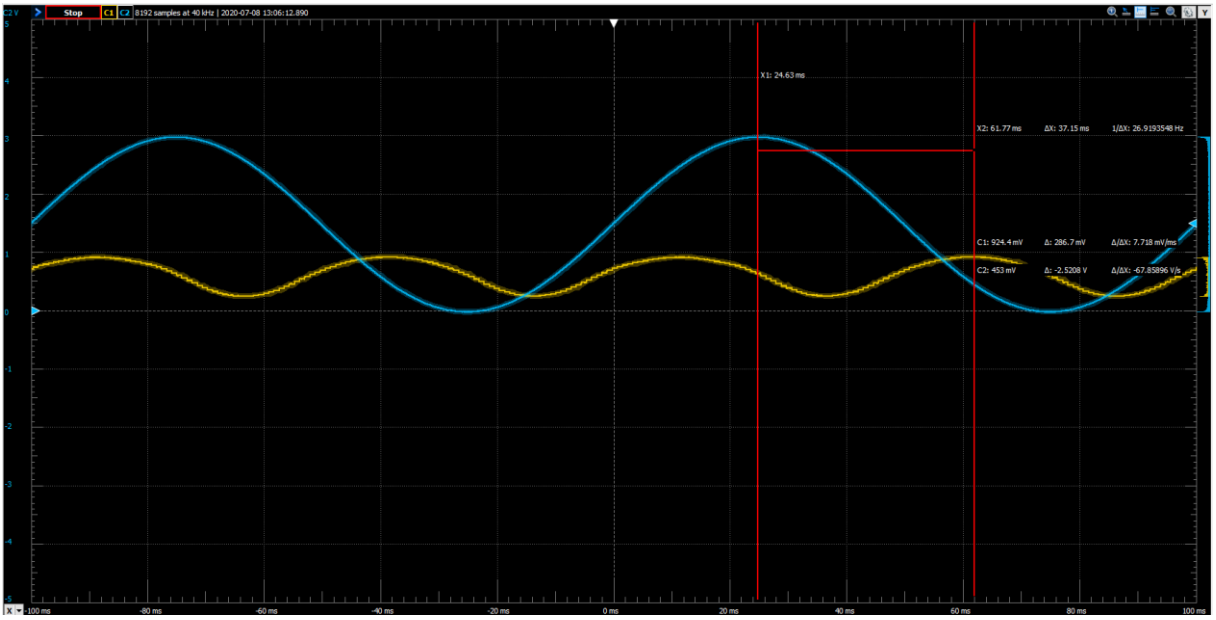


Figura 46: *Medición 2 de desfase temporal entre señal original [azul] y señal filtrada [amarillo] en el contexto de DSP tradicional* (Fuente Propia)

El retardo de propagación en ambos casos es de 35.71ms y de 37.15ms dando un promedio de 36.43ms, que es el tiempo que le toma al algoritmo filtrar las señales

5.2. Procesamiento 'On-the-Edge'

En la Figura 47 y Figura 48 se muestra que el algoritmo necesita un tiempo mínimo para iniciar el funcionamiento, inicialmente la señal se muestra con más ruido que la señal original y transcurrido un tiempo se aprecia que los efectos de filtrado son más notorios.

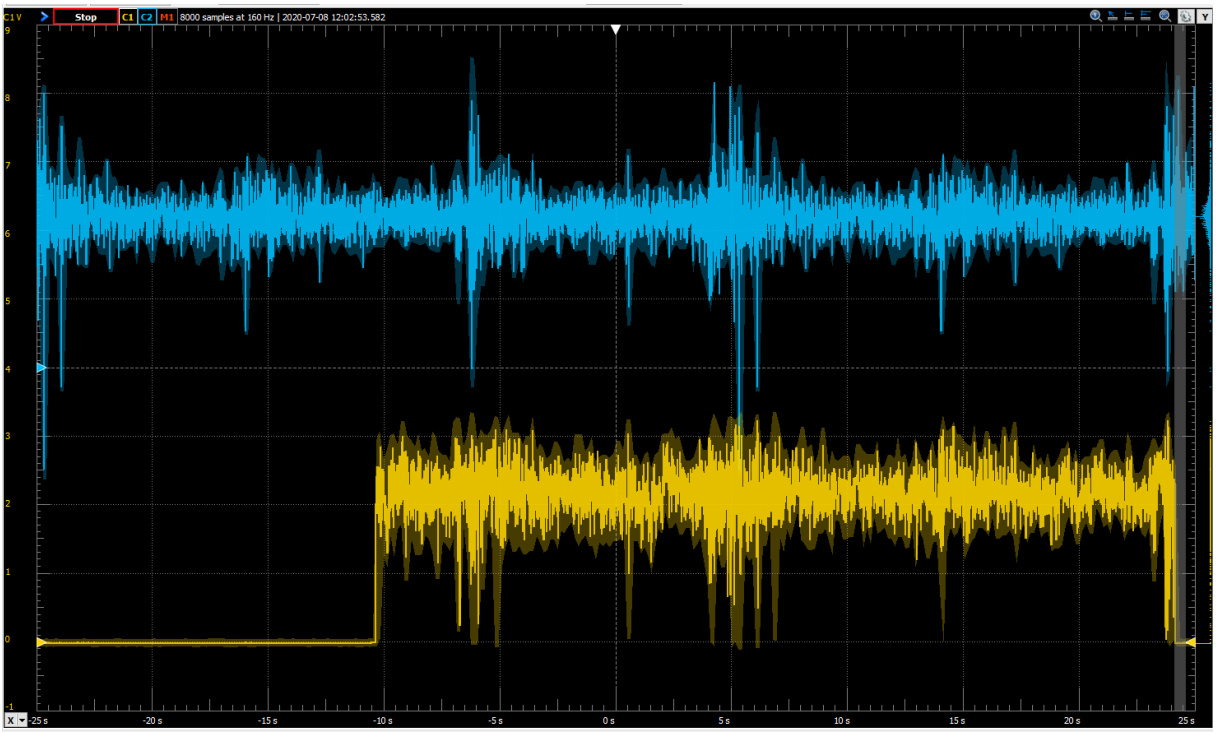


Figura 47: *Inicio de adquisición con procesamiento 'on-the-edge', señal original [azul] y señal filtrada [amarillo]* (Fuente Propia)

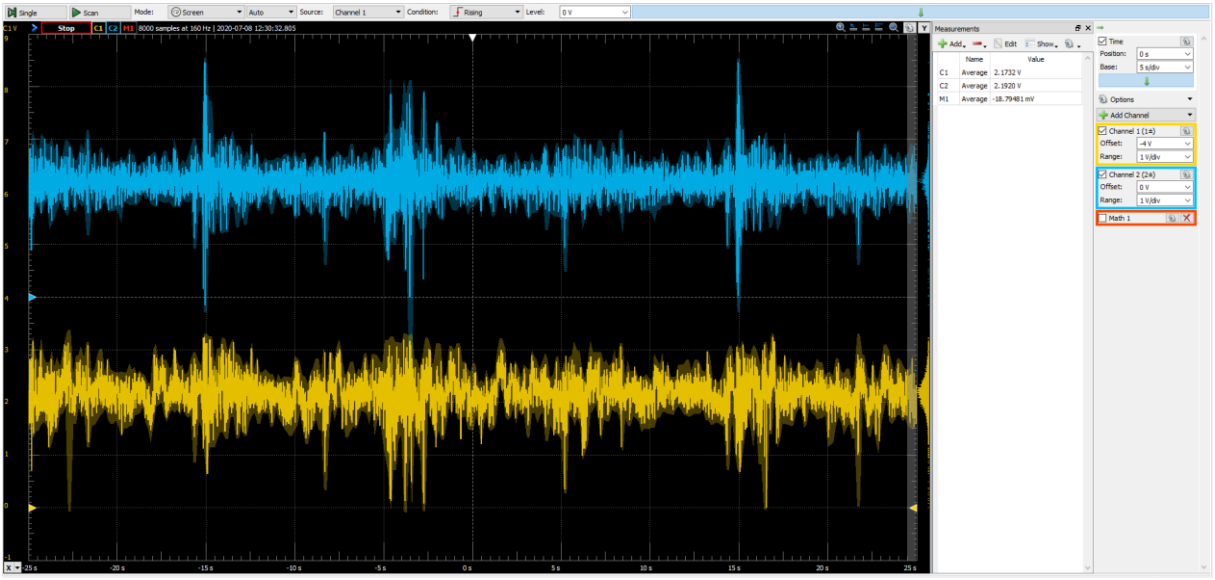


Figura 48: *Funcionamiento normal con procesamiento 'on-the-edge', señal original [azul] y señal filtrada [amarillo]* (Fuente Propia)

El análisis visual de las señales muestra que mantienen una gran relación de aspecto, sin embargo, el cambio no es radical dada la fuente de ruido, la media de la señal original es de 2.192V y la señal filtrada es de 2.173V lo que muestra que mantienen sus características en gran manera

En cuanto al análisis de espectros, se pueden observar en la Figura 49 y Figura 50

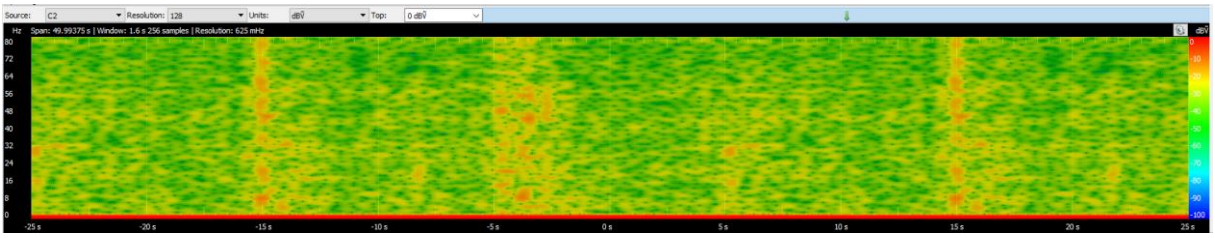


Figura 49: *Espectrograma de señal original en contexto de procesamiento 'on-the-edge'* (Fuente Propia)

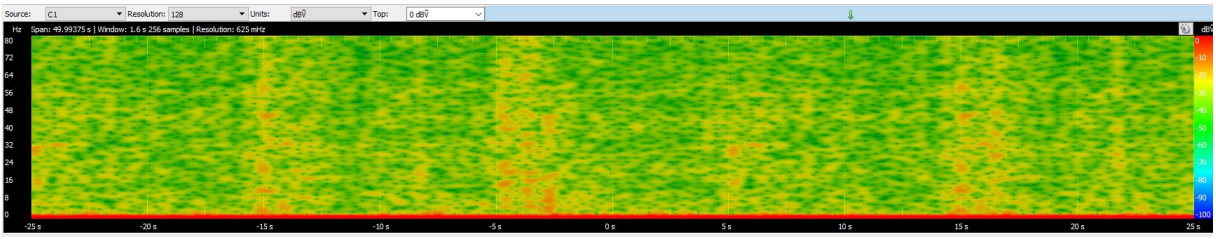


Figura 50: *Espectrograma de señal filtrada en contexto de procesamiento 'on-the-edge'* (Fuente Propia)

En la comparativa de espectrogramas no se aprecia gran cambio, excepto el efecto de difuminado en los picos impulsivos, regularizando la señal

El análisis visual de AC RMS mostrado en la Figura 51 muestra que ambas señales mantienen en gran manera la correlación entre sus señales, y que busca adaptarse a la fuente de ruido para minimizarla, la señal filtrada en morado varía conforme el algoritmo se adapta a la fuente de ruido.

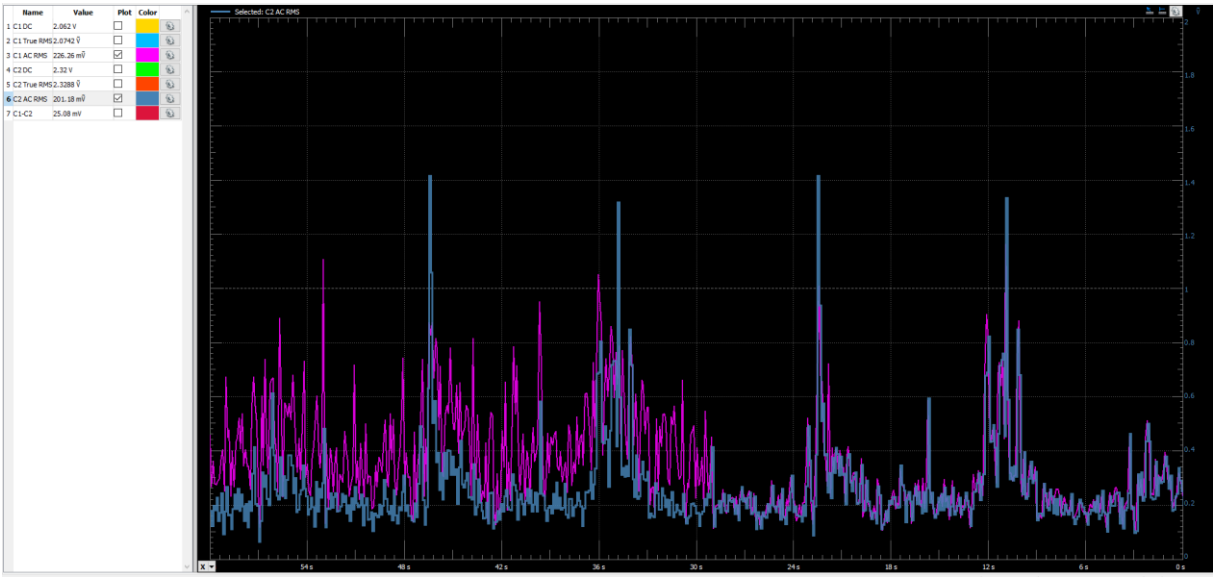


Figura 51: *AC RMS de las señales original [azul] y señal filtrada [morado] en contexto de procesamiento 'on-the-edge'* (Fuente Propia)

En el análisis comparativo de la diferencia de la señal original con la señal filtrada que se presenta en la Figura 52 con su espectrograma en la Figura 53 permite identificar la señal de ruido que se está eliminando, con los artefactos impulsivos en tiempo y frecuencia que le restan calidad a la señal. La media de dicha señal se encuentra en -18.79mV.

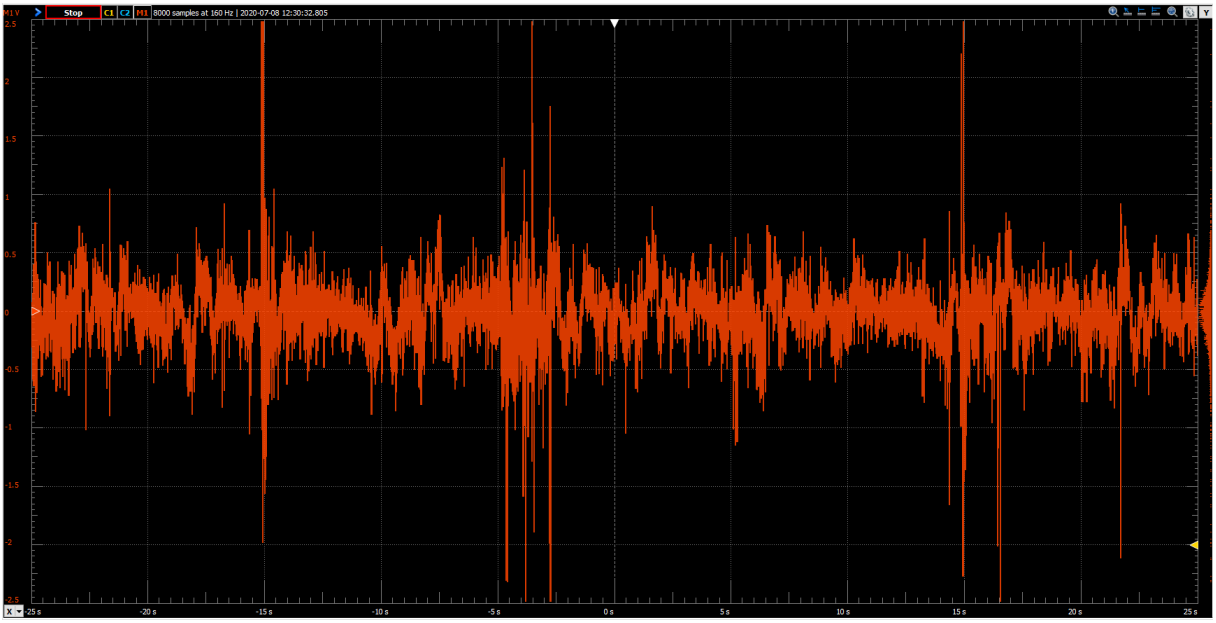


Figura 52: *Resta entre señal original y filtrada en el contexto de procesamiento 'on-the-edge'* (Fuente Propia)

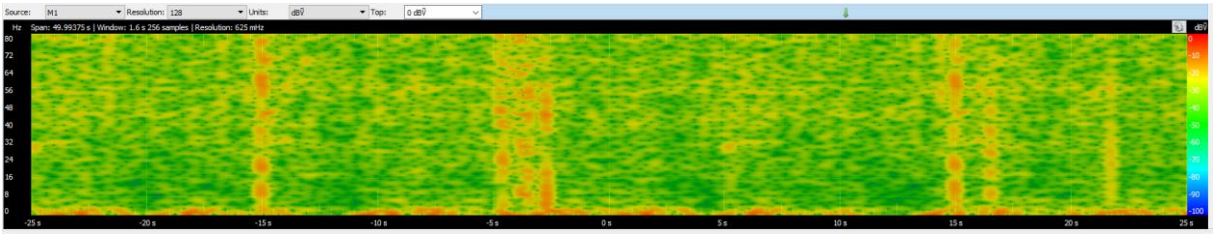


Figura 53: *Espectrograma de la señal resta entre señal original y filtrada en el contexto de procesamiento 'on-the-edge'* (Fuente Propia)

Para el análisis de retardo temporal de las señales se analiza la medición en dos puntos diferentes, presentados en la Figura 54 y en la Figura 45

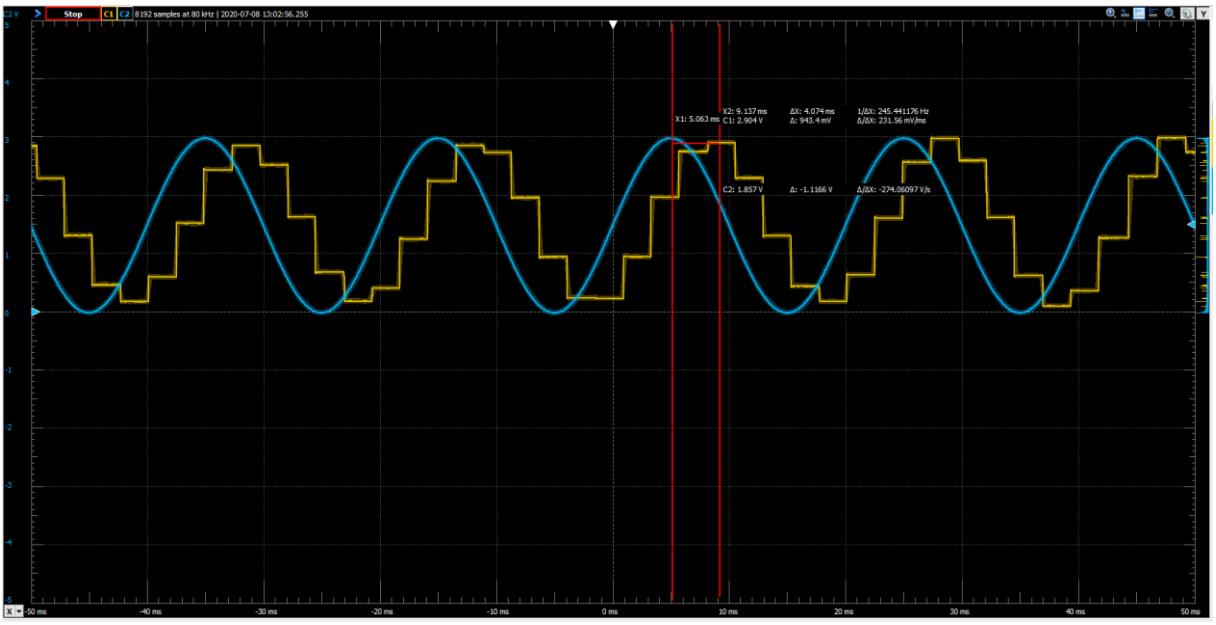


Figura 54: *Medición 1 de desfase temporal entre señal original [azul] y señal filtrada [amarillo] en el contexto de procesamiento 'on-the-edge'* (Fuente Propia)

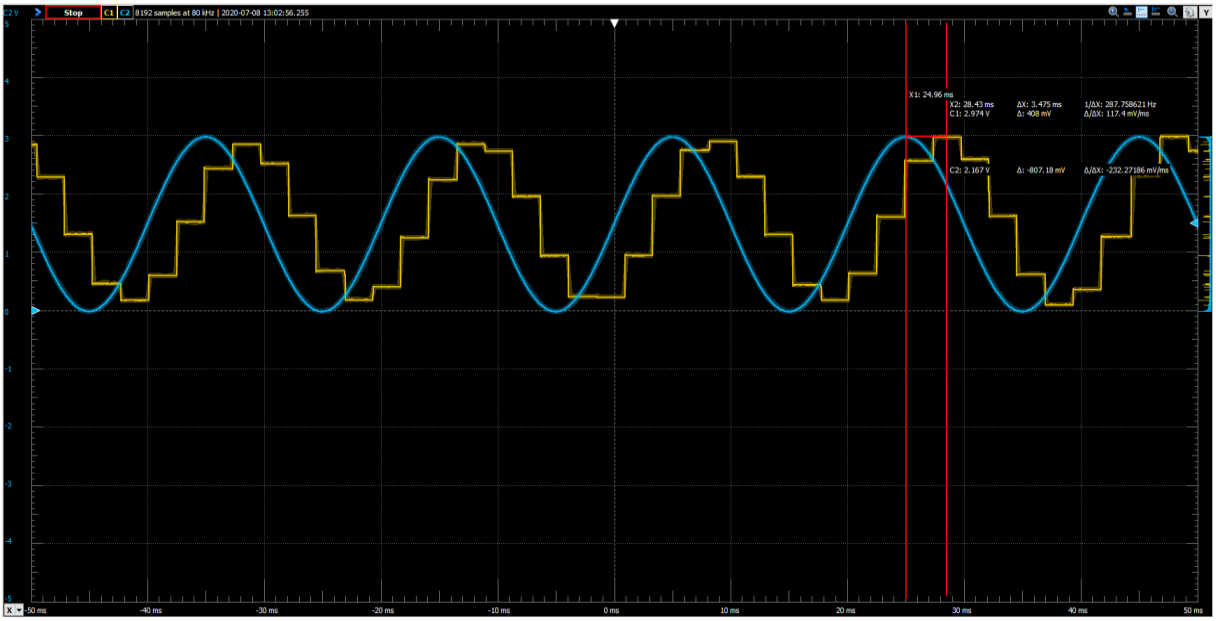


Figura 55: *Medición 2 de desfase temporal entre señal original [azul] y señal filtrada [amarillo] en el contexto de procesamiento 'on-the-edge'* (Fuente Propia)

El retardo de propagación en ambos casos es de 4.074ms y de 3.47ms dando un promedio de 3.822ms, que es el tiempo que le toma al algoritmo filtrar las señales

5.3. Procesamiento en computación distribuida

En la Figura 56 y Figura 57 se muestra que el algoritmo necesita un tiempo mínimo para iniciar el funcionamiento, inicialmente la señal se muestra con ruido considerable y donde casi no se puede identificar a la señal original y transcurrido un tiempo se aprecia que los efectos de filtrado son más notorios.

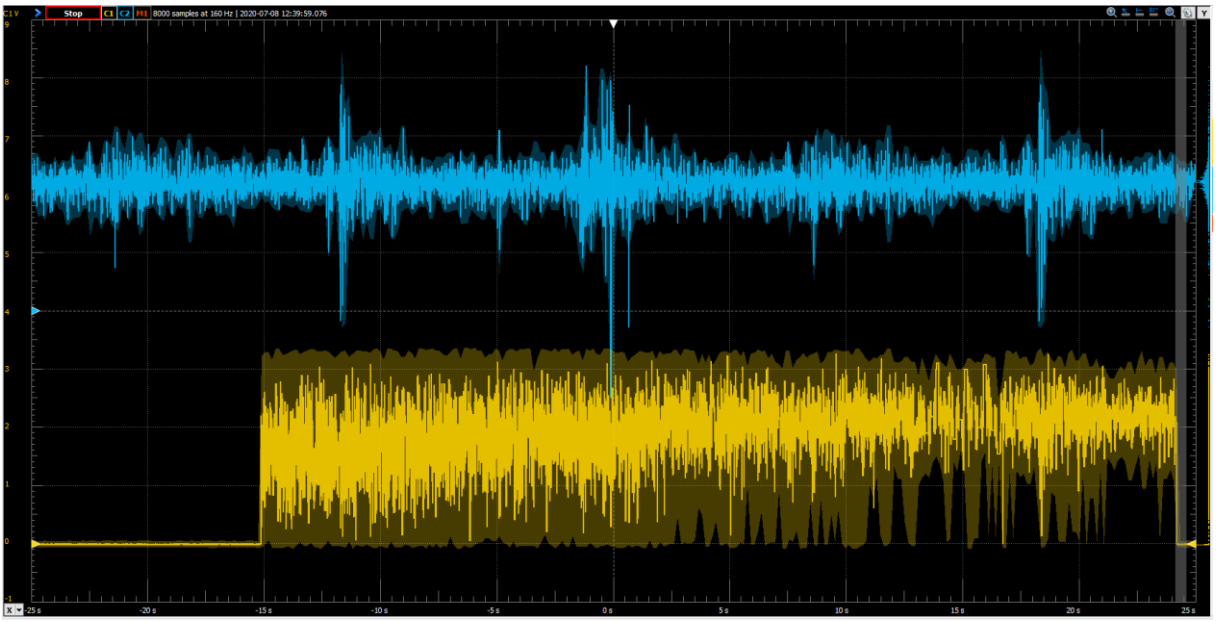


Figura 56: *Inicio de adquisición con sistema de computación distribuida, señal original [azul] y señal filtrada [amarillo]* (Fuente Propia)

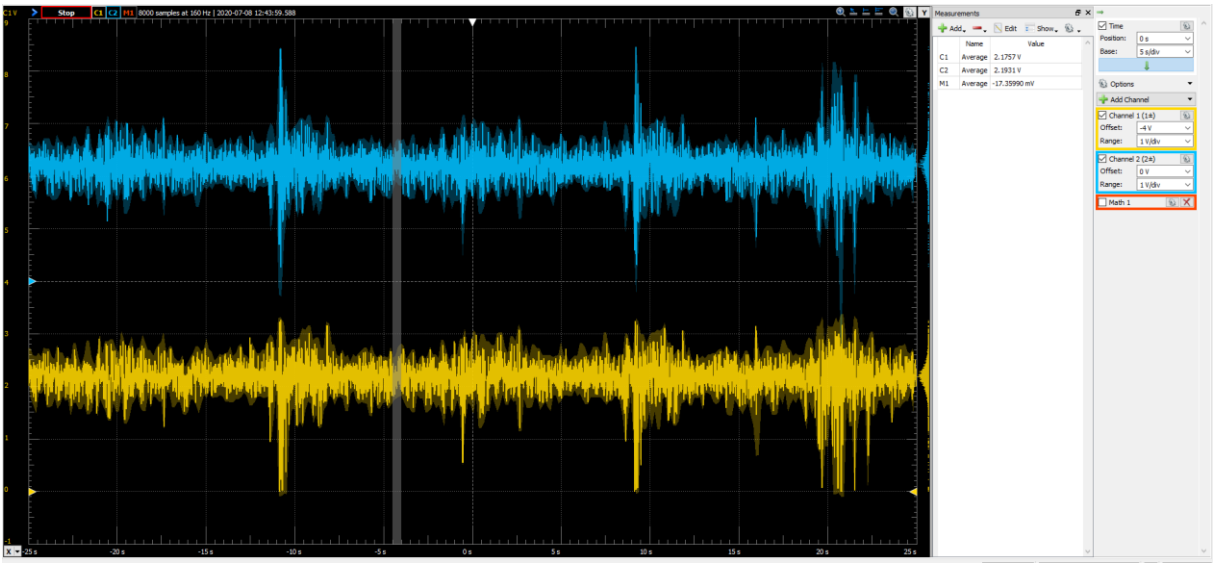


Figura 57: *Funcionamiento normal de sistema de computación distribuida, señal original [azul] y señal filtrada [amarillo]* (Fuente Propia)

El análisis visual de las señales muestra que mantienen una gran relación de aspecto, sin embargo, el cambio no es radical dada la fuente de ruido, la media de la señal original es de 2.193V y la señal filtrada es de 2.175V lo que muestra que mantienen sus características en gran manera

En cuanto al análisis de espectros, se pueden observar en la Figura 58 y Figura 59

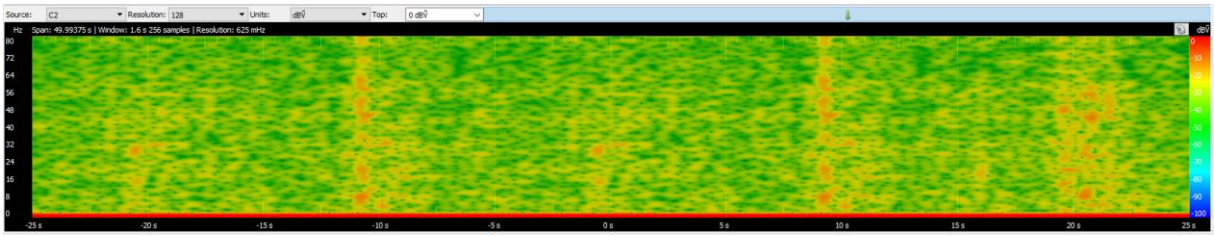


Figura 58: *Espectrograma de señal original en contexto de sistema de cómputo distribuido* (Fuente Propia)

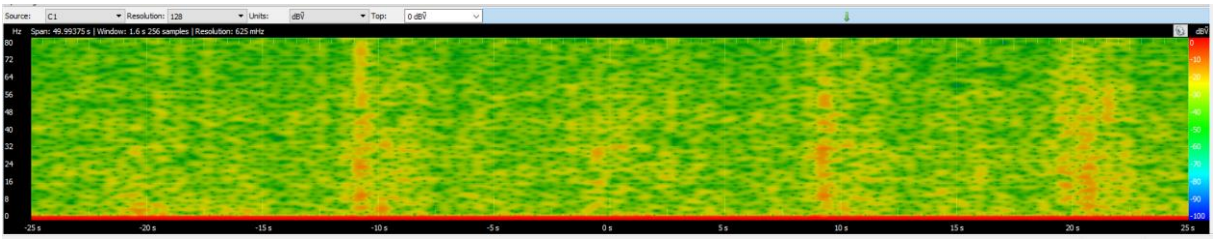


Figura 59: *Espectrograma de señal filtrada en contexto de sistema de cómputo distribuido* (Fuente Propia)

En la comparativa de espectrogramas no se aprecia gran cambio, excepto el efecto de difuminado en los picos impulsivos, regularizando la señal y la concentración de las componentes frecuenciales en las bandas bajas.

El análisis visual de AC RMS mostrado en la Figura 60 muestra que ambas señales mantienen en gran manera la correlación entre sus señales, y que busca adaptarse a la fuente de ruido para minimizarla, la señal filtrada en morado varía conforme el algoritmo se adapta a la fuente de ruido.

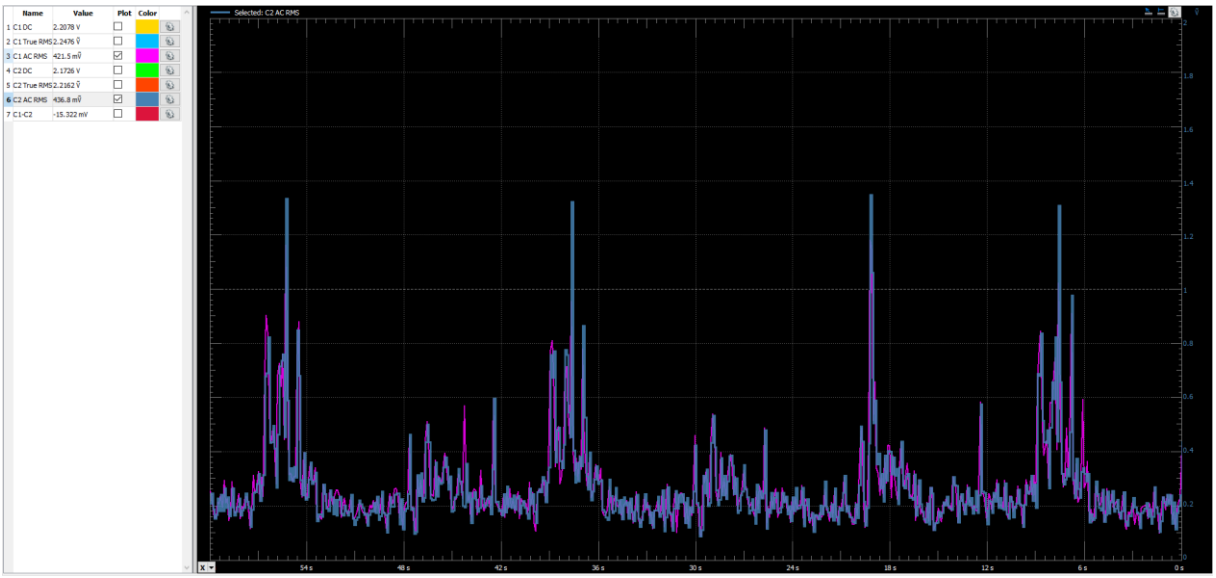


Figura 60: *AC RMS de las señales original [azul] y señal filtrada [morado] en contexto de sistema de cómputo distribuido* (Fuente Propia)

En el análisis comparativo de la diferencia de la señal original con la señal filtrada que se presenta en la Figura 61 con su espectrograma en la Figura 62 permite identificar la señal de ruido que se está eliminando, con los artefactos impulsivos en tiempo y frecuencia que le restan calidad a la señal. La media de dicha señal se encuentra en -17.359mV.

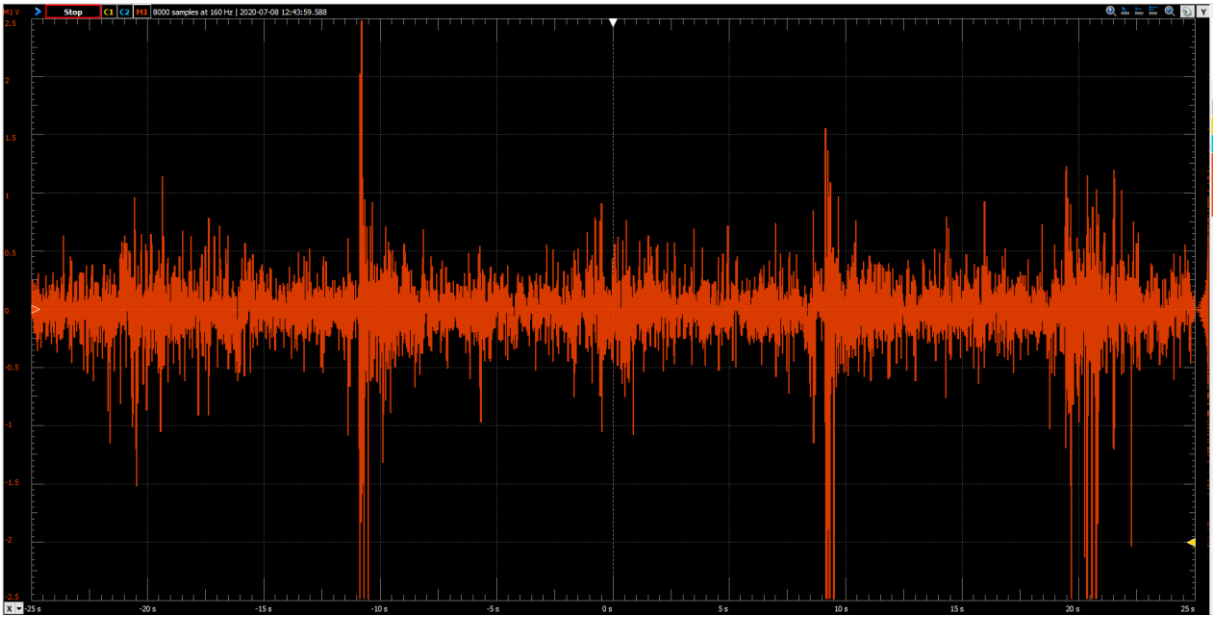


Figura 61: *Resta entre señal original y filtrada en el contexto de sistema de cómputo distribuido* (Fuente Propia)

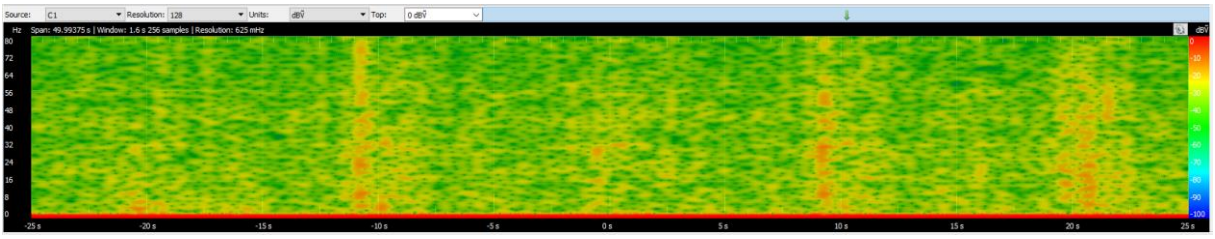


Figura 62: *Espectrograma de la señal resta entre señal original y filtrada en el contexto de sistema de cómputo distribuido* (Fuente Propia)

Para el análisis de retardo temporal de las señales se analiza la medición en dos puntos diferentes, presentados en la Figura 63 y en la Figura 64

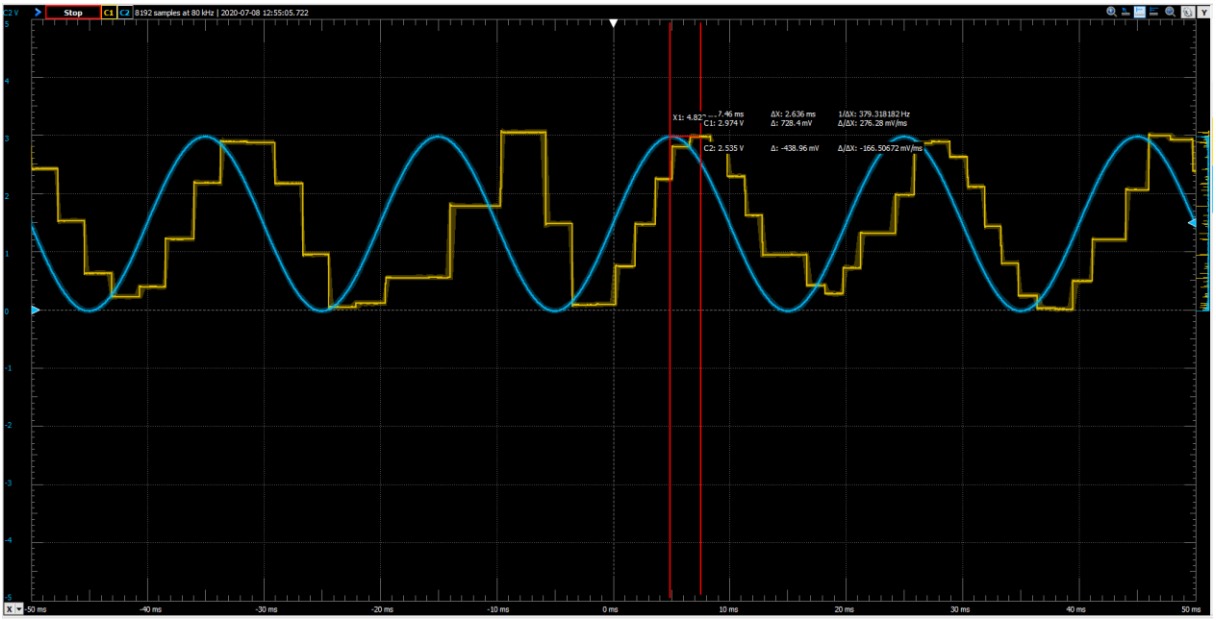


Figura 63: *Medición 1 de desfase temporal entre señal original [azul] y señal filtrada [amarillo] en el contexto de sistema de cómputo distribuido* (Fuente Propia)

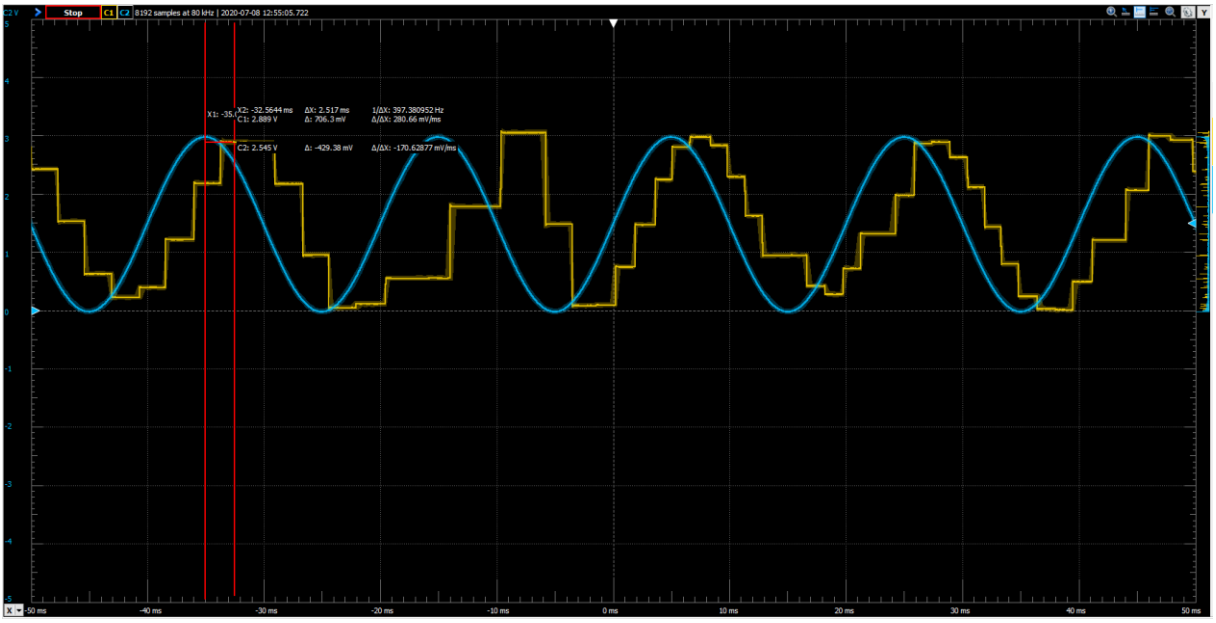


Figura 64: *Medición 2 de desfase temporal entre señal original [azul] y señal filtrada [amarillo] en el contexto de sistema de cómputo distribuido* (Fuente Propia)

El retardo de propagación en ambos casos es de 2.63ms y de 2.517ms dando un promedio de 2.5735ms, que es el tiempo que le toma al algoritmo filtrar las señales

Los resultados se presentan en la Tabla 1 como un perfil de Harris para la evaluación objetiva de las tres metodologías de procesamiento

Tabla 1: *Perfil de Harris para la evaluación de propuestas de procesamiento de señales sEMG* (Fuente Propia)

	DSP Tradicional				**Procesamiento 'on-the-edge'**				**Procesamiento distribuido**			
	- -	-	+	++	- -	-	+	++	- -	-	+	++
Tiempo de procesamiento												
Convergencia del sistema												
Uso de recursos físicos												
Calidad de filtrado												
Complejidad del código o algoritmo ejecutado												
Estabilidad de la ejecución												

La metodología de procesamiento más balanceada dentro del perfil de Harris es la de procesamiento 'on-the-edge' ya que puntúa en la mayoría de las categorías como positivo o fuertemente positivo

6. Discusión

Los resultados y mediciones presentados en los apartados anteriores muestran que, si bien el procesamiento distribuido es el más rápido, es más inestable, generando pérdidas de información cuando se saturan los buffer de los socket de comunicación, además necesitar más equipos de hardware, debido a la implementación de un sistema de comunicación inalámbrica basada en wifi. Se necesita un intermediario en la estructura de red que puede estar conectado a internet, lo que afecta la seguridad de los datos transmitidos y un sistema central de cómputo de buenas características para que pueda llevar a cabo todos los procesos de cálculo y comunicación en un tiempo adecuado.

En cuanto al sistema de DSP tradicional, es el más utilizado hasta la actualidad por su relativa simplicidad de implementación y que no necesita que se modele o se adquiera el ruido de la señal que se va a eliminar, permitiendo que el sistema de procesamiento actúe por sí mismo. Además, los algoritmos presentados en este trabajo de fin de máster pueden ser optimizados utilizando lenguaje C o incluso construyendo módulos de hardware en HDL que puedan ser

implementados en la parte de FPGA del sistema embebido, acelerando radicalmente la velocidad de cómputo del sistema. En este caso como se evalúa los tres sistemas en un mismo marco imparcial, resulta ser el más lento en procesamiento tarda 10 veces más.

En el sistema de procesamiento ‘on-the-edge’, si bien el retardo de procesamiento es de aproximadamente un 70 %, sigue siendo mucho más rápido que el sistema de procesamiento de DSP tradicional, lo que implica un adecuado uso de recursos de hardware y software con la gran ventaja de la adaptabilidad a las fuentes de ruido que puede sufrir una señal EMG de superficie, las cuales pueden cambiar en el tiempo en amplitud y naturaleza. En el presente trabajo no se puede observar la capacidad completa de estos sistemas de filtrado adaptativo por la imposibilidad logística de hacer adquisiciones de señal y ruido sobre sujetos de prueba, lo que limita y sesga los resultados expuestos. Al igual que en el sistema de DSP tradicional, los algoritmos implementados en Python pueden ser optimizados en C, o directamente en HDL para su ejecución en hardware en celdas de FPGA, lo que permitiría la paralelización del cálculo y la escalabilidad del procesamiento sin comprometer el tiempo de procesamiento de estos.

7. Conclusiones y trabajo futuro

7.1. Conclusiones

El procesamiento y, en este caso, el filtrado de señales EMG son un procedimiento de crucial importancia para la utilización de estas señales en procesos de control de dispositivos protésicos o diagnóstico médico. Tradicionalmente se han utilizado estructuras rígidas de filtrado basadas en DSP, y posteriormente, para procesos más complejos se envía toda la información que será procesada a una unidad de cálculo discreta con la que, por lo general, se comunican de manera inalámbrica. Por los avances tecnológicos en la miniaturización de los sistemas de procesamiento y su aumento en capacidad de cálculo, se considera apropiado abordar una metodología de procesamiento integral 'on-the-edge'. Lo anterior implica que los algoritmos de procesamiento inteligentes y toma de decisiones se encuentren implementadas sobre el dispositivo que realiza la adquisición, eliminando los sistemas discretos de cálculo y mejorando sustancialmente el procesamiento que pueden ofrecer las técnicas de DSP tradicionales.

Para tener una evaluación objetiva de los sistemas de procesamiento, se realiza una comparación entre el sistema de DSP tradicional, el sistema de procesamiento discreto y el sistema de procesamiento 'on-the-edge' en un mismo dispositivo, utilizando una misma base de datos reconstruida por un generador de funciones y se evalúan los espectrogramas de las señales, los retardos debidos al procesamiento, y valores como la raíz media cuadrática eliminando componentes de corriente continua. El algoritmo inteligente de filtrado es un filtro adaptativo RLS, al cual como fuente de ruido se le introduce una señal de ruido blanco simulada, ya que la simulación de una línea base de EMG o la adquisición de esta es imposible en el contexto en el que se desarrolla el presente trabajo de fin de máster.

En la revisión de literatura no se ha encontrado un aporte sobre la implementación de filtros adaptativos en un dispositivo 'on-the-edge', la mayoría utilizan sistemas distribuidos de cálculo en MATLAB. En el presente trabajo se presenta una implementación a tiempo real de algoritmos de filtrado adaptativo, la mayor contribución del trabajo de investigación.

Con la disponibilidad tecnológica actual, es adecuado e incluso recomendable realizar el procesamiento inteligente sobre el dispositivo, incluso si es necesario un sistema de procesamiento distribuido, ya que disminuirá la carga computacional al recibir información útil y no únicamente datos en crudo. Esto permitirá construir arquitecturas de procesamiento y uso de las señales más complejas ofreciendo mejores resultados a los usuarios finales.

Por lo tanto, respondiendo a las preguntas de investigación, un procesamiento inteligente 'on-the-edge' de señales electromiográficas es más efectivo que el procesamiento tradicional en hardware. Lo anterior debido a que el tiempo de retardo entre las señales es menor, y si se caracteriza adecuadamente la señal de ruido que interfiere con la señal deseada, o se la adquiere con sensores secundarios, puede ofrecer información muy pertinente sin cambios considerables en la naturaleza frecuencial y temporal de las señales electromiográficas. Por otro lado, considerando el estado actual de la tecnología, es justificada la aplicación de técnicas de inteligencia artificial 'on-the-edge' ya que disminuyen la carga computacional que sufren otros dispositivos, permiten generar arquitecturas con mayor escalabilidad y se puede considerar pertinente la autonomía de procesamiento entre cada uno de los nodos inteligentes que pueden conformar un sistema de adquisición en incluso es posible la autonomía total de cada sensor y que no requiera de un sistema auxiliar de cómputo para realizar el proceso de toma de decisiones o acondicionamiento secundario de las señales.

7.2. Líneas de trabajo futuro

Para futuros trabajos investigativos, se bajará de nivel lo más posible para la implementación de los algoritmos, en el caso de DSP tradicional y de procesamiento 'on-the-edge' los algoritmos deberán ser implementados en lenguaje HDL en la lógica programable de un FPGA, lo que permitirá acelerar por hardware las funciones y crear sistemas realmente paralelos y multicanal de procesamiento, ya que no se limitarán a la lógica secuencial de la programación de software. En cuanto al procesamiento distribuido, se optimizará el procesamiento realizando un código en C/C++ que permita cálculos incluso más rápidos que los presentados en el presente trabajo.

Igualmente es necesario comparar las tecnologías disponibles para procesamiento en el dispositivo como microcontroladores de 16 y 32 bits, tarjetas gráficas de propósito general, FPGA y SoC, en los cuales se implementará el algoritmo de filtrado adaptativo, y se analizará la relación costo-beneficio entre los mismos, el consumo de potencia y la escalabilidad de la arquitectura para implementar otros sistemas necesarios para el funcionamiento de un dispositivo protésico, que es el fin último de este trabajo de investigación.

8. Bibliografía

Analog Devices. (2011). *AD7474 Datasheet*. Retrieved from https://www.analog.com/media/cn/technical-documentation/evaluation-documentation/AD7476A_7477A_7478A.pdf?_ga=2.190611627.1204819131.1591302926-358339790.1567696285

Cerutti, S., & Marchesi, C. (2011). *Advanced Methods of Biomedical Signal Processing*. (S. Cerutti & C. Marchesi, Eds.), *Advanced Methods of Biomedical Signal Processing*. IEEE Press. https://doi.org/10.1002/9781118007747

Corcoran, P. M. (1998). Mapping home-network appliances to tcp/ip sockets using a three-tiered home gateway architecture. *IEEE Transactions on Consumer Electronics*, *44*(3), 729–736. https://doi.org/10.1109/30.713188

DIGILENT. (2016a). *PmodAD1™ Reference Manual*. Retrieved from https://reference.digilentinc.com/_media/reference/pmod/pmodad1/pmodad1_rm.pdfcom

DIGILENT. (2016b). *PmodDA2™ Reference Manual*. Retrieved from https://reference.digilentinc.com/_media/reference/pmod/pmodda2/pmodda2_rm.pdf

Diniz, P. S. R. (2008). *Adaptive Filtering*. *Adaptive Filtering*. Boston, MA: Springer US. https://doi.org/10.1007/978-0-387-68606-6

Fang, Y., Zhou, D., Li, K., & Liu, H. (2018). ISRMyo-I: A database for sEMG-based hand gesture recognition | IEEE DataPort. https://doi.org/10.21227/H26Q26

Gerardo, J., C., J., & Velazquez, J. (2011). Applications of Adaptive Filtering. In *Adaptive Filtering Applications* (Vol. i, p. 13). InTech. https://doi.org/10.5772/16873

Ghalyan, I. F., Abouelenin, Z. M., & Kapila, V. (2019). Gaussian Filtering of EMG Signals for Improved Hand Gesture Classification. *2018 IEEE Signal Processing in Medicine and Biology Symposium, SPMB 2018 - Proceedings*, 1–6. https://doi.org/10.1109/SPMB.2018.8615596

Hägg, G. M., Melin, B., & Kadefors, R. (2005). Applications in Ergonomics. In R. Merletti & P. Parker (Eds.), *Electromyography* (pp. 343–363). IEEE Press. https://doi.org/10.1002/0471678384.ch13

Hsueh, Y. H., Yin, C., & Chen, Y. H. (2015). Hardware System for Real-Time EMG Signal Acquisition and Separation Processing during Electrical Stimulation. *Journal of Medical Systems*, *39*(9). https://doi.org/10.1007/s10916-015-0267-6

Jamal, M. Z., Lee, D. H., & Hyun, D. J. (2019). Real Time Adaptive Filter based EMG Signal Processing and Instrumentation Scheme for Robust Signal Acquisition Using Dry EMG Electrodes. *2019 16th International Conference on Ubiquitous Robots, UR 2019*, 683–688. https://doi.org/10.1109/URAI.2019.8768662

Kasaeyan Naeini, E., Shahhosseini, S., Subramanian, A., Yin, T., Rahmani, A. M., & Dutt, N. (2019). An Edge-Assisted and Smart System for Real-Time Pain Monitoring. *Proceedings - 4th IEEE/ACM Conference on Connected Health: Applications, Systems and Engineering Technologies, CHASE 2019*, 47–52. https://doi.org/10.1109/CHASE48038.2019.00023

Kimura, J. (2013). Electrodiagnosis in Diseases of Nerve and Muscle, 1177. https://doi.org/10.1097/00004691-199010000-00011

Konrad, P. (2005). *The abc of emg*. *A practical introduction to kinesiological electromyography*. https://doi.org/10.1016/j.jacc.2008.05.066

Kosko, B., & Haykin, S. S. (Eds.). (2001). *Intelligent Signal Processing*. *Signal Processing* (Vol. 81). IEEE Press. https://doi.org/10.1016/S0165-1684(01)00152-9

Lee, Y. L., Tsung, P. K., & Wu, M. (2018). Techology trend of edge AI. In *2018 International Symposium on VLSI Design, Automation and Test, VLSI-DAT 2018* (pp. 1–2). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/VLSI-DAT.2018.8373244

Li, C., Li, G., Jiang, G., Chen, D., & Liu, H. (2018). Surface EMG data aggregation processing for intelligent prosthetic action recognition. *Neural Computing and Applications*, *0123456789*. https://doi.org/10.1007/s00521-018-3909-z

Limem, M., Hamdi, M. A., & Maaref, M. A. (2016). Denoising uterine EMG signals using LMS and RLS adaptive algorithms. In *2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)* (pp. 273–276). IEEE. https://doi.org/10.1109/ATSIP.2016.7523113

Maata, R. L., Cordova, R., Sudramurthy, B., & Halibas, A. (2018). Design and Implementation of Client-Server Based Application Using Socket Programming in a Distributed

Computing Environment. In *2017 IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2017*. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICCIC.2017.8524573

Mahmoud, W. H., & Zhang, N. (2013). Software/hardware implementation of an adaptive noise cancellation system. *ASEE Annual Conference and Exposition, Conference Proceedings*.

Márquez-Figueroa, S., Shmaliy, Y. S., & Ibarra-Manzano, O. (2020). Optimal extraction of EMG signal envelope and artifacts removal assuming colored measurement noise. *Biomedical Signal Processing and Control*, *57*, 101679. https://doi.org/10.1016/j.bspc.2019.101679

Menegaldo, L. L. (2017). Real-time muscle state estimation from EMG signals during isometric contractions using Kalman filters. *Biological Cybernetics*, *111*(5–6), 335–346. https://doi.org/10.1007/s00422-017-0724-z

Merletti, R., & Farina, D. (Eds.). (2016). *Surface Electromyography*. IEEE Press.

Mugdha, A. C., Rawnaque, F. S., & Ahmed, M. U. (2015). A study of recursive least squares (RLS) adaptive filter algorithm in noise removal from ECG signals. In *2015 International Conference on Informatics, Electronics & Vision (ICIEV)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICIEV.2015.7333998

Okoniewski, P., Kocon, S., & Piskorowski, J. (2018). Linear Time-Varying Multi-Notch FIR Filter for Fast EMG Measurements. *2018 23rd International Conference on Methods and Models in Automation and Robotics, MMAR 2018*, 634–637. https://doi.org/10.1109/MMAR.2018.8486122

Onay, F., & Mert, A. (2020). Phasor represented EMG feature extraction against varying contraction level of prosthetic control. *Biomedical Signal Processing and Control*, *59*, 101881. https://doi.org/10.1016/j.bspc.2020.101881

Oppenheim, A., Schafer, R., & Buck, J. (1998). *Discrete-Time Signal Processing* (2nd ed.). New Jersey: Prentice Hall Press. Retrieved from https://www.pearson.com/us/higher-education/program/Oppenheim-Discrete-Time-Signal-Processing-3rd-Edition/PGM212808.html

Parker, P. A., Englehart, K. B., & Hudgins, B. S. (2005). Control of Powered Upper Limb Prostheses. In R. Merletti & P. A. Parker (Eds.), *Electromyography* (pp. 453–475). IEEE Press. https://doi.org/10.1002/0471678384.ch18

Prasad, D. R., Naganjaneyulu, P. V, & Prasad, K. S. (2018). Modi fi ed LEACH Protocols in Wireless Sensor Networks — A Review. *IEEE ACcess*, 681–688. https://doi.org/10.1007/978-981-10-4280-5

Rangayyan, R. M. (2015). *Biomedical Signal Analysis*. *Medical Devices and Systems* (2nd ed.). IEEE Press.

Restrepo-Agudelo, S., Roldan-Vasco, S., Ramirez-Arbelaez, L., Cadavid-Arboleda, S., Perez-Giraldo, E., & Orozco-Duque, A. (2017). Improving surface EMG burst detection in infrahyoid muscles during swallowing using digital filters and discrete wavelet analysis. *Journal of Electromyography and Kinesiology*, *35*, 1–8. https://doi.org/10.1016/j.jelekin.2017.05.001

Roland, T., Amsuess, S., Russold, M. F., & Baumgartner, W. (2019). Ultra-low-power digital filtering for insulated EMG sensing. *Sensors (Switzerland)*, *19*(4), 1–24. https://doi.org/10.3390/s19040959

Rozaqi, L., Nugroho, A., Sanjaya, K. H., & Ivonita Simbolon, A. (2019). Design of Analog and Digital Filter of Electromyography. *Proceeding - 2019 International Conference on Sustainable Energy Engineering and Application: Innovative Technology Toward Energy Resilience, ICSEEA 2019*, 186–192. https://doi.org/10.1109/ICSEEA47812.2019.8938645

Sahu, A. K., & Sahu, A. K. (2018). A Review on Different Filter Design Techniques and Topologies for Bio-potential Signal Acquisition Systems. *Proceedings of the 3rd International Conference on Communication and Electronics Systems, ICCES 2018*, (Icces), 934–937. https://doi.org/10.1109/CESYS.2018.8723912

Salamea Palacios, C., & Luna Romero, S. (2011). Calibración automática en filtros adaptativos para el procesamiento de señales EMG. *Revista Iberoamericana de Automatica e Informatica Industrial*, *16*, 1–6.

Singh, P., Bhole, K., & Sharma, A. (2017). Adaptive Filtration Techniques for Impulsive Noise Removal from ECG. In *2017 14th IEEE India Council International Conference (INDICON)* (pp. 1–4). IEEE. https://doi.org/10.1109/INDICON.2017.8488064

Smith, S. W. (1997). *The scientist and engineer's guide to digital signal processing* (2nd ed.). San Diego: California Technical Publishing. Retrieved from http://www.dspguide.com/

Stegeman, D., & Hermens, H. (2007). Standards for surface electromyography: The European

project Surface EMG for non-invasive assessment of muscles (SENIAM). *SENIAM*, 108–112. Retrieved from http://www.med.uni-jena.de/motorik/pdf/stegeman.pdf

Tam, S., Boukadoum, M., Campeau-Lecours, A., & Gosselin, B. (2020). A Fully Embedded Adaptive Real-Time Hand Gesture Classifier Leveraging HD-sEMG and Deep Learning. *IEEE Transactions on Biomedical Circuits and Systems*, *14*(2), 232–243. https://doi.org/10.1109/TBCAS.2019.2955641

Tankisi, H., Burke, D., Cui, L., de Carvalho, M., Kuwabara, S., Nandedkar, S. D., … Fuglsang-Frederiksen, A. (2020). Standards of instrumentation of EMG. *Clinical Neurophysiology*, *131*(1), 243–258. https://doi.org/10.1016/j.clinph.2019.07.025

Texas Instruments. (2005). *DAC121S101, DAC121S101-Q1*. Retrieved from www.ti.com

Toledo-Pérez, D. C., Martínez-Prado, M. A., Rodríguez-Reséndiz, J., Tovar Arriaga, S., & Márquez-Gutiérrez, M. A. (2017). IIR Digital Filter Design Implemented on FPGA for Myoelectric Signals. *2017 13th International Engineering Congress, CONIIN 2017*. https://doi.org/10.1109/CONIIN.2017.7968184

Verma, A. R., Singh, Y., & Gupta, B. (2018). Adaptive filtering method for EMG signal using bounded range artificial bee colony algorithm. *Biomedical Engineering Letters*, *8*(2), 231–238. https://doi.org/10.1007/s13534-017-0056-x

Woods, R., McAllister, J., Lightbody, G., & Yi, Y. (2008). *FPGA-Based Implementation of Signal Processing Systems*. *FPGA-Based Implementation of Signal Processing Systems*. West Sussex: Wiley. https://doi.org/10.1002/9780470713785

Xilinx Inc. (2017). Python productivity for Zynq (Pynq) Documentation. *Data Sheet*, 1–235. Retrieved from http://www.pynq.io/home.html

Yu, T., Akhmadeev, K., Le Carpentier, E., Aoustin, Y., & Farina, D. (2019). On-line recursive decomposition of intramuscular EMG signals using GPU-implemented Bayesian filtering. *IEEE Transactions on Bio-Medical Engineering*. https://doi.org/10.1109/TBME.2019.2948397

Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., & Zhang, J. (2019). Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing. *Proceedings of the IEEE*, *107*(8). https://doi.org/10.1109/JPROC.2019.2918951

Anexos

Anexo I. Código VHDL para lectura del protocolo de señales del convertidor analógico-digital

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity SPI_IN is
    Port (
        clk : in STD_LOGIC;
        rst : in STD_LOGIC;
        cs : out STD_LOGIC;
        spi_clk : out STD_LOGIC;
        miso : in STD_LOGIC;
        data_out_valid : out STD_LOGIC;
        data_out : out STD_LOGIC_VECTOR (15 downto 0)
    );
end SPI_IN;

architecture Behavioral of SPI_IN is
    -- SIGNALS
    signal chip_select : std_logic := '1';
    signal spi_clk_pos_edge : std_logic;
    signal spi_clk_neg_edge : std_logic;
    signal spi_clk_int : std_logic;
    signal data_rx : std_logic_vector (15 downto 0);
    type state_type is (IDLE, CS_DOWN, READ_SPI, CS_UP);
    signal state : state_type;
begin

    cs <= chip_select;
    spi_clk <= spi_clk_int;

    --clock gen
    process(clk)
        variable count_100kHz : integer := 0;
    begin
        if rising_edge(clk) then
            if rst = '1' then
                spi_clk_int <= '0';
                spi_clk_pos_edge <= '0';
                spi_clk_neg_edge <= '0';
                count_100kHz := 0;
            else
                spi_clk_pos_edge <= '0';
                spi_clk_neg_edge <= '0';
                count_100kHz := count_100kHz + 1;
                if count_100kHz = 50 then
                    spi_clk_pos_edge <= '1';
                end if;
                if count_100kHz <= 50 then --100kHz
                    spi_clk_int <= '0';
                elsif count_100kHz < 100 then
                    spi_clk_int <= '1';
                else
                    spi_clk_neg_edge <= '1';
                    count_100kHz := 0;
                end if;
            end if;
```

```
        end if;
    end process;


    process(clk)
        variable  count_bits : integer := 0;
    begin
        if rising_edge(clk) then
            if rst = '1' then
                chip_select <= '1';
                state <= IDLE;
                data_out_valid <= '0';
                data_out <= (others => '0');
                count_bits := 0;
            else
                chip_select <= '1';
                data_out_valid <= '0';
                case state is
                    when IDLE =>
                        state <= CS_DOWN;
                    when CS_DOWN =>
                        chip_select <= '0';
                        state <= READ_SPI;
                    when READ_SPI =>
                        chip_select <= '0';
                        if spi_clk_pos_edge = '1' then
                            data_rx(15 - count_bits) <= miso;
                            count_bits := count_bits + 1;
                            if count_bits = 16 then
                                count_bits := 0;
                                state <= CS_UP;
                            end if;
                        end if;
                    when CS_UP =>
                        chip_select <= '0';
                        if spi_clk_neg_edge = '1' then
                            chip_select <= '1';
                            data_out <= data_rx;
                            data_rx <= (others => '0');
                            data_out_valid <= '1';
                            state <= CS_DOWN;
                        end if;
                    when others =>
                        state <= IDLE;
                end case;
            end if;
        end if;
    end process;

end Behavioral;
```

Anexo II. Código VHDL para escritura del protocolo de señales del convertidor digital-analógico

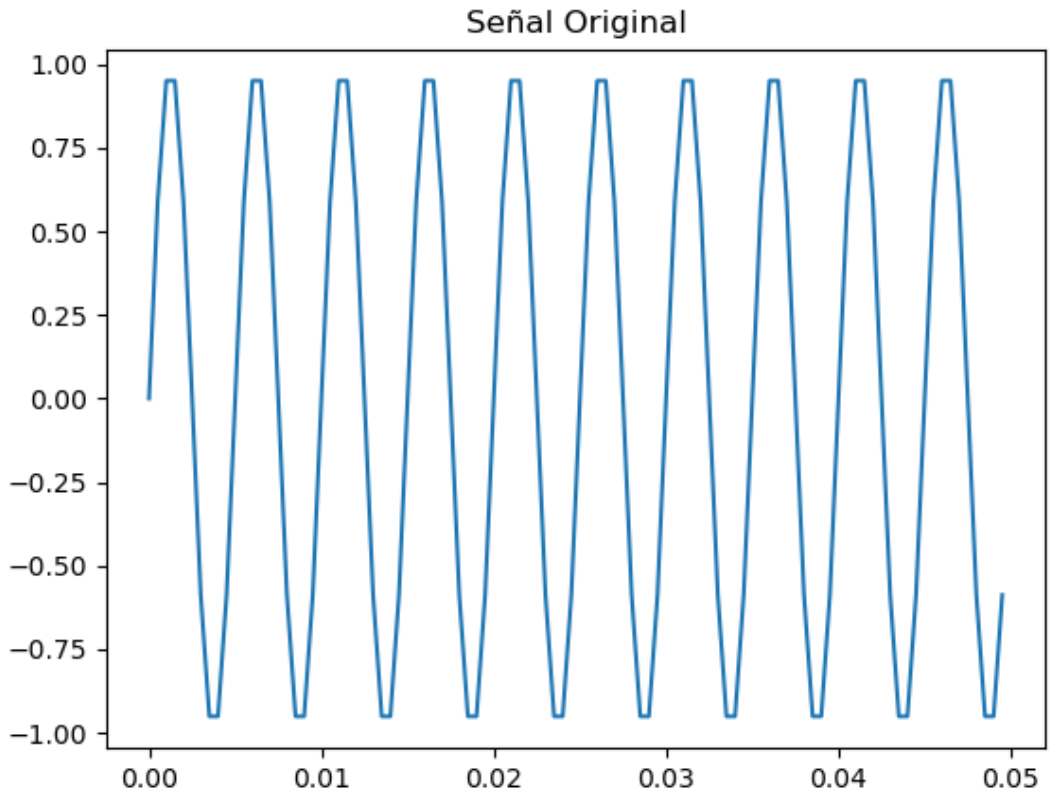
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity SPI_OUT is
    Port (
        clk : in STD_LOGIC;
        rst : in STD_LOGIC;
        cs : out STD_LOGIC;
```

```
        spi_clk : out STD_LOGIC;
        mosi : out STD_LOGIC;
        data_out_valid : out STD_LOGIC;
        data_in : in STD_LOGIC_VECTOR (11 downto 0)
    );
end SPI_OUT;

architecture Behavioral of SPI_OUT is
    -- SIGNALS
    signal chip_select : std_logic := '1';
    signal spi_clk_pos_edge : std_logic;
    signal spi_clk_neg_edge : std_logic;
    signal spi_clk_int : std_logic;
    signal data_tx : std_logic_vector (15 downto 0) := (others => '0');
    type state_type is (IDLE, CS_DOWN, WRITE_SPI, CS_UP);
    signal state : state_type;
begin

    cs <= chip_select;
    spi_clk <= spi_clk_int;

    --clock gen
    process(clk)
        variable count_100kHz : integer := 0;
    begin
        if rising_edge(clk) then
            if rst = '1' then
                spi_clk_int <= '0';
                spi_clk_pos_edge <= '0';
                spi_clk_neg_edge <= '0';
                count_100kHz := 0;
            else
                spi_clk_pos_edge <= '0';
                spi_clk_neg_edge <= '0';
                count_100kHz := count_100kHz + 1;
                if count_100kHz = 50 then
                    spi_clk_pos_edge <= '1';
                end if;
                if count_100kHz <= 50 then --100kHz
                    spi_clk_int <= '0';
                elsif count_100kHz < 100 then
                    spi_clk_int <= '1';
                else
                    spi_clk_neg_edge <= '1';
                    count_100kHz := 0;
                end if;
            end if;
        end if;
    end process;


    process(clk)
        variable  count_bits : integer := 0;
    begin
        if rising_edge(clk) then
            if rst = '1' then
                chip_select <= '1';
                state <= IDLE;
                data_out_valid <= '0';
                mosi <= '0';
                count_bits := 0;
            else
                chip_select <= '1';
                data_out_valid <= '0';
                case state is
                    when IDLE =>
                        state <= CS_DOWN;
                    when CS_DOWN =>
```

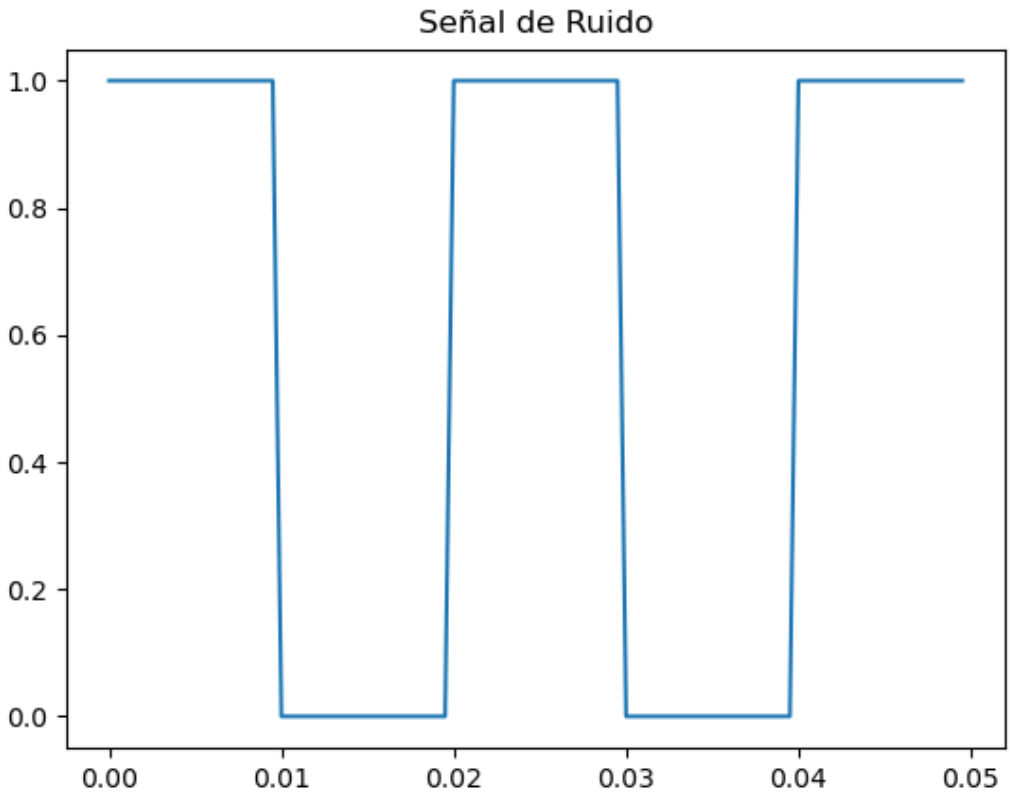
```
                        if spi_clk_pos_edge = '1' then
                            chip_select <= '0';
                            data_tx(11 downto 0) <= data_in;
                            state <= WRITE_SPI;
                        end if;
                    when WRITE_SPI =>
                        chip_select <= '0';
                        if spi_clk_neg_edge = '1' then
                            mosi <= data_tx(15 - count_bits);
                            count_bits := count_bits + 1;
                            if count_bits = 16 then
                                count_bits := 0;
                                state <= CS_UP;
                            end if;
                        end if;
                    when CS_UP =>
                        chip_select <= '0';
                        if spi_clk_pos_edge = '1' then
                            chip_select <= '1';
                            data_tx <= (others => '0');
                            data_out_valid <= '1';
                            state <= CS_DOWN;
                        end if;
                    when others =>
                        state <= IDLE;
                end case;
            end if;
        end if;
    end process;

end Behavioral;
```

Anexo III. Diseño de filtros tradicionales

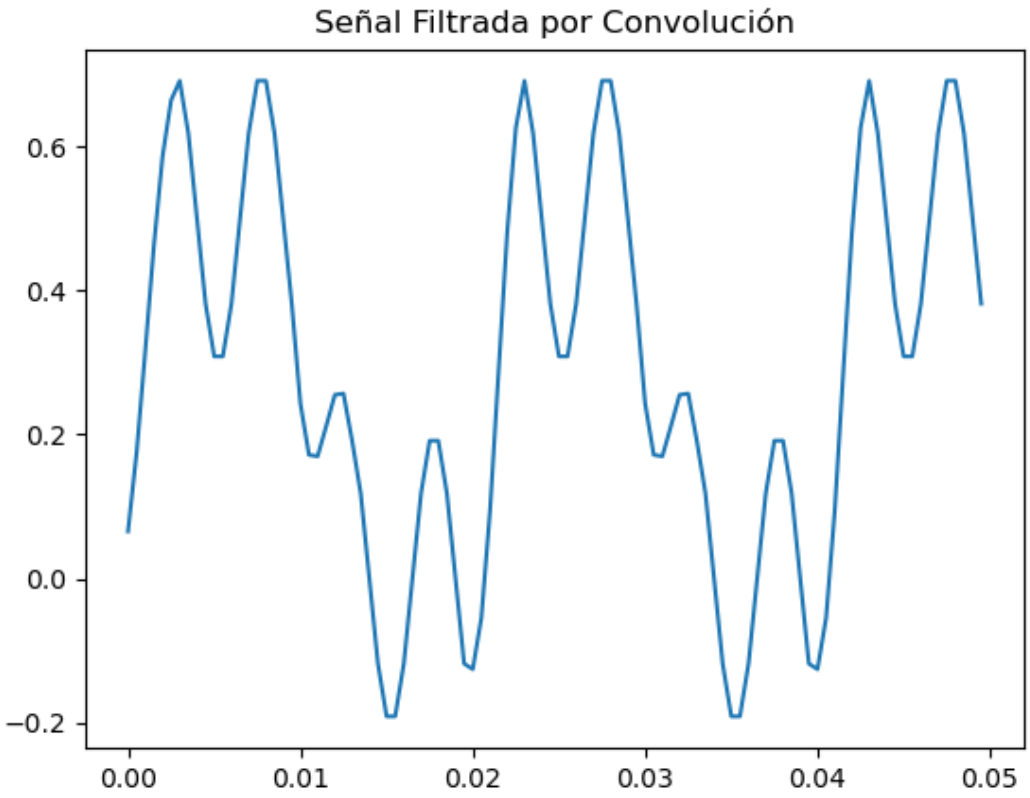
```
import numpy as np
import matplotlib.pyplot as plt

t = np.linspace(0, 1.0 ,2001)

sig1 = np.sin(2*np.pi*200*t)
sig2 = np.arange(2001) % 40 < 20

plt.figure()
plt.plot(t[:100],sig1[:100])
plt.title("Señal Original")
plt.show()
```



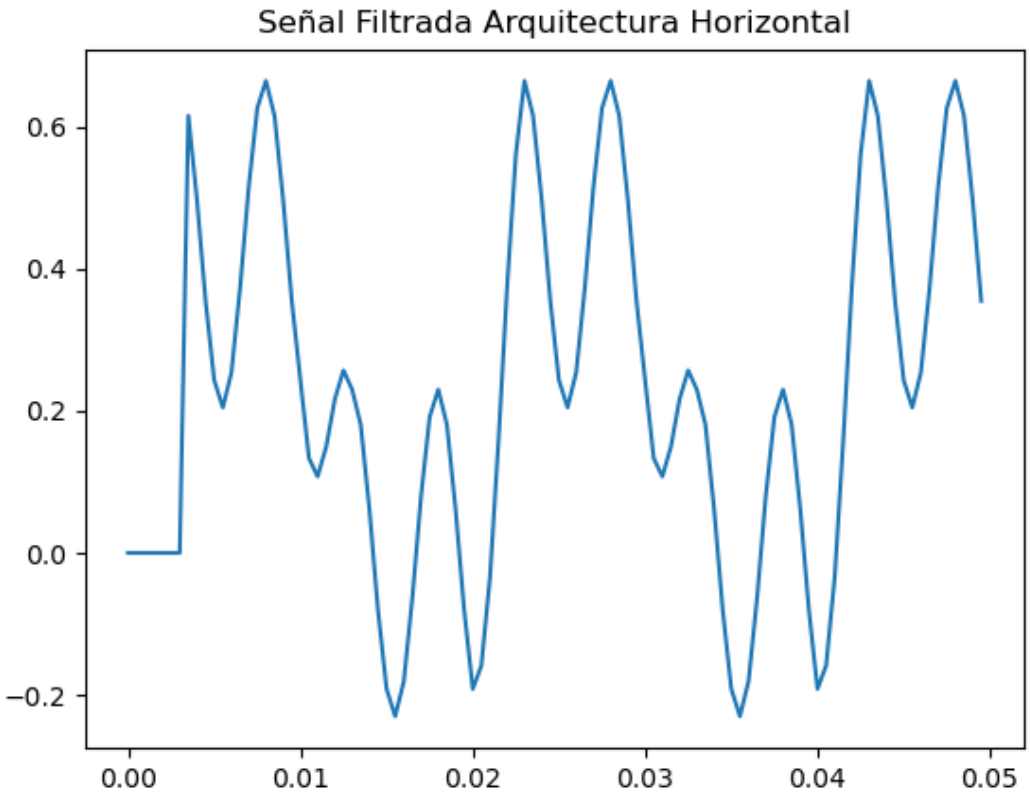

```
plt.figure()
plt.plot(t[:100],sig2[:100])
plt.title("Señal de Ruido")
plt.show()
```



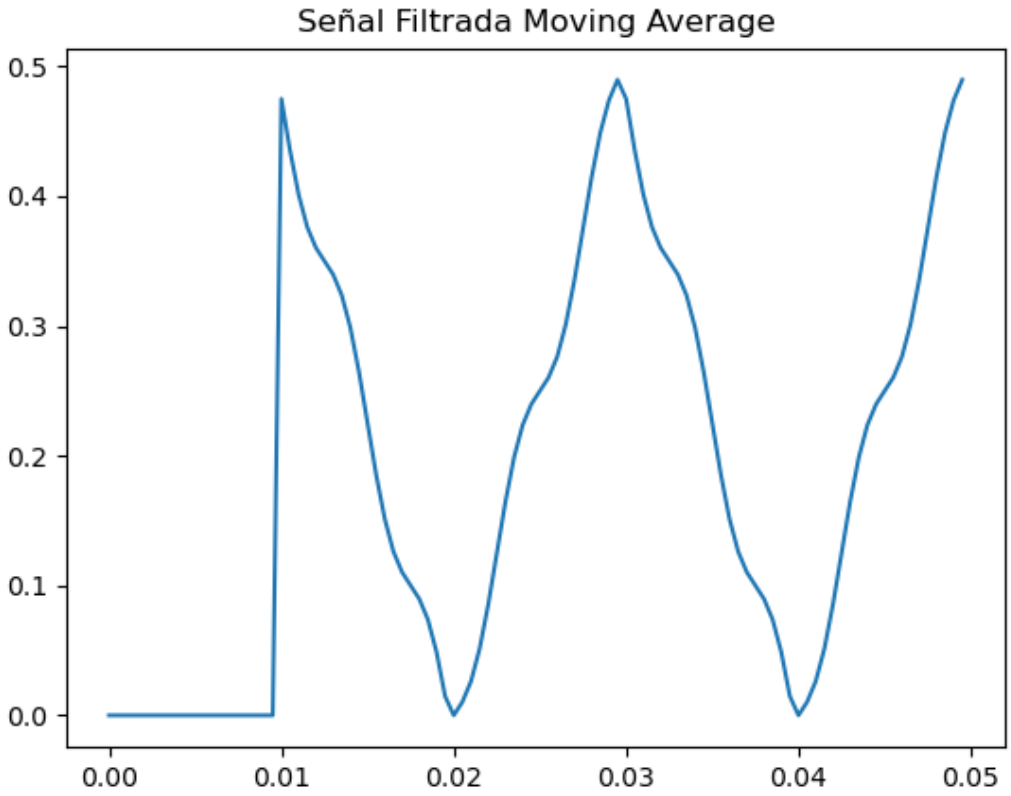

```
noised = (sig1 + sig2)/2

plt.figure()
plt.plot(t[:100],noised[:100])
plt.title("Señal Corrupta")
plt.show()


#Filtro FIR
taps = np.array([
    0.130951924144877080,
    0.142713213023077851,
    0.150057552393435101,
    0.152554620877219965,
    0.150057552393435101,
    0.142713213023077851,
    0.130951924144877080,
])

filtered_signal = np.convolve(noised,taps)
plt.figure()
plt.plot(t[:100], filtered_signal[:100])
plt.show()
plt.title("Señal Filtrada por Convolución")
```




```
N_fir = len(taps) #Window size
y = np.zeros(N_fir)
filter_out = np.zeros(2001)
i = 0
filt = 0

while(i < 2001):
    read = noised[i]
    filt = 0

    #implementacion del registro de corrimiento
    if i < N_fir:
        y[i] = read
    else:
        for j in range(N_fir-1):
            y[j] = y[j+1]
        y[N_fir-1] = read

        #implementación dle filtro
        for k in range(N_fir-1):
            filt += y[k] * taps[k]
        #filt /= M

        filter_out[i] = filt
    i += 1

plt.figure()
plt.plot(t[:100],filter_out[:100])
plt.title("Señal Filtrada Arquitectura Horizontal")
plt.show()
```




```
#moving average
M = 20 #Window size
y = np.zeros(M)
filter_out = np.zeros(2001)
i = 0
filt = 0

while(i < 2001):
    read = noised[i]
    filt = 0

    #implementacion del registro de corrimiento
    if i < M:
        y[i] = read
    else:
        for j in range(M-1):
            y[j] = y[j+1]
        y[M-1] = read

        #implementación dle filtro
        for k in range(M-1):
            filt += y[k]
        filt /= M

        filter_out[i] = filt
    i += 1

plt.figure()
plt.plot(t[:100],filter_out[:100])
plt.title("Señal Filtrada Moving Average")
plt.show()
```



Anexo IV. Coeficientes de filtrado que utilizará el sistema FIR

```
taps = np.array([
    0.000631811429314009,
    0.000644723000622591,
    -0.000072132372454839,
    -0.000075654664841864,
    0.000782445048596496,
    0.000796659357287206,
    -0.000243096433006447,
    -0.000252334512039298,
    0.001092554492287446,
    0.001111495832438203,
    -0.000766325676266116,
    -0.000788817288673872,
    0.002348601848808357,
    0.002392425694632418,
    -0.007043738268029271,
    -0.043735737878352249,
    -0.032014688317583098,
    -0.007653006501586431,
    -0.030054441160780716,
    -0.060052208699229975,
    -0.032863170716859416,
    0.002781676626998498,
    -0.032343521904552280,
    -0.079061599349311257,
    -0.034006259700062692,
```

```
    0.030680090500608749,
    -0.033625852540950421,
    -0.142030226559258566,
    -0.034622123023053716,
    0.290041850422137037,
    0.476001202626314568,
    0.290041850422138314,
    -0.034622123023053675,
    -0.142030226559257733,
    -0.033625852540950497,
    0.030680090500608999,
    -0.034006259700062741,
    -0.079061599349310951,
    -0.032343521904552183,
    0.002781676626998603,
    -0.032863170716859437,
    -0.060052208699229712,
    -0.030054441160780765,
    -0.007653006501586443,
    -0.032014688317583272,
    -0.043735737878352263,
    -0.007043738268029289,
    0.002392425694632367,
    0.002348601848808240,
    -0.000788817288673917,
    -0.000766325676266145,
    0.001111495832438158,
    0.001092554492287388,
    -0.000252334512039313,
    -0.000243096433006433,
    0.000796659357287207,
    0.000782445048596475,
    -0.000075654664841877,
    -0.000072132372454870,
    0.000644723000622573,
    0.000631811429314012,
])
```

Anexo V. Simulación de señales SEMG

El modelo para una señal electromiográfica bipolar (Cerutti & Marchesi, 2011; Merletti & Farina, 2016; Rangayyan, 2015) se define como:

$$SEMG = SEMG(Z_a, t) - SEMG(Z_b, t)$$

Dónde la ecuación de propagación de impulsos eléctricos desde la unidad motora (nervio) hasta el punto de medición superficial con electrodo se describe por

$$SEMG = \sum_{m=0}^{m-1} Km \cdot f(t) * Im(t) * \delta\left(t - \frac{n}{r} - \tau_m\right) + noise$$

La atenuación de un MUAP se define por

$$f(t) = \frac{1}{4\pi\sigma_e} \cdot \frac{1}{\sqrt{(Z - Z')^2 + \sigma[(X - X')^2 + (Y - Y')^2]}}$$

Donde Z es la distancia longitudinal entre la MUAP y el punto de adquisición, X es el diámetro de la fibra muscular e Y es la distancia transversal entre el punto superficial de adquisición y la MUAP, $\sigma = \left(\frac{\sigma_i}{\sigma_e}\right)^2$ es la relación entre la conductividad de la fibra y el medio externo, siendo σ_e la conductividad del medio externo y σ_i la conductividad del axoplasma muscular

La distribución de corriente de una fibra muscular se define con la ecuación:

$$Im(t) = C \cdot A(\lambda v)^2(\lambda vt)(t - 6\lambda vt + \lambda v^2 t^2)e^{-\lambda vt}$$

Donde v → velocidad de conducción de la fibra muscular, A → amplitud del potencial de acción y C es una constante de relación definida por

$$C = \frac{d^2\sigma_i\pi}{4v^2}$$

Siendo en este caso 'd' el diámetro de la fibra muscular, λ es un factor de escala en mm^-1 (influencia la modulación del potencial de acción)

El estímulo de activación viene definido por

$$\delta\left(t - \frac{n}{r} - \tau_m\right)$$

Siendo 'n' el número de pulso, 'r' la tasa de disparo que se ve modificada entre activaciones por la relación r=r+dr y τ_m que es un offset temporal, siendo un tiempo antes del disparo de la unidad.

En la ecuación de SEMG se considera 'm' a cada una de las fibras musculares para las cuales se debe obtener la ecuación con la información espacial, de conductividad y variación característica de la fibra tras su contracción, Km es una constante de atenuación aplicada a manera de ruido para conseguir las señales más o menos atenuadas dependiendo de su cercanía con focos de interferencia electromagnética, la señal de estímulo debe ser generada a varias frecuencias para cubrir todo el espectro de la señal SEMG.

Todas estas consideraciones causan que el programa computacional resultante sea extremadamente pesado y complicado de cargar con información fiable y veraz para simular adecuadamente una señal SEMG

El código desarrollado en Python se describe a continuación, dada la carga computacional, no fue posible llegar a un resultado.

```
import numpy as np
import matplotlib.pyplot as plt
```

```
#number of fibers
fib = 100
#Firing rate
r = 3
Fs = 10000
r_sampled = r*Fs
#Time definition
t = np.arange(0, 10, 1/Fs)

#Pulse generation
dirac = np.zeros_like(t)
mat_dirac = np.zeros((fib,len(t)))

for y in np.arange(0, fib):
    r_var = r_sampled
    dr = round(np.random.randn()*250)
    r_var = r_var + dr
    for x in np.arange(0,len(t)):
        if x == r_var:
            dirac[x] = 1
            dr = round(np.random.randn()*250)
            r_var = r_sampled + r_var + dr
    mat_dirac[y,:] = dirac


#Atenuation function
sigma_e = 0.0379 #Siemens / m (grasa cutanea)
sigma_i = 0.1437 #Siemens / m
sigma = (sigma_i / sigma_e)**2

y = 35E-3 #35mm depth  of MU from surface
x = (25E-6)/2 #fiber diameter 25um
z1 = 65E-3 #65mm half fiber lenght
z2 = 85E-3 #2mm added for 2nd electrode
ftt1 = np.zeros(100)
ftt2 = np.zeros(100)


for i in np.arange(0, fib):
    dy = np.random.uniform(-2E-3, 2E-3)
    yf = y + dy
    ft1 =
(1/(4*np.pi*sigma_e))*(1/np.sqrt(((z1**2)+sigma*((x**2)+(yf**2)))))
    ftt1[i] = ft1
    ft2 =
(1/(4*np.pi*sigma_e))*(1/np.sqrt(((z2**2)+sigma*((x**2)+(yf**2)))))
    ftt2[i] = ft2

#Current distribution of fiber
av = 4.3 #oVelocidad Promedio de conducción
C = np.zeros(fib)
for i in np.arange(0, fib):
    dv = np.random.uniform(-0.29, 0.29)
    vf = av + dv
    C[i] = (pow(x*2,2)*sigma_i*np.pi)/(4*pow(vf,2))

cvf = 4.9 # Velocidad de conducción de fibras rapidas
cvs = 3.9 # Velocidad de conducción de fibras rapidas

A = 96E-3 #Amplitud del potencial de accion
```

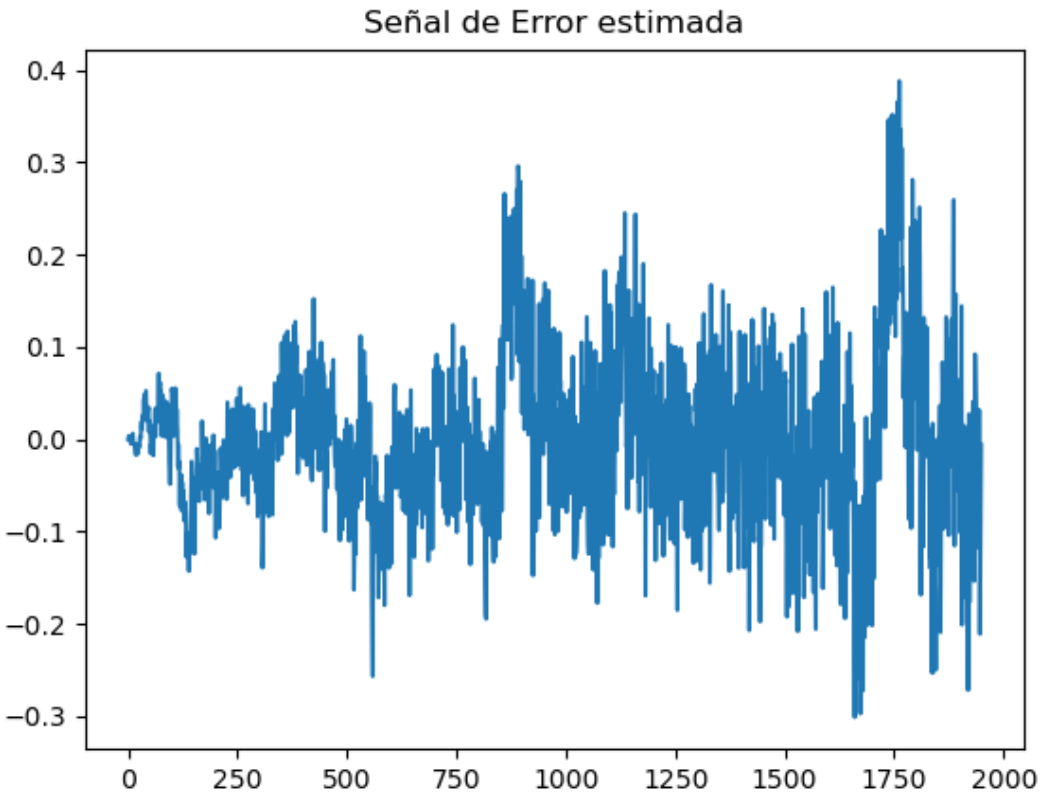
```
B = -90E-3 #Potencial de membrana remanente
lamb = 20 #Modulación del potencial de acción


Im = np.zeros((fib,len(t)))
for i in np.arange(0, fib):
    if i < int(fib * 0.42):
        dcvf = np.random.uniform(-0.3, 0.3)
        cvfm = cvf + dcvf
        Im[i] = C[i] * A* pow(lamb * cvfm, 2) * (lamb * cvfm * t) * (6 -
6*lamb * cvfm * t + lamb * pow(cvfm,2) * pow(t,2)) * np.exp(-lamb * cvfm *
t)
    else:
        dcvs = np.random.uniform(-0.3, 0.3)
        cvsm = cvs + dcvs
        Im[i] = C[i] * A* pow(lamb * cvsm, 2) * (lamb * cvsm * t) * (6 -
6*lamb * cvsm * t + lamb * pow(cvsm,2) * pow(t,2)) * np.exp(-lamb * cvsm *
t)

#SEMG construcion
semg1 = np.zeros((fib, 2*len(t)-1))
semg2 = np.zeros((fib, 2*len(t)-1))
for i in np.arange(0, fib):
    semg1[i,:] = ftt1[i] * np.convolve(Im[i],mat_dirac[i])
    semg2[i,:] = ftt2[i] * np.convolve(Im[i],mat_dirac[i])
semg = semg1 - semg2

emg = semg.sum(axis=0)
plt.figure()
plt.plot(emg)
```

Anexo VI. Diseño de filtro RLS

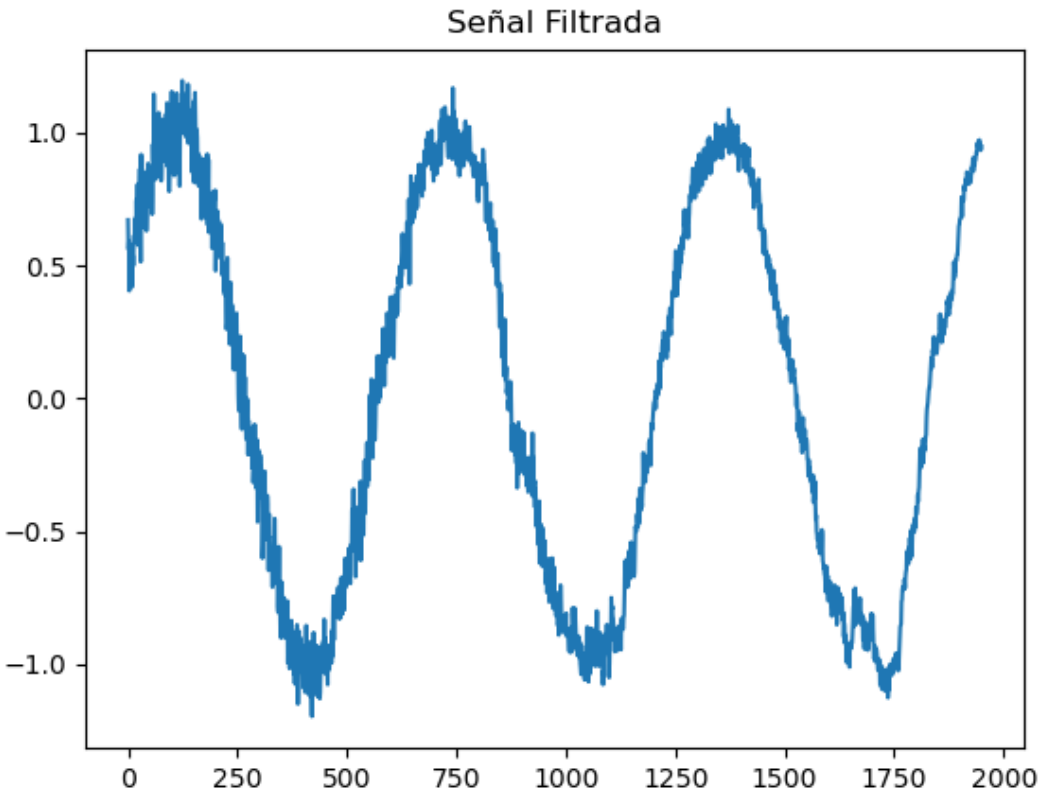
```
# # Diseño filtro RLS

# ## Creación de vectores de prueba

# In[1]:


import numpy as np
import matplotlib.pyplot as plt


# In[2]:


N = 2000
t = np.arange(0, N)
o = np.sin(0.01*t)
noise = np.random.normal(0, 0.1, N)
n_sin = o + noise


# ## Algoritmo RLS

# ### Hiperparámetros del filtro
```

```
# In[3]:


#Tamaño de ventana del filtro
M = 50
#Factor de regularización
delta = 0.1
#Factor de olvido
lbda = 0.95


# ### Inicialización vectores filtro

# In[4]:


#Vector de autocorrelación inversa
P = np.identity(M)
P = P*delta
#Matrices de entrada y pesos
W = np.zeros(M)
X = np.zeros(M)
Y = np.zeros(M)
D = np.zeros(M)
#variables auxiliares
j = 0
Out = []
Out2 = []


# In[5]:


for i in range(N):
    des = n_sin[i]
    inp = noise[i]
    if j < M:
        X[i] = inp
        #D[0,i] = des
        j += 1
    else:
        #shift register
        for k in range(M-1):
            X[k]= X[k+1]
            #D[0,k]= D[0,k+1]
        X[M-1] = inp
        #D[0,M-1] = des

        #cálculo del vector de salida
        y = np.dot(W, X)
        e = des - y
        P1 = np.dot(np.dot(np.dot(P, X), X.T), P)
        P2 = lbda + np.dot(np.dot(X, P), X.T)
        P = (1/lbda)*(P - P1/P2)
        dw = np.dot(P, X.T) * e
        W += dw

        Out.append(y)
        Out2.append(e)
```

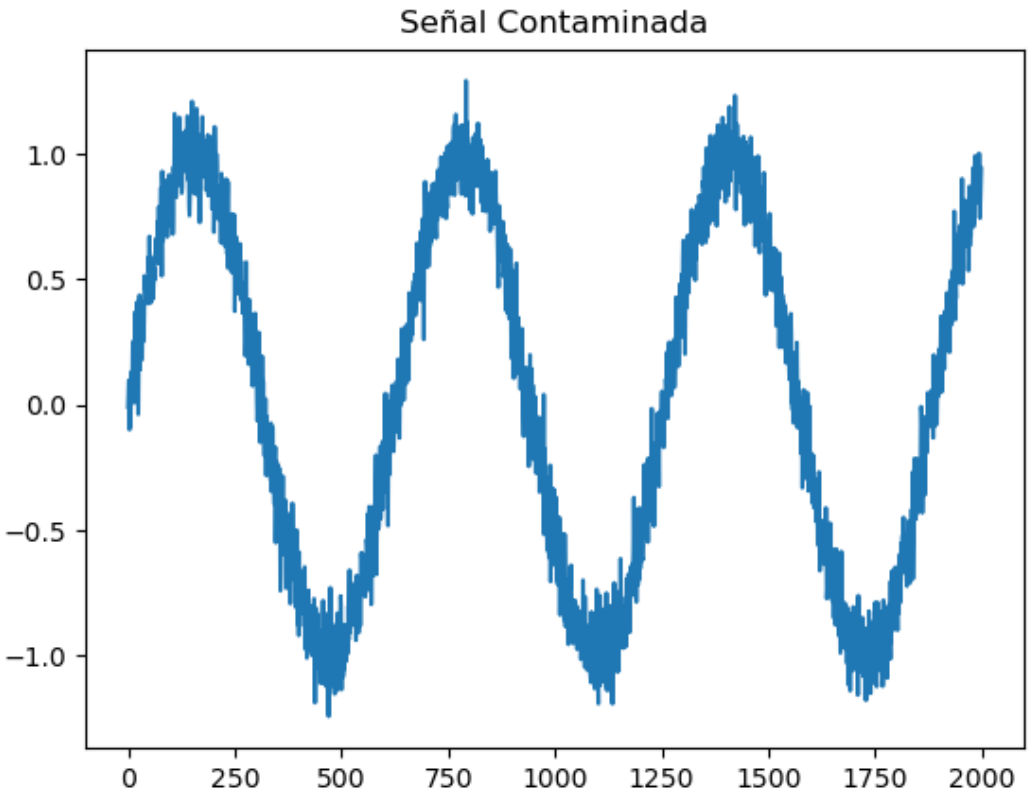
```
# In[6]:

plt.figure()
plt.plot(Out)
plt.title("Señal de Error estimada")
plt.show()
```



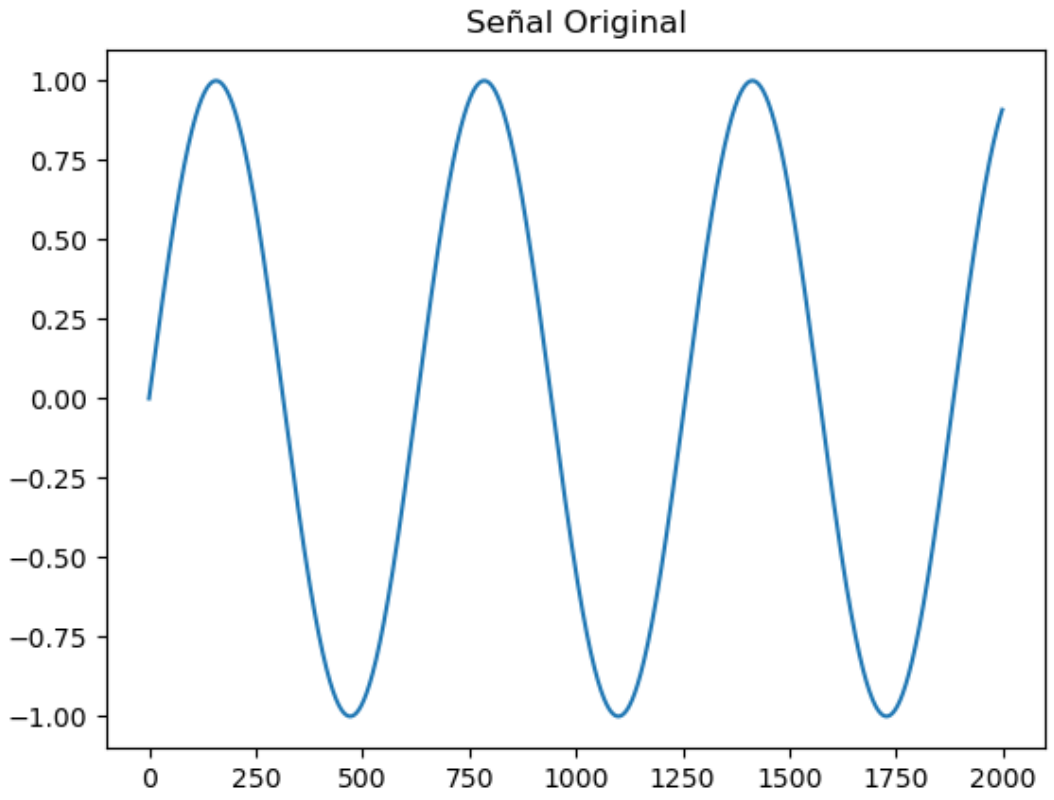

```
# In[7]:

plt.figure()
plt.plot(Out2)
plt.title("Señal Filtrada")
plt.show()
```




```
# In[8]:

plt.figure()
plt.plot(n_sin)
plt.title("Señal Contaminada")
plt.show()
```




```
# In[9]:

plt.figure()
plt.plot(o)
plt.title("Señal Original")
plt.show()
```



Anexo VII. Programa de Servidor Socket

```
import socket
import numpy as np

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #(IPV4 TCP-IP)
s.bind(('192.168.100.8', 1234)) #IP del servidor, puerto de conexión
s.listen(5) #Cola de 5 conexiones posibles que aceptará
#s.close para cerrar servidor
clientsocket, address = s.accept()
print(f"Se ha establecido una conexión desde {address}!")


#Generación del ruido correlacionado al que interfiere con la señal deseada
noise = np.random.normal(0,50,10000)

# # Algoritmo RLS
# ## Hiperparámetros del filtro
#Tamaño de ventana del filtro
M = 100
#Factor de regularización
delta = 0.01
#Factor de olvido
lbda = 0.99

# ## Inicializar vectores del filtro
#Vector de autocorrelación inversa
P = np.identity(M)
```

```
P = P*delta
#Matrices de entrada y pesos
W = np.zeros(M)
X = np.zeros(M)
Y = np.zeros(M)
D = np.zeros(M)
#Variables auxiliares
i = 0
j = 0


while True:

    #Recibir lectura desde SoC
    ans = clientsocket.recv(8)
    des = int.from_bytes(ans, byteorder = 'little')

    if i < 10000:
        inp = noise[i]
    else:
        i = 0
    i += 1


    #Shift register
    for k in range(M-1):
        X[k] = X[k+1]
    X[M-1] = inp

    #Calculo del vector de salida
    y = np.dot(W, X)
    e = des - y
    P1 = np.dot(np.dot(np.dot(P, X), X.T), P)
    P2 = lbda + np.dot(np.dot(X, P), X.T)
    P = (1/lbda)*(P - P1/P2)
    dw = np.dot(P, X.T) * e
    W += dw

    clientsocket.send((int(abs(e))).to_bytes(8, 'little'))
```

Anexo VIII. Programa de cliente Socket

```
# # Cliente socket de procesamiento distribuido

# ## Inicializar estructura de hardware

# In[1]:


from pynq import Overlay
overlay = Overlay('./PYNQ_read.bit')


# In[2]:


ADC = overlay.adc_read
```

```
DAC = overlay.axi_gpio_0


# ## Importar librerías

# In[3]:


import numpy as np
import matplotlib.pyplot as plt
import socket


# In[1]:


## Iniciar interfaz de red


# In[5]:


from pynq.lib.wifi import Wifi
port = Wifi()


# In[9]:


get_ipython().system('ifconfig')


# In[10]:


get_ipython().system('iwconfig')


# In[ ]:


get_ipython().system('ifconfig wlan0 up')


# In[6]:


ssid = ('DProano')
pwd = ('Daniel12021995')
port.connect(ssid, pwd)
#Si no funciona probar
#!sudo iwconfig wlan0 essid DProano key s:Daniel12021995


# In[8]:


get_ipython().system('ping 192.168.100.1 -c 10')


# ## Establecer cliente socket
```

```
# In[68]:


client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('192.168.100.8', 1234)) #IP del servidor y puerto de
conexión


# In[69]:


while True:
    #Lectura del ADC y envío al servidor
    lect = ADC.read()
    client.send((lect).to_bytes(8, 'little'))
    #Espera una respuesta del servidor y lo escribe en la DAC
    recv = client.recv(8) #Recibir bufer de 8 bytes
    escrt = int.from_bytes(recv, byteorder = 'little')
    DAC.write(0x0000,escrt)
```

Anexo IX. Artículo de investigación

Procesamiento de señales electromiográficas utilizando algoritmos de inteligencia artificial ‘on-the-edge’

Daniel David Proaño Guevara

Universidad Internacional de la Rioja, Cuenca (Ecuador)

19 de julio de 2020

Resumen

En los últimos años, la inteligencia artificial se ha posicionado en todos los campos tecnológicos, principalmente ejecutando cálculos en sistemas distribuidos o en la nube, pero en aplicaciones como el procesamiento de señales electromiográficas, se ha observado la necesidad de procesar en el dispositivo de adquisición, utilizando una metodología 'on-the-edge'. Este documento compara un sistema DSP tradicional, con un sistema informático distribuido y uno 'en el borde', para este propósito se utiliza un sistema embebido PYNQ-Z1, que evalúa las tres metodologías de procesamiento en escenarios similares. Los resultados muestran que el sistema de procesamiento 'en el borde' es más efectivo que los demás, ya que tiene una latencia aceptable, mayor seguridad en el manejo de la información y, dado el estado de la tecnología actual, su uso es justificable y recomendado en vez de los sistemas informáticos distribuidos

Palabras Clave

DSP, Edge Computing, EMG, Filtro RLS, Inteligencia Artificial, PYNQ-Z1

I. Introducción

En años recientes, la inteligencia artificial se encuentra en todos los campos de tecnología y tradicionalmente los cálculos inherentes a esta se han desarrollado en sistemas distribuidos o en la nube, pero recientemente se ha hecho evidente la necesidad de desarrollar nuevas soluciones que integren el cálculo dentro de los dispositivos de adquisición, es decir ‘edge-computing’ [1], [2]. Uno de estos campos tecnológicos es el procesamiento de señales electromiográficas [3], por lo que en este trabajo se muestra el desarrollo de sistemas de procesamiento de señales electromiográficas que permitan comparar la aplicación de algoritmos inteligentes ‘on-the-edge’, con un sistema distribuido de cómputo, y técnicas de DSP tradicionales para identificar las ventajas y desventajas de cada sistema y planificar a futuro la incorporación de sistemas inteligentes sobre hardware para la adquisición de señales EMG enfocadas al control de dispositivos protésicos de miembro superior.

Para la implementación de los sistemas de procesamiento se utilizará el sistema embebido PYNQ-Z1 sobre el cual se implementará un sistema de DSP tradicional compuesto por una etapa de filtrado paso alto a 20Hz, paso bajo a 500Hz, rectificación por valor absoluto y un filtro moving average, como segundo sistema se implementa un filtro RLS que se ejecute en tiempo real con una arquitectura horizontal en el sistema embebido y por último se utilizará al sistema embebido para enviar datos a un sistema distribuido de computación que ejecutará al filtro RLS diseñado y reconstruirá las señales resultantes de este en el sistema embebido.

Las métricas de evaluación para los escenarios de prueba se basarán en la comparación de medias y desviación estándar de la señal de entrada y salida, la relación señal a ruido, el desfase temporal asociado al tiempo de procesamiento, y un análisis frecuencial, que se lleva a cabo visualmente con resultados del analizador de señales Analog Discovery 2.

Es necesario diseñar un sistema de hardware desde HDL que se implementará en el sistema embebido, construyendo de esta manera una arquitectura específica para la comparación de los sistemas de procesamiento y se diseñan e implementa los sistemas de filtrado. Finalmente se comparan las métricas de evaluación buscando de esta manera determinar la arquitectura más adecuada para las tareas de filtrado de señales EMG.

Los resultados muestran que el sistema de DSP tradicional es el más lento de todos los planteados, sin embargo no necesita de conocimiento sobre el ruido inherente a la señal para poder filtrar, el sistema de procesamiento distribuido es el más rápido en ejecución pero tiene problemas de seguridad en la comunicación de las señales, errores de comunicación que corrompen la señal y se tarda más tiempo en aprender la fuente inherente de ruido del sistema, y el sistema de procesamiento ‘on-the-edge’ muestra ser el sistema lo suficientemente rápido para que rivalice con el sistema distribuido e integre todas las características de adaptabilidad que se desean en el sistema.

Por lo tanto, se concluye que el sistema de procesamiento ‘on-the-edge’ es más efectivo que un sistema de procesamiento tradicional y es justificable su aplicación en los nuevos sistemas de procesamiento de señales electromiográficas, superando con creces al procesamiento tradicional DSP y utilizando menos recursos que el sistema distribuido con un excelente tiempo de procesamiento.

II. Estado del Arte

La electromiografía es una técnica experimental dedicada al desarrollo, registro y análisis de las señales mioeléctricas. Las señales mioeléctricas, se construyen por variaciones fisiológicas en el estado de las membranas de las fibras musculares [4]. Esta técnica prueba la integridad del sistema motor entero, que consiste en las motoneuronas superiores e inferiores, unión neuromuscular y el músculo [5].

Uno de los principales problemas con las señales SEMG es que son una de las señales electrofisiológicas más fáciles de medir, pero también es una de las más complejas de interpretar cuantitativamente por lo tanto [6] expresa que "para su detrimento, la electromiografía es muy fácil de usar y consecuentemente muy fácil de abusar", lo que implica que hay que tener especial cuidado en las técnicas de adquisición y procesamiento de dichas señales.

El procesamiento digital de señales (DSP por sus siglas en inglés) se diferencia de otras áreas de las ciencias de la computación por el único tipo de datos que utiliza: señales, que por lo general son adquiridas del mundo real, como ondas sonoras, sísmicas, biométricas, etc. El procesamiento digital de señales consiste en las matemáticas, algoritmos y las técnicas utilizadas para manipular, transformar y representar estas señales después que hayan sido digitalizadas [7], [8].

La evolución de los diferentes enfoques empleados en el procesamiento de señales biomédicas, dificultan la adecuada elección de un método que se utilizará en tareas de filtrado de señales, extracción de información de validez clínica, clasificación diagnóstica, monitoreo de pacientes, etc. [3].

Tomando como cierto que las señales biológicas llevan información relativa al sistema o sistemas que lo generan, se establece que el procesamiento de la señal debe seguir los siguientes objetivos:

1) Resaltar la información útil de la señal original

2) Interpretar los resultados y validar los parámetros obtenidos de la subsecuente etapa de decisión

3) Producir innovación en la mejora del conocimiento fisiológico, la producción de nuevos equipos y dispositivos "inteligentes" y la definición de nuevos protocolos para la prevención, diagnóstico y terapia.

La señal EMG sin procesamiento ofrece muy poca información, por lo que para el análisis inicial se pueden considerar análisis frecuencial, de amplitud, o ambos. Las mediciones de amplitud están relacionadas a la fuerza, torque y activación muscular, mientras que el análisis frecuencial ofrece información sobre la fatiga muscular, algunas de las técnicas más comunes de procesamiento tradicional son la rectificación de la onda, sea calculando el valor absoluto o cuadrática, elevando los valores de las muestras al cuadrado y se realiza un promediado temporal, móvil o ponderado. Esto ofrece valores, en el caso lineal del valor rectificado promedio (ARV) y en el caso cuadrático, como la media cuadrática (RMS). En el análisis frecuencial, se obtienen parámetros del cálculo del espectro de potencia en épocas (epochs) temporales con una duración promedio de 0.25 a 1 segundos. De aquí se obtiene la frecuencia media (MNF) y mediana (MDF); también para el análisis y descomposición de las unidades de disparo de las motoneuronas se analiza la razón de cruce por cero de la señal (ZCR). Los filtros frecuenciales que se implementan de tipo FIR e IIR, comparten la eliminación de banda por debajo de los 20Hz hasta los 1000Hz [9]–[16].

El procesamiento inteligente de señales (ISP por sus siglas en inglés) utiliza aprendizaje y otras técnicas 'inteligentes' para extraer tanta información como sea posible de los datos de la señal que se está recibiendo. El procesamiento de señales clásico ha trabajado ampliamente con modelos matemáticos que son lineales, estacionarios, gaussianos y por esto son ampliamente utilizados, pero los sistemas reales son no lineales, con estructuras estadísticas erráticas o impulsivas que pueden variar en el tiempo. Cambios mínimos en la señal o en la estructura del ruido pueden llevar a cambios cualitativos en cómo los sistemas de procesamiento clásico filtran el ruido o mantienen la estabilidad [17].

Los filtros adaptativos utilizan técnicas que se adecúan a las condiciones de la señal de entrada manteniendo estable la respuesta del sistema, así se han constituido en uno de los métodos más eficientes para la adquisición de señales fisiológicas [18].

Una aproximación para la eliminación de ruido es la cancelación adaptativa de ruido de las señales EMG utilizando una fuente de ruido externa que esté vagamente relacionada con el ruido implícito en la señal EMG. Para esta tarea se han implementado algoritmos de filtrado como el Filtro de Kalman, LMS, RLS, Wiener, UFIR, Gaussiano, algoritmos de colonia de abejas, Bayesianos, entre otros [19]–[25].

Se han encontrado escasos trabajos relacionados al procesamiento inteligente de señales EMG en el borde, [26], [27] realizan el proceso de adquisición y filtrado digital tradicional utilizando como dispositivo común una GPU de propósito general 'NVidia Jetson', sobre la cual implementan algoritmos de aprendizaje automático y toma de decisiones, pero no se evidencia un procesamiento inteligente de la señal per se.

En el presente trabajo se utilizará la arquitectura propuesta por [14] para el procesamiento digital tradicional de las señales, el cual está compuesto por filtros paso bajo y paso alto, una etapa de rectificación y suavizado. Para el procesamiento inteligente se utilizará un filtro RLS con una fuente de ruido blanco que será implementado en un computador y en un dispositivo en el borde

III. Objetivos y Metodología

Objetivo general, objetivos específicos y metodología de trabajo aplicada.

Objetivo general

Desarrollar sistemas de procesamiento de señales electromiográficas que permitan comparar la aplicación de algoritmos inteligentes 'on-the-edge', con un sistema distribuido de cómputo, y técnicas de DSP tradicionales para identificar las ventajas y desventajas de cada sistema y planificar a futuro la incorporación de sistemas inteligentes sobre hardware para la adquisición de señales EMG enfocadas al control de dispositivos protésicos de miembro superior.

Objetivos específicos

- Conocer las técnicas de procesamiento digital de señales para señales electromiográficas.
- Diseñar un escenario de pruebas para la comparativa de tres sistemas de procesamiento de señales que sigan paradigmas diferentes de implementación.
- Implementar algoritmos de procesamiento de

señales utilizando técnicas clásicas, técnicas inteligentes utilizando un sistema distribuido de cálculo y técnicas inteligentes sobre un dispositivo en el borde

- Evaluar los sistemas de procesamiento de señales

Metodología

Para la implementación de los sistemas de procesamiento se utilizará el sistema embebido PYNQ-Z1, que utiliza un SoC ZYNQ-7000 de XILINX compuesto por un ARM Cortex-A9 y celdas de lógica programable de la familia Artix-7

De cara a la evaluación de tres sistemas de procesamiento se establece un escenario de pruebas, el cual está dividido en los siguientes pasos:

1. Diseñar una arquitectura de filtrado tradicional compuesta por:
    a. Filtro paso alto con frecuencia de corte en 20Hz
    b. Filtro paso bajo con frecuencia de corte en 500Hz
    c. Rectificación obteniendo el valor absoluto de la muestra
    d. Suavizado utilizando un filtro de promediado móvil (MA)
2. Diseñar una arquitectura de filtrado inteligente que utilice filtrado RLS
3. Implementar el filtro tradicional en el sistema embebido
4. Utilizar el sistema embebido para captura, envío de datos al PC, y reconstrucción de la señal analógica, procesando la señal en la arquitectura de filtrado inteligente
5. Implementar la arquitectura inteligente en el sistema embebido

Las métricas de evaluación para los escenarios de prueba se basarán en la comparación de medias y raíz media cuadrática de la señal de entrada y salida, el desfase temporal asociado al tiempo de procesamiento, y un análisis del espectrograma de las señales.

Finalmente se comparan las métricas de evaluación buscando de esta manera determinar la arquitectura más adecuada para las tareas de filtrado de señales EMG.

IV. CONTRIBUCIÓN

La contribución se divide en los tres experimentos diseñados, DSP tradicional, implementación 'on-the-edge' e implementación en un sistema distribuido.

DSP Tradicional

El sistema de procesamiento tradicional consta de un filtro paso banda entre 20 y 500 Hz en estructura FIR, un filtro de promediados con una ventana de 7 muestras. Los filtros fueron diseñados originalmente en Python utilizando vectores de simulación y se implementan en el sistema PYNQ.

La estructura del filtro que se va a implementar, para que sea capaz de procesar y operar en tiempo real, es la conocida como estructura horizontal del filtro [28], el filtro diseñado consta de 61 coeficientes que describen un filtro paso banda entre 20 y 500Hz, posteriormente se aplica un filtro de promediado con una ventana de 20 elementos, de los cuales previamente se obtiene su valor absoluto a manera de rectificación.

Implementación 'on-the-edge'

El filtrado adaptativo es una técnica de 'Online Learning' ya que entrena sus parámetros mientras adquiere la información a diferencia del aprendizaje automático que por lo general se entrena con todo el conjunto de datos o por lo menos con un mini lote de datos, pero el filtrado adaptativo se puede considerar una técnica de Inteligencia Artificial ya que busca minimizar una señal de error utilizando el descenso estocástico del gradiente (SGD).

La diferencia más importante entre un filtro adaptativo y un filtro tradicional es que el filtro adaptativo cambia sus coeficientes y forma de comportarse en el tiempo. Los coeficientes se calculan cuando el filtro se implementa y se ajustan durante su fase de aprendizaje.

El algoritmo de filtrado RLS (Recursive Least Square) busca la minimización de la suma de los cuadrados de las diferencias entre la señal deseada y la salida del filtro, actualizándose de manera iterativa conforme adquiera nueva información. Este algoritmo resuelve la estimación de mínimos cuadrados de manera recursiva [29]–[31].

Las ecuaciones que se utilizarán en este filtro están basada en los trabajos de [29]–[32]:

1. Inicialización
    a. Determinar el tamaño de ventana del filtro M
    b. Determinar el factor de olvido $0 \ll \lambda < 1$
    c. Determinar el factor de regularización δ que es la inversa de la potencia estimada de entrada
    d. Creación de la matriz de autocorrelación inversa

$$\boldsymbol{P}(-1) = \delta^{-1} * I$$

    e. Creación de matrices de pesos

$$\boldsymbol{W}(-1) = \boldsymbol{X}(-1) = [0\ 0\ 0\ \ldots M_{-1}]^T$$

2. Algoritmo de filtrado (repetir)
    a. Lectura de entrada y valor estimado, X y d respectivamente
    b. Cálculo de la salida del filtro

$$y(n) = \boldsymbol{W}(n) \cdot \boldsymbol{X}(n)$$

    c. Cálculo del error

$$e = d - y$$

    d. Actualización de la matriz de autocorrelación inversa

$$\boldsymbol{P}(n) = \left(\frac{1}{\lambda}\right) * \left(\boldsymbol{P}(n-1) - \frac{\boldsymbol{P}(n-1) \cdot \boldsymbol{X}(n) \cdot \boldsymbol{X}^T(n) \cdot \boldsymbol{P}(n-1)}{\lambda + \boldsymbol{X}^T(n) \cdot \boldsymbol{P}(n-1) \cdot \boldsymbol{X}(n)}\right)$$

    e. Actualización de pesos por descenso del

gradiente (SGD)

$$dw = \boldsymbol{P}(n) \cdot \boldsymbol{X^T}(n) * e$$

$$\boldsymbol{W}(n) = \boldsymbol{W}(n) + dw$$

Para esta aplicación se utiliza una estructura lineal de filtro FIR y RLS, lo que permitirá un aprendizaje en tiempo real de las variaciones respecto al ruido. En esta arquitectura la entrada al filtro es la señal modelada o medida de ruido $X(n) = r'(n)$, la señal deseada es la señal adquirida (que se entiende que es la señal deseada añadida con ruido) $d(n) + r(n)$, por lo tanto, la salida $y(n)$ del filtro sería el modelo del ruido, y la salida $e(n)$ corresponde a la señal filtrada.

En la Figura 1 se presenta la arquitectura de un filtro RLS horizontal, el cual puede procesar en tiempo real los datos adquiridos. Esta arquitectura recuerda en gran manera a una unidad neuronal de un perceptrón, ya que en el paso hacia adelante realiza la sumatoria de las multiplicaciones de las entradas por sus pesos $\sum_k X(k) * W(k)$, lo compara con la señal deseada $d(k)$ y genera una señal de error $\epsilon(k)$ que se retro propaga por la estructura con un algoritmo basado en el descenso estocástico del gradiente (SGD).

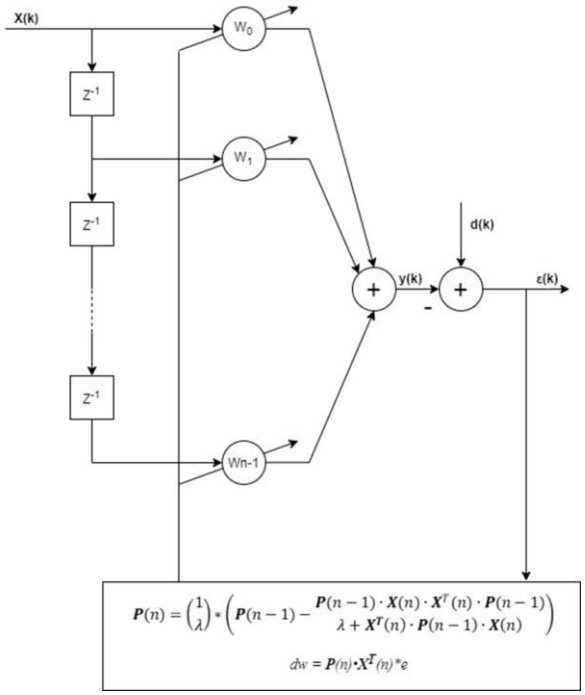


Fig. 1. Filtro RLS en arquitectura horizontal.

Implementación en computación distribuida

Para establecer el sistema distribuido de computación, es decir que se utilicen elementos discretos especializados para diferentes tareas, la estructura de red utilizada se presenta en la Figura 2. Al sistema embebido se le agrega una tarjeta de red inalámbrica tp-link TL-WN823N, con una velocidad de transmisión a 300Mbps, que se comunicará por sockets utilizando protocolo TCP IPv4 con un servidor de cálculo establecido en un computador Lenovo con procesador Intel Xeon CPU E3-1535M v6 de 3.1 GHz, y memoria RAM de 64 Gb.

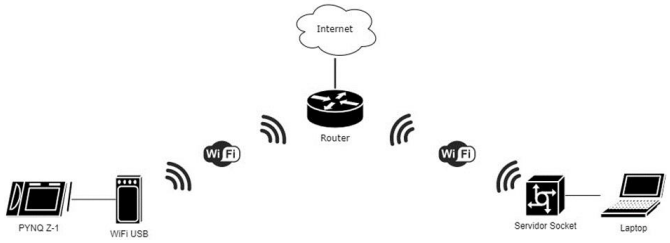


Fig. 2. Estructura de red para comunicación con servidor de cálculo.

En ambientes de computación distribuida, una aplicación cliente-servidor puede ser diseñada utilizando programación por sockets. Un Socket es un enlace de comunicación bidireccional entre un programa de cliente y servidor que se ejecutan en un ambiente de red [33], [34]. De esta manera el sistema embebido digitaliza las señales, las envía al procesador de cálculo donde se aplica el filtro RLS horizontal y envía el resultado de vuelta al sistema embebido, donde es reconstruida.

V. RESULTADOS

Para la obtención de resultado se realizaron pruebas y mediciones de espectrogramas, señales de AC RMS y un análisis de retardo de propagación analizado con señales senoidales puras.

DSP Tradicional

En la Figura 3 se muestra el funcionamiento normal del algoritmo de DSP .

El análisis visual de las señales muestra un cambio radical en la forma de onda, rectificando los impulsos y llevándola a oscilar desde la línea 0, en vez de una componente aditiva que mueve la línea de oscilación como es el caso de la señal original, la media de la señal original es de 2.1908V y la señal filtrada es de 0.9419V, lo que comprueba lo descrito anteriormente.

En la comparativa de espectrogramas se observó que en el sistema filtrado se elimina buena parte de la información de alta frecuencia mientras que se contamina la señal de baja frecuencia. Lo que puede resultar en la inutilización de las señales para los propósitos de detección de la intención de movimiento.

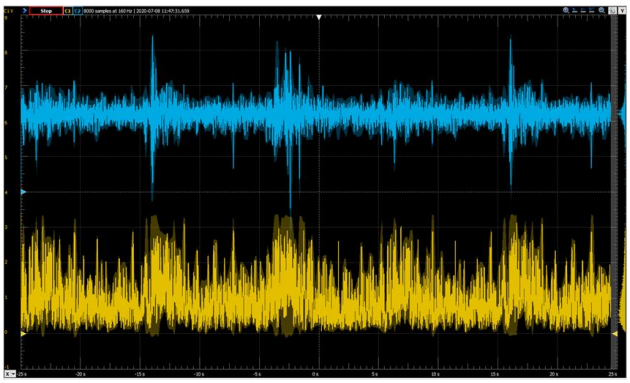


Fig. 3. Funcionamiento normal de DSP tradicional, señal original [azul] y señal filtrada [amarillo]

El análisis visual de AC RMS se observó que ambas señales mantienen en gran manera la correlación entre sus señales, pero la señal filtrada se sobrepone a la señal original como si estuviese tratando de ser su señal portadora.

En el análisis comparativo de la diferencia de la señal original con la señal filtrada es una composición aditiva entre ambas señales, lo que impide identificar adecuadamente la relación de señal a ruido. La media de dicha señal se encuentra en -1.2488V.

El retardo de propagación entre las señales analizadas es de 35.71ms y de 37.15ms dando un promedio de 36.43ms, que es el tiempo que le toma al algoritmo filtrar las señales.

Procesamiento 'on-the-edge'

En la Figura 4 se muestra el funcionamiento normal del sistema de adquisición y procesamiento 'on-the-edge'.

El análisis visual de las señales muestra que mantienen una gran relación de aspecto, sin embargo, el cambio no es radical dada la fuente de ruido, la media de la señal original es de 2.192V y la señal filtrada es de 2.173V lo que muestra que mantienen sus características en gran manera

En la comparativa de espectrogramas no se aprecia gran cambio, excepto el efecto de difuminado en los picos impulsivos, regularizando la señal

El análisis visual de AC RMS se observa que ambas señales mantienen en gran manera la correlación entre sus señales, y que busca adaptarse a la fuente de ruido para minimizarla, la señal filtrada varía conforme el algoritmo se adapta a la fuente de ruido.

En el análisis comparativo de la diferencia de la señal original con la señal filtrada permite identificar la señal de ruido que se está eliminando, con los artefactos impulsivos en tiempo y frecuencia que le restan calidad a la señal. La media de dicha señal se encuentra en -18.79mV.

El retardo de propagación en ambos casos es de 4.074ms y de 3.47ms dando un promedio de 3.822ms, que es el tiempo que le toma al algoritmo filtrar las señales.

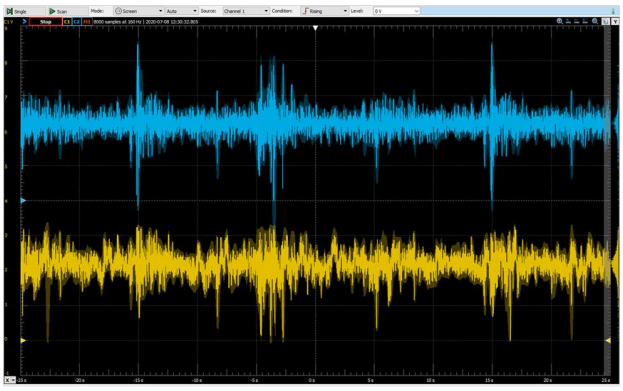


Fig. 4. Funcionamiento normal con procesamiento 'on-the-edge', señal original [azul] y señal filtrada [amarillo]

Procesamiento en computación distribuida

En la Figura 5 se muestra el funcionamiento normal del sistema de procesamiento distribuido.

El análisis visual de las señales muestra que mantienen una gran relación de aspecto, sin embargo, el cambio no es radical dada la fuente de ruido, la media de la señal original es de 2.193V y la señal filtrada es de 2.175V lo que muestra que mantienen sus características en gran manera. Al iniciar el proceso de filtrado de la seña, se observa una gran cantidad de ruido aleatorio que corrompe la señal.

En la comparativa de espectrogramas no se aprecia gran cambio, excepto el efecto de difuminado en los picos impulsivos, regularizando la señal y la concentración de las componentes frecuenciales en las bandas bajas.

El análisis visual de AC RMS se observa que ambas señales mantienen en gran manera la correlación entre sus señales, y que busca adaptarse a la fuente de ruido para minimizarla, la señal filtrada varía conforme el algoritmo se adapta a la fuente de ruido.

En el análisis comparativo de la diferencia de la señal original con la señal filtrada permite identificar la señal de ruido que se está eliminando, con los artefactos impulsivos en tiempo y frecuencia que le restan calidad a la señal. La media de dicha señal se encuentra en -17.359mV.

El retardo de propagación en ambos casos es de 2.63ms y de 2.517ms dando un promedio de 2.5735ms, que es el tiempo que le toma al algoritmo filtrar las señales

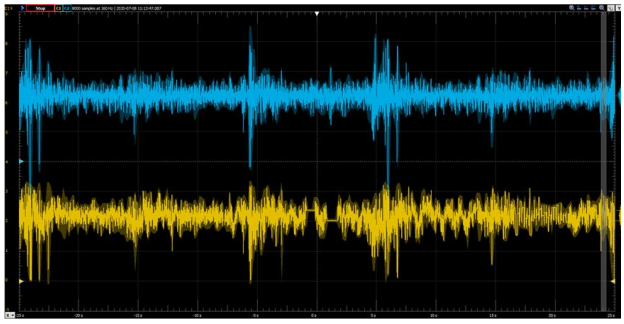


Fig. 5. Funcionamiento normal de sistema de computación distribuida, señal original [azul] y señal filtrada [amarillo]

Los resultados se presentan en la Figura 6 como un perfil de

	DSP Tradicional				Procesamiento 'on-the-edge'				Procesamiento distribuido			
	- -	-	+	++	- -	-	+	++	- -	-	+	++
Tiempo de procesamiento	■						■					■
Convergencia del sistema				■		■			■			
Uso de recursos físicos			■				■		■			
Calidad de filtrado		■					■					■
Complejidad del código o algoritmo	■							■		■		
Estabilidad de la ejecución				■				■		■		

Fig. 6. Perfil de Harris para la evaluación de propuestas de procesamiento de señales sEMG

Harris para la evaluación objetiva de las tres metodologías de procesamiento.

La metodología de procesamiento más balanceada dentro del perfil de Harris es la de procesamiento 'on-the-edge' ya que puntúa en la mayoría de las categorías como positivo o fuertemente positivo

VI. Discusión

Los resultados y mediciones presentados en los apartados anteriores muestran que, si bien el procesamiento distribuido es el más rápido, es más inestable, generando pérdidas de información cuando se saturan los buffer de los socket de comunicación, además necesitar más equipos de hardware, debido a la implementación de un sistema de comunicación inalámbrica basada en wifi. Se necesita un intermediario en la estructura de red que puede estar conectado a internet, lo que afecta la seguridad de los datos transmitidos y un sistema central de cómputo de buenas características para que pueda llevar a cabo todos los procesos de cálculo y comunicación en un tiempo adecuado.

En cuanto al sistema de DSP tradicional, es el más utilizado hasta la actualidad por su relativa simplicidad de implementación y que no necesita que se modele o se adquiera el ruido de la señal que se va a eliminar, permitiendo que el sistema de procesamiento actúe por sí mismo. Además, los algoritmos presentados en este trabajo de fin de máster pueden ser optimizados utilizando lenguaje C o incluso construyendo módulos de hardware en HDL que puedan ser implementados en la parte de FPGA del sistema embebido, acelerando radicalmente la velocidad de cómputo del sistema. En este caso como se evalúa los tres sistemas en un mismo marco imparcial, resulta ser el más lento en procesamiento tarda 10 veces más.

En el sistema de procesamiento 'on-the-edge', si bien el retardo de procesamiento es de aproximadamente un 70 %, sigue siendo mucho más rápido que el sistema de procesamiento de DSP tradicional, lo que implica un adecuado uso de recursos de hardware y software con la gran ventaja de la adaptabilidad a las fuentes de ruido que puede sufrir una señal EMG de superficie, las cuales pueden cambiar en el tiempo en amplitud y naturaleza. En el presente trabajo no se puede observar la capacidad completa de estos sistemas de filtrado adaptativo por la imposibilidad logística de hacer adquisiciones de señal y ruido sobre sujetos de prueba, lo que limita y sesga los resultados expuestos. Al igual que en el sistema de DSP tradicional, los algoritmos implementados en Python pueden ser optimizados en C, o directamente en HDL para su ejecución en hardware en celdas de FPGA, lo que permitiría la paralelización del cálculo y la escalabilidad del procesamiento sin comprometer el tiempo de procesamiento de estos.

VII. Conclusiones

Para tener una evaluación objetiva de los sistemas de procesamiento, se realiza una comparación entre el sistema de DSP tradicional, el sistema de procesamiento discreto y el sistema de procesamiento 'on-the-edge' en un mismo dispositivo, utilizando una misma base de datos reconstruida por un generador de funciones y se evalúan los espectrogramas de las señales, los retardos debidos al procesamiento, y valores como la raíz media cuadrática eliminando componentes de corriente continua. El algoritmo inteligente de filtrado es un filtro adaptativo RLS, al cual como fuente de ruido se le introduce una señal de ruido blanco simulada, ya que la simulación de una línea base de EMG o la adquisición de esta es imposible en el contexto en el que se desarrolla el presente trabajo de fin de máster.

En la revisión de literatura no se ha encontrado un aporte sobre la implementación de filtros adaptativos en un dispositivo 'on-the-edge', la mayoría utilizan sistemas distribuidos de cálculo en MATLAB. En el presente trabajo se presenta una implementación a tiempo real de algoritmos de filtrado adaptativo, la mayor contribución del trabajo de investigación.

Con la disponibilidad tecnológica actual, es adecuado e incluso recomendable realizar el procesamiento inteligente sobre el dispositivo, incluso si es necesario un sistema de procesamiento distribuido, ya que disminuirá la carga computacional al recibir información útil y no únicamente datos en crudo. Esto permitirá construir arquitecturas de procesamiento y uso de las señales más complejas ofreciendo mejores resultados a los usuarios finales.

Por lo tanto, respondiendo a las preguntas de investigación, un procesamiento inteligente 'on-the-edge' de señales electromiográficas es más efectivo que el procesamiento tradicional en hardware. Lo anterior debido a que el tiempo de retardo entre las señales es menor, y si se caracteriza adecuadamente la señal de ruido que interfiere con la señal deseada, o se la adquiere con sensores secundarios, puede ofrecer información muy pertinente sin cambios considerables en la naturaleza frecuencial y temporal de las señales electromiográficas. Por otro lado, considerando el estado actual de la tecnología, es justificada la aplicación de técnicas de inteligencia artificial 'on-the-edge' ya que disminuyen la carga computacional que sufren otros dispositivos, permiten generar arquitecturas con mayor escalabilidad y se puede considerar pertinente la autonomía de procesamiento entre cada uno de los nodos inteligentes que pueden conformar un sistema de adquisición en incluso es posible la autonomía total de cada sensor y que no requiera de un sistema auxiliar de cómputo para realizar el proceso de toma de decisiones o acondicionamiento secundario de las señales.

Para futuros trabajos investigativos, se bajará de nivel lo más posible para la implementación de los algoritmos, en el caso de DSP tradicional y de procesamiento 'on-the-edge' los algoritmos deberán ser implementados en lenguaje HDL en la lógica programable de un FPGA, lo que permitirá acelerar por hardware las funciones y crear sistemas realmente paralelos y multicanal de procesamiento, ya que no se limitarán a la lógica secuencial de la programación de software. En cuanto al procesamiento distribuido, se optimizará el procesamiento realizando un código en C/C++ que permita cálculos incluso más rápidos que los presentados en el presente trabajo.

Igualmente es necesario comparar las tecnologías disponibles para procesamiento en el dispositivo como microcontroladores de 16 y 32 bits, tarjetas gráficas de propósito general, FPGA y SoC, en los cuales se implementará el algoritmo de filtrado adaptativo, y se analizará la relación costo-beneficio entre los mismos, el consumo de potencia y la escalabilidad de la arquitectura para implementar otros sistemas necesarios para el funcionamiento de un dispositivo protésico, que es el fin último de este trabajo de investigación.

Referencias

[1] Y. L. Lee, P. K. Tsung, and M. Wu, "Techology trend of edge AI," in 2018 International Symposium on VLSI Design, Automation and Test, VLSI-DAT 2018, 2018, pp. 1–2.

[2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge

Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," Proc. IEEE, vol. 107, no. 8, 2019.

[3] S. Cerutti and C. Marchesi, Advanced Methods of Biomedical Signal Processing. IEEE Press, 2011.

[4] P. Konrad, The abc of emg, no. April. 2005.

[5] J. Kimura, "Electrodiagnosis in Diseases of Nerve and Muscle," p. 1177, 2013.

[6] D. Stegeman and H. Hermens, "Standards for surface electromyography: The European project Surface EMG for non-invasive assessment of muscles (SENIAM)," SENIAM, pp. 108–112, 2007.

[7] S. W. Smith, The scientist and engineer's guide to digital signal processing, 2nd ed. San Diego: California Technical Publishing, 1997.

[8] A. Oppenheim, R. Schafer, and J. Buck, Discrete-Time Signal Processing, 2nd ed. New Jersey: Prentice Hall Press, 1998.

[9] Y. H. Hsueh, C. Yin, and Y. H. Chen, "Hardware System for Real-Time EMG Signal Acquisition and Separation Processing during Electrical Stimulation," J. Med. Syst., vol. 39, no. 9, 2015.

[10] G. M. Hägg, B. Melin, and R. Kadefors, "Applications in Ergonomics," in Electromyography, R. Merletti and P. Parker, Eds. IEEE Press, 2005, pp. 343–363.

[11] R. M. Rangayyan, Biomedical Signal Analysis, 2nd ed. IEEE Press, 2015.

[12] R. Merletti and D. Farina, Eds., Surface Electromyography. IEEE Press, 2016.

[13] H. Tankisi et al., "Standards of instrumentation of EMG," Clin. Neurophysiol., vol. 131, no. 1, pp. 243–258, 2020.

[14] T. Roland, S. Amsuess, M. F. Russold, and W. Baumgartner, "Ultra-low-power digital filtering for insulated EMG sensing," Sensors (Switzerland), vol. 19, no. 4, pp. 1–24, 2019.

[15] L. Rozaqi, A. Nugroho, K. H. Sanjaya, and A. Ivonita Simbolon, "Design of Analog and Digital Filter of Electromyography," Proceeding - 2019 Int. Conf. Sustain. Energy Eng. Appl. Innov. Technol. Towar. Energy Resilience, ICSEEA 2019, pp. 186–192, 2019.

[16] A. K. Sahu and A. K. Sahu, "A Review on Different Filter Design Techniques and Topologies for Bio-potential Signal Acquisition Systems," Proc. 3rd Int. Conf. Commun. Electron. Syst. ICCES 2018, no. Icces, pp. 934–937, 2018.

[17] B. Kosko and S. S. Haykin, Eds., Intelligent Signal Processing, vol. 81, no. 12. IEEE Press, 2001.

[18] C. Salamea Palacios and S. Luna Romero, "Calibración automática en filtros adaptativos para el procesamiento de señales EMG," Rev. Iberoam. Autom. e Inform. Ind., vol. 16, pp. 1–6, 2011.

[19] T. Yu, K. Akhmadeev, E. Le Carpentier, Y. Aoustin, and D. Farina, "On-line recursive decomposition of intramuscular EMG signals using GPU-implemented Bayesian filtering.," IEEE Trans. Biomed. Eng., 2019.

[20] S. Márquez-Figueroa, Y. S. Shmaliy, and O. Ibarra-Manzano, "Optimal extraction of EMG signal envelope and artifacts removal assuming colored measurement noise," Biomed. Signal Process. Control, vol. 57, p. 101679, 2020.

[21] P. Okoniewski, S. Kocon, and J. Piskorowski, "Linear Time-Varying Multi-Notch FIR Filter for Fast EMG Measurements," 2018 23rd Int. Conf. Methods Model. Autom. Robot. MMAR 2018, pp. 634–637, 2018.

[22] I. F. Ghalyan, Z. M. Abouelenin, and V. Kapila, "Gaussian Filtering of EMG Signals for Improved Hand Gesture Classification," 2018 IEEE Signal Process. Med. Biol. Symp. SPMB 2018 - Proc., pp. 1–6, 2019.

[23] A. R. Verma, Y. Singh, and B. Gupta, "Adaptive filtering method for EMG signal using bounded range artificial bee colony algorithm," Biomed. Eng. Lett., vol. 8, no. 2, pp. 231–238, 2018.

[24] M. Z. Jamal, D. H. Lee, and D. J. Hyun, "Real Time Adaptive Filter based EMG Signal Processing and Instrumentation Scheme for Robust Signal Acquisition Using Dry EMG Electrodes," 2019 16th Int. Conf. Ubiquitous Robot. UR 2019, pp. 683–688, 2019.

[25] L. L. Menegaldo, "Real-time muscle state estimation from EMG signals during isometric contractions using Kalman filters," Biol. Cybern., vol. 111, no. 5–6, pp. 335–346, 2017.

[26] E. Kasaeyan Naeini, S. Shahhosseini, A. Subramanian, T. Yin, A. M. Rahmani, and N. Dutt, "An Edge-Assisted and Smart System for Real-Time Pain Monitoring," Proc. - 4th IEEE/ACM Conf. Connect. Heal. Appl. Syst. Eng. Technol. CHASE 2019, pp. 47–52, 2019.

[27] S. Tam, M. Boukadoum, A. Campeau-Lecours, and B. Gosselin, "A Fully Embedded Adaptive Real-Time Hand Gesture Classifier Leveraging HD-sEMG and Deep Learning," IEEE Trans. Biomed. Circuits Syst., vol. 14, no. 2, pp. 232–243, 2020.

[28] W. H. Mahmoud and N. Zhang, "Software/hardware implementation of an adaptive noise cancellation system," ASEE Annu. Conf. Expo. Conf. Proc., 2013.

[29] P. S. R. Diniz, Adaptive Filtering. Boston, MA: Springer US, 2008.

[30] M. Limem, M. A. Hamdi, and M. A. Maaref, "Denoising uterine EMG signals using LMS and RLS adaptive algorithms," in 2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), 2016, pp. 273–276.

[31] A. C. Mugdha, F. S. Rawnaque, and M. U. Ahmed, "A study of recursive least squares (RLS) adaptive filter algorithm in noise removal from ECG signals," in 2015 International Conference on Informatics, Electronics & Vision (ICIEV), 2015, pp. 1–6.

[32] P. Singh, K. Bhole, and A. Sharma, "Adaptive Filtration Techniques for Impulsive Noise Removal from ECG," in 2017 14th IEEE India Council International Conference (INDICON), 2017, pp. 1–4.

[33] P. M. Corcoran, "Mapping home-network appliances to tcp/ip sockets using a three-tiered home gateway architecture," IEEE Trans. Consum. Electron., vol. 44, no. 3, pp. 729–736, 1998.

[34] R. L. Maata, R. Cordova, B. Sudramurthy, and A. Halibas, "Design and Implementation of Client-Server Based Application Using Socket Programming in a Distributed Computing Environment," in 2017 IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2017, 2018.