

# **SISTEMA EMBEBIDO DESTINADO A LA DETECCIÓN Y ELIMINACIÓN DE ARTEFACTOS EN SEÑALES EEG PARA APLICACIONES MÉDICAS.**

Cesar Augusto Lozada Tenorio  
Bolívar Daniel David Medina.



UNIVERSIDAD DE SAN BUENAVENTURA CALI  
FACULTAD DE INGENIERÍA  
PROGRAMA DE INGENIERÍA ELECTRÓNICA  
CALI, 2015

# **SISTEMA EMBEBIDO DESTINADO A LA DETECCIÓN Y ELIMINACIÓN DE ARTEFACTOS EN SEÑALES EEG PARA APLICACIONES MÉDICAS.**

Cesar Augusto Lozada Tenorio  
Bolívar Daniel David Medina.

## **Informe Final**

**Director**  
**Ing. José Fernando Valencia**



**UNIVERSIDAD DE SAN BUENAVENTURA CALI**  
**FACULTAD DE INGENIERÍA**  
**PROGRAMA DE INGENIERÍA ELECTRÓNICA**  
**CALI, 2015**

Este trabajo de grado, en la modalidad de proyecto de investigación es aceptado como uno de los requisitos para obtener el título de Ingeniero Electrónico en la Universidad de San Buenaventura Cali.

---

Ing. Jose Fernando Valencia, PhD

---

Ing. Juan Carlos Cruz Ardila

---

Ing. Oscar Casas García

Santiago de Cali, 14 de Octubre del 2015

## **Dedicatoria**

Esta tesis se la dedicamos a Dios quién supo guiarnos por el buen camino, darnos fuerzas para seguir adelante y superar los problemas que se presentaban, enseñándonos a encarar las adversidades sin perder nunca la dignidad ni desfallecer en el intento.

A mi familia quienes les debemos todo lo que somos hoy. Para nuestros padres por su apoyo, consejos, comprensión, amor, ayuda en los momentos difíciles, y por ayudarnos con los recursos necesarios para estudiar. A nuestros hermanos por estar siempre presentes, acompañándonos para podernos realizar.

A nuestros maestros, Ing. Jose Fernando Valencia por su gran apoyo y motivación para la culminación de nuestros estudios profesionales y para la elaboración de esta tesis; Al Ing. Daniel Valencia por su apoyo ofrecido en este trabajo, por su tiempo compartido y por impulsar el desarrollo de nuestra formación profesional.

## **CONTENIDO**

Capítulo 1.....	1
1.1    Resumen .....	1
1.2    Definición del Problema .....	1
1.3    Justificación .....	3
1.4    Objetivos.....	6
1.4.1    Objetivo General.....	6
1.4.2    Objetivos Específicos .....	6
1.5    Marco de Referencia .....	7
1.5.1    Origen y Naturaleza de las Señales EEG .....	7
1.5.2    Registros EEG.....	8
1.5.3    Propósitos del EEG.....	9
1.5.4    Ritmos EEG.....	10
1.5.5    Artefactos .....	13
1.5.6    Técnicas para Detección de Artefactos .....	17
1.5    Estructura del Documento .....	23
Capítulo 2.....	25
2.1    Selección de Métodos para la Detección de Artefactos .....	25
2.2    Descripción de métodos para eliminación de artefactos.....	27
2.2.1    Métodos de Detección de artefactos por análisis en el dominio del tiempo.	27
2.3    Descripción Y Análisis de la Base de Datos.....	31
2.3.1    Acondicionamiento de la señal .....	32
2.3.2    Formas de onda .....	33
Capítulo 3.....	39
3.1    Implementación de Métodos en Matlab.....	39
3.1.1    MatLab como herramienta de simulación .....	39
3.1.2    Diseño de la interfaz gráfica en MatLab .....	40
3.1.3    Metodología para la implementación y simulación de los algoritmos para la detección de artefactos en la herramienta MatLab.....	41
3.1.4    Resultados .....	48
3.1.5    Resultados de la Implementación de Métodos en MatLab .....	50
3.1.6    Implementación de Métodos en Python.....	67
3.1.7    Diseño de la Interfaz Gráfica en Python.....	68
3.1.8    Resultados de la Implementación de Métodos en Python .....	69
Capítulo 4.....	80
4.1    Selección del Sistema Embebido .....	80
4.2    Descripción Raspberry Pi Modelo B .....	82
4.3    Acondicionamiento de la Raspberry .....	83
4.4    Implementación de Métodos en el Sistema Embebido: Raspberry modelo B ....	84

4.5 Resultados .....	86
4.5.1 Resultados Simulados de Threshold de Amplitud en la Herramienta Python® Raspberry.....	86
4.5.2 Resultados Simulados de Threshold de Pendiente en la Herramienta Python® Raspberry.....	90
4.5.3 Resultados Método de Correlación Implementado en Raspberry.....	94
4.5.4 Comparación de Resultados Matlab, Python y Raspberry .....	96
Capítulo 5.....	101
5.1 Conclusiones.....	101
5.2 Recomendaciones.....	104
LISTA DE Referencias .....	128

## LISTA DE TABLAS

Tabla 1: Selección de técnicas para detección de artefactos.....	25
Tabla 2: Clasificación de resultados.....	48
Tabla 3: Resultados threshold de amplitud en MatLab .....	52
Tabla 4: Resultados método de threshold de amplitud en MatLab.....	53
Tabla 5: Clasificación de resultados threshold de amplitud en MatLab .....	54
Tabla 6: Resultados evaluación método de threshold de amplitud.....	54
Tabla 7: Resultados threshold de amplitud en MatLab .....	56
Tabla 8: Resultados threshold de pendiente en MatLab .....	59
Tabla 9: Resultados detección de artefactos método threshold de amplitud en MatLab ..	60
Tabla 10: Clasificación de resultados método threshold de amplitud en MatLab .....	60
Tabla 11: Resultados método threshold de pendiente en MatLab valor de threshold de 4.0 .....	62
Tabla 12: Resultados evaluación de registros con método de correlación.....	63
Tabla 13: Clasificación de resultados método de correlación en MatLab. ....	64
Tabla 14: Resultados de la evaluación método de correlación.....	65
Tabla 15: Resultados método threshold de amplitud en Python .....	71
Tabla 16: Resultados test de validez aplicado al método de threshold de amplitud en Python.....	72
Tabla 17: Clasificación de resultados método de threshold de amplitud en Python .....	73
Tabla 18: Resultados de la evaluación del método de threshold de amplitud en Python. ....	73
Tabla 19: Resultados método threshold de pendiente en Python .....	74
Tabla 20: Resultados test de validez aplicado al método de threshold de pendiente en Python.....	76
Tabla 21: Clasificación de resultados método de threshold de pendiente en Python .....	77
Tabla 22: Resultados de la evaluación del método de threshold de pendiente en Python. ....	77
Tabla 23: Resultados test de validez aplicado al método de correlación en Python.....	77
Tabla 24: Comparación entre sistemas embebidos .....	81
Tabla 25: Resultados método de Threshold de Amplitud Raspberry.....	87
Tabla 26: Resultados test de Validez aplicado al método de Threshold de Amplitud .....	89
Tabla 27: Clasificacion de resultados método de Threshold de Amplitud Raspberry .....	90
Tabla 28: Resultados de la evaluación del método de threshold de Amplitud en Raspberry. ....	90
Tabla 29: Resultados método Threshold de Pendiente en Rapsberry .....	91
Tabla 30: Resultados test de Validez aplicado al método Threshold de Pendiente en Raspberry.....	93
Tabla 31: Clasificación de resultados de Threshold de Pendiente en Raspberry.....	94
Tabla 32: Resultados de la evaluación del método de threshold de pendiente en Raspberry.....	94
Tabla 33: Resultados test de validez aplicado al método correlación en Raspberry.....	94
Tabla 34: Comparación de resultados de detección método Threshold de Amplitud .....	97

Tabla 35: Comparación de test de validez método Threshold de Amplitud.....	98
Tabla 36: Comparación de resultados de detección método Threshold de Pendiente.....	99
Tabla 37: Comparación de test de validez método Threshold de Pendiente .....	100
Tabla 38: Comparación de test de validez método correlación.....	100

## LISTA DE FIGURAS

Figura 1: Adquisición de señal EEG.....	7
Figura 2: Tipos de electrodos para EEG .....	8
Figura 3: Representación esquemática del sistema 10-20.....	9
Figura 4: Ritmos Delta .....	11
Figura 5: Ritmos Theta.....	11
Figura 6: Ritmos Alfa.....	12
Figura 7: Ritmos Beta.....	12
Figura 8: Ritmos Gamma .....	13
Figura 9: Superposición Perfiles Anatómicos EEG .....	14
Figura 10: Artefacto EEG en tiempo y espectro de frecuencia .....	14
Figura 11: Artefacto EOG.....	15
Figura 12: Espectro de frecuencia artefacto EOG (rojo) vs EEG normal .....	16
Figura 13: Representación dipolo formado por la córnea y la retina. ....	16
Figura 14: Señal pico en EEG .....	18
Figura 15: Flujo aplicación de filtros digitales .....	19
Figura 16: Diagrama de bloque filtro óptimo .....	22
Figura 17: Artefacto presente en la señal EEG referenciado a través de anormalidad en la amplitud.....	28
Figura 18: Actividad rápida o actividad muscular presente en la señal EEG y detectable a través de los umbrales de pendiente .....	28
Figura 19: Respuesta en frecuencia filtro notch .....	30
Figura 20: Monitor A2000 .....	31
Figura 21: Posicionamiento de electrodos según el fabricante .....	31
Figura 22: a) Epoch Crudo b) Epoch Escalado .....	32
Figura 23: Señal EEG Final .....	33
Figura 24: Ritmos EEG.....	34
Figura 25: Artefacto Muscular .....	34
Figura 26: Artefacto Ocular .....	35
Figura 27: Artefacto 50 Hz .....	35
Figura 28: Interfaz y Logo de MatLab .....	39
Figura 29: Captura de pantalla de la interfaz gráfica implementada en MatLab .....	41
Figura 30: Ventana deslizante .....	42
Figura 31: Ecuación método umbral de amplitud .....	43
Figura 32: Ventana deslizante método de la pendiente .....	44
Figura 33: Ecuación método de la pendiente.....	44
Figura 34: Flujo para la selección del patrón.....	45
Figura 35: Grafica duración artefactos EOG .....	45
Figura 36: Desviación estándar artefactos EOG .....	46
<b>Figura 37: Patrón final método correlación .....</b>	<b>47</b>
Figura 38: Índices de correlación.....	47

Figura 39: Detección de artefactos a través del método threshold de amplitud con un threshold de 6.....	51
Figura 40: Ajuste para el método de amplitud.....	55
Figura 41: Detección de artefactos a través del método threshold de amplitud con un threshold de 3.5.....	57
Figura 42: Parámetros iniciales método de la pendiente .....	58
Figura 43: Detección de artefactos a través del método threshold de Pendiente con un threshold de 6.....	58
Figura 44: Ajuste para el método de pendiente.....	61
Figura 45: <i>Detección de artefactos a través del método threshold de Pendiente con un threshold de 4.0 .....</i>	63
Figura 46: a) Artefacto no detectado vs b) artefacto detectado por el método de correlación.....	66
Figura 47: Diferencia entre los umbrales de detección MatLab .....	66
Figura 48: Logo de Python .....	67
Figura 49: Interfaz Gráfica en Python.....	69
Figura 50: Detección de artefactos a través del método Threshold de Amplitud con un threshold de 3.5.....	70
Figura 51: Detección de artefactos a través del método Threshold de pendiente con un Threshold de 4.0 .....	75
Figura 52: a) Artefacto no detectado vs b) artefacto detectado por el método de correlación.....	78
Figura 53: Diferencia entre los umbrales de detección Phyton .....	79
Figura 54: Hardware Raspberry Pi Modelo B .....	82
Figura 55: Selección de la unidad de almacenamiento.....	83
Figura 56: Selección de sistema operativo .....	84
Figura 57: Resultados método de threshold de pendiente en Raspberry .....	88
Figura 58: Resultados método threshold de pendiente en raspberry .....	92
Figura 59: a) Artefacto no detectado vs b) artefacto detectado por el método de correlación.....	95
Figura 60: Diferencia entre los umbrales de detección Phyton .....	96

## **LISTA DE ANEXOS**

Anexo 1: código de métodos implementados en Matlab .....	105
Anexo 2: código de interface diseñada en Python .....	116
Anexo 3: código de métodos implementados en Python .....	122

# CAPÍTULO 1

## 1.1 Resumen

El propósito de este trabajo es implementar y comparar en un sistema embebido algoritmos para detectar y reducir los artefactos presentes en señales EEG de un canal. Para esto se realizó una investigación de los algoritmos existentes para detectar y reducir artefactos en las señales EEG, se seleccionaron los algoritmos que podían ser aplicados a la base de datos disponible. Los métodos seleccionados son: umbral de amplitud, umbral de pendiente y correlación. Estos métodos fueron, en primera instancia, implementados en MATLAB, después de comprobar la validez de los métodos seleccionados, se resolvió implementar los algoritmos en un lenguaje de programación de código abierto, el lenguaje seleccionado fue Python. Se implementaron los algoritmos en esta herramienta, se verificaron resultados y se compararon con los resultados obtenidos en MATLAB. La ventaja de utilizar Python es que permite migrar a diferentes plataformas sin realizar modificaciones mayores al código fuente. El objetivo final de este trabajo es implementar en un sistema embebido, en este caso una Raspberry modelo B, esto se logró programando la raspberry en lenguaje Python y diseñando una interfaz gráfica en PyQt. En conclusión se puede decir que al aplicar las pruebas de verificación de validez a algoritmos, en cada una de las herramientas de simulación se comprobó que los algoritmos son altamente sensibles a los cambios de amplitud súbitos, tales como los artefactos oculares, pero no es muy sensible en la detección de artefactos de baja amplitud. A pesar de que los algoritmos muestran resultados aceptables, no reemplazan la inspección de la señal EEG por parte de un especialista.

## 1.2 Definición del Problema

Uno de los retos actuales en la medicina, y en la ingeniería, es el monitoreo de profundidad de anestesia en los pacientes. La profundidad de la anestesia es una construcción teórica para conceptualizar los efectos de la anestesia en el sistema nervioso central como fases o estados discretos o continuos. Mientras que un nivel de anestesia demasiado alto puede resultar en depresión cardiovascular y un tiempo de despertar prolongado, un nivel de anestesia demasiado bajo es más aterrador desde el punto de vista de los pacientes quienes pueden llegar a sentir dolor o verse afectados sicológicamente (Röpcke, 2004). Es por esto que la evaluación precisa de la profundidad de la anestesia puede contribuir al ajuste de la administración del fármaco en cada paciente individualmente.

La monitorización del paciente es un elemento clave en la prestación de cuidados durante la anestesia. La palabra "monitor" se deriva del verbo latino "monere" que significa "para advertir". Como indica su derivación, un monitor sólo puede advertir. No hay ningún dispositivo mecánico o eléctrico que pueda sustituir a la cuidadosa observación del

paciente por parte del anestesista. La información que provee un equipo de monitoreo siempre requiere de interpretación clínica. El propósito de un dispositivo de monitoreo es indicar tendencias de cambio en las variables fisiológicas inducidas por la anestesia, cirugía, o pacientes con una enfermedad subyacente, es decir de monitorización fisiológica, permitiendo así, que se adopte una acción terapéutica adecuada. Además, la monitorización también debe detectar problemas potencialmente peligrosos tan pronto como sea posible, permitiendo la corrección en el momento oportuno, es decir, control de la seguridad (Fuchs-Buder, 2003).

Tradicionalmente, la profundidad de la anestesia se ha evaluado a partir de las respuestas autónomas y de movimiento, centrando esfuerzos en el mantenimiento de la estabilidad cardiovascular junto con la inmovilidad. Sin embargo, se ha demostrado que el movimiento en un sujeto no paralizado durante la anestesia representa un reflejo medular, es decir una respuesta motriz de tipo involuntaria que ocurre inmediatamente después de aplicar un estímulo en particular (IJ, 1994). La inmovilidad es fácil de lograr con los agentes bloqueadores neuromusculares (NBMA). Cuando se utilizan NBMA, ni la inmovilidad ni la estabilidad cardiovascular pueden ser considerados para representar la depresión o la presencia de funciones corticales tales como la conciencia y la memoria. Anteriormente, el nivel de sedación, es decir, el componente de sueño en anestesia equilibrada, estaba fuera del alcance de los equipos de monitorización (Vakkuri, 2006).

En la última década ha habido un aumento significativo en el número de estudios sobre el desarrollo, la comparación y la validación de los dispositivos comerciales que estiman la profundidad de la anestesia mediante el análisis de la actividad eléctrica del cerebro, utilizando para ello herramientas como el electroencefalograma (Klass, 2008). Un electroencefalograma o EEG es una exploración neurofisiológica que se basa en el registro de la actividad bioeléctrica cerebral en condiciones basales de reposo, en vigilia o sueño, y durante diversas activaciones (habitualmente hiperpnea y estimulación luminosa intermitente). La señal de EEG es generada básicamente por la suma de actividad eléctrica de distintas poblaciones neuronales, las cuales pueden generar potenciales eléctricos y magnéticos que pueden ser registrados a cierta distancia de sus fuentes de producción (a nivel de la superficie de la corteza cerebral mediante electrodos en el cuero cabelludo-EEG de superficie). Un EEG se puede utilizar para determinar si el nivel de alerta o de conciencia del paciente es normal, si existen anomalías en una parte específica del cerebro, si el paciente tiene una tendencia a tener ataques o convulsiones, y si es probable que tenga un tipo particular de epilepsia (Nunez & Srinivasan, 2006).

Sin embargo, cuando se adquiere una señal EEG, utilizando medios no invasivos y en un ambiente no controlado, se suman señales que no provienen específicamente de la actividad cerebral. Estas señales se pueden considerar como ruido y son llamadas artefactos. Los artefactos son los enemigos más importantes de señales de EEG de alta calidad y por lo tanto su filtrado es crucial para la evaluación precisa de la señal del EEG (Klass, 2008). Se clasifican en dos categorías principales: los artefactos técnicos y

fisiológicos. Los artefactos fisiológicos más frecuentemente observados se deben a la actividad ocular, cardiaca o muscular. Entre ellos, los artefactos de la actividad ocular son los más problemáticos. Dentro de los artefactos técnicos, es decir, los que son de origen extra fisiológico se destacan los generados por los electrodos, los instrumentos y el artefacto de 60Hz.

Para el proyecto propuesto los artefactos son indeseables y deben ser eliminados por varias razones: primero, los artefactos pueden estar presentes en todas partes, en cada trazo del EEG, y si son prominentes, pueden enmascarar la actividad del EEG y hacer que no se pueda interpretar el resultado; segundo, los artefactos pueden imitar casi cualquier tipo de actividad eléctrica cerebral y llevar a una mala interpretación que puede resultar grave; por último, los artefactos pueden conducir a conclusiones falsas cuando el EEG se transforma por medio de técnicas tales como el análisis espectral de potencia, a menos que se tenga mucho cuidado para reconocer y excluir actividades causadas por ellos a partir de los datos procesados.

La base de datos clínicos disponible para este proyecto contiene señales EEG de un solo canal registradas en pacientes bajo sedación en procedimientos de endoscopia. Las condiciones bajo las cuales se registran estas señales traen varios retos adicionales. La mayoría de las técnicas que se utilizan para eliminar artefactos, están diseñadas para señales EEG multicanal, como es el caso de las técnicas que utilizan separación ciega de fuentes o BSS, las cuales emplean canales separados para la señal EEG y EOG, para después procesarlas. Estas técnicas no se pueden aplicar directamente a señales EEG de canal simple. Por otra parte, los pacientes en sedación están mucho más activos que los pacientes bajo anestesia general, esto provoca que la señal registrada bajo sedación esté más contaminada por artefactos procedentes del movimiento muscular, facial y ocular. Por último, al tener menor cantidad de información, la eliminación de las etapas de la señal contaminadas con artefactos resultaría inaceptable, debido a que se pueden perder gran cantidad de datos en este proceso.

Para cumplir con el objetivo de este trabajo se debe responder satisfactoriamente a la siguiente pregunta: ¿Cuál o cuáles de las técnicas propuestas para la eliminación de los artefactos en señales EEG multicanal presentan mayor eficacia y nivel de confiabilidad al ser utilizada para eliminar los artefactos de una señal EEG de un solo canal?

### **1.3 Justificación**

En cualquier tipo de procedimiento quirúrgico es sumamente importante saber la cantidad y el tipo de anestesia a suministrar la cual puede variar según el tipo de procedimiento, complejidad del mismo o las características de cada paciente. Un exceso en cuanto a la cantidad de anestesia aplicada puede generar repercusiones temporales o

permanentes en cada persona. Al mismo tiempo una cantidad insuficiente puede generar incomodidad y dolor al paciente durante o después del procedimiento. Fuera de lo anteriormente mencionado, es también indispensable seguir un proceso de monitoreo y análisis posteriormente a la anestesia y durante el procedimiento quirúrgico, pues conocer a cada instante el nivel de sedación y analgesia de la persona al igual que sus reacciones propiciaran mayor seguridad, comodidad, decremento de los índices de riesgo y dolor al paciente y un incremento de los índices confiabilidad y veracidad en diagnósticos de anestesiología, mejorando a su vez la calidad del servicio en todos los aspectos. Por lo tanto, es importante conocer el nivel de sedación y analgesia en cada paciente para poder ejercer el monitoreo constante del mismo (Jones, 2012).

El desarrollo de la tecnología de equipos médicos capaces de registrar la actividad eléctrica cerebral (EEG) en una señal continua, plantea nuevas alternativas para el control de variables farmacológicas en procedimientos médicos, en este caso nivel de sedación y analgesia de un paciente. Estos niveles de sedación se usan en situaciones médicas en conjunción con una historia clínica en la evaluación del grado de aplicación de la sedación en pacientes con el fin de evitar, por un lado, un bajo grado de sedación, que puede hacer experimentar al paciente dolor o angustia, y por otro lado, un grado de sedación excesivo, pues expone al paciente a efectos secundarios tales como la supresión de la respiración o hipoxia.

Para recoger la señal eléctrica cerebral se utilizan normalmente electrodos (EEG de superficie), a los que se añade una pasta conductora para facilitar que la señal eléctrica cerebral (magnitud de  $\mu$ -voltios) se pueda registrar y analizar (Corral-Fernández, 2007). Cada electrodo es un punto de registro, sin embargo, para poder realizar este registro es preciso disponer de dos terminales, es decir dos electrodos, los que forman un canal. La «American Electroencephalographic Society» recomienda el uso de 16 canales como mínimo, sin embargo pueden emplearse canales adicionales para registrar otras funciones biológicas como ECG, movimientos oculares, respiración, EMG, entre otros. Los nombres y la localización de los electrodos de cada canal son especificados por el sistema internacional 10-20, para la mayoría de aplicaciones clínicas y de investigación (H, 1958).

El tipo de señales EEG que se van a analizar en este proyecto son señales de un solo canal, registrado en la frente, lo cual sugiere que las señales están más contaminadas por ruido, ruido que es denominado como artefactos, una señal EEG nunca está libre de ellos y todos los artefactos varían según las molestias de cada paciente puesto que los artefactos que se presentan son causados principalmente por contracciones musculares (EMG), movimientos del cuerpo causados por la circulación de la sangre, movimientos oculares, movimientos faciales, etc. Comprendiendo esto es preciso concluir cuán necesario y obligatorio es el debido procesamiento de la señal EEG que implica la aplicación de filtros para eliminar dichas mediciones no deseadas o artefactos, ya que evidentemente no será posible realizar un diagnóstico válido en cuanto al nivel de sedación con una señal eléctrica cerebral directamente obtenida.

Para poder interpretar las señales EEG y a partir de ellas extraer variables que se puedan relacionar con el nivel de sedación y analgesia, se debe asegurar que la señal sea lo

suficientemente confiable para que la probabilidad de un diagnóstico erróneo sea lo más reducido posible. Esto se logra eliminando los artefactos anteriormente mencionados que están presentes en la señal EEG original, para este fin se utilizan diversos métodos de filtrado, tales como el filtrado adaptativo o la regresión lineal.

En el presente proyecto se propone implementar en un sistema embebido varios de los métodos existentes para el filtrado de artefactos en señales EEG, y así comparar su eficiencia a partir de los resultados obtenidos, concluyendo cual o cuales de los métodos son más aplicables a las señales EEG de un canal registradas en la frente, durante el procedimiento de endoscopia digestiva. En la primera etapa del proyecto se construirá un estado del arte en cuanto a los métodos y algoritmos que se utilizan actualmente para la eliminación de artefactos en señales EEG. Posteriormente, se analizará cada algoritmo y se comparará su eficacia frente al objetivo establecido. Por último, se implementará el sistema embebido con una interface gráfica para visualizar la señal del EEG antes y después del filtrado viendo así, si la señal EEG obtenida con su debido procesamiento es lo suficientemente confiable para realizar un diagnóstico en cuanto al nivel de sedación de un paciente.

## **1.4 Objetivos**

### **1.4.1 Objetivo General**

Implementar y comparar en un sistema embebido algoritmos para reducir los artefactos presentes en señales EEG de un canal, registradas durante sedación en procedimientos de endoscopia.

### **1.4.2 Objetivos Específicos**

- Crear un marco de referencia en el que se describirán las técnicas desarrolladas en la actualidad para filtrar los artefactos de una señal EEG, y sobre el cual se fundamentara el desarrollo del proyecto y que a su vez será el soporte de los procesos metodológicos.
- Describir la base de datos que contiene los registros EEG de un solo canal en cuanto a los formatos, formas de onda, y otra información relevante para el desarrollo del proyecto.
- Analizar y comparar, por medio de una simulación, los algoritmos existentes para seleccionar los que mayor eficiencia presenten en cuanto a al rechazo de los artefactos en una señal EEG.
- Desarrollar un sistema embebido para la implementación de los algoritmos, seleccionados de visualización de las señales de entrada y de salida.
- Validar el sistema a partir de una comprobación de resultados y sacar conclusiones de cuál fue el o los algoritmos que mejores resultados presenten para cumplir con el objetivo principal de la investigación.

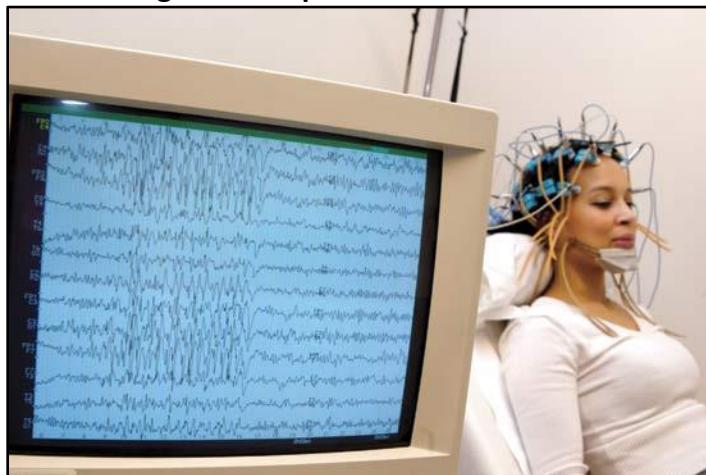
## 1.5 Marco de Referencia

### 1.5.1 Origen y Naturaleza de las Señales EEG

El médico inglés Richard Caton (1842-1926), un médico de Liverpool, presentó en 1875 sus hallazgos sobre los fenómenos bioeléctricos en los hemisferios cerebrales de ratones, perros y monos, expuestos por craneotomía.

Hans Berger (1873-1941), psiquiatra alemán, introdujo en 1930 el término de electroencefalograma luego de comprobar con ayuda de un aparato amplificador, conocido como electroencefalógrafo, la existencia de un potencial eléctrico en el cerebro humano; Este potencial eléctrico es conocido como EEG el cual es catalogado como una bioseñal, y como tal, presenta la particularidad de poderse medir, monitorear y de reflejar de alguna manera un estado biológico funcional. “En cualquier organismo, la bioseñal se toma a partir de alguna corriente producida por el cambio de potencial eléctrico sobre algún tejido, órgano o sistema celular” (Orozco & Suárez, 2009) comúnmente la señal bioeléctrica referida al cerebro o señal EEG se adquiere a través del cuero cabelludo.

**Figura 1: Adquisición de señal EEG**



Fuente: Enciclopedia Británica <http://www.britannica.com/science/electroencephalography/>

La electroencefalografía hace referencia a la exploración neurofisiológica de la actividad bioeléctrica de las neuronas en el encéfalo, dicho lo anterior el término “electroencefalograma” en sí, hace referencia al registro de la actividad bioeléctrica bien sea plasmada en papel o desplegada en una pantalla en la que se mide la magnitud de los pulsos eléctricos del cerebro, “los registros poseen formas muy complejas que varían mucho con la localización de los electrodos y entre individuos, Esto es debido al gran

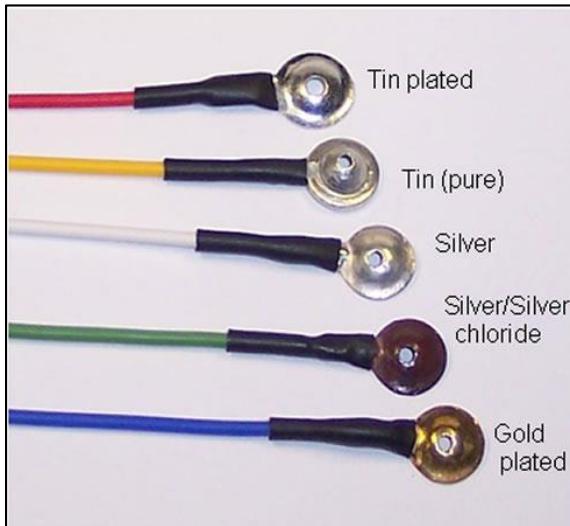
número de interconexiones que presentan las neuronas y por la estructura no uniforme del encéfalo” (Navarro).

### 1.5.2 Registros EEG

La adquisición de bioseñales eléctricas del cerebro o señal EEG se logra comúnmente mediante electrodos superficiales dispuestos en el cuero cabelludo del paciente al que se le aplica un gel conductor en el mismo para facilitar y favorecer el registro de la actividad cerebral.

Es habitual la presentación de los electrodos en pequeños parches, discos o copas como también es habitual el favoritismo por la utilización de electrodos Ag (plata)/AgCl (cloruro de plata); los electrodos diseñados para la adquisición de EEG o bien electrodos EEG transforman las corrientes iónicas procedentes del tejido cerebral en corrientes eléctricas que luego son amplificadas y filtradas. También existen y son de gran uso los electrodos de inserción o aguja que están específicamente diseñados para alcanzar el tejido muscular. En la Figura 2 se muestran varios tipos de electrodos usados para la adquisición de señales EEG.

**Figura 2: Tipos de electrodos para EEG**

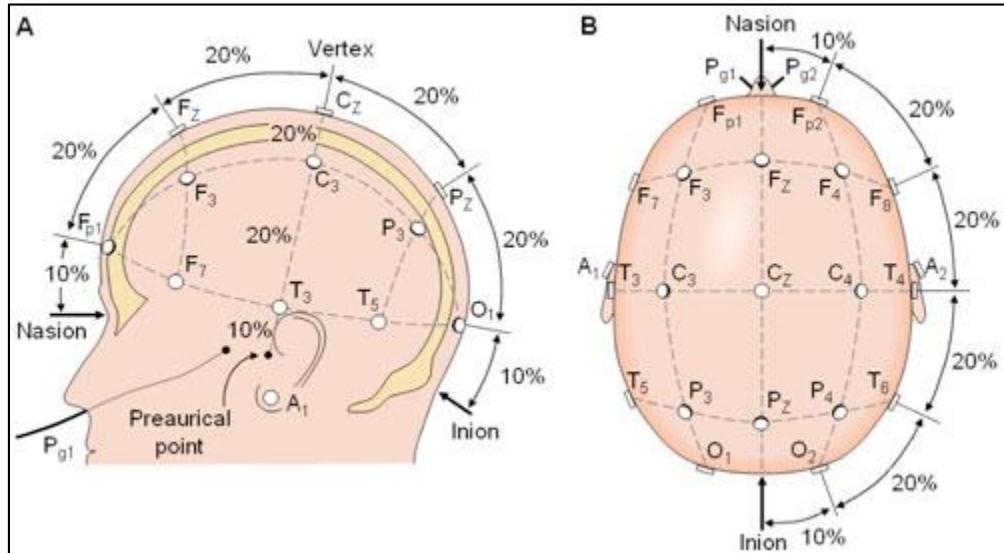


Fuente: <http://www.medifactory.nl/images/foto/eeg/foto01.jpg>

Las señales adquiridas a través de electrodos de superficie varían su fase, frecuencia y amplitud según las condiciones fisiológicas del paciente y las condiciones físicas presenciadas durante el procedimiento, como el tipo de electrodos y el canal de medición, que dicho de otra forma es la localización o ubicación de los electrodos. Aunque existen gran variedad de sistemas que indican las posiciones de los electrodos (Illinois, Montreal, Aird, Cohn, Lennox, Merlis, Oastaut, Schwab, Marshall, etc), la Federación Internacional de

Sociedades de Electroencefalografía y Neurofisiología Clínica recomendó el uso de un estándar en el posicionamiento de electrodos conocido como 10-20 (Wanchai, 2012) la cual consiste el posicionamiento de 19 electrodos activos sobre la corteza cerebral y uno más de referencia, como se muestra en la Figura 3.

**Figura 3: Representación esquemática del sistema 10-20**



Fuente: <http://www.bci2000.org/wiki/images/1/15/ElectrodePositions1020.PNG>

En general, la duración aproximada de un EEG es de 15 a 25 minutos y las señales presentan amplitudes que oscilan entre unos pocos micro voltios hasta aproximadamente 100  $\mu$ V y un contenido de frecuencia entre 0.5 a 40 Hz siendo adquiridas sobre la corteza craneal, si dichas señales son adquiridas directamente sobre la superficie del cerebro estas pueden llegar a ser 10 veces más intensas y presentar componentes en frecuencia entre 100Hz a 3Khz.

### 1.5.3 Propósitos del EEG

Los motivos más comunes que impulsan al estudio de la actividad cerebral de una persona son el control y diagnóstico de los trastornos convulsivos, el EEG también propicia la posibilidad los especialistas de analizar trastornos emocionales, cambios repentinos de comportamiento o problemas de sueño; El EEG no solo se utiliza con fines de detección o diagnósticos de trastornos sino que también permite llevar el control y la evolución de un paciente luego de una lesión craneal o el estado y reacción del paciente previo, durante y después de un procedimiento quirúrgico.

Los avances tecnológicos permiten que se obtengan hoy por hoy los registros de EEG en formato digital, permitiendo que cada vez sean más acertadas las valoraciones y los diagnósticos que se llevan a cabo por parte de los especialistas; gracias al registro en el mismo formato digital, “cada vez es más frecuente la aplicación de técnicas avanzadas de procesamiento digital de señales a dichos registros, logrando ejecutar de manera automática algunas de las labores realizadas por los especialistas y así aliviar la pesada carga que le representa el gran volumen de datos a procesar” (Arango & Naranjo, 2010). Estas son algunas de las aplicabilidades específicas del EEG:

- Detección de estado de alerta de vigilancia, coma y muerte cerebral.
- Localización de la zona de los daños tras una lesión de cabeza, derrame cerebral, y tumores.
- Ensayos de las vías aferentes (por potenciales evocados).
- Seguimiento de la participación cognitiva (Ritmo Alfa).
- Control de profundidad de anestesia (servo anestesia).
- Investigación de la epilepsia y la localización de origen de focos.
- Pruebas de los efectos de drogas en epilepsia.
- Asistencia para la extirpación experimental cortical de focos epilépticos.
- Seguimiento del desarrollo del cerebro.
- Ensayos sobre los efectos convulsivos de drogas.
- Investigación de los trastornos del sueño y la fisiología.
- Investigación de los trastornos mentales.
- Proporcionar un sistema de grabación de datos híbrido junto con otras modalidades de imágenes.

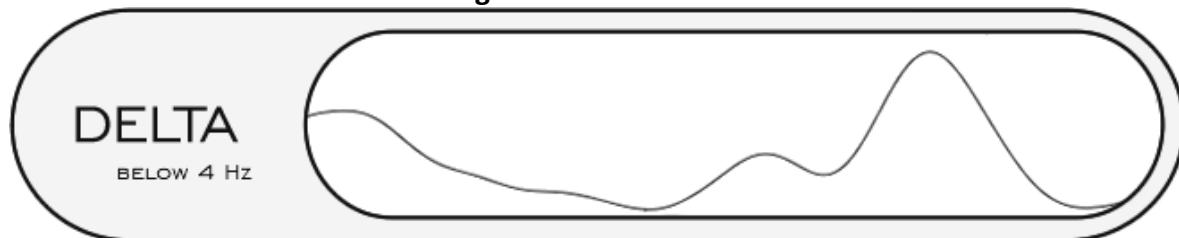
#### **1.5.4 Ritmos EEG**

De las características más importantes del registro EEG se encuentra el espectro de frecuencia de la señal, es de aquí donde se extrae la mayor información debido a que el comportamiento en frecuencia de la misma es el que más se ve involucrada y afectado cuando se presenta algún tipo de evento en el paciente como somnolencia, estrés, decepción, entre otros; las señales de EEG como se expresó con anterioridad presentan componentes en frecuencias desde 4 hasta 40 Hz dicho contenido de frecuencias está clasificado por bandas y se denominan ritmos EEG que permiten valorar de una manera más práctica y rápida el estado de una persona; los ritmos de EEG se clasifican de la siguiente manera:

### **Delta: Menores a 4 Hz**

Este ritmo de señal presente la mayor amplitud y la onda mucho más lenta de las demás. Principalmente se asocia con el sueño profundo o un grave trastorno cerebral.

**Figura 4: Ritmos Delta**

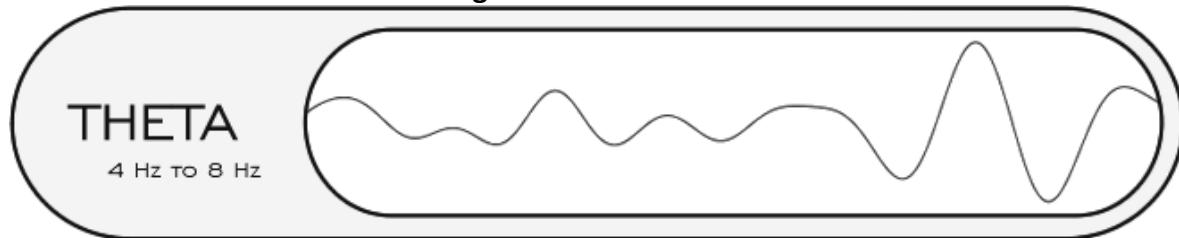


Fuente: <http://www.brainev.com/core/research-benefits/Brainwaves.aspx>

### **Theta: Entre 4 a 7 Hz**

Generalmente los ritmos theta son conocidos por presentar amplitudes que comúnmente sobrepasan los 20uV, este ritmo surge de la tensión emocional, sobre todo la frustración o decepción, inspiración o creatividad y meditación profunda.

**Figura 5: Ritmos Theta**

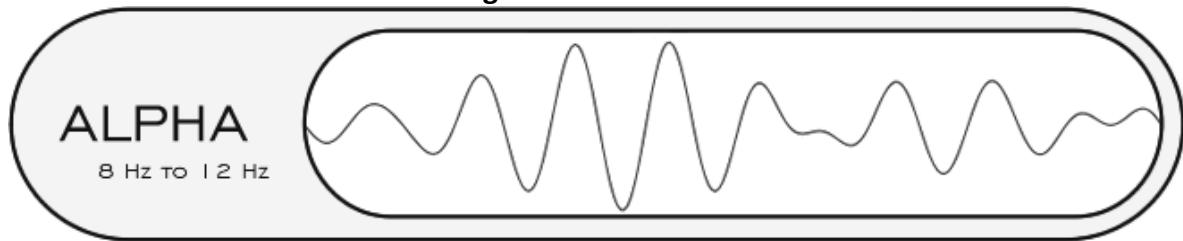


Fuente: <http://www.brainev.com/core/research-benefits/Brainwaves.aspx>

### **Alfa: Entre 8 a 13 Hz**

Los ritmos alfa suelen presentar amplitudes entre 30 y 50uV, y aparecen cuando el paciente tiene los ojos cerrados y atraviesa un estado de relajación; Por lo general se asocia con una intensa actividad mental, estrés y tensión; Abrir los ojos y enfocar la atención visual en objetos reduce las ondas de este tipo.

**Figura 6: Ritmos Alfa**

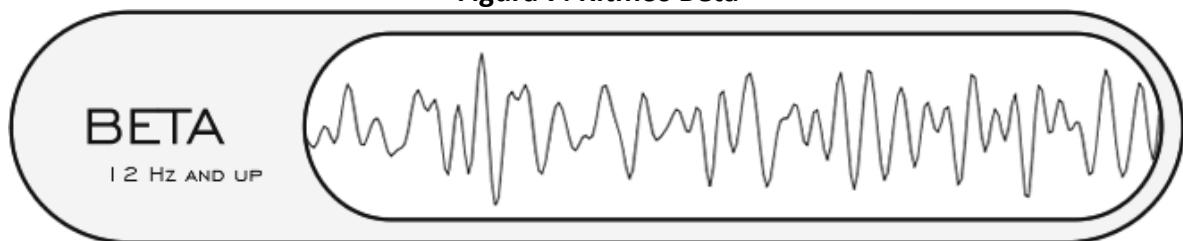


Fuente: <http://www.brainev.com/core/research-benefits/Brainwaves.aspx>

**Beta: Entre 14 a 30 Hz**

Los ritmos beta usualmente presentan amplitud baja y varían su frecuencia de una manera simétrica a ambos lados de la zona frontal; estos ritmos aparecen cuando el cerebro se encuentra activo y participa en actividades mentales, por lo general están asociadas a la resolución de problemas concretos y los eventos del mundo exterior.

**Figura 7: Ritmos Beta**



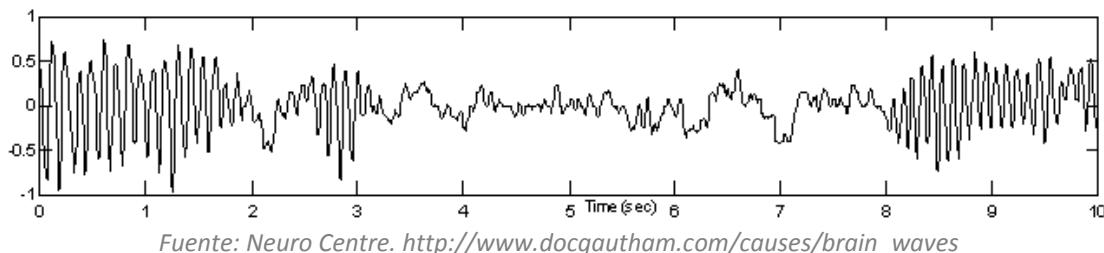
Fuente: <http://www.brainev.com/core/research-benefits/Brainwaves.aspx>

**Gamma: Mayores a 30 Hz**

La actividad Gamma es inferior a 2  $\mu$ V pico a pico y consiste en onda de baja amplitud y alta frecuencia como resultado de la fijación de la atención o estímulos sensoriales.

Los ritmos de baja amplitud y altas frecuencias reflejan un cerebro activo asociado con un estado de alerta o cuando se está soñando, mientras que las señales de gran amplitud y bajas frecuencias se asocian con estados de somnolencia o cuando se duerme sin la aparición de sueños. Esta relación es lógica, debido a que cuando la corteza está procesando información de forma más activa, ya sea por una entrada sensora o por algún proceso interno, el nivel de actividad de las neuronas corticales es relativamente alto, pero igualmente de sincronizado (Sörnmo & Laguna, 2005).

**Figura 8: Ritmos Gamma**



Fuente: Neuro Centre. [http://www.docgautham.com/causes/brain\\_waves](http://www.docgautham.com/causes/brain_waves)

### 1.5.5 Artefactos

Los artefactos son señales que se encuentran en el registro EEG, pero no se originan propiamente en el cerebro. En su libro (Fisch B. J., 1999) divide los artefactos en dos categorías dependiendo de su origen: artefactos fisiológicos y artefactos no fisiológicos.

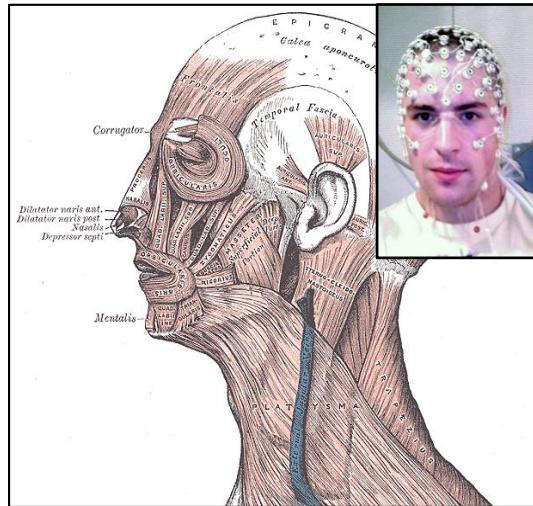
Los artefactos fisiológicos surgen de una variedad de actividades corporales, que pueden ser debidas a *movimientos*: movimiento de la cabeza, el cuerpo o el cuero cabelludo (ej. Pulsaciones de las arterias del cuero cabelludo), *potenciales bioeléctricos*: debidos a mover potenciales eléctricos dentro del cuerpo (tales como los producidos por el movimiento del ojo), o potenciales eléctricos generados por los músculos en el cuero cabelludo, corazón o glándulas sudoríparas, o *cambios en la resistencia de la piel*: debido a la actividad de las glándulas sudoríparas, transpiración y la actividad vasomotora.

Los artefactos no fisiológicos surgen de dos fuentes principales: *interferencia eléctrica externa* de fuentes de poder tales como líneas eléctricas y equipo eléctrico, y *mal funcionamiento eléctrico del equipo de grabación* provenientes de los electrodos, cables, amplificadores, entre otros.

#### **Artefacto Muscular**

La actividad eléctrica generada por los músculos peri craneales, que se filtra en una señal EEG, es llamada artefacto muscular o EMG. La contaminación por EMG es a menudo un problema en el registro del electroencefalograma, sobre todo, para aquellas aplicaciones tales como interfaces cerebro-ordenador que se basan en mediciones automáticas de características del EEG. La separación de dos señales biológicas, en este caso miogénica de neurogénica, depende del grado en que pueden ser discriminadas en una o más de estas dimensiones: temporal, anatómica, o espectral. Desafortunadamente, además de ocurrir al mismo tiempo, las señales EMG y EEG poseen amplia superposición de perfiles anatómicos y espectrales (Brenton W. McMenamin, 2011). La superposición del perfil anatómico se puede observar en la Figura 9.

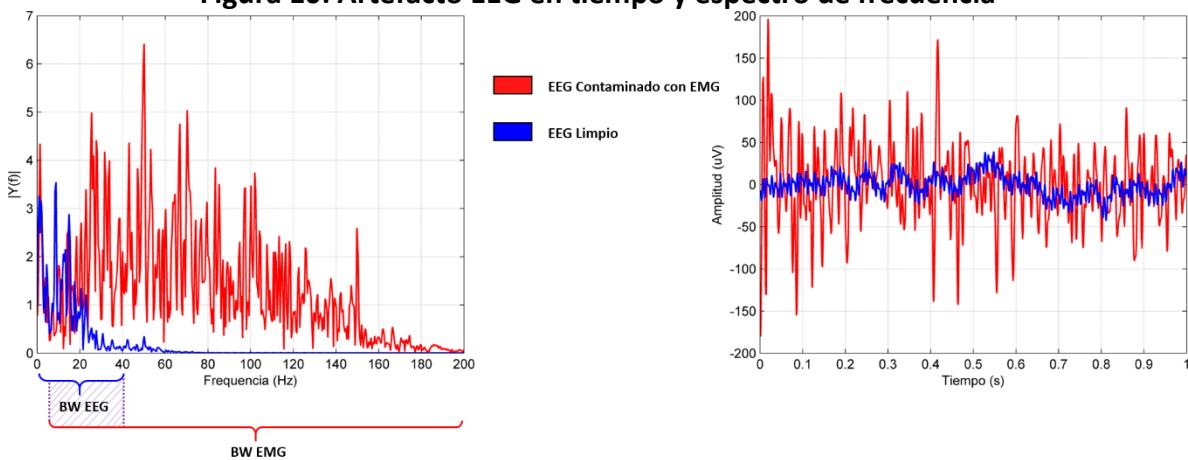
**Figura 9: Superposición Perfiles Anatómicos EEG**



Fuente: Lewis (1918) Gray's Anatomy 20th ed

En general, la amplitud de un EMG puede ser aproximadamente de 1 a 2 órdenes de magnitud mayor que la amplitud media de un EEG, por esta razón, incluso contracciones relativamente pequeñas de los músculos circundantes al cráneo, producen actividad EMG que puede ser detectada sobre todo el cuero cabelludo (Xinyi Yong, 2008). Se sabe que un EMG de los músculos esqueléticos registrados a partir de la piel tiene una distribución de frecuencia amplio desde 0 hasta más de 200 Hz con componentes espectrales distintas. Los EMG de los músculos faciales también tienen una distribución de frecuencia amplia y presentan picos de mayor actividad en las bandas 20-30Hz, también llamada ritmo EMG beta, y 35-60Hz.

**Figura 10: Artefacto EEG en tiempo y espectro de frecuencia**

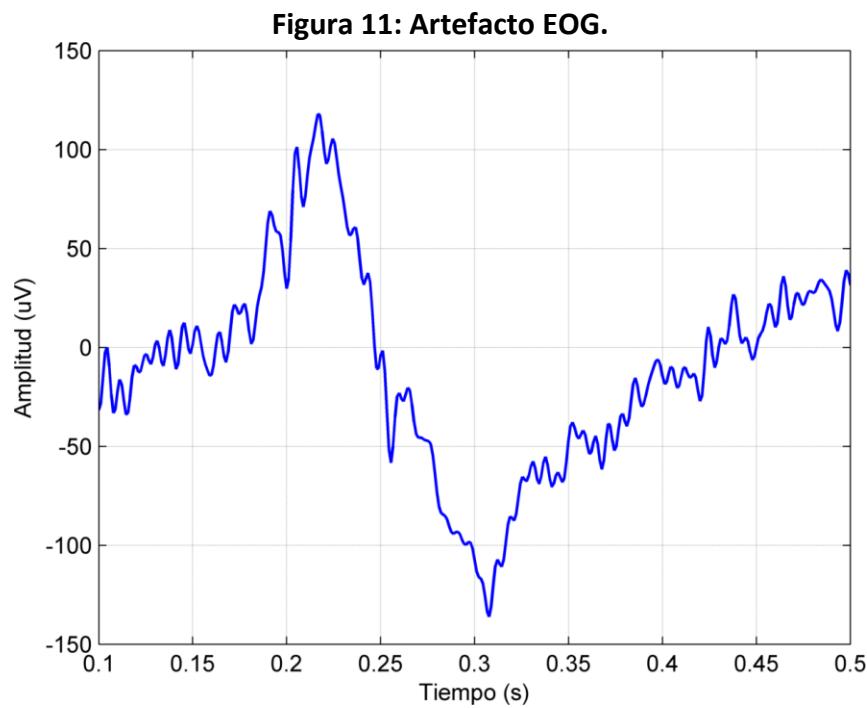


Fuente: De los Autores. Base de datos registro 041221HL.bin

En teoría, varios métodos pueden ser utilizados para reducir o eliminar la contaminación por EMG. Estos incluyen métodos relativamente simples, como filtros pasa bajas no lineales o lineales, el rechazo de los segmentos de EEG que superan un umbral de amplitud predefinida, y métodos más sofisticados como la descomposición en factores utilizando análisis componentes principales (PCA) o análisis de componentes independientes (ICA) (I.I. Goncharova, 2003). La aplicación exitosa de cualquiera de estos métodos requiere un conocimiento detallado de los patrones espectrales y topográficos de las señales EMG.

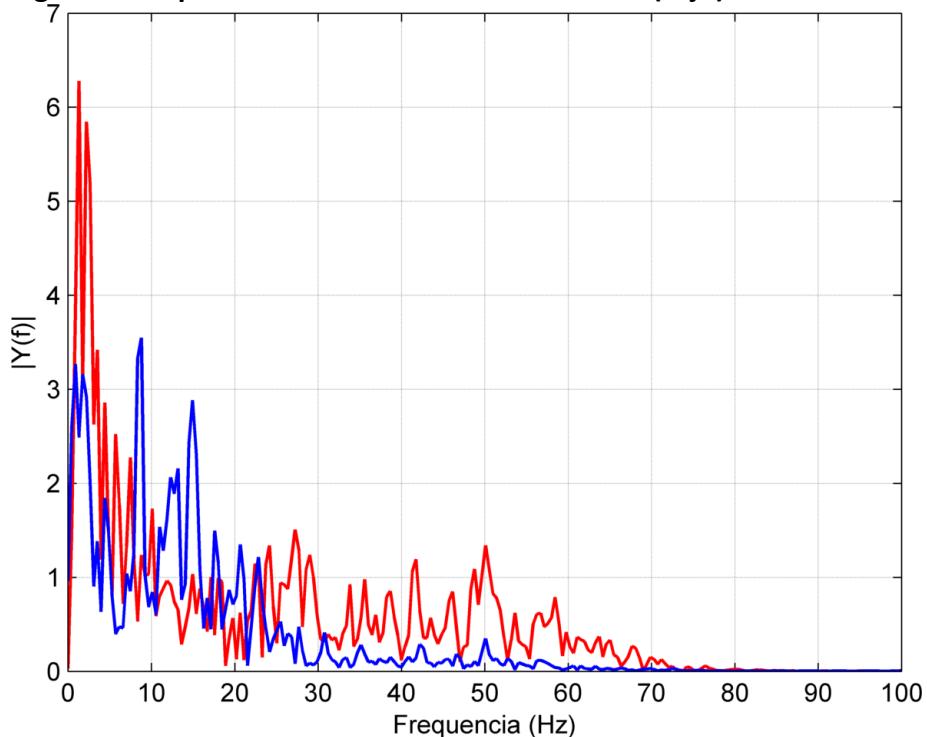
### **Artefacto Ocular EOG**

Este tipo de artefacto es producido por el parpadeo y otros movimientos del ojo. Estos movimientos causan una diferencia de potencial que es captada principalmente por los electrodos posicionados en la parte frontal y central del cráneo. El parpadeo se caracteriza por una señal de baja frecuencia ( $<4$  Hz) con una alta amplitud, por lo general es una señal simétrica que afecta principalmente a los electrodos frontales y tiene una baja propagación. Movimientos de los ojos también están representados por una señal de baja frecuencia ( $<4$  Hz), pero tienen una propagación mayor, especialmente, en el electrodo temporal (Fisch B. J., 1999).



Fuente: De los Autores. Base de datos artefactos oculares

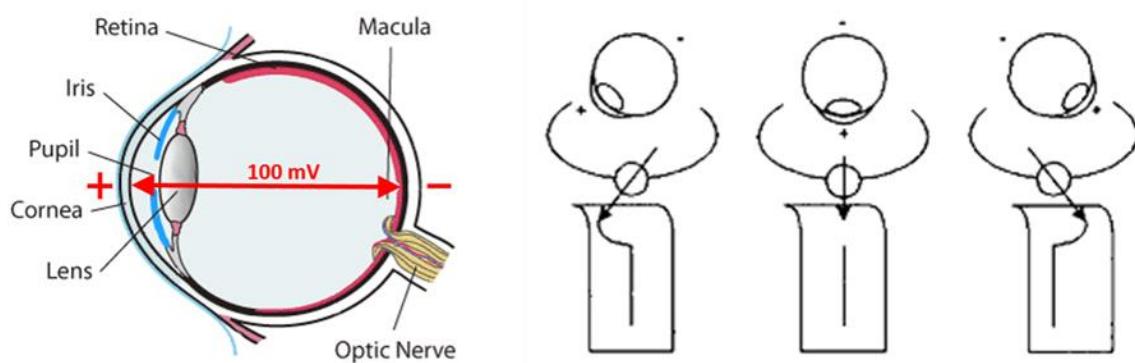
**Figura 12: Espectro de frecuencia artefacto EOG (rojo) vs EEG normal**



Fuente: De los Autores. Base de Datos artefactos oculares

Los artefactos producidos por movimientos oculares son causados por el hecho de que los ojos representan dipolos, donde la córnea es cerca de 100mV positiva con respecto a la retina, y el movimiento de este dipolo corneo-retina conduce a una alteración del campo eléctrico alrededor del ojo, tal como se muestra en la Figura 13.

**Figura 13: Representación dipolo formado por la córnea y la retina.**



Fuente: <http://www.johnopticians.co.uk/health-check/eye-anatomy/> (modificada)

Básicamente, los enfoques propuestos para hacer frente a las señales de EEG contaminadas con EOG se dividen en dos categorías principales, el rechazo o la corrección de los segmentos de señal contaminados. En los métodos que involucran el rechazo de los segmentos contaminados, los datos grabados son escaneados por un experto o por un algoritmo, los segmentos con artefactos serán excluidos de los datos (Eleni Kroupi, 2011). Los métodos convencionales para la corrección artefacto EOG incluyen proyección subespacial, como el análisis de componentes independientes (ICA) y el análisis de componentes principales (PCA), que descompone la señal en componentes independientes y no correlacionadas, respectivamente. Por otra parte, se han propuesto métodos basados en regresión para la corrección de artefactos. Por ejemplo, las técnicas de filtrado adaptativo para quitar las partes contaminadas de la señal mediante la producción de una estimación óptima de la fuente original.

### **1.5.6 Técnicas para Detección de Artefactos**

Básicamente, los enfoques propuestos para procesar las señales EEG contaminadas por artefactos se dividen en dos categorías principales, el rechazo o la corrección de los segmentos de señal contaminados. En los procedimientos de rechazo, los datos grabados son a menudo analizados por un experto y los segmentos de la señal contaminados por los artefactos serán excluidos de los datos. Este enfoque se utiliza frecuentemente en la investigación médica debido a su simplicidad. Sin embargo, puede ser un procedimiento complicado en el análisis de grandes cantidades de datos y muchos artefactos, que no son tan evidentes, se pueden pasar por alto. El otro inconveniente del método de rechazo es la reducción y la pérdida de datos, el cual restringe en gran medida el uso de este método (Kroupi, Yazdani, Vesin, & Ebrahimi, 2006).

Los métodos más comunes para la corrección de artefactos, especialmente EOG, incluyen la proyección subespacial, entre ellos el análisis de componentes independientes (ICA) y el análisis de componentes principales (PCA), que descompone la señal en componentes independientes y componentes no correlacionadas, respectivamente. Por otra parte, se han propuesto métodos basados en regresión para la corrección de artefactos. Por ejemplo, las técnicas de filtrado adaptativo para eliminar las partes contaminadas de la señal mediante la producción de una estimación optimizada de la fuente original. Por último, los métodos basados en la descomposición wavelet han sido desarrollados y utilizados para corregir artefactos oculares.

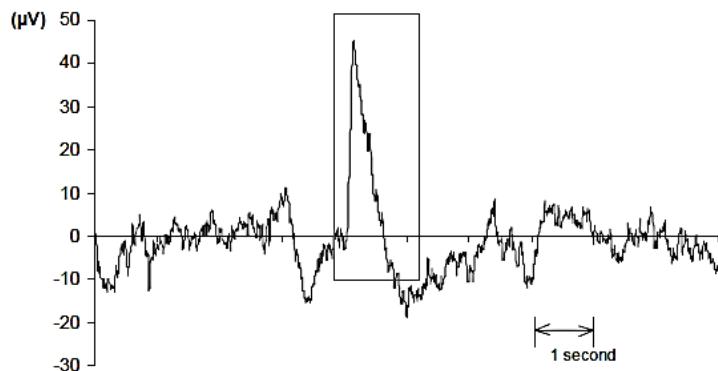
#### **1.5.6.1 Análisis en el dominio del tiempo:**

Las técnicas basadas en el análisis en el dominio del tiempo son aplicadas generalmente para la detección de artefactos oculares, artefactos musculares entre otros que impliquen cambios bruscos en amplitud de la señal EEG

- ***Threshold:***

Las técnicas de detección de artefactos relacionadas con el análisis en el dominio del tiempo generalmente se basan en un umbral de amplitud; dicho lo anterior el threshold es una de las técnicas implementadas en el proyecto la cual consiste en definir como artefacto la amplitud EEG que sobre pase un umbral de N veces la amplitud media de los M segundos muestreados anteriormente.

**Figura 14: Señal pico en EEG**



Fuente: (Velde, 2000)

- ***Correlación:***

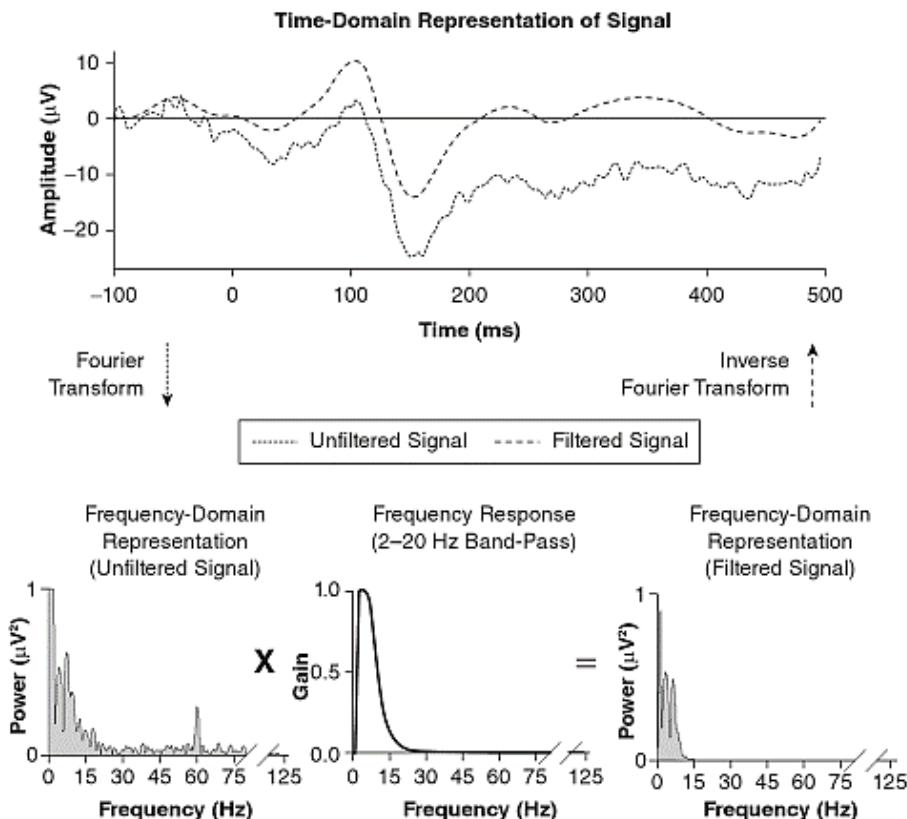
Técnica fundamentada en el conocimiento del comportamiento de una artefacto para obtener patrones o señales de referencia generalizadas propias a cada artefacto con el propósito de realizar a través de algoritmos una correlación entre la señal de referencia y la señal cruda del EEG estableciendo con criterio un margen de error con el cual se detecten la mayor cantidad de artefactos posible sin llegar a detectar como artefacto una señal libre de ellos.

#### **1.5.6.2 Análisis en el dominio de frecuencia**

- ***Filtros digitales:***

Un filtro digital es un filtro que opera sobre señales digitales. Es una operación matemática que toma una secuencia de números (una señal de entrada) y la modifica produciendo otra secuencia de números (señal de salida) con el objetivo de resaltar o atenuar otras características. En la Figura 15 se puede visualizar el espectro de frecuencia de una época de señal EEG la cual presenta componentes altos de frecuencia de 60 Hz y la recuperación de la señal a través de la transformada inversa de Fourier luego del filtrado.

**Figura 15: Flujo aplicación de filtros digitales**



Fuente: Cheryl L Dickter, EEG Methods for the Psychological Sciences

#### - Análisis de componentes independientes (Independent Component Analysis ICA)

La separación ciega de fuentes (BSS) ha recibido una considerable atención debido al hecho de que aborda el problema significativo de encontrar una representación adecuada para los datos multivariados. El análisis de componentes independientes es un método popular de BSS utilizando la hipótesis de que las fuentes originales son no gaussianas, independientes entre sí, y las medidas son una transformación lineal de las fuentes originales. Bajo estos supuestos el problema ICA se puede escribir como:

$$\mathbf{y} = \mathbf{Ax} + \mathbf{u} \quad (1)$$

Donde la variable  $\mathbf{y}$  de dimensión  $\mathbf{S}$  se modela como una combinación lineal de las fuentes estadísticamente independientes  $\mathbf{x}$  de dimensión  $\mathbf{L}$  con ruido gaussiano agregado  $\mathbf{u}$  de dimensión  $\mathbf{S}$  y  $\mathbf{A}$  es una matriz de  $S \times L$ . Si el ruido no se toma en consideración.

#### - FastICA

FastICA es un algoritmo rápido para ICA, que se puede utilizar para el SRS y extracción de características. Se introdujo inicialmente en (Hyvärinen & Oja, 1997). Este algoritmo se basa en un esquema de iteración de punto fijo y maximiza la no-Gaussianidad como una

medida de independencia estadística. Este algoritmo está públicamente disponible en el Paquete FastICA (<http://www.cis.hut.fi/projects/ica/fastica/>).

- ***Identificación ciega de segundo orden (Second Order Blind Identification SOBI)***

Introducido por Belouchrani en (Belouchrani, Abed-Meraim, & Cardoso, 1997), el algoritmo de identificación ciega de segundo orden (SOBI) es una técnica de separación ciega de fuentes para fuentes temporalmente correlacionadas. Más específicamente, se basa en la diagonalización conjunta de un conjunto arbitrario de matrices de covarianza. Por lo tanto, se basa únicamente en las estadísticas de segundo orden de las señales procesadas, en contraste con FastICA, que utiliza técnicas de Cumulant de alto orden. El algoritmo está implementado en el toolbox EEGLAB que disponible públicamente (<http://sccn.ucsd.edu/eeglab/>).

- ***RunICA***

Bell y Sejnowski propusieron el algoritmo ICA Infomax en (Bell & Sejnowski, 1995) sobre la base de una maximización de la información mutua entre las fuentes y los sensores. Cardoso, en (Cardoso, 1997), mostró que la información de maximización es realmente idéntica a la minimización de la divergencia KL (Kullback-Leibler) entre la distribución del vector de salida y el vector de fuentes. Por último, en el mismo estudio de investigación, se muestra que el principio Infomax coincide con el principio de máxima probabilidad en el caso de la separación de fuentes. Algoritmo RunICA, que se implementa en EEGLAB, realiza la descomposición ICA de datos de entrada utilizando el algoritmo ICA Infomax. Para estos algoritmos los sensores dependen deterministamente en las fuentes independientes. En otras palabras, una vez que la matriz de mezcla A se encuentra, las fuentes pueden recuperarse exactamente a partir de los datos observados, utilizando la inversa o la pseudo-inversa de A.

- ***Análisis de Componentes Independientes Bayesiano Variacional (Variational Bayesian Independent Component Analysis - VbICA)***

En el análisis de componentes independientes Bayesiano variacional (Choudrey & Roberts, 2001), se supone que el ruido sea gaussiano con media cero y matriz de precisión diagonal  $\Lambda$  y la distribución de las fuentes están representadas por mezclas gaussianas (Mixures od Gaussians MoGs) (Choudrey & Roberts, 2001). Una MoG se representa en forma general como se muestra en la ecuación 2:

$$p(x) = \sum_k \pi_k \nu(x | \mu_k, \beta_k^{-1}) \quad (2)$$

Donde  $\pi_k$ ,  $\mu_k$  y  $\beta_k$  son, respectivamente, las ponderaciones, la media de los vectores y las matrices de precisión de los componentes Gaussianos. Los métodos bayesianos variacionales tienen como finalidad el aproximar distribuciones posteriores intratables mediante la búsqueda de una distribución apropiada sobre los parámetros y las variables

latentes, de tal manera que esta se factoriza (Attias, 1999). Esta factorización consta de las distribuciones previas sobre las variables y los parámetros. El enfoque variacional del algoritmo se encuentra en el hecho de que cada factor de la distribución aproximada itera hasta la convergencia de forma individual, con el fin de reducir al mínimo la divergencia de Kullback-Leibler (KL) entre el verdadero y las distribuciones aproximadas. El algoritmo VbICA está disponible públicamente en <http://www.robots.ox.ac.uk/~parg/projects/ica/riz/code.html>.

- **Análisis de Componentes Principales (Principal Component Analysis - PCA)**

El análisis de componentes principales (PCA) es otro enfoque para construir la matriz de mezcla **A** y descomponer los datos en sus componentes espaciales. En esta técnica, los vectores de coeficientes normalizados son los vectores propios de la matriz de covarianza. Estos son ordenados de acuerdo a la varianza, con el primer componente que tiene la mayor varianza. Los vectores de coeficientes construidos de esta manera son ortogonales. La principal diferencia entre el PCA y el ICA es que se asume que las fuentes son no correlacionadas y no estadísticamente independientes.

- **Transformada Wavelet**

La Transformada Wavelet (TW) consiste en comparar la señal con ciertas funciones Wavelet, las cuales se obtienen a partir de las Wavelet madre. Un requisito básico es la posibilidad de invertir la transformada, recuperando la señal a partir de esos coeficientes Wavelet calculados (Mathews, 2006). La **TW** es una operación lineal que descompone una señal en componentes a diferentes escalas, donde  **$\Psi(t)$**  es una función de valor real o complejo en **L2(R)** (Señales de energía finita). La **TW** de una función **f(t)** en la escala **a** y posición  **$\tau$** , está dada por la fórmula de la ecuación 3.

$$Wf(a, \tau) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(t) \Psi\left(\frac{t - \tau}{a}\right) dt \quad (3)$$

Donde el factor de escala **a** debe ser un valor positivo y el factor **1/ $\sqrt{a}$**  es usado para la normalización de la energía, este tipo de transformación satisface la conservación de la energía, y la señal original puede ser reconstruida. En la ecuación anterior **TW** depende de dos parámetros, la escala **a** y la posición  **$\tau$** , que varían continuamente sobre los números reales; para valores pequeños de **a**, la Wavelet se expande y la transformada muestra datos sobre una señal aproximada (Berg & Scherg, 1991).

- **Máquinas de Soporte Vectorial**

Vapnik demostró que estando en un cierto espacio de dimensión, solo es posible llegar a separar tantos datos como la dimensión más una unidad. Esta demostración se puede encontrar en (Shoker & Latif, 2002). Con esto se puede concluir entonces que ciertos datos no pueden llegar a ser clasificados en forma lineal. Vapnik llegó a una solución para este problema; la solución es llevar los datos de una dimensión hacia otra dimensión, haciendo uso de Kernels.

Se supone la función descrita en la ecuación 4

$$f(X) = \sum_{j=1}^n w_j \cdot x_j \quad (4)$$

Se puede escribir la función en términos de la combinación lineal de la función de mapeo  $\Phi$ . Por lo tanto:

$$f(X) = \sum_{i=1}^m \alpha_i \sum_{j=1}^m \varphi(x_i) \cdot \varphi(x_j) = \sum_{i=1}^m \alpha_i K(x_i, x_j) \quad (5)$$

Donde  $K$  es llamado Kernel, es considerado como la función Kernel Clasificador y  $\alpha \in \mathbb{R}^m$  son coeficientes de expansión. Lo anterior muestra que haciendo uso de una función Kernel clasificadora, se puede llegar a separar o clasificar los objetos o vectores en un cierto espacio, ya que permite simplificar la función separadora en una forma más simple. Vapnik usó esta idea para llevar los datos que se encuentran en una dimensión que no pueden ser clasificados en forma lineal, a un estado de dimensión diferente, donde puedan ser clasificados, haciendo uso del método de clasificación lineal (Arias, Alzate, & Bejarano, 2009).

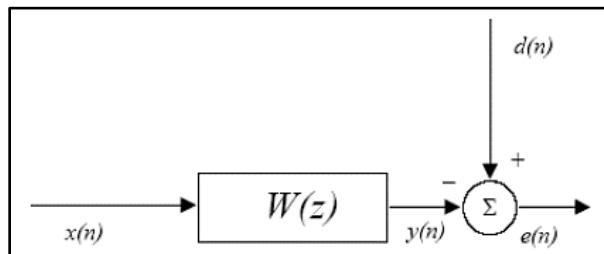
#### - *Filtrado Adaptativo*

Este tipo de filtro se puede definir de manera sencilla y resumida como un filtro digital clásico, con la diferencia que sus parámetros pueden variar en el tiempo, delegando la tarea de variar dichos parámetros a un algoritmo de optimización, el cual calcula constantemente los valores más adecuados. Como primera medida, al tratar filtros digitales necesariamente se hace referencia a dos tipos FIR (de respuesta finita al impulso) e IIR (de respuesta infinita al impulso) (Haykin, 1996). En el recorrido por el mundo del filtrado digital se puede encontrar que existen diferencias entre estos dos tipos, y escoger uno u otro depende de la aplicación que se requiera. Luego se examina la solución de Wiener como la solución óptima en el sentido de reducción de la función de error (Haykin, 1996), conocida también como función de costo, ecuación 6.

$$\mathcal{E} = E[e^2(n)] = E[d^2(n)] - 2 \cdot p^T \cdot w + w^T \cdot R \cdot w \quad (6)$$

Donde:

**Figura 16: Diagrama de bloque filtro óptimo**



Fuente: (Haykin, 1996)

$$e(n) = d(n) - y(n) \quad (7)$$

Se presenta el problema del filtro óptimo, se muestra en la Figura 16, en el que se busca minimizar dicha función; se denotan los coeficientes del filtro como  $w$ , los cuales son invariantes en el tiempo; una señal de entrada  $x(n)$ , que produce una señal de salida  $y(n)$ ,  $d(n)$  (señal deseada),  $p$  que es el vector de correlación cruzada entre la señal de entrada  $x(n)$  y la señal  $d(n)$  y  $R$  la matriz de auto correlación (Haykin, 1996). La salida del filtro  $y(n)$  es el resultado de la convolución entre la entrada y los coeficientes del filtro.  $L$  indica el número de coeficientes del filtro:

$$e(n) = d(n) - \sum_{i=0}^{L-1} w_i \cdot x(n-i) = d(n) - W^T \cdot X(n) \quad (8)$$

Para encontrar el valor de estos coeficientes se presentan muchos métodos, los cuales pueden requerir grandes cálculos matriciales que resultan en pesados procesos para la máquina que los vaya a ejecutar; el objetivo, o principio básico de un filtro adaptativo es actualizar estos coeficientes de manera constante de acuerdo con la señal de entrada.

## 1.5 Estructura del Documento

Este trabajo se divide en cinco capítulos sin incluir los anexos, la descripción de cada uno de los capítulos se resume a continuación:

**Capítulo 1:** en esta capítulo se realiza la introducción al problema de investigación, justificación, objetivos y alcance. La sección más grande trata lo concerniente al marco de referencia, en donde se exponen las técnicas desarrolladas actualmente para la eliminación de artefactos en señales EEG multicanal y de canal simple y también las generalidades del electroencefalograma, es decir, adquisición de la señal, posicionamiento de los electrodos, formas de onda, etc. Este marco de referencia incluye un marco teórico y un marco conceptual, con el fin de proveer al proyecto de unas bases teóricas sobre el cual se fundamenta este trabajo de grado. Este marco de referencia se construyó utilizando las bases de datos disponibles en la universidad y referencias bibliográficas que estuvieron a nuestra disposición.

**Capítulo 2:** este capítulo contiene el proceso que se realizó para seleccionar los métodos para la detección de artefactos y la descripción detallada de cada uno de ellos, así como el diagrama de flujo, mostrando la implementación de cada método. De igual forma, en esta fase se analizaron las bases de datos suministradas por el Hospital Clinic de Barcelona, que contienen registros de señales EEG de canal simple captadas bajo sedación en procedimientos de endoscopia.

**Capítulo 3:** este capítulo inicia indicando la metodología para la programación en MatLab de los métodos seleccionados en el capítulo 2, y la metodología para implementar la interfaz gráfica. A continuación se muestran los resultados de simulación de estos algoritmos, utilizando como herramienta de simulación MatLab, con lo cual se pretende mostrar la base experimental sobre la cual se compararán los resultados obtenidos en capítulos posteriores.

De igual forma se muestra la metodología para la implementación de los métodos seleccionados en el capítulo 2, esta vez en el lenguaje de programación Python. Al igual que con MatLab, en Python se describe como se implementó la interfaz gráfica.

Este capítulo termina comparando los métodos obtenidos en la simulación de MatLab y Python. Con esto se pretende demostrar que la implementación de los métodos para la detección de artefactos en cualquiera de estas dos herramientas da resultados similares, y la programación en Phyton nos permite migrar fácilmente al sistema embebido seleccionado.

**Capítulo 4:** en este capítulo realiza la selección de la tecnología a utilizar, teniendo en cuenta que el prototipo final debe soportar una interfaz gráfica y un modo de introducir las bases de datos por medio de un dispositivo de almacenamiento. Se realiza una comparación de las características más importantes de tres sistemas embebidos: Arduino Uno, Raspberry Pi, y Beagle Bone. Después del proceso de selección del sistema embebido, se optó por utilizar para este proyecto la Raspberry Pi. También se muestra el proceso para implementar los métodos seleccionados en el capítulo 2 en el sistema embebido. Al final se procede a comparar los resultados obtenidos con los resultados teóricos y simulados.

**Capítulo 5:** en este capítulo se dan las conclusiones y recomendaciones de este trabajo de grado, y se termina el capítulo con el trabajo futuro.

## CAPÍTULO 2

### 2.1 Selección de Métodos para la Detección de Artefactos

La selección de los métodos para la eliminación de artefactos estuvo basada en ciertos requerimientos, necesarios para ser utilizados en este trabajo. Estos requerimientos son:

- Los métodos seleccionados deben ser aplicables a señales EEG de un solo canal, esto debido a que la base de datos disponible para este trabajo de grado se compone únicamente de señales EEG de un solo canal.
- Los métodos seleccionados deben ser efectivos en la detección de los artefactos de más interés para este trabajo, tales como artefactos musculares y artefactos oculares, pues son los que distorsionan la señal EEG en mayor medida e impiden su análisis.

Para exponer de forma más simple el resumen del proceso de selección de los métodos, se construyó la Tabla 1. En esta se muestran los aspectos más importantes de las principales técnicas para la reducción de artefactos y las técnicas seleccionadas para ser implementadas en este trabajo.

**Tabla 1: Selección de técnicas para detección de artefactos.**

Método	Fuente o artículo de metodología	Aplicación	Software	Resultados	Aplicabilidad a EEG de canal simple
Filtro adaptativo	<a href="http://hal.arc-hives-ouvertes.fr/docs/00/41/99/18/PDF/Artic_FA_esp.pdf">http://hal.arc-hives-ouvertes.fr/docs/00/41/99/18/PDF/Artic_FA_esp.pdf</a>	Identificación y reducción de artefactos oculares y artefactos causados por la alimentación AC que están presentes en señal EEG	Se desconoce software existente	Se logra reducir en gran porcentaje la presencia de artefactos oculares y artefactos de la línea de tensión en señal EEG y se concluye que es aplicable a señal EEG de una canal simple	SI
Separación ciega de fuentes	<a href="http://www.ncbi.nlm.nih.gov/pmc/articles/PMC8492050/">http://www.ncbi.nlm.nih.gov/pmc/articles/PMC8492050/</a>	Su desarrollo y aplicación se centra usualmente en la reducción de artefactos oculares en la señal EEG.	Se desconoce software existente	Es un método que ofrece excelentes resultados pero implica tener señales de referencias y una señal EEG adquirida con más de un canal.	NO
Análisis de Componentes Independientes (ICA)	<a href="#">Fundamentos básicos del Análisis de Componentes Independientes</a>	Método capaz de reconocer y eliminar automáticamente artefactos provenientes de movimientos oculares y funciones cardíacas.	ICAtoolbox 2.0 ICAlab V2.1	Es un método que ofrece excelentes resultados al eliminar artefactos pero a la vez es un método basado en la separación ciega de fuentes por tal motivo no es aplicable a señales de un solo canal.	NO

<b>Análisis de Componentes Principales (PCA)</b>	<a href="http://sccn.ucsd.edu/~jung/pdf/PSYOO.pdf">http://sccn.ucsd.edu/~jung/pdf/PSYOO.pdf</a>	En la mayoría de los casos se aplica para reducir en alto porcentaje la presencia de señales cardíacas, señales oculares o cualquier otra señal que se haya adherido a la señal EEG.	Se desconoce software existente	Es una extensión de la metodología ICA que ofrece omitir las señales de referencia para la eliminación de artefactos; se basa de igual manera en la separación ciega de fuentes.	NO
<b>Transformada Wavelet</b>	<a href="#">Removal of Ocular Artifacts in the EEG through Wavelet Transform without using an EOG Reference Channel</a>	Su aplicabilidad en el EEG en mayor parte se centra al manejo de la proveniente de actividad ocular.	EEGLab (MatLab Toolbox)	Se logra reducir en buena medida los artefactos existentes con relación a EOG en la señal EEG y es aplicable a señales de EEG en un solo canal	SI
<b>Señales de referencia</b>	<a href="http://alexander.tue.nl/extra2/9903998.pdf">http://alexander.tue.nl/extra2/9903998.pdf</a>	Consiste en la reducción de artefactos mediante la sustracción de señales que describen los diferentes artefactos y la señal EEG	Se desconoce software existente	Es aplicable a señales de EEG en un canal simple sin ofrecer gran eficiencia puesto que no se identifican todos los artefactos	NO
<b>Correlación</b>		Puede aplicarse para la detección de artefactos oculares a través de patrones elementales de la señal EEG.	Se desconoce software existente	El método de correlación demuestra mejores resultados que los métodos análogos, en la detección de artefactos oculares.	SI
<b>Umbral de Amplitud</b>	<a href="http://alexander.tue.nl/extra2/9903998.pdf">http://alexander.tue.nl/extra2/9903998.pdf</a>	Un artefacto para este método está definido como las amplitudes de la señal EEG que exceden un umbral o threshold definido previamente.	Se desconoce software existente	Este método muestra gran precisión en la detección de artefactos de gran amplitud	SI
<b>Slope o Derivada</b>	<a href="http://alexander.tue.nl/extra2/9903998.pdf">http://alexander.tue.nl/extra2/9903998.pdf</a>	El método aplica la primera derivada o la pendiente para detectar la actividad muscular en la señal EEG.	Se desconoce software existente	El método de derivada es aplicado para detectar EMG basado en el cambio rápido en la señal a prueba.	SI

Uno de los grandes condicionantes para la selección de los métodos a implementar en este proyecto es que las señales EEG que se tienen a disposición fueron adquiridas con un equipo de un solo canal, esto limita en gran medida la selección de métodos y deja 5 posibles candidatos. La propuesta para este trabajo es implementar 3 métodos y comprobar su eficiencia en la detección de distintos tipos de artefactos.

Las señales que alteran en mayor medida la amplitud de la señal EEG son las provenientes de la actividad muscular y ocular. Por ende son los artefactos EMG y EOG los que

generalmente son identificados a través de análisis en el dominio del tiempo; mas no se podrá descartar nunca un errado funcionamiento de los electrodos que permiten la adquisición de la señal, puesto a que es muy frecuente que se presenten grandes amplitudes en la señal de EEG debido a la baja impedancia que requieren los electrodos. Por esta razón, seleccionamos los métodos de umbrales de amplitud y derivada, pues son los métodos que, además de ser aplicables a señales EEG de un solo canal, pueden mostrar un mejor desempeño detectando cambios bruscos en la amplitud de la señal.

El último método que se ha de implementar es el método de correlación. Se escogió este método ya que se han realizado varios estudios involucrando la correlación de patrones, comprobando su eficacia a la hora de detectar artefactos que siguen un patrón, como es el caso de los artefactos oculares. Además es un método cuyo análisis e implementación es posible utilizando las herramientas contempladas para este trabajo, como son Matlab y la posterior implementación en un sistema embebido.

## **2.2 Descripción de métodos para eliminación de artefactos**

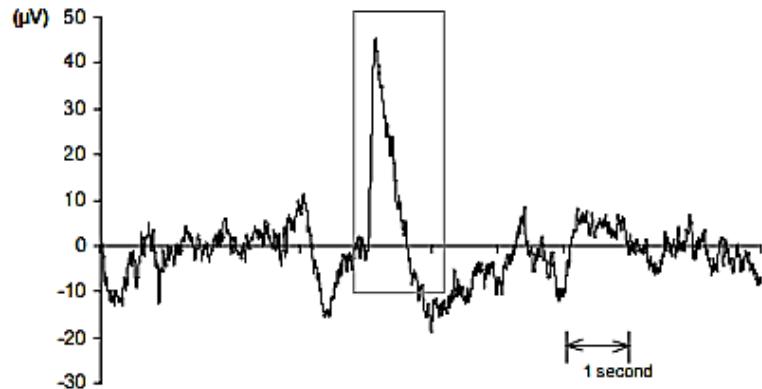
### **2.2.1 Métodos de Detección de artefactos por análisis en el dominio del tiempo.**

Los métodos que requieran de análisis de la señal en el dominio del tiempo generalmente son aplicados para referenciar artefactos a través anomalías en la amplitud de la señal analizada. Las señales que no provienen del cerebro aportan energía a la señal EEG causando variaciones o alteraciones en la amplitud que son fácilmente identificables, puesto a que ya están definidos los rangos de amplitud propios de la actividad encefálica.

#### **2.2.1.1 UMBRALES DE AMPLITUD**

Los umbrales o threshold en amplitud de una señal EEG son determinados de manera empírica y a razón del procesamiento y aplicaciones para las cuales se empleen las señales de EEG. Un artefacto está definido como las amplitudes de la señal de EEG que excede por seis veces el umbral de amplitud previamente definido como la media aritmética de 10 segundos de muestreo anteriores al dato analizado; en referencia a esta definición es posible detectar diferentes artefactos inmersos en la señal EEG a través de una análisis en el dominio del tiempo, discretizando la señal de EEG con el fin de localizar muestras que excedan en seis veces el umbral de amplitud establecido.

Figura 17: Artefacto presente en la señal EEG referenciado a través de anormalidad en la amplitud.



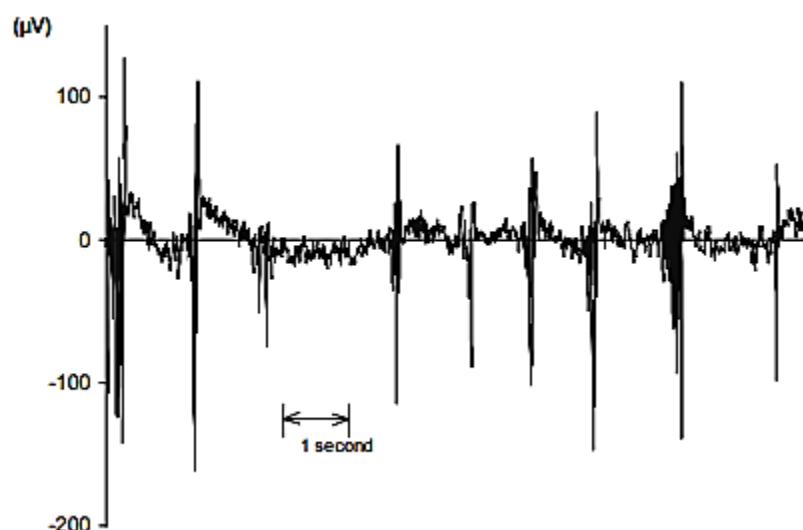
Fuente: (Velde, 2000)

#### 2.2.1.2 UMBRALES DE PENDIENTE O DERIVADA

La actividad rápida en la señal EEG es atribuida a la señal proveniente de la actividad muscular y es un artefacto inevitablemente presente. Varios estudios realizados por (Barlow, 1985) demostraron la efectividad que presenta la aplicabilidad de la primera derivada o pendiente así como la segunda derivada, al momento de detectar la actividad muscular.

Es posible aplicar los umbrales de pendiente o derivada teniendo la señal EEG discretizada con el propósito de establecer el threshold de pendiente entre muestra y muestra. De esta forma será considerada como actividad muscular las muestras que superen dicho threshold.

Figura 18: Actividad rápida o actividad muscular presente en la señal EEG y detectable a través de los umbrales de pendiente



Fuente: (Velde, 2000)

### **2.2.1.3 Método de Correlación**

Para poder reconocer un el patrón EOG, es necesario disponer de un sistema que cuantifique el grado de interdependencia o similitud entre la señal EEG y la señal patrón. En otras palabras determinar la correlación existente entre las dos señales, para ellos se utilizó la técnica denominada correlación cruzada, la cual proporciona en un índice el grado de similitud entre las señales. Con estos índices se puede diseñar un algoritmo para que el sistema adquiera la capacidad de identificar la presencia de un artefacto EOG independiente de la persona a la que pertenezca el EEG.

#### *Correlación Cruzada*

El estudio de los sistemas de procesamiento de señal ha sido dominado por el concepto de convolución, y se ha descuidado un poco su pariente cercano la correlación. Aunque son en forma similares (de hecho una convolución con un filtro FIR simétrico se puede considerar una correlación), la manera en que uno debe pensar en los dos es diferente (Stein, 2000). La convolución es por lo general entre una señal y un filtro; se puede pensar en ella como un sistema con una sola entrada y coeficientes almacenados. La correlación cruzada es por lo general entre dos señales; se puede pensar en ella como un sistema con dos entradas y sin coeficientes almacenados.

La ecuación (9) muestra la forma general de la correlación cruzada de  $x$  y  $h$  como la integral del producto de dos funciones, una desplazada por la otra por un tiempo  $\tau$  entre el intervalo  $a < t < b$  y viene dada por (Yarlagadda, 2010):

$$R_{xh}(\tau) = x(\tau) \ast h(\tau) = \int_a^b x(t)h(t + \tau)dt = (x(t)h(t + \tau)) \quad (9)$$

Función de correlación cruzada da la similitud entre las dos funciones:  $x(t)$  y  $h(t - \tau)$ . Muchas veces la segunda función  $h(t)$  puede ser una versión corrupta de  $x(t)$ , tal como  $h(t) = x(t) + n(t)$ , donde  $n(t)$  es una señal de ruido. En el caso donde  $x(t) = h(t)$ , la función correlación cruzada se reduce a la función auto correlación. En este caso la integral de auto correlación da el valor más alto en  $\tau = 0$ .

### **2.2.1.4 Filtro Notch**

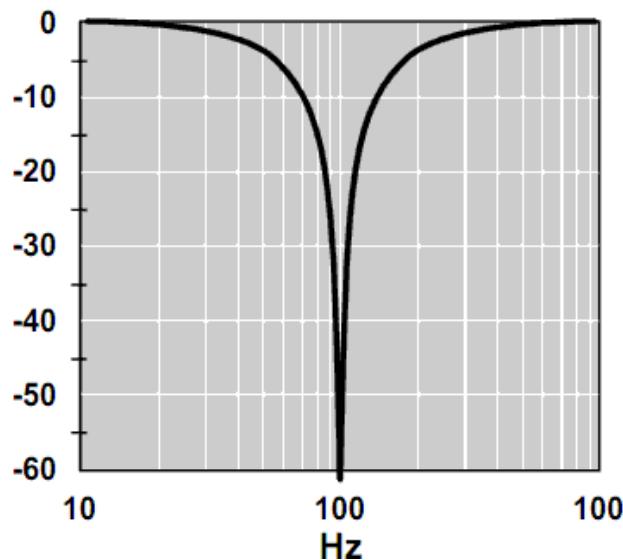
Uno de los motivos más comunes para el filtrado del EEG es cuando las señales de interés ocupan un rango de frecuencia conocida, la atenuación de las frecuencias extrañas a través del filtrado puede mejorar la relación señal a ruido. Uno de los objetivos del presente trabajo consiste en el diseño e implementación de éste filtro. La naturaleza de la acción de filtrado, es determinada por las característica de la respuesta en frecuencia, la cual depende de la elección de los parámetros del sistema. Entonces, con una adecuada selección de parámetros, podemos diseñar un filtro selectivo en frecuencia el cual deje

pasar componentes de frecuencia en ciertas bandas, mientras atenúa señales que contienen componentes de frecuencia en otras bandas.

El objetivo de esta sección es diseñar un filtro rechazo de banda tipo notch, para atenuar los artefactos inducidos por la línea de alimentación en la banda de frecuencias de 50Hz. Ha habido varios intentos de eliminar la interferencia de línea de alimentación (50Hz) mediante el uso de técnicas de procesamiento de señales digitales. En aplicaciones en las que la información de interés está contenida dentro de las bandas de EEG clásicas: delta (0-4 Hz), theta (4-7 Hz), alfa (7-13 Hz) y beta (, 13-35 Hz), es común el uso de un filtro notch 60/50 Hz con un zero fijo en su respuesta en frecuencia para eliminar el ruido de los datos. A veces, la señal de EEG es adicionalmente filtrada con un filtro paso bajo con una frecuencia de corte de menos de 50 Hz para asegurar la integridad de los datos (Olguín, Bouchereau, & Martínez, 2005).

Un filtro notch, es un filtro que contiene una o más hendiduras o notches, o idealmente, un “null” perfecto en su respuesta en una frecuencia característica. En la Figura 19 se muestra la respuesta en frecuencia ideal de un filtro notch en la frecuencia Fc. Este tipo de filtros son muy útiles en aplicaciones en las cuales componentes específicos de frecuencia deben ser eliminados (Proakis & Manolakis, 2007).

**Figura 19: Respuesta en frecuencia filtro notch**  
**dBV**



Fuente: <http://witcircuit.blogspot.com.co/2013/04/hq-notch-filter-without-close-tolerance.html>

Para crear un área nula en la respuesta en frecuencia de un filtro en la frecuencia x, se introducen un par de zeros complejos conjugados en el círculo unitario al ángulo x. Esto es:

$$Z_{1,2} = e^{\pm j\omega_0} \quad (10)$$

Entonces la función para un filtro notch con respuesta finita al impulso es simplemente

$$\begin{aligned} H(z) &= b_0(1 - e^{j\omega_0}z^{-1})(1 - e^{-j\omega_0}z^{-1}) \\ &= b_0(1 - 2\cos\omega_0 z^{-1} + z^{-2}) \end{aligned} \quad (11)$$

### 2.3 Descripción Y Análisis de la Base de Datos

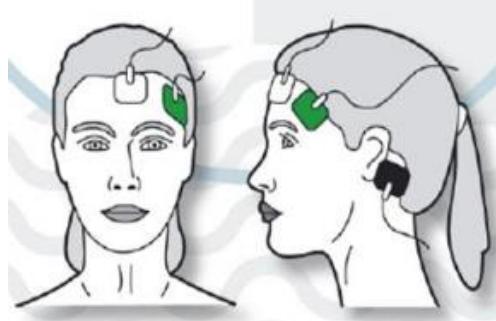
Para el desarrollo de este proyecto contamos con una base de datos de 120 registros EEG de un solo canal de aproximadamente una hora de duración, suministrada por el Hospital Clinic de Barcelona. Cada registro pertenece a un paciente sometido a un procedimiento de endoscopia, a los cuales se les indujo el estado de sedación administrándoles diferentes dosis de propofol y remifentalino. La señal EEG fue adquirida usando el sistema de monitoreo A-Line AEP monitor fabricado por Danmeter el cual utiliza una tasa de muestreo de 900Hz, y adquiere la señal del EEG utilizando tres electrodos, donde uno de ellos hace de referencia (Figuras 19 y 20).

Figura 20: Monitor A2000



Fuente:<http://www.danmeter.dk/products/neuromonitoring/dmtech/danmeter%20loc%20technology%20ver%201.3.pdf>

Figura 21: Posicionamiento de electrodos según el fabricante



Fuente:<http://www.danmeter.dk/products/neuromonitoring/dmtech/danmeter%20loc%20technology%20ver%201.3.pdf>

Debido a la forma de posicionar los electrodos en los registros EEG adquiridos por este sistema, en la señal hay una fuerte presencia de artefactos oculares, a causa de la cercanía de los electrodos a la región peri orbitaria. También hay presencia de artefactos musculares, especialmente originada en los músculos faciales. Por otro lado artefactos, como el cardiaco, no tienen gran influencia en esta señal.

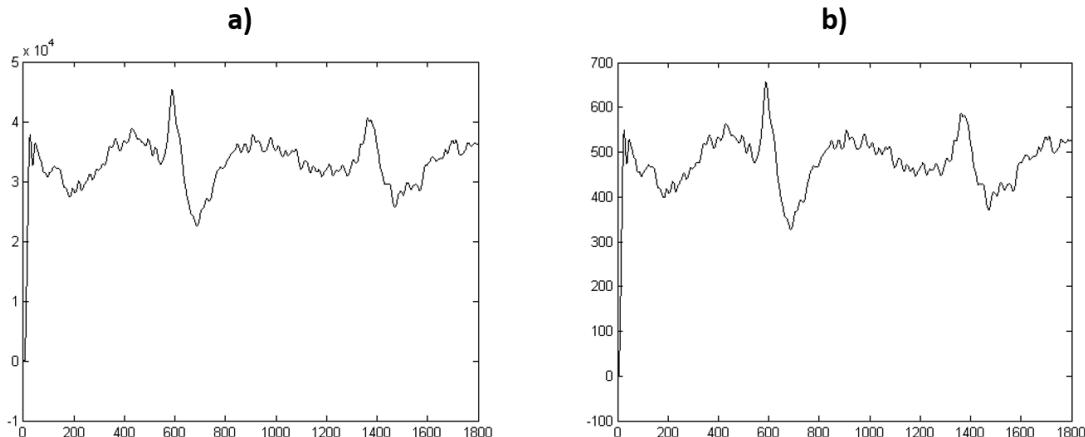
### 2.3.1 Acondicionamiento de la señal

El equipo con el que se adquirió la señal EEG, entrega los registros de la señal en formato binario de 16 bits sin signo. Para visualizar y analizar la información contenida en el registro EEG, fue necesario escalar la señal para que coincida con los rangos de amplitud de una señal EEG real (20-100  $\mu$ V). El valor escalado de la señal se obtuvo utilizando la siguiente ecuación:

$$\text{Señal Escalada} = \frac{\text{Señal Cruda}}{65535 * 950}$$

En la Figura 22a se muestra un epoch del registro 041221JM sin procesar y en la Figura 22b se muestra el mismo epoch después del proceso de escalamiento.

**Figura 22: a) Epoch Crudo b) Epoch Escalado**



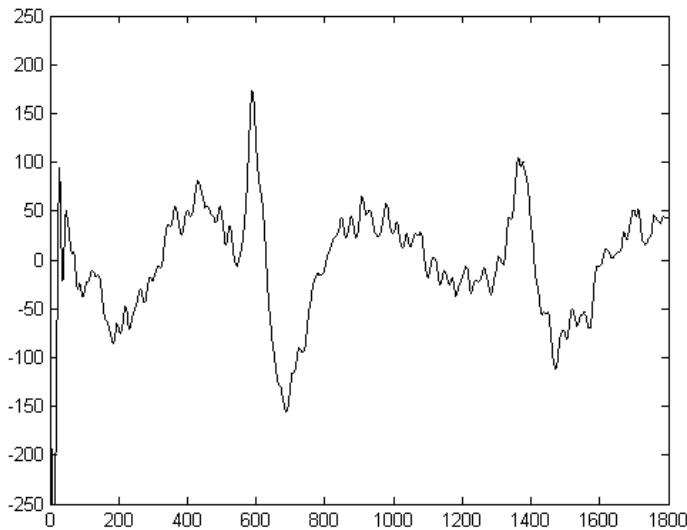
Fuente: De los Autores. Base de Datos registros 041221JM

Después del proceso de escalamiento, la señal sigue teniendo un nivel de DC que mantiene los valores siempre positivos. Con el fin de analizar una señal con características reales es necesario quitar el nivel de DC para obtener la señal bipolar, eso se logró obteniendo la media de cada registro y realizando la diferencia entre esta y los valores del registro, como se muestra la siguiente ecuación:

$$\text{Señal sin Nivel de DC} = \text{Señal Escalada} - \text{Media Aritmetica Señal Escalada}$$

En la Figura 23 se muestra un ejemplo de la señal después del proceso de escalamiento y remoción del nivel de DC, esta tiene los rangos típicos en amplitud de las señales EEG reales.

**Figura 23: Señal EEG Final**



Fuente: De los Autores. Base de Datos registros 041221HL, JM y LX

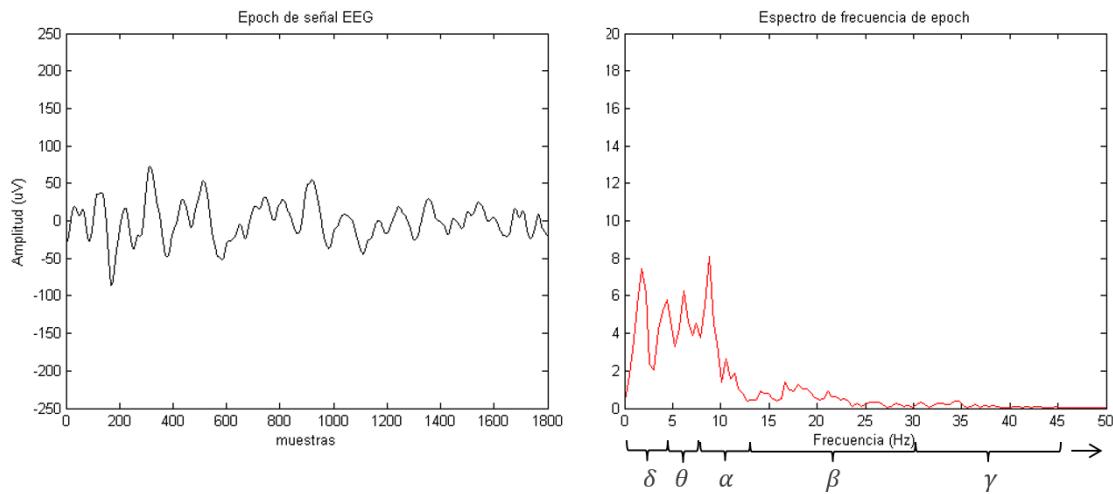
### 2.3.2 Formas de onda

En los registros EEG de la base de datos que tenemos a nuestra disposición, se presentan los ritmos EEG:

- Delta: Menores a 4 Hz
- Theta: Entre 4 a 7 Hz
- Alfa: Entre 8 a 13 Hz
- Beta: Entre 14 a 30 Hz
- Gamma: Mayores a 30 Hz

En la Figura 24 se muestra un epoch del registro 041221JL, del cual se muestra una gráfica en amplitud y otra en el dominio de la frecuencia utilizando transformada de Fourier. En el espectro de frecuencia de la señal se indican los rangos de cada uno de los ritmos presentes en una señal EEG normal. Todos los ritmos están presentes en toda señal EEG, lo que cambia es su intensidad o amplitud.

**Figura 24: Ritmos EEG**

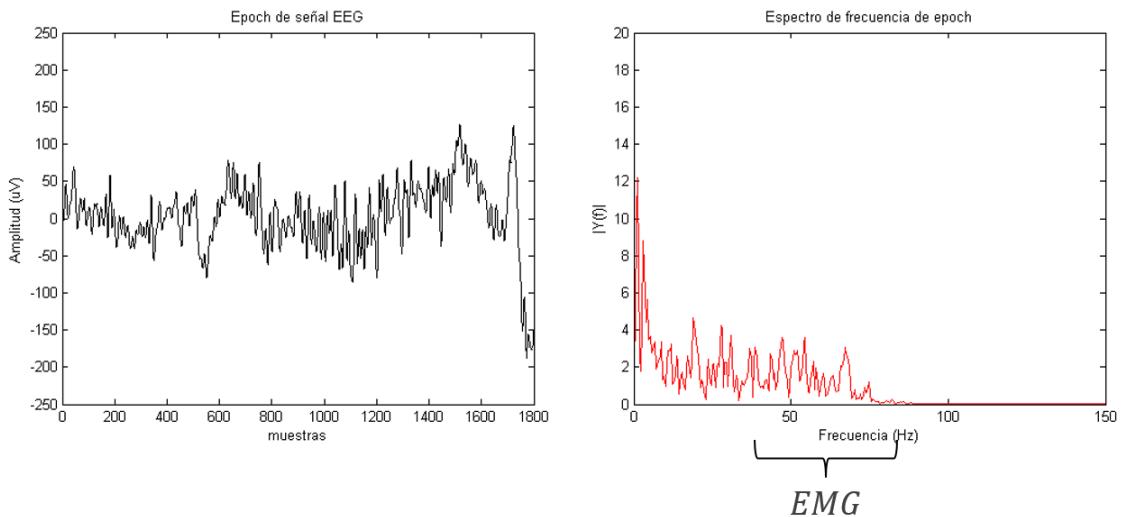


Fuente: De los Autores. Base de Datos registros 041221JM

En la señal EEG también existe una fuerte presencia de artefactos, los cuales sirven al propósito de este trabajo de grado. A continuación se explican los artefactos más comunes y sus características:

- **Artefacto muscular o EMG:** se caracteriza por presencia de componentes de alta frecuencia de hasta 200 Hz, como se puede observar en la Figura 25.

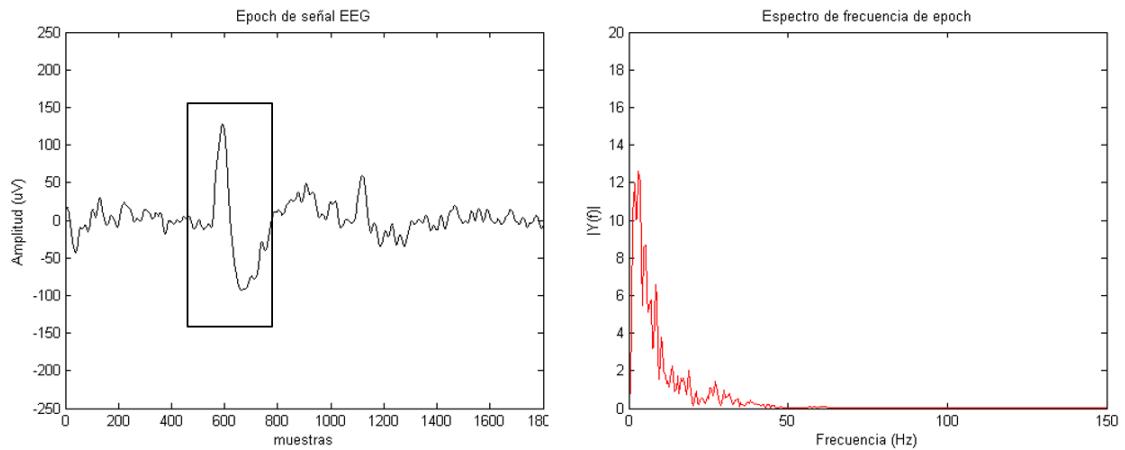
**Figura 25: Artefacto Muscular**



Fuente: De los Autores. Base de Datos registros 041221HL, JM y LX

- **Artefacto ocular o EOG:** tiene una forma de onda característica, la cual no es simétrica entre el semiciclo positivo y negativo. La forma de onda característica se puede observar en la Figura 26.

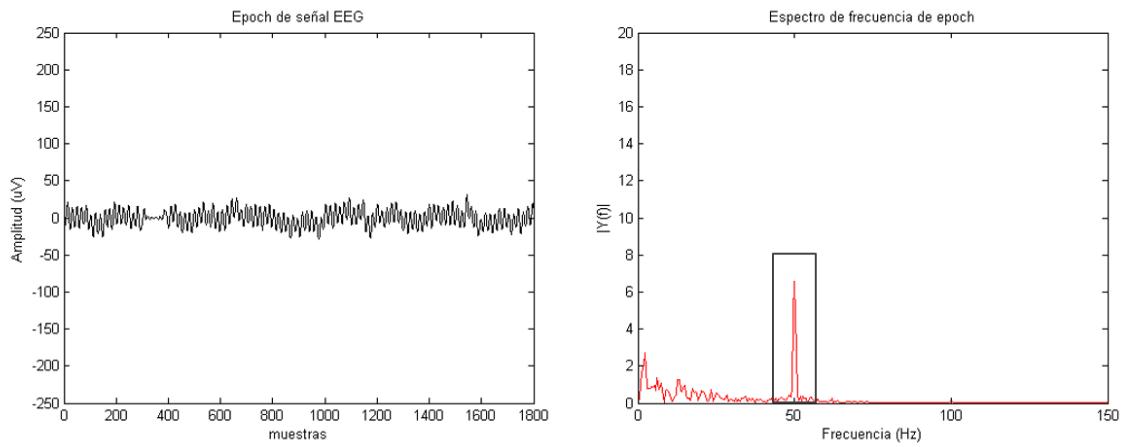
**Figura 26: Artefacto Ocular**



Fuente: De los Autores. Base de Datos registros 041221HL, JM y LX

- **Artefacto de 50Hz (60):** la fuente más significativa de este tipo de artefacto es la inducción electromagnética proveniente de la línea de alimentación. Se caracteriza por un componente de alta amplitud en el rango de 50 Hz (60 Hz) en la gráfica del espectro de frecuencia, esto se evidencia en la figura.

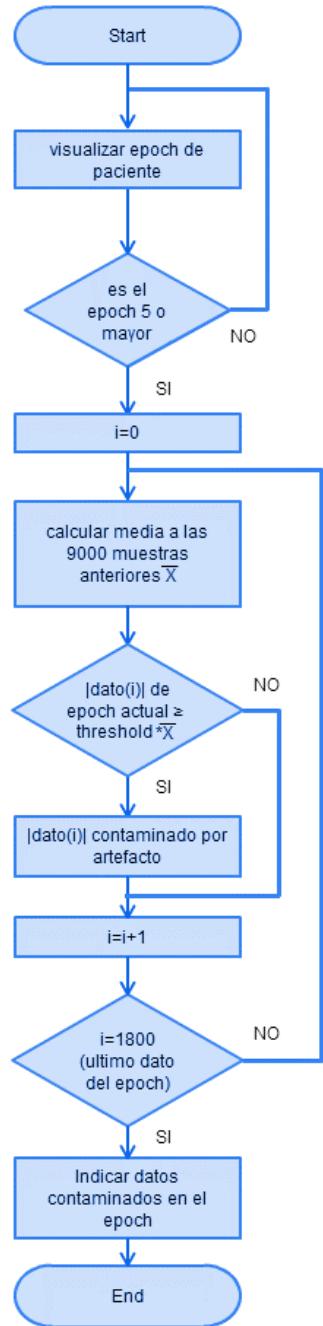
**Figura 27: Artefacto 50 Hz**



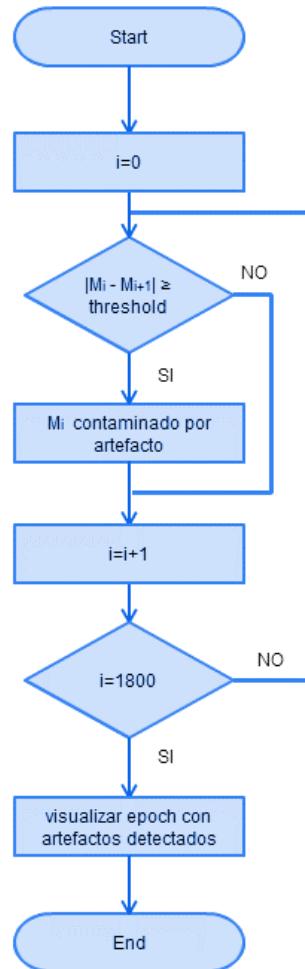
Fuente: De los Autores. Base de Datos registros 041221HL, JM y LX

## Diagramas de Flujo para la Implementación y Evaluación de los Métodos

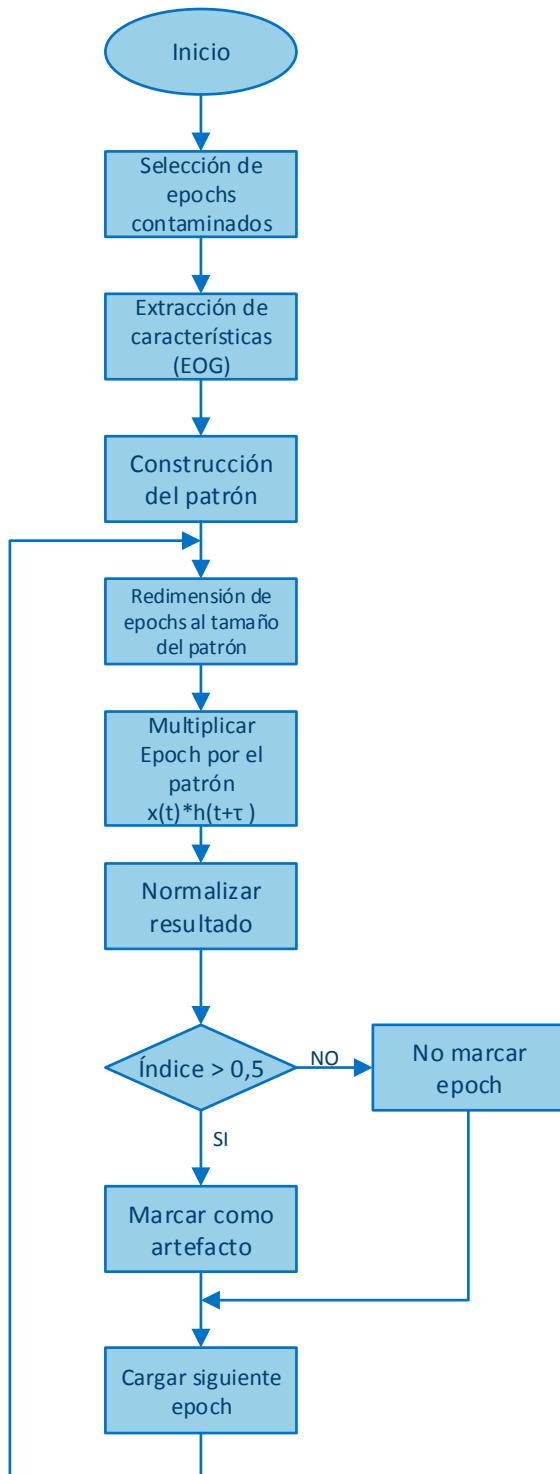
### 2.4.1 Diagrama de Flujo Método Umbral de Amplitud



#### 2.4.2 Diagrama de Flujo Método Umbral de Pendiente



### 2.4.3 Diagrama de Flujo Implementación del Método de Correlación



## CAPITULO 3

### 3.1 Implementación de Métodos en Matlab

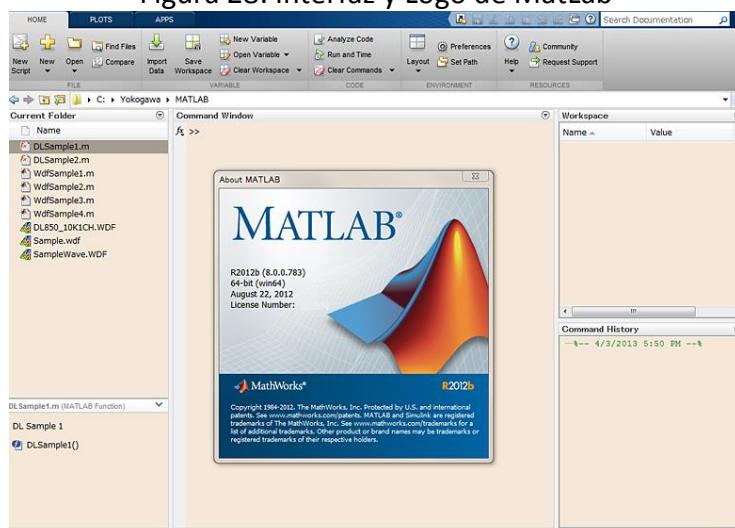
#### 3.1.1 MatLab como herramienta de simulación

MatLab (Matrix Laboratory) es un software fabricado por la empresa americana MathWorks, fue creado por el matemático y programador de computadoras Cleve Moler en 1984, surgiendo la primera versión con la idea de emplear paquetes de subrutinas escritas en Fortran en los cursos de álgebra lineal y análisis numérico, sin necesidad de escribir programas en dicho lenguaje. Según la página del fabricante (MathWorks, s.f.):

“MatLab® es el lenguaje de alto nivel y el entorno interactivo utilizado por millones de ingenieros y científicos en todo el mundo. Le permite explorar y visualizar ideas, así como colaborar interdisciplinariamente en procesamiento de señales e imagen, comunicaciones, sistemas de control y finanzas computacionales.”

En primera instancia este software va a ser en el que se implementen y prueben los algoritmos para la detección de artefactos, debido a que MatLab proporciona un ambiente integrado de programación, especialmente enfocado en procesamiento de datos. Adicionalmente cuenta con numerosos Toolboxes, los cuales facilitan en gran medida la implementación y ensayo de los algoritmos a implementar. Por ultimo está el hecho de que la licencia está disponible para su uso en el campus de la Universidad de San Buenaventura y se puede contar con esta valiosa herramienta de trabajo. En la Figura 22 se muestra la interfaz y el logo de MatLab.

Figura 28: Interfaz y Logo de MatLab



Fuente: De los Autores.

### **3.1.2 Diseño de la interfaz gráfica en MatLab**

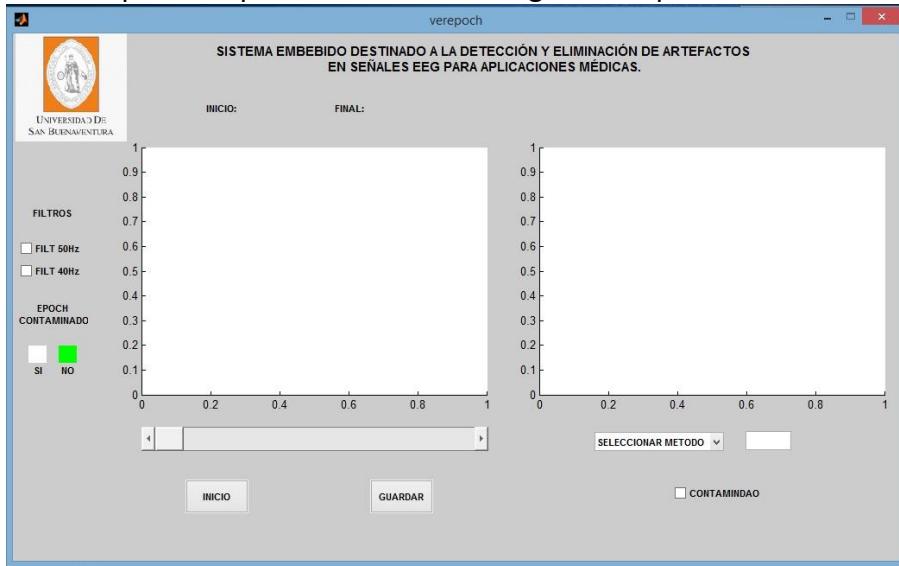
Es importante aclarar que la interfaz implementada en esta herramienta de simulación tiene como objetivo principal, visualizar los resultados de los métodos aplicados permitiendo al usuario seleccionar y variar entre ellos, analizando al momento uno a uno los epochs de EEG pertenecientes al paciente que se seleccione. Para el diseño de la apropiada interface gráfica se establecieron como características principales la facilidad de manejo, la interactividad con el usuario, representación concreta y concisa de la información de manera que el usuario pueda visualizar todo lo necesario para realizar un buen diagnóstico.

Planteado lo anterior se fijó dos ventanas de visualización, una para la visualización de los epochs junto con una barra deslizante que permite desplazarse a través de todo el registro seleccionado y visualizar el epoch correspondiente. La segunda ventana se dio debido a la importancia de poder visualizar de momento el espectro de frecuencia del registro seleccionado y así validar la presencia de actividad que no es propiamente del EEG.

Se decidió que la interfaz contuviese un menú el cual permitiera al usuario seleccionar el registro deseado para analizar y el método a aplicar, proporcionando la opción de trasladarse de método a método con el fin de que el usuario pueda apreciar las diferencias de eficiencia, aplicabilidad y funcionalidad existentes entre los mismos. También se implementó la opción de aplicar o no los filtros desarrollados (filtro pasa bajas, y filtro notch) con el propósito de verificar su debido funcionamiento al igual que percibir de momento la diferencia entre la señal cruda de EEG y la señal posteriormente filtrada.

Se implementó en la interfaz gráfica la opción de variar el threshold del método aplicado con propósito de analizar el comportamiento del mismo al aplicar diferentes valores; se vio la necesidad de ilustrar en la misma interfaz, el rango de segundos del epoch que de momento se está visualizando. La interfaz gráfica finalizada se muestra en la Figura 29.

Figura 29: Captura de pantalla de la interfaz gráfica implementada en MatLab



Fuente: De los Autores.

### 3.1.3 Metodología para la implementación y simulación de los algoritmos para la detección de artefactos en la herramienta MatLab

Antes de empezar la implementación de todos los métodos en conjunto, se hizo en primera instancia por facilidad y eficiencia el análisis individual de los métodos, ya que de esta manera una vez hayan sido implementados, solo se requiera unificarlos llegando a obtener de una interfaz de usuario donde sea posible hacer selección del método a aplicar.

Para cualquier método a implementar es esencial considerar que la información contenida en la base de datos suministrada fue muestreada a una frecuencia de 900 Hz, lo que indica que cada segundo se tendrán 900 muestras de información. Tal como se hizo mención anteriormente, la representación de la señal proveniente del encéfalo (EEG), generalmente se da en épocas o epochs delimitados por 2 segundos de información, lo que implica 1800 muestras por cada época o epoch a una frecuencia de muestreo de 900 Hz. Este registro de muestras es cargado y leído por la herramienta de programación MATLAB® y convertido al tipo de dato conocido como vector o matriz de una fila y n columnas, donde se pueden extraer uno a uno los datos contenidos en el arreglo de valores.

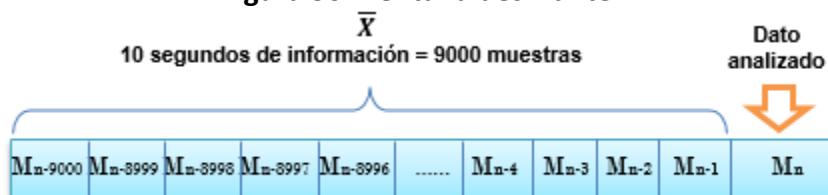
### **3.1.3.1 Umbral de Amplitud**

El método threshold de amplitud a implementar, indica que son artefactos aquellas amplitudes de la señal EEG que superen o excedan por seis veces un umbral de amplitud definido como la media aritmética de 10 segundos de información anteriores al dato analizado.

Para lograr lo anterior se requiere de un análisis uno a uno de los datos contenidos por cada registro, al igual que calcular al mismo tiempo la media aritmética de las muestras que demanden 10 segundos de información previa al dato que está siendo analizado. Teniendo esto se procede a evaluar la condición principal, la cual consiste en determinar si el dato analizado excede por seis veces dicha media aritmética.

Planteado lo anterior se implementó una ventana deslizante como muestra en la Figura 30, la cual tiene la capacidad de recorrer cada uno de los datos del vector o registro y evaluar si anterior al dato analizado existen 10 segundos de información.

**Figura 30: Ventana deslizante**

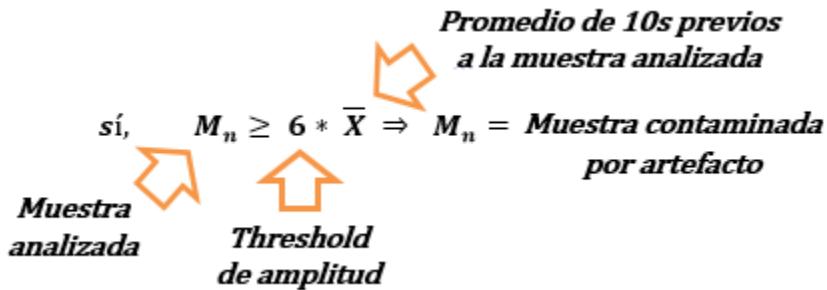


Fuente: De los Autores.

Si no existen 10 segundos de información previos al dato analizado el método no será aplicado debido a que se requiere de dichos segundos para obtener una determinación fiable acerca de si el dato analizado es artefacto o no. En contra parte si existen los 10 segundos previos al dato analizado se aplica el método haciendo efectiva la media aritmética.

Después de realizar el cálculo de la media aritmética, se compara este valor con el dato analizado, si el dato analizado supera en 6 veces la media aritmética calculada, el dato será considerado como artefacto.

**Figura 31: Ecuación método umbral de amplitud**



Fuente: De los Autores.

Este proceso se repetirá automáticamente a través del comando for(), hasta analizar por completo los datos contenidos en cada época, es decir que la ventana se desplazara dato por dato evaluando la respectiva condición. Los índices de las muestras que son determinadas como contaminadas, es decir los datos en donde se cumple la condición evaluada, son almacenados en una matriz con el fin de obtener la posición exacta y de este modo poder indicar todas las muestras dentro del epoch visualizado.

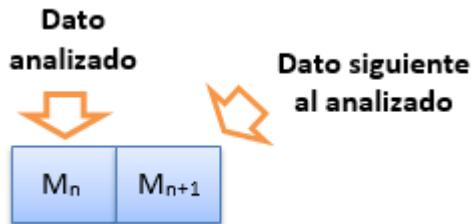
La señal de EEG no presenta nivel de DC y contiene valores positivos y negativos en amplitud, por lo tanto fue necesario aplicar el valor absoluto al dato analizado o de lo contrario la media aritmética calculada sería cero o cercana a cero.

### **3.1.3.2 Umbrales de Pendiente o Derivada**

Este método consiste en hacer una evaluación permanente de la diferencia de amplitudes entre las muestras, y definir un límite o threshold que indicara a partir de que diferencia entre las amplitudes, una muestra será caracterizada como contaminada por artefacto. Por esta razón, la parte más significativa de este método es la definición de dicho threshold, puesto a que no hay documentado un criterio o umbral de pendiente establecido para la aplicabilidad de este método. Es decir que la definición del umbral de amplitud se dará de manera empírica teniendo como fundamentos el conocimiento de un especialista.

Para la implementación se requería hacer la diferencia entre dos muestras con el fin de calcular el valor exacto de amplitud en que ambas difieren y posteriormente comparar el valor obtenido con el threshold establecido. Si el valor de amplitud supera el umbral de pendiente, se considera que la muestra analizada está contaminada por un artefacto, por el contrario si el valor de amplitud obtenido de la diferencia entre las dos muestras no supera el umbral de pendiente se considera que es una muestra propia de la señal EEG y limpia de artefactos; Teniendo en cuenta lo anterior se implementó una ventana deslizante como se muestra en la Figura 32, la cual recorrerá todas las muestras contenidas por el epoch.

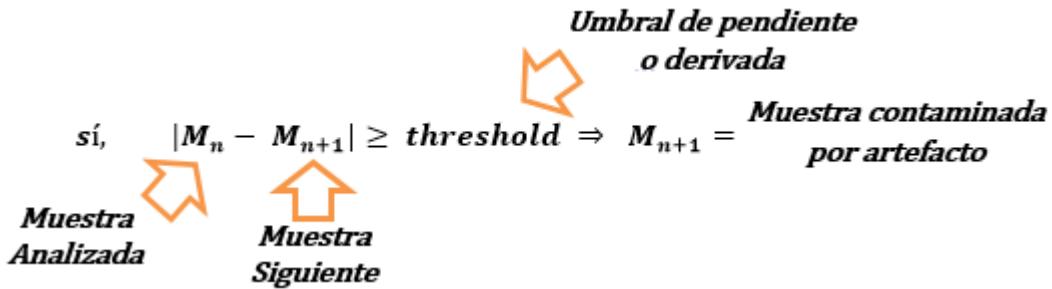
**Figura 32: Ventana deslizante método de la pendiente**



Fuente: De los Autores.

Con la ventana anterior se obtiene un dato y se analiza con respecto al dato siguiente en el registro del paciente, la diferencia entre ambas muestras se tomó en valor absoluto por la razón de que la señal tiene valores de amplitud tanto positivos como negativos. A diferencia de la detección de artefactos por umbral de amplitud, la detección por umbrales de pendiente no requiere de datos previos para poder realizarse un análisis lo que indica que el método es funcional desde el primer dato del registro EEG. Una manera de representar lo expuesto es la Figura 33.

**Figura 33: Ecuación método de la pendiente**



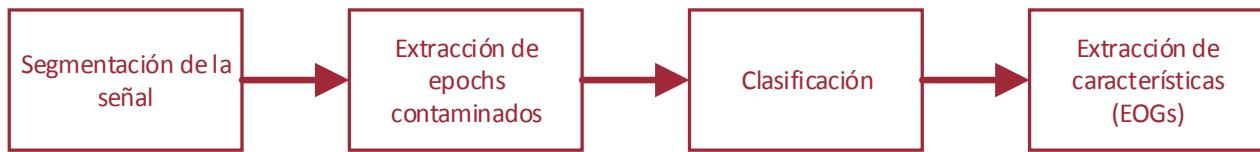
Fuente: De los Autores.

Este proceso deberá repetirse hasta haber evaluado todas las muestras contenidas por el epoch, lo que se consigue a través del ciclo o función `for()`, los índices de las muestras que son determinadas como contaminadas, son almacenados en una matriz, con el fin de obtener la posición exacta y este modo poder indicar todas las muestras dentro del epoch visualizado.

### 3.1.3.3 Método de Correlación

El método de correlación busca la semejanza entre la señal del EEG y una muestra patrón, en este caso se implementara este método para la detección de artefactos oculares EOG. La selección del patrón, para implementar el método de correlación, es fundamental. Para la selección del patrón se siguió el proceso mostrado en la Figura 34.

**Figura 34: Flujo para la selección del patrón**

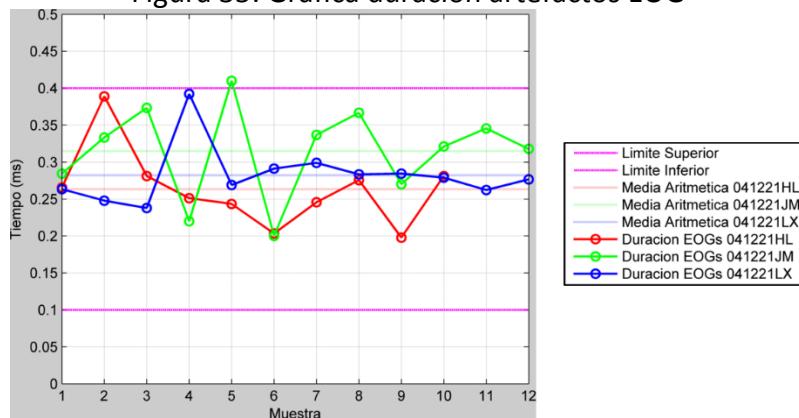


Fuente: De los Autores.

Cada registro EEG perteneciente a un paciente (se utilizaron los registros 041221HL, 041221JM y 041221LX) se segmentó en epochs de 2 segundos de duración (1800 muestras). Luego se seleccionaron los epochs contaminados por artefactos y se creó una base de datos de epochs contaminados clasificándolos por el tipo de artefacto presente. Después de consolidada la base de datos se prosiguió a extraer el segmento que contenía la señal EOG de cada epoch contaminado. La duración de este segmento fue seleccionada partiendo de los la base de datos de biológicos de la universidad de Harvard (Milo, 2008), según la cual la duración promedio de un parpadeo está en un rango de 0,1 a 0,4 segundos. Con esta información se realizó una verificación partiendo de la base de datos disponible y se graficaron las duraciones de los artefactos EOG de los registros 041221HL, 041221JM, 041221LX. Se seleccionaron estos registros debido a la fuerte incidencia de artefactos oculares y a la poca contaminación con componentes de 50Hz.

En la Figura 35 se graficaron las duraciones de 36 artefactos EOG, agrupados por paciente, al igual que los límites sugeridos por (Milo, 2008) y la media aritmética de cada grupo. Partiendo de los resultados obtenidos en la gráfica, se puede decir que el rango sugerido es correcto, solo uno de los datos supera el límite superior y la media aritmética de los datos es cercana a la mitad del rango.

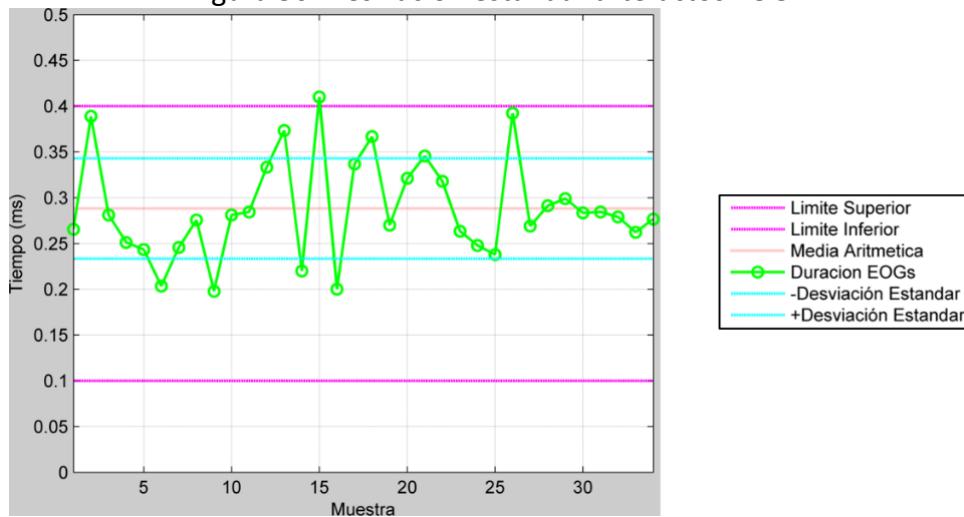
**Figura 35: Grafica duración artefactos EOG**



Fuente: De los Autores. Base de Datos registros 041221HL, JM y LX

Para analizar con más detalle los datos anteriores, no basta con conocer las medidas de tendencia central, sino que necesitamos conocer también la desviación que presentan los datos en su distribución respecto de la media aritmética de dicha distribución, con objeto de tener una visión de los mismos más acorde con la realidad al momento de describirlos e interpretarlos para tomar una decisión adecuada acerca de la duración del patrón. Por esto en la Figura 36 se grafican las duraciones de los artefactos EOG indistintamente del paciente, y se obtiene de estos datos la desviación estándar.

Figura 36: Desviación estándar artefactos EOG

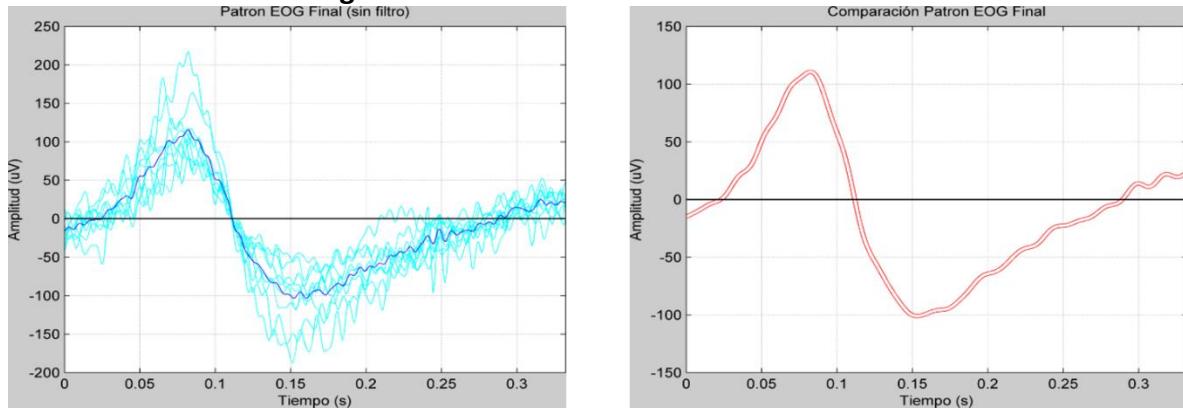


Fuente: De los Autores. Base de Datos registros 041221HL, JM y LX

Partiendo de la información de las figuras anteriores se estableció una duración de 0,343 segundos, correspondiente a la media aritmética más una desviación estándar. Esta duración corresponde aproximadamente a 300 datos del registro, ya que este tiene una frecuencia de muestreo de 900Hz.

Para la construcción del patrón se ajustaron los segmentos que contienen los artefactos EOG a una longitud de 300 datos, se debe tener en cuenta que un EOG típico no es simétrico, es decir, se es más corto en el primer semicírculo, por esto se debió tener especial cuidado al adaptar la señal a la longitud del patrón. Para lograr una buena adaptación, se toma un punto de referencia, el cual es el cruce por cero, se alinean las señales que contienen los artefactos (30 en total) y para finalizar se promedian todas las señales y obtenemos el patrón final, este se muestra en la **Figura 37**. Este será el patrón con el que se realizará la correlación cruzada para la detección de artefactos oculares en señales EEG.

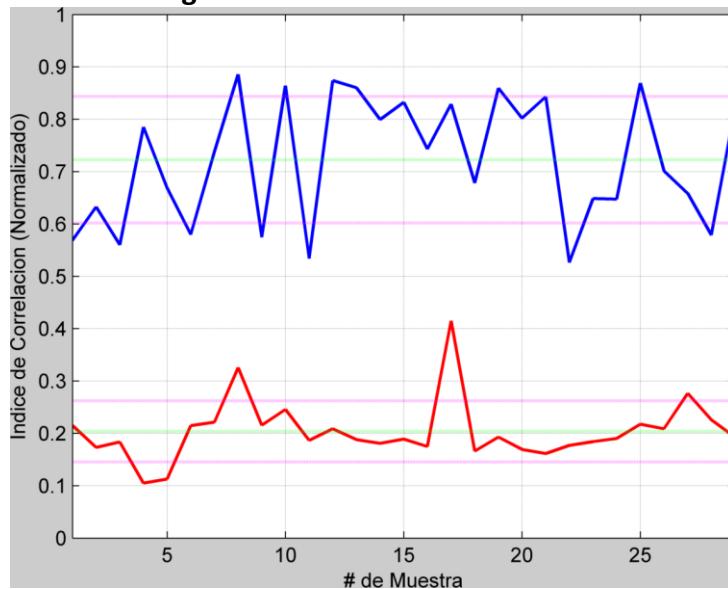
**Figura 37: Patrón final método correlación**



Fuente: De los Autores. Base de Datos registros 041221HL, JM y LX

Es necesario establecer un umbral de detección, por encima del cual el algoritmo de correlación determine los segmentos de la señal que están contaminados por artefactos. Si no se establece un umbral, todos los segmentos serán determinados como artefactos, pues en todos los casos el algoritmo calcula un índice de correlación. Para establecer el umbral, se sometieron 30 segmentos contaminados con artefactos EOG y 30 segmentos libres de artefactos al algoritmo de correlación. Los índices normalizados se muestran en la Figura 38.

**Figura 38: Índices de correlación**



Fuente: De los Autores. Base de Datos registros 041221HL, JM y LX

En azul se muestran los índices de correlación de los segmentos contaminados con artefactos, observando que el punto de máximo está en 0.886, el punto de mínimo es 0.526 y la media aritmética es 0.722. En rojo se grafican los índices de correlación de los 30 segmentos no contaminados por artefactos EOG, donde el punto de máximo está en

0.414, el punto de mínimo es 0.104 y la media aritmética es 0.204. Partiendo de esta grafica se puede decir que no hay superposición entre los índices de los dos grupos de segmentos, teniendo una diferencia de 0.112 entre mínimo y máximo, y 0,518 entre la media aritmética de cada uno de los grupos de segmentos. Por esto se estableció un umbral inicial de 0.5, es decir, cualquier segmento que después de ser evaluado por el algoritmo, dé como resultado un índice de correlación de 0,5, será considerado un artefacto ocular.

### 3.1.4 Resultados

#### 3.1.4.1 Método para la Validación de Resultados

De acuerdo con el antiguo filósofo griego Epicteto de Frigia (Hierápolis, 55 – Nicópolis, 135):

*“Las apariencias a la mente son de cuatro clases.*

*Cosas hay que son lo que parecen ser;  
o no lo son y no parecen serlo;  
o lo son y no parecen serlo;  
o no son y sí parecen serlo.*

*Es tarea del hombre sabio el decidir correctamente  
en todos esos casos”*

Para comprobar la validez de los métodos implementados, se empleó un estándar, el cual generalmente se aplica para comprobar la validez y fiabilidad de pruebas diagnósticas. Este estándar permite calcular la sensibilidad, especificidad y valores predictivos. Para aplicar este estándar, es necesario caracterizar visualmente los registros EEG y señalar los epochs contaminados con artefactos, después de realizado este procedimiento, se evalúa el registro aplicando los métodos implementados en las herramientas de simulación. Para calcular los valores de sensibilidad, especificidad y valores predictivos es necesario clasificar los resultados en 4 grupos, cada uno de ellos se explica en la Tabla 2

**Tabla 2: Clasificación de resultados**

	Artefacto Detectado por el método de correlación	Artefacto NO Detectado por el método de correlación
<b>Epoch con presencia de artefacto EOG en la inspección visual</b>	Verdadero Positivo VP	Falso Negativo FN
<b>Epoch sin presencia de artefacto EOG en la inspección visual</b>	Falso Positivo FP	Verdadero Negativo VN

### *Sensibilidad*

Es la probabilidad de clasificar correctamente en un epoch la presencia de un artefacto EOG, es decir, la probabilidad de que para un epoch contaminado se obtenga en la prueba con el algoritmo de correlación resultado positivo. La sensibilidad es, por lo tanto, la capacidad del algoritmo para detectar artefactos EOG. Partiendo de la clasificación de epochs en la Tabla 4, es fácil estimar a partir de ella la sensibilidad como la proporción de artefactos detectados que obtuvieron un resultado positivo en la prueba. Es decir:

$$\text{Sensibilidad} = \frac{VP}{VP + FN} \quad (12)$$

El valor que puede asumir la sensibilidad varía del 0 al 1 (100%), es decir, cuanto más alto es el valor, hay una mejor capacidad en la detección de artefactos por medio de la prueba. Una sensibilidad baja produce como resultado un bajo desempeño del algoritmo al momento de detectar artefactos EOG.

### *Especificidad*

Es la probabilidad de clasificar correctamente a un epoch libre de artefactos, es decir, la probabilidad de que para un epoch de la señal EEG libre de artefactos EOG se obtenga un resultado negativo. En otras palabras, se puede definir la especificidad como la capacidad para detectar los epochs sin contaminación de artefactos EOG. A partir de la Tabla 4, la especificidad se estimaría como:

$$\text{Especificidad} = \frac{VN}{VN + FP} \quad (13)$$

Al igual que la sensibilidad, el valor de la especificidad varía del 0 al 1 (100%), lo que significa que cuanto mayor sea el valor, mayor es la capacidad de detección de epochs libres de artefactos EOG por el método de correlación.

Todo el propósito de una prueba diagnóstica es utilizar sus resultados para hacer un diagnóstico, por lo que necesita saber la probabilidad de que el resultado de la prueba dará el diagnóstico correcto (Akobeng, 2006). Por esto se han desarrollado otros parámetros para poder determinar qué tanta validez tiene un método al ser utilizado como prueba diagnóstico. Entre estos parámetros se encuentran los valores predictivos positivos y negativos.

### *Valor Predictivo Positivo VPP*

El valor predictivo positivo (VPP) o índice de precisión, o probabilidad posterior a la prueba de la enfermedad de una prueba, se define como la proporción de personas con un resultado positivo de la prueba que realmente tienen la enfermedad. En nuestro caso

la proporción de epochs clasificados como contaminados que realmente tienen presencia de artefactos EOG. A partir de la Tabla 4 el VPP se calcula:

$$VPP = \frac{VP}{VP + FP} \quad (14)$$

#### *Valor Predictivo Negativo VPN*

El VPN de una prueba es la proporción de personas con un resultado negativo de la prueba que no tienen la enfermedad. El VPN de una prueba es la proporción de personas con un resultado negativo de la prueba que no tienen la enfermedad (Akobeng, 2006). En nuestro caso la proporción de epochs clasificados como libres de artefactos que realmente no tienen presencia de artefactos EOG. A partir de la Tabla 4 el VPN se calcula:

$$VPN = \frac{FN}{FN + VN} \quad (15)$$

#### *Exactitud*

La exactitud es la proporción de los resultados verdaderos, ya sea verdaderos positivos o verdaderos negativos, en una población. Mide el grado de veracidad de una prueba de diagnóstico de una condición (Zhu, Zeng, & Wang, 2010). A partir de la Tabla 4 el VPN se calcula:

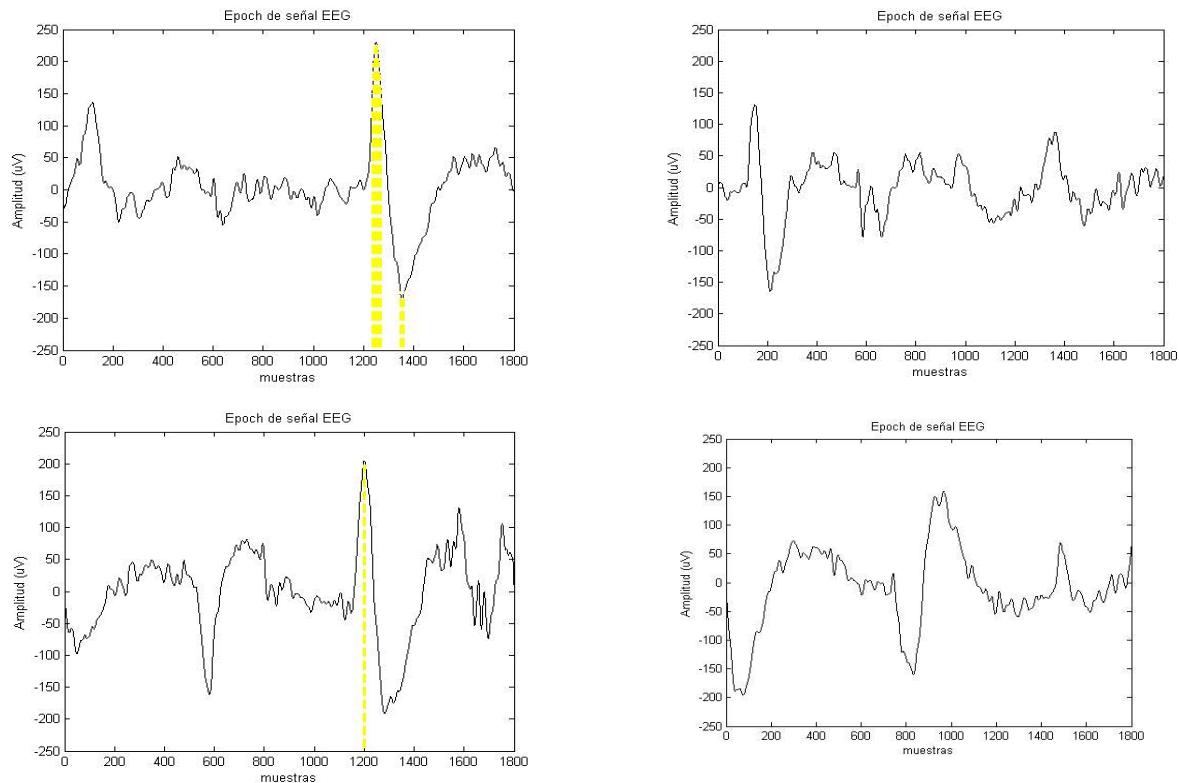
$$Exactitud = \frac{VP + VN}{VP + VN + FN + TN} \quad (16)$$

### **3.1.5 Resultados de la Implementación de Métodos en MatLab**

#### **3.1.5.1 Resultados Simulados de Threshold de Amplitud en MatLab**

Para cuantificar la efectividad del método threshold en amplitud se creó una base de datos que contiene 50 epochs contaminados con artefactos oculares, de este modo podemos analizar cuantos epochs detecta como artefactos el método con un threshold de 6. A continuación se muestran algunos ejemplos de artefactos detectados versus algunos no detectados en la Figura 39.

**Figura 39: Detección de artefactos a través del método threshold de amplitud con un threshold de 6**



Fuente: De los Autores. Base de Datos registros 041221HL, JM y LX

Como se puede apreciar en la figura anterior existe la presencia de artefactos oculares en algunos epochs que no es identificado por el método; en exactitud el método threshold de amplitud detectó 14 epochs contaminados con artefactos de amplitud de una totalidad de 50 epochs contaminados, esto indica que los índices de efectividad son de 28% lo que demuestra una muy baja eficiencia. A continuación en la Tabla 3 se muestra los epochs detectados como artefactos con el fin de validar los resultados expuestos anteriormente.

**Tabla 3: Resultados threshold de amplitud en MatLab**

THRESHOLD DE AMPLITUD RESULTADOS MATLAB				
THRESHOLD 6.0	EPOCH Nro.	INTERVALO DE TIEMPO	ARTEFACTOS DETECTADOS	EFFECTIVIDAD
	1	0 sg - 2 sg	NA	28 %
	2	2 sg - 4 sg	NA	
	3	4 sg - 6 sg	NA	
	4	6 sg - 8 sg	NA	
	5	8 sg - 10 sg	NA	
	6	10 sg - 12 sg	SI	
	7	12 sg - 14 sg	NO	
	8	14 sg - 16 sg	SI	
	9	16 sg - 18 sg	NO	
	10	18 sg - 20 sg	SI	
	11	20 sg - 22 sg	SI	
	12	22 sg - 24 sg	NO	
	13	24 sg - 26 sg	SI	
	14	26 sg - 28 sg	NO	
	15	28 sg - 30 sg	SI	
	16	30 sg - 32 sg	SI	
	17	32 sg - 34 sg	NO	
	18	34 sg - 36 sg	NO	
	19	36 sg - 38 sg	NO	
	20	38 sg - 40 sg	NO	
	21	40 sg - 42 sg	SI	
	22	42 sg - 44 sg	NO	
	23	44 sg - 46 sg	NO	
	24	46 sg - 48 sg	NO	
	25	48 sg - 50 sg	SI	
	26	50 sg - 52 sg	NO	
	27	52 sg - 54 sg	SI	
	28	54 sg - 56 sg	SI	
	29	56 sg - 58 sg	NO	
	30	58 sg - 60 sg	NO	
	31	60 sg - 62 sg	NO	
	32	62 sg - 64 sg	NO	
	33	64 sg - 66 sg	NO	
	34	66 sg - 68 sg	NO	
	35	68 sg - 70 sg	NO	
	36	70 sg - 72 sg	NO	
	37	72 sg - 74 sg	NO	
	38	74 sg - 76 sg	NO	
	39	76 sg - 78 sg	NO	
	40	78 sg - 80 sg	NO	
	41	80 sg - 82 sg	NO	
	42	82 sg - 84 sg	NO	
	43	84 sg - 86 sg	NO	
	44	86 sg - 88 sg	NO	
	45	88 sg - 90 sg	NO	
	46	90 sg - 92 sg	NO	
	47	92 sg - 94 sg	NO	
	48	94 sg - 96 sg	NO	
	49	96 sg - 98 sg	NO	
	50	98 sg - 100 sg	NO	
	51	100 sg - 102 sg	NO	
	52	102 sg - 104 sg	SI	
	53	104 sg - 106 sg	SI	
	54	106 sg - 108 sg	NO	
	55	108 sg - 110 sg	SI	
	TOTAL DETECTADOS		14	

En la Tabla 3 se puede ver los 10 segundos (5 epochs) necesarios para el correcto funcionamiento del método y a los cuales no se aplica el método de umbral de amplitud. Al obtener los resultados mencionados anteriormente se replanteo el método y se decidió cambiar de manera empírica y con ayuda de un especialista el threshold con el objetivo de mejorar la efectividad del método aplicado.

En vista a los resultados obtenidos y en pro de implementar un método más robusto en cuanto a la efectividad y sensibilidad al momento de detectar los artefactos presentes en los epochs, se optó en primera instancia por realizar inspecciones visuales directamente a dos registros cualesquiera de los suministrados con el fin de definir de manera manual epochs verdaderamente contaminados por artefactos y analizar el comportamiento del método directamente en una señal real con diferentes valores de threshold, como se verifico la baja eficiencia del método con un valor de 6, se seleccionaron valores menores determinantes entre el rango [3,5 -6], el valor mínimo seleccionado de 3,5 se debe a que un valor menor a ese provocaría más detecciones erradas que verdaderas, en la tabla que se ilustra a continuación se indican los resultados obtenidos referentes a los registros seleccionados, las inspección visual, los epochs determinados como contaminados por artefactos y la efectividad del método al detectarlos.

**Tabla 4: Resultados método de threshold de amplitud en MatLab**

RESULTADOS DETECCIÓN DE ARTEFACTOS MÉTODO THRESHOLD DE AMPLITUD EN MATLAB®												
Epoch	Registro 041222F9.bin						Registro 041222I6.bin					
	Inspección Visual	Matlab 3,5	Matlab 3,9	Matlab 4,3	Matlab 5,0	Matlab 6,0	Inspección Visual	Matlab 3,5	Matlab 3,9	Matlab 4,3	Matlab 5,0	Matlab 6,0
0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
1	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI
2	NO	NO	NO	NO	NO	NO	SI	SI	SI	SI	SI	NO
3	SI	SI	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO
4	NO	SI	NO	NO	NO	NO	SI	NO	NO	NO	NO	NO
5	SI	SI	NO	NO	NO	NO	SI	SI	SI	NO	NO	NO
6	SI	SI	SI	SI	SI	NO	SI	SI	NO	NO	NO	NO
7	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI
8	SI	SI	SI	SI	NO	NO	SI	SI	SI	SI	SI	NO
9	SI	SI	SI	SI	SI	SI	SI	NO	NO	NO	NO	NO
10	NO	NO	NO	NO	NO	NO	SI	SI	NO	NO	NO	NO
11	NO	NO	NO	NO	NO	NO	SI	SI	SI	SI	SI	SI
12	NO	NO	NO	NO	NO	NO	SI	SI	NO	NO	NO	NO
13	SI	SI	SI	NO	NO	NO	SI	SI	NO	NO	NO	NO
14	NO	SI	SI	SI	SI	NO	NO	NO	NO	NO	NO	NO
15	NO	SI	SI	SI	NO	NO	SI	SI	SI	SI	SI	SI
16	NO	SI	SI	SI	NO	NO	SI	SI	SI	SI	SI	SI
17	NO	SI	SI	NO	NO	NO	SI	SI	SI	SI	SI	NO
18	NO	SI	SI	SI	SI	NO	SI	SI	SI	SI	SI	SI
19	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI
20	NO	SI	SI	NO	NO	NO	NO	NO	NO	NO	NO	NO
21	NO	SI	NO	NO	NO	NO	SI	SI	SI	SI	SI	SI
22	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	NO
23	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI
24	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	NO
25	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI
TOTAL DETECTADOS	13	21	18	15	12	9	23	20	16	15	14	10

Los colores en la Tabla 4 anterior hacen alusión a los valores de threshold para los cuales el método presenta un desempeño cercano al deseado; las celdas en color gris hacen referencia a los valores que difieren entre la inspección visual que sería el resultado del método al que se desea llegar, es decir “optimo” y el resultado del método propio del valor de threshold aplicado

Posteriormente se procedió a evaluar la validez del método para los resultados propios a cada valor de threshold, por tratarse de pruebas, lo que conlleva a calcular los valores de sensibilidad, especificidad, valor predictivo positivo y valor predictivo, pero para ello es necesario previamente hallar los valores de Falsos Positivos (FP), Falsos Negativos (FN), Verdaderos Positivos (VP) y Verdaderos Negativos (VN); dichos resultados son mostrados a continuación en la Tabla 5.

**Tabla 5: Clasificación de resultados threshold de amplitud en MatLab**

	Registro 041222F9.bin					Registro 041222I6.bin				
	3,5	3,9	4,3	5,0	6,0	3,5	3,9	4,3	5,0	6,0
Valor de Threshold	3,5	3,9	4,3	5,0	6,0	3,5	3,9	4,3	5,0	6,0
Falso Positivo FP	9	6	4	2	0	0	0	0	0	0
Falso Negativo FN	0	1	2	3	4	3	7	8	9	13
Verdadero Positivo VP	12	12	11	10	9	20	16	15	14	10
Verdadero Negativo VN	4	6	8	10	12	2	2	2	2	2

En base a los resultados anteriormente ilustrados se calcularon los valores de sensibilidad, especificidad, valor predictivo positivo y valor predictivo y asimismo a través de los resultados se definió el valor de threshold más conveniente los resultados se indican en la Tabla 6

**Tabla 6: Resultados evaluación método de threshold de amplitud**

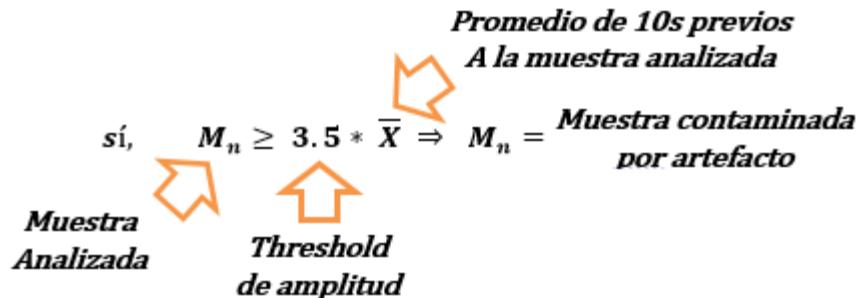
	Registro 041222F9.bin					Registro 041222I6.bin				
	3,5	3,9	4,3	5,0	6,0	3,5	3,9	4,3	5,0	6,0
Valor de Threshold	3,5	3,9	4,3	5,0	6,0	3,5	3,9	4,3	5,0	6,0
Sensibilidad	100%	92%	85%	77%	69%	87%	70%	65%	61%	43%
Especificidad	31%	50%	67%	83%	100%	100%	100%	100%	100%	100%
Valor Predictivo Positivo	57%	67%	73%	83%	100%	100%	100%	100%	100%	100%
Valor Predictivo Negativo	0%	14%	20%	23%	25%	60%	78%	80%	82%	87%
Exactitud	64%	72%	76%	80%	84%	88%	72%	68%	64%	48%
Exactitud Promedio	76%	72%	72%	72%	66%	76%	72%	72%	72%	66%
Valor de Threshold Definido	✓					✓				

Como se puede visualizar en la Tabla 6 el valor seleccionado como el apropiado es un threshold de 3,5 pues es el valor que muestra una mejor combinación de los índices anteriores, si bien es cierto que dicho Threshold conlleva a detecciones erradas como se puede apreciar en su valor de Falsos Positivos de 9. Es el valor que mayor sensibilidad presenta para ambos registros sin olvidar que en una prueba médica es valedero de cierto modo que una prueba general detecte anomalías que posiblemente no existan (Falsos Positivos) a que no detecte anomalías que sí están presentes (Falsos Negativos), debido a que estas posibles detecciones erradas (Falsos Positivos) podrán ser descartadas con pruebas específicas mucho más profundas.

Aunque un valor menor de Threshold aumentarían mucho más los índices de detección, es importante recordar que de igual proporción se incrementarían las detecciones erradas del método y es de aclarar que tampoco es eficiente una prueba que presente demasiados resultados equívocos, debido a que se aumentarían en gran medida los diagnósticos errados lo que a su vez repercutiría en gastos económicos desmesurados y otros factores como, pérdida de tiempo, afectación directa del paciente, entre otros.

En relación a todo lo anteriormente expuesto, el método de Threshold de Amplitud queda acondicionado como se muestra en la Figura 40.

**Figura 40: Ajuste para el método de amplitud**



Fuente: De los Autores.

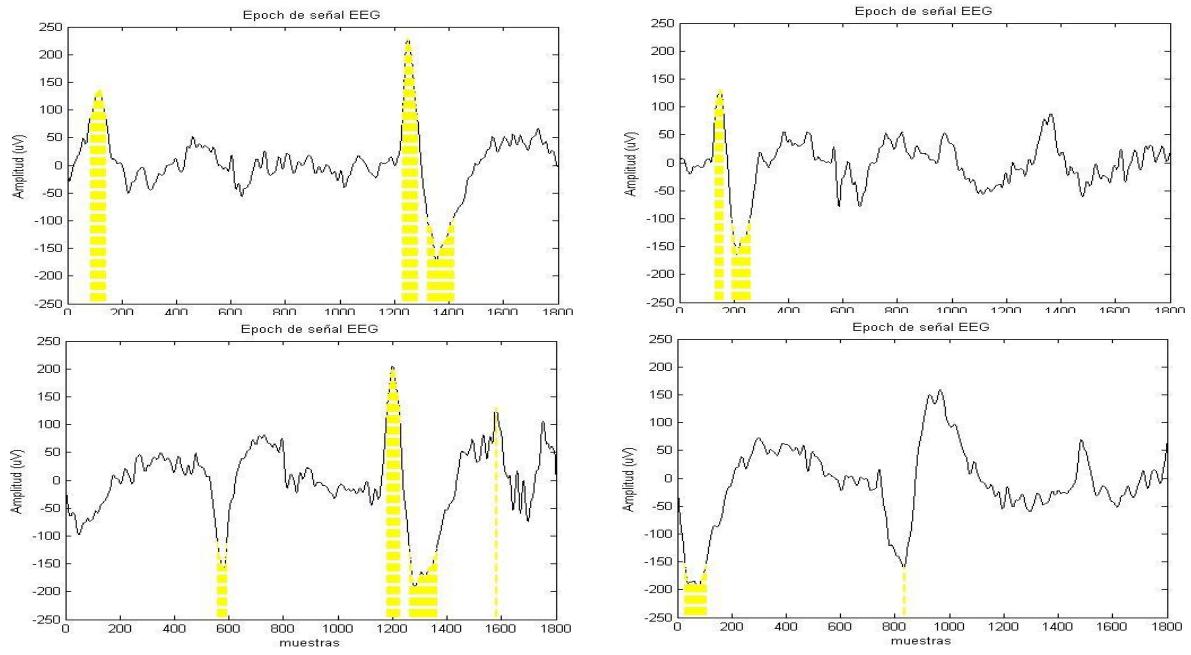
Con el fin de verificar la variación de efectividad del método, nuevamente se aplicó con el nuevo valor del Threshold a la base de datos de artefactos y se obtuvo un índice de detección de 86% ya que pudieron detectar 43 epochs contaminados con artefactos oculares (EOG) de 50; se ilustra a continuación la Tabla 7.

**Tabla 7: Resultados threshold de amplitud en MatLab**

THRESHOLD DE AMPLITUD RESULTADOS MATLAB				
THRESHOLD 3.5	EPOCH Nro.	INTERVALO DE TIEMPO	ARTEFACTOS DETECTADOS	EFFECTIVIDAD
	1	0 sg - 2 sg	NA	86 %
	2	2 sg - 4 sg	NA	
	3	4 sg - 6 sg	NA	
	4	6 sg - 8 sg	NA	
	5	8 sg - 10 sg	NA	
	6	10 sg - 12 sg	SI	
	7	12 sg - 14 sg	SI	
	8	14 sg - 16 sg	SI	
	9	16 sg - 18 sg	SI	
	10	18 sg - 20 sg	SI	
	11	20 sg - 22 sg	SI	
	12	22 sg - 24 sg	SI	
	13	24 sg - 26 sg	SI	
	14	26 sg - 28 sg	SI	
	15	28 sg - 30 sg	SI	
	16	30 sg - 32 sg	SI	
	17	32 sg - 34 sg	SI	
	18	34 sg - 36 sg	SI	
	19	36 sg - 38 sg	NO	
	20	38 sg - 40 sg	NO	
	21	40 sg - 42 sg	SI	
	22	42 sg - 44 sg	SI	
	23	44 sg - 46 sg	SI	
	24	46 sg - 48 sg	SI	
	25	48 sg - 50 sg	SI	
	26	50 sg - 52 sg	SI	
	27	52 sg - 54 sg	SI	
	28	54 sg - 56 sg	SI	
	29	56 sg - 58 sg	SI	
	30	58 sg - 60 sg	SI	
	31	60 sg - 62 sg	SI	
	32	62 sg - 64 sg	SI	
	33	64 sg - 66 sg	SI	
	34	66 sg - 68 sg	SI	
	35	68 sg - 70 sg	SI	
	36	70 sg - 72 sg	SI	
	37	72 sg - 74 sg	SI	
	38	74 sg - 76 sg	NO	
	39	76 sg - 78 sg	SI	
	40	78 sg - 80 sg	SI	
	41	80 sg - 82 sg	NO	
	42	82 sg - 84 sg	SI	
	43	84 sg - 86 sg	SI	
	44	86 sg - 88 sg	SI	
	45	88 sg - 90 sg	NO	
	46	90 sg - 92 sg	NO	
	47	92 sg - 94 sg	SI	
	48	94 sg - 96 sg	NO	
	49	96 sg - 98 sg	SI	
	50	98 sg - 100 sg	SI	
	51	100 sg - 102 sg	SI	
	52	102 sg - 104 sg	SI	
	53	104 sg - 106 sg	SI	
	54	106 sg - 108 sg	SI	
	55	108 sg - 110 sg	SI	
	TOTAL DETECTADOS		43	

Los nuevos resultados con un threshold de 3.5 se pueden apreciar a continuación en la Figura 41.

**Figura 41: Detección de artefactos a través del método threshold de amplitud con un threshold de 3.5**



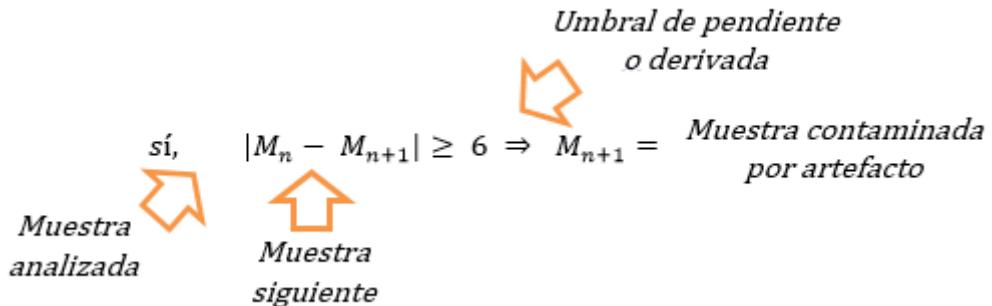
Fuente: De los Autores. Base de Datos registros 041221HL, JM y LX

### 3.1.5.2 Resultados Simulados de Threshold de Pendiente en MatLab

A diferencia del método umbral de amplitud, el umbral por pendiente no establece un threshold para la detección de la actividad rápida, es decir, el threshold será definido de manera empírica luego de haber realizado pruebas al ensayo y error y haber analizado el comportamiento del método según los umbrales planteados.

En primera instancia se fijó realizar pruebas con un threshold equivalente al del método de umbral de amplitud, es decir un threshold de 6, la expresión que obedece a lo anterior se puede observar con la Figura 42.

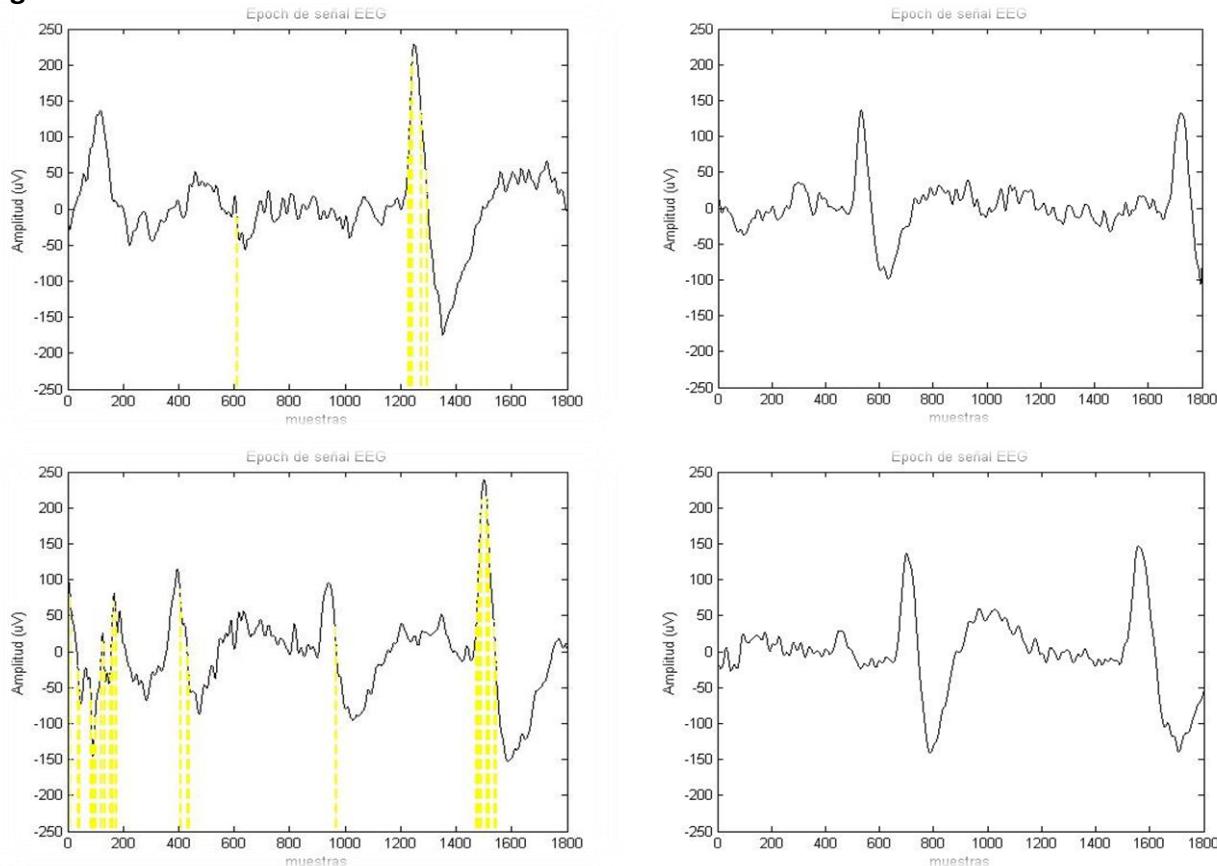
**Figura 42: Parámetros iniciales método de la pendiente**



Fuente: De los Autores.

Se aplicó el método a la base de datos con 50 epochs contaminados por artefactos oculares, de la cual se ha hecho mención con anterioridad y se lograron detectar 36 epochs contaminados es decir, un threshold de valor 6 da como resultado un índice de efectividad del 72%, el cual es relativamente bajo, a continuación se ilustran Figura 43 de algunos epochs detectados y no detectados

**Figura 43: Detección de artefactos a través del método threshold de Pendiente con un threshold de 6**



**Tabla 8: Resultados threshold de pendiente en MatLab**

THRESHOLD DE PENDIENTE RESULTADOS MATLAB				
THRESHOLD 6.0	EPOCH Nro.	INTERVALO DE TIEMPO	ARTEFACTOS DETECTADOS	EFFECTIVIDAD
	1	0 sg - 2 sg	SI	74 %
	2	2 sg - 4 sg	SI	
	3	4 sg - 6 sg	SI	
	4	6 sg - 8 sg	SI	
	5	8 sg - 10 sg	SI	
	6	10 sg - 12 sg	SI	
	7	12 sg - 14 sg	NO	
	8	14 sg - 16 sg	SI	
	9	16 sg - 18 sg	SI	
	10	18 sg - 20 sg	SI	
	11	20 sg - 22 sg	SI	
	12	22 sg - 24 sg	NO	
	13	24 sg - 26 sg	SI	
	14	26 sg - 28 sg	NO	
	15	28 sg - 30 sg	NO	
	16	30 sg - 32 sg	SI	
	17	32 sg - 34 sg	SI	
	18	34 sg - 36 sg	NO	
	19	36 sg - 38 sg	NO	
	20	38 sg - 40 sg	SI	
	21	40 sg - 42 sg	SI	
	22	42 sg - 44 sg	SI	
	23	44 sg - 46 sg	SI	
	24	46 sg - 48 sg	SI	
	25	48 sg - 50 sg	SI	
	26	50 sg - 52 sg	SI	
	27	52 sg - 54 sg	SI	
	28	54 sg - 56 sg	SI	
	29	56 sg - 58 sg	SI	
	30	58 sg - 60 sg	SI	
	31	60 sg - 62 sg	SI	
	32	62 sg - 64 sg	SI	
	33	64 sg - 66 sg	SI	
	34	66 sg - 68 sg	SI	
	35	68 sg - 70 sg	SI	
	36	70 sg - 72 sg	NO	
	37	72 sg - 74 sg	SI	
	38	74 sg - 76 sg	SI	
	39	76 sg - 78 sg	SI	
	40	78 sg - 80 sg	SI	
	41	80 sg - 82 sg	NO	
	42	82 sg - 84 sg	NO	
	43	84 sg - 86 sg	NO	
	44	86 sg - 88 sg	NO	
	45	88 sg - 90 sg	SI	
	46	90 sg - 92 sg	NO	
	47	92 sg - 94 sg	SI	
	48	94 sg - 96 sg	SI	
	49	96 sg - 98 sg	NO	
	50	98 sg - 100 sg	SI	
	TOTAL DETECTADOS		37	

En la Tabla 8 se puede visualizar que la totalidad de epochs es de 50, es decir solo se aplicó el método a la base de artefactos EOG debido a que el método Threshold de Pendiente a diferencia del Threshold de Amplitud no requiere de 10 segundo previos para su correcta aplicabilidad, esto quiere decir que los primeros 5 epochs son excluidos del análisis puesto a que hacen referencia a señal limpia de EEG.

Con el fin de mejorar los índices de efectividad y detectar una mayor presencia de se optó por fijar un nuevo threshold, para ello se hizo uso nuevamente del test de validez para seleccionar un valor apropiado.

**Tabla 9: Resultados detección de artefactos método threshold de amplitud en MatLab**

Epoch	Registro 041222F9.bin					Registro 041222I6.bin				
	Inspección Visual	Matlab 4,0	Matlab 4,4	Matlab 5,0	Matlab 6,0	Inspección Visual	Matlab 4,0	Matlab 4,4	Matlab 5,0	Matlab 6,0
1	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI
2	NO	SI	SI	NO	NO	SI	SI	SI	SI	SI
3	SI	SI	SI	SI	SI	SI	SI	SI	SI	NO
4	NO	SI	SI	SI	SI	SI	SI	NO	NO	NO
5	SI	SI	SI	SI	SI	SI	SI	SI	SI	NO
6	SI	SI	SI	SI	SI	SI	SI	SI	SI	NO
7	SI	SI	SI	SI	SI	SI	SI	SI	SI	NO
8	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI
9	SI	SI	SI	SI	NO	SI	SI	SI	NO	NO
10	NO	SI	SI	SI	NO	SI	SI	SI	SI	NO
11	NO	SI	SI	SI	NO	SI	SI	SI	SI	SI
12	NO	NO	NO	NO	NO	SI	SI	SI	SI	NO
13	SI	NO	NO	NO	NO	SI	SI	NO	NO	NO
14	NO	NO	NO	NO	NO	NO	SI	SI	SI	NO
15	NO	NO	NO	NO	NO	SI	SI	SI	SI	SI
16	NO	SI	NO	NO	NO	SI	SI	SI	SI	NO
17	NO	SI	NO	NO	NO	SI	NO	NO	NO	NO
18	NO	SI	SI	NO	NO	SI	SI	SI	SI	NO
19	SI	NO	NO	NO	NO	SI	SI	SI	SI	NO
20	NO	SI	NO	NO	NO	NO	NO	NO	NO	NO
21	NO	NO	NO	NO	NO	SI	SI	NO	NO	NO
22	SI	SI	SI	SI	SI	SI	SI	NO	NO	NO
23	SI	SI	SI	SI	NO	SI	SI	SI	SI	NO
24	SI	SI	SI	SI	SI	SI	SI	SI	NO	NO
25	SI	SI	SI	SI	SI	SI	SI	SI	SI	NO
TOTAL DETECTADOS	13	19	16	14	10	23	23	19	17	5

Nuevamente y resaltados de color gris, se pueden apreciar los resultados que difieren con las inspección visual, esto con el fin de facilitar los cálculos de FP, FN, VP y VN, los cuales se indican en la Tabla 10.

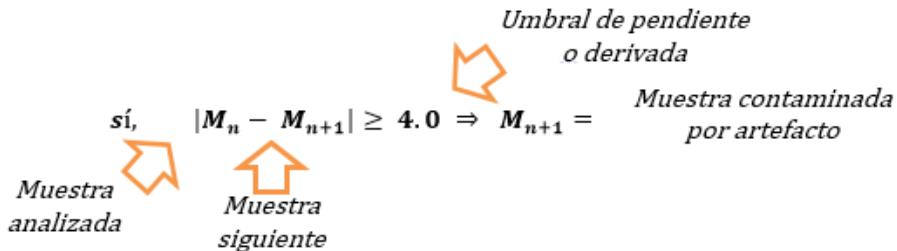
**Tabla 10: Clasificación de resultados método threshold de amplitud en MatLab**

Valor de Threshold	Registro 041222F9.bin				Registro 041222I6.bin			
	4	4,4	5,0	6,0	4	4,4	5,0	6,0
Falso Positivo FP	8	5	3	1	1	1	0	0
Falso Negativo FN	2	2	2	4	1	5	7	18
Verdadero Positivo VP	11	11	11	9	22	18	17	5
Verdadero Negativo VN	4	7	9	11	1	1	1	2

	Registro 041222F9.bin				Registro 041222I6.bin			
Valor de Threshold	4,0	4,4	5,0	6,0	4,0	4,4	5,0	6,0
Sensibilidad	85%	85%	85%	69%	96%	78%	71%	22%
Especificidad	33%	58%	75%	92%	50%	50%	100%	100%
Valor Predictivo Positivo	58%	69%	79%	90%	96%	95%	100%	100%
Valor Predictivo Negativo	33%	22%	18%	27%	50%	83%	88%	90%
Exactitud	60%	72%	80%	80%	92%	76%	72%	28%
Exactitud Promedio	76%	74%	76%	54%	76%	74%	76%	54%
Valor de Threshold Definido	✓				✓			

Basados en los resultados que fueron obtenidos al variar el valor del threshold se definió como un threshold de 4.0 como valor contundente, la representación que corresponde al cambio realizado en el método se muestra en la Figura 44.

**Figura 44: Ajuste para el método de pendiente**



Fuente: De los Autores.

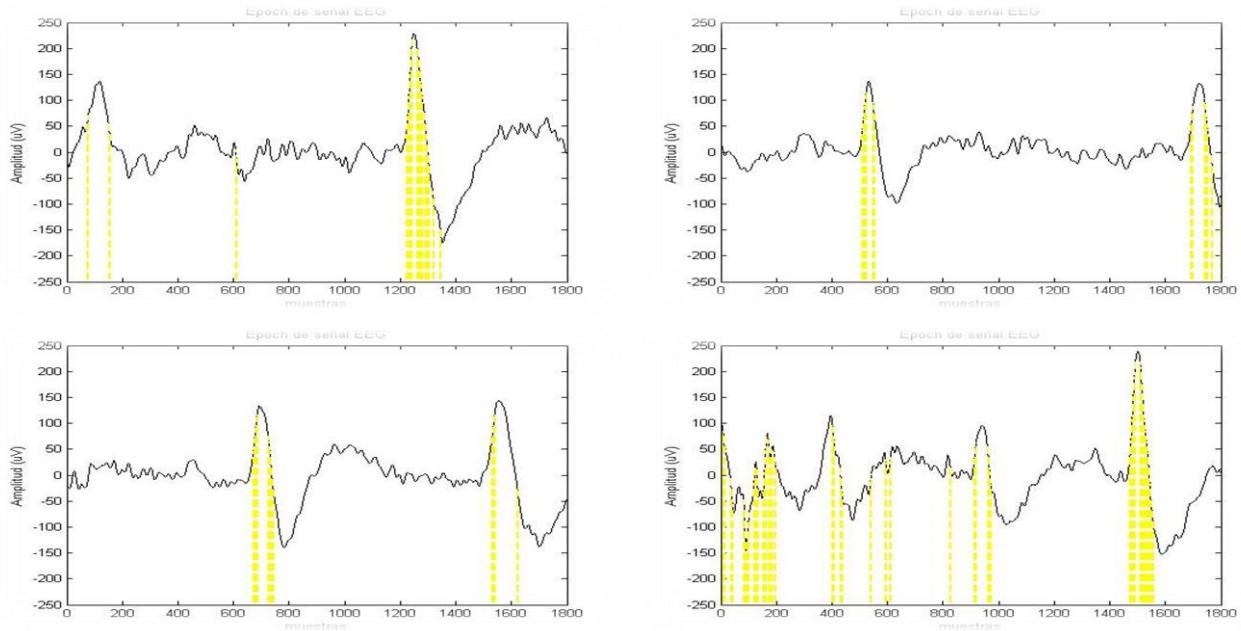
El método responde con una efectividad de 96% a este valor, pues 48 de los 50 epochs contaminados, en la Tabla 11 se ilustran los resultados obtenidos.

**Tabla 11: Resultados método threshold de pendiente en MatLab valor de threshold de 4.0**

THRESHOLD DE PENDIENTE RESULTADOS MATLAB				
THRESHOLD 4.0	EPOCH Nro.	INTERVALO DE TIEMPO	ARTEFACTOS DETECTADOS	EFFECTIVIDA D
	1	0 sg - 2 sg	SI	96 %
	2	2 sg - 4 sg	SI	
	3	4 sg - 6 sg	SI	
	4	6 sg - 8 sg	SI	
	5	8 sg - 10 sg	SI	
	6	10 sg - 12 sg	SI	
	7	12 sg - 14 sg	SI	
	8	14 sg - 16 sg	SI	
	9	16 sg - 18 sg	SI	
	10	18 sg - 20 sg	SI	
	11	20 sg - 22 sg	SI	
	12	22 sg - 24 sg	SI	
	13	24 sg - 26 sg	SI	
	14	26 sg - 28 sg	SI	
	15	28 sg - 30 sg	NO	
	16	30 sg - 32 sg	SI	
	17	32 sg - 34 sg	SI	
	18	34 sg - 36 sg	SI	
	19	36 sg - 38 sg	SI	
	20	38 sg - 40 sg	SI	
	21	40 sg - 42 sg	SI	
	22	42 sg - 44 sg	SI	
	23	44 sg - 46 sg	SI	
	24	46 sg - 48 sg	SI	
	25	48 sg - 50 sg	SI	
	26	50 sg - 52 sg	SI	
	27	52 sg - 54 sg	SI	
	28	54 sg - 56 sg	SI	
	29	56 sg - 58 sg	SI	
	30	58 sg - 60 sg	SI	
	31	60 sg - 62 sg	SI	
	32	62 sg - 64 sg	SI	
	33	64 sg - 66 sg	SI	
	34	66 sg - 68 sg	SI	
	35	68 sg - 70 sg	SI	
	36	70 sg - 72 sg	SI	
	37	72 sg - 74 sg	SI	
	38	74 sg - 76 sg	SI	
	39	76 sg - 78 sg	SI	
	40	78 sg - 80 sg	SI	
	41	80 sg - 82 sg	NO	
	42	82 sg - 84 sg	SI	
	43	84 sg - 86 sg	SI	
	44	86 sg - 88 sg	SI	
	45	88 sg - 90 sg	SI	
	46	90 sg - 92 sg	SI	
	47	92 sg - 94 sg	SI	
	48	94 sg - 96 sg	SI	
	49	96 sg - 98 sg	SI	
	50	98 sg - 100 sg	SI	
	TOTAL DETECTADOS		48	

En la Figura 45, se muestran algunas de las imágenes alusivas a la tabla anterior.

Figura 45: *Detección de artefactos a través del método threshold de Pendiente con un threshold de 4.0*



Fuente: De los Autores.

### 3.1.5.3 Resultados Simulados del Método de Correlación en MatLab

Esta prueba se realiza para confirmar o descartar la presencia de artefactos EOG en la señal EEG analizada. Lo ideal sería que este método identifique correctamente todos los epochs contaminados con artefactos, y de manera similar identificar correctamente todos los epochs que están libres de artefactos. Para tener la posibilidad de comparar resultados, se realizaron pruebas utilizando 4 umbrales de detección diferentes (0.47, 0.5, 0.55, 0.6) en dos registros de señal EEG (041222I6 y 041222F9), los resultados se muestran en la Tabla 12.

Tabla 12: Resultados evaluación de registros con método de correlación

Epoch	Registro 041222F9.bin			Registro 041222I6.bin		
	Inspección Visual	Python 0,47	Python 0,5	Inspección Visual	Python 0,47	Python 0,5
1	NO	SI	SI	SI	SI	SI
2	NO	SI	NO	NO	SI	NO
3	SI	NO	NO	SI	SI	SI
4	NO	NO	NO	SI	SI	SI
5	NO	NO	NO	SI	SI	SI

6	SI	SI	SI	SI	SI	SI
7	SI	SI	SI	SI	SI	SI
8	SI	SI	SI	SI	SI	SI
9	NO	SI	SI	NO	SI	NO
10	NO	NO	NO	SI	SI	SI
11	NO	NO	NO	SI	SI	SI
12	NO	SI	SI	SI	SI	SI
13	NO	SI	SI	SI	SI	SI
14	NO	SI	SI	NO	NO	NO
15	NO	NO	NO	SI	SI	SI
16	NO	NO	NO	NO	NO	NO
17	NO	NO	NO	SI	SI	SI
18	NO	SI	NO	NO	NO	NO
19	NO	NO	NO	SI	SI	SI
20	NO	NO	NO	NO	NO	NO
21	NO	NO	NO	SI	SI	SI
22	NO	NO	NO	SI	SI	SI
23	SI	SI	SI	SI	SI	SI
24	NO	NO	NO	NO	SI	SI
25	SI	SI	SI	SI	SI	SI

Fueron analizados de manera visual 25 epochs de cada uno de los registros, el registro 041222F9 tiene 6 epochs contaminados con artefactos EOG y 19 epochs libres de artefactos EOG, mientras que el registro 041222I6 tiene 18 epochs contaminados con artefactos EOG y 7 epochs libres de artefactos EOG. Con base a los resultados, Tabla 2, la clasificación se muestra en la Tabla 13.

**Tabla 13: Clasificación de resultados método de correlación en MatLab.**

	Registro 041222F9.bin				Registro 041222I6.bin			
	0,47	0,5	0,55	0,6	0,47	0,5	0,55	0,6
<b>Falso Positivo FP</b>	7	5	4	3	3	1	1	1
<b>Falso Negativo FN</b>	1	1	3	3	0	0	1	4
<b>Verdadero Positivo VP</b>	5	5	3	3	18	18	17	14
<b>Verdadero Negativo VN</b>	12	14	15	16	4	6	6	6

Se calcularon los valores para sensibilidad, especificidad, valores predictivos positivo y negativo. Los resultados se muestran en la Tabla 14.

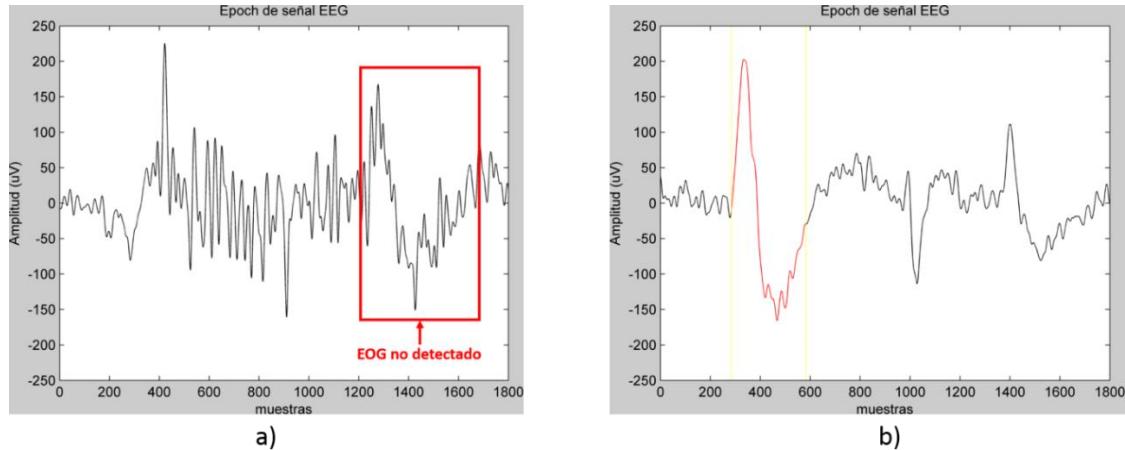
**Tabla 14: Resultados de la evaluación método de correlación**

	Registro 041222F9.bin				Registro 041222I6.bin			
	0,47	0,5	0,55	0,6	0,47	0,5	0,55	0,6
<b>Sensibilidad</b>	83,3%	83,3%	50%	50%	100%	100%	94,4%	73,6%
<b>Especificidad</b>	63,1%	73,6%	78%	84,2%	57,1%	85,71%	85,71%	85,71%
<b>Valor Predictivo Positivo</b>	41,6%	50%	42,8%	50%	85,7%	94,7%	94,4%	93,3%
<b>Valor Predictivo Negativo</b>	7,7%	6,6%	16,6%	15,7%	0%	0%	14,28%	40%
<b>Exactitud</b>	68%	76%	72%	76%	88%	96%	92%	80%

Es claro que el método de correlación da mejores resultados con un parámetro de 0,5. Con este índice, 5 de 6 epochs contaminados, en el caso del registro 222F9, tuvieron un resultado positivo (verdaderos positivos). 1 no se detectó y, por tanto, tuvo un resultado negativo (falsos negativos). En los epochs libres de artefactos, 14 de 19 se clasificaron como libres de artefactos oculares (verdaderos negativos) y 5 fueron clasificados como contaminados (falsos positivos). En el caso del registro 041222I6, 18 de 18 epochs contaminados tuvieron un resultado positivo. No ocurrieron falsos negativos. En los epochs libres de artefactos, 6 de 7 se clasificaron como libres de artefactos y 1 fue clasificado como contaminado. En este punto la sensibilidad oscila entre el 0,833 y 1, lo que significa que el método tiene una posibilidad del 83,3% al 100% de dar un resultado positivo cuando hay un artefacto ocular presente. La especificidad oscila entre 0,736 y 0,857, lo que significa que hay de un 73,6% a un 85,7% que el método de un resultado negativo cuando no hay artefacto ocular presente. Adicionalmente el valor predictivo positivo nos indica que hay de un 50 a un 94,7% de probabilidad de que un epoch marcado como contaminado por el método de correlación, esté efectivamente contaminado. El valor predictivo negativo muestra que hay una posibilidad de 0 a 6,6% de que un epoch marcado como libre de artefactos en realidad este contaminado.

En la Figura 46a se muestra un epoch de una señal EEG el cual está contaminado por un artefacto ocular, sin embargo, no es detectado por el método de correlación (falso negativo) incluso con un umbral de 0,45, que debe ser el menos selectivo. Esto se debe a que en el epoch se presentan simultáneamente artefactos musculares, los cuales distorsionan la señal al punto que el índice de correlación está por debajo del umbral seleccionado. En la Figura 46b se muestra un epoch contaminado por un artefacto ocular que es detectado por el método con todos los umbrales que se probaron (verdadero positivo).

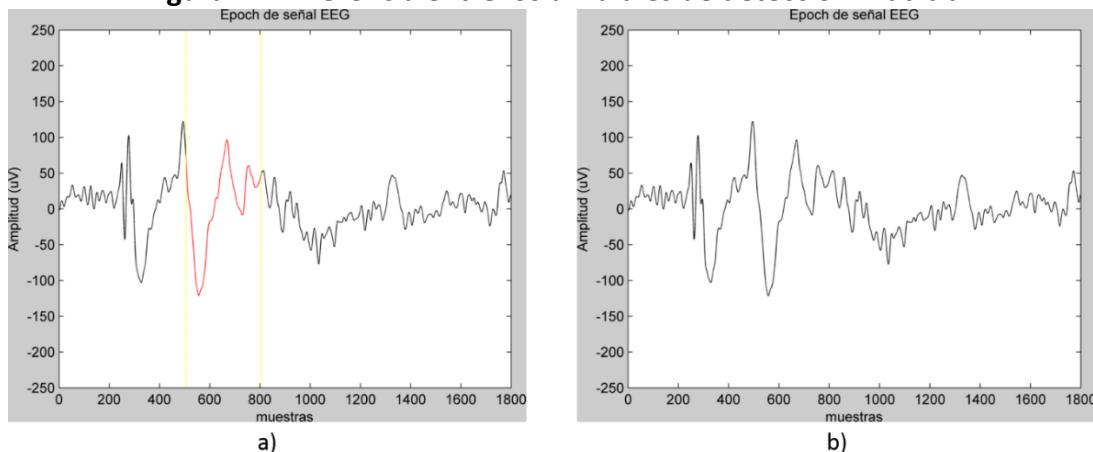
**Figura 46: a) Artefacto no detectado vs b) artefacto detectado por el método de correlación**



Fuente: De los Autores. Base de Datos registros 041222I6

En la Figura 47a se muestra un artefacto ocular detectado por el método de correlación con un umbral de 0,5. En la Figura 47b se muestra el mismo epoch contaminado, esta vez al ser sometido al método de correlación con umbral de 0,6, no se detecta el artefacto (falso negativo). Esta diferencia se debe a que al aumentar el umbral, que va de 0 a 1, el método se vuelve más selectivo, detectando solo artefactos que guardan gran semejanza con el patrón.

**Figura 47: Diferencia entre los umbrales de detección MatLab**



Fuente: De los Autores. Base de Datos registros 041222F9

Como se esperaba, por los datos mostrados en la Figura 38: Índices de correlación, del capítulo 2, el método de correlación expone mejores resultados con un umbral cercano a 0,5.

### 3.1.6 Implementación de Métodos en Python

#### *Descripción del Lenguaje de Programación Python*

Python es un lenguaje de programación de alto nivel orientado a objetos. Está diseñado para ser fácil de programar y fácil de leer. Guido van Rossum comenzó el diseño de Python en 1980. Con los años, Python ha ganado popularidad en una amplia gama de campos de desarrollo web, juegos, el uso como un lenguaje de scripting, y por supuesto de la ciencia y la ingeniería. Según la página del desarrollador ([pyhton.org](http://python.org)):

*"Python is powerful... and fast;  
plays well with others;  
runs everywhere;  
is friendly & easy to learn;  
is Open."*

**Figura 48: Logo de Python**



Fuente: <https://www.python.org/>

El lenguaje de programación Python, al no ser orientado principalmente al manejo de matrices, no muestra gran eficiencia al momento de procesar vectores o arreglos numéricos de grandes dimensiones. Por esta razón para reducir el tiempo de ejecución del programa se limitó la longitud del registro original que contiene la señal EEG, de tal forma que se obtuvieran 400 segundos o 200 epochs de señal. Esta longitud de la señal es suficiente para establecer el funcionamiento de los algoritmos y comparar resultados con la herramienta MatLab.

Existen similitudes entre los lenguajes MatLab y Python debido a que se han desarrollado librerías para ser utilizadas en Python, las cuales contienen funciones muy similares a las desarrolladas por MatLab. Algunos ejemplos de librerías son: NumPy, esta librería es el paquete fundamental para la computación científica con Python, es muy importante para la implementación de los métodos ya que es necesario el uso de funciones aritméticas que reduzcan la complejidad de implementación como promedios aritméticos, transformada de Fourier, operaciones matriciales y vectoriales; Matplotlib, librería indispensable si se requiere el uso de gráficas para presentar resultados, la sintaxis de las funciones contenidas en esta librería es muy similar a la de MatLab; Y por último SciPy, librería requerida para el diseño y aplicación de los diferentes tipos de filtros.

Las versiones descargadas e instaladas de las librerías antes mencionadas son:

- NumPy: numpy-1.9.2+mkl-cp34-none-win\_amd64.whl
- Matplotlib: matplotlib-1.4.3-cp34-none-win\_amd64.whl
- SciPy: scipy-0.15.1-cp34-none-win\_amd64.whl

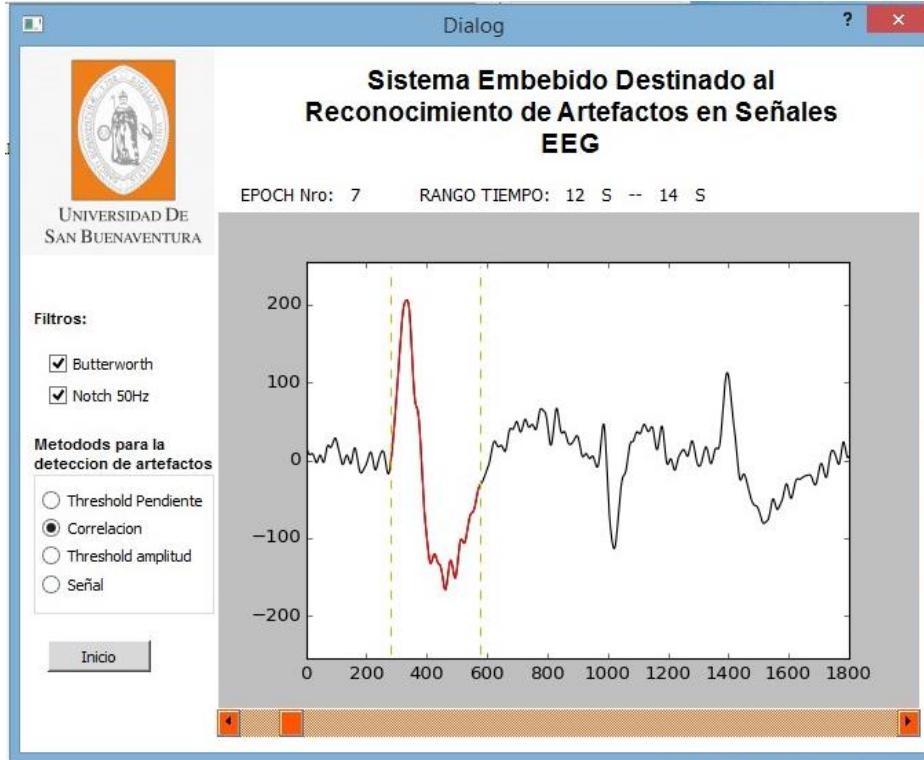
Dichas instalaciones fueron posibles a través de la herramienta PIP de Python.

Teniendo en cuenta las similitudes entre los lenguajes de desarrollo MatLab y Python, se abordaron todos los métodos de la misma manera que se hizo con MatLab, es decir para la implementación de los métodos en Python aplican tanto los algoritmos ya definidos como los diagramas de flujos expuestos anteriormente.

### **3.1.7 Diseño de la Interfaz Gráfica en Python**

Para implementar la interfaz gráfica en Python, se tienen dos opciones: la primera consiste en el desarrollo de la interfaz a través de codificación y la otra opción, siendo esta la empleada, consiste en uso de la herramienta QtDesigner que proporciona un entorno gráfico donde se puede definir de manera interactiva los elementos gráficos a utilizar, y con el programa PyQt se traduce el lenguaje gráfico o el diseño realizado al lenguaje Python. Se optó por diseñar la interfaz gráfica a través de estas herramientas debido a la gran facilidad y practicidad que presentan sin mencionar que su entorno de desarrollo es similar al entorno de Interface Gráfica de Usuario (GUI) de MatLab. Considerando las especificaciones de desarrollo descritas en el desarrollo de la interfaz gráfica en MatLab se implementó la siguiente interfaz gráfica.

**Figura 49: Interfaz Gráfica en Python**



Fuente: De los Autores.

Como se puede ver en la Figura 49, la interfaz implementada en Python difiere a la implementada en MatLab y eso se debe a que la implementación de los métodos en esta herramienta, tenía como fin establecer los parámetros con los cuales presenta mayor eficiencia la detección de artefactos por parte de los métodos seleccionados. Después de haber realizado el análisis y seleccionado los parámetros ideales para cada método, ya no es necesaria la visualización de los epochs en el dominio de la frecuencia. En esta interfaz solo se evaluaran los métodos con los parámetros que mejor resultado arrojaron utilizando MatLab.

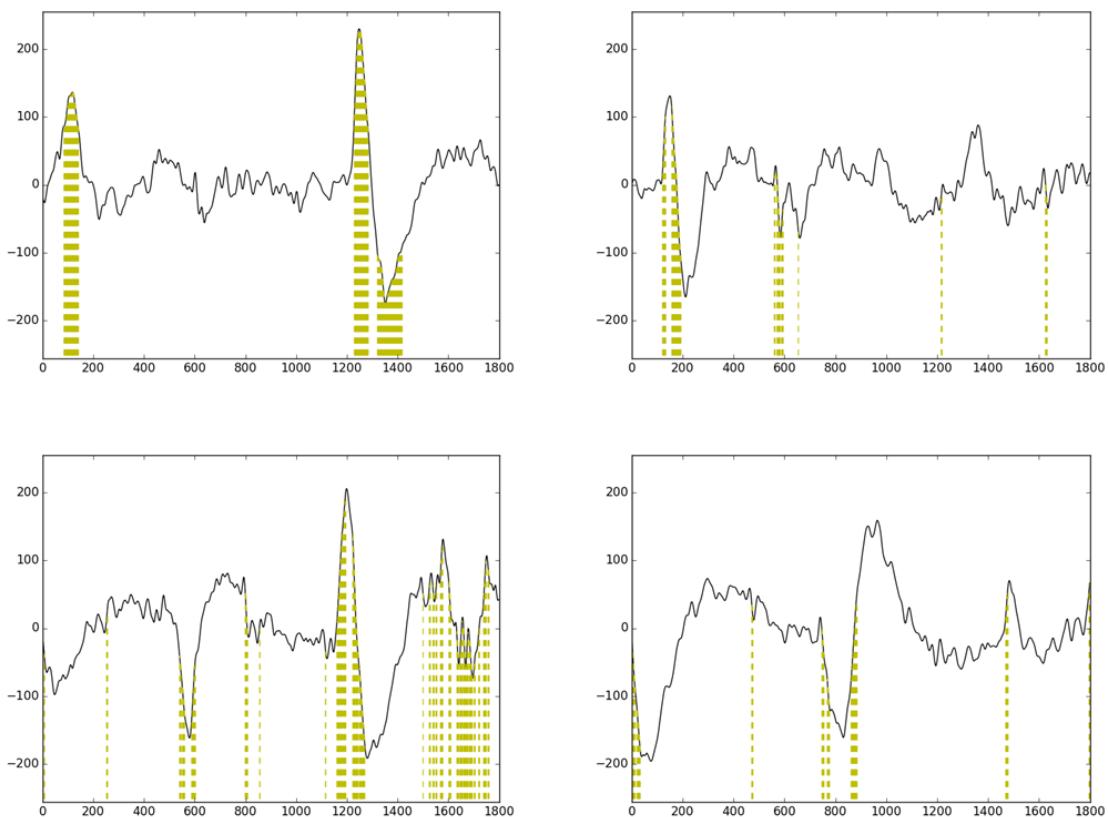
### 3.1.8 Resultados de la Implementación de Métodos en Python

#### *Resultados Simulados de Threshold de Amplitud en la Herramienta Python*

Para la debida verificación de la funcionalidad de los métodos en Python se empleó la base de datos de artefactos EOG diseñada para MatLab. Como dicha base de datos es un archivo de extensión “.mat” la cual hace alusión a MatLab fue necesario emplear la función SCIPY.IO.LOADMAT() incluida en la librería SciPy. Posteriormente se aplicó el método Threshold de Amplitud con el threshold establecido (3.5) el cual fue definido gracias a las previas simulaciones y análisis realizados a través de MatLab.

Luego de haber aplicado el método se constató la similitud entre las dos herramientas de simulación MATLAB y PYTHON debido a que los resultados fueron iguales en cuanto al factor detección, es decir que a través de PYTHON y con un umbral de amplitud o Threshold de 3.5 se lograron detectar 43 epochs contaminados con EOG de una totalidad de 50, lo que indica que la efectividad de 86% se mantiene a pesar de tratarse de una herramienta de simulación diferente. En la Tabla 15: Resultados método threshold de amplitud en Python se documentan los resultados donde se pueden visualizar tanto los epochs detectados como los que no. Algunos ejemplos de artefactos detectados se muestran en la Figura 50

**Figura 50: Detección de artefactos a través del método Threshold de Amplitud con un threshold de 3.5**



Fuente: De los Autores.

**Tabla 15: Resultados método threshold de amplitud en Python**

THRESHOLD DE AMPLITUD RESULTADOS PYTHON				
THRESHOLD 3.5	EPOCH Nro.	INTERVALO DE TIEMPO	ARTEFACTOS DETECTADOS	EFFECTIVIDAD
	1	0 sg - 2 sg	NA	86 %
	2	2 sg - 4 sg	NA	
	3	4 sg - 6 sg	NA	
	4	6 sg - 8 sg	NA	
	5	8 sg - 10 sg	NA	
	6	10 sg - 12 sg	SI	
	7	12 sg - 14 sg	SI	
	8	14 sg - 16 sg	SI	
	9	16 sg - 18 sg	SI	
	10	18 sg - 20 sg	SI	
	11	20 sg - 22 sg	SI	
	12	22 sg - 24 sg	SI	
	13	24 sg - 26 sg	SI	
	14	26 sg - 28 sg	SI	
	15	28 sg - 30 sg	SI	
	16	30 sg - 32 sg	SI	
	17	32 sg - 34 sg	SI	
	18	34 sg - 36 sg	SI	
	19	36 sg - 38 sg	NO	
	20	38 sg - 40 sg	NO	
	21	40 sg - 42 sg	SI	
	22	42 sg - 44 sg	SI	
	23	44 sg - 46 sg	SI	
	24	46 sg - 48 sg	SI	
	25	48 sg - 50 sg	SI	
	26	50 sg - 52 sg	SI	
	27	52 sg - 54 sg	SI	
	28	54 sg - 56 sg	SI	
	29	56 sg - 58 sg	SI	
	30	58 sg - 60 sg	SI	
	31	60 sg - 62 sg	SI	
	32	62 sg - 64 sg	SI	
	33	64 sg - 66 sg	SI	
	34	66 sg - 68 sg	SI	
	35	68 sg - 70 sg	SI	
	36	70 sg - 72 sg	SI	
	37	72 sg - 74 sg	SI	
	38	74 sg - 76 sg	NO	
	39	76 sg - 78 sg	SI	
	40	78 sg - 80 sg	SI	
	41	80 sg - 82 sg	NO	
	42	82 sg - 84 sg	SI	
	43	84 sg - 86 sg	SI	
	44	86 sg - 88 sg	SI	
	45	88 sg - 90 sg	NO	
	46	90 sg - 92 sg	NO	
	47	92 sg - 94 sg	SI	
	48	94 sg - 96 sg	NO	
	49	96 sg - 98 sg	SI	
	50	98 sg - 100 sg	SI	
	51	100 sg - 102 sg	SI	
	52	102 sg - 104 sg	SI	
	53	104 sg - 106 sg	SI	
	54	106 sg - 108 sg	SI	
	55	108 sg - 110 sg	SI	
	TOTAL DETECTADOS		43	

Después de aplicar el método a la base de datos se procedió a aplicar, de igual manera, el test de validez que anteriormente se llevó a cabo para los resultados obtenidos en MatLab. Se seleccionaron los mismos registros y debido a que ya está definido el valor de Threshold como 3.5 solo se indicaran a continuación los resultados con esta configuración de parámetros. Los resultados se muestran en la Tabla 16.

**Tabla 16: Resultados test de validez aplicado al método de threshold de amplitud en Python.**

Epoch	Registro 041222F9.bin		Registro 041222I6.bin	
	Inspeccion Visual	Python 3,5	Inspeccion Visual	Python 3,5
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
1	SI	SI	SI	SI
2	NO	NO	SI	SI
3	SI	SI	SI	NO
4	NO	SI	SI	NO
5	SI	SI	SI	SI
6	SI	SI	SI	SI
7	SI	SI	SI	SI
8	SI	SI	SI	SI
9	SI	SI	SI	NO
10	NO	NO	SI	SI
11	NO	NO	SI	SI
12	NO	NO	SI	SI
13	SI	SI	SI	SI
14	NO	SI	NO	NO
15	NO	SI	SI	SI
16	NO	SI	SI	SI
17	NO	SI	SI	SI
18	NO	SI	SI	SI
19	SI	SI	SI	SI
20	NO	SI	NO	NO
21	NO	SI	SI	SI
22	SI	SI	SI	SI
23	SI	SI	SI	SI
24	SI	SI	SI	SI
25	SI	SI	SI	SI
TOTAL DETECTADOS	13	21	23	20

En la Tabla 16 se pueden apreciar los epochs detectados como contaminados a su vez las detecciones erradas y el porcentaje de efectividad, de entrada para esta prueba se puede

verificar que la implementación de los métodos en Python y MatLab dan los mismos resultados. Por consiguiente los valores siguientes a calcular serán iguales a los obtenidos en el test aplicado para MatLab.

**Tabla 17: Clasificación de resultados método de threshold de amplitud en Python**

	Registro 041222F9.bin	Registro 041222I6.bin
<b>Valor de Threshold</b>	<b>3,5</b>	<b>3,5</b>
<b>Falso Positivo FP</b>	9	0
<b>Falso Negativo FN</b>	0	3
<b>Verdadero Positivo VP</b>	12	20
<b>Verdadero Negativo VN</b>	4	2

**Tabla 18: Resultados de la evaluación del método de threshold de amplitud en Python.**

	Registro 041222F9.bin	Registro 041222I6.bin
<b>Valor de Threshold</b>	<b>3,5</b>	<b>3,5</b>
<b>Sensibilidad</b>	100%	87%
<b>Especificidad</b>	31%	100%
<b>Valor Predictivo Positivo</b>	57%	100%
<b>Valor Predictivo Negativo</b>	0%	60%
<b>Exactitud</b>	64%	88%
<b>Exactitud Promedio</b>	76%	

Con las tablas 17 y 18 anteriores se confirma la similitud planteada anteriormente.

#### ***Resultados Simulados del Método Threshold de Pendiente en la Herramienta Python***

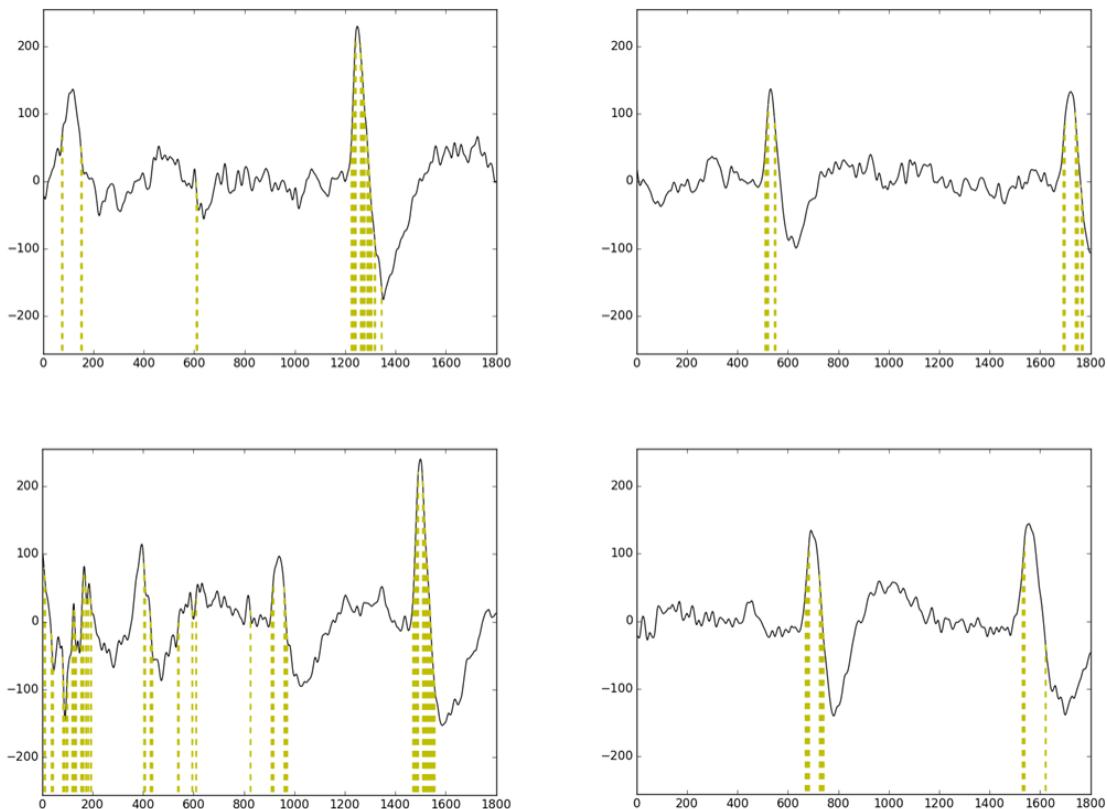
El valor de Threshold de pendiente definitivo para este método fue de 4.0, de este modo se aplicó el método a la Base de Datos obteniendo una efectividad del 96%, pues se lograron detectar 48 epochs contaminados de la base de datos, los resultados se pueden contemplar específicamente los epochs no detectados en la Tabla 19.

**Tabla 19: Resultados método threshold de pendiente en Python**

THRESHOLD DE PENDIENTE RESULTADOS PYTHON				
THRESHOLD 4.0	EPOCH Nro.	INTERVALO DE TIEMPO	ARTEFACTOS DETECTADOS	EFFECTIVIDA D
	1	0 sg - 2 sg	SI	96 %
	2	2 sg - 4 sg	SI	
	3	4 sg - 6 sg	SI	
	4	6 sg - 8 sg	SI	
	5	8 sg - 10 sg	SI	
	6	10 sg - 12 sg	SI	
	7	12 sg - 14 sg	SI	
	8	14 sg - 16 sg	SI	
	9	16 sg - 18 sg	SI	
	10	18 sg - 20 sg	SI	
	11	20 sg - 22 sg	SI	
	12	22 sg - 24 sg	SI	
	13	24 sg - 26 sg	SI	
	14	26 sg - 28 sg	SI	
	15	28 sg - 30 sg	NO	
	16	30 sg - 32 sg	SI	
	17	32 sg - 34 sg	SI	
	18	34 sg - 36 sg	SI	
	19	36 sg - 38 sg	SI	
	20	38 sg - 40 sg	SI	
	21	40 sg - 42 sg	SI	
	22	42 sg - 44 sg	SI	
	23	44 sg - 46 sg	SI	
	24	46 sg - 48 sg	SI	
	25	48 sg - 50 sg	SI	
	26	50 sg - 52 sg	SI	
	27	52 sg - 54 sg	SI	
	28	54 sg - 56 sg	SI	
	29	56 sg - 58 sg	SI	
	30	58 sg - 60 sg	SI	
	31	60 sg - 62 sg	SI	
	32	62 sg - 64 sg	SI	
	33	64 sg - 66 sg	SI	
	34	66 sg - 68 sg	SI	
	35	68 sg - 70 sg	SI	
	36	70 sg - 72 sg	SI	
	37	72 sg - 74 sg	SI	
	38	74 sg - 76 sg	SI	
	39	76 sg - 78 sg	SI	
	40	78 sg - 80 sg	SI	
	41	80 sg - 82 sg	NO	
	42	82 sg - 84 sg	SI	
	43	84 sg - 86 sg	SI	
	44	86 sg - 88 sg	SI	
	45	88 sg - 90 sg	SI	
	46	90 sg - 92 sg	SI	
	47	92 sg - 94 sg	SI	
	48	94 sg - 96 sg	SI	
	49	96 sg - 98 sg	SI	
	50	98 sg - 100 sg	SI	
	TOTAL DETECTADOS		48	

En la Tabla 19 anterior al igual que para MatLab se puede visualizar que los primeros 5 epochs son excluidos en el análisis de este método porque de que no se requieren 10 segundos previos de EEG limpio, en la Figura 51 se muestran algunos de los resultados obtenidos.

**Figura 51: Detección de artefactos a través del método Threshold de pendiente con un Threshold de 4.0**



*Fuente: De los Autores.*

Para efectos de comparación de resultados entre MatLab y Python también se aplicó el test de validez. Los resultados del test se indican en la Tabla 20.

**Tabla 20: Resultados test de validez aplicado al método de threshold de pendiente en Python.**

	Registro 041222F9.bin		Registro 041222I6.bin	
Epoch	Inspeccion Visual	Python 4,0	Inspeccion Visual	Python 4,0
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
1	SI	SI	SI	SI
2	NO	SI	SI	SI
3	SI	SI	SI	SI
4	NO	SI	SI	SI
5	SI	SI	SI	SI
6	SI	SI	SI	SI
7	SI	SI	SI	SI
8	SI	SI	SI	SI
9	SI	SI	SI	SI
10	NO	SI	SI	SI
11	NO	SI	SI	SI
12	NO	NO	SI	SI
13	SI	NO	SI	SI
14	NO	NO	NO	SI
15	NO	NO	SI	SI
16	NO	SI	SI	SI
17	NO	SI	SI	NO
18	NO	SI	SI	SI
19	SI	NO	SI	SI
20	NO	SI	NO	NO
21	NO	NO	SI	SI
22	SI	SI	SI	SI
23	SI	SI	SI	SI
24	SI	SI	SI	SI
25	SI	SI	SI	SI
TOTAL DETECTADOS	13	19	23	23

**Tabla 21: Clasificación de resultados método de threshold de pendiente en Python**

	Registro 041222F9.bin	Registro 041222I6.bin
<b>Valor de Threshod</b>	<b>4,0</b>	<b>4,0</b>
<b>Falso Positivo FP</b>	8	1
<b>Falso Negativo FN</b>	2	1
<b>Verdadero Positivo VP</b>	11	22
<b>Verdadero Negativo VN</b>	4	1

**Tabla 22: Resultados de la evaluación del método de threshold de pendiente en Python.**

	Registro 041222F9.bin	Registro 041222I6.bin
<b>Valor de Threshold</b>	<b>4,0</b>	<b>4,0</b>
<b>Sensibilidad</b>	85%	96%
<b>Especificidad</b>	33%	50%
<b>Valor Predictivo Positivo</b>	58%	96%
<b>Valor Predictivo Negativo</b>	33%	50%
<b>Exactitud</b>	60%	92%
<b>Exactitud Promedio</b>	76%	

***Resultados Simulados del Método de Correlación implementado en Python***

Al igual que en los resultados exhibidos en los métodos anteriores, los resultados de la aplicación del método de correlación en Python son iguales a los resultados mostrados en la simulación en MatLab. Para ser coherentes con el método de validación de resultados se realizó nuevamente la prueba con el método descrito en el capítulo anterior. Los resultados de esta prueba se muestran en la Tabla 23.

**Tabla 23: Resultados test de validez aplicado al método de correlación en Python.**

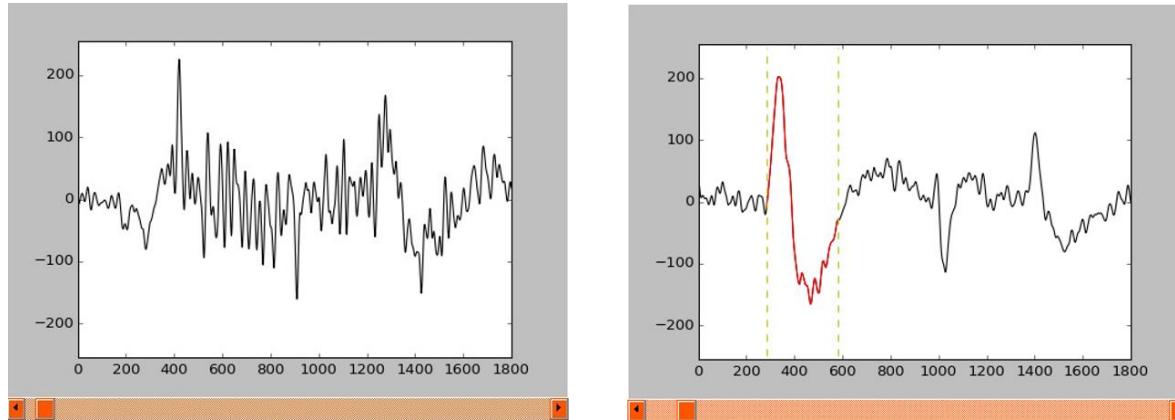
Epoch	Registro 041222F9.bin			Registro 041222I6.bin		
	Inspección Visual	Python 0,47	Python 0,5	Inspección Visual	Python 0,47	Python 0,5
1	NO	SI	SI	SI	SI	SI
2	NO	SI	NO	NO	SI	NO
3	SI	NO	NO	SI	SI	SI
4	NO	NO	NO	SI	SI	SI
5	NO	NO	NO	SI	SI	SI
6	SI	SI	SI	SI	SI	SI
7	SI	SI	SI	SI	SI	SI
8	SI	SI	SI	SI	SI	SI
9	NO	SI	SI	NO	SI	NO

<b>10</b>	NO	NO	NO	SI	SI	SI
<b>11</b>	NO	NO	NO	SI	SI	SI
<b>12</b>	NO	SI	SI	SI	SI	SI
<b>13</b>	NO	SI	SI	SI	SI	SI
<b>14</b>	NO	SI	SI	NO	NO	NO
<b>15</b>	NO	NO	NO	SI	SI	SI
<b>16</b>	NO	NO	NO	NO	NO	NO
<b>17</b>	NO	NO	NO	SI	SI	SI
<b>18</b>	NO	SI	NO	NO	NO	NO
<b>19</b>	NO	NO	NO	SI	SI	SI
<b>20</b>	NO	NO	NO	NO	NO	NO
<b>21</b>	NO	NO	NO	SI	SI	SI
<b>22</b>	NO	NO	NO	SI	SI	SI
<b>23</b>	SI	SI	SI	SI	SI	SI
<b>24</b>	NO	NO	NO	NO	SI	SI
<b>25</b>	SI	SI	SI	SI	SI	SI

Como se puede ver en la Tabla 23, los resultados no difieren de los obtenidos en MatLab, y por consiguiente, los valores de sensibilidad, especificidad y valores predictivos positivos y negativos son exactamente iguales. La razón por la cual no se evaluaron los índices de 0,55 y 0,6 es el pobre desempeño mostrado por el método cuando se realizó la simulación en MatLab utilizando estos índices.

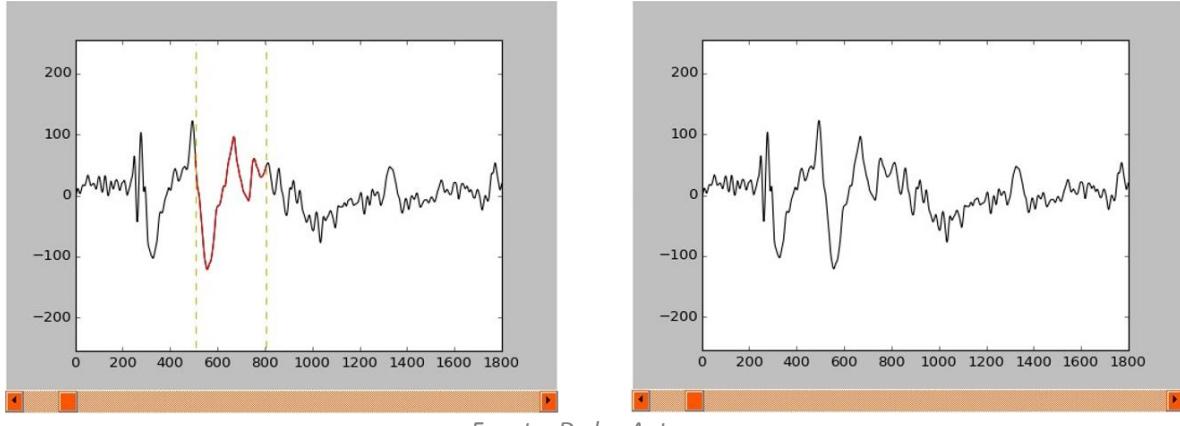
En la Figura 52a, se muestra un artefacto ocular no detectado por el método de correlación implementado en Python. En la Figura 52b, se muestra un artefacto ocular detectado por el método de correlación.

**Figura 52: a) Artefacto no detectado vs b) artefacto detectado por el método de correlación.**



En la Figura 53, se muestra un artefacto ocular detectado por el método de correlación con un coeficiente de 0,5. Mientras que en el segundo cuadro de la figura se muestra el mismo epoch, esta vez el algoritmo ha sido configurado con un coeficiente de 0,6, lo que hace que no detecte el artefacto ocular.

**Figura 53: Diferencia entre los umbrales de detección Phyton**



Fuente: De los Autores.

El objetivo de esta simulación en lenguaje Python, verificar y garantizar que la programación en el sistema embebido tendrá un desempeño similar o igual al de la simulación en MATLAB. Python es un lenguaje que permite migrar de sistemas operativos y de plataformas de una forma sencilla, sin hacer cambios mayores al código. Es por eso que se eligió este lenguaje para la implementación final de este trabajo, además del hecho de que es un lenguaje de fuente abierta.

## CAPÍTULO 4

### 4.1 Selección del Sistema Embebido

Los sistemas embebidos están presentes en prácticamente todas las actividades humanas y, debido a los costes tecnológicos actuales bajas, hay una tendencia a aumentar su presencia en la vida cotidiana de los usuarios.

Una de las partes más importantes en la selección de un sistema embebido es la elaboración de los requisitos para la aplicación que se busca implementar. Cuanto mejor se realice esta fase previa a la implementación de un prototipo, menos probables serán los cambios, tanto en hardware como en software. Aun así hay que recordar que el desarrollo del sistema generalmente es un proceso iterativo, es decir que a medida que avance el proyecto, las necesidades para el funcionamiento del prototipo pueden variar. Para la elección del sistema embebido se establecieron las siguientes consideraciones:

#### Definición de las interfaces de control.

- El sistema debe tener memoria no volátil, con posibilidad de conectar almacenamiento externo para ingresar la señal a analizar.
- Salida de video compuesto o HDMI para visualización de la interfaz gráfica.
- (Opcional) Puertos de entrada y salida de propósito general GPIO para que en trabajos futuros pueda conectarse directamente a un sensor.

#### Definición de la aplicación software.

- Posibilidad de programación con lenguajes de alto y bajo nivel
- (Opcional) Posibilidad de contar con un sistema operativo completo en el sistema embebido.
- Compatibilidad con drivers para el funcionamiento del hardware externo.
- Software libre.

#### Requisitos de alimentación.

- Ningún requerimiento especial

#### Diseño de la forma y tamaño.

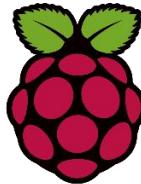
- Tamaño reducido
- Al no ser expuesto a condiciones adversas, no es necesario robustez en la construcción

#### Necesidades de rendimiento.

- Procesamiento rápido de operaciones matemáticas complejas.
- Capacidad de manejar interfaces gráficas.
- (Opcional) Capacidad de comunicarse con otros interfaces de alta velocidad.

Para facilitar la selección se creó una tabla comparativa de los principales sistemas embebidos disponibles en el mercado local.

**Tabla 24: Comparación entre sistemas embebidos**

			
	Arduino	Raspberry Pi	BeagleBone
<b>Modelo</b>	UNO	Modelo B	Rev. A5
<b>Precio</b>	30 USD	35 USD	90 USD
<b>Tamaño</b>	7,5 x 5,3 cm	8,5 x 5,4 cm	8,6 x 5,3 cm
<b>Procesador</b>	ATMega 328	ARM1176JZ-F	AM3559
<b>Velocidad CPU</b>	16 MHz	700 MHz	720 MHz
<b>Memoria RAM</b>	2 KB	256 KB	256 KB
<b>Memoria Flash</b>	32 KB	SD Card	microSD Card
<b>Voltaje Entrada</b>	7 - 12 v	5v	5v
<b>Consumo min</b>	0,3 W	3,5 W	0,85 W
<b>GPIO</b>	14	8	66
<b>Entrada Análoga</b>	6 10-bit	N/A	7 12-bit
<b>UART</b>	1	1	5
<b>Ethernet</b>	N/A	10/100	10/100
<b>USB</b>	N/A	2 USB 2.0	1 USB 2.0
<b>Video</b>	N/A	HDMI, Compuesto	N/A
<b>Audio</b>	N/A	HDMI, Análogo	Análogo
<b>Sistema Operativo</b>	N/A	Soporta sistemas operativos principalmente basados en Linux como: RASPBIAN, PIDORA, OSMC, Entre otros	Soporta sistemas operativos principalmente basados en Linux como: AMSTRONG, UBUNTU, ANDROID, Entre otros.

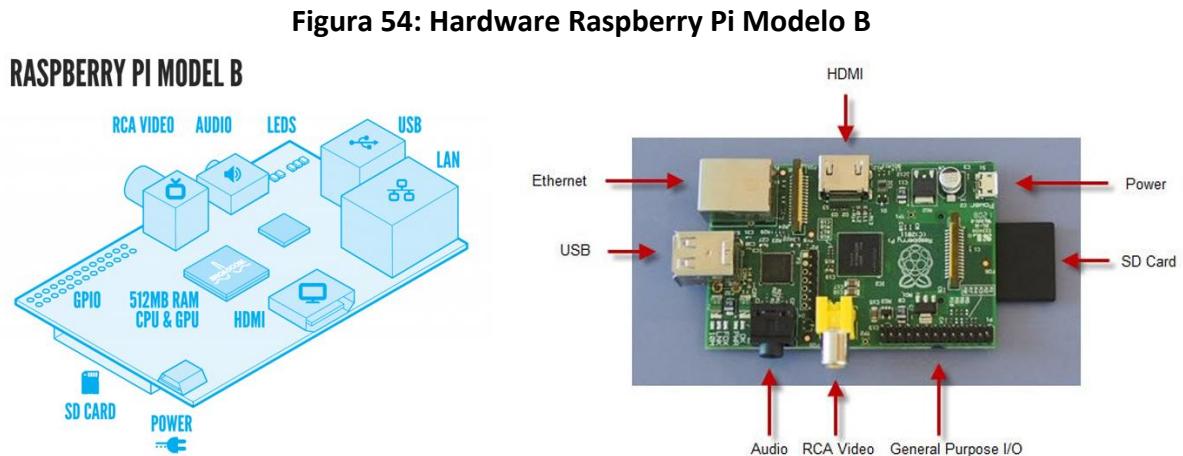
Después de valorar las posibilidades se seleccionó la Raspberry Pi, pues cumple con todos los requerimientos técnicos antes planteados, permite instalar varios sistemas operativos, tiene la capacidad de manejar interfaces gráficas directamente y se puede programar en lenguajes como Phyton y C, además de poder usar las librerías propias de estos lenguajes.

#### 4.2 Descripción Raspberry Pi Modelo B

La Raspberry Pi es un ordenador de placa reducida (SBC) de bajo coste desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. Según la página web del fabricante (<https://www.raspberrypi.org/>):

*“La Raspberry Pi es ordenador de bajo costo, del tamaño de una tarjeta de crédito que se conecta a un monitor de ordenador o un televisor, y utiliza un teclado y un ratón estándar. Es un pequeño dispositivo que permite a las personas de todas las edades explorar la computación, y aprender a programar en lenguajes como Scratch y Python. Es capaz de hacer todo lo que espera de una computadora de escritorio, desde navegar por Internet y reproducción de video de alta definición, a hacer hojas de cálculo, procesadores de texto, y jugar.”*

En la Figura 54 se pueden observar las principales características en cuanto a hardware de la raspberry Pi modelo B.



Fuente: <https://www.raspberrypi.org/>

#### 4.3 Acondicionamiento de la Raspberry

Para concluir en la implementación de los métodos en la raspberry es necesario primeramente realizar el acondicionamiento adecuado del dispositivo, esto implica llevar a cabo la instalación del sistema operativo Raspbian para la cual seguidamente se indicará el proceso.

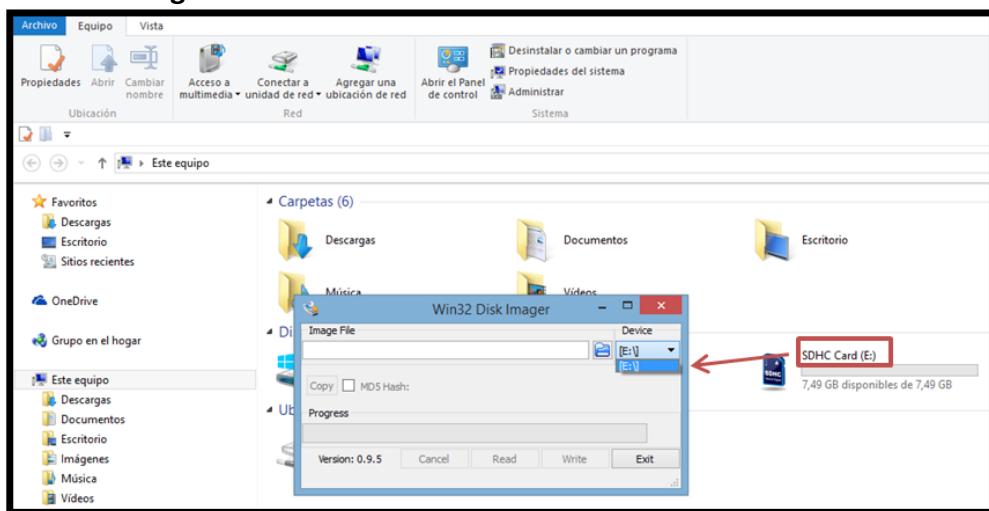
Es necesario disponer primeramente de una memoria SD con la capacidad necesaria para la debida instalación del sistema operativo. Se optó porque dicha capacidad fuera del orden superior o igual a 8 GB y a su vez dicha memoria debe contar con su respectivo portador SD con el fin de que pueda ser insertada tanto en el computador como en la raspberry. Posteriormente se debe contar con el archivo imagen referente al sistema operativo de la raspberry en donde se resalta que se seleccionó RASPBIAN basado en DEBIAN WHEEZY el cual fue obtenido directamente de la página del fabricante a través de la ruta <https://www.raspberrypi.org/downloads/raspbian/>.

Para llevar a cabo la instalación del sistema operativo es necesario el programa Win32DiskImager el cual fue obtenido a través de la siguiente ruta <http://sourceforge.net/projects/win32diskimager/files/latest/download>

Luego de haber insertado la memoria SD en el equipo se procedió a instalar la imagen del sistema operativo a través del programa anteriormente mencionado. A continuación se indica el procedimiento realizado y se ilustran algunas imágenes del mismo:

Se ejecutó el programa en modo administrador y seleccionó la ruta de ubicación de la memoria SD insertada

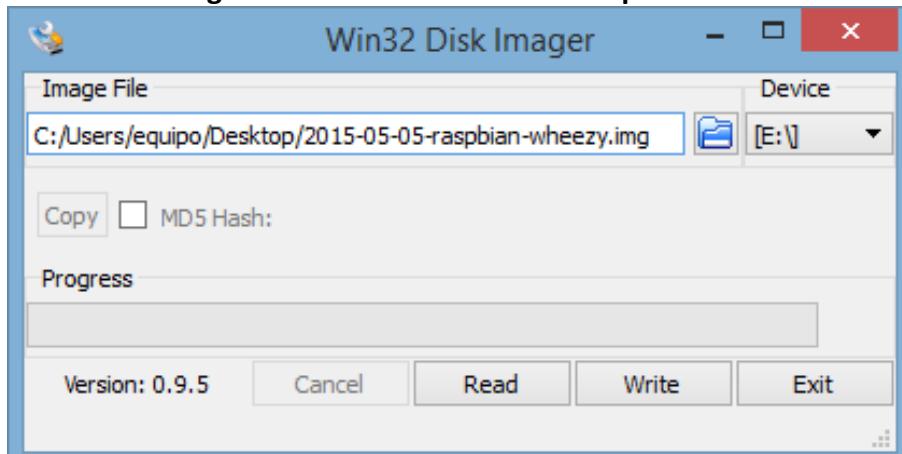
**Figura 55: Selección de la unidad de almacenamiento**



Fuente: De los Autores.

Posteriormente se seleccionó a través del ícono de carpeta azul el archivo imagen referente al sistema operativo RASPBIAN WHEEZY antes descargado y se seleccionó la opción Write.

**Figura 56: Selección de sistema operativo**



Fuente: De los Autores.

De la anterior forma ilustrada fue como se logró satisfactoriamente la instalación del sistema operativo en la memoria SD. Seguidamente se insertó la memoria en el dispositivo embebido Raspberry y se realizó la configuración inicial del mismo, de esta manera queda el dispositivo listo para su funcionamiento puesto que el equipo por defecto tiene instalada la herramienta de programación y simulación Python versión 2.7 y versión 3.2, en nuestro caso se hará uso de la versión 3.2

#### **4.4 Implementación de Métodos en el Sistema Embebido: Raspberry modelo B**

Para este bloque vale la pena recordar la razón por la cual se escogió el lenguaje Python para la implementación y desarrollo de los métodos antes mencionados; Python es un lenguaje de programación totalmente compatible con la tarjeta raspberry, es la herramienta matemática más eficiente en cuanto a procesamientos cuantitativos que presenta la tarjeta y permite a su vez el desarrollo dinámico de interfaces gráficas de usuario, es decir una herramienta muy similar MATLAB como ya antes se había hecho mención. Python es un lenguaje global en el que la programación no depende del tipo de dispositivo en que este se esté empleando, es decir, la codificación de los métodos realizada para la versión 3,4 de Python para 64 bits instalada en el sistema operativo 7 u 8 no deberá ser remplazada o modificada si se desea implementar en otro sistema operativo, es ente caso haciendo referencia al sistema operativo propio de la raspberry, lo anteriormente manifestado se cumple si y solo si la versión de Python instalada en la

raspberry es igual a la versión instalada en el sistema operativo Windows y se condiciona a su vez la nueva instalación de las librerías destinadas propiamente al sistema operativo de la raspberry pues los sistemas operativos difieren, por ende las librerías; el proceso de instalación de las librerías adecuadas para un óptimo desempeño se relaciona a continuación:

Partiendo de que Python 3 se encuentra previamente instalado en la Raspberry, se procede a instalar el asistente correspondiente a la versión de Python 3, que permitirá realizar la correcta descarga e instalación de las librerías que se requieran. El asistente es la herramienta PIP el cual se logra instalar a través de la ejecución del siguiente código en el LXTerminal de la Raspberry:

- Sudo apt-get update
- Sudo apt-get install python3-dev
- sudo apt-get install python3-pip
- sudo pip-3.2 install --upgrade distribute

Posterior a haber realizado la instalación del asistente PIP se realiza la instalación de las librerías antes mencionadas a través de la ejecución de las siguientes líneas en el terminal de la Raspberry:

- NumPy:
  - sudo pip-3.2 install --upgrade numpy
- SciPy:
  - sudo apt-get install python3-scipy
- PyQt4:
  - sudo apt-get install python3-pyqt4
- Matplotlib:
  - sudo pip-3.2 install --upgrade matplotlib

Luego de realizar la correcta instalación de las librerías se procedió a trasladar el código antes desarrollado para la herramienta Phyton, a la Raspberry. En virtud de lo expuesto, se reitera que no fue necesario realizar adecuaciones o llevar a cabo la implementación de los métodos ya que la codificación no varía entre los dos sistemas operativos.

Cabe mencionar que es necesario para la ejecución del código trasportar a su vez a la Raspberry todos los archivos que conciernen a código implementado. Lo anterior hace referencia a Base de Datos de Registros, Base de Datos de Artefactos, señal o patrón de referencia EOG definido en Matlab, archivo JPG que contiene el logo Universitario, y el archivo Python obtenido de la herramienta PyQT4.

## **4.5 Resultados**

### **4.5.1 Resultados Simulados de Threshold de Amplitud en la Herramienta Python® Raspberry**

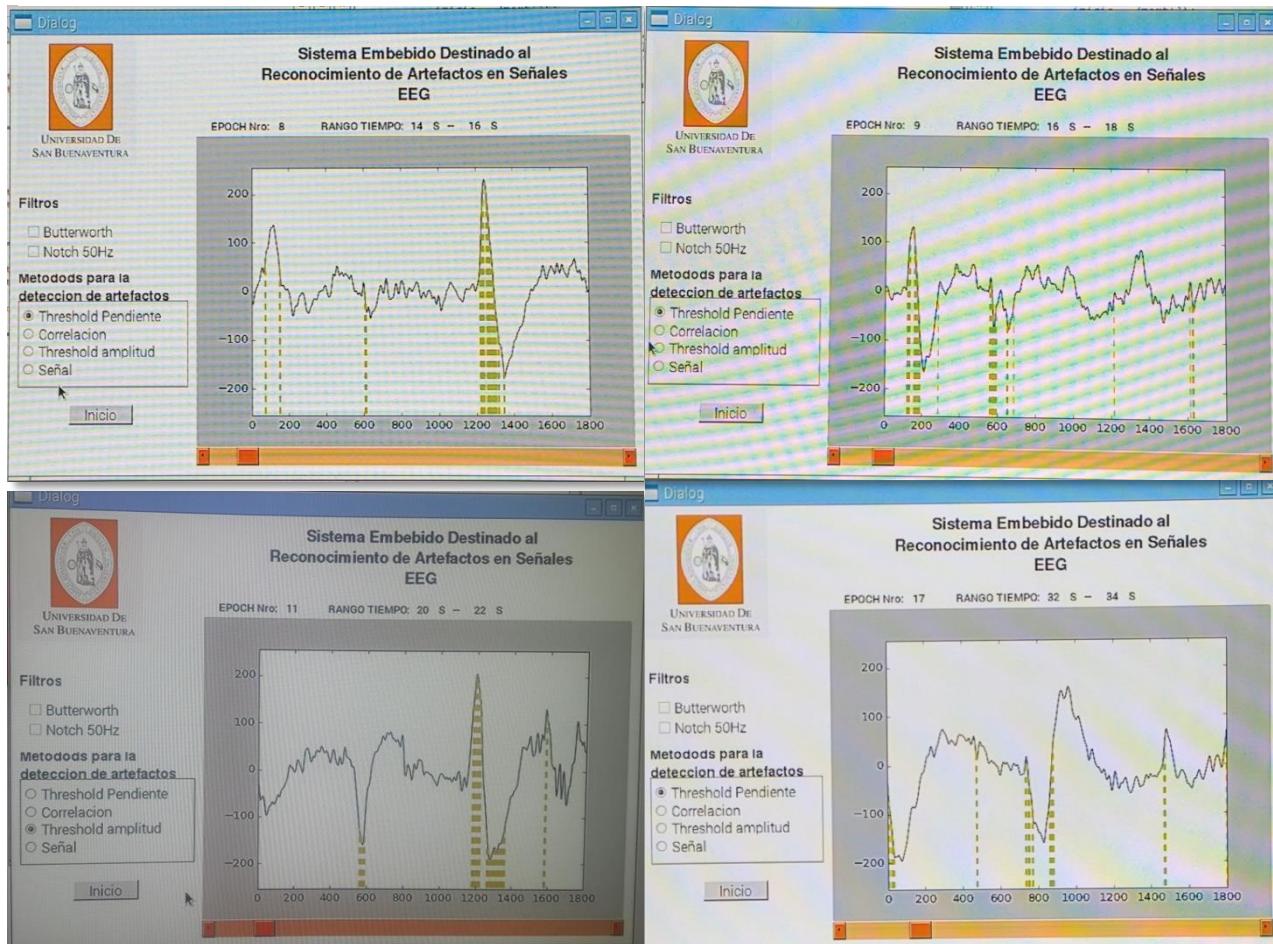
Primeramente se sometió el método implementado en la Raspberry a la prueba de detección a través de la base de datos de artefactos EOG

**Tabla 25: Resultados método de Threshold de Amplitud Raspberry**

THRESHOLD DE AMPLITUD RESULTADOS RASPBERRY				
THRESHOLD 3.5	EPOCH Nro.	INTERVALO DE TIEMPO	ARTEFACTOS DETECTADOS	EFFECTIVIDA D
	1	0 sg - 2 sg	NA	86 %
	2	2 sg - 4 sg	NA	
	3	4 sg - 6 sg	NA	
	4	6 sg - 8 sg	NA	
	5	8 sg - 10 sg	NA	
	6	10 sg - 12 sg	SI	
	7	12 sg - 14 sg	SI	
	8	14 sg - 16 sg	SI	
	9	16 sg - 18 sg	SI	
	10	18 sg - 20 sg	SI	
	11	20 sg - 22 sg	SI	
	12	22 sg - 24 sg	SI	
	13	24 sg - 26 sg	SI	
	14	26 sg - 28 sg	SI	
	15	28 sg - 30 sg	SI	
	16	30 sg - 32 sg	SI	
	17	32 sg - 34 sg	SI	
	18	34 sg - 36 sg	SI	
	19	36 sg - 38 sg	NO	
	20	38 sg - 40 sg	NO	
	21	40 sg - 42 sg	SI	
	22	42 sg - 44 sg	SI	
	23	44 sg - 46 sg	SI	
	24	46 sg - 48 sg	SI	
	25	48 sg - 50 sg	SI	
	26	50 sg - 52 sg	SI	
	27	52 sg - 54 sg	SI	
	28	54 sg - 56 sg	SI	
	29	56 sg - 58 sg	SI	
	30	58 sg - 60 sg	SI	
	31	60 sg - 62 sg	SI	
	32	62 sg - 64 sg	SI	
	33	64 sg - 66 sg	SI	
	34	66 sg - 68 sg	SI	
	35	68 sg - 70 sg	SI	
	36	70 sg - 72 sg	SI	
	37	72 sg - 74 sg	SI	
	38	74 sg - 76 sg	NO	
	39	76 sg - 78 sg	SI	
	40	78 sg - 80 sg	SI	
	41	80 sg - 82 sg	NO	
	42	82 sg - 84 sg	SI	
	43	84 sg - 86 sg	SI	
	44	86 sg - 88 sg	SI	
	45	88 sg - 90 sg	NO	
	46	90 sg - 92 sg	NO	
	47	92 sg - 94 sg	SI	
	48	94 sg - 96 sg	NO	
	49	96 sg - 98 sg	SI	
	50	98 sg - 100 sg	SI	
	51	100 sg - 102 sg	SI	
	52	102 sg - 104 sg	SI	
	53	104 sg - 106 sg	SI	
	54	106 sg - 108 sg	SI	
	55	108 sg - 110 sg	SI	
	TOTAL DETECTADOS		43	

Al evaluar el método de threshold de amplitud se pudo constatar la determinación de 43 epoch como contaminados esto indica que este método responde con una efectividad ante esta prueba de 86%, en la Figura 57 se ilustran imágenes de los resultados

**Figura 57: Resultados método de threshold de pendiente en Raspberry**



Fuente: De los Autores.

De igual manera y con el fin de evaluar los mismos parámetros anteriores, se aplicó el test de validez a los registros anteriormente evaluados y los resultados obtenidos se pueden constatar continuación

**Tabla 26: Resultados test de Validez aplicado al método de Threshold de Amplitud**

	Registro 041222F9.bin		Registro 041222I6.bin	
Epoch	Inspeccion Visual	Python 3,5	Inspeccion Visual	Python 3,5
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
1	SI	SI	SI	SI
2	NO	NO	SI	SI
3	SI	SI	SI	NO
4	NO	SI	SI	NO
5	SI	SI	SI	SI
6	SI	SI	SI	SI
7	SI	SI	SI	SI
8	SI	SI	SI	SI
9	SI	SI	SI	NO
10	NO	NO	SI	SI
11	NO	NO	SI	SI
12	NO	NO	SI	SI
13	SI	SI	SI	SI
14	NO	SI	NO	NO
15	NO	SI	SI	SI
16	NO	SI	SI	SI
17	NO	SI	SI	SI
18	NO	SI	SI	SI
19	SI	SI	SI	SI
20	NO	SI	NO	NO
21	NO	SI	SI	SI
22	SI	SI	SI	SI
23	SI	SI	SI	SI
24	SI	SI	SI	SI
25	SI	SI	SI	SI
TOTAL DETECTADOS	13	21	23	20

**Tabla 27: Clasificacion de resultados método de Threshold de Amplitud Raspberry**

	Registro 041222F9.bin	Registro 041222I6.bin
<b>Valor de Threshod</b>	<b>3,5</b>	<b>3,5</b>
<b>Falso Positivo FP</b>	9	0
<b>Falso Negativo FN</b>	0	3
<b>Verdadero Positivo VP</b>	12	20
<b>Verdadero Negativo VN</b>	4	2

**Tabla 28: Resultados de la evaluación del método de threshold de Amplitud en Raspberry.**

	Registro 041222F9.bin	Registro 041222I6.bin
<b>Valor de Threshold</b>	<b>3,5</b>	<b>3,5</b>
<b>Sensibilidad</b>	100%	87%
<b>Especificidad</b>	31%	100%
<b>Valor Predictivo Positivo</b>	57%	100%
<b>Valor Predictivo Negativo</b>	0%	60%
<b>Exactitud</b>	64%	88%
<b>Exactitud Promedio</b>	76%	

Se puede apreciar de momento que los resultados del test aplicado para Python son iguales para ambos casos es decir en el computador portátil con sistema operativo Windows 8 de 64 y raspberry son sistema operativo raspbian wheezy

#### 4.5.2 Resultados Simulados de Threshold de Pendiente en la Herramienta Python® Raspberry

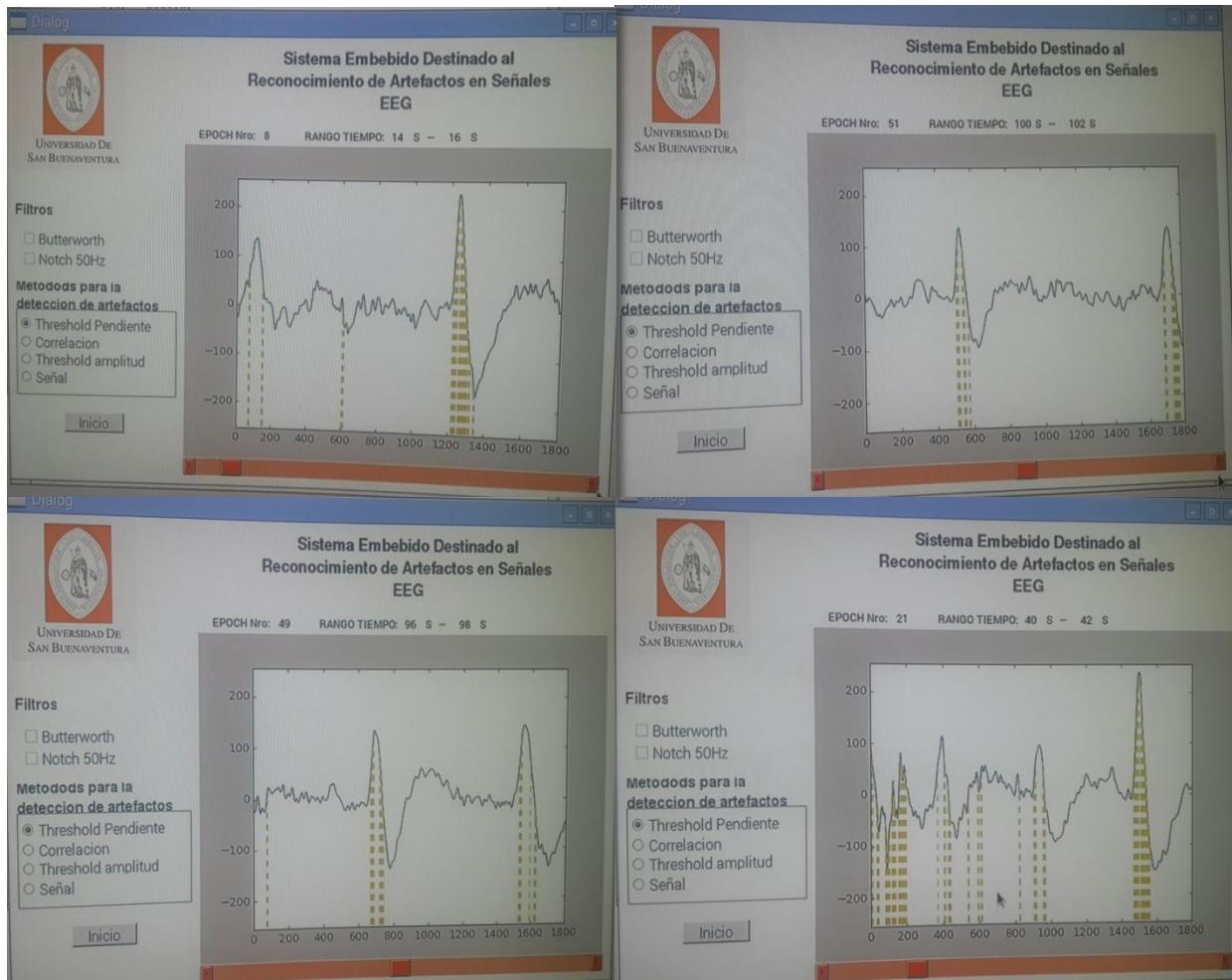
De igual forma que en los anteriores procedimientos se evaluó el método primeramente a través de la base de datos que contiene artefactos de EOG los resultados obtenidos se ilustran a continuación

**Tabla 29: Resultados método Threshold de Pendiente en Rapsberry**

THRESHOLD DE PENDIENTE RESULTADOS RASPBERRY				
THRESHOLD 4.0	EPOCH Nro	INTERVALO DE TIEMPO	ARTEFACTOS DETECTADOS	EFFECTIVIDAD
	1	0 sg - 2 sg	SI	96 %
	2	2 sg - 4 sg	SI	
	3	4 sg - 6 sg	SI	
	4	6 sg - 8 sg	SI	
	5	8 sg - 10 sg	SI	
	6	10 sg - 12 sg	SI	
	7	12 sg - 14 sg	SI	
	8	14 sg - 16 sg	SI	
	9	16 sg - 18 sg	SI	
	10	18 sg - 20 sg	SI	
	11	20 sg - 22 sg	SI	
	12	22 sg - 24 sg	SI	
	13	24 sg - 26 sg	SI	
	14	26 sg - 28 sg	SI	
	15	28 sg - 30 sg	NO	
	16	30 sg - 32 sg	SI	
	17	32 sg - 34 sg	SI	
	18	34 sg - 36 sg	SI	
	19	36 sg - 38 sg	SI	
	20	38 sg - 40 sg	SI	
	21	40 sg - 42 sg	SI	
	22	42 sg - 44 sg	SI	
	23	44 sg - 46 sg	SI	
	24	46 sg - 48 sg	SI	
	25	48 sg - 50 sg	SI	
	26	50 sg - 52 sg	SI	
	27	52 sg - 54 sg	SI	
	28	54 sg - 56 sg	SI	
	29	56 sg - 58 sg	SI	
	30	58 sg - 60 sg	SI	
	31	60 sg - 62 sg	SI	
	32	62 sg - 64 sg	SI	
	33	64 sg - 66 sg	SI	
	34	66 sg - 68 sg	SI	
	35	68 sg - 70 sg	SI	
	36	70 sg - 72 sg	SI	
	37	72 sg - 74 sg	SI	
	38	74 sg - 76 sg	SI	
	39	76 sg - 78 sg	SI	
	40	78 sg - 80 sg	SI	
	41	80 sg - 82 sg	NO	
	42	82 sg - 84 sg	SI	
	43	84 sg - 86 sg	SI	
	44	86 sg - 88 sg	SI	
	45	88 sg - 90 sg	SI	
	46	90 sg - 92 sg	SI	
	47	92 sg - 94 sg	SI	
	48	94 sg - 96 sg	SI	
	49	96 sg - 98 sg	SI	
	50	98 sg - 100 sg	SI	
	TOTAL DETECTADOS		48	

En la tabla anterior se puede visualizar el resultado de efectividad que presenta el método ante la base de datos el cual es de 96% el cual es a su vez igual al obtenido en los resultados de Python en el equipo portátil. En la Figura 58 se muestran imágenes de los resultados obtenidos.

**Figura 58: Resultados método threshold de pendiente en raspberry**



Fuente: De los Autores.

Se procedió a evaluar el método a través del test de validez obteniendo los siguientes resultados

**Tabla 30: Resultados test de Validez aplicado al método Threshold de Pendiente en Raspberry**

	Registro 041222F9.bin		Registro 041222I6.bin	
Epoch	Inspeccion Visual	Python 4,0	Inspeccion Visual	Python 4,0
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
NA	NA	NA	NA	NA
1	SI	SI	SI	SI
2	NO	SI	SI	SI
3	SI	SI	SI	SI
4	NO	SI	SI	SI
5	SI	SI	SI	SI
6	SI	SI	SI	SI
7	SI	SI	SI	SI
8	SI	SI	SI	SI
9	SI	SI	SI	SI
10	NO	SI	SI	SI
11	NO	SI	SI	SI
12	NO	NO	SI	SI
13	SI	NO	SI	SI
14	NO	NO	NO	SI
15	NO	NO	SI	SI
16	NO	SI	SI	SI
17	NO	SI	SI	NO
18	NO	SI	SI	SI
19	SI	NO	SI	SI
20	NO	SI	NO	NO
21	NO	NO	SI	SI
22	SI	SI	SI	SI
23	SI	SI	SI	SI
24	SI	SI	SI	SI
25	SI	SI	SI	SI
TOTAL DETECTADOS	13	19	23	23

**Tabla 31: Clasificación de resultados de Threshold de Pendiente en Raspberry**

	Registro 041222F9.bin	Registro 041222I6.bin
<b>Valor de Threshod</b>	<b>4,0</b>	<b>4,0</b>
<b>Falso Positivo FP</b>	8	1
<b>Falso Negativo FN</b>	2	1
<b>Verdadero Positivo VP</b>	11	22
<b>Verdadero Negativo VN</b>	4	1

**Tabla 32: Resultados de la evaluación del método de threshold de pendiente en Raspberry.**

	Registro 041222F9.bin	Registro 041222I6.bin
<b>Valor de Threshold</b>	<b>4,0</b>	<b>4,0</b>
<b>Sensibilidad</b>	85%	96%
<b>Especificidad</b>	33%	50%
<b>Valor Predictivo Positivo</b>	58%	96%
<b>Valor Predictivo Negativo</b>	33%	50%
<b>Exactitud</b>	60%	92%
<b>Exactitud Promedio</b>	76%	

Al igual que los resultados obtenidos en Python en el computador portátil se puede observar que los resultados entre los registros seleccionados difieren por grandes cantidades pero a su vez se puede visualizar que el índice más relevante que es la sensibilidad se mantiene relativamente alto.

#### 4.5.3 Resultados Método de Correlación Implementado en Raspberry

El algoritmo del método de correlación, programado en Python, fue implementado en la Raspberry, sin modificación en su código fuente. Los resultados fueron los mismos expuestos por las simulaciones, tanto en Matlab como en Phyton. Para ser coherentes con el método de validación de resultados se realizó nuevamente la prueba con el método descrito en el capítulo anterior. Los resultados de esta prueba se muestran en la tabla.

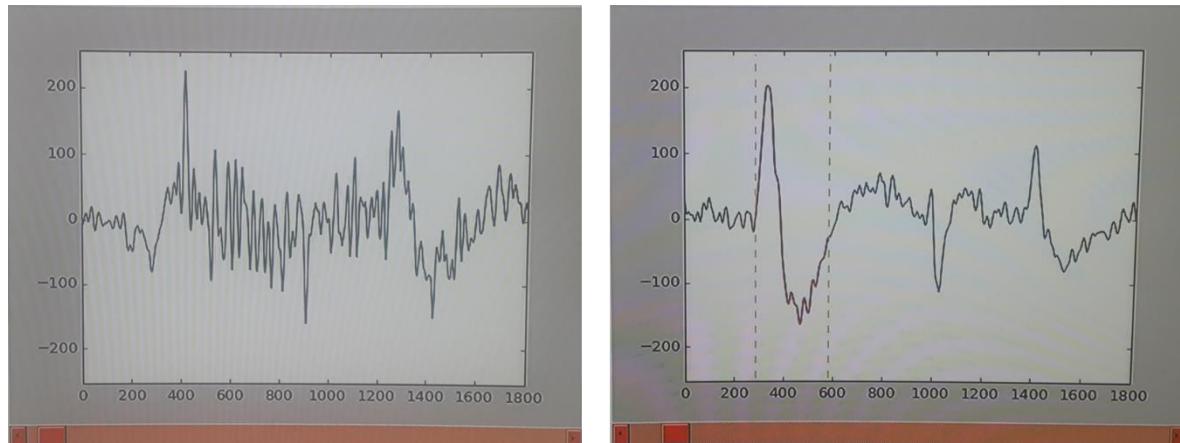
**Tabla 33: Resultados test de validez aplicado al método correlación en Raspberry**

Epoch	Registro 041222F9.bin		Registro 041222I6.bin	
	Inspección Visual	Python 0,5	Inspección Visual	Python 0,5
1	NO	SI	SI	SI
2	NO	NO	NO	NO
3	SI	NO	SI	SI
4	NO	NO	SI	SI

<b>5</b>	NO	NO	SI	SI
<b>6</b>	SI	SI	SI	SI
<b>7</b>	SI	SI	SI	SI
<b>8</b>	SI	SI	SI	SI
<b>9</b>	NO	SI	NO	NO
<b>10</b>	NO	NO	SI	SI
<b>11</b>	NO	NO	SI	SI
<b>12</b>	NO	SI	SI	SI
<b>13</b>	NO	SI	SI	SI
<b>14</b>	NO	SI	NO	NO
<b>15</b>	NO	NO	SI	SI
<b>16</b>	NO	NO	NO	NO
<b>17</b>	NO	NO	SI	SI
<b>18</b>	NO	NO	NO	NO
<b>19</b>	NO	NO	SI	SI
<b>20</b>	NO	NO	NO	NO
<b>21</b>	NO	NO	SI	SI
<b>22</b>	NO	NO	SI	SI
<b>23</b>	SI	SI	SI	SI
<b>24</b>	NO	NO	NO	SI
<b>25</b>	SI	SI	SI	SI

En la Figura 59a, se muestra un epoch con un EOG, contaminado por actividad muscular e interferencia de 50 Hz, el cual no es detectado por el algoritmo. En Figura 59b se muestra un epoch contaminado con un EOG, el cual si es detectado por el método de correlación.

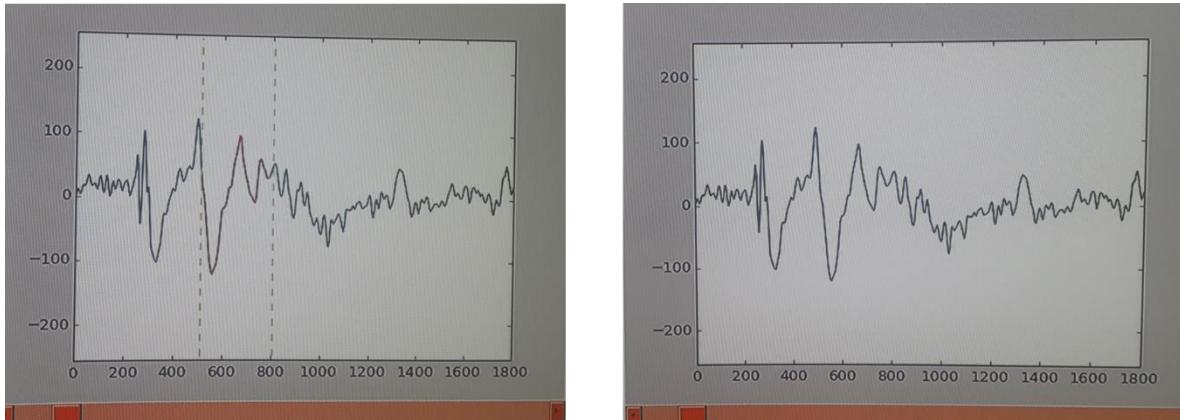
**Figura 59: a) Artefacto no detectado vs b) artefacto detectado por el método de correlación.**



Fuente: De los Autores.

En la figura se muestra un artefacto ocular que es detectado por el método de correlación con un índice de 0,5, pero no es detectado por el mismo método configurado con un índice de 0,6. Esto se debe a que el algoritmo se vuelve más selectivo mientras el índice se acerca a su valor máximo, que es 1.

**Figura 60: Diferencia entre los umbrales de detección Phyton**



Fuente: De los Autores.

El objetivo de esta sección es comprobar que los métodos seleccionados para este trabajo, muestran los mismos o similares resultados en simulación y en la implementación final en el sistema embebido.

#### 4.5.4 Comparación de Resultados Matlab, Python y Raspberry

En primera instancia se evaluarán los resultados provenientes de la evaluación a través de la base de datos de artefactos EOG, esto implica comparar los resultados de efectividad y los epochs que fueron determinados por el método como contaminados por artefactos entre Matlab Python, y Python en Raspberry. En la siguiente tabla se puede apreciar específicamente los índices mencionados.

**Tabla 34: Comparación de resultados de detección método Threshold de Amplitud**

THRESHOLD 3.5		THRESHOLD DE AMPLITUD RESULTADOS MATLAB		THRESHOLD DE AMPLITUD RESULTADOS PYTHON-PC		THRESHOLD DE AMPLITUD RESULTADOS PYTHON-RASPBERRY	
EPOCH Nro.	INTERVALO DE TIEMPO	ARTEFACTOS DETECTADOS	EFFECTIVIDAD	ARTEFACTOS DETECTADOS	EFFECTIVIDAD	ARTEFACTOS DETECTADOS	EFFECTIVIDAD
1	0 sg - 2 sg	NA	86 %	NA	86 %	NA	86 %
2	2 sg - 4 sg	NA		NA		NA	
3	4 sg - 6 sg	NA		NA		NA	
4	6 sg - 8 sg	NA		NA		NA	
5	8 sg - 10 sg	NA		NA		NA	
6	10 sg - 12 sg	SI		SI		SI	
7	12 sg - 14 sg	SI		SI		SI	
8	14 sg - 16 sg	SI		SI		SI	
9	16 sg - 18 sg	SI		SI		SI	
10	18 sg - 20 sg	SI		SI		SI	
11	20 sg - 22 sg	SI		SI		SI	
12	22 sg - 24 sg	SI		SI		SI	
13	24 sg - 26 sg	SI		SI		SI	
14	26 sg - 28 sg	SI		SI		SI	
15	28 sg - 30 sg	SI		SI		SI	
16	30 sg - 32 sg	SI		SI		SI	
17	32 sg - 34 sg	SI		SI		SI	
18	34 sg - 36 sg	SI		SI		SI	
19	36 sg - 38 sg	NO		NO		NO	
20	38 sg - 40 sg	NO		NO		NO	
21	40 sg - 42 sg	SI		SI		SI	
22	42 sg - 44 sg	SI		SI		SI	
23	44 sg - 46 sg	SI		SI		SI	
24	46 sg - 48 sg	SI		SI		SI	
25	48 sg - 50 sg	SI		SI		SI	
26	50 sg - 52 sg	SI		SI		SI	
27	52 sg - 54 sg	SI		SI		SI	
28	54 sg - 56 sg	SI		SI		SI	
29	56 sg - 58 sg	SI		SI		SI	
30	58 sg - 60 sg	SI		SI		SI	
31	60 sg - 62 sg	SI		SI		SI	
32	62 sg - 64 sg	SI		SI		SI	
33	64 sg - 66 sg	SI		SI		SI	
34	66 sg - 68 sg	SI		SI		SI	
35	68 sg - 70 sg	SI		SI		SI	
36	70 sg - 72 sg	SI		SI		SI	
37	72 sg - 74 sg	SI		SI		SI	
38	74 sg - 76 sg	NO		NO		NO	
39	76 sg - 78 sg	SI		SI		SI	
40	78 sg - 80 sg	SI		SI		SI	
41	80 sg - 82 sg	NO		NO		NO	
42	82 sg - 84 sg	SI		SI		SI	
43	84 sg - 86 sg	SI		SI		SI	
44	86 sg - 88 sg	SI		SI		SI	
45	88 sg - 90 sg	NO		NO		NO	
46	90 sg - 92 sg	NO		NO		NO	
47	92 sg - 94 sg	SI		SI		SI	
48	94 sg - 96 sg	NO		NO		NO	
49	96 sg - 98 sg	SI		SI		SI	
50	98 sg - 100 sg	SI		SI		SI	
51	100 sg - 102 sg	SI		SI		SI	
52	102 sg - 104 sg	SI		SI		SI	
53	104 sg - 106 sg	SI		SI		SI	
54	106 sg - 108 sg	SI		SI		SI	
55	108 sg - 110 sg	SI		SI		SI	
TOTAL DETECTADOS		43		43		43	

En la tabla anterior se puede tener la apreciación de los resultados obtenidos del método Threshold de amplitud en la detección de la presencia de artefactos en los diferentes epochs contenidos en la base de datos de artefactos EOG y a su vez se puede constatar la igualdad de estos resultados, analizando por ende que los índices de detección responden con iguales resultados de 86% en las dos diferentes herramientas de programación Matlab y Python y a su vez en diferentes sistemas operativos.

Posteriormente para este método se procedió a comparar los resultados obtenidos a través del test de validez aplicado.

**Tabla 35: Comparación de test de validez método Threshold de Amplitud**

Valor Threshold 3.5	Threshold de amplitud					
	Matlab		Python-Pc		Python-Raspberry	
	Registro 041222F9.bin	Registro 041222I6.bin	Registro 041222F9.bin	Registro 041222I6.bin	Registro 041222F9.bin	Registro 041222I6.bin
Sensibilidad	100%	87%	100%	87%	100%	87%
Especificidad	31%	100%	31%	100%	31%	100%
Valor Predictivo Positivo	57%	100%	57%	100%	57%	100%
Valor Predictivo Negativo	0%	60%	0%	60%	0%	60%
Exactitud	64%	88%	64%	88%	64%	88%
Exactitud Promedio	76%		76%		76%	

A través de los resultados ilustrados anteriormente se puede constatar con mayor veracidad que la aplicabilidad y resultados no dependen de la herramienta a utilizar.

Se realizaron las mismas comparaciones para el método Threshold de Pendiente con el fin de visualizar las posibles diferencias existentes, en primera instancia se compararon los resultados de detección de artefactos obtenidos de la base de datos de artefactos de EOG

**Tabla 36: Comparación de resultados de detección método Threshold de Pendiente**

THRESHOLD 4.0		THRESHOLD DE PENDIENTE RESULTADOS MATLAB		THRESHOLD DE PENDIENTE RESULTADOS PYTHON-PC		THRESHOLD DE PENDIENTE RESULTADOS PYTHON-RAPSERRY	
EPOCH Nro.	INTERVALO DE TIEMPO	ARTEFACTOS DETECTADOS	EFFECTIVIDAD	ARTEFACTOS DETECTADOS	EFFECTIVIDAD	ARTEFACTOS DETECTADOS	EFFECTIVIDAD
1	0 sg - 2 sg	SI		SI		SI	
2	2 sg - 4 sg	SI		SI		SI	
3	4 sg - 6 sg	SI		SI		SI	
4	6 sg - 8 sg	SI		SI		SI	
5	8 sg - 10 sg	SI		SI		SI	
6	10 sg - 12 sg	SI		SI		SI	
7	12 sg - 14 sg	SI		SI		SI	
8	14 sg - 16 sg	SI		SI		SI	
9	16 sg - 18 sg	SI		SI		SI	
10	18 sg - 20 sg	SI		SI		SI	
11	20 sg - 22 sg	SI		SI		SI	
12	22 sg - 24 sg	SI		SI		SI	
13	24 sg - 26 sg	SI		SI		SI	
14	26 sg - 28 sg	SI		SI		SI	
15	28 sg - 30 sg	NO		NO		NO	
16	30 sg - 32 sg	SI		SI		SI	
17	32 sg - 34 sg	SI		SI		SI	
18	34 sg - 36 sg	SI		SI		SI	
19	36 sg - 38 sg	SI		SI		SI	
20	38 sg - 40 sg	SI		SI		SI	
21	40 sg - 42 sg	SI		SI		SI	
22	42 sg - 44 sg	SI		SI		SI	
23	44 sg - 46 sg	SI		SI		SI	
24	46 sg - 48 sg	SI		SI		SI	
25	48 sg - 50 sg	SI		SI		SI	
26	50 sg - 52 sg	SI		SI		SI	
27	52 sg - 54 sg	SI		SI		SI	
28	54 sg - 56 sg	SI		SI		SI	
29	56 sg - 58 sg	SI		SI		SI	
30	58 sg - 60 sg	SI		SI		SI	
31	60 sg - 62 sg	SI		SI		SI	
32	62 sg - 64 sg	SI		SI		SI	
33	64 sg - 66 sg	SI		SI		SI	
34	66 sg - 68 sg	SI		SI		SI	
35	68 sg - 70 sg	SI		SI		SI	
36	70 sg - 72 sg	SI		SI		SI	
37	72 sg - 74 sg	SI		SI		SI	
38	74 sg - 76 sg	SI		SI		SI	
39	76 sg - 78 sg	SI		SI		SI	
40	78 sg - 80 sg	SI		SI		SI	
41	80 sg - 82 sg	NO		NO		NO	
42	82 sg - 84 sg	SI		SI		SI	
43	84 sg - 86 sg	SI		SI		SI	
44	86 sg - 88 sg	SI		SI		SI	
45	88 sg - 90 sg	SI		SI		SI	
46	90 sg - 92 sg	SI		SI		SI	
47	92 sg - 94 sg	SI		SI		SI	
48	94 sg - 96 sg	SI		SI		SI	
49	96 sg - 98 sg	SI		SI		SI	
50	98 sg - 100 sg	SI		SI		SI	
TOTAL DETECTADOS		48		48		48	

En la tabla anterior se puede visualizar los índices de efectividad conjunto a los epochs determinados como contaminados por el método en cada una de las herramientas empleadas y se puede evidenciar que dichos valores no difieren.

En las siguientes tablas se ilustrara la correspondiente comparación de los resultados del test de validez aplicado al método Threshold de Pendiente

**Tabla 37: Comparación de test de validez método Threshold de Pendiente**

Valor Threshold 4.0	Threshold de pendiente					
	Matlab		Python-Pc		Python-Raspberry	
	Registro 041222F9.bin	Registro 041222I6.bin	Registro 041222F9.bin	Registro 041222I6.bin	Registro 041222F9.bin	Registro 041222I6.bin
Sensibilidad	85%	96%	85%	96%	85%	96%
Especificidad	33%	50%	33%	50%	33%	50%
Valor Predictivo Positivo	58%	96%	58%	96%	58%	96%
Valor Predictivo Negativo	33%	50%	33%	50%	33%	50%
Exactitud	60%	92%	60%	92%	60%	92%
Exactitud Promedio	76%		76%		76%	

De forma similar al método de Threshold de pendiente los resultados para el método Threshold de pendiente no sufren variación alguna, por ende, se corrobora que la selección de la herramienta Python para llevar a cabo la programación de los métodos fue un buen recurso independientemente de la rapidez y la capacidad de ejecución que posee la tarjeta Raspberry

Para finalizar esta sección se realizó una comparación, utilizando el test de validez, del método de correlación. Como se esperaba, los resultados son idénticos en las tres herramientas, con diferencia únicamente en el tiempo de procesamiento, pues el sistema embebido tiene limitaciones en cuanto a desempeño. Los resultados se muestran en la tabla 38.

**Tabla 38: Comparación de test de validez método correlación**

	MATLAB		PYTHON		RASPBERRY	
	Registro 041222F9.bin	Registro 041222I6.bin	Registro 041222F9.bin	Registro 041222I6.bin	Registro 041222F9.bin	Registro 041222I6.bin
Índice/Threshold	5,0	5,0	5,0	5,0	5,0	5,0
Sensibilidad	83%	100%	83%	100%	83%	100%
Especificidad	74%	86%	74%	86%	74%	86%
Valor Predictivo Positivo	50%	95%	50%	95%	50%	95%
Valor Predictivo Negativo	7%	0%	7%	0%	7%	0%
Exactitud	60%	92%	60%	92%	60%	92%

## CAPÍTULO 5

### 5.1 Conclusiones

- El método threshold de amplitud es un método que detecta anomalías en la señal EEG a través de los valores de amplitud y para su ejecución requiere de un antecedente o historial de 10 segundos de la señal a examinar, por ende se vuelve dependiente de las condiciones de la señal evaluada, es decir que el índice de sensibilidad se verá afectado a razón de las fluctuaciones que se presentan en la señal. Por tal razón es mucho más eficiente su aplicabilidad en pacientes en estado de sedación, que en pacientes netamente despiertos o levemente sedados debido a que en los dos últimos casos la señal adquirida del paciente presenta mayor índice de artefactos musculares y artefactos oculares, lo que conlleva al que el promedio del antecedente necesario sea relativamente alto afectando directamente la sensibilidad del método. Al aplicar el método de Threshold de amplitud se observó que los resultados más óptimos y viales derivaban de un valor de Threshold de 3.5, pues a un valor mayor el método presentaba baja sensibilidad y a valores menores presenta una muy alta sensibilidad hasta el punto de detectar de forma errada muchos epochs como contaminados por artefactos
- El método Threshold de pendiente esta implementado con el fin de detectar los cambios bruscos y repentinos en la señal evaluada, por tal razón y como se pudo constatar el método presenta mayores detecciones cuando el paciente se encuentra consciente, es decir cuando las señales obtenidas presentan mayores fluctuaciones debido a la gran cantidad de artefactos adictivos presentes en la señal. A través de las evaluaciones realizadas se pudo determinar como valor óptimo un Threshold de 4,0 pues el método responde a un valor menor con demasiadas detecciones erradas lo cual no es muy productivo y repercute en muchos aspectos las falsas detecciones pero a su vez a un valor mayor el método perdía su rigurosidad y se hacía muy permisible, es decir su índice de sensibilidad se reducía, en consecuencia se disminuían las detecciones de artefactos.
- Al aplicar el método de correlación se observó que los mejores resultados se obtuvieron con un índice de 0,5. Esta conclusión es la misma para las dos herramientas de simulación (Matlab y Python) y la implementación final en el sistema embebido, Raspberry. La sensibilidad, es el valor que más importancia tiene para este trabajo, pues indica la probabilidad de clasificar correctamente en un epoch la presencia de un artefacto EOG, es decir, la probabilidad de que para un epoch contaminado se obtenga en la prueba con el algoritmo de correlación resultado positivo. En este parámetro los resultados del método de correlación

están en un rango de 83,3 a 100%, lo cual demuestra la eficiencia del método en la detección de artefactos EOG. Este método no pudo ser aplicado en la detección de artefactos de naturaleza estocástica, como los artefactos musculares, pues estos no siguen un patrón determinado, el cual es fundamental en la implementación de este método.

- Los métodos presentan diferentes campos de acción, es decir el método Threshold de Amplitud se aplica con objetivos de detectar EOG, EMG, SPIKES; el método Threshold de Pendiente tiene como finalidad detectar la presencia de artefactos EMG, y los artefactos que son productos de fuentes no fisiológicos como 50Hz, SPIKES, etc. Y el método de correlación tiene como finalidad la detección de los artefactos propios de EOG; por tal razón no es posible realizar una correcta comparación entre los resultados obtenidos de los métodos; sin embargo los tres métodos implementados presentan índices de sensibilidad altos.
- En general, los algoritmos implementados para la detección de artefactos en señales EEG de un solo canal, muestran resultados aceptables. No fue posible alcanzar el 100% de detección de artefactos con los métodos implementados en este trabajo. La reducción de artefactos fue posible utilizando filtros en el pre-procesamiento de la señal EEG. Una opción para reducir en mayor medida los artefactos en la señal EEG es eliminar el epoch contaminado del registro, esta opción no se implementó pues conlleva una pérdida de datos que resulta inaceptable.
- El procesamiento de la señal, específicamente la aplicación de los filtros notch y pasa bajas redujo en gran medida la presencia de artefactos de alta frecuencia como los provocados por la frecuencia de la red eléctrica o los provenientes de la actividad muscular
- Se comprobó la facilidad de migración entre las herramientas de programación Matlab y Python debido a gran porcentaje de similitud entre ellos gracias a las librerías enfocadas en el entorno de Matlab.
- A través de la implementación de los métodos en la tarjeta Raspberry se pudo constatar la eficiencia de este dispositivo en cuanto al procesamiento de datos y al razonamiento cuantitativo, pero es poco eficiente para efectos de construcción de gráficas y visualización de datos dinámicamente debido a que esta se ve limitada ya que el solo hecho de que la Raspberry ejecute el sistema operativo genera un gran consumo de recursos debido a que el Raspbian es un sistema operativo que

ejecuta un entorno gráfico con el fin de ser muy amigable con el usuario, por esta razón al ejecutar más interfaces gráficas genera sobrecargas en el procesador

## **5.2 Recomendaciones**

- En el desarrollo del trabajo, fue evidente que el sistema embebido seleccionado no tenía una capacidad apropiada de procesamiento. Con el fin de mejorar el tiempo de procesamiento es recomendable mejorar o cambiar el sistema embebido para que tenga mayor capacidad de procesamiento, en especial procesamiento de gráficos.
- Los métodos implementados, muestran buenos resultados individualmente, pero cada uno muestra mayor sensibilidad a un tipo específico de artefacto, por ejemplo, el método de correlación es altamente sensible a artefactos oculares. Es posible que al implementar un programa que aplique los tres métodos simultáneamente, los resultados obtenidos sean superiores a los resultados obtenidos aplicando los métodos individualmente.
- Es posible modificar el programa para utilizar los puertos GPIO como un sistema de adquisición de datos para la señal de un electroencefalógrafo, con el fin de adquirir y procesar la señal en tiempo real. Esto se debe implementar, preferiblemente, utilizando un sistema operativo en tiempo real, RTOS, para asegurar un tiempo de procesamiento muy corto.

## ANEXO 1: CÓDIGO DE MÉTODOS IMPLEMENTADOS EN MATLAB

```
function varargout = verepoch(varargin)
% VEREPOCH M-file for verepoch.fig
%     VEREPOCH, by itself, creates a new VEREPOCH or raises the existing
%     singleton*.
%
%     H = VEREPOCH returns the handle to a new VEREPOCH or the handle to
%     the existing singleton*.
%
%     VEREPOCH('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in VEREPOCH.M with the given input
%     arguments.
%
%     VEREPOCH('Property','Value',...) creates a new VEREPOCH or raises
%     the
%     existing singleton*. Starting from the left, property value pairs
%     are
%     applied to the GUI before verepoch_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property
%     application
%     stop. All inputs are passed to verepoch_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
%     one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help verepoch

% Last Modified by GUIDE v2.5 29-Apr-2015 14:37:23

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',         mfilename, ...
                   'gui_Singleton',   gui_Singleton, ...
                   'gui_OpeningFcn', @verepoch_OpeningFcn, ...
                   'gui_OutputFcn',  @verepoch_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```

% --- Executes just before verepoch is made visible.
function verepoch_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to verepoch (see VARARGIN)

% Choose default command line output for verepoch
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
a=imread('C:\Users\carmen elena\Desktop\Matlab Tesis\unisanbue.jpg');
axes(handles.axes3)
imshow(a);

% UIWAIT makes verepoch wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = verepoch_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
global Data DataPts senalreal cadena artef2 artef3 artef4 a1 a2 b1 b2
FileName
% hObject    handle to pushbutton1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Data.rutaorigen='C:\Users\carmen elena\Desktop\Matlab Tesis'
%Data.name='041221JM'
%Data.filename='041221JM.BIN'
[FileName Path]=uigetfile({'*.BIN'}, 'ABRIR REGISTRO');
rutaserie=[Path FileName];
%rutaserie=[Data.rutaorigen '\' Data.filename]; %archivo con
datos crudos del EEG
fid=fopen(rutaserie);

```

```

DataPts.aep=fread(fid,'uint16'); %reads a binary file in integer
format. The vector aep contains all data from the file and is therefore
large
st=fclose(fid);

%resample: 900 --> 1024
DataPts.aep=resample(DataPts.aep,900,900); %resample applies an
anti-aliasing (lowpass) FIR filter. It designs the filter using fircls
with a Kaiser window
Data.fs=900;
fs=Data.fs;
%changing scale
%DataPts.aep=DataPts.aep(1:450000)%datos recortados para comparar
con python 100 epochs =180000 muestras
senalreal=DataPts.aep/65535*950; %cambio de amplitud para señal real
%DataPts.aep = round((DataPts.aep - 32768)*0.4 +31000 );
senalreal= round((senalreal - mean(senalreal)));
load('EEG_L.mat')
senalreal=[EEG_L; senalreal];
cantepochs=length(DataPts.aep)/1800;%cantidad de epochs en el
registro
artef2=zeros(floor(cantepochs),1); %inicializacion de banco de
artefactos de metodo1
artef3=zeros(floor(cantepochs),1); %inicializacion de banco de
artefactos de metodo2
artef4=zeros(floor(cantepochs),1);
maximo=length(DataPts.aep)/(Data.fs*2);
set(handles.slider1,'Max',maximo);
set(handles.slider1,'SliderStep',[1/maximo 1]);
Data.slider_value = get(handles.slider1,'Value');
%%%%-----diseño de filtros-----
%filt notch 50Hz
%#sampling rate
f0 = 50; %#notch frequency
fn = fs/2; %#Nyquist frequency
freqRatio = f0/fn; %#ratio of notch freq. to Nyquist freq.

notchWidth = 0.1; %#width of the notch

%Compute zeros
xeros = [exp( sqrt(-1)*pi*freqRatio ), exp( -sqrt(-1)*pi*freqRatio
)];

%#Compute poles
poles = (1-notchWidth) * xeros;
b2 = poly( xeros ); %# Get moving average filter coefficients
a2 = poly( poles ); %# Get autoregressive filter coefficients

%pasa banda 50Hz
[b1,a1] = cheby2(6,10,50/(fs/2));
grafica(handles)

```

```

function grafica(handles)
global FileName Data DataPts senalreal i epoch cadena opcion filt50
threshold filt40 artef2 artef3 artef4 a1 a2 b1 b2
%%%%%%%%%%%%%
%G=[0.0438;0.0379;0.0347;0.1834;1];
%SOS=[1 2 1 1 -1.6509 0.8262; 1 2 1 1 -1.4274 0.5790; 1 2 1 1 -1.3052
0.4438; 1 1 0 1 -0.6332 0];
fs=Data.fs;
win=2*fs; % ventana de 2 segundos.
[b,a] = cheby2(6,60,127/(fs/2));
epochfilt=filtfilt(b,a,senalreal);
epochfilt=filtfilt(SOS,G,DataPts.aep);

filt50=get(handles.filt,'Value');
filt40=get(handles.PB40,'Value');
%
if filt50==1
    %#filter signal
    epochfilt = filter(b2,a2,epochfilt);
end
if filt40==1
    epochfilt = filter(b1,a1, epochfilt);
end
i=floor(Data.slider_value);
noart=1;
threshold=str2double(get(handles.edit1,'String')) %returns contents of
edit1 as a double

if (i+1)*win<=length(DataPts.aep)

    epoch= epochfilt((i*win)+1:(i+1)*win); %toma un tramo de 2 segundos
    del EEG en el dominio del tiempo
    contador=0;

opcion=get(handles.menu,'Value');
if(opcion==2)
    if i>4
        winart= epochfilt(((i-5)*win)+1:((i)*win));
        for k=1:1:length(epoch)
            promedioEEG=mean(abs(winart));
            mstsig=abs(epoch(k));
            winart=vertcat(winart,mstsig);
            winart(1)=[];
            if mstsig>(threshold*promedioEEG)
                contador=contador+1;
                ubicacion(contador,:,:)= (i*fs+k);
                ubicaciongrafica(contador,:,:)=k;
            end
        end
    end
    set(handles.artefacto,'value',artef2(i+1,1));
    if artef2(i+1,1)==0

```

```

        set(handles.text8,'BackgroundColor',[1 1 1]);
        set(handles.text9,'BackgroundColor',[0 1 0]);
    end
    if artef2(i+1,1)==1
        set(handles.text8,'BackgroundColor',[0 1 0]);
        set(handles.text9,'BackgroundColor',[1 1 1]);
    end
end
if (opcion==3)
    for k=1:1:length(epoch)-1
        muestra=epoch(k);
        muestrasig=epoch(k+1);
        diferencia=abs (muestra-muestrasig);
        if diferencia>=threshold;
            contador=contador+1;
            ubicaciongrafica(contador,:,1)=k+1;
        end
    end
    set(handles.artefacto,'value',artef3(i+1,1));
    if artef3(i+1,1)==0
        set(handles.text8,'BackgroundColor',[1 1 1]);
        set(handles.text9,'BackgroundColor',[0 1 0]);
    end
    if artef3(i+1,1)==1
        set(handles.text8,'BackgroundColor',[0 1 0]);
        set(handles.text9,'BackgroundColor',[1 1 1]);
    end
end
if (opcion==4)
    load('PatronFinal.mat')
    Time = 0:1:1799;
    patron = patron_final; % Seleccion de patron
    tamano=(length(epoch)-length(patron));
    ceros =(1:tamano)';
    ceros=ceros.*0;
    patron1 = [patron ; ceros];
    [xCorr,lags] = xcorr(epoch,patron1,'coeff'); % correlacion entre el
    patron y el epoch contaminad

    [a,I] = max(abs(xCorr)); % Indica la posicion en la cual se
    encuentra la mayor correlacion
    maxt = lags(I); % Indica la muestra en la cual se encuentra la
    mayor correlacion
    if a>=0.47
        Vvacio = NaN(size(epoch)); % Vector de NaN
        if maxt<0

            Vvacio(1:maxt+length(patron)) = epoch(1:maxt+length(patron));

        end
        if maxt>=0&&maxt<=1500
            Vvacio(maxt+1:maxt+length(patron)) =
epoch(maxt+1:maxt+length(patron));
        end
    end
end

```

```

        end
        if maxt>1500
            Vvacio(maxt:length(epoch)) = epoch(maxt:length(epoch));
        end
        noart=0
    end
    set(handles.artefacto,'value',artef4(i+1,1));
    if artef4(i+1,1)==0
        set(handles.text8,'BackgroundColor',[1 1 1]);
        set(handles.text9,'BackgroundColor',[0 1 0]);
    end
    if artef4(i+1,1)==1
        set(handles.text8,'BackgroundColor',[0 1 0]);
        set(handles.text9,'BackgroundColor',[1 1 1]);
    end
end

else
    epoch=ones(win,1);
end

set(handles.inicio,'String',i*2)
set(handles.fin,'String',(i+1)*2)
axes(handles.axes1)
hold off
if opcion==4 && noart==0
plot(Time,epoch,'k',Time,Vvacio,'r')
title('Epoch de señal EEG')
xlabel('muestras')
ylabel('Amplitud (uV)')
ylim([-250,250])
xlim([0,1800])
grid off
% axis tight

hold on
%
m = -250;
M = 250;
inicio = (maxt+1);
fin = (maxt+length(patron));
plot([inicio fin;inicio fin],[m m;M M],'y--')
noart=1;
else
plot(epoch,'k');
ylim([-250 250]);
if contador~=0
    for n=1:1:contador
        hold on
        plot( [ubicaciongrafica(n) ubicaciongrafica(n)], [-250
epoch(ubicaciongrafica(n))], 'y--')

```

```

    end
end
title('Epoch de señal EEG')
xlabel('muestras')
ylabel('Amplitud (uV)')
end
L = length(epoch);
WHam=hamming(L);
epochham=epoch.*WHam;
if opcion==2
if contador~=0
for n=1:1:contador
    hold on
    plot( [ubicaciongrafica(n) ubicaciongrafica(n)], [-250
epoch(ubicaciongrafica(n))], 'y--')
end
end
end
if opcion==3
if contador>=25
for n=1:1:contador
    hold on
    plot( [ubicaciongrafica(n) ubicaciongrafica(n)], [-250
epoch(ubicaciongrafica(n))], 'y--')
end
end
end
NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(epochham,NFFT)/L;
f = fs/2*linspace(0,1,NFFT/2+1);
axes(handles.axes2)
plot(f,2*abs(Y(1:NFFT/2+1)), 'r')
title('Espectro de frecuencia de epoch')
xlabel('Frecuencia (Hz)')
ylabel('|Y(f)|')
ylim([0 20]);
xlim([0 150]);
cadena='registro_';
cadena=horzcat(cadena,FileName, '_segundo_');
cadena=[cadena mat2str(i*2) '-' mat2str((i+1)*2)]
```

```

% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Data DataPts

Data.slider_value=get(hObject, 'Value') ;

grafica (handles)
```

```

% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns
% called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
global cadena epoch artef2 artef3 i

uisave('epoch',cadena);
% hObject    handle to pushbutton3 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function inicio2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inicio2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns
% called

% --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns
% called

% Hint: place code in OpeningFcn to populate axes2

% --- Executes on selection change in menu.
function menu_Callback(hObject, eventdata, handles)

```

```

% hObject    handle to menu (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns menu contents
%        as cell array
%        contents{get(hObject,'Value')} returns selected item from menu
global opcion Data DataPts

opcion=get(hObject, 'Value');

grafica(handles);

% --- Executes during object creation, after setting all properties.
function menu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to menu (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns
% called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

% --- Executes on button press in filt.
function filt_Callback(hObject, eventdata, handles)
% hObject    handle to filt (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hint: get(hObject, 'Value') returns toggle state of filt
global filt50 Data DataPts

filt50=get(hObject, 'Value');

grafica(handles);

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
global threshold Data DataPts
    threshold=str2double(get(hObject, 'String')) %returns contents of
edit1 as a double
grafica(handles);

```

```

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns
% called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end


% --- Executes on button press in PB40.
function PB40_Callback(hObject, eventdata, handles)
% hObject    handle to PB40 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hint: get(hObject, 'Value') returns toggle state of PB40
global filt40 Data DataPts
filt40=get(hObject, 'Value');
grafica(handles);

% --- Executes on button press in artefacto.
function artefacto_Callback(hObject, eventdata, handles)
% hObject    handle to artefacto (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
global contaminado artef2 artef3 artef4 i opcion
contaminado=get(hObject, 'Value') %returns toggle state of artefacto
if contaminado==1;

    set(handles.text9, 'BackgroundColor', [1 1 1]);
    set(handles.text8, 'BackgroundColor', [0 1 0]);
    if opcion==2
        artef2(i+1,1)=1;
    end
    if opcion==3
        artef3(i+1,1)=1;
    end
    if opcion==4
        artef4(i+1,1)=1;
    end
end
if contaminado==0;
    set(handles.text8, 'BackgroundColor', [1 1 1]);
    set(handles.text9, 'BackgroundColor', [0 1 0]);
    if opcion==2

```

```
    artef2(i+1,1)=0;
end
if opcion==3
    artef3(i+1,1)=0;
end
if opcion==4
    artef4(i+1,1)=0;
end
end
```

## ANEXO 2: CÓDIGO DE INTERFACE DISEÑADA EN PYTHON

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'interfaz.ui'
#
# Created: Sun Oct 11 01:47:03 2015
#   by: PyQt4 UI code generator 4.11.3
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)

class Ui_Dialog(object):
    def setupUi(self, Dialog):
        Dialog.setObjectName(_fromUtf8("Dialog"))
        Dialog.resize(743, 511)
        Dialog.setStyleSheet(_fromUtf8("background-color: rgb(255, 255, 255);\n"))
        self.labelTitulo = QtGui.QLabel(Dialog)
        self.labelTitulo.setGeometry(QtCore.QRect(277, 1, 401, 91))
        font = QtGui.QFont()
        font.setFamily(_fromUtf8("Arial"))
        font.setPointSize(14)
        font.setBold(True)
        font.setWeight(75)
        self.labelTitulo.setFont(font)
        self.labelTitulo.setTextFormat(QtCore.Qt.AutoText)
```

```

self.labelX.setScaledContents(False)
self.labelX.setAlignment(QtCore.Qt.AlignCenter)
self.labelX.setWordWrap(True)
self.labelX.setObjectName(_fromUtf8("labelTitulo"))
self.labelX_2 = QtGui.QLabel(Dialog)
self.labelX_2.setGeometry(QtCore.QRect(10, 190, 46, 13))
font = QtGui.QFont()
font.setFamily(_fromUtf8("Arial"))
font.setBold(True)
font.setWeight(75)
self.labelX_2.setFont(font)
self.labelX_2.setObjectName(_fromUtf8("label_2"))
self.checkBox = QtGui.QCheckBox(Dialog)
self.checkBox.setGeometry(QtCore.QRect(21, 221, 131, 17))
self.checkBox.setCheckable(True)
self.checkBox.setChecked(False)
self.checkBox.setTristate(False)
self.checkBox.setObjectName(_fromUtf8("checkBox"))
self.checkBox_2 = QtGui.QCheckBox(Dialog)
self.checkBox_2.setEnabled(True)
self.checkBox_2.setGeometry(QtCore.QRect(21, 244, 141, 17))
self.checkBox_2.setObjectName(_fromUtf8("checkBox_2"))
self.labelX_3 = QtGui.QLabel(Dialog)
self.labelX_3.setGeometry(QtCore.QRect(10, 279, 201, 31))
font = QtGui.QFont()
font.setFamily(_fromUtf8("Arial"))
font.setBold(True)
font.setWeight(75)
self.labelX_3.setFont(font)
self.labelX_3.setWordWrap(True)
self.labelX_3.setObjectName(_fromUtf8("label_3"))
self.horizontalScrollBar = QtGui.QScrollBar(Dialog)
self.horizontalScrollBar.setGeometry(QtCore.QRect(220, 480, 511, 20))
self.horizontalScrollBar.setAutoFillBackground(False)
self.horizontalScrollBar.setStyleSheet(_fromUtf8("background-color:    rgb(255,    85,
0);\n" "border-top-color: rgb(0, 0, 0);"))
self.horizontalScrollBar.setPageStep(1)
self.horizontalScrollBar.setProperty("value", 0)
self.horizontalScrollBar.setSliderPosition(0)
self.horizontalScrollBar.setOrientation(QtCore.Qt.Horizontal)
self.horizontalScrollBar.setInvertedAppearance(False)

```

```

self.horizontalScrollBar.setObjectName(_fromUtf8("horizontalScrollBar"))
self.pushButton_2 = QtGui.QPushButton(Dialog)
self.pushButton_2.setGeometry(QtCore.QRect(70, 430, 75, 23))
self.pushButton_2.setStyleSheet(_fromUtf8("background-color: rgb(216, 216, 216);"))
self.pushButton_2.setCheckable(False)
self.pushButton_2.setChecked(False)
self.pushButton_2.setAutoRepeat(False)
self.pushButton_2.setDefault(False)
self.pushButton_2.setFlat(False)
self.pushButton_2.setObjectName(_fromUtf8("pushButton_2"))
self.groupBox = QtGui.QGroupBox(Dialog)
self.groupBox.setGeometry(QtCore.QRect(10, 310, 201, 101))
self.groupBox.setCheckable(False)
self.groupBox.setObjectName(_fromUtf8("groupBox"))
self.radioButton_2 = QtGui.QRadioButton(self.groupBox)
self.radioButton_2.setGeometry(QtCore.QRect(6, 30, 191, 17))
self.radioButton_2.setObjectName(_fromUtf8("radioButton_2"))
self.radioButton_3 = QtGui.QRadioButton(self.groupBox)
self.radioButton_3.setGeometry(QtCore.QRect(6, 50, 191, 17))
self.radioButton_3.setObjectName(_fromUtf8("radioButton_3"))
self.radioButton_4 = QtGui.QRadioButton(self.groupBox)
self.radioButton_4.setGeometry(QtCore.QRect(6, 71, 191, 17))
self.radioButton_4.setObjectName(_fromUtf8("radioButton_4"))
self.radioButton = QtGui.QRadioButton(self.groupBox)
self.radioButton.setGeometry(QtCore.QRect(6, 9, 181, 17))
self.radioButton.setObjectName(_fromUtf8("radioButton"))
self.mplwindow = QtGui.QWidget(Dialog)
self.mplwindow.setGeometry(QtCore.QRect(220, 119, 511, 361))
sizePolicy = QtGui.QSizePolicy(QtGui.QSizePolicy.Expanding,
QtGui.QSizePolicy.Preferred)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.mplwindow.sizePolicy().hasHeightForWidth())
self.mplwindow.setSizePolicy(sizePolicy)
self.mplwindow.setMaximumSize(QtCore.QSize(16777215, 16777215))
self.mplwindow.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
self.mplwindow.setContextMenuPolicy(QtCore.Qt.NoContextMenu)
self.mplwindow.setObjectName(_fromUtf8("mplwindow"))
self.mplvl = QtGui.QVBoxLayout(self.mplwindow)
self.mplvl.setSpacing(6)
self.mplvl.setMargin(1)
self.mplvl.setObjectName(_fromUtf8("mplvl"))

```

```
self.label = QtGui.QLabel(Dialog)
self.label.setGeometry(QtCore.QRect(237, 97, 81, 20))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
font.setWeight(50)
self.label.setFont(font)
self.label.setObjectName(_fromUtf8("label"))
self.label_4 = QtGui.QLabel(Dialog)
self.label_4.setGeometry(QtCore.QRect(317, 97, 31, 20))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
font.setWeight(50)
self.label_4.setFont(font)
self.label_4.setText(_fromUtf8(""))
self.label_4.setObjectName(_fromUtf8("label_4"))
self.label_5 = QtGui.QLabel(Dialog)
self.label_5.setGeometry(QtCore.QRect(367, 97, 111, 20))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
font.setWeight(50)
self.label_5.setFont(font)
self.label_5.setObjectName(_fromUtf8("label_5"))
self.label_6 = QtGui.QLabel(Dialog)
self.label_6.setGeometry(QtCore.QRect(473, 97, 31, 20))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
font.setWeight(50)
font.setKerning(False)
self.label_6.setFont(font)
self.label_6.setText(_fromUtf8(""))
self.label_6.setObjectName(_fromUtf8("label_6"))
self.label_7 = QtGui.QLabel(Dialog)
self.label_7.setGeometry(QtCore.QRect(499, 97, 31, 20))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
font.setWeight(50)
font.setKerning(False)
```

```

self.label_7.setFont(font)
self.label_7.setObjectName(_fromUtf8("label_7"))
self.label_8 = QtGui.QLabel(Dialog)
self.label_8.setGeometry(QtCore.QRect(541, 97, 31, 20))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
font.setWeight(50)
font.setKerning(False)
self.label_8.setFont(font)
self.label_8.setText(_fromUtf8(""))
self.label_8.setObjectName(_fromUtf8("label_8"))
self.label_9 = QtGui.QLabel(Dialog)
self.label_9.setGeometry(QtCore.QRect(567, 97, 31, 20))
font = QtGui.QFont()
font.setPointSize(10)
font.setBold(False)
font.setWeight(50)
font.setKerning(False)
self.label_9.setFont(font)
self.label_9.setObjectName(_fromUtf8("label_9"))
self.label_10 = QtGui.QLabel(Dialog)
self.label_10.setGeometry(QtCore.QRect(8, 1, 141, 151))
self.label_10.setText(_fromUtf8(""))
self.label_10.setObjectName(_fromUtf8("label_10"))

self.retranslateUi(Dialog)
QtCore.QMetaObject.connectSlotsByName(Dialog)

def retranslateUi(self, Dialog):
    Dialog.setWindowTitle(_translate("Dialog", "Dialog", None))
    self.labelTitulo.setText(_translate("Dialog", "Sistema Embebido Destinado al
Reconocimiento de Artefactos en Señales EEG", None))
    self.label_2.setText(_translate("Dialog", "Filtros:", None))
    self.checkBox.setText(_translate("Dialog", "Butterworth", None))
    self.checkBox_2.setText(_translate("Dialog", "Notch 50Hz", None))
    self.label_3.setText(_translate("Dialog", "Metodods para la          deteccion de
artefactos", None))
    self.pushButton_2.setText(_translate("Dialog", "Inicio", None))
    self.radioButton_2.setText(_translate("Dialog", "Correlacion", None))
    self.radioButton_3.setText(_translate("Dialog", "Threshold amplitud", None))
    self.radioButton_4.setText(_translate("Dialog", "Señal", None))

```

```
self.radioButton.setText(_translate("Dialog", "Threshold Pendiente", None))
self.label.setText(_translate("Dialog", "EPOCH Nro:", None))
self.label_5.setText(_translate("Dialog", "RANGO TIEMPO:", None))
self.label_7.setText(_translate("Dialog", "S --", None))
self.label_9.setText(_translate("Dialog", "S", None))
```

### ANEXO 3: CÓDIGO DE MÉTODOS IMPLEMENTADOS EN PYTHON

```
import os, sys
import tkinter as tk
from tkinter import filedialog
from interfaz import *
from PyQt4 import QtCore, QtGui
from numpy import *
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt
import scipy.io as sio
from matplotlib.figure import Figure
from matplotlib.backends.backend_qt4agg import (
    FigureCanvasQTAgg as FigureCanvas,
    NavigationToolbar2QT as NavigationToolbar)
from PyQt4 import QtGui
class Proyecto(QtGui.QDialog):
    global fig1
    pulsador=0;
    fig1=plt.figure()
    def __init__(self, parent=None):
        QtGui.QWidget.__init__(self, parent)
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.ui.label_10.setPixmap(QtGui.QPixmap(os.getcwd() + "/universidad3.jpg"))
        QtCore.QObject.connect(self.ui.pushButton_2, QtCore.SIGNAL('clicked()'), self.iniciar)
        QtCore.QObject.connect(self.ui.checkBox, QtCore.SIGNAL('clicked()'), self.clearplot)
        QtCore.QObject.connect(self.ui.checkBox_2, QtCore.SIGNAL('clicked()'), self.clearplot)
        self.ui.horizontalScrollBar.valueChanged.connect(self.clearplot)
        QtCore.QObject.connect(self.ui.radioButton, QtCore.SIGNAL('clicked()'),
        self.clearplot)
        QtCore.QObject.connect(self.ui.radioButton_2, QtCore.SIGNAL('clicked()'),
        self.clearplot)
        QtCore.QObject.connect(self.ui.radioButton_3, QtCore.SIGNAL('clicked()'),
        self.clearplot)
        QtCore.QObject.connect(self.ui.radioButton_4, QtCore.SIGNAL('clicked()'),
        self.clearplot)
    def clearplot(self):
        self.ui.mplvl.removeWidget(self.canvas)
        self.funciongrafica2()
```

```

def funciongrafica2(self):
    self.funciongrafica()
def addmpl(self, fig):
    self.canvas = FigureCanvas(fig)
    self.ui.mplvl.addWidget(self.canvas)
    self.canvas.draw()
def iniciar(self):
    global a,b,a1,b1,a2,b2,fs,win,eegdef,mat

    stot=array('I')
    s=array([0,0],dtype=uint8)
    eeg=array([],dtype=int16)
    eegreal=array([],dtype=int16)
    eegreal.dtype=int16
    mat=array([],dtype=int16)
    root = tk.Tk()
    root.withdraw()
    registro = filedialog.askopenfilename()
    f=open(registro,'rb') #vamos al final del archivo
    stot=f.read()
    f.close
    fin = len(stot)
    dato=0
    k=0;
    win=1800
    fs=900
    #####-----diseño de filtros-----
    #filtro pasa baja
    b,a=signal.cheby2(6,60,(127/(900/2)));
    #filt notch 50Hz
        #sampling rate
    f0 = 50;          #notch frequency
    fn = fs/2;         #Nyquist frequency
    freqRatio = f0/fn;   #ratio of notch freq. to Nyquist freq.

    notchWidth = 0.1;    #width of the notch
    r=0.0000 + 1.0000j
        #Compute zeros
    xeros = [np.exp( r*np.pi*freqRatio ), np.exp( -r*np.pi*freqRatio )];
    xeros=np.around(xeros,3)
        #Compute poles
    poles = (1-notchWidth) * xeros;

```

```

b2 = poly( xeros ); # Get moving average filter coefficients
a2 = poly( poles ); # Get autoregressive filter coefficients
#filtro pasa baja 50 Hz
b1,a1 = signal.cheby2(6,10,50/(fs/2));
while k< 360000: # valor para cargar 20 epochs
    s[1] = stot[k]
    s[0] = stot[k+1]
    info=int.from_bytes(s, byteorder='big')
    dato2=uint16(info)
    eeg=np.append(eeg,dato2)
    eeg=uint16(eeg)
    k+=2;
    eeg=eeg/65535*950
    promedio=round(np.mean(eeg))
    eegreal=eeg-promedio
    eegdef=signal.filtfilt(b,a,eegreal, padlen=None)
    archivo = open('patron.txt')
    for linea in archivo:
        mat=np.append(mat,linea.strip().split())
    archivo.close()
    self.funciongrafica()
def funciongrafica(self):
    global a,b,a1,b1,a2,b2,fs,win,eegdef,mat,i,noart,Time,epoch,Vvacio,ubicaciongrafica,
    contador,maxt,patron
    ubicaciongrafica=array([],dtype=uint16)
    print(self.ui.horizontalScrollBar.singleStep(),self.ui.horizontalScrollBar.pageStep())
    i=self.ui.horizontalScrollBar.sliderPosition()
    eegfilt=eegdef
    self.ui.label_4.setText(str(i+1))
    self.ui.label_6.setText(str(i*2))
    self.ui.label_8.setText(str((i+1)*2))
    if self.ui.checkBox_2.isChecked():
        eegfilt=signal.filtfilt(b2,a2,eegfilt, padlen=None)
    if self.ui.checkBox.isChecked():
        eegfilt=signal.filtfilt(b1,a1,eegfilt, padlen=None)
    print(eegfilt.shape)
    epoch= eegfilt[(i*win):(i+1)*win]
    k2=0;
    thresholdslope=4.2;
    thresholdamplitud=3.7
    noart=1;
    contador=0;

```

```

s=0
if self.ui.radioButton_3.isChecked():
    if i>4:
        winart= eegfilt[((i-5)*win):((i)*win)];
        while k2<len(epoch)-1:
            promedioEEG=np.mean(abs(winart));
            mstsig=abs(epoch[k2]);
            winart=np.append(winart,mstsig);
            winart=np.delete(winart,0)
            if mstsig>(thresholdamplitud*promedioEEG):
                contador=contador+1;
                #ubicacion[contador,:,1]=(i*fs+k2);
                ubicaciongrafica=np.append(ubicaciongrafica,k2+1)
            k2+=1;

elif self.ui.radioButton.isChecked():
    while k2<size(epoch)-1:
        muestra=epoch[k2];
        muestrasig=epoch[k2+1];
        diferencia=abs(muestra-muestrasig);
        if diferencia>=thresholdslope:
            contador=contador+1;
            ubicaciongrafica=np.append(ubicaciongrafica,k2+1)
        k2+=1;

elif self.ui.radioButton_2.isChecked():
    Time=np.linspace(0, 1800, num=1800)
    patron=mat
    tamano=(len(epoch)-len(patron));
    ceros=zeros(tamano)
    patron1=np.append(patron,ceros)
    patron1=float64(patron1)
    flags=array([],dtype=float64)
    xcorr=array([],dtype=float64)
    print(len(epoch), len(patron1), len(ceros),(tamano), len(patron), epoch.shape)
    flags,xcorr,nada1,nada2=plt.xcorr(epoch, patron1, usevlines=True, maxlags=None,
normed=True, lw=2)
    maximo=np.amax(xcorr)
    indice=np.where(xcorr==maximo)
    maxt=flags[indice]
    Vvacio=array([],dtype=float64)

```

```

if maximo>=0.47:
    s=0
    while s<len(epoch):
        Vvacio=np.append(Vvacio,np.NaN)# Vector de NaN
        s+=1
    if maxt<0:

        Vvacio[1:maxt+len(patron)] = epoch[1:maxt+len(patron)];

    if maxt>=0 and maxt<=1500:

        Vvacio[maxt+1:maxt+len(patron)] = epoch[maxt+1:maxt+len(patron)]
    if maxt>1500:
        Vvacio[maxt:len(epoch)] = epoch[maxt:len(epoch)]
        noart=0
    self.plotear()
def plotear(self):
    global noart,Time,epoch,Vvacio,ubicaciongrafica, contador,maxt,patron
    if self.ui.radioButton_2.isChecked() and noart==0:
        plt.clf()
        plt.plot(Time,epoch,'k',Time,Vvacio,'r')
        plt.axis([0, 1800, -255, 255])
        m = -250;
        M = 250;
        inicio = (maxt+1);
        fin = (maxt+len(patron));
        plt.plot([inicio, inicio],[m,M],'y--')
        plt.plot([fin, fin],[m,M],'y--')
        print("este epoch esta contmainado por EOG")
    if noart==1 or self.ui.radioButton_4.isChecked():
        plt.clf()
        plt.plot(epoch,'k')
        plt.axis([0, 1800, -255, 255])
        plt.ylabel("Amplitud (uv)")
        plt.xlabel("muestras")
        plt.title("EPOCH SEÑAL EEG")
        k=0;
        while k<=contador-1:
            limite=ubicaciongrafica[k]
            plt.plot( [limite,limite], [-250,epoch[limite]],'y--')
            k+=1
        if k>=100:

```

```
    print("epoch contamindado por 50 Hz y EMG")
else:
    print("epoch aprobado para analizar")
    self.addmpl(fig1)
if __name__=="__main__":
    app=QtGui.QApplication(sys.argv)
    myapp=Proyecto()
    myapp.show()

sys.exit(app.exec_())
```

## LISTA DE REFERENCIAS

- Akobeng, A. K. (2006). Understanding diagnostic tests 1: sensitivity, specificity and predictive values. *Acta Paediatrica*, 338-341.
- Arango, R., & Naranjo, J. J. (2010). Desarrollo de una herramienta de arquitectura abierta para la visualización y análisis de señales EEG. *Revista de investigaciones de Uniquindío*.
- Arias, H. F., Alzate, S. L., & Bejarano, J. B. (2009). detección y clasificación de artefactos. *detección y clasificación de artefactos*, 2.
- Attias, H. (1999). Independent factor analysis. *Neural Computation*, vol. 11, 803-851.
- Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation* vol. 7, 1129-1159.
- Belouchrani, A., Abed-Meraim, K., & Cardoso, J. F. (1997). A blind source separation technique using second-order statistics. *IEEE Transaction on Signal*, 434-444.
- Berg, P., & Scherg, M. (1991). Dipole modeling of eye activity and its application to the removal of eye artefacts from the EEG and EMG. *Clin. Phys. Physiol. Meas*, vol. 12, 49-54.
- Brenton W. McMenamin, A. J. (2011). Electromyogenic Artifacts and Electroencephalographic Inferences. *NeuroImage* 54, 4-9.
- Cardoso, J. (1997). Infomax and maximum likelihood for blind source separation. *IEEE Signal Processing Letters* vol.4, 112-114.
- Choudrey, R., & Roberts, S. J. (2001). Flexible bayesian independent component analysis for blind source separation. En *Proceedings of ICA*. San Diego.
- Corral-Fernández, E. (2007). Nociones básicas de epilepsia en adultos en medicina interna. *Revisiones en medicina interna basadas en la experiencia*, 3.
- Eleni Kroupi, A. Y.-M. (2011). OCULAR ARTIFACT REMOVAL FROM EEG: A COMPARISON OF SUBSPACE PROJECTION AND ADAPTIVE FILTERING METHODS. *Proceedings of 19th European Signal Processing Conference*. Barcelona: Eusipc.
- Fisch, B. (1999). *Basic Principles of Digital and Analog EEG*. New Orleans: Elsevier.
- Fisch, B. J. (1999). *EEG Primer: Basic Principles of Digital and Analog EEG*. New Orleans: Elsevier.
- Fuchs-Buder, T. (2003). Intraoperative Monitoring: What is Necessary, Useful and What is Futile? *European Society of Anaesthesiologists*, 121-124.
- H, J. (1958). Report of committee on methods of clinical exam in EEG. *Electroencephalogr Clin*, 10:370-5.
- Haykin, S. (1996). *Adaptive filters theory*, 3 ed. Prentice Hall.
- Hyvärinen, A., & Oja, E. (1997). A fast fixed-point algorithm for independent component analysis. 1483-1492.
- I.I. Goncharova, D. M. (2003). EMG contamination of EEG: spectral and topographical characteristics. *Clinical Neurophysiology* 114, 1580–1593.
- IJ, R. (1994). Anesthetic potency is not altered after hypothermic spinal cord transection in rats. *Anesthesiology*, 606.

- Jones, J. A. (2012). Conceptos básicos de la anestesia. *Helen Devos children's Hospital*.
- Klass, D. W. (2008). *The Continuing Challenge o Artifacts in the EEG*. Rochester: Departament of Neurology, Mayo Clinic and Mayo Fundation.
- Kroupi, E., Yazdani, A., Vesin, J.-M., & Ebrahimi, T. (2006). *Ocular Artifact Removal from EEG: A Comparison of Subespace Projection and adaptative Filtering Methods*. Lausanne, Switzerland: École Polytechnique Fédérale de Lausanne.
- Mathews, S. e. (2006). EEG dipole analysis of motor-priming foreperiod activity reveals separate sources for motor and spatial attention components. *Clinical Neurophysiology*, 2675-2683.
- MathWorks. (s.f.). *MatLab*. Obtenido de <http://www.mathworks.com/products/matlab/>
- Milo, R. (25 de Abril de 2008). *Bionumbers*. Obtenido de BNID 100706: <http://bionumbers.hms.harvard.edu/bionumber.aspx?id=100706&ver=0>
- Navarro, R. B. (s.f.). *Instrumentacion biomedica, electroencefalografia*. Alcala.
- Nunez, P. L., & Srinivasan, R. (2006). *Electric Fields of the Brain: The Neurophysics of EEG*. New York: Oxford University Press.
- Olguín, D., Bouchereau, F., & Martínez, S. (2005). Adaptive Notch Filter for EEG Signals Based on the LMS Algorithm with Variable Step-Size Parameter. *Conference on Information Sciences and Systems*.
- Orozco, M., & Suarez, J. F. (2009). *Entrenamiento de Sistemas de Identificacion Automatoca de Patologias*. Centro de Publicaciones UTP.
- Proakis, J., & Manolakis, D. (2007). *Digital Signal Processing: Principles, Algorithma and Applications*. New Jersey: Prentice-Hall.
- Röpcke, H. (2004). *Depth of Anaesthesia - Concepts and Monitoring*. University of Bonn, Department of Anaesthesiology. Bonn, Germany: ESA.
- Shoker, S., & Latif, M. A. (2002). Removal of eye blinking artifacts from eeg incorporating a new constrained bss algorithm. *Conf. Proc. IEEE Eng. Med. Biol. Soc. vol 2*, 909-912.
- Sörnmo, L., & Laguna, P. (2005). *Bioelectrical Signal Processing in Cardiac and Neurological Applications*. Burlington: Elsevier Academic Press.
- Stein, J. Y. (2000). *Digital Signal Processing: A Computer Science Perspective*. Tel Aviv: Wiley.
- Vakkuri, A. (2006). *EEG Monitoring in Anaesthesia*. Helsinki, Finland: Helsinki University Hospital.
- Velde, M. v. (2000). *Signal Validation in Electroencephalography Research*. Eindhoven: Universidad técnica de Eindhoven.
- Wanchai. (2012). *10/20 System Positioning*. Hong Kong: Trans Cranial Technologies.
- Xinyi Yong, R. K. (2008). Facial EMG Contamination of EEG Signals: Characteristics and Effects of Spatial Filtering. *ISCCSP*, 12-14.
- Yarlagadda, R. K. (2010). *Analog and Digital Signals and Systems*. US: Springer.
- Zhu, W., Zeng, N., & Wang, N. (2010). Sensitivity, Specificity, Accuracy, Associated Confidence Interval and ROC Analysis with Practical SAS® Implementations. *Health Care and Life Sciences*, 1-9.

