

University of Memphis

University of Memphis Digital Commons

Electronic Theses and Dissertations

5-27-2015

Real-Time, Hardware Efficient Ocular Artifact Removal From Single Channel EEG data Using a Hybrid Algebraic and Wavelet Algorithm

Charvi Anand Majmudar

Follow this and additional works at: <https://digitalcommons.memphis.edu/etd>

Recommended Citation

Majmudar, Charvi Anand, "Real-Time, Hardware Efficient Ocular Artifact Removal From Single Channel EEG data Using a Hybrid Algebraic and Wavelet Algorithm" (2015). *Electronic Theses and Dissertations*. 1185.

<https://digitalcommons.memphis.edu/etd/1185>

This Thesis is brought to you for free and open access by University of Memphis Digital Commons. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of University of Memphis Digital Commons. For more information, please contact khhgerty@memphis.edu.

REAL-TIME, HARDWARE EFFICIENT OCULAR ARTIFACT REMOVAL FROM
SINGLE CHANNEL EEG DATA USING A HYBRID ALGEBRAIC AND WAVELET
ALGORITHM

by

Charvi Majmudar

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Electrical and Computer Engineering

The University of Memphis

August 2015

This thesis is dedicated to my husband Dr. Anand Majmudar, mother Mrs. Avani Nanavaty, father Dr. Ravindra Nanavaty, aunt Dr. Nivedita Vora and beloved grandmother Late Dolar Mankad for their extreme support, encouragement and ever flowing blessings.

ACKNOWLEDGEMENTS

I would like to take the opportunity to thank Dr. Bashir Morshed, for his constant support, motivation, suggestions and help during the thesis work and writing this thesis. I am extremely grateful to have him as my advisor throughout the course of study.

I am highly thankful to Dr. Eddie Jacobs and Dr. Amy Curry who devoted their precious time for being in my thesis committee and for giving their valuable suggestions.

I appreciate and feel thankful for the efforts, suggestions and support shown by my lab mates Ruhi Mahajan, Saleha Khatun and other members at the ESARP during my research work.

Last but not the least, I would like to thank my family and friends for their constant support and encouragement with complete patience to accomplish my research goal.

ABSTRACT

Majmudar, Charvi, A., M.S. The University of Memphis. August 2015. Thesis title: Real-Time, Hardware Efficient Ocular Artifact Removal from Single channel EEG data using a Hybrid Algebraic and Wavelet algorithm. Dr. Bashir I. Moshed.

Electroencephalography (EEG) is a promising technique to record brain activities in natural settings. EEG signal usually gets contaminated by Ocular Artifacts (OA), removal of which is critical for the feature extraction and classification. With the increasing interest in wearable technologies, single channel EEG systems are becoming more prevalent that often require real-time signal processing for immediate feedback. In this context, a new hybrid algorithm to detect OA and subsequently remove OA from single channel streaming EEG data is proposed here. The algorithm first detects the OA zones using Algebraic approach, and then removes artifact from the detected OA zones using Discrete Wavelet Transform (DWT) decomposition method. De-noising technique is applied only to the OA zone that minimizes interference to neural information outside of OA zone. The microcontroller hardware implemented hybrid OA removal algorithm demonstrated real-time execution with sufficient accuracy in both OA detection and removal. The performance evaluation was carried out qualitatively and quantitatively for 0.5 sec epoch in overlapping manner using time-frequency analysis, mean square coherence, Correlation Coefficient (CC) and Mutual Information statistics. Matlab implementation resulted in average CC of 0.3242 and average MI of 1.0042, while microcontroller implementation resulted in average CC of 0.4033 for all blinks. Successful implementation of OA removal from single channel real-time EEG data using the proposed algorithm shows promise for real-time feedback applications of wearable EEG devices.

TABLE OF CONTENTS

Chapter		Page
1	Introduction	1
	1.1 Motivation	1
	1.2 OA removal algorithm requirements	3
	1.3 Overview of the literature review	4
	1.4 Overview of the proposed OA removal algorithm	4
	1.5 Key Results	5
	1.6 Key Outcomes	6
	1.7 Organization of this Thesis	6
2	Literature Review	7
	2.1 Wavelet Based OA removal Approach	7
	2.2 Modified Multiscale Sample entropy (mMSE)-wICA	9
	2.3 EMD-CCA based OA removal approach	10
	2.4 Algebraic method to detect OA	10
3	The proposed hybrid algorithm	12
	3.1 Algebraic Method – OA Detection	12
	3.2 Wavelet Transform – OA removal	14
	3.2.1 Wavelet basis functions	16
	3.2.2 DWT implementation steps	16
	3.3 Hybrid OA Detection and Removal algorithm	17
4	Test Setup and Procedure	21
	4.1 System Components	22
	4.1.1 Software Utilized	22
	4.1.2 Hardware Utilized	23
	4.1.3 Communication interfaces	24
	4.2 EEG Datasets utilized	25
	4.2.1 Offline Mode	25
	4.2.2 Online (Real-Time) Mode	25
	4.3 EEG Data Acquisition	26
	4.3.1 EEG Device data acquisition	26
	4.3.2 User console data acquisition	27
	4.4 Methodology	29
	4.4.1 Offline Mode	29
	4.4.1.1 MATLAB based algorithm	29
	4.4.1.2 C based algorithm	36
	4.4.2 Real-Time Mode	39
	4.4.2.1 MATLAB based algorithm	39
	4.4.2.2 C based algorithm	42

5	Results and Performance Evaluation	52
	5.1 OA detection and removal Results	52
	5.1.1 Offline MATLAB based algorithm	52
	5.1.2 Online MATLAB based algorithm	55
	5.1.3 Online C based algorithm	57
	5.2 Performance Evaluation	60
	5.2.1 Performance Metrics	60
	5.2.2 Performance Evaluation Results	61
	5.2.2.1 Offline MATLAB based algorithm	61
	5.2.2.2 Online C based algorithm	66
6	Conclusion and Future scope	68
	6.1 Conclusion	68
	6.2 Future scope	71
	References	72
	Appendices	75
	A. Offline MATLAB based OA removal Code	75
	B. Online MATLAB based OA removal Code	80
	C. Offline C based OA removal Code	87
	D. Online C based OA removal Code	91

LIST OF FIGURES

Figure		Page
1	A representative raw EEG signal (FP1) vs a clean EEG signal	2
2	Block diagram of the proposed hybrid OA removal algorithm	4
3	Graphical presentation of sliding window method for OA detection algorithm.	12
4	Block diagram of DWT based n-levels decomposition	15
5	Frequency bands achieved at every DWT decomposition level	15
6	Wave-shapes of different wavelet functions	16
7	Proposed hybrid OA removal algorithm	18
8	Flow chart describing epoch overlap method	19
9	Detailed flow chart of hybrid OA removal algorithm	20
10	Overview of offline and online mode OA removal algorithm implementation	21
11	(a) Neuro Monitor device (b) Electrode placement used during EEG recording	23
12	Data acquisition: subject to NM device and NM device to user console	26
13	EEG data acquisition system of NM EEG device	27
14	User Console remote Data acquisition software Graphical User Interface panel	28
15	User console data acquisition system	28
16	Block diagram of MATLAB based algorithm implementation in offline mode	29
17	OA detection results for Emotive device (dataset-1)	31
18	OA detection results for NM device (dataset-4)	31
19	Undetected eye blinks by OA detection without overlapping method	32

20	Processing time bar plot DWT Vs SWT	33
21	OA removal results for Emotive device (subject-1)	35
22	OA removal results of NM device (channel-1(FP1) subject-2)	35
23	Block diagram of C based algorithm implementation in offline mode	36
24	Graphical presentation of DWT based decomposed signal vector	38
25	Hybrid algorithm results for NM device (C based – offline mode) (subject-1) (a) MATLAB result, (b) C based result	38
26	Hybrid algorithm results for NM device (C based – offline mode) (subject-2) (a) MATLAB result, (b) C based result	39
27	Block diagram of MATLAB based algorithm implementation in online mode	40
28	Online mode – MATLAB based hybrid Algorithm result (channel-1,subject-1)	41
29	Online mode – MATLAB based hybrid Algorithm result (channel-1,subject-2)	41
30	Block diagram of C based algorithm implementation in online mode	42
31	Graphical presentation of ADC ISR execution steps	43
32	Online mode C based <i>main</i> program flowchart implemented on PSoC-3 MCU	45- 46
33	Graphical presentation of real-time overlapping method implementation	46
34	(a) ‘Clean_EEG’ output buffer formation, (b) Header formation	47
35	Ring buffer output comparison on user console – online mode (C based)	48
36	(a) Online mode – C based hybrid Algorithm result (FP1 – dataset1) (b) Verification of OA zone detection C based algorithm in real-time (dataset1)	50 51
37	Plot showing minor mismatch in non-OA zone between raw and Clean EEG of real-time hardware implemented algorithm output	51
38	OA detection results for Emotive device (dataset-2)	53

39	OA detection results for Emotive device (dataset-3)	53
40	OA removal results of NM device (channel-1(FP1) subject-3)	54
41	OA removal results of NM device (channel-1(FP1) subject-4)	54
42	OA removal results of NM device (channel-2(FP2) subjects 2, 3 & 4)	55
43	Online mode – MATLAB based hybrid Algorithm result (channel-1, subject-3)	56
44	Online mode – MATLAB based hybrid Algorithm result (channel-1, subject-4)	56
45	(a) Online mode – C based hybrid Algorithm result (FP1 – dataset2)	58
	(b) Verification of OA zone detection C based algorithm in real-time (dataset2)	58
46	(a) Online mode – C based hybrid Algorithm result (FP1 – dataset3)	59
	(b) Verification of OA zone detection C based algorithm in real-time (dataset3)	59
47	Time-Frequency Analysis plot for subject-1 EEG data	61
48	Time-Frequency Analysis plot for subject-2 EEG data (a) Channel-1 (FP1) (b) Channel-2 (FP2)	62
49	Time-Frequency Analysis plot for subject-3 EEG data (a) Channel-1 (FP1) (b) Channel-2 (FP2)	62
50	Time-Frequency Analysis plot for subject-4 EEG data (a) Channel-1 (FP1) (b) Channel-2 (FP2)	63
51	MSC plot for raw Vs clean EEG (subject-1)	64
52	MSC plot for raw Vs clean EEG (subject-2) (a) Channel-1 (FP1) (b) Channel-2 (FP2)	64
53	MSC plot for raw Vs clean EEG (subject-3) (a) Channel-1 (FP1) (b) Channel-2 (FP2)	64
54	MSC plot for raw Vs clean EEG (subject-4) (a) Channel-1 (FP1) (b) Channel-2 (FP2)	65

LIST OF ABBREVIATIONS

Abbreviation	Full name
ADC	Analog to Digital Converter
ADHD	Attention Deficit Hyperactive Disorder
AM	Algebraic Method
ANC	Adaptive Noise Cancellation
ANN	Artificial Neural Network
BCI	Brain Computer Interface
BT	Bluetooth
CC	Correlation of Coefficient
CCA	Canonical Correlation Analysis
DWT	Discrete Wavelet Transform
ECG	Electrocardiography
EEG	Electroencephalography
EEMD	Ensembled Empirical Mode Decomposition
EMD	Empirical Mode Decomposition
EMG	Electromyography
EOG	Electrooculography
FFT	Fast Fourier Transform
FIFO	First In First Out
FIR	Finite Impulse Response
GPIO	General Purpose Input- Output
GUI	Graphic User Interface
HNN	Hardware Neural Network
ICA	Independent Component Analysis
ICs	Independent Components
IMF	Intrinsic Mode Functions
ISR	Interrupt Service subroutine
MCU	Microcontroller Unit
MI	Mutual Information

mMSE	Modified Multi-scale Sample Entropy
MSC	Magnitude Square Coherence
NM	Neuromonitor
NN	Neural Network
OA	Ocular Artifacts
PSD	Power Spectral Display
PSoC	Programmable System on Chip
RAM	Random Access Memory
RLS	Recursive least Square
SAR	Signal to Artifact Ratio
SURE	Stein's Unbiased Risk Estimate
SWT	Stationary Wavelet Transform
TFA	Time Frequency Analysis
UART	Universal Asynchronous Receiver Transmitter
WNN	Wavelet Neural Network
WT	Wavelet Transform

INTRODUCTION

1.1 Motivation

Electroencephalography (EEG) is the depiction of the neurological signals in terms of the electrical signals corresponding to the brain activities from the scalp surface using special metal electrodes and conductive media. EEG recording is a completely non-invasive procedure that can be applied repeatedly to the patients, normal adults, and children without virtually any counter effects, risk or limitation. The greatest advantage of EEG is speed [1]. Complex continuously varying patterns of neural activity can be recorded and displayed on the EEG machine screen as waveforms of varying frequency and amplitude measured in micro-voltage. Being non-invasive and painless procedure, the EEG signals are widely used to study the brain organization of cognitive processes such as perception, memory, attention, language, and emotion in normal adults and children. Also the results given by an EEG are commonly used to investigate information about certain disorders such as Seizures, Epilepsy, Alzheimer's diseases, ADHD (Attention-Deficit/Hyperactive Disorder) etc.

Electrical signals those detected along the scalp by an EEG, but originated from non-cerebral origin are called artifacts. The most common EEG artifact sources can be classified in mainly two types: (i) Physiological artifacts: such as due to any minor body movements, EMG (Electromyography), ECG (pulse, pace-maker), eye movements, sweating. (ii) Non-Physiological artifacts: such as due to 50/60 Hz line frequency, impedance fluctuation, cable movements, broken wire contacts, loose electrodes and low battery [1].

The non-physiological artifacts are usually at a separate spectral band and can be dealt with high order analog and digital filters. However, physiological artifacts are usually in the same spectral bands and very difficult to remove without loss of critical neuronal information during the occurrence of the artifact. This work focuses on removal of physiological artifacts arising from eye-blinks and movement of the eyeballs which are collectively known as Ocular Artifacts (OA), while preserving as much neuronal information as possible. Regular EEG signals in the order of microvolts are contaminated by these OA in the order of millivolts. The frequency range of interest for most of the EEG applications lies up to 100 Hz, and typical amplitude are $0.5\mu\text{V}$ to $100\mu\text{V}$ [1] whereas OA occurs within the range of 0 to 16 Hz having amplitude more than 10 times the regular cortical signals [2]. The Fig.1 shows a typical raw EEG data (FP1) with OA and the cleaned EEG data.

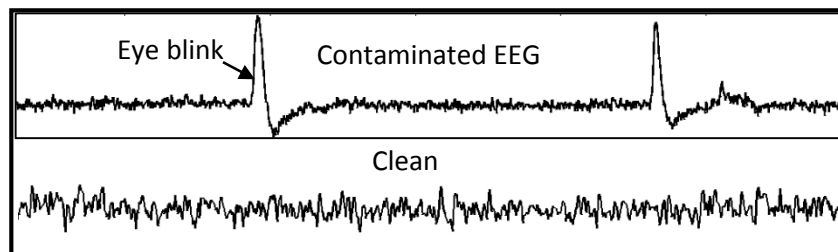


Fig.1: A representative raw EEG signal (FP1) vs a clean EEG signal

Eye blink artifacts can create inaccuracy or even can cause critical errors for feature extraction and classification effecting diagnosis process or automated brain-computer interfacing (BCI) applications. To improve the processing of EEG signals for accurate clinical and experimental analysis, removal of these artifacts are of prime interest.

In today's wearable technology and trend, ambulatory devices are of more interest and therefore EEG devices are also emerging with portability in nature. Such EEG ambulatory devices should be portable enough, light in weight and comfortable to carry on for the patients or subject under test. EEG electrode application for infants is difficult and challenging, due to the small head size and the limited space within a humidified incubator [3]. Also Alzheimer's disease recognition is often based on single channel EEG systems [4]. To accomplish these requirements and criticality of the EEG recordings, single channel EEG devices will be more relevant as well as convenient.

1.2 OA removal algorithm requirements

For remote monitoring of the subject using such ambulatory EEG devices, it is substantial to process EEG signals automatically in real-time for better diagnosis purpose. Moreover, to receive directly the clean EEG without eye blinks in real-time, the OA removal process should be running on the remote EEG device itself for the automation. Accordingly, the designed and developed OA removal algorithm mainly fulfilled basic three requirements:

i.e. OA removal algorithm should be:

1. Applicable to *Single Channel* EEG devices.
2. Executable in *Real-time*
3. Implementable on *Microcontroller Hardware*

1.3 Overview of the literature review

Literature on removal of the eye blink artifacts was critically reviewed to examine algorithm's single channel applicability with low computational complexity. The frequency spectrum of OA overlaps with the EEG frequency band, therefore filtering

techniques to remove OA directly, might also eliminate important neural information. Many techniques have been developed to eliminate OA from EEG signals. Some methods used Wavelet Transform (WT) [5,6,7], EMD (Empirical Mode Decomposition) - CCA (Canonical Correlation Analysis) [8,9] and Algebraic Method (AM) [4] based approaches, which are covered in Chapter-2.

1.4 Overview of the proposed OA removal algorithm

In this thesis we have developed a hybrid algorithm to detect and thereby remove OA from online EEG data. Two methods are combined:

- (i) *Algebraic method* – to detect the OA zone [4],
- (ii) *Discrete WT (DWT)* - to de-noise the detected OA zone [7] so as to obtain the artifact free EEG signal.

The proposed fully automated method neither needs recording of additional reference EOG signal nor relies on the other EEG channels. The algebraic method to detect OA zone has fast processing time, a key feature suitable for real-time applications. Whereas, WT method is suitable for non-stationary signals such as EEG and it also can process single channel data. The overall representation of this hybrid algorithm is as shown in Fig.2.

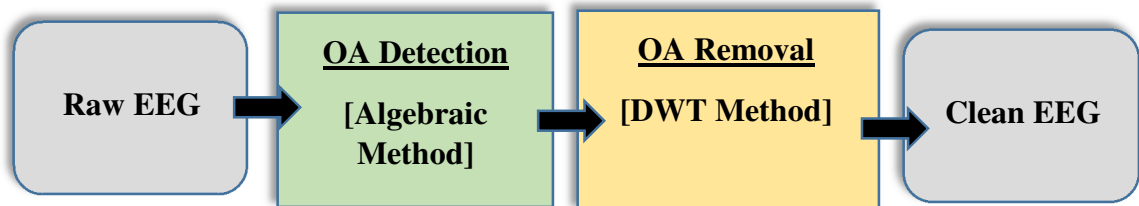


Fig.2: Block diagram of the proposed hybrid OA removal algorithm

In the first phase of offline mode, the hybrid OA removal algorithm was developed and tested using MATLAB (MathWorks Inc., Natwick, MA) software tool to verify the results and finalize the algorithm. In the second phase, algorithm was developed in C to run on PSoC-3 Microcontroller-Unit (MCU) hardware and results were cross checked with MATLAB results for verification. The NeuroMonitor (NM) EEG device and its single channel placed at FP1 or FP2 location (international 10-20 system) was used throughout the work for capturing EEG as well as hardware based algorithm implementation for the online mode.

The performance of the OA de-noising algorithm was statistically evaluated with time domain metrics - Correlation of Coefficient (CC) and Mutual Information (MI) and frequency domain metrics - Time-Frequency analysis (TFA) and Magnitude Square Coherence (MSC) estimation.

1.5 Key Results

The OA removal algorithm performed as expected in real-time settings on user console using MATLAB based OA removal algorithm. CC and MSC were evaluated for OA zone and non-OA zones separately where it was observed that for non-OA zone their values were '1', indicated that the EEG signal information remain unaltered whereas for OA zone, their values were less than '1', highlighted significant cleaning of eye blink in those zones. Also on MCU of EEG device using C based OA removal algorithm achieved proper OA de-noising from real-time single channel EEG recordings. The hybrid OA removal algorithm implemented on micro-controller utilized 28.1% of flash memory and executed in near real-time which further can be optimized. Also it resulted in average CC of 0.4033 for all blinks indicated significant removal of artifacts from detected OA zones.

1.6 Key outcomes

- Conference paper accepted: C. Majmudar, R. Mahajan, and B. I. Morshed, “Real-Time Hybrid Ocular Artifact Detection and Removal for Single Channel EEG”, IEEE Electro/Information Technology (EIT), (accepted), 2015.
- Conference paper abstract accepted: Charvi Majmudar and B. I. Morshed, “*Hardware implementation of real-time hybrid OA detection and removal algorithm for single channel EEG signals*”, *IEEE GHTC conf.*, (Abstract accepted) Oct. 2015.

1.7 Organization of this thesis

This thesis contains total of six chapters with separate Reference and Appendix sections. Chapter 2 is furnished to take a glance on past work done related to OA removal techniques from EEG. In chapter 3, theory background of OA detection and removal methods are covered up. It ends with giving the overview of basic steps followed in proposed hybrid OA removal algorithm with required flowcharts. The core contents of the thesis resides in chapter 4. This chapter takes upon in detail how the thesis work has been divided in two major part: offline mode and online mode having MATLAB and C based algorithms with theirs thorough explanation and respective results. Chapter 5 is included to present all the remaining results for every mode of operation of the thesis work which have not been included in chapter 4. The Chapter 5 also depicts the performance analysis for the results achieved in offline MATLAB and online C based algorithms. Finally, the chapter 6 concludes the entire work with the future scope of the research topic considered.

LITERATURE REVIEW

In this section the summary of the reviewed literatures for the relevant prior work done in the same EEG ocular artifact removal area is presented.

2.1 Wavelet Based OA removal Approach

WT as a novel approach was introduced in [5] in the year 2004 to remove eye blink from EEG. Here, Stationary Wavelet Transform (SWT) with Haar wavelet of high orders was applied to decompose the original contaminated signal. De-noising was done by hard - thresholding and finally wavelet inverse transform was applied to reconstruct the EEG signal. The author of this paper concluded that wavelet based technique removed eye blink from EEG successfully. Thus, this paper gave good initial foundation for WT techniques to remove OA from EEG effectively applicable to single channel.

Adaptive Algorithm using WT to de-noise the EEG data, sampled at 128 samples/sec, was carried out in [6]. The significance of the proposed method was that it didn't affect the low frequency components in non-OA zones and preserved the information of the EEG signal by applying the de-noising technique only to the OA zone rather than to the entire EEG signal. The Algorithm proposed in this paper [6], detected the OA zone using Discrete Wavelet Transform (DWT) with Haar wavelet as the basis function. Next, SWT was applied only to the detected OA zone and Adaptive algorithm based on Stein's Unbiased Risk Estimation (SURE) and the soft-like thresholding function were used to de-noise EEG. The performance metrics such as Power Spectral Density (PSD) and correlation in frequency domain evaluated here, showed that by de-noising the EEG only in eye

blink zone, improved the performance by preserving low frequency components in the non-OA zone.

The proposed technique in [7] was motivated to reduce the computational complexity of the adaptive algorithm of [6] and used Statistical Method to de-noise the EEG. The algorithm proposed here applied SWT to the contaminated EEG signal with ‘Sym3’ as a basis function with 8-level of decomposition for the data sampled at 128 samples/sec. The OA zone was identified using a statistical approach - coefficient of variation. The de-noising technique was then applied using fixed suitable threshold value and threshold function from the detected artifact zones. On examining the approach it was found that only for threshold calculation detected OA zone was utilized otherwise de-noising was applied to the 10 sec of EEG epoch, which resulted in distortion of EEG signal in non-OA zone.

In [10], a wavelet neural network (WNN) based algorithm for EEG artifact removal was discussed. The algorithm combined the universal approximation characteristics of neural networks and the time/frequency property of wavelet transform, where the Neural Network (NN) was trained on a simulated dataset with known ground truths. Coefficients at low frequency sub-bands: 0–2, 2–4, 4–8 and 8–16 Hz after WT using ‘coif3’ as basis function up to 6 decomposition level, were passed through an Artificial NN (ANN) with the structure of 4-6-4 (4-input units, 6-hidden units and 4-output units) for training purposes. The performance of WNN method was found better when compared with WT using adaptive algorithm. Also the computational cost point of view this method outperformed ICA (Independent Component Analysis) but no evidence to show faster processing time than WT based technique. Additionally, if wavelet decomposition level is

higher in value then NN design would become complex and hardware components needed would be higher. Also specialized hardware called Hardware Neural Networks (HNN) [11] are required to obtain maximum advantage of ANN implementation on hardware but HNN then would be an additional requirement for the existing system.

In another method [12], DWT decomposition up to 7 levels with ‘db4’ basis function and thresholding based on SURE with soft thresholding technique was applied to the single channel EEG to construct the reference OA signal. Next, Adaptive Noise Cancellation (ANC) technique based on Recursive Least Square (RLS) algorithm was applied to remove OA from contaminated EEG. Results of this paper revealed that when compared to SWT with thresholding technique, this proposed method outperformed that approach. But the processing time plot as compared with ICA showed that for higher samples under processing, time taken by this algorithm drastically increases.

2.2 Modified Multiscale Sample entropy (mMSE)-wICA

This paper [13] presented an unsupervised, fast algorithm for fully automatic identification and suppression of the eye blink related ICs by using mMSE and Kurtosis as markers and wavelet decomposition as de-noising tool. The mMSE efficiently identified the ICs with eye blink characteristics by fetching the regularity information from the ICs over multiple temporal scales. Kurtosis was used to enhance the performance by identifying the ICs with super-Gaussian ‘peaked’ probability distributions, which imitates the eye blink distributions. The blink related artifactual ICs identified were then denoised using DWT with the Biorthogonal wavelet function. But the method described required more number of EEG channels which contradicted the single channel requirements of the thesis work.

2.3 EMD-CCA based OA removal approach

Another single channel EEG OA removal technique described in [8] used EMD-CCA based approach. Here, OA template was determined from contaminated EEG using EMD (Empirical Mode Decomposition), which decomposes a time series signal into multiple “intrinsic mode functions” (IMFs) and then CCA (Canonical Correlation Analysis) was applied on the OA template and contaminated signal to get the clean EEG. The article compared results based on Correlation Coefficient and SAR (signal to artifact ratio), which concluded that this proposed method was found better than ICA, CCA, EMD-ICA. There was no comparison given with WT based technique but was provided in article [9], which was the modification of EMD-CCA. In [9], EEMD (Ensemble EMD) improved the performance by eliminating the mode mixing dilemma existing with [8]. The computational cost comparison given in article [9] revealed that WT based method was better or equal in performance as compared with EEMD-CCA method. Also [14] concluded that WT and EMD both have their own advantages and limitations using different metrics such as time consumption, SAR and PSD. Additionally, CCA method has considerable amount of spectral error and thus it cannot be implemented in real-time [15] which limits its use for the proposed OA removal algorithm considering the basic three requirements.

2.4 Algebraic method to detect OA

Only the Detection of OA using Algebraic method was presented in [4]. Proposed method used Operational Calculus leading to joint detection and change point detection, where the signal was represented with a piecewise polynomial model in interval $[0, T]$. Using algorithms defined in [16] and [17] second order equation derived as:

$$a_k * (t_k)^2 + b_k * t_k + c_k = 0; \quad (1)$$

Where, each term was the output of Finite Impulse Response (FIR) filter and if there existed discontinuity (eye blink) in the interval $[0, T]$, these coefficients would be non-zero else they all would be zero. Thus, in the proposed method ' $Q_k = |a_k * b_k * c_k|$ ' was calculated and if this value exceeded the given threshold, spike existed at time ' t_k ' otherwise not. Proposed method only detected the OA but it showed that accuracy was nearly same as WT, whereas computational cost was significantly less when compared with WT based OA detection. Article showed that for 7.5min long input EEG signal, WT took 27.31sec whereas algebraic method took 3.22sec.

We have implemented the hybrid approach combining OA detection using Algebraic method [4] for its fast processing time benefit and OA removal using WT based de-noising technique applied only to the detected OA zone referring [7], as WT is commonly used robust method for single channel and hardware implementation capability. Each of the methods and final hybrid algorithm is described in next chapter.

THE PROPOSED HYBRID ALGORITHM

This section describes each method used for the proposed hybrid algorithm in detail along with its required mathematical background. The section starts with Algebraic Method, followed by Wavelet Transform Method and finally represents overview of the implemented overall hybrid algorithm for OA removal from single channel EEG.

3.1 Algebraic Method – OA Detection

Due to the eye blink during regular EEG signal recording, spike like artifact is generated, which is considered as irregularity in the neuronal signal. This method [4] detects the abrupt changes and estimate their locations for the given noisy observation $y(t)$, of a piecewise regular signal $x(t)$. It considers that there exists at most one spike in each interval $'I'$, $[\tau, \tau + T]$, where, τ is the origin and T is the length of $'I'$. In this interval, FIR filter of the order $'M'$ is applied to extract FIR filter coefficients using sliding window technique, which is repeated $'n'$ times, where $'n'$ represents the number of sliding windows of interval $'T'$ for the considered EEG epoch length, refer Fig.3.

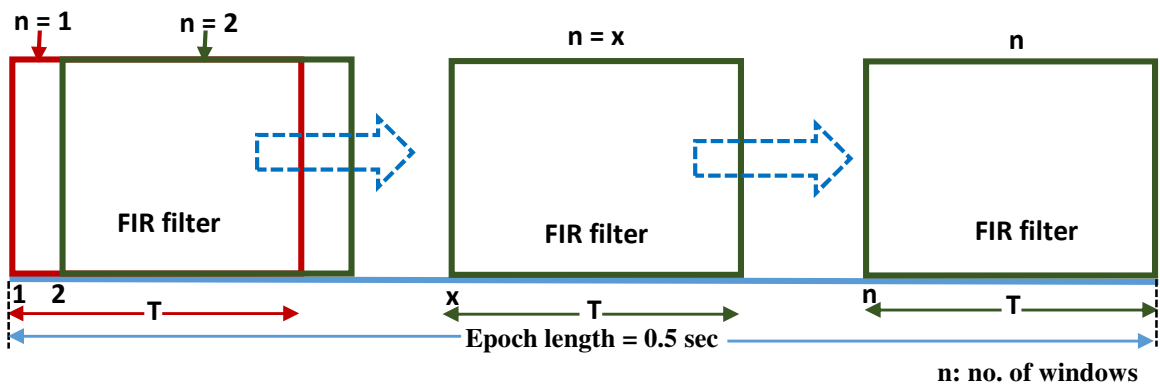


Fig.3: Graphical presentation of sliding window method for OA detection algorithm

The impulse response equation used for implementing FIR filter is shown in (2) which has been derived using operational calculus method to find the change point detection described in [16].

$$h_k(t_m) = \frac{(-1)^{k+1}}{(v-1)!} \frac{d^2}{dt_m^2} (1 + t_m)^{k+2} t_m^{v-1} \quad ; 0 \leq t_m \leq T \quad (2)$$

$$= 0 \quad ; \text{otherwise}$$

Using this Impulse response, discrete FIR filter coefficients are calculated using convolution method as (3).

$$v_k(\tau) \approx v_{k,n} = \sum_{m=0}^M h_{k,m} y_{n+M-m} \quad (3)$$

Where, $v_{k,n}$ = FIR filter coefficients for 'n' sliding windows for four values of 'k' (0, 1, 2, 3). ' v_k ' value represents the each term in (1), such that with 'k' values as 0, 1 and 2 for a_k , b_k and c_k respectively. Similarly, FIR coefficients are also calculated for the values of 'k' as 1, 2 and 3 for a_k , b_k and c_k respectively which then can be represented as the matrix form representing two different quadratic equations as shown in (4).

$$\begin{bmatrix} v_k & v_{k+1} \\ v_{k+1} & v_{k+2} \end{bmatrix} \begin{bmatrix} t_k^2 \\ t_k \end{bmatrix} = - \begin{bmatrix} v_{k+2} \\ v_{k+3} \end{bmatrix} \quad (4)$$

Next, the Decision function, ' $F_{k,n}$ ' is calculated using (4) as described by (5) for every sliding windows 'n'.

$$F_{k,n} = [v_{k+1,n}]^2 - v_{k,n}v_{k+2,n} \quad (5)$$

This decision function for spike detection which corresponds to Volterra filtering of the neural signal reducing the noise and highlighting spikes [18]. Final Decision function ‘ F_n ’ is derived as (6) where in our case ‘ K ’ has been taken as 2 as only two different quadratic equations were considered here as described by (4).

$$F_n = \prod_{k=0}^{K-1} F_{k,n} \quad ; n = 0,1,2 \dots \quad (6)$$

This decision function (6) is then compared with the threshold value to decide whether the eye blink (spike) exists in the given considered period or not. For the same, the threshold equation used here was:

$$\gamma = \frac{N}{\mu + \sigma} \quad (7)$$

where, N = Constant; μ = mean; σ = Standard Deviation.

The threshold equation (7) was referenced from [7] and ‘ N ’ was fixed to 0.001 so as to detect only the OA in the given EEG epoch. ‘ N ’ was fixed after number of trials using different EEG data sets and found consistent enough to detect eye blink only in EEG signals ignoring possible artifacts due to some other muscle movements or loose electrodes etc. Once this threshold value is calculated, ‘ F_n ’ is compared with it to make the decision for the existence of the OA at the time ‘ t_k ’. Due to the presence of the noise in the actual signal, ‘ F_n ’ is compared with the threshold instead of zero. Each eye blink would have two spikes detected at two different ‘ t_k ’ corresponding to starting edge and ending edge of the eye blink. Finally based on detected edges, the sample numbers for OA-Zones in each epoch were stored for further de-noising purpose.

3.2 Wavelet Transform – OA removal

WT has emerged as one of the robust methods in processing non-stationary signals such as EEG. The advantage of Wavelet Transform over Fourier Transform is that the windows in WT vary. The Discrete WT (DWT) decomposition for denoising OA from EEG is implemented here. The result obtained at each decomposition level is composed by two types of coefficients: Approximation coefficients and Detail coefficients as shown in Fig.4. The original signal is convolved with a low and high pass filter whose impulse response is determined by the wavelet chosen [19]. The approximation coefficients are obtained by low-pass filtering the input sequence, followed by down-sampling. The detail coefficients are obtained by high-pass filtering the input sequence followed by down-sampling. The sequence of approximation coefficients constitutes the input for the next iteration. Different possible scaling with DWT is as shown in Fig.5 at every level.

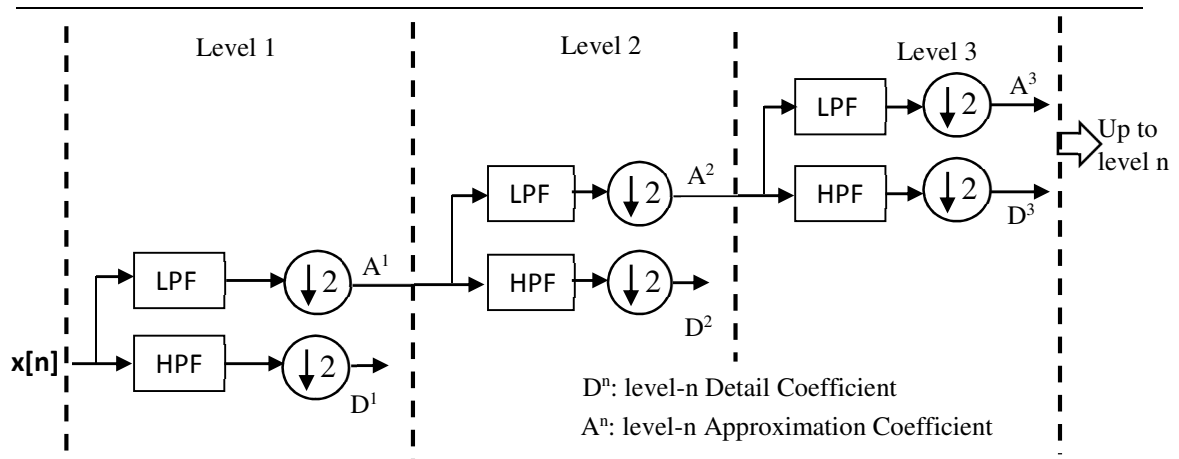


Fig.4: Block diagram of DWT based n-levels decomposition

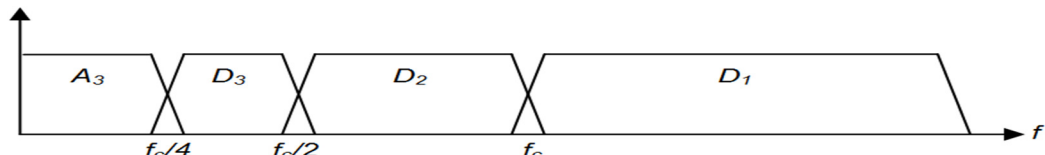


Fig.5: Frequency bands achieved at every DWT decomposition level [20]

The Discrete Wavelet Transform has two features: the wavelet mother ψ and the number of decomposition levels. Discrete wavelets can be scaled and translated in discrete steps having general wavelet representation as following: [21]

$$\psi_{j,n}(t) = \frac{1}{\sqrt{2^j}} \psi\left(\frac{t-2^j n}{2^j}\right) \quad (8)$$

where, j is the scale factor and n is the translation index.

The DWT is invertible, meaning after decomposing the original signal up to desired levels, decomposed signal can be composed back using inverse DWT method to get back original signal. It is the reverse process of decomposition where starting from highest level of decomposition, the coefficients are first up-sampled by the factor of 2 and then passed from low-pass and high-pass filters and merged to form the approximation coefficients for the next lower level.

3.2.1 Wavelet basis functions

The eye blink shapes vary for every subject and thus different wavelet functions (mother wavelets - refer Fig.6) were tested for their suitability during the developmental phase of the de-noising algorithm. For the hardware implementation, 'haar' wavelet was used with DWT based de-noising as a first stepping stone for the algorithm verification.

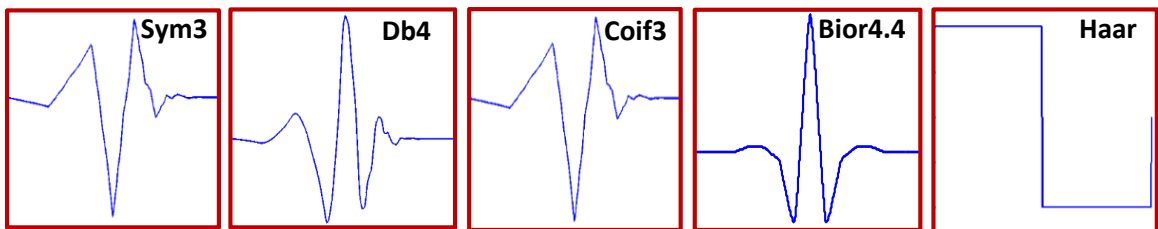


Fig.6: Wave-shapes of different wavelet functions

3.2.2 DWT implementation steps

The proposed DWT based OA removal algorithm involved the following steps:

1. Apply DWT only to the detected OA zone to decompose it up to eight levels using selected basis function.
2. The detail coefficients of decomposition levels 4 to 8 are compared with the threshold equation (7). If the coefficient value exceeds the threshold value, it is replaced with zero else retains its value as it is.
3. Finally, the inverse DWT was applied to reconstruct the clean EEG signal from the decomposed signal.

Two types of WT decomposition methods, SWT and DWT were compared during developmental phase for their respective performances to determine the final approach using different mother wavelets such as 'coif3', 'sym3', 'db4', 'haar' and their other variants. Due to the discrepancy of up-sampling and down-sampling at the every decomposition level in SWT and DWT respectively, DWT was preferred over SWT for its faster processing time and equivalent accurate de-noising ability, considering real-time implementation aspect of the algorithm. Moreover, to preserve the low frequency components in the non-OA zone, **DWT was applied only to the detected OA zone** (identified by algebraic method) and non-OA zone remained intact ensuring the critical EEG background information persisted in this region.

3.3 Hybrid OA Detection and Removal algorithm

Overall algorithm proposed in this thesis work implemented **Algebraic approach** to detect the OA zone and **DWT based de-noising** technique applied only to the detected OA zone for eye blink removal, refer Fig.7. Algorithm was found to be suitable for real-

time implementation for single channel EEG signal OA removal on MATLAB software platform as well as executable on actual microcontroller hardware to perform as expected preserving the regular EEG information in the non-OA zones.

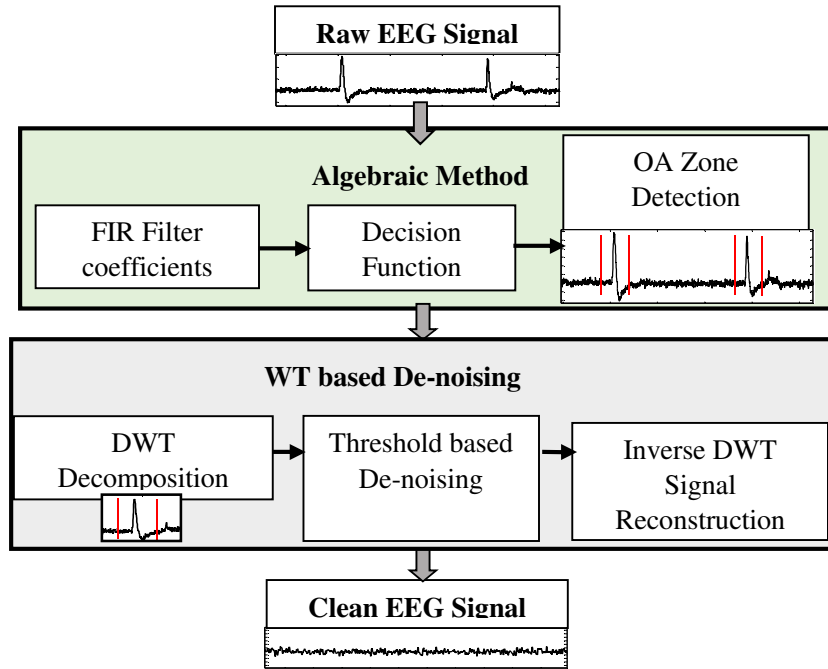


Fig.7: Proposed hybrid OA removal algorithm

- Some key considerations for implementing hybrid algorithm were as described:
 - a. For the algebraic method, to ensure that there exists at the most one eye blink (irregular point) in ‘ T ’, the length of the interval ‘ T ’ had been taken as 0.29 sec, which gave sufficient resolution as eye blinks are typically longer than 0.3 sec. (the average eye blink duration is generally 0.2 to 0.4 sec [2,22]).
 - b. The FIR filter delay which is generally half of the filter order [23] which was adjusted while calculating the exact OA zone locations.
 - c. For online EEG data testing, the EEG epoch length of 128 samples (0.5sec data @ 256 sps) was considered in the real-time to execute the entire de-noising algorithm.

- d. Due to the filter delay, there are chances that the eye blinks occurring at the end of the epoch, might get undetected by the algorithm. It has been avoided by overlapping the epochs with the ratio of ~31% as illustrated in Fig.8, to ensure the correct OA zone detection by testing over several datasets.

The implemented EEG epoch overlapping method for the hybrid OA removal algorithm is as depicted in Fig.8, where out of the total epoch length of 128 samples, last 40 samples were reconsidered for the next epoch processing to ensure the correct OA detection. The overall hybrid OA removal algorithm is illustrated as flowchart in Fig.9.

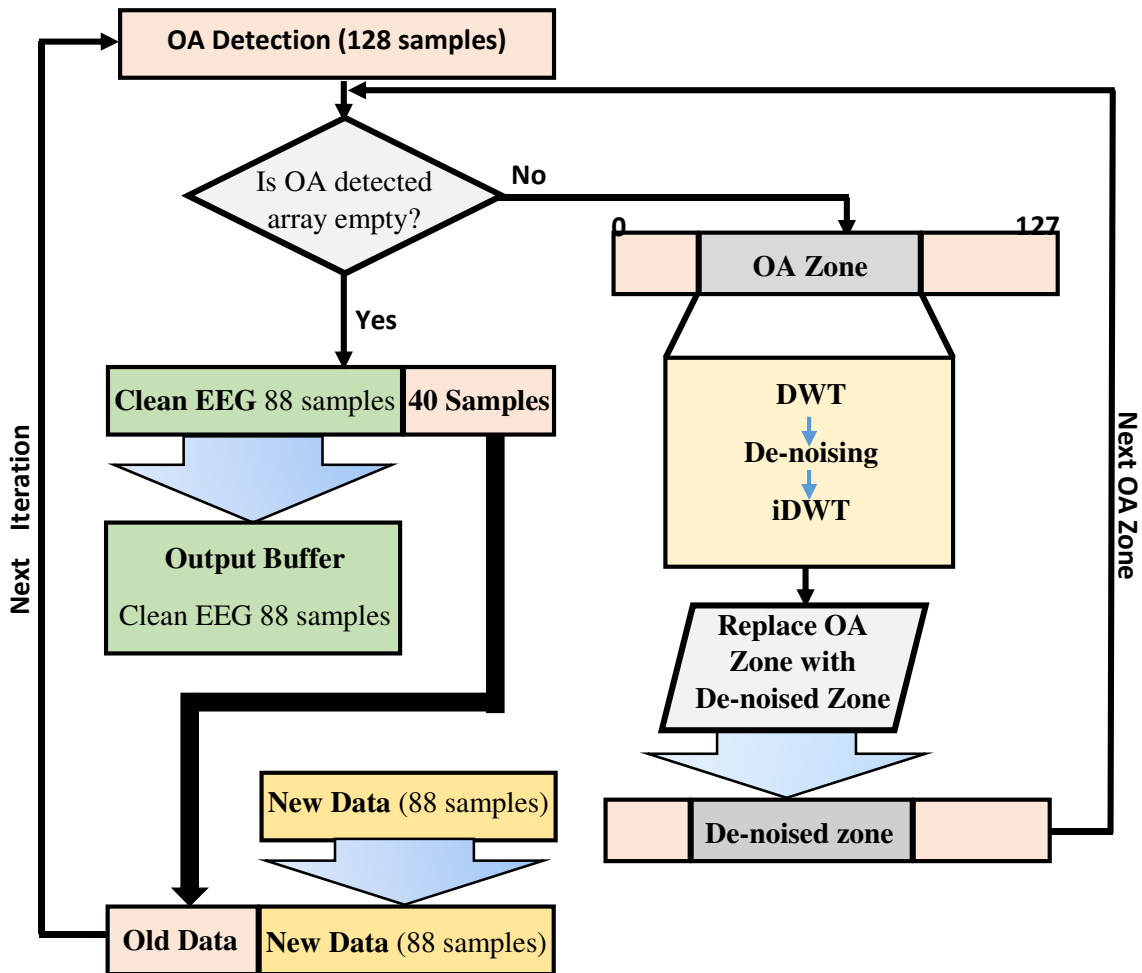


Fig.8: Flow chart describing epoch overlap method

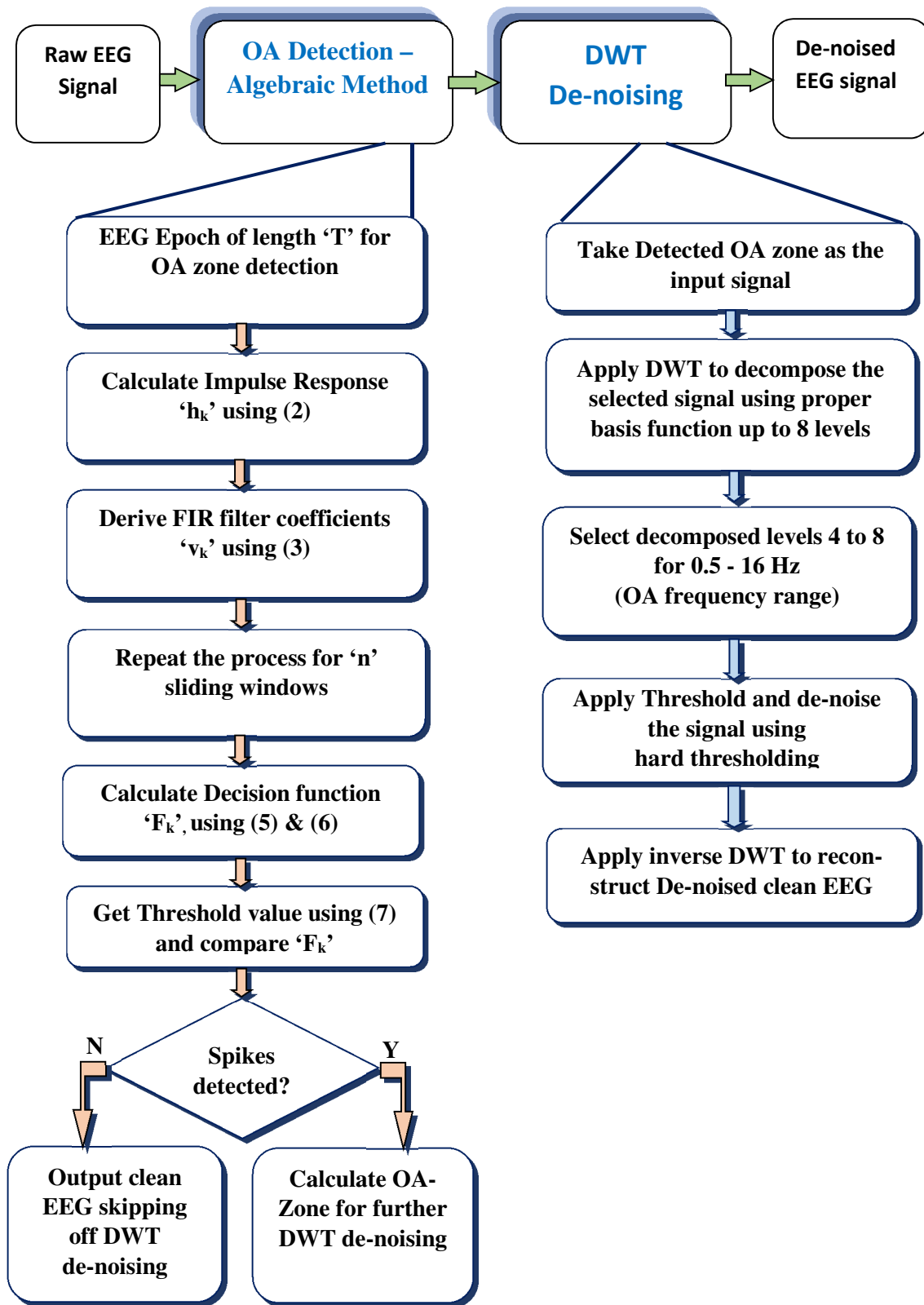
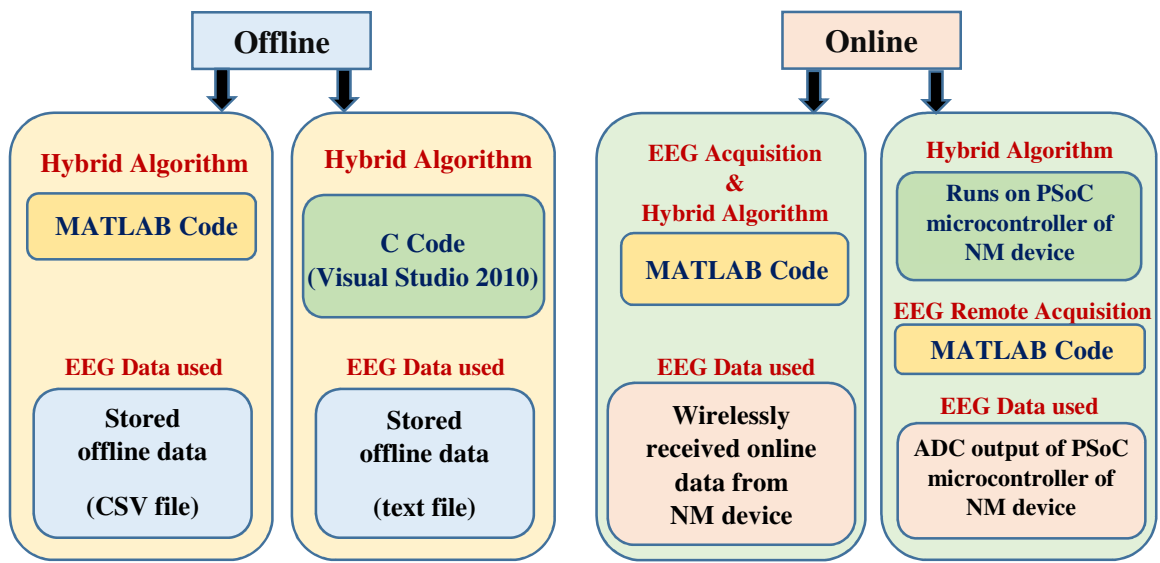


Fig.9: Detailed flow chart of hybrid OA removal algorithm

TEST SETUP AND PROCEDURE

Testing of the OA detection and removal algorithm was divided in two parts:

I.) Offline mode, II.) Online (real-time) mode. Offline mode was carried out to verify and finalize the hybrid algorithm and then the same algorithm was checked for its online execution in Real-Time mode. Overview of the entire process carried out is as illustrated in Fig.10.



Hybrid Algorithm: (OA Detection + OA Removal)

Fig.10: Overview of offline and online mode OA removal algorithm implementation

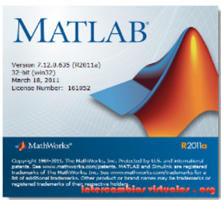



This section describes the system components such as software, hardware and communication interfaces used throughout the experiment, datasets and data acquisition details and finally procedure followed in offline – online modes in detail with respective results achieved.

4.1 System Components

All testing were carried out on computer having Intel(R) Core i5-3337U CPU @ 1.80 GHz, with 4.0 GB RAM and 64-bit Operating system of Windows 7/8.1. Other system components utilized during the work are listed below.

4.1.1 Software Utilized

Table.1: LIST OF SOFTWARE TOOLS USED WITH THEIR UTILITY IN THESIS WORK

Software name	Symbol / Logo	Version	Utility in thesis work
MATLAB (MathWorks Inc., Natwick, MA)		2011a	<ul style="list-style-type: none"> Algorithm testing in offline mode Raw/clean EEG remote acquisition code Algorithm Performance Evaluation and result verification
EEGLAB free software from SCCN (ucsd) [24]		EEGLAB v13	<ul style="list-style-type: none"> For Time-Frequency analysis as performance evaluation
Microsoft Visual Studio		Microsoft Visual Studio C++ 2010	<ul style="list-style-type: none"> Verification of C code of OA detection and removal algorithm in offline mode
PSoC Creator		PSoC Creator 3.0	<ul style="list-style-type: none"> Raw EEG digitization and transmission via Bluetooth in C language Implementation of proposed algorithm on PSoC microcontroller hardware for online mode in C language

4.1.2 Hardware Utilized

I. NeuroMonitor:

NeuroMonitor (NM) is an ambulatory EEG device used to capture raw EEG signal and transmit it wirelessly to the remote device [25]. It is a miniature, lightweight, two-channel referential montage based EEG device that is practically deployable in real-life settings using PSoC-3 MCU and can wirelessly transmit data using Bluetooth at the baud rate of 115.2 kbps in online mode, while being concealed within head accessories like a cap/headband having sampling rate of 256 sps as shown in Fig.11 (a).

- **Electrodes utilized:**

The commercial disposable adhesive pre-gelled electrodes (GS-26, Bio-Medical Instruments, Warren, MI) suitable for EEG data collection from the prefrontal cortex are used in NM device. The sensor contains a 0.5 percent saline base gel on a 10 mm flat pellet Ag/AgCl electrode surrounded by a paper-thin transparent self-adhesive tape disc of 1-inch diameter [26]. Electrodes position FP1 and FP2 on human scalp (based on 10-20 International electrode system) were used throughout the work as shown in Fig.11 (b).

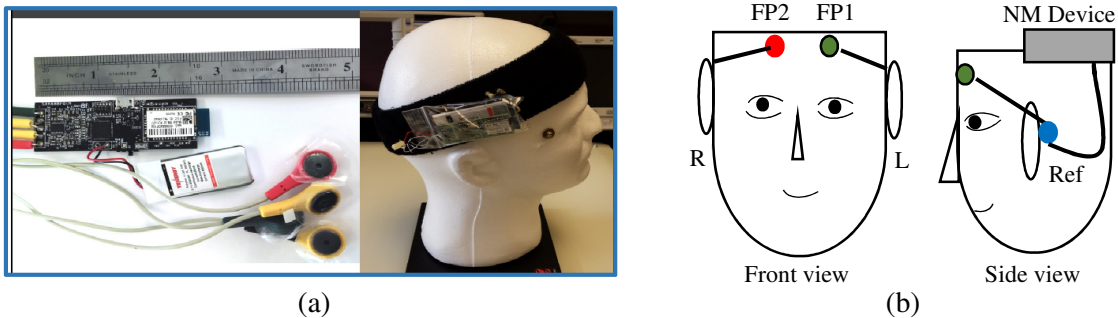


Fig.11:(a) NeuroMonitor device [26] (b) Electrode placement used during EEG recording

- **NM EEG device was used for the following purposes:**

- i. To Capture EEG signals from human scalp using electrodes attached

- ii. To Digitize and process captured EEG using on-board Microcontroller
- iii. To implement proposed hybrid algorithm on on-chip MCU in online mode
- iv. To transmit raw and/or clean EEG signal wirelessly using Bluetooth device

II. PSoC-3 Microcontroller

- **Basic Technical Features [25]**

Some of the useful technical details about PSoC-3 MCU are listed here.

PSoC-3 CY8C38 family	: 8-bit 8051 CPU
RAM	: 8 KB
Flash memory	: 64KB
Clock	: 3-62 MHz internal Oscillator
On-chip ADC	: 16-bit
On-Chip UART	: -

- **PSoC-3 MCU Usage:**

PSoC-3 MCU which is on-board of NM device was basically deployed to capture EEG using electrodes, filter and amplify the analog EEG signal, digitize the same using on-chip 16-bit ADC on interrupt basis at 256 sps and finally to output digitized EEG wirelessly using on-board Bluetooth device, refer Fig.13.

4.1.3 Communication interfaces

I. Bluetooth transceiver (RN-42 (Roving Networks)):

- **Specifications:**

It's a Class 2 Bluetooth (BT) module with inbuilt antenna. Running in the 2.4 GHz ISM band, this BT device can cover range up to 20 meters [27]. The baud rate for transmission was set to be 115,200 bps.

- **BT Usage:**

BT transceiver was used on NM device for initial synchronization and wireless data transfer in online mode.

4.2 EEG Datasets utilized

4.2.1 Offline Mode

- MATLAB based algorithm: The datasets used were as mentioned in Table.2 and they were read from respective ‘CSV’ file for the further processing.

Table.2: DATASET LIST FOR OFFLINE MATLAB BASED ALGORITHM

	Device Name	No. of Datasets	Sampling rate Samples/sec
OA Detection	Emotive	3 subjects (single channel)	128
	NueroMonitor	1 subject (single channel)	256
OA Removal	Emotive	1 subject (single channel)	128
	NueroMonitor	3 subjects (single channel)	256

- C based algorithm: Total 2 stored datasets of two different subjects from only NM device were used to test entire hybrid algorithm, which were read as respective ‘text’ file for further processing.

4.2.2 Online (Real-Time) Mode

- MATLAB based algorithm: The MATLAB written OA removal code was tested in real-time mode using online data received from NM EEG device by existing MATLAB acquisition software [25]. The EEG was recorded from single-channel FP1 (channel-1) (based on 10-20 International electrode system) location of total 4 subjects by following mentioned protocol: During the EEG recording process, subject was asked to blink at every 5 second of interval for 30 seconds. Thus, for each subject, data from each channel, contained 30 seconds of EEG recording comprised of 5 eye blinks.

- Hardware implemented C code: The C based hybrid algorithm implementation on hardware was verified using total three different datasets belonging to single subject from single-channel (FP1) using NM EEG device.

4.3 EEG Data Acquisition

Data acquisition was required on two sides: one on NM device from subject and the other on user console monitor from NM device as shown in Fig.12. Both data acquisition methods are described in the following sections.

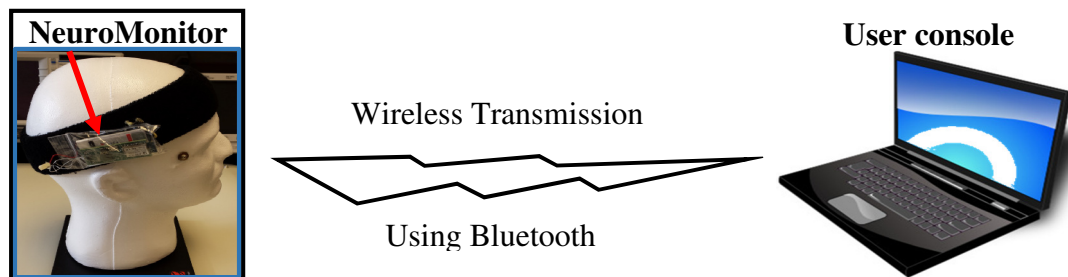


Fig.12: Data acquisition: subject to NM device and NM device to user console

4.3.1 EEG Device data acquisition

Captured EEG from scalp using electrodes attached to NM device, passes from several analog signal conditioning stages using instrumentation amplifiers, filtering and gain amplifiers successively. The amplified and filtered signal is required to be digitized before being wirelessly transmitted which is achieved via a 16-bit analog-to-digital (ADC) converter on-chip to the PSoC-3 at the sampling frequency of 256 sps. The digitized signal gets stored in real-time with two FIFO (first in first out) buffers acquiring data in tandem, while data can be sent wirelessly. Each buffer size considered was 512 bytes requiring 0.5 seconds to fill up holding the sampled data every 3.9 ms for the two channels [27]. Interrupts were triggered when a buffer was filled and the buffered data was

then wirelessly transmitted using Bluetooth in online mode, while the other buffer was being filled up. The complete block diagram of the data acquisition system of NM EEG device is given in Fig.13.

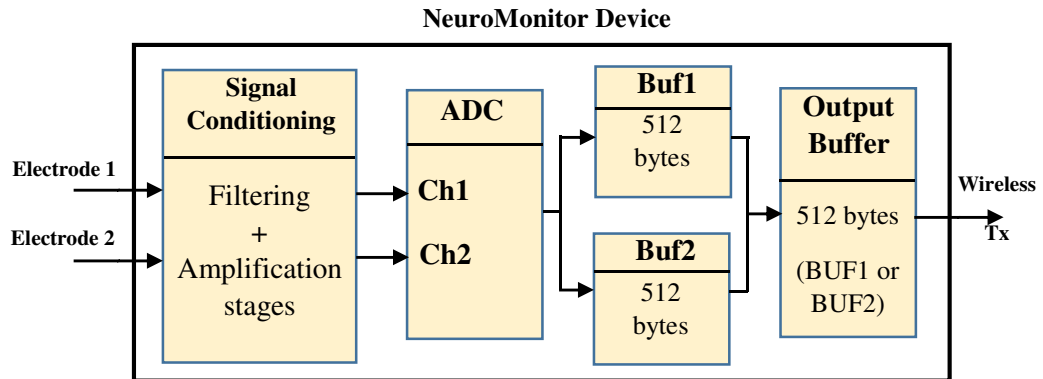


Fig.13: EEG data acquisition system of NM EEG device

4.3.2 User console data acquisition

The EEG raw data sent by NM device wirelessly using Bluetooth was captured remotely on user console where MATLAB software based data acquisition program was executed. Before sending the 512 bytes of sampled ADC data wirelessly as described in previous section, 22 bytes of header was appended to the buffer (making total 534 bytes of buffer size to be transmitted). All data packets were marked with headers of the packet start reference and packet count to ensure the reliable transmission from NM device. At the receiver side in MATLAB, wireless data was captured serially and header start reference and packet count were checked for the reliability of the received data. In case of any data loss, the user was notified by the Missed Packets count in Graphical User Interface (GUI) panel of data acquisition software, refer Fig.14. On receiving the data packet successfully, acquisition software separates channel-1 and channel-2 from received 512 bytes of buffer discarding the header. The original sample was 16-bit long which was

sent as 2 bytes remotely and thus, for each channel, every sample was reformed to represent the 16-bit data. Next, the digitized and the amplified sample was scaled down to original microvolt level by respective scaling function. Finally, processed and converted actual raw data was stored as 'CSV' file for further analysis. Entire data acquisition software process on remote user console side is depicted as a block diagram in Fig.15.

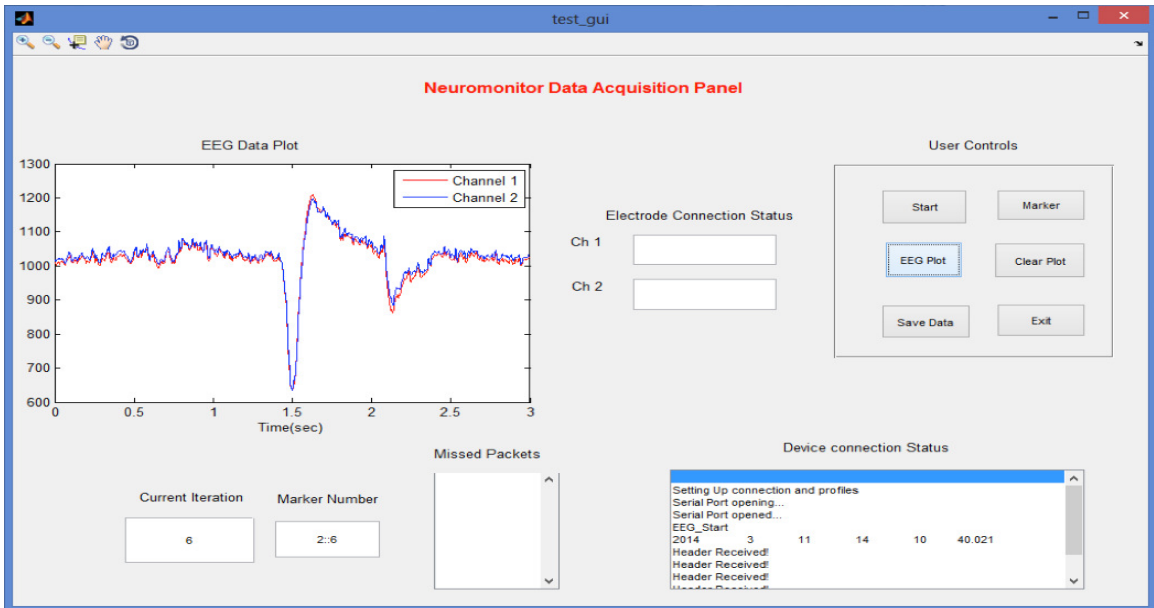


Fig.14: User Console remote Data acquisition software Graphical User Interface panel

User console Data Acquisition

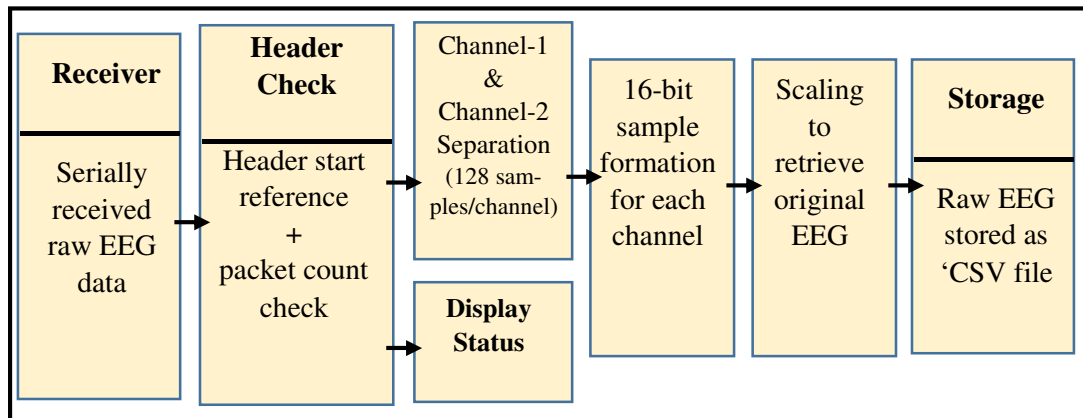


Fig.15: User console data acquisition system

4.4 Methodology

As depicted in Fig.10, entire thesis work was organized in two parts: Offline and Online for the proposed hybrid algorithm, each part having the MATLAB based and C language based microcontroller applicable algorithms. This section illustrates each mode with algorithm specific details, implemented methodology along with the achieved respective results to depict the validity of the developed algorithm.

4.4.1 Offline Mode

4.4.1.1 MATLAB based algorithm

Main designing part of the proposed hybrid algorithm resides in this developmental phase i.e. MATLAB based algorithm development in offline mode. Being offline, as explained in Table.2 of section 4.2.1, total 4 stored EEG datasets from two different devices (Emotive and NM) were used to verify the results at each stage of the algorithm for its overall performance evaluation. As mentioned in Fig.16, EEG data reading from CSV files, OA Detection – Removal and performance evaluation, everything was carried out using MATLAB software.

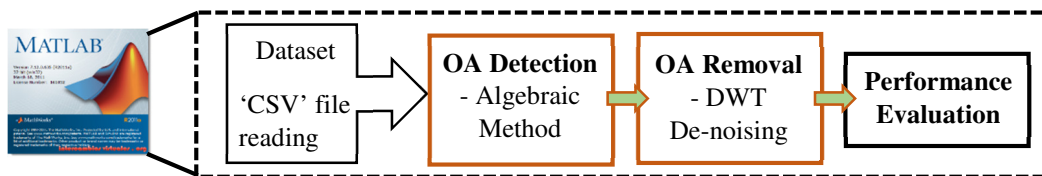


Fig.16: Block diagram of MATLAB based algorithm implementation in offline mode

- **OA Detection:**

OA detection code in MATLAB was developed as described in section 3.1 where FIR filter was implemented using MATLAB inbuilt function '*filter*' to extract the filter coefficients at every sliding window. The EEG epoch length as input to the algorithm

was taken for the entire length of recording as read from the CSV file. The interval length 'T' and the FIR filter order considered were as mentioned below:

Interval Length 'T' (sec)	Sampling rate 'Ts' (sec)	FIR filter order M = T/Ts
0.29	Emotive device: 0.0078	37
	NM device: 0.0039	74

The threshold function used was ' $\sigma * 1.5$ ', where, σ is the standard deviation of the EEG epoch. Using this threshold value, OA zone having starting edge and ending edge was determined in the algorithm. The verification of the entire process was tested using 4 datasets out of which the results of one emotive and one NM device EEG data are shown here, whereas for the other datasets the results are mentioned in Chapter 5.

- Results:** For emotive EEG data, epoch length taken was 9984 samples (78 seconds data) which was having total 9 eye blinks. The comparative plot in Fig.17 shows that OA detection algorithm accurately detected the eye blinks as well as the detected OA zones exactly matched with the actual existing eye blinks placements in raw EEG. The contents of OA zone detection array represents the OA zones in pairwise manner representing starting edge and ending edge respectively as indicated in Fig.17. The OA zone sample numbers were adjusted with FIR filter delay and 10 additional samples on each side. Similar results for FP1 (channel-1) location are presented in Fig.18 for NM device EEG data having epoch length of 11488 samples (45 seconds), consisting total of 8 eye blinks. For the FP2 (channel-2) location also similar results were observed.

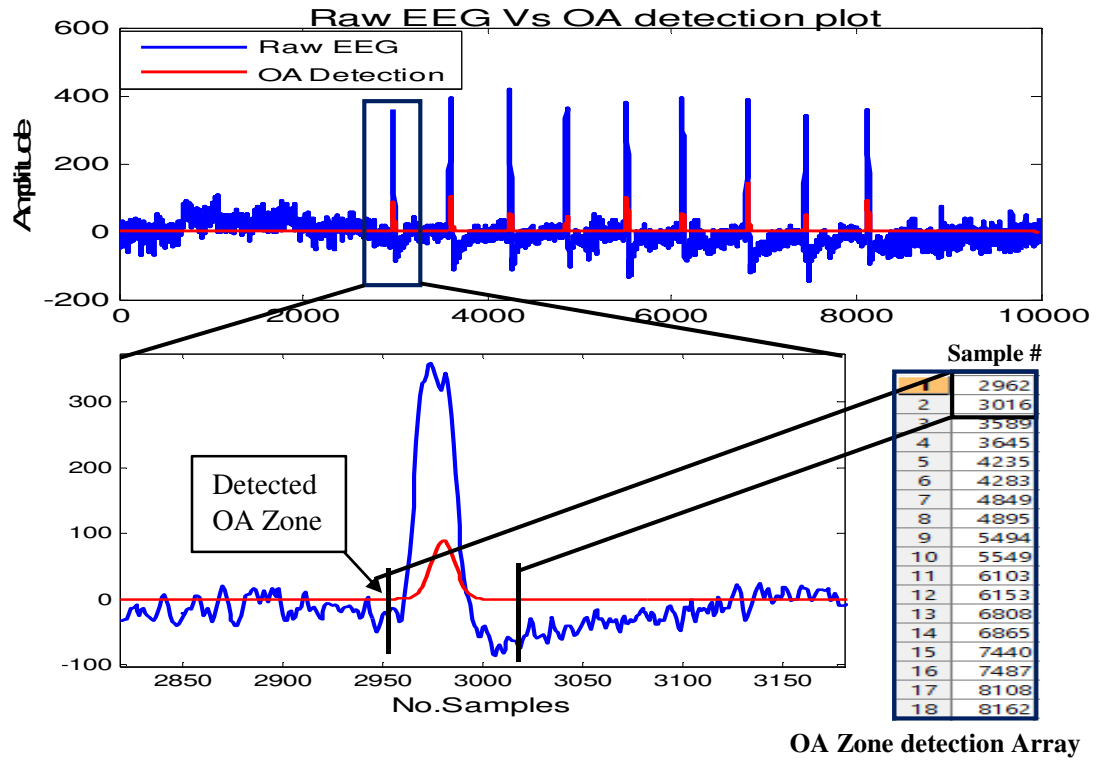


Fig.17: OA detection results for Emotive device (dataset-1)

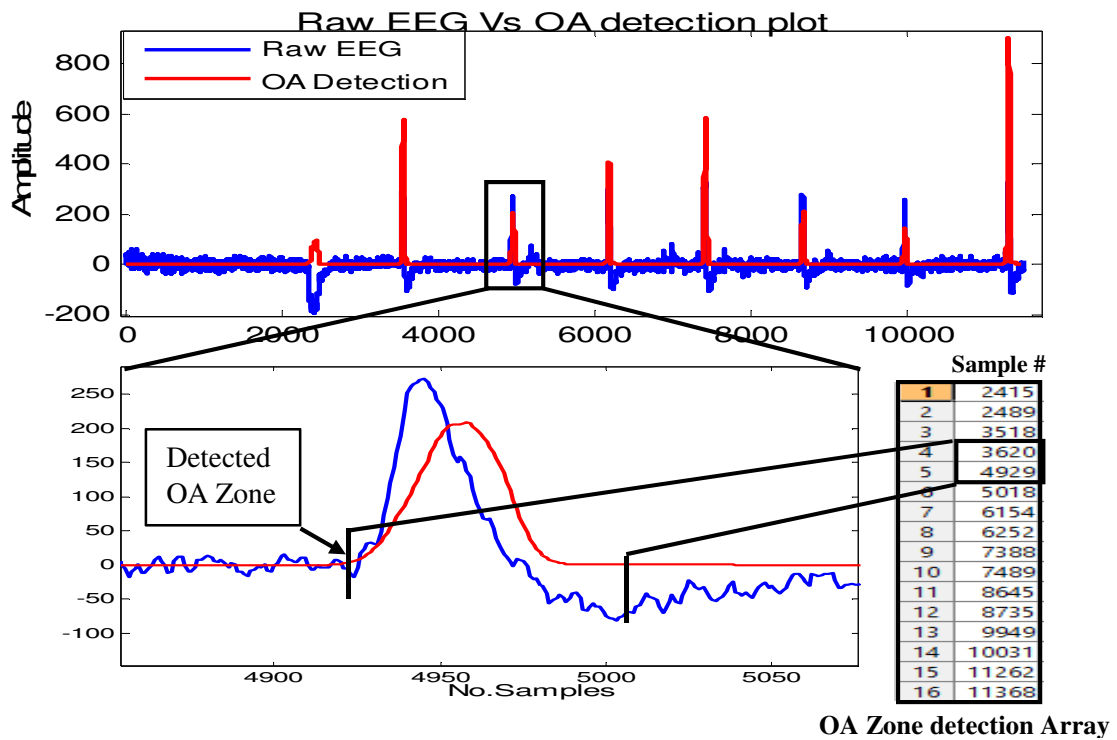


Fig.18: OA detection results for NM device (dataset-4)

- Addition of Overlapping Concept:** As discussed, the OA detection was carried out for the entire length of EEG recording. Whereas for the real-time application, the EEG epoch length should be as small as possible around 1 second or even 0.5 second to ensure the online processing without the longer latency periods. Thus, the same algorithm was tested for 2 seconds of epoch length for emotive dataset1. But it was observed that for the less than 2 seconds of epoch length, the OA detection algorithm missed some of the eye blinks getting detected as shown in Fig.19. This was probably due to the eye blink occurring at the end of the epoch. To overcome this issue, overlapping concept of every epoch was introduced as described in Fig.8 and as a result, even for the epoch length as small as 0.5 seconds, the eye blinks were detected without getting missed even a single one. With overlapping technique to detect the OA, threshold function (7) was found consistent.

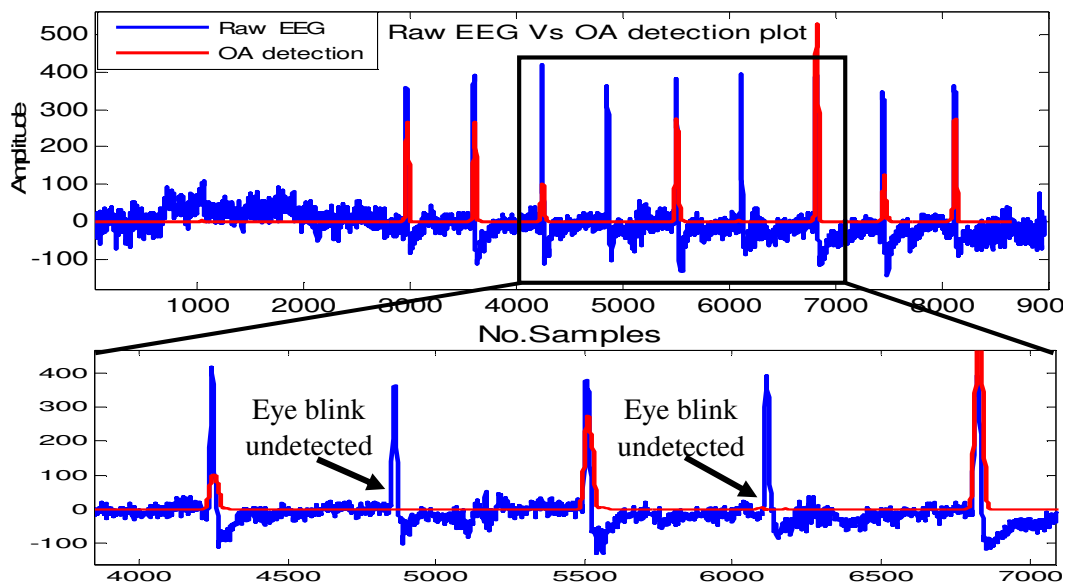


Fig.19: Undetected eye blinks by OA detection without overlapping method

Thus, OA detection code was verified for its performance with two different devices having different sampling rates and it was found consistent enough in detecting OAs.

- **OA Removal**

As discussed before, OA removal algorithm uses WT based de-noising technique and it was applied only to the detected OA zones rather than to the entire EEG epoch. Thus, only on detection of the OA zone, OA removal code gets executed otherwise it is skipped and hybrid algorithm seeks for the next EEG epoch. As per the section 3.1.2, SWT and DWT both were implemented initially to remove the detected OA from EEG and compared for their performances to select one out of the two for the final OA removal method. It was evident from the result shown in Fig.20 that the processing time of DWT was much lesser than SWT as predicted because of their individual processing methods of down-sampling and up-sampling respectively. The Fig.20 shows processing time comparison for each eye blink in the considered dataset. And looking to the real-time and hardware applicability, DWT was selected for the final hybrid OA removal algorithm.

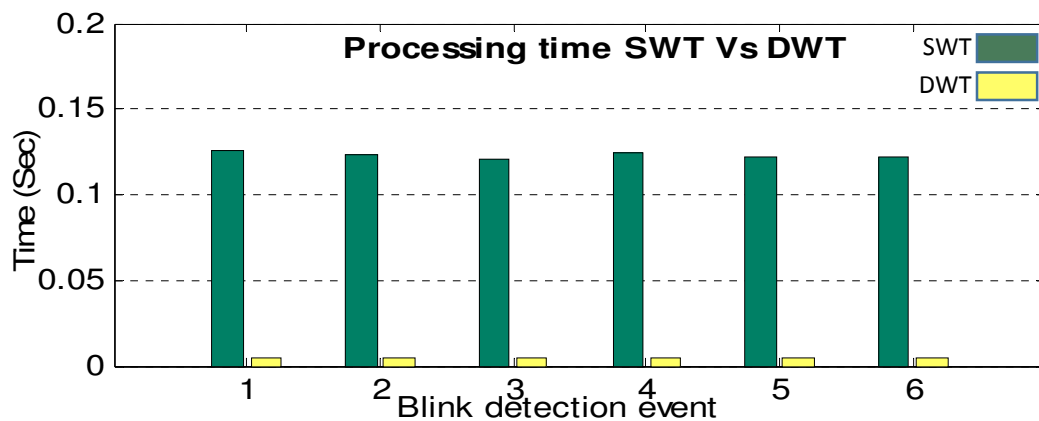


Fig.20: Processing time bar plot DWT Vs SWT

DWT decomposition was carried out up to level 7 for Emotive EEG device having 128 samples/sec data-rate whereas up to level 8 for NM EEG device having sampling rate of 256 samples/sec. For the decomposition, different wavelet functions, namely db6,

coif3/5 and *bior4.4* were evaluated for their performances. It was observed that wavelet function effectiveness was subjective depending on the individual's eye blink shape. But *bior4.4* was found to be consistent with majority of the datasets and all results shown here used *bior4.4* wavelet function. For the threshold based de-noising, the detailed coefficients of levels 3-7 for Emotive and levels 4-8 for NM were compared with threshold value and made to '0' if exceeded the threshold value. The eye blink occurs between 0-16Hz and the detail coefficients for the levels 3 to 7 and 4 to 8 belong to the frequency range of 0.5 - 16Hz [28] for the respective devices. The frequency band information at each DWT decomposition level is as mentioned below:

Decomposition level	Frequency bandwidth (Hz)	Decomposition level	Frequency bandwidth (Hz)
D1	64-128	D5	4-8
D2	32-64	D6	2-4
D3	16-32	D7	1-2
D4	8-16	D8	0.5-1

Finally, the reconstruction was carried out by replacing these de-noised chunks with the original signal to form the clean EEG.

For the OA removal algorithm evaluation, as per the Table.2, total 4 datasets were used to verify the OA removal algorithm out of which one dataset belonged to Emotive EEG device, whereas remaining three were of NM device as finally the hybrid algorithm was to be implemented on NM device only. All results are plotted for the raw EEG Vs de-noised EEG after OA removal along with zoomed plot using *bior4.4* wavelet function. The Emotive dataset results are shown in Fig.21, although it was observed that *db6* worked better for this particular dataset. Also it is evident from Fig.21 that de-noising was carried out only in OA Zone and non-OA zones remained entirely intact without modifying any EEG information in that region.

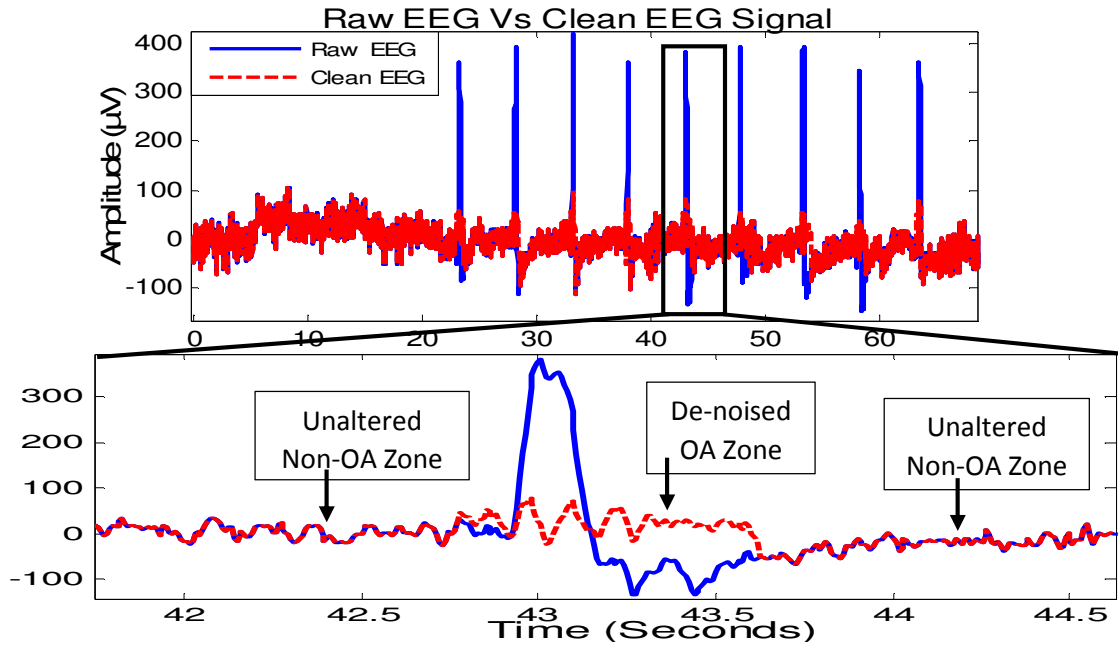


Fig.21: OA removal results for Emotive device (subject-1)

For all NM device datasets comprising two channels of three different subjects, *bior4.4* showed equivalent satisfactory results out of which one dataset (channel1) result is mentioned in Fig.22 whereas others are shown in result section of Chapter 5.

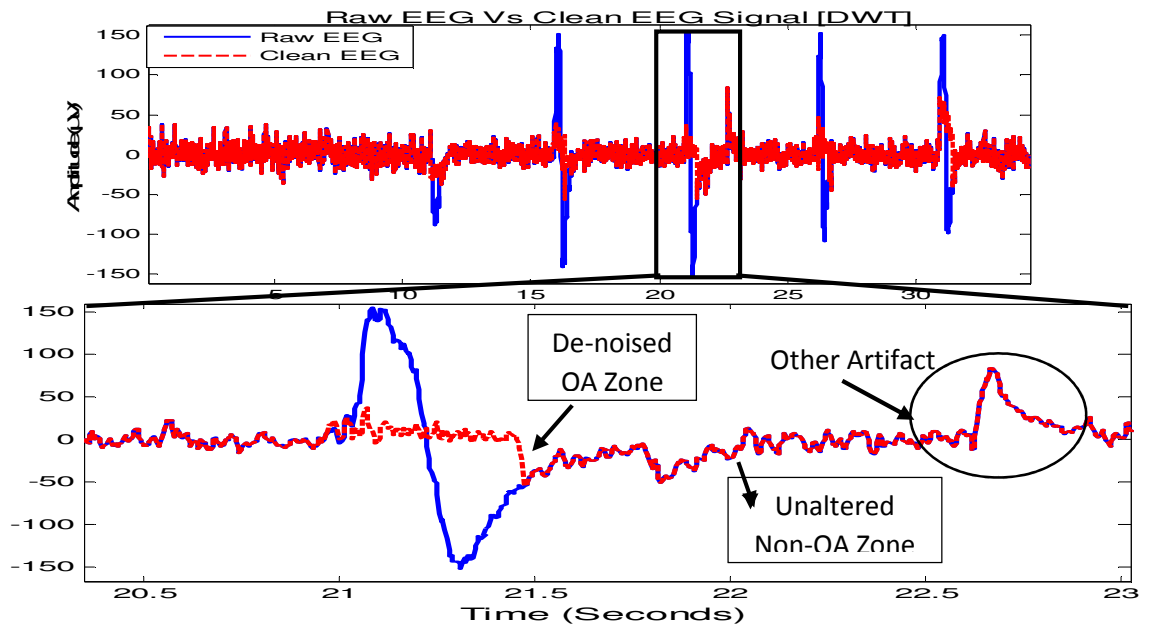


Fig.22: OA removal results of NM device (channel-1(FP1) subject-2)

Discussion: It was noticed after de-noising in OA Zone that at times the edges had sharp drops or falls as visible in Fig.20. One possible reason of this might be the abrupt OA-zone cut-off from the regular EEG signal. Also from Fig.22 it is evident that artifacts other than OA didn't get detected or de-noise.

After this developmental phase of MATLAB based offline algorithm evaluation, the hybrid OA removal algorithm was finalized with algebraic method based OA detection and DWT based OA removal using *bior4.4* wavelet function by implementing overlapping of EEG epochs having 0.5 second of length for NM EEG device.

4.4.1.2 C based algorithm

Offline C language based code was simply the prototype of online C based microcontroller implementable OA removal code which was developed similar to the MATLAB based finalized hybrid algorithm using Visual Studio 2010 software tool. The verification of the offline C code was done in MATLAB by comparing MATLAB and C code very precisely at every step of the algorithm, refer Fig.23.

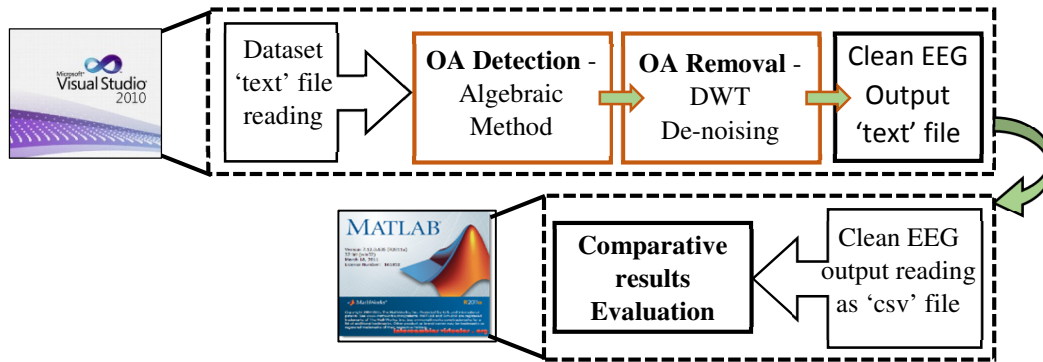


Fig.23: Block diagram of C based algorithm implementation in offline mode

There was one major difference in C based algorithm as compared to finalized hybrid OA removal technique and that was the wavelet function used. So far discussed, *bior4.4*

wavelet was used to decompose the EEG epoch using DWT for all the datasets as that matched the most with the shape of the eye blinks of different subjects producing satisfactory results. The C based algorithm was basically developed for implementing it on MCU and thus, as a first stepping stone, to simplify and reduce the algorithm complexity, instead of *bior4.4* wavelet, *haar* wavelet was implemented.

A *haar* wavelet is the simplest type of wavelet and it serves as a prototype for all the other wavelet transforms. The *Haar* transform decomposes a discrete signal into approximation and detailed coefficients each of half its length. The approximation coefficients are the running average whereas the detailed coefficients are the running difference. The precise formula for the approximation (a_m) and detailed (d_m) coefficients for the discrete signal f of length N are given as (9) and (10) respectively, for $m = 1, 2, 3, \dots, N/2$. [29]

$$a_m = \frac{f_{2m-1} + f_{2m}}{\sqrt{2}} \quad (9)$$

$$d_m = \frac{f_{2m-1} - f_{2m}}{\sqrt{2}} \quad (10)$$

The discrete signal under process was EEG signal of length 128 samples (i.e. 0.5 sec data - using 256 sps) and another minor change made to induce simplicity was that the signal decomposition levels instead of eight, seven was performed using *haar* wavelet (without compromising with the result accuracy). For the threshold based de-noising, levels 4-7 (0.5 - 16 Hz) were compared with threshold value for de-noising purpose. The graphical representation of the decomposed signal vector is shown in Fig.24. Finally, inverse DWT for the signal reconstruction was carried out as per (11) to retrieve the clean EEG.

$$f = \left(\left(\frac{a_1 + d_1}{\sqrt{2}} \right), \left(\frac{a_1 - d_1}{\sqrt{2}} \right), \dots, \left(\frac{a_{N/2} + d_{N/2}}{\sqrt{2}} \right), \left(\frac{a_{N/2} - d_{N/2}}{\sqrt{2}} \right) \right) \quad (11)$$

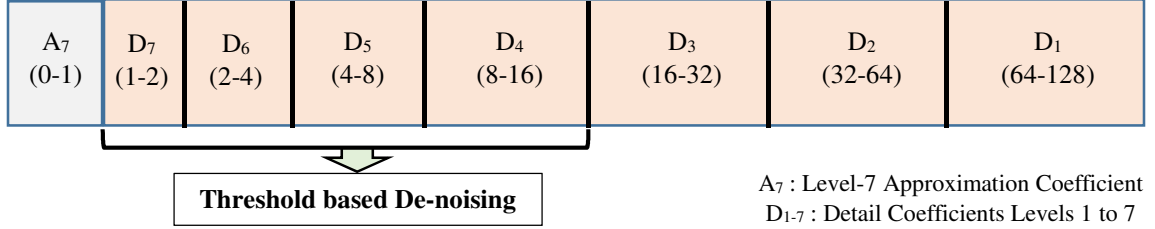


Fig.24: Graphical presentation of DWT based decomposed signal vector

NM device datasets of two subjects were tested to verify C code based OA removal algorithm against the same MATLAB algorithm results using *haar* wavelet. Comparative results are shown in Fig.25 and Fig.26 which clearly depicts that the developed C code gave better to equivalent results when compared with MATLAB based algorithm outputs.

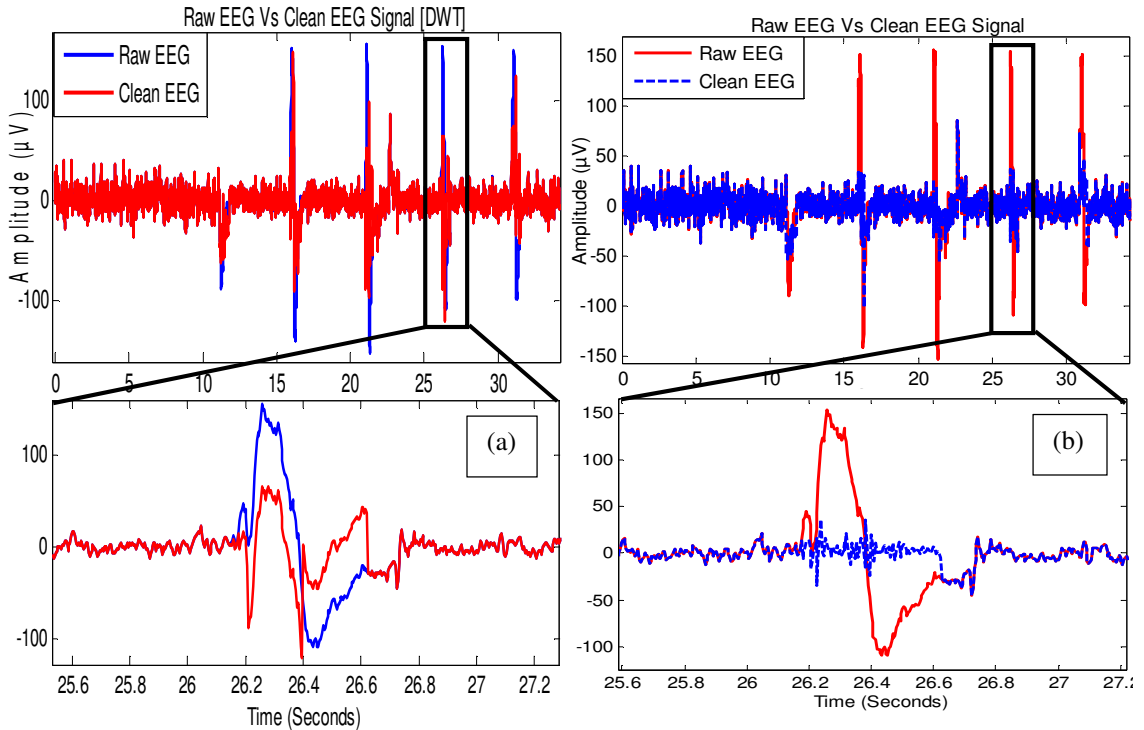


Fig.25: Hybrid algorithm results for NM device (C based – offline mode) (subject-1)
 (a) MATLAB result, (b) C based result

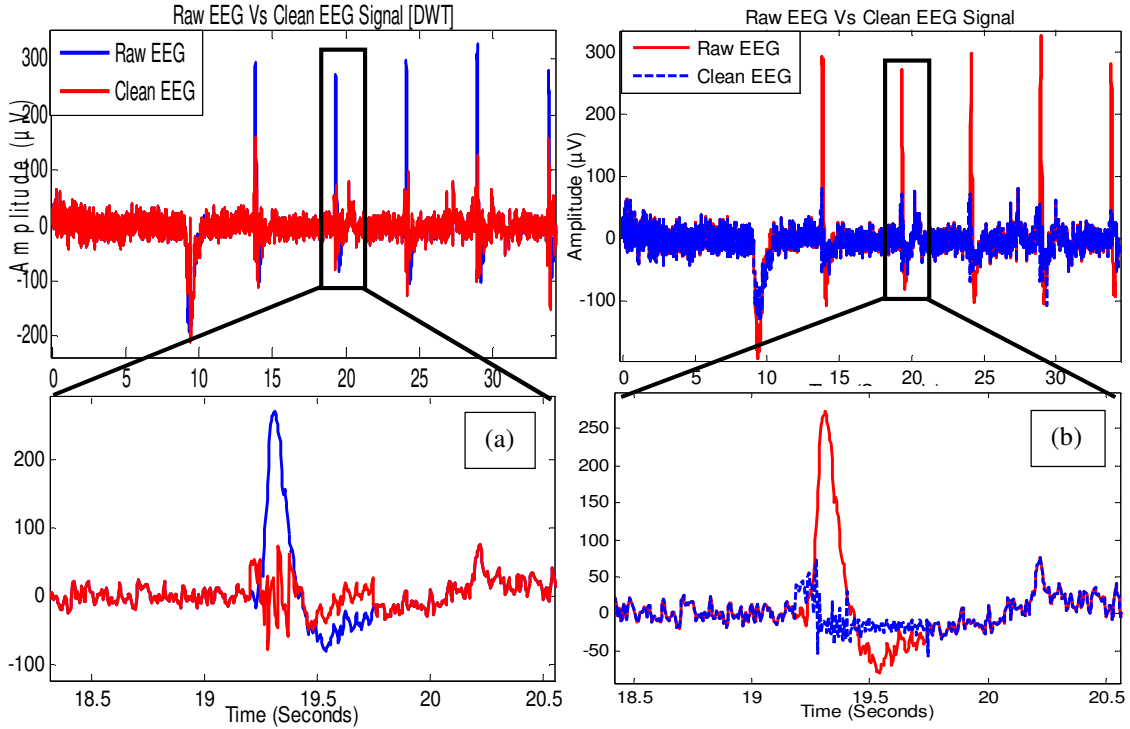


Fig.26: Hybrid algorithm results for NM device (C based – offline mode) (subject-2)
 (a) MATLAB result, (b) C based result

This verified C based OA removal algorithm in offline mode was then ready to be implemented on PSoC-3 MCU of NM device in real-time as described in next section.

4.4.2 Real-Time Mode

4.4.2.1 MATLAB based algorithm

In real-time mode, instead of already stored EEG datasets, raw EEG was acquired using NM device as described in section 4.3 following the protocol mentioned in section 4.2.2 where EEG recordings from 4 subjects in real-time, consisting total of 30 seconds of EEG having total 5 eye blinks at every 5 seconds interval were considered. The wireless EEG data were captured serially using MATLAB data acquisition software [25] and hybrid algorithm written in MATLAB was ran as a sequential part of data acquisition software to obtain clean EEG in real-time, refer block diagram of Fig.27. Because

of the 40 samples of overlapping method, on the receipt of every 0.5 second (128 samples) EEG data, 88 samples were stored as a clean EEG output and last 40 samples were reconsidered as a part of new epoch of 128 samples, refer Fig.8.

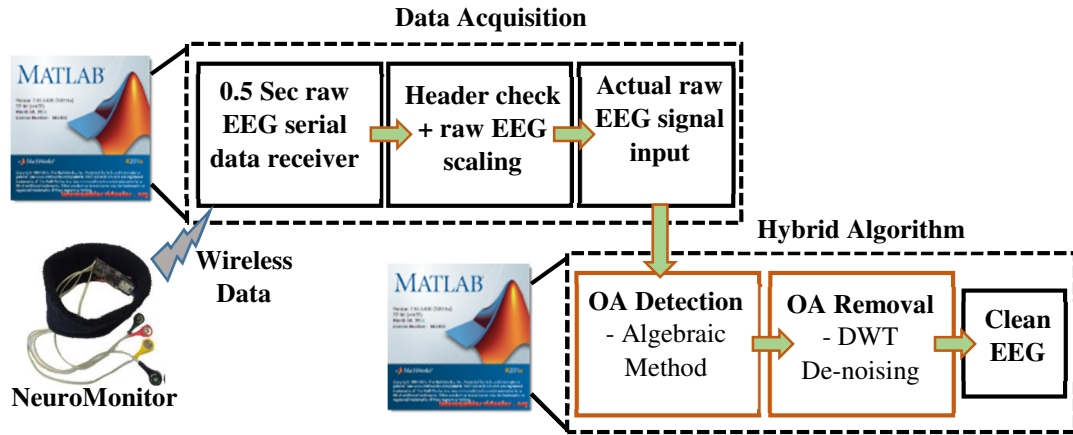


Fig.27: Block diagram of MATLAB based algorithm implementation in online mode

The real-time performance of the MATLAB based OA removal algorithm using channel-1 EEG recording of 2 subjects are shown in Fig.28 to Fig.29 whereas results for the other 2 subjects are shown in chapter 5. The results show the comparative plots of raw EEG and clean EEG stored at the end of the execution of the online OA removal algorithm. It was found that MATLAB based finalized hybrid OA removal algorithm (section 4.4.1.1) worked accurately and efficiently in real-time without any miss of the data or glitch in the OA detection or OA removal algorithm.

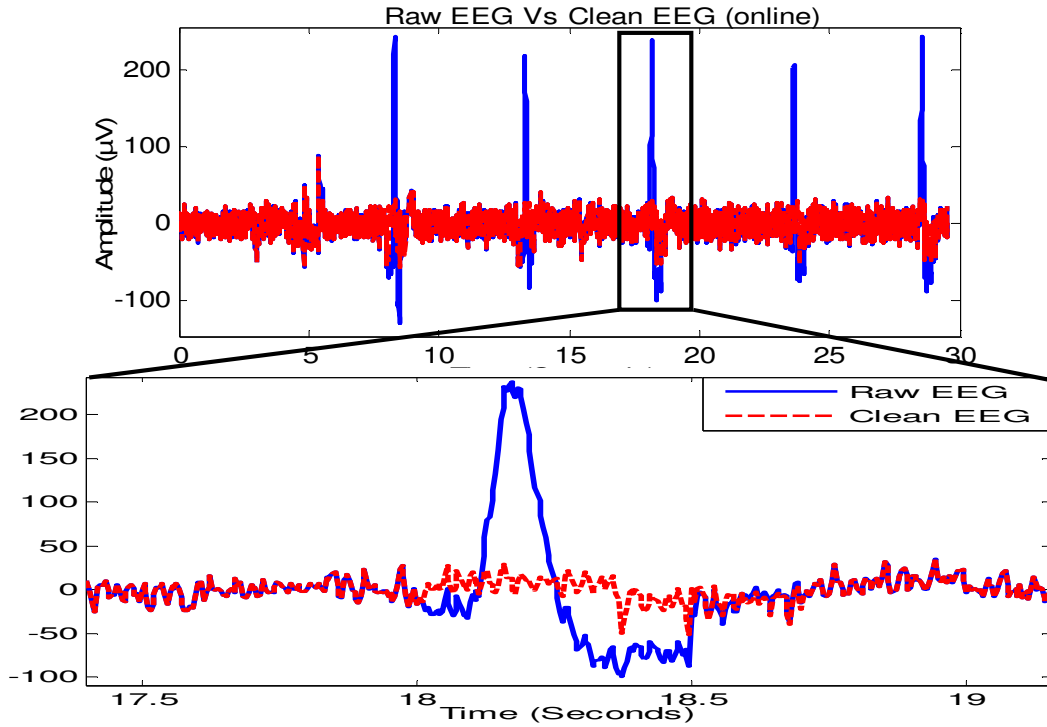


Fig.28: Online mode – MATLAB based hybrid Algorithm result (channel-1, subject-1)

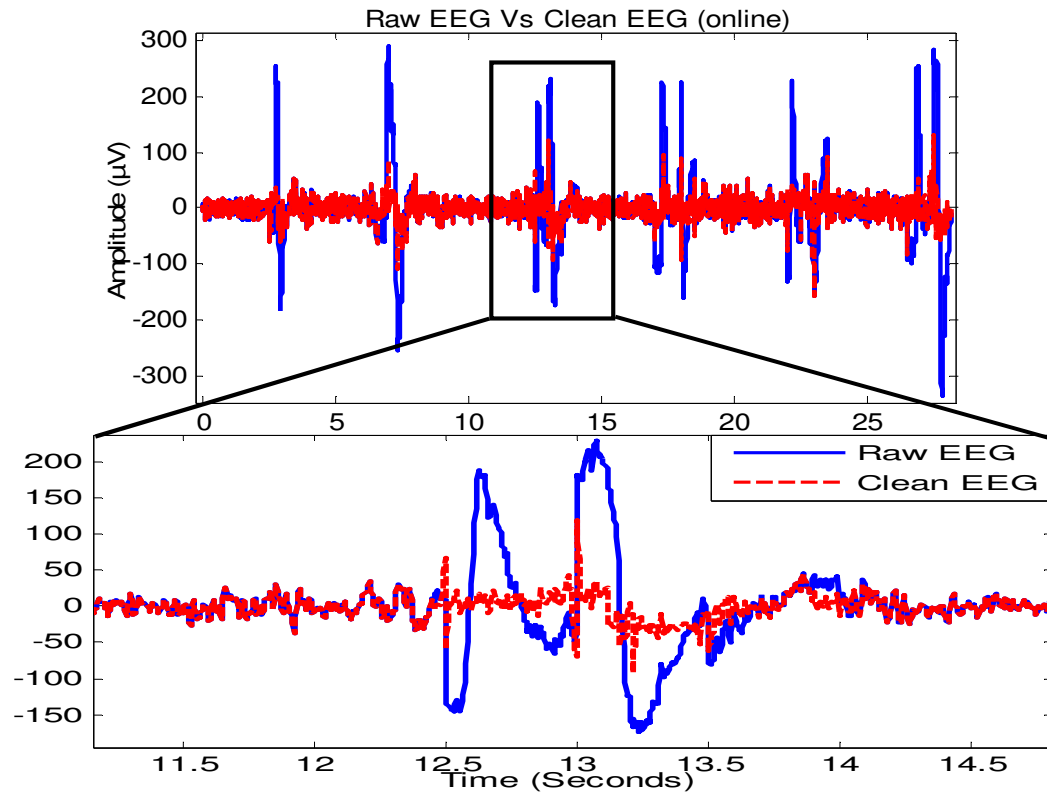


Fig.29: Online mode – MATLAB based hybrid Algorithm result (channel-1, subject-2)

4.4.2.2 C based algorithm

Online C based hybrid OA removal algorithm was developed to be executed in real-time on PSoC-3 MCU situated on NM board. The PSoC-3 MCU of NM device was already in-use for recording EEG in real-time settings and sending EEG data wirelessly to the user console. Thus, this existing hardware was chosen to implement the hybrid OA removal algorithm for the algorithm verification purpose. The verified C based hybrid OA removal algorithm in offline mode was combined with PSoC-3 based EEG acquisition software (described in section 4.3.1) and finally the raw EEG along with the clean EEG was transmitted wirelessly using Bluetooth communication. This wirelessly sent output buffer was received remotely on MATLAB based user console, where scaling of the received raw EEG along with mean adjustment was carried out to compare with PSoC-3 processed and de-noised EEG, for the online C based algorithm verification. The entire process is illustrated as block diagram in Fig.30. The software tool used to program PSoC-3 MSU, was ‘PSoC creator 3.0’ whereas the remote data acquisition software was written in MATLAB (refer section 4.3.2).

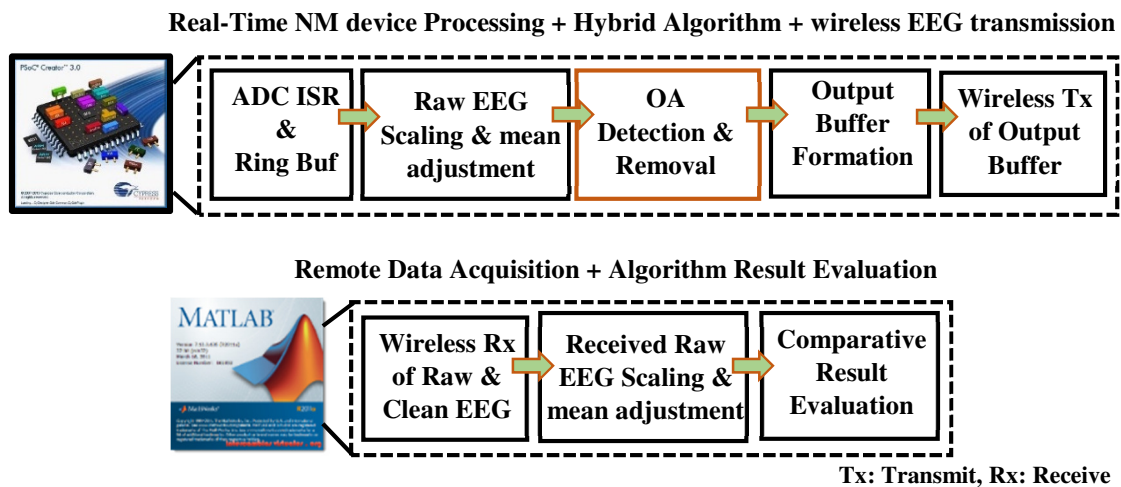


Fig.30: Block diagram of C based algorithm implementation in online mode

- Overlapping method implementation challenge:** Main challenge for the online C based hybrid OA removal method was to implement the overlapping of the epochs in real-time. This was carried out using the single *ring-buffer* technique. The size of the ring buffer designed in the code was 768 bytes long where 16-bit ADC sampled EEG data gets stored at every 3.9 ms (256 sps) on ADC generated interrupt. The respective interrupt service subroutine (ISR) stored two bytes (16-bit sample) sequentially in ring buffer. The ring-buffer was indexed with 'LAST' and 'FIRST' variables as '*put index*' and '*get index*' respectively. On every 16-bit ADC data arrival, ISR gets executed where two bytes were stored in ring buffer at the locations pointed by the *put index* – LAST. The *get index* – FIRST was used to fetch the samples for processing from ring buffer in *main* program running in real-time. The ISR execution graphical representation is given in Fig.31. 'OverFlow' variable is set to '1' whenever LAST crosses the buffer size and gets reset to starting index '0'. As in Fig.31, the race condition between FIRST and LAST was also checked whenever OverFlow was '1' and avoided new sampled data to get stored by holding (dropping) them.

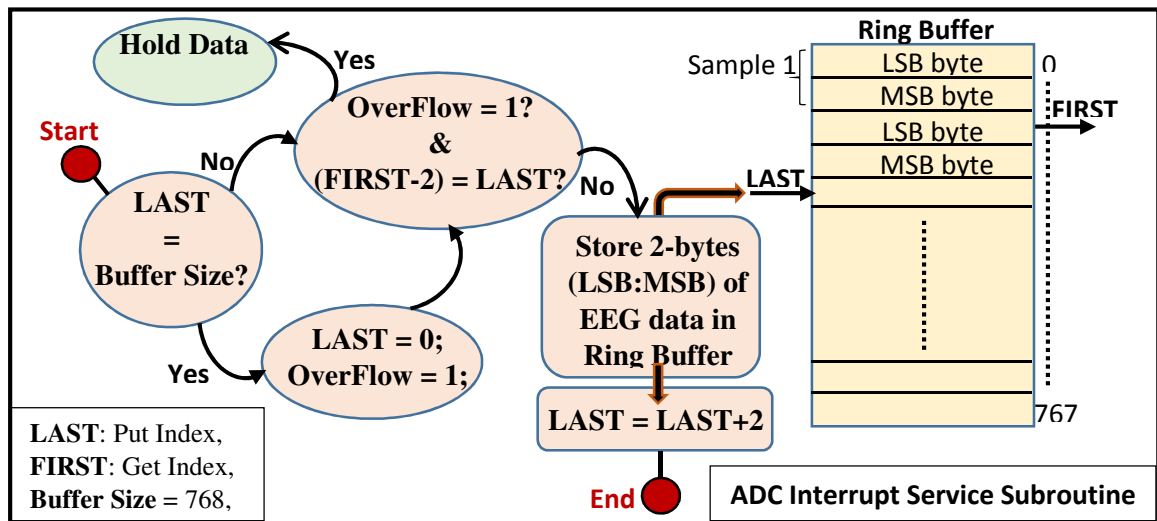


Fig.31: Graphical presentation of ADC ISR execution steps

- **Microcontroller implemented program flow:** Fig.32 illustrates the overall *main* program flow implemented on PSoC-3 MCU comprising ring buffer access, steps to perform epoch overlapping, OA detection, OA removal, output data buffer formation and its serial transmission. The code is attached in Appendix A. In the flowchart of Fig.32, the ‘count’ indicates the output buffer packet number which was incremented after every output buffer transmission. In the algorithm, for the very first EEG epoch i.e. processing at the time zero, ‘count’ value was ‘0’, when 128 samples were fetched from the ring buffer starting from the location pointed by the global variable *get index*-FIRST. For all other consecutive EEG epochs, only 88 samples of new data from ring buffer pointed by FIRST were fetched because the last 40 samples from previous epoch were considered as the starting 40 samples for the next epoch out of total 128 samples to be processed. ‘Valid_Data’ holds the count of available data between the two ring buffer indexes, LAST and FIRST for the next epoch processing. The ‘y1’ buffer holds the scaled and mean adjusted actual 128 samples of raw EEG as input for the on-chip OA detection and removal algorithm to get the de-noised EEG in real-time. ‘Clean_EEG’ is the output buffer of length 537 bytes which is finally transmitted wirelessly using Bluetooth serial communication. The ‘Clean_EEG’ buffer is comprised of three parts: Header (9 bytes), raw EEG ($88 \times 2 = 176$ bytes) and clean EEG ($88 \times 4 = 352$ bytes). The raw EEG data being 16-bits long, occupied 2-bytes whereas clean EEG data was the outcome of hybrid OA removal algorithm having float values, occupied 4-bytes per data sample. Considering overlapping method, for the processed EEG of array ‘y1’, always the first de-noised 88 samples were stored in ‘Clean_EEG’ buffer as clean EEG whereas for the raw EEG, overlapping of the data was performed as shown in Fig.33.

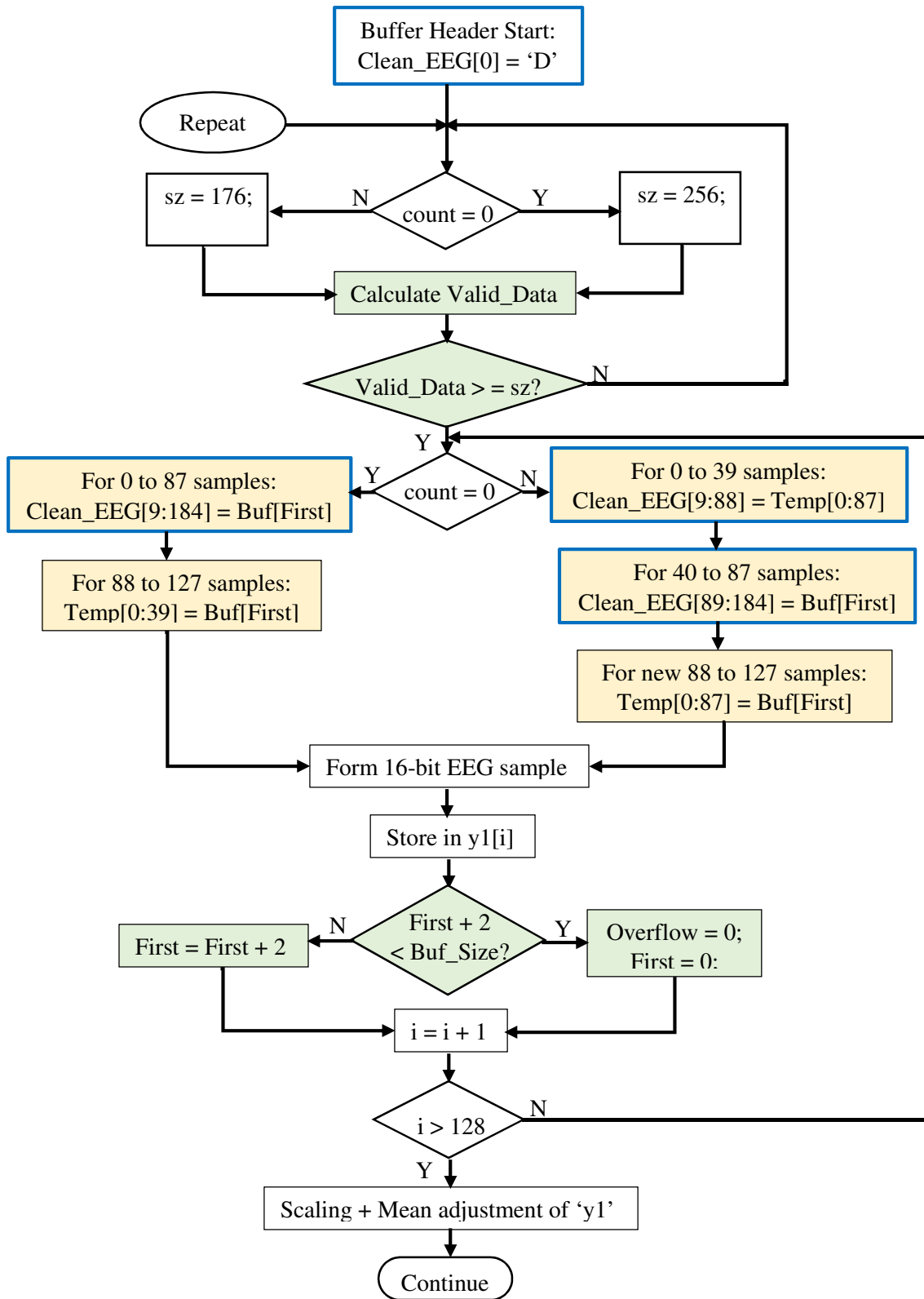


Fig.32: Online mode C based *main* program flowchart implemented on PSoC-3 MCU

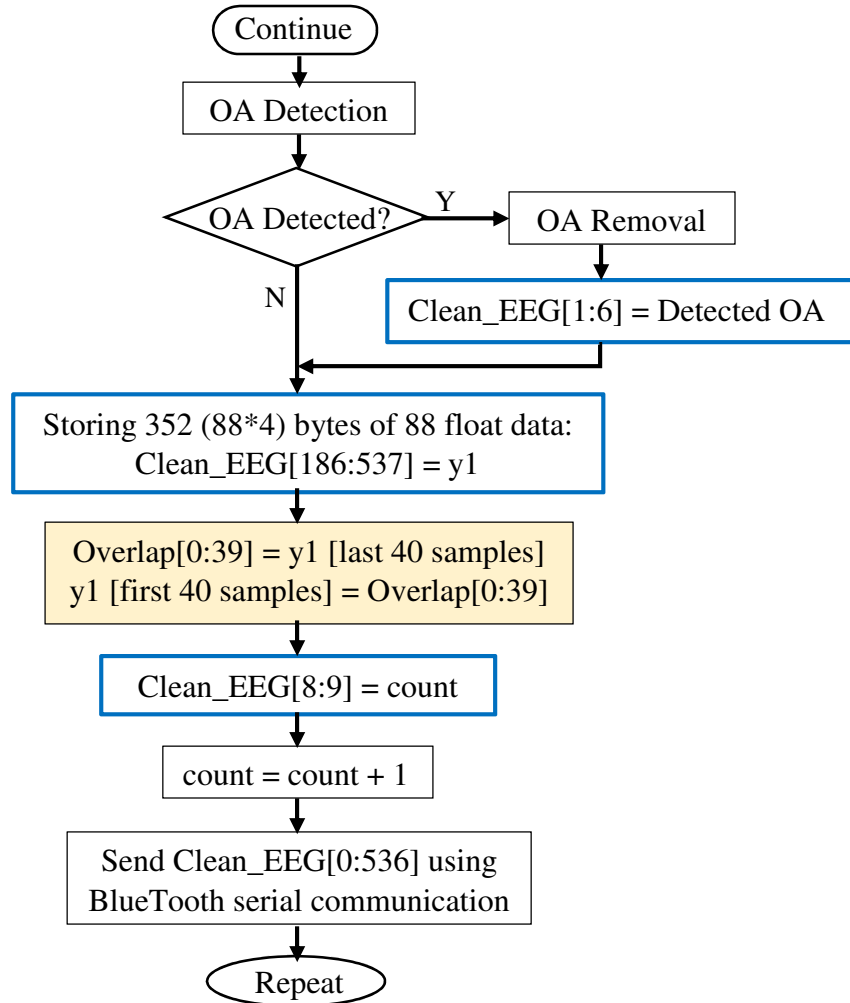


Fig.32: Online mode C based *main* program flowchart implemented on PSoC-3 MCU

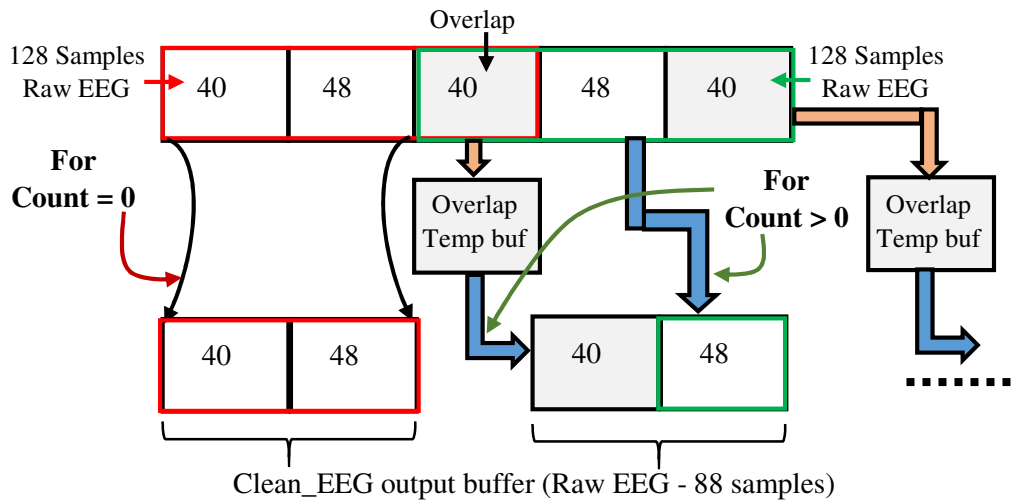


Fig.33: Graphical presentation of real-time overlapping method implementation

For the 1st data packet (i.e. count =0), out of 128 new data fetched from ring buffer, initial 88 samples were stored in ‘Clean_EEG’. The last 40 samples were moved to the temporary buffer which was then stored in ‘Clean_EEG’ for all the next epochs (i.e count > 0) followed by next new 48 samples as shown in Fig.33. The ‘Clean_EEG’ output buffer formation is indicated in Fig.34 (a). In 9-bytes long header, 1st byte contains the start-up key designated as character ‘D’ which was extracted at the remote user console to authenticate the start of the received packet and the packet count was stored as 8th and 9th bytes. The intermediate 6 bytes were utilized to store up to three detected OA zone sample locations in pairwise manner (starting edge and ending edge) per epoch. The header formation described is represented graphically in Fig.34 (b).

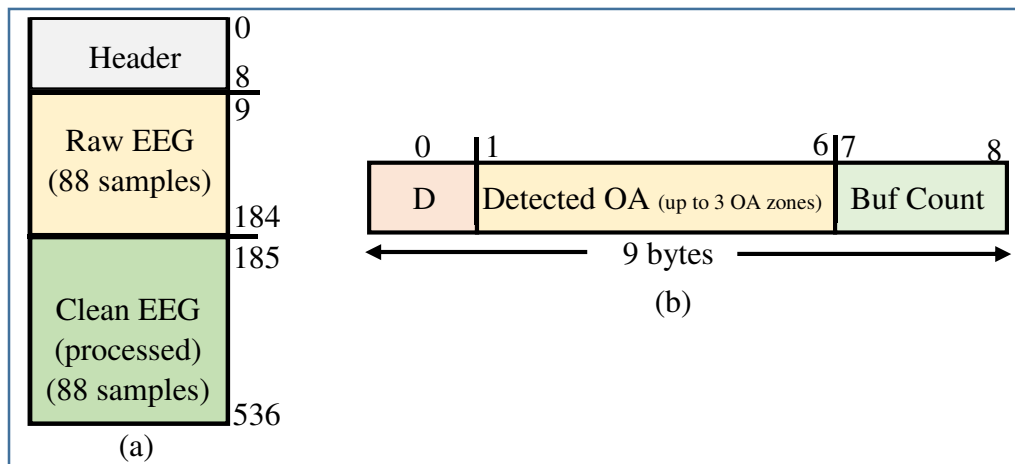


Fig.34: (a) ‘Clean_EEG’ output buffer formation, (b) Header formation

- **Results verification:** The implemented online hybrid OA removal algorithm, was verified for its performance using MATLAB based remote data acquisition software. The ‘Clean_EEG’ was received at the remote console, where raw EEG and clean EEG were separated and compared for the MCU implemented algorithm evaluation. It was important to first verify the ring buffer performance as that was the most critical part in the

online C based code. For this purpose, only the code with ring buffer performing overlapping for raw and clean EEG was tested in real-time, by skipping off OA detection and removal algorithms for the instance. To authenticate the performance of the ring buffer in real-time, the raw and clean EEG should contain exactly the same data at the receiver as no OA removal processing is carried out for this test. The implemented ring buffer and overlapping methods worked perfectly without any mismatch. The raw and clean EEG received at the user console were plotted for comparison and it was observed that both matched one to one without any discrepancy, refer Fig.35.

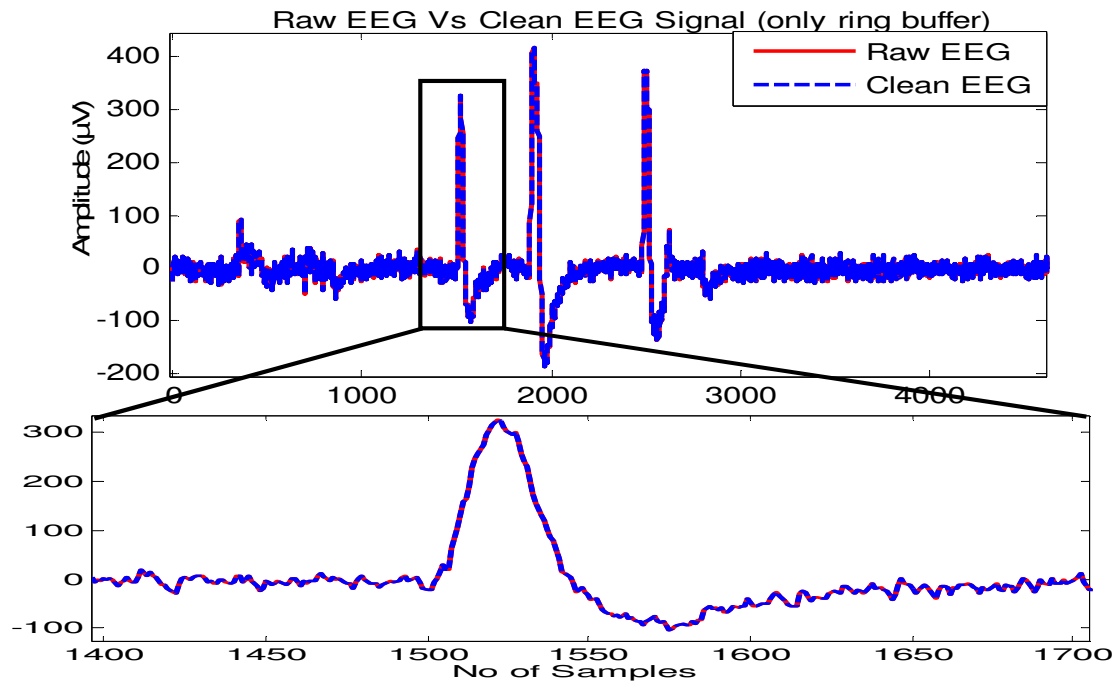


Fig.35: Ring buffer output comparison on user console – online mode (C based)

After verifying the ring buffer consistency, the hybrid OA removal algorithm was then included along with the ring buffer to observe the final outcome. On comparing raw and clean EEG on MATLAB platform, it was noticed that hybrid OA removal algorithm did de-noise the existing eye blink in the EEG successfully. The test was carried out for 3

datasets of EEG recordings from single subject and results for dataset1 is shown in Fig.36 (a). The detected OA zones for dataset1 is shown in Fig.36 (b), where the snapshots of MATLAB results of 'Clean_EEG' array header contents (index 1 to 9) are highlighted. Every packet of length 537 bytes starts with character 'D' (received as '68'), next 6 bytes contains the OA zones if detected followed by the packet count numbers as highlighted in the Fig.36 (b).

Discussion: Before the OA zone was detected, there were total 9 complete packets having 88 samples in each. Thus, the calculation for OA zone for the 10th packet as detected is shown in the Fig.36 (b), which matched with the actual position of the eye blink. This shows that OA detection algorithm worked perfectly on PSoC-3 hardware in real-time accurately. Accordingly, the de-noised eye blink zone is noticeable in Fig.36 (a) which depicts that also the OA removal algorithm implemented on PSoC-3 MCU performed satisfactorily as desired in the online mode.

- **Challenges faced:**

1. The original order of FIR filter implemented was 74. But it was observed that implementing this higher order FIR on hardware for real-time operation, the MCU processing became very slow and remotely received EEG data encountered some packet misses. This slow processing was resolved by reducing the FIR filter order down to 38 by considering small interval 'T' (refer section 3.1) as 0.15 seconds instead of original 0.29 seconds.

2. The small region of the raw and clean EEG as zoomed-in plots for dataset1 is shown in Fig.37. It was observed that in non-OA zone, both of the signals had little dis-

crepancy and didn't overlap on each other. After investigating it was found that the reason for this mismatch was the difference in mean adjustment done for raw and clean EEG. For the clean EEG, mean was calculated in real-time on PSoC-3 MCU whereas for the raw EEG, it was carried out in offline mode in MATLAB before comparative results study. Thus, the epoch length considered for calculating mean differed in both cases which ultimately resulted in little mismatch or non-overlapping EEG samples as shown in Fig.37. Though this does not create much adverse effect on the received EEG signal but the mismatch problem can be resolve by adding small MATLAB script to extract and calculate the exact epoch considered in real-time on PSoC-3 using the information of actual packet count received on the user console and then carry out the mean for raw EEG. This additional MATLAB script can help to resolve the mismatch issue which has not been implemented here in the thesis work. Also this mismatch still implies that the EEG information in non-OA zone remained unchanged as there was no OA detection in that region.

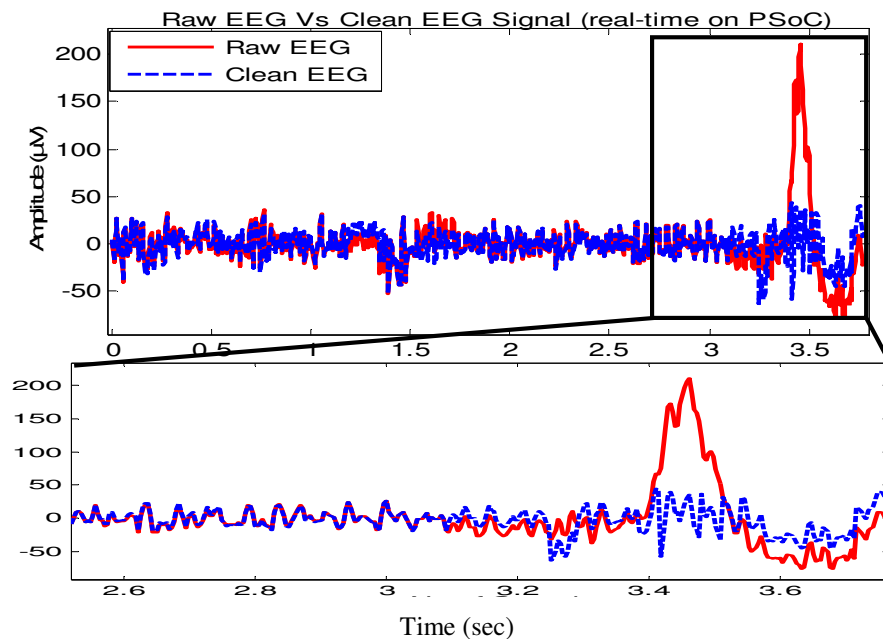


Fig.36: (a) Online mode – C based hybrid Algorithm result (FP1 – dataset1)

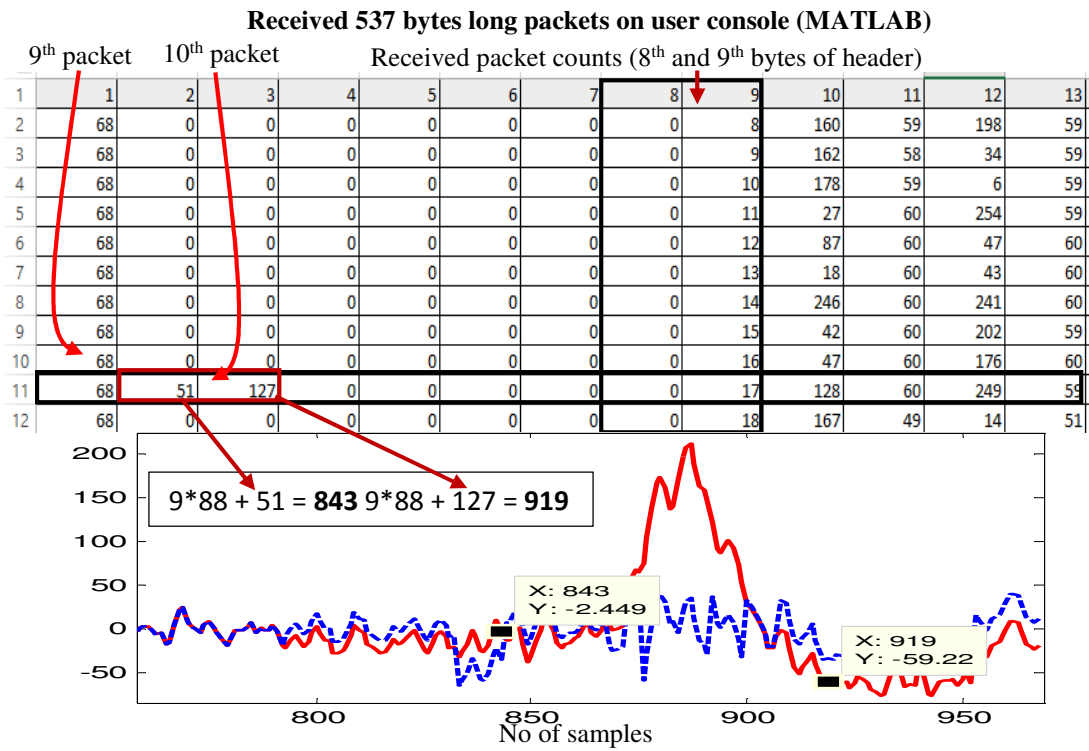


Fig.36: (b) Verification of OA zone detection C based algorithm in real-time (dataset1)

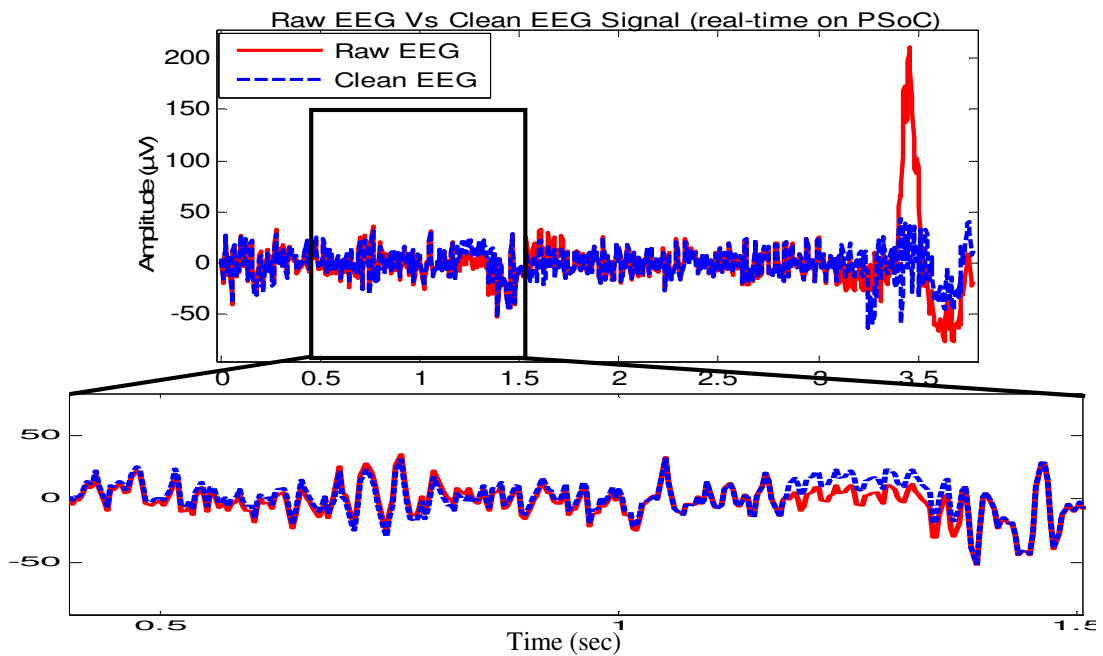


Fig.37: Plot showing minor mismatch in non-OA zone between raw and Clean EEG of real-time hardware implemented algorithm output

RESULTS AND PERFORMANCE EVALUATION

5.1 OA Detection and Removal Results

In chapter 4 along with the methodology, the respective results achieved were presented in detail for some of the datasets. In this section the remaining datasets for each offline and online modes which were not covered up in previous chapter, are presented with required discussion. The results of offline C based algorithm for all datasets were mentioned in previous chapter and thus it has not been included in here.

5.1.1 Offline MATLAB based algorithm

The offline MATLAB based algorithm results are presented in two parts:

i.) OA detection results ii.) OA removal results

- **OA detection results:**

The OA detection algorithm results of datasets (Emotive device – 128 sps) for subject-2 and subject-3 are as shown in Fig.38 and Fig.39.

Discussion:

It is evident from the results that OA detection algorithm detected existing eye blinks in EEG accurately without detecting the noise present as other artifact. The detected starting and ending edge of the eye blink exactly pointed the OA position in the raw EEG signal.

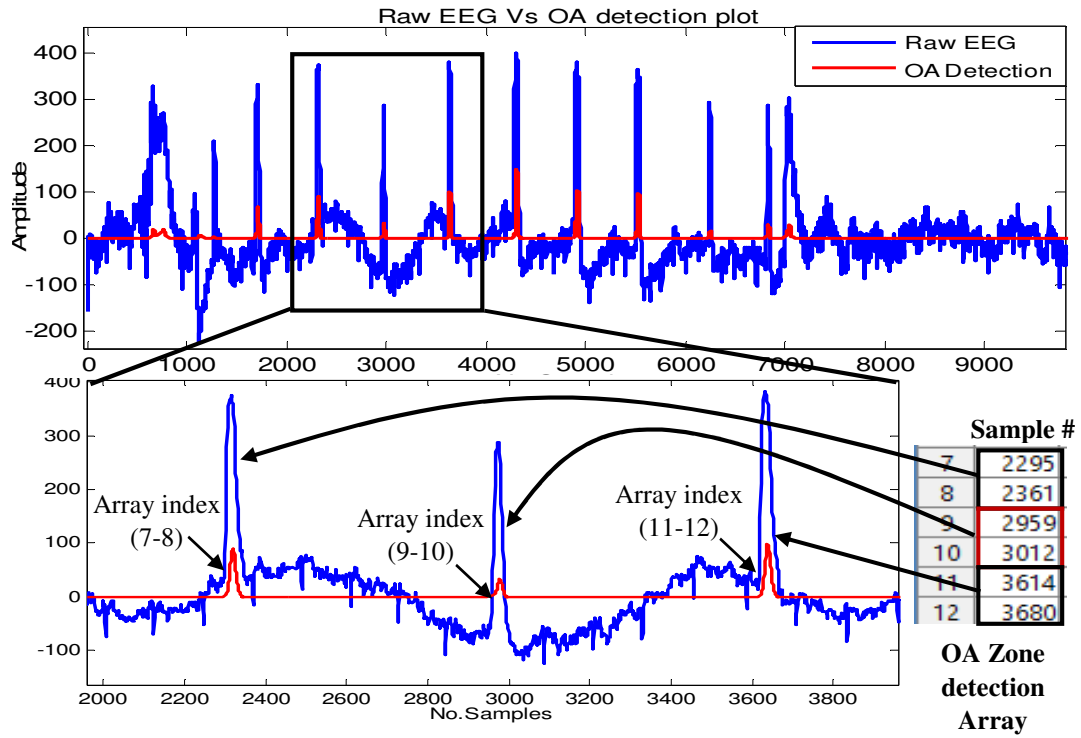


Fig.38: OA detection results for Emotive device (dataset-2)

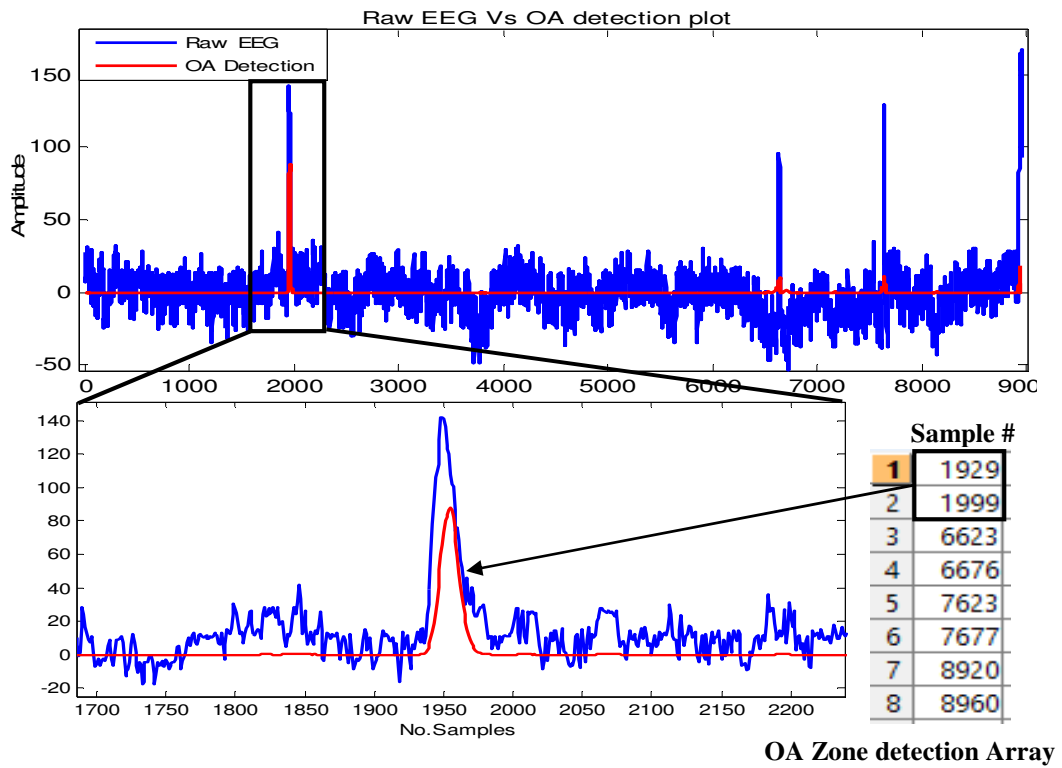


Fig.39: OA detection results for Emotive device (dataset-3)

- **OA removal results:**

Offline MATLAB based OA removal algorithm results for the subjects 3 and 4 for channel1 (FP1) and for the subjects 2, 3 and 4 for channel2 (FP2) are shown in Fig.40, Fig.41 and Fig.42.

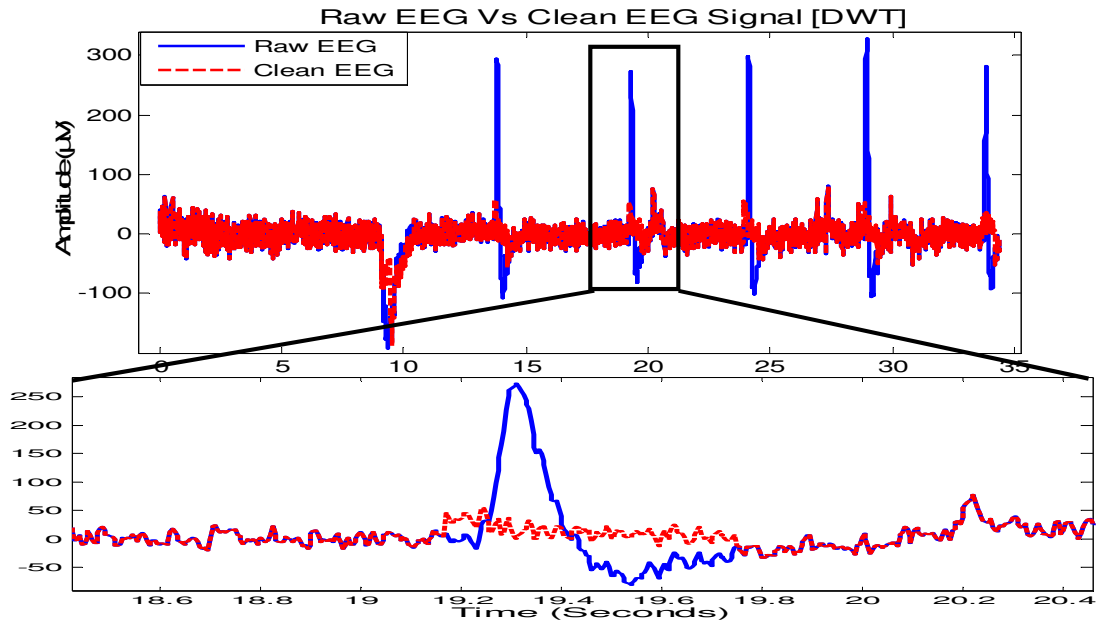


Fig.40: OA removal results of NM device (channel-1(FP1) subject-3)

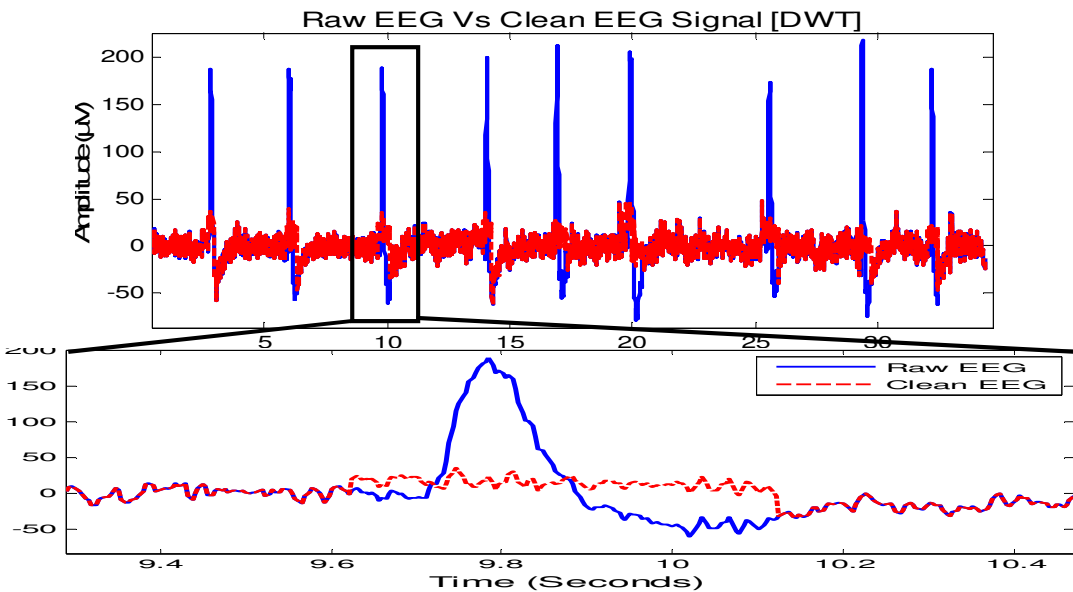


Fig.41: OA removal results of NM device (channel-1(FP1) subject- 4)

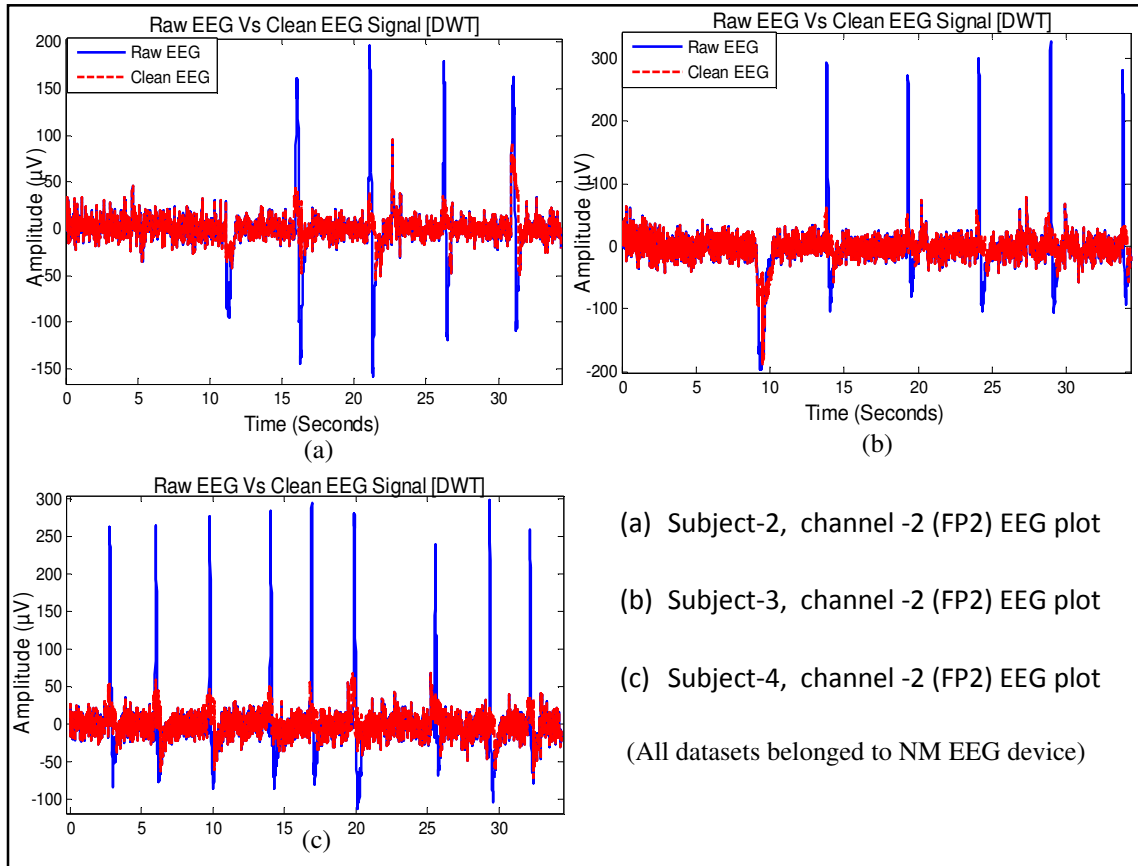


Fig.42: OA removal results of NM device (channel-2(FP2), subjects 2, 3 & 4)

The subject-2 dataset had total 5 OA, subject-3 had 6 OA and subject-4 had total 9 eye blinks in the EEG recording which all were removed using DWT decomposition method with *bior4.4* successfully. It was observed that in non-OA zone the background EEG information was not modified at all by retaining all important information.

5.1.2 Online MATLAB based algorithm

Out of the total 4 subjects EEG data captured in real-time, results of subjects 3 and 4 for MATLAB based hybrid OA removal algorithm in online mode is presented in Fig.43 and Fig.44.

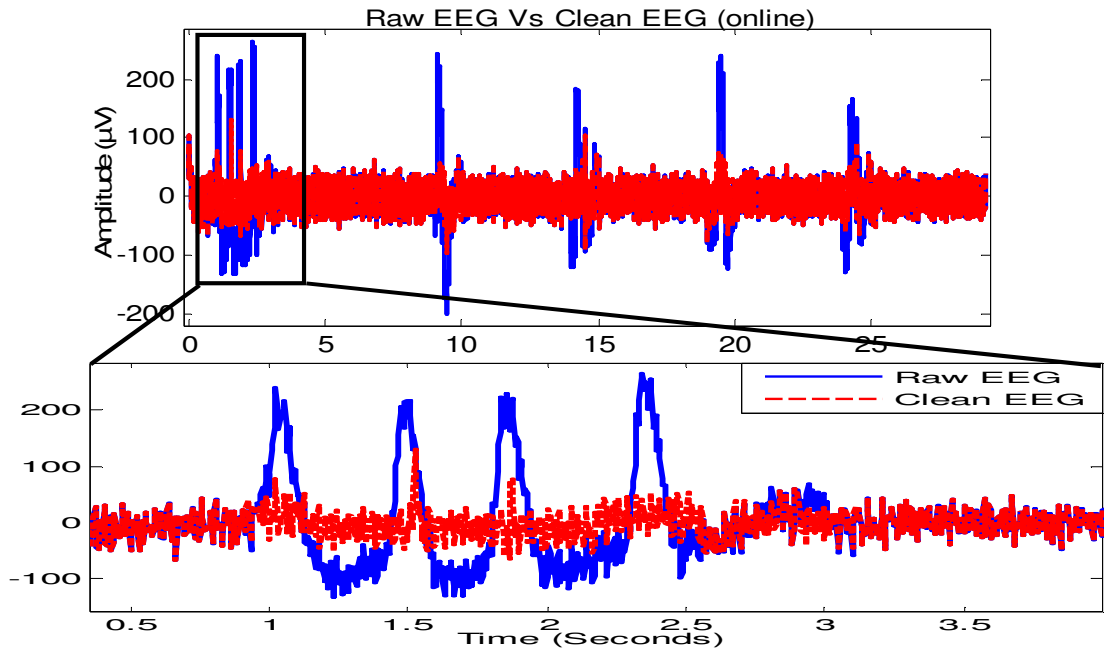


Fig.43: Online mode – MATLAB based hybrid Algorithm result (channel-1, subject-3)

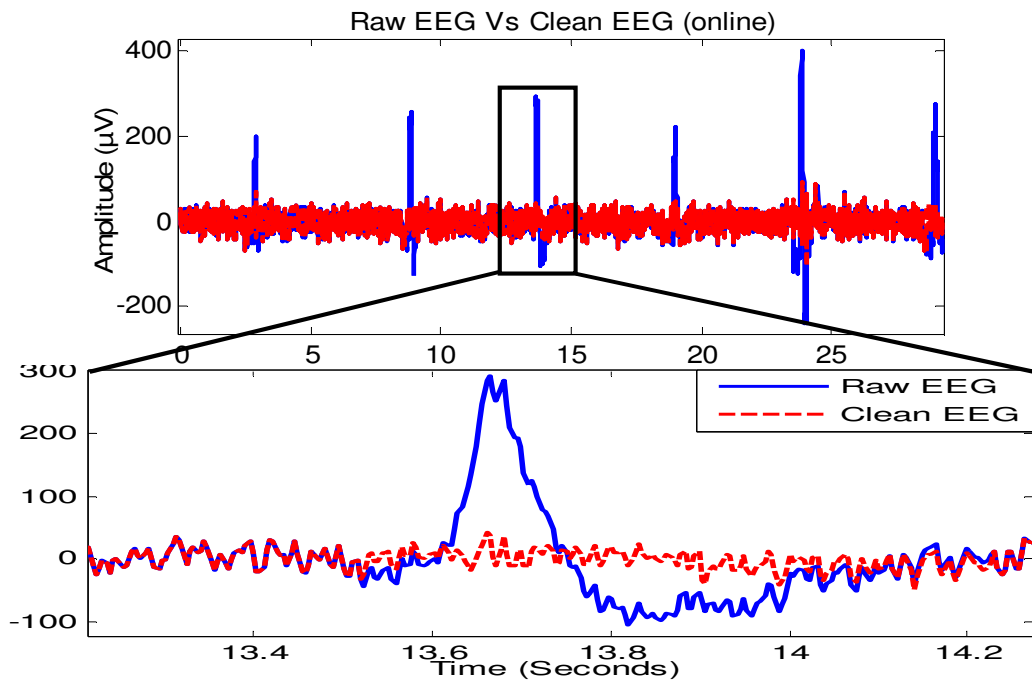


Fig.44: Online mode – MATLAB based hybrid Algorithm result (channel-1, subject-4)

As shown in Fig.43, the subject-3 blinked consecutively four times in 1sec to 2.5 sec duration. It was observed that even for such immediate occurrences of eye blinks in real-time, the hybrid OA removal algorithm, worked accurately for OA detection as well as removal of the exact OA zones in EEG. Again the non-OA zone remained unchanged and retained original EEG information for both subjects 3 and 4.

5.1.3 Online C based algorithm

This result category belongs to when hybrid OA removal algorithm was implemented on PSoC-3 MCU mounted on NM EEG device and evaluated in real-time. The performance of the C based hybrid algorithm implemented on MCU in real-time was tested using total three real-time EEG acquisitions from single subject. Out of them for 2 datasets, the overall hybrid OA removal algorithm performance results in real-time are shown in Fig.45 (a) and Fig.46 (a). The results in Fig 45 (b) and Fig 46 (b) illustrates and maps the position of the detected OA zone on actual EEG signal for verification.

For the simplicity in algorithm to be implemented on PSoC-3 MCU, OA removal technique used DWT decomposition with the very simplest wavelet function *haar*. The real-time algorithm implemented ring buffer (refer section 4.4.2.2) to perform overlapping method along with hybrid OA detection and removal algorithm and Fig.45 (a) and Fig.46(a) reveals that the microcontroller hardware implementation of the same, worked efficiently. Also Fig 45(b) and Fig.45 (b) clearly indicates that the OA detection algorithm accurately detected and removed the eye blinks in real-time.

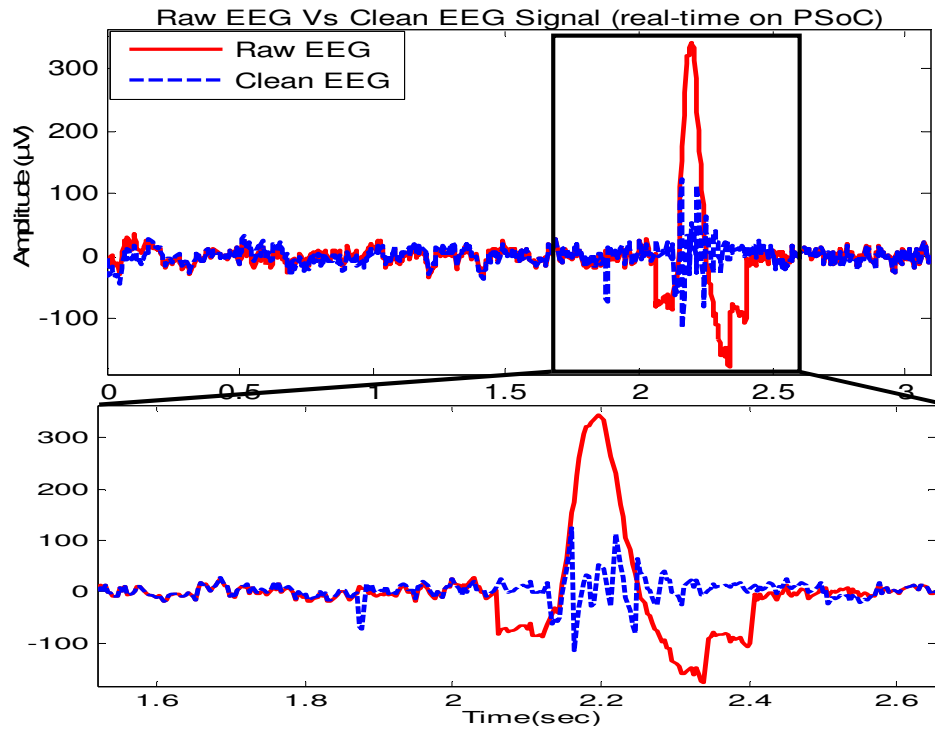


Fig.45: (a) Online mode – C based hybrid Algorithm result (FP1 – dataset2)

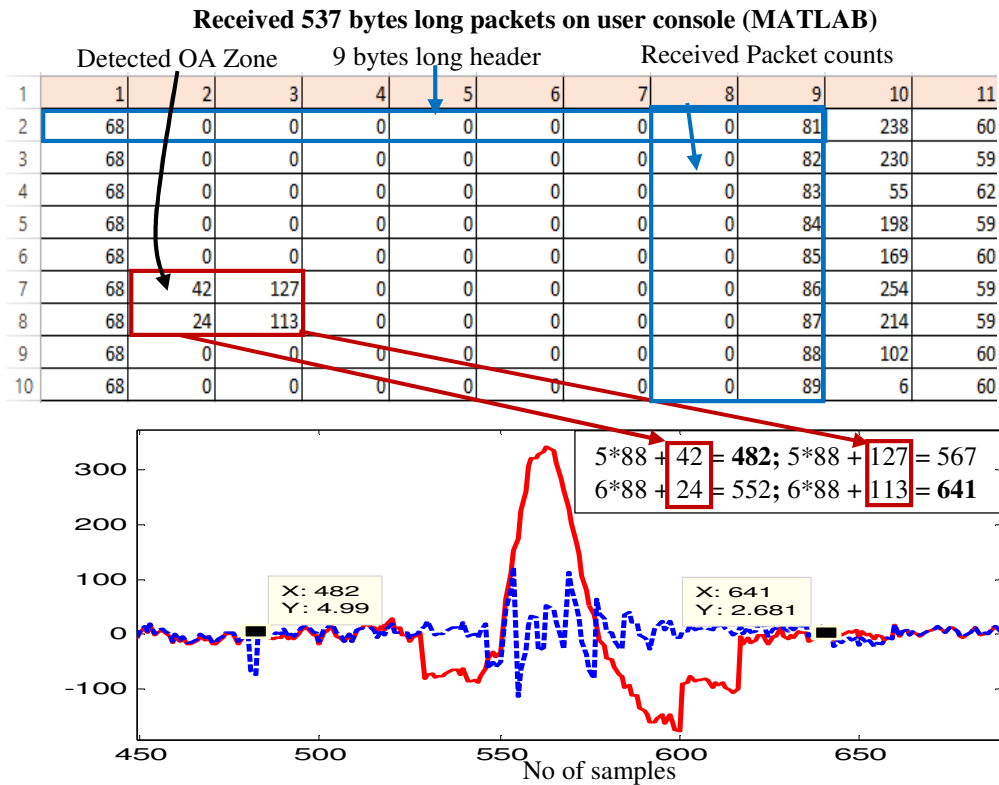


Fig.45: (b) Verification of OA zone detection C based algorithm in real-time (dataset2)

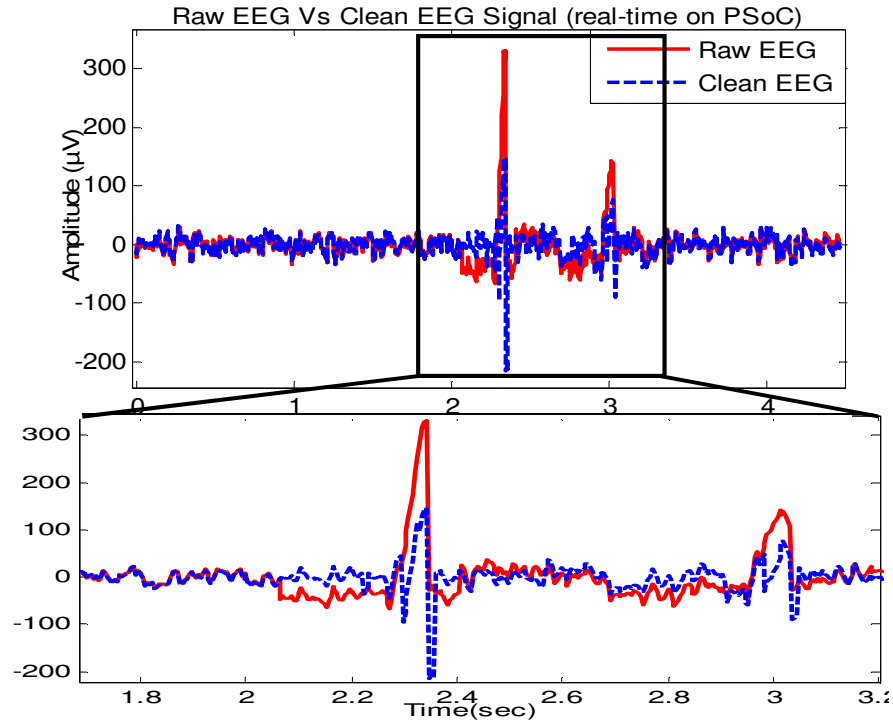


Fig.46: (a) Online mode – C based hybrid Algorithm result (FP1 – dataset3)

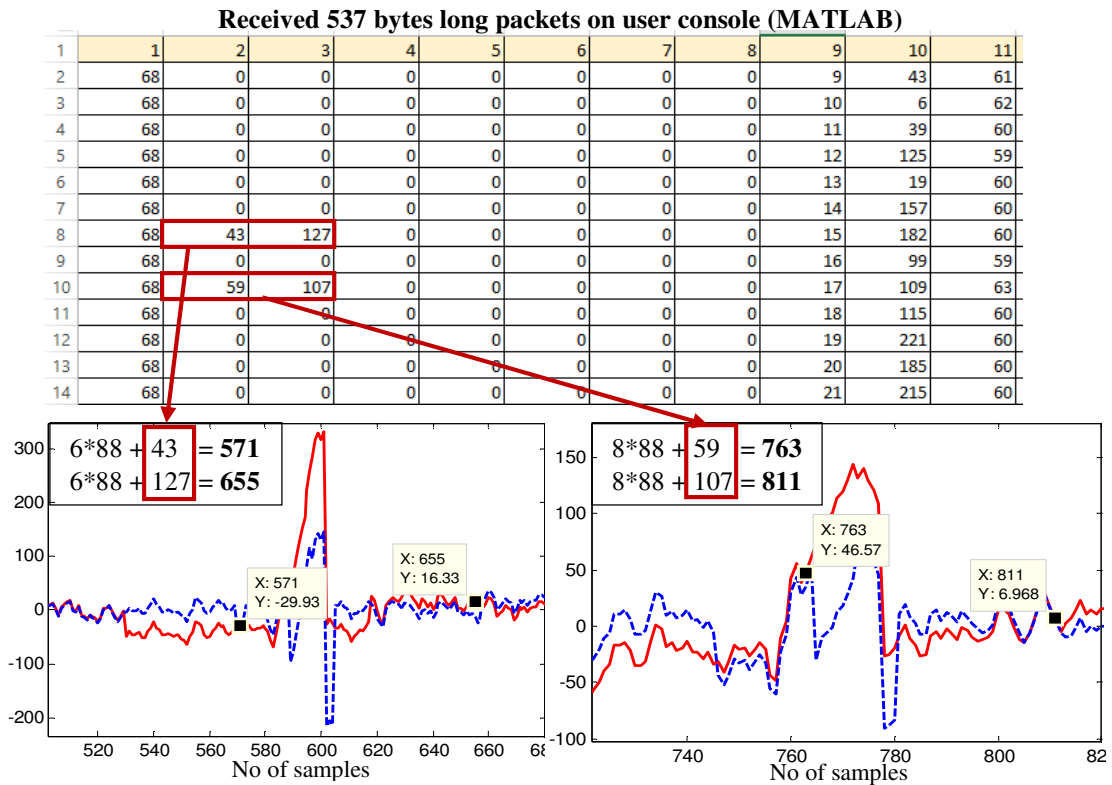


Fig.46: (b) Verification of OA zone detection C based algorithm in real-time (dataset3)

5.2 Performance Evaluation

For the validation of the developed hybrid OA removal technique, Time-Frequency Analysis (TFA), Magnitude Square Coherence (MSC) plot and two statistical parameters: Correlation of Coefficient (CC) and Mutual Information (MI), are utilized.

5.2.1 Performance Metrics

- **Time-Frequency analysis:**

The EEG being non-stationary signal, the wavelet based time-frequency analysis is one of the most suitable methods to analyze EEG. It provides the information of energy of the frequencies existing at a given time simultaneously. The TFA is carried out here using EEGLAB [24] function.

- **Magnitude Square Coherence:**

MSC plot is generated using the MATLAB function '*mscohere*'. MSC gives the estimate of the frequency coherence between the two signals x and y, where values between 0 and 1 indicates how well signal x corresponds to y at each frequency. The MSC is a function of the power spectral densities, $P_{xx}(f)$ and $P_{yy}(f)$, of x and y, and the cross power spectral density, $P_{xy}(f)$, of x and y as given in (12).

$$c_{xy}(f) = \frac{|P_{xy}(f)|^2}{P_{xx}(f)P_{yy}(f)} \quad (12)$$

Power spectral densities are calculated over FFT length of 256 with 50% overlapped Hamming window using MATLAB *mscohere* function.

- **Correlation of Coefficient:**

CC computes the similarity between the raw and corrected EEG signals.

MATLAB (MathWorks Inc., Natwick, MA) function '*corrcoef*' is used to determine it

and result ranges between 0 being *no match at all* and 1 being *exact match*. CC is separately computed for non-OA zone and OA zone to clearly show the correlation in respective areas.

- **Mutual Information:**

MI measures how much one random variables tells us about another. The higher the value of MI metric, the better the mutual information content. The open source MATLAB function *minfo.m* developed by Dr.Jason Palmer [online] (available at: <http://scn.ucsd.edu/~jason/minfo.m>) is used to compute MI.

5.2.2 Performance Evaluation Results

5.2.2.1 Offline MATLAB based algorithm

1. Time – Frequency Analysis

The TFA was carried out for total four subjects where for subject-1 (Emotive device) TFA plot was generated for Raw Vs Clean EEG for the epoch length of 6sec to 68sec whereas for remaining three (NM device) subjects, it was 5sec to 34sec as shown in Fig.47 to Fig.50 respectively.

- **Dataset-1 (Subject-1: single channel)**

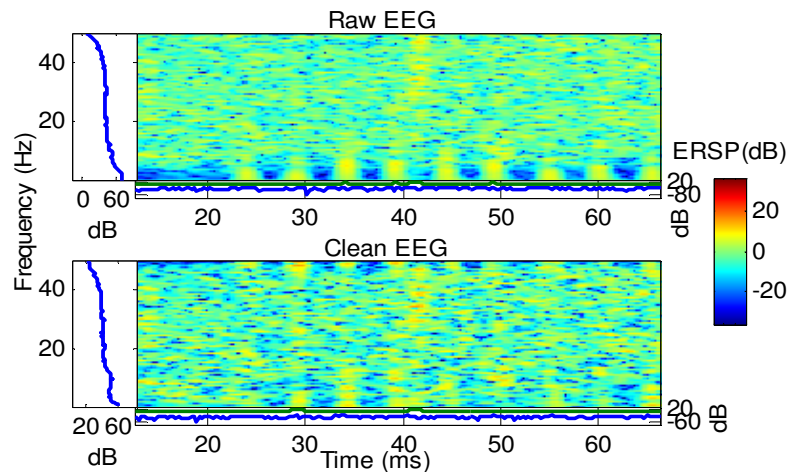


Fig.47: Time-Frequency Analysis plot for subject-1 EEG data

- **Dataset-2 (Subject-2: two channels)**

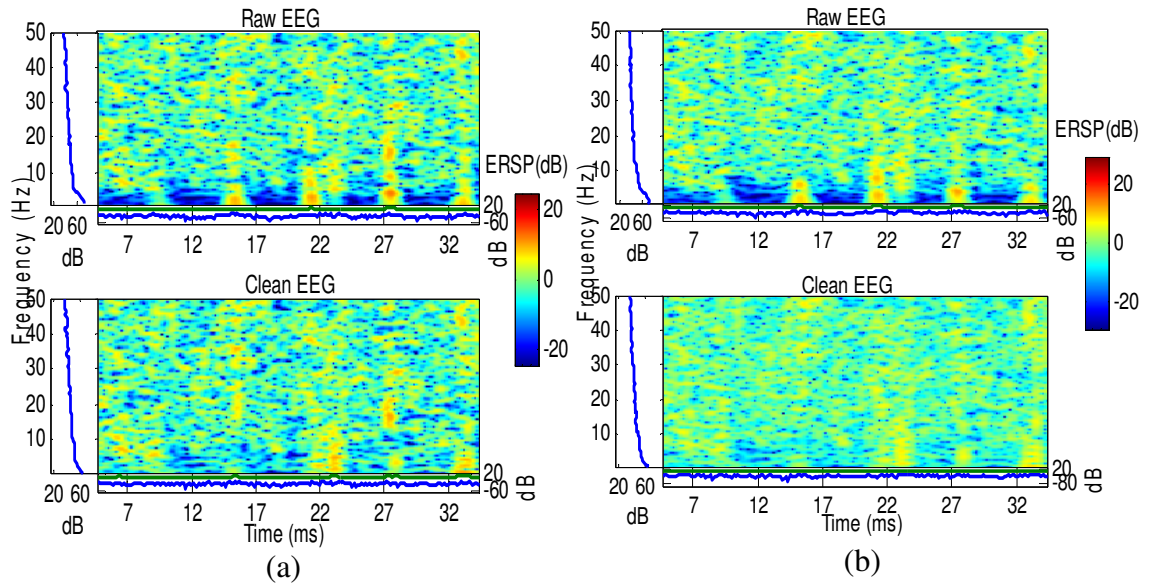


Fig.48: Time-Frequency Analysis plot for subject-2 EEG data
 (a) Channel-1 (FP1) (b) Channel-2 (FP2)

- **Dataset-3 (Subject-3: two channels)**

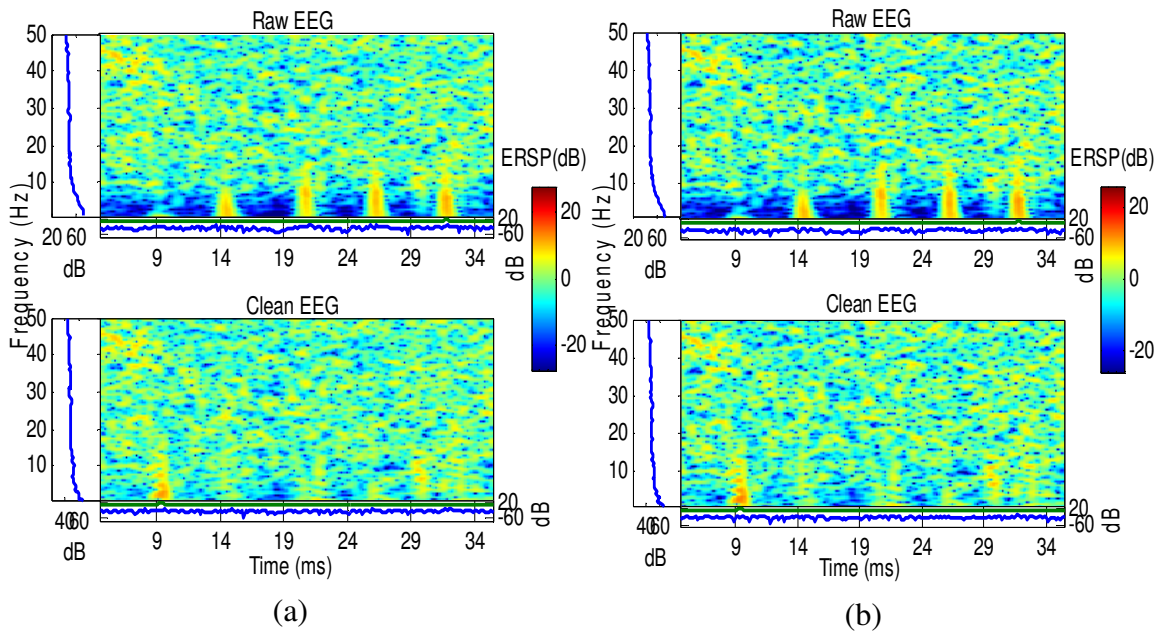


Fig.49: Time-Frequency Analysis plot for subject-3 EEG data
 (a) Channel-1 (FP1) (b) Channel-2 (FP2)

- **Dataset-4 (Subject-4: two channels)**

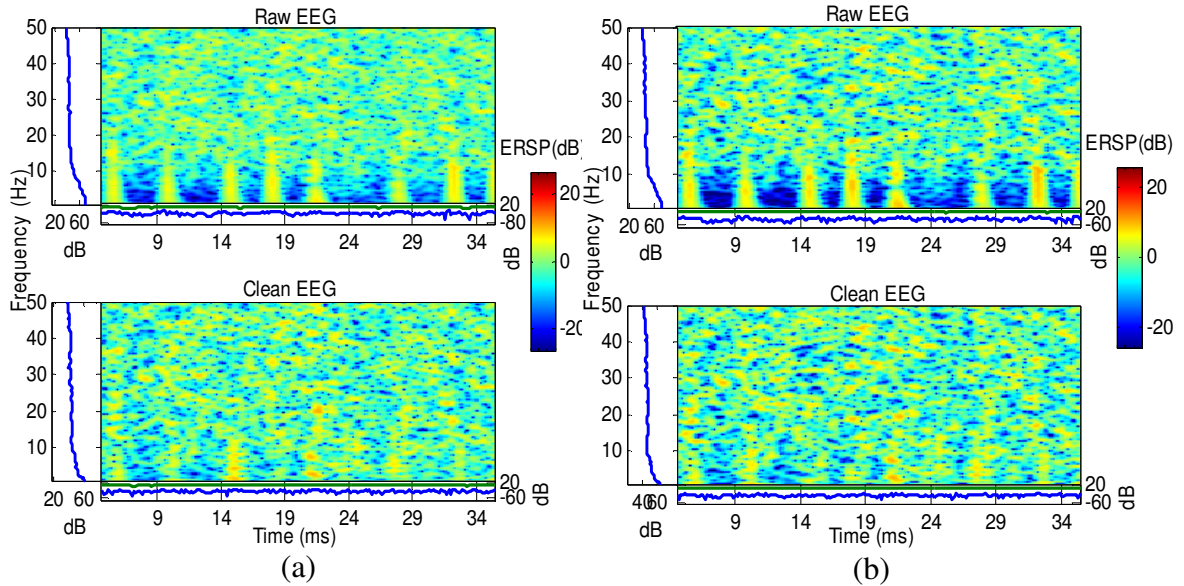


Fig.50: Time-Frequency Analysis plot for subject-4 EEG data
 (a) Channel-1 (FP1) (b) Channel-2 (FP2)

From all the TFA plots, it is evident that the energy level of low frequency components in EEG was significantly dampened in eye blink zones whereas high frequency component energy retained its values. These results of TFA shows that de-noising of EEG by suppressing OA from raw EEG worked as anticipated.

2. Mean-Squared Coherence

Next performance evaluation considered was MSC between raw and clean EEG. For the lower frequency components where OA resides should show less coherence whereas higher than 16 Hz frequency components should show maximum coherence with each other as those frequency components are not taken into account while de-noising the raw EEG. The results of MSC plots for the frequencies between 0 – 40 Hz are shown in Fig.51 to Fig.54.

- **Dataset-1 (Subject-1: single channel)**

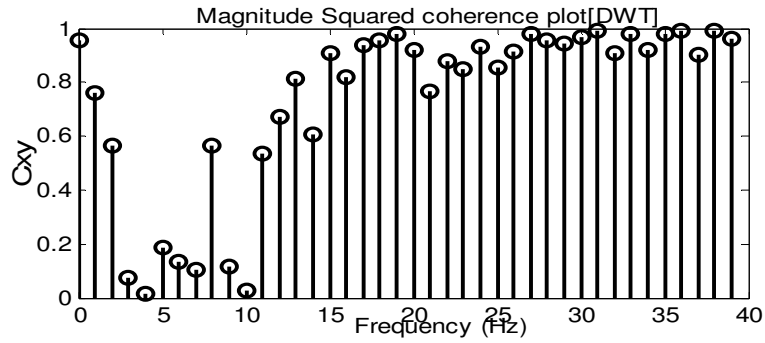


Fig.51: MSC plot for raw Vs clean EEG (subject-1)

- **Dataset-2 (Subject-2: two channels)**

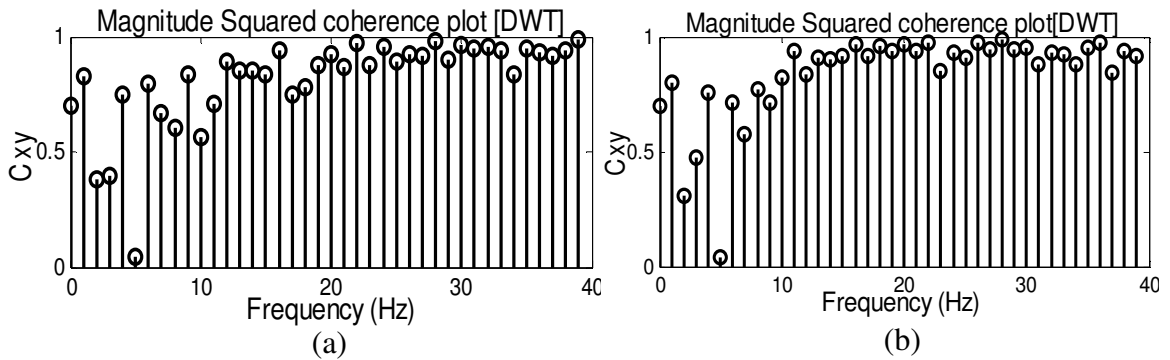


Fig.52: MSC plot for raw Vs clean EEG (subject-2)
(a) Channel-1 (FP1) (b) Channel-2 (FP2)

- **Dataset-3 (Subject-3: two channels)**

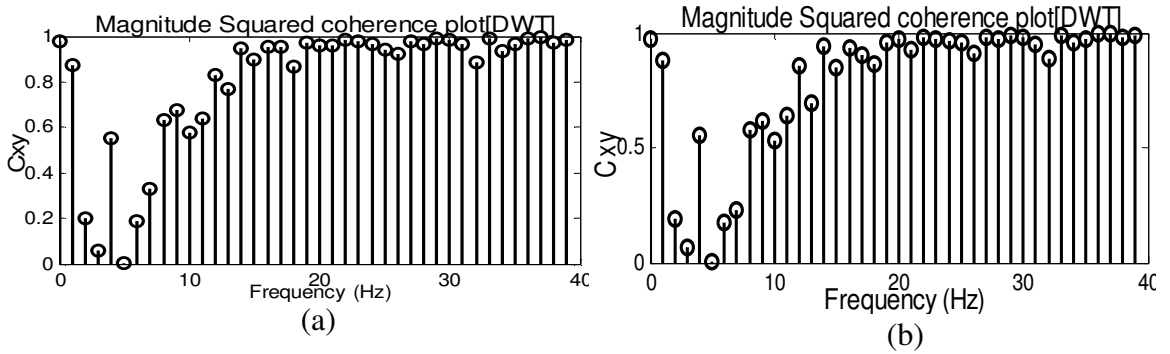


Fig.53: MSC plot for raw Vs clean EEG (subject-3)
(a) Channel-1 (FP1) (b) Channel-2 (FP2)

- **Dataset-4 (Subject-4: two channels)**

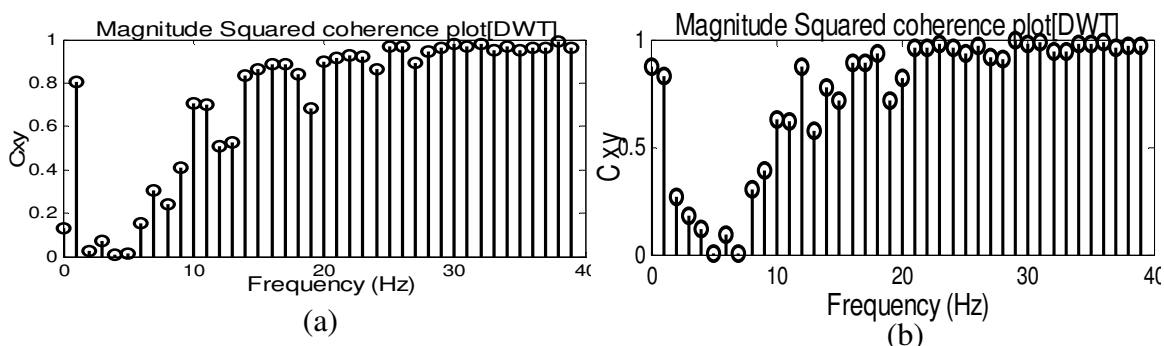


Fig.54: MSC plot for raw Vs clean EEG (subject-4)
 (a) Channel-1 (FP1) (b) Channel-2 (FP2)

The coherence between the frequencies higher than 16 Hz should be 1 as DWT based de-noising in OA zone was carried out only for the frequency range of 0.5-16 Hz. But as observed in MSC plots, the frequencies higher than the 16 Hz also are getting modified. This variances of higher than 16 Hz frequencies were affected differently with different wavelet functions used and this requires further investigation. Although, the less coherence values in lower frequency range 0.5-16 Hz shows significant removal of artifacts.

3. Coefficients of Correlation & Mutual Information

Table.3: PERFORMANCE METRICS FOR CC AND MI FOR OFFLINE MODE (MATLAB BASED)

Subject		Correlation of Coefficient									MI
		Blink 1	Blink 2	Blink 3	Blink 4	Blink 5	Blink 6	Blink 7	Blink 8	Blink 9	
1	FP1	0.1459	0.2493	0.3469	0.2562	0.2491	0.1609	0.2203	0.1543	0.1483	1.4973
2	FP1	0.1798	0.670	0.2813	0.314	0.5962	NA				1.1399
	FP2	-0.054	0.7940	0.2981	0.4118	0.6230	NA				1.1224
3	FP1	0.0396	0.2118	0.2576	0.1691	0.2443	0.3351	NA			0.9621
	FP2	0.0480	0.1953	0.2442	0.1581	0.2039	0.2876	NA			0.9520
4	FP1	0.7690	0.3036	0.3735	0.6040	0.3394	0.5683	0.3994	0.1601	0.2672	1.2530
	FP2	0.2086	0.3145	0.3846	0.6718	0.1640	0.5218	0.363	0.1668	0.2977	1.0545

Table.3 shows the CC and MI for each datasets where CC has been tabulated for the each OA zone (blink) present in the respective EEG signal whereas MI calculated over the entire dataset. For calculating CC for every blink, OA zones were selected manually with visual inspection. For subject-1 only single channel EEG evaluation was carried out whereas for all three other subjects CC and MI for both channels are mentioned. Due to the manual selection of OA zone for computing CC, there might be some variation in the values but overall it clearly indicates that OA zones were satisfactorily removed from contaminated EEG to obtain clean EEG. It is to be noted that CC and MSC for all of the non-OA zones were '1', which indicated that the entire EEG information in terms of amplitude and frequency both, in non-OA zone were intact and completely preserved.

5.2.2.2 Online C based algorithm

To verify the performance of online C based algorithm, only CC was computed for OA and non-OA zone for all existing eye blinks in the all the three datasets and the results are tabulated below:

Table.4: PERFORMANCE METRICS FOR CC FOR ONLINE MODE (C BASED)

Dataset	Zone	Correlation of Coefficient	
		Blink 1	Blink 2
1	OA zone	0.2476	NA
	Non- OA	0.9981	
2	OA zone	0.1255	NA
	Non- OA	0.9459	
3	OA zone	0.6303	0.6096
	Non- OA	0.9839	

From the lesser values of CC in the OA zone in the above table, it is evident that the EEG got de-noised by suppressing the eye blink spikes. Whereas it was also observed that like in all other cases, non-OA zone CC was not ‘1’. The reason for this is mentioned before that there exist little discrepancy in both received raw EEG and cleaned EEG in real-time because of the mean adjustment calculation difference. But even then the CC values for non-OA was obtained nearly equal to ‘1’ which indicated that both the raw and clean EEG signals differ insignificantly and thus preserves the critical neural information in that region.

Finally, the Table.5 lists the total flash memory and RAM utilized by the Microcontroller implemented proposed OA removal algorithm. From the Table.5 it is clear that the entire code implemented on PSoC-3 has not utilized more than 30% of the flash memory and still there exist significant space for any additional algorithms such as for feature extraction or characterization to be implemented along with the proposed hybrid OA removal algorithm to make the EEG data capturing and its analysis entirely in real-time and micro-controller hardware implementable.

Table.5: MICROCONTROLLER UNIT MEMORY USAGE SUMMARY

PSoC-3 CY8C38 family	8-bit 8051 CPU
RAM	8 KB
Flash memory	64KB
Used RAM	67.9% (5.56KB)
Used Flash memory	28.1% (18.1KB)

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

In this thesis, a hybrid algorithm to remove OA from single channel EEG data is proposed which comprised of algebraic method based OA detection, followed by DWT decomposition based OA removal technique. DWT was chosen over SWT mainly for its faster operational speed. The developed algorithm was tested in real-time settings using MATLAB based algorithm running on user console as well as C based algorithm running on actual hardware of PSoC-3 MCU.

The chapter 2 described the various existing OA removal techniques suitable for single EEG channel such as different WT based methods, EMD-CCA based algorithm and Algebraic method to detect the irregularity in the signal (eye blink in EEG). After the review, in order to have the algorithm to be real-time applicable and the hardware implementable, the methods using less computational complexities were chosen. Accordingly, the Algebraic method to detect the OA and DWT based de-noising technique was finalized for hybrid OA removal algorithm.

In chapter 3, basic theory for each method, i.e. Algebraic and DWT were covered up to give sufficient mathematical background. The algebraic method basically used FIR filter to get the coefficients to be compared with the threshold value to detect the change point in the considered small interval signal. DWT based decomposition was carried out up to level-8 and detail coefficients from frequency range of 0.5-16 Hz were suppressed if they exceeded the threshold value to get the de-noised EEG signal by reconstructing using inverse DWT. The chapter 3 also illustrated the actual implementation method

where, algebraic method applied to detect the OA zone and DWT based de-noising was applied ONLY to the detected eye blink zones to preserve the low frequency components of original EEG in non-OA zone.

The chapter 4 was the core content chapter describing every method in detail with respective results. It showed the relevant results to demonstrate that the developed algorithm worked efficiently in both offline and online modes using MATLAB and C based algorithm strategies. In this chapter it was also clearly indicated that how the overlapping method improved the real-time functionality of the hybrid algorithm. The key conclusion which can be derived from chapter 4 is that the implemented hybrid algorithm was suitable for single channel EEG as well as microcontroller executable in online mode. For the MCU based algorithm implementation, ring buffer technique was introduced to realize overlapping of 0.5 sec EEG epochs in real-time.

In the chapter 5, all achieved results in offline and online modes using MATLAB and C based code were clearly shown. It was concluded that the algorithm was efficient enough to detect and remove OA for consecutively occurring eye blinks and online microcontroller hardware executable algorithm written in C demonstrated successful OA detection and respective de-noising in real-time. The second part of the chapter 5, was dedicated to mention the performance metrics for the implemented hybrid OA removal algorithm. The Correlation of Coefficient and Magnitude Squared Coherence plot indicate that the raw EEG signal completely matches with the corrected EEG signal in non-OA zone. This concludes that the algorithm output, i.e. de-noised EEG, does not impact the useful EEG information in non-OA zone by retaining its all values as original raw

EEG. In the OA zone, the neuronal information were retained while artifacts were significantly suppressed. This shows effectiveness of applying WT de-noising only to the Eye blink zone rather than entire signal. The TFA plots and MI values derived the conclusion about successful suppression of lower frequency components in OA zones and sufficient mutual information between the raw and clean EEG respectively. Also the MSC variation for the frequencies higher than 16 Hz was required to be investigated.

Thus, the thesis work presented here concludes that the Implemented hybrid OA removal algorithm was:

- Suitable for single EEG channel
- Online Applicable by processing as small as 0.5 second EEG epoch in real-time
- Found accurate detection of eye blinks without missing them at the boundary of the epoch because of the epoch overlapping technique addition.
- Capable of preserving important background EEG information in non-OA zones
- Successfully implemented on microcontroller hardware to achieve satisfactory OA detection and removal results in real-time settings.
- This is probably the first work done to detect and then remove OA from single channel EEG in real-time implemented on microcontroller unit on actual EEG device.

Moreover, this hybrid approach can also be applied for any number of multichannel EEG systems making it versatile in nature.

6.2 Future Scope

The determination of threshold function was found very challenging in OA detection and WT based de-noising technique. The OA removal algorithm developed can be made robust and device independent by investigating more on best suitable threshold function for the given objective. The microcontroller implemented hybrid algorithm was verified using the simplest wavelet function, *haar*. The algorithm further can be furnished to be efficient using more relevant wavelet function for proper OA removal from EEG.

The algorithm was implemented on NM device which uses BT to transmit EEG wirelessly. The algorithm can be upgraded for more sophisticated network communication channels. This can lead to more promising algorithm application area in ‘Wearable technologies’ and ‘Internet of Things’. The MCU implemented OA detected algorithm in real-time can be further optimized to run faster in nearly real-time by implementing lower order FIR filter. Also to optimize the MCU hardware implemented DWT based de-noising algorithm, instead of implemented recursive low pass and high pass filtering, sequential and parallel DWT techniques can be utilized to increase efficiency in terms of computation requirements, storage requirements and reconstructed signal with better signal-to-ratio [30].

Finally, OA removal algorithm can further be developed to make it generic for other artifacts removal generated in EEG, such as artifacts due to ECG, muscle movements, etc. such that EEG signal feature extraction and characterization can be implemented along with the proposed algorithm for entirely automatic EEG analysis and processing in real-time using single channel EEG on microcontroller hardware.

REFERENCES

- [1] M. Teplan, "Fundamentals of EEG measurement," *Measurement Science Review*, vol. 2, pp. 1-11, 2002.
- [2] C. A. Joyce, I. F. Gorodnitsky and M. Kutas, "Automatic removal of eye movement and blink artifacts from EEG data using blind component separation," *Psychophysiology*, vol. 41, pp. 313-325, 2004.
- [3] R. Lloyd, R. Goulding, P. Filan and G. Boylan, "Overcoming the practical challenges of electroencephalography for very preterm infants in the neonatal intensive care unit," *Acta Paediatrica*, vol. 104, pp. 152-157, 2015.
- [4] Z. Tiganj, M. Mboup, C. Pouzat and L. Belkoura, "An algebraic method for eye blink artifacts detection in single channel EEG recordings," in *17th International Conference on Biomagnetism Advances in Biomagnetism–Biomag2010*, 2010, pp. 175-178.
- [5] S. V. Ramanan, N. Kalpakam and J. Sahambi, "A novel wavelet based technique for detection and de-noising of ocular artifact in normal and epileptic electroencephalogram," *Brain Inspired Cognitive Systems conference*, 2004.
- [6] V. Krishnaveni, S. Jayaraman, L. Anitha and K. Ramadoss, "Removal of ocular artifacts from EEG using adaptive thresholding of wavelet coefficients," *Journal of Neural Engineering*, vol. 3, pp. 338, 2006.
- [7] P. S. Kumar, R. Arumuganathan, K. Sivakumar and C. Vimal, "A wavelet based statistical method for de-noising of ocular artifacts in EEG signals," *International Journal of Computer Science and Network Security*, vol. 8, pp. 87-92, 2008.
- [8] M. H. Soomro, N. Badruddin, M. Z. Yusoff and M. A. Jatoi, "Automatic eye-blink artifact removal method based on EMD-CCA," in *Complex Medical Engineering (CME), 2013 ICME International Conference on*, 2013, pp. 186-190.
- [9] K. T. Sweeney, S. F. McLoone and T. E. Ward, "The use of ensemble empirical mode decomposition with canonical correlation analysis as a novel artifact removal technique," *Biomedical Engineering, IEEE Transactions on*, vol. 60, pp. 97-105, 2013.
- [10] H. T. Nguyen, J. Musson, F. Li, W. Wang, G. Zhang, R. Xu, C. Richey, T. Schnell, F. D. McKenzie and J. Li, "EOG artifact removal using a wavelet neural network [WNN]." *Neurocomputing*, vol. 97, pp. 374-389, 2012.
- [11] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, pp. 239-255, 2010.

- [12] H. Peng, B. Hu, Q. Shi, M. Ratcliffe, Q. Zhao, Y. Qi and G. Gao, "Removal of Ocular Artifacts in EEG—an Improved Approach Combining DWT and ANC for Portable Applications," *Biomedical and Health Informatics, IEEE Journal of*, vol. 17, pp. 600-607, 2013.
- [13] Ruhi Mahajan, B. I. Morshed, "Unsupervised Eye Blink Artifact Denoising in EEG data with Modified Multiscale Sample Entropy, Kurtosis and Wavelet-ICA", *IEEE Journal of Biomedical and Health Informatics*, vol. 19 (1), pp. 158-165, Jan. 2015.
- [14] A. Joseph and G. Titus, "Removal of blink artifacts from EEG: Performance comparison of wavelet transform and empirical mode decomposition," in *International Journal of Engineering Research and Technology*, 2014.
- [15] P. Balaiah and I. Ilavennila, "Comparative evaluation of adaptive filter and neuro-fuzzy filter in artifacts removal from electroencephalogram signal," *American Journal of Applied Sciences*, vol. 9, 2012.
- [16] M. Mboup, "A Volterra filter for neuronal spike detection *Research report, INRIA, <http://hal.inria.fr/inria-00347048/fr/>*." 2008.
- [17] M. Fliess and H. Sira-Ramirez, "An algebraic framework for linear identification," *ESAIM Controle Optimisation Et Calcul Des Variations*, vol. 9, pp. 151, 2004.
- [18] N. Debbabi, M. Kratz, M. Mboup and S. El Asmi, "Combining algebraic approach with extreme value theory for spike detection," in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, 2012, pp. 1836-1840.
- [19] H. T. Gorji, A. Koohpayezadeh and J. Haddadnia, "Ocular Artifact Detection and Removing from EEG by wavelet families: A Comparative Study," *Journal of Information Engineering and Applications*, vol. 3, pp. 39-47, 2013.
- [20] M. O. Oliveira and A. S. Bretas, "Application of discrete wavelet transform for differential protection of power transformers," in *PowerTech, 2009 IEEE Bucharest*, 2009, pp. 1-8.
- [21] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic press, 1999.
- [22] M. Haak, S. Bos, S. Panic and L. Rothkrantz, "Detecting stress using eye blinks and brain activity from EEG signals," *Proceeding of the 1st Driver Car Interaction and Interface (DCII 2008)*, 2009.
- [23] W. Xiaohua and H. Yigang, "Design of complex FIR filters with arbitrary magnitude and group delay responses," *Systems Engineering and Electronics, Journal of*, vol. 20, pp. 942-947, 2009.

- [24] Delorme A and Makeig S, "EEGLAB: An open source toolbox for analysis of single-trial EEG dynamics," *J Neuroscience Methods*, vol.134, pp.9-21, 2004.
- [25] R. Mahajan, C. A. Majmudar, S. Khatun and B. I. Morshed, "NeuroMonitor Ambulatory EEG Device: Comparative Analysis and Its Application for Cognitive Load Assessment," *Healthcare Innovation Conference (HIC)*, pp. 133-136, 2014.
- [26] S. Consul-Pacareu, R. Mahajan, N. Sahadat and B. I. Morshed, "Wearable Ambulatory 2-Channel EEG NeuroMonitor Platform for Real-Life Engagement Monitoring Based on Brain Activities at the Prefrontal Cortex," *Proceedings of The 2014 IAJC-ISAM International Conference*, 2014.
- [27] R. Mahajan, S. Consul-Pacareu, M. Abusaud, M. N. Sahadat and B. I. Morshed, "Ambulatory EEG NeuroMonitor platform for engagement studies of children with development delays," in *SPIE Defense, Security, and Sensing*, 2013, pp. 87190L-87190L-10.
- [28] S. A. Hosseini and M. B. Naghibi-Sistani, *Classification of Emotional Stress using Brain Activity*. INTECH Open Access Publisher, 2011.
- [29] J. S. Walker, "A primer on wavelets and their scientific applications. 1999," *Chapman&Hall/CRC, New York*,.
- [30] K. Shukla and A. K. Tiwari, *Efficient Algorithms for Discrete Wavelet Transform: With Applications to Denoising and Fuzzy Inference Systems*. Springer Science & Business Media, 2013.

APPENDICES

A: OFFLINE MATLAB BASED OA REMOVAL CODE

```
% Offline mode: MATLAB based hybrid OA detection and removal algorithm
% Considering 128 samples EEG epoch
% WT denoising applied using DWT; level = 8
% By: Charvi Majmudar: 01/25/2015 (Thesis work)

clc; close all; clear all;

%%===== Offline mode EEG Data reading =====
%% EMOTIVE Device datasets -----
filename = 'Fp1.csv'; % Dataset-1 (subject1)
% filename = 'Fp1_1.csv'; % Dataset-2 (subject2)
% filename = 'ManNorm.csv'; % Dataset-3 (subject3)
y2 = csvread(filename);
y2 = y2(1:8961,1);

%% Neuromonitor (NM) Datasets -----
% a1 = csvread('final_value.csv'); % Dataset-4 (subject4)
% channel1= a1(:,1);
% channel2= a1(:,2);
% Ch1_offclean= mean(channel1)-channel1; % - ve because Inverting
% opamp %output
% Ch2_offclean=-(channel2- mean(channel2));
% y2 = Ch2_offclean;
% y2 = y2(513:end); % ignoring inital 2sec data
%%=====

y1 = y2;
y_swt = y2;

%% intialization .....
T = 0.29; % Signal epoch
Ts = 0.0078; %sampling rate 128sps
% Ts = 0.0039; %sampling rate 256sps
M = round(T/Ts);
k = [0 1 2 3];
v = 3; % v >=2
m = 1:1:M+1;
tm = Ts.* m;
K = 2;
Wm = ones(1,length(m));
N = 128;
Delay = ceil(M/2); % filter order/2 appx
wName='bior4.4'; %coif3 coif5 sym3 db5
level = 8;
CleanEEG = [];
CleanEEG_swt = [];
Overlap = 40;
```

```

for iter = 1:100           % iteration for every 0.5 sec epoch to process
                           % entire % EEG signal
y = y1(1:N);             % 128 samples epoch
n = (length(y)- M);     % no. of sliding windows
vkn = zeros(M,n,length(k));
array = [];
y3 = zeros(2^level,1);
y4 = zeros(2^level,1);
f = zeros(M,n,K);
f1 = zeros(M,n);
f2sec_epoch = zeros(length(y),1);

%% calculating impulse response (hk)-----
g0 = Wm.*((-1)^(k(1)+1))/factorial(v-1).*(4.*tm.*(2.*tm - 2) + 2.*(tm -
1).^2 + 2.*tm.^2);

g1 = Wm.*((-1)^(k(2)+1))/factorial(v-1).*(-12.*tm.*(tm - 1).^2 - 2.*(tm
- 1).^3 - 3.*tm.^2.*(2.*tm - 2));

g2 = Wm.*((-1)^(k(3)+1))/factorial(v-1).*(16.*tm.*(tm - 1).^3 + 2.*(tm
- 1).^4 + 12.*tm.^2.*(tm - 1).^2);

g3 = Wm.*((-1)^(k(4)+1))/factorial(v-1).*(-20.*tm.*(tm - 1).^4 - 2.*(tm
- 1).^5 - 20.*tm.^2.*(tm - 1).^3);

%% FIR filter -----
for j = 1:n
    vkn(:,j,1) = filter(g0, 1, y(j:(j+M-1)));    %k = 0
    vkn(:,j,2) = filter(g1, 1, y(j:(j+M-1)));    %k = 1
    vkn(:,j,3) = filter(g2, 1, y(j:(j+M-1)));    %k = 2
    vkn(:,j,4) = filter(g3, 1, y(j:(j+M-1)));    %k = 3
end

%% Volterra Filter for decision function calculation -----
for s = 1:K
    for r = 1:n
        f(:,r,s) = (vkn(:,r,s+1).^2) - (vkn(:,r,s).* vkn(:,r,s+2));
        f(:,r,s) = (max(0,f(:,r,s))/1000000);
    end
end
for d = 1:n
    f1(:,d) = f(:,d,1).* f(:,d,2);
    f2sec_epoch(d:d+M-1) = f2sec_epoch(d:d+M-1) + f1(:,d);
end

%% Threshold Function -----
Gamma1 = 0.001/(mean(f2sec_epoch)+ std(f2sec_epoch));

Threshold = (f2sec_epoch > Gamma1);
Threshold = Threshold';
%% OA Zone detection -----
ind = find(Threshold);
ii = 1;
while(ii <= (length(ind)-2))
    temp = ind(ii);

```



```

    while((ii < length(ind)) && ((ind(ii+1) - ind(ii)) == 1))
        ii = ii+1;
    end
    Edge_start = max(1, temp - Delay -10); % to avoid -ve index...so
                                         min. it % can take is '1'
    Edge_end = min(numel(y),ind(ii)); % Max 128 can be the OA index
                                         for % the given epoch

    array = [array; Edge_start; Edge_end];
    ii = ii+1;
end

%% Wavelet Transform Denoising .....
%% DWT
len1 = 0;
if(numel(array) ~= 0)
    for den = 1:2:length(array)
        y3(1:array(den+1)-array(den)+1) = y1(array(den):array(den+1));

        [C,L] = wavedec(y3,level,wName); % WT decomposition levle=8

        for iL = 2:1:6 % Detail coefficients from level 4 to 8
            len1 = len1 + L(iL-1,:);
            Decoef = C(len1+1:len1+L(iL,:)); % detail coefficients
            G1 = mad(Decoef,1)*1.5; % Threshold function

% threshold based denoising.....
            for u = 1:size(Decoef)
                if(Decoef(u) > G1)
                    Decoef(u) = 0;
                elseif(Decoef(u) < -G1)
                    Decoef(u) = 0;
                else
                    Decoef(u) = Decoef(u);
                end
            end
            C(len1+1:len1+L(iL,:)) = Decoef;
        end
        A0 = waverec(C,L,wName); % Reconstructing signal
        y1(array(den):array(den+1)) = A0(1:array(den+1)-array(den)+1);
        % replacing with original signal
    end
end

%----- Steps performed for Overlapping method -----
CleanEEG = [CleanEEG; y1(1:N-Overlap)];
y1(1:N-Overlap) = [];

%% SWT de-noising used for comparison .....
if(numel(array) ~= 0)
    for den = 1:2:length(array)
        y4(1:array(den+1)-array(den)+1) = y_swt(array(den):array(den+1));
        tic;
        [swa,swd] = swt(y4,level,wName);
        for iL = 4:1:level
            [rw,clm] = size(swd(iL,:));

            G2 = mad(swd(iL,:),1)*1.5;
        end
    end
end

```

```

        for u = 1:clm
            if(swd(iL,u) >= G2)
                swd(iL,u) = 0;
            elseif(swd(iL,u) <= -G2)
                swd(iL,u) = 0;
            end
        end
    end
    end
    A0 = iswt(swa,swd,wName);
    y_swt(array(den):array(den+1)) = A0(1:array(den+1)-array(den)+1);
end

end

%-----
CleanEEG_swt = [CleanEEG_swt; y_swt(1:N-Overlap)];
y_swt(1:N-Overlap) = [];

end

csvwrite('CleanEEG.csv',CleanEEG);
csvwrite('Raw_Data1_S1.csv',y2(1:length(CleanEEG)));

count = size(CleanEEG,1);
fs = 256;
time = 0:1/fs:(((count))/fs));
time = time';

%% DWT time-base plot .....
figure(1);plot(time(1:end-1),y2(1:length(CleanEEG)),'b-','lin-
ewidth',2);
hold on;
figure(1);plot(time(1:end-1),CleanEEG,'r--','linewidth',2);
xlabel('Time (Seconds)','FontSize',12,'FontName','Arial'); ylabel('Am-
plitude (µV)','FontSize',12,'FontName','Arial');
legend('Raw EEG','Clean EEG');
title('Raw EEG Vs Clean EEG Signal','FontSize',12,'FontName','Arial');

figure(3);plot(y2(1:length(CleanEEG)),'b-');    % Sample base DWT

%% Performance Evaluation is shown only for single dataset1: .....

%% Correlation Coefficients (CC) for OA zones (manually selected)
%% CC : Dataset1 [FP1] DWT
CC1 = corrcoef(y2(2937:3028),CleanEEG(2937:3028));
CC2 = corrcoef(y2(3561:3649),CleanEEG(3561:3649));
CC3 = corrcoef(y2(4225:4266),CleanEEG(4225:4266));
CC4 = corrcoef(y2(4837:4878),CleanEEG(4837:4878));
CC5 = corrcoef(y2(5485:5568),CleanEEG(5485:5568));
CC6 = corrcoef(y2(6091:6194),CleanEEG(6091:6194));
CC7 = corrcoef(y2(6800:6889),CleanEEG(6800:6889));
CC8 = corrcoef(y2(7428:7517),CleanEEG(7428:7517));
CC9 = corrcoef(y2(8100:8140),CleanEEG(8100:8140));

```

```

DWT_CC_FP1 =
[CC1(1,2);CC2(1,2);CC3(1,2);CC4(1,2);CC5(1,2);CC6(1,2);CC7(1,2);CC8(1,2
);CC9(1,2)]
csvwrite('DWT_CC_DS_1_ch2.csv',DWT_CC_FP1);

%% Mutual Information DWT .....
y2_MI = y2(1:length(CleanEEG));
CleanEEG_MI = CleanEEG;
MI_dwt = minfo(y2_MI,CleanEEG_MI)

%% MSCOHERE DWT .....
[Cxy,F] = mscohere(y2(1:length(CleanEEG)),CleanEEG,[],[],256,128);
figure;stem(F(1:40),Cxy(1:40),'ko','linewidth',2);
title('Magnitude Squared coherence plot[DWT]','FontSize',12,'Font-
Name','Arial');
xlabel('Frequency (Hz)','FontSize',12,'FontName','Arial');
ylabel('Cxy','FontSize',12,'FontName','Arial');

%% Time-Freq_Analysis DWT.....
[ALLEEG EEG CURRENTSET ALLCOM] = eeglab;
EEG = pop_importdata('dataform-
at','ascii','nbchan',0,'data','C:\\Charvi\\Spring_15\\Thesis\\Spike_De-
tec-
tion_Code_Feb12_15\\Raw_Data1_S1.csv','srate',128,'pnts',0,'xmin',0);
[ALLEEG EEG CURRENTSET] = pop_newset(ALLEEG, EEG, 0,'set-
name','Ch1','gui','off');
EEG = eeg_checkset( EEG );
figure(5);subplot(2,1,1); pop_newtimef( EEG, 1, 1, [6000 69000], [3
0.5] , 'baseline',[0], 'plotitc' , 'off', 'plotphase', 'off', 'pa-
dratio', 1);
title('Raw EEG');

EEG = pop_importdata('dataform-
at','ascii','nbchan',0,'data','C:\\Charvi\\Spring_15\\Thesis\\Spike_De-
tection_Code_Feb12_15\\CleanEEG.csv','srate',128,'pnts',0,'xmin',0);
[ALLEEG EEG CURRENTSET] = pop_newset(ALLEEG, EEG, 0,'set-
name','Clean_EEG','gui','off');
EEG = eeg_checkset( EEG );
subplot(2,1,2); pop_newtimef( EEG, 1, 1, [6000 69000], [3 0.5]
, 'baseline',[0], 'plotitc' , 'off', 'plotphase', 'off', 'padratio',
1);
title('Clean EEG');

```

B: ONLINE MATLAB BASED OA REMOVAL CODE

MATLAB script presented here is a part of the entire code of “data acquisition system” for the two GUI buttons: ‘Start’ and ‘Save Data’. Where ‘Start’ button starts the online data acquisition with hybrid OA removal algorithm. ‘Save Data’ saves the raw EEG, Clean EEG on hard disk for the future analysis purpose.

```
%% Executes on pressing the button "Start" on GUI.
function data_Callback(hObject, eventdata, handles)

global handle;
global uC32;
global chan1_final_uv;
global chan2_final_uv;
global time;
global M;
global marker;
global numIteration;
global Loop_Time;

global raw_data;
global newraw_data; % For saving Raw data
%===== EEG Denoising =====
global y1;
global CleanEEG;
global NM_Ch1;
global NM_Ch2;
%=====
set(handles.edit3, 'String', '');
set(handles.listbox1, 'String', '');
set(handles.listbox2, 'String', '');
set(handles.edit5, 'String', '');

oldmsgs = cellstr(get(handles.listbox1, 'String'));
set(handles.listbox1, 'String', [oldmsgs; {'Setting Up connection and pro-
files'}]); % Status msg in GUI

if ismac == 0
    uC32 = serial('COM4', 'BaudRate', 115200, 'DataBits', 8, 'Parity',
'none', 'StopBits', 1, 'InputBufferSize', 1*534);
else
    uC32 = serial('/dev/tty.usbserial-AE00DNQ1', 'BaudRate', 115200, 'Dat-
aBits', 8, 'Parity', 'none', 'StopBits', 1, 'InputBufferSize', 1*534);
end

oldmsgs = cellstr(get(handles.listbox1, 'String'));
set(handles.listbox1, 'String', [oldmsgs; {'Serial Port opening...'}]);
% Status msg in GUI

fopen(uC32)
oldmsgs = cellstr(get(handles.listbox1, 'String'));
```

```

set(handles.listbox1, 'String', [oldmsgs; {'Serial Port opened...'}]);
% Status msg in GUI

pause(4);

raw_data=[];
newraw_data = [];
chan1_final_uv=[];
chan2_final_uv=[];
marker=[];
Loop_Time=[];
NM_Ch1 = [];
NM_Ch2 = [];

%% ===== Hybrid OA removal algorithm appended here =====
%% intialization .....
y1 = [];
CleanEEG = [];
T = 0.29; % Signal epoch
Ts = 0.0039; %sampling rate 256sps
M = round(T/Ts);
k = [0 1 2 3];
v = 3;
m = 1:1:M+1;
tm = Ts.* m;
K = 2;
Wm = ones(1,length(m));
N = 128;
Delay = ceil(M/2); % filter order/2 appx : [36/2]
wName='bior4.4'; %coif3 bior4.4
level = 8;
Overlap = 40;
%===== Denoising Declaration ends =====
try
set(handles.pushbutton5, 'UserData', 0) % To ensure stop button also
stops %reading port

EEG_start=clock

oldmsgs = cellstr(get(handles.listbox1, 'String'));
set(handles.listbox1, 'String', [oldmsgs; {'EEG_Start '};
{num2str(EEG_start)}]); % Status msg in GUI

for numIteration = 1:1:500

set(handles.edit3, 'String', num2str(numIteration)); % displays cur-
rent %Iteration
on GUI

pause(0.1);
if get(handles.pushbutton5, 'UserData')
break
end

a = fread(uC32); % stores received online raw data
raw_data = a;

```

```

newraw_data(numIteration,:) = a;
count1 = size(raw_data,1);
count2=count1-22;
chan1_final = [];
chan2_final = [];
fs = 256;
time = 0:1/fs:(((count2-1)/4)/fs);
time = time';

i=1;

%% 22 bytes Header checking code -----

if(a(i)=='D') %Start key header check

    oldmsgs = cellstr(get(handles.listbox1,'String'));
    set(handles.listbox1,'String',[oldmsgs;{'Header Received!'}]);
% Status msg in GUI

    cnt_0 = a(18);
    cnt_8 = a(17);
    cnt_16 = a(16);
    cnt = 2^16*cnt_16+2^8*cnt_8+cnt_0; % bit 15,16,17 repre-
                                     sents the % packet count

    if(numIteration == 1)
        cnt_ref = cnt;
    else
        diff = cnt - cnt_ref;
        if(diff > 1)

            oldmsgs = cellstr(get(handles.listbox2,'String'));
            set(handles.listbox2,'String',[oldmsgs;{num2str(num-
mIteration)}]); % packet missed display on GUI
        end
        cnt_ref = cnt;
    end
else
    oldmsgs = cellstr(get(handles.listbox2,'String'));
    set(handles.listbox2,'String',[oldmsgs;{num2str(numItera-
tion)}]); % Header missed packet number display on GUI

    oldmsgs = cellstr(get(handles.listbox1,'String'));
    set(handles.listbox1,'String',[oldmsgs;{'Header Missed!'}]);
% Status msg in GUI
end

raw_data=raw_data(23:end) ;
%% 16 bit Raw EEG data formation .....
while i<count2

    chan1_lsb=raw_data(i);
    i=i+1;
    chan1_msb=raw_data(i);
    i=i+1;

```

```

chan1=2^8*chan1_msb+chan1_lsb;
chan1_final=[chan1_final;chan1];
chan2_lsb=raw_data(i);
i=i+1;
chan2_msb=raw_data(i);
i=i+1;

chan2=2^8*chan2_msb+chan2_lsb;
chan2_final=[chan2_final;chan2];
end

e1=find(chan1_final>29821);
e2=find(chan2_final>29821);

e11=isempty(e1);
e22=isempty(e2);

if(e11==0)
    set(handles.edit1,'String','Check elect1');
else
    set(handles.edit1,'String','');
end

if(e22==0)
    set(handles.edit2,'String','Check elect2');
else
    set(handles.edit2,'String','');
end

Ch1 = ((chan1_final)*3.3*1000000)/(65536*750.8);
% Actual data conversion with gain adjustment
Ch2 = ((chan2_final)*3.3*1000000)/(65536*750.8);

chan1_final_uv=[chan1_final_uv;Ch1];
chan2_final_uv=[chan2_final_uv;Ch2];

%===== Ch1 & Ch2 conversion to actual raw data =====

NM_S1_Ch1= -(Ch1 - mean(Ch1)); % - ve because Inverting opamp output
NM_S1_Ch2= -(Ch2 - mean(Ch2));
NM_Ch1 = [NM_Ch1;NM_S1_Ch1];
NM_Ch2 = [NM_Ch2;NM_S1_Ch2];

%===== Ch1 or Ch2 EEG De-Noising =====
if(numIteration > 4)
    y1 = [y1;NM_S1_Ch2];

    y = y1(1:N);
    n = (length(y)- M);
    vkn = zeros(M,n,length(k));
    array = [];
    y3 = zeros(2^level,1);
    f = zeros(M,n,K);

```

```

f1 = zeros(M,n);
f2sec_epoch = zeros(length(y),1);

%----- calculating impulse response (hk)-----
g0 = Wm.*((-1)^(k(1)+1))/factorial(v-1).*(4.*tm.*(2.*tm - 2) + 2.*(tm -
1).^2 + 2.*tm.^2);
g1 = Wm.*((-1)^(k(2)+1))/factorial(v-1).*(-12.*tm.*(tm - 1).^2 - 2.*(tm
- 1).^3 - 3.*tm.^2.*(2.*tm - 2));
g2 = Wm.*((-1)^(k(3)+1))/factorial(v-1).*(16.*tm.*(tm - 1).^3 + 2.*(tm
- 1).^4 + 12.*tm.^2.*(tm - 1).^2);
g3 = Wm.*((-1)^(k(4)+1))/factorial(v-1).*(-20.*tm.*(tm - 1).^4 - 2.*(tm
- 1).^5 - 20.*tm.^2.*(tm - 1).^3);

%% FIR filter -----
for j = 1:n
    vkn(:,j,1) = filter(g0, 1, y(j:(j+M-1)));    %k = 0
    vkn(:,j,2) = filter(g1, 1, y(j:(j+M-1)));    %k = 1
    vkn(:,j,3) = filter(g2, 1, y(j:(j+M-1)));    %k = 2
    vkn(:,j,4) = filter(g3, 1, y(j:(j+M-1)));    %k = 3
end

%% Volterra Filter for decision function calculation -----
for s = 1:K
    for r = 1:n
        f(:,r,s) = (vkn(:,r,s+1).^2) - (vkn(:,r,s).* vkn(:,r,s+2));
        f(:,r,s) = (max(0,f(:,r,s))/1000000);
    end
end

for d = 1:n
    f1(:,d) = f(:,d,1).* f(:,d,2);
    f2sec_epoch(d:d+M-1) = f2sec_epoch(d:d+M-1) + f1(:,d);
end

%% Threshold Function -----
Gamma1 = 0.001/(mean(f2sec_epoch)+ std(f2sec_epoch));

Threshold = (f2sec_epoch > Gamma1);
Threshold = Threshold';

%% OA Zone detection -----
ind = find(Threshold);
ii = 1;
while(ii <= (length(ind)-2))
    temp = ind(ii);
    while((ii < length(ind)) && ((ind(ii+1) - ind(ii)) == 1))
        ii = ii+1;
    end
    Edge_start = max(1, temp - Delay - 10);
    % to avoid -ve index...so min it can take is '1'
    Edge_end = min(numel(y),ind(ii));
    % if detected spike edge at the end it should retain
    array = [array; Edge_start; Edge_end];
    ii = ii+1;
end

```



```

%-----Wavelet Denoising.....

%% DWT
len1 = 0;
if(numel(array) ~= 0)
    for den = 1:2:length(array)
        y3(1:array(den+1)-array(den)+1) = y1(array(den):array(den+1));
        tic;
        [C,L] = wavedec(y3,level,wName);

        for iL = 2:1:6
            len1 = len1 + L(iL-1,:);
            Decoef = C(len1+1:len1+L(iL,:));

            G1 = Gamma1;

            for u = 1:size(Decoef)
                if(Decoef(u) > G1)
                    Decoef(u) = 0;
                elseif(Decoef(u) < -G1)
                    Decoef(u) = 0;
                else
                    Decoef(u) = Decoef(u);
                end
            end
            C(len1+1:len1+L(iL,:)) = Decoef;

        end
        A0 = waverec(C,L,wName); % Reconstructing signal
        timelog = [timelog;toc];
        y1(array(den):array(den+1)) = A0(1:array(den+1)-array(den)+1);
    % replacing with original wave to denoise
    end
end
%-----
% gives exact location on actual data
CleanEEG = [CleanEEG; y1(1:N-Overlap)];
y1(1:N-Overlap) = [];
%=====EEG De-noising ends =====
end

end

catch
    fclose(uC32);
    delete(uC32);
    disp('Fcloseruncatch')
end

EEG_portstop=clock
fclose(uC32);
delete(uC32);
disp('Fcloserun')
disp('Channel closed and cleared');

```

```

%% --- Executes on button press "Save Data":
function pushbutton5_Callback(hObject, eventdata, handles)

global chan1_final_uv;
global chan2_final_uv;
global final_val;
global newraw_data;
global CleanEEG;
global NM_Ch1;
global NM_Ch2;

disp('Stop invoked')
set(handles.pushbutton5, 'UserData', 1)
EEG_stop=clock
csvwrite('Newraw_data.csv', newraw_data);
final_val=[chan1_final_uv,chan2_final_uv];
csvwrite('final_value.csv', final_val);
csvwrite('CleanEEG.csv', CleanEEG);
csvwrite('Ch1_Data.csv', NM_Ch1);
csvwrite('Ch2_Data.csv', NM_Ch2);

```

C: OFFLINE C BASED OA REMOVAL CODE

```

#include <stdio.h>
#include <math.h>
#include <string.h>

void main()
{
    int M=74,win = 54, n=M+1,d,f11=0,temp=0,OA_index[10]={0},i,overlap=40,iter;
    float T = 0.29,Ts = 0.0039,Mean_f_DF = 0.0,SD_DF = 0.0, Gamma1,y_mean;
    float g0[75],g1[75],g2[75],g3[75],y[9000],f,y1[128]={0};
    float vkn0,vkn1,vkn2,vkn3;
    float f1,f2,DF,final_DF[128]= {0},temp_DF[128]= {0};
    float y3[128] = {0},y4[128] = {0},clean_EEG[9000]={0};
    int j=1,w,FIR_delay = 37,Index_Falg = 0,level=7;
    int g,h,L[9]={0};      // y = input to the filter; vkn = o/p coefficients

    // input data text file reading.....
    FILE *ptr_file = fopen("input_C_Ch1.txt","r");    //input_C_Ch1.txt
    if (!ptr_file)
        d = 1;
    while(fscanf(ptr_file,"%f",&f) != EOF)
    {
        y[f11] = f;
        f11+=1;
    }
    fclose(ptr_file);

    // impulse response h(n) calculations.....
    for(j=1; j<=M+1; j++)
    {
        g0[j-1] = -0.5*(4*(j*Ts)*(2*(j*Ts)-2) + 2*((j*Ts)-1)*((j*Ts)-1) +
        2*(j*Ts)*(j*Ts));

        g1[j-1] = 0.5*(-12*(j*Ts)*((j*Ts)-1)*((j*Ts)-1) - 2*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-
        1) - 3*(j*Ts)*(j*Ts)*(2*(j*Ts)-2));

        g2[j-1] = -0.5*(16*(j*Ts)*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1) + 2*((j*Ts)-1)*((j*Ts)-
        1)*((j*Ts)-1)*((j*Ts)-1) + 12*(j*Ts)*(j*Ts)*((j*Ts)-1)*((j*Ts)-1));

        g3[j-1] = 0.5*(-20*(j*Ts)*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1) - 2*((j*Ts)-
        1)*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1) - 20*(j*Ts)*(j*Ts)*((j*Ts)-
        1)*((j*Ts)-1)*((j*Ts)-1));
    }

    for(iter=0;iter<=99;iter++)
    {
        for (w=0;w<128;w++)
            y1 [w] = y[iter*(128-overlap)+w]; // overlapping purpose

        // FIR filter implementation using Convolution.....

        for(w=0; w<win; w++) //No of sliding windows = 'win'
        {
            for (g=0; g<M+1; g++)
            {
                vkn0 = 0.0;
            }
        }
    }
}

```

```

vkn1 = 0.0;
vkn2 = 0.0;
vkn3 = 0.0;

for (h=0; h<=g; h++)
{
    vkn0 = vkn0 + y1[h+w]*g0[g-h]; //k=0
    vkn1 = vkn1 + y1[h+w]*g1[g-h]; //k=1
    vkn2 = vkn2 + y1[h+w]*g2[g-h]; //k=2
    vkn3 = vkn3 + y1[h+w]*g3[g-h]; //k=3
}
// Volterra Filter.....
f1 = (vkn1*vkn1 - vkn0*vkn2)/1000000;
f2 = (vkn2*vkn2 - vkn1*vkn3)/1000000;
if (f1 < 0.0)
    f1 = 0.0;
if (f2 < 0.0)
    f2 = 0.0;

// Decision Function.....
DF = f1 * f2;
temp_DF[g+w] = temp_DF[g+w] + DF;
}
}

// Mean Calculations.....
Mean_f_DF=0;
for (w=0;w<128;w++){
    final_DF[w] = temp_DF[w];
    temp_DF[w] = 0.0;
    Mean_f_DF = Mean_f_DF + final_DF[w]; }
Mean_f_DF = Mean_f_DF/128; //Mean

//STD Calculations.....
SD_DF=0;
for (w=0;w<128;w++)
    SD_DF += (final_DF[w] - Mean_f_DF)*(final_DF[w] - Mean_f_DF);
SD_DF = sqrt(SD_DF/127); //SD

// Threshold Calculation.....
Gamma1 = 0.001/(Mean_f_DF + SD_DF);

// OA Zone detection.....
for(w=0;w<=9;w++)
    OA_index[w]=0; //Array Initialization to 0

w=0;Index_Falg=0;int s=0,e=0;
while(w<128)
{
    if(final_DF[w] > Gamma1)
    {
        temp = w;
        while((final_DF[w] > Gamma1) && w <128)
            w+=1;
    }
}

```

```

//Start Index
s = (temp-FIR_delay);
if (s > 0)
    OA_index[Index_Falg] = s;

//end Index
e = w-1+10;
if (e < 128)
    OA_index[Index_Falg+1] = e;
else
    OA_index[Index_Falg+1] = 127;
Index_Falg += 2;
}
w+=1;
} // end of OA zones detection

// OA Removal code starts here.....

if(Index_Falg != 0)
{
    for (h=0;h<Index_Falg/2;h++) //loop for each OA zone...
    {
        g=0;
        for(j=OA_index[2*h];j<=OA_index[2*h+1];j++)
        {
            y4[g] = y1[j];
            g+=1;
        }
        // start of DWT using HAAR wavelet function
        w = 128; j=0;
        while(w>1)
        {
            L[j] = w;
            w/=2;
            for(i=0;i<w;i++)
            {
                y3[i] = (y4[2*i] + y4[2*i+1])/sqrt(2.0); //Approximation coefficients
                y3[i+w] = (y4[2*i] - y4[2*i+1])/sqrt(2.0); //Detail coefi
            }
            for(i=0;i<w;i++)
            y4[i] = y3[i];
            j+=1;
        } // End of DWT
//thresholding strats here.....
for(g=1;g<16;g++) // detail coefficients form level 4-8
{
    if(y3[g] > Gamma1)
        y3[g] = 0.0;
    else if (y3[g] < -Gamma1)
        y3[g] = 0.0;
}

// iDWT starts here.....
w=1;
while(w<65)
{

```

```

    for(i=0;i<w;i++)
    {
        y4[2*i] = (y3[i] + y3[i+w])/sqrt(2.0); //Approximation coefficients
        y4[2*i+1] = (y3[i] - y3[i+w])/sqrt(2.0); //Detail coefi
    }
    w*=2;
    for(i=0;i<w;i++)
    y3[i] = y4[i];
    } // End of iDWT
// iDWT ends here....
g=0;
for(j=OA_index[2*h];j<=OA_index[2*h+1];j++)
{
    y1[j] = y4[g]; //clean samples replaced with original
    g+=1;
}
} // end of denoising for single OA zone
} // end of denoising of all existing OAs in the current chunk

//clean EEG storing.....

for(i=0;i<88;i++)
clean_EEG[88*iter + i] = y1[i]; //output buffer with clean EEG
for(i=0;i<40;i++)
    y[(iter+1)*(128-overlap)+i] = y1[88+i]; // moving last 40 samples to first 40
sample block for next iteration considering overlapping

} // end of iter loop

// Clean EEG storage in .txt file.....
FILE *r = fopen("DS_C_output.txt", "w"); //output_C_Ch1.txt
if (f == NULL)
    printf("Error opening file!\n");
for(i=0;i<8800;i++)
    fprintf(r, "%f \n",clean_EEG[i]);

fclose(r);
//-----
} // end of main

```

D: ONLINE C BASED OA REMOVAL CODE

The online C based hybrid OA removal code was developed using PSoC creator 3.0 and here in appendix D, only the parts of ADC ISR and main code blocks are mentioned related to the thesis work done.

- **ADC ISR code block:**

```
// ===== ADC ISR block =====
// Check for the Overflow
if(Last == buffer_size)
{
    Last = 0;
    Overflow = 1;
}

//To avoid race condition between First and Last
// First always > Last if Overflow = 1
if((Overflow == 1) && ((First - 2) < Last))
// hold_Flag = 1 holds new data to get stored in Ring buffer
    hold_Flag = 1;
else
    hold_Flag = 0; //allows new ADC data to get stored in ring buffer

// New ADC sampled EEG 16- bit data storage in ring buffer

if((Last < buffer_size) && (hold_Flag == 0))
{
    buffer1[Last++] = chan1 & 0X00FF;
    buffer1[Last++] = (chan1>>8) & 0X00FF;
}

// ===== ADC ISR block ends =====
```

- **'Main' code block:**

```
// ===== PSoC-3 main code block =====

#include <device.h>
#include <math.h>
#include <string.h>

void timer_configuration(uint8);
void BT_datarate(void);
void writeData(uint8);

void main()
{
    int M = 38,i,y_mean,overlap=40,y,w,g,win = 90,h,s,e,flag=0;
    int Valid_Data = 0,sz,iter = 0,temp=0,Index_Falg;
    int m,j=1,FIR_delay = 37,level=7,OA_index[10];
    float Ts=0.0039,Mean_f_DF = 0.0,SD_DF = 0.0, Gamma1;
    float g0[75],g1[75],g2[75],g3[75],y1[128]={0};
    float vkn0,vkn1,vkn2,vkn3;
    float f1,f2,DF,final_DF[129]={0},temp_DF[129]= {0};
    float y3[128] = {0},y4[128] = {0},Temp_Overlap[40]={0};
```

```

char clean_EEG[537]={0},Temp_Out_Buf[80]={0};
char a[sizeof(float)];

// ===== Pre-existing code block =====

/* Enable Global Interrupts */
CyGlobalIntEnable;

// initialize components
enAMP_Write(0); //Instr. Ampl. Enable (Active low)

AMuxSeq_1_Start();
//VDAC8_1_Start();//ch2

AMuxSeq_1_Start();
//VDAC8_1_Start();//ch2
VDAC8_2_Start();//ch1
//VDAC8_3_Start();//VBiasElec
VDAC8_4_Start();
Opamp_1_Start();
Opamp_2_Start();
Opamp_3_Start();
Opamp_4_Start();
ResetBT_Write(1);

UART_Init();
UART_Start();

CyXTAL_32KHZ_Start();

UART_ClearRxBuffer();
UART_ClearTxBuffer();
ADC_DelSig_1_Start();
timer_configuration(1);
// ===== Pre-existing code block ends here =====

// ===== OA Detection algorithm starts here (Thesis work)=====

clean_EEG[0] = 0x44; // 'D' : header start key
// impulse response h(n) calculations.....
for(j=1; j<=M+1; j++)
{
    g0[j-1] = -0.5*(4*(j*Ts)*(2*(j*Ts)-2) + 2*((j*Ts)-1)*((j*Ts)-1) +
2*(j*Ts)*(j*Ts));

    g1[j-1] = 0.5*(-12*(j*Ts)*((j*Ts)-1)*((j*Ts)-1) - 2*((j*Ts)-
1)*((j*Ts)-1)*((j*Ts)-1) - 3*(j*Ts)*(j*Ts)*(2*(j*Ts)-2));

    g2[j-1] = -0.5*(16*(j*Ts)*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1) +
2*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1) +
12*(j*Ts)*(j*Ts)*((j*Ts)-1)*((j*Ts)-1));

    g3[j-1] = 0.5*(-20*(j*Ts)*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-
1) - 2*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1) -
20*(j*Ts)*(j*Ts)*((j*Ts)-1)*((j*Ts)-1)*((j*Ts)-1));
}

```



```

while(1)
{
    Pin_1_Write(1);
    Pin_2_Write(0);

    //cnt = 0: very first iteration at time '0'; cnt = 1; otherwise;
    if(cnt == 0){
        sz = 256;
        g = 128;
        i = 0;
        m=9;
        flag=0;
    }
    else{
        sz = 176;
        g = 88;
        i = overlap;
        m=89;
    }

    // Valid_Data calculation
    if(Overflow == 0)
        Valid_Data = Last - First;
    else
        Valid_Data = Last + 768 - First; // due to ring counter

    // start raw EEG processing if Valid_Data >= 176 or 256
    if(Valid_Data >= sz)
    {
        // Raw EEG storage : Temp buf to clean_EEG buffer
        if(flag==1){
            j=0;
            while(j<overlap){
                clean_EEG[9 + 2*j] = Temp_Out_Buf[2*j];
                clean_EEG[10 + 2*j] = Temp_Out_Buf[2*j+1];
            }
        }
        y_mean = 0;
        w = i;

        j=0;
        while(w < 128)
        {
            if(w<88){
                clean_EEG[m++] = buffer1[First]; //last 48 Raw samples
                clean_EEG[m++] = buffer1[First+1];
            }
            else{
                // temp overlapped 40 raw sampled storage
                Temp_Out_Buf[j++] = buffer1[First];
                Temp_Out_Buf[j++] = buffer1[First+1];
            }
            // making 16 bit data
            y = (buffer1[First+1] & 0xFF00) | buffer1[First];
            //conversion to original data
            y1[w] = y*0.067099; // gain adjustment
        }
        // mean calculation
    }
}

```

```

y_mean += y1[w];
w++;

    if((First+2) < 768)
        First += 2;
    else{
        Overflow = 0;
        First = 0;
        }
    }

flag=1;
y_mean = y_mean/g;    // mean

while(i < 128)
    y1[i++] = -(y1[i] - y_mean);    // mean adjustment

//=====OA Detection Code:=====

// FIR filter implementation using Convolution....

for(w=0; w<win; w++)    //No of win = 'n' = 90
{
    for (g=0; g<M+1; g++)
    {
        vkn0 = 0.0;
        vkn1 = 0.0;
        vkn2 = 0.0;
        vkn3 = 0.0;
        for (h=0; h<=g; h++)
        {
            vkn0 = vkn0 + y1[h+w]*g0[g-h];    //k=0
            vkn1 = vkn1 + y1[h+w]*g1[g-h];    //k=1
            vkn2 = vkn2 + y1[h+w]*g2[g-h];    //k=2
            vkn3 = vkn3 + y1[h+w]*g3[g-h];    //k=3
        }

// Volterra Filter.....
        f1 = (vkn1*vkn1 - vkn0*vkn2)/1000000;
        f2 = (vkn2*vkn2 - vkn1*vkn3)/1000000;
        if (f1 < 0.0)
            f1 = 0.0;
        if (f2 < 0.0)
            f2 = 0.0;

// Decision Function.....
        DF = f1 * f2;
        temp_DF[g+w] = temp_DF[g+w] + DF;
    }
}

// Mean Calculations.....
Mean_f_DF=0;
for (w=0;w<128;w++){
    final_DF[w] = temp_DF[w];
    temp_DF[w] = 0.0;
}

```

```

Mean_f_DF = Mean_f_DF + final_DF[w]; }
Mean_f_DF = Mean_f_DF/128; //Mean

//STD Calculations.....
SD_DF=0;
for (w=0;w<128;w++)
SD_DF += (final_DF[w] - Mean_f_DF)*(final_DF[w] - Mean_f_DF);
SD_DF = sqrt(SD_DF/127); //SD

// Threshold Calculation.....
Gamma1 = 0.001*(1/(Mean_f_DF + SD_DF));

// OA Zone detection.....
for(w=0;w<=9;w++)
    OA_index[w]=0; //Array Initialization to 0
for(w=1;w<=6;w++)
    clean_EEG[w] = 0;

w=0;Index_Falg=0;s=0,e=0;
while(w<128)
{
    if(final_DF[w] > Gamma1)
    {
        temp = w;
        while((final_DF[w] > Gamma1) && w <128)
            w+=1;

//Start Index
s = (temp-FIR_delay);
if (s > 0)
    OA_index[Index_Falg] = s;
//end Index
e = w-1+10;
if (e < 128)
    OA_index[Index_Falg+1] = e;
else
    OA_index[Index_Falg+1] = 127;
Index_Falg += 2;
    }
    w+=1;
} // end of OA zones detection

// OA Removal code starts here =====

if(Index_Falg != 0)
{
    for (h=0;h<Index_Falg/2;h++) //loop for each OA zone...
    {
        clean_EEG[2*h+1] = OA_index[2*h] & 0X00FF; //storing Detected
OA_Zones Starting edge
        clean_EEG[2*h+2] = OA_index[2*h+1] & 0X00FF; //storing Detected
OA_Zones ending edge

        g=0;
        for(j=OA_index[2*h];j<=OA_index[2*h+1];j++)

            y4[g++] = y1[j]; //copying only OA zone data for denoising in y4
    }
}

```

```

// start of DWT using Haar
w = 128;
while(w>1)
{
w/=2;
for(i=0;i<w;i++)
{
y3[i] = (y4[2*i] + y4[2*i+1])/sqrt(2.0); //Approximation coeffi-
cients
y3[i+w] = (y4[2*i] - y4[2*i+1])/sqrt(2.0); //Detail coeffi
}
for(i=0;i<w;i++)
y4[i] = y3[i]; //y3 = C
} // End of DWT
//thresholding stats here.....
for(g=1;g<16;g++)
{
if(y3[g] > Gamma1)
y3[g] = 0.0;
else if (y3[g] < -Gamma1)
y3[g] = 0.0;
}
// iDWT starts here.....
w=1;
while(w<65)
{
for(i=0;i<w;i++)
{
y4[2*i] = (y3[i] + y3[i+w])/sqrt(2.0); //Approximation coeffi-
cients
y4[2*i+1] = (y3[i] - y3[i+w])/sqrt(2.0); //Detail coeffi
}
w*=2;
for(i=0;i<w;i++)
y3[i] = y4[i];
} // End of iDWT
// iDWT ends here....
g=0;
for(j=OA_index[2*h];j<=OA_index[2*h+1];j++)
y1[j] = y4[g++]; //clean samples replaced with original

} // end of denoising for single OA zone
} // end of denoising of all existing OAs in the current chunk

//clean EEG storing.....
i=0;w=0;
while(i<352){
//7-0 bits
memcpy(a, &y1[w], sizeof(float));

clean_EEG[185 + i] = a[0]; //output buffer with clean EEG
i++;
//15-8 bits
clean_EEG[185 + i] = a[1];
i++;
//23-16 bits

```

```

clean_EEG[185 + i] = a[2]; //output buffer with clean EEG
i++;
//31-24 bits
clean_EEG[185 + i] = a[3];
i++;
w++;
}
for(i=0;i<40;i++){
Temp_Overlap[i] = y1[88+i]; // moving last 40 samples to first 40
sample block for next iteration considering overlapping
y1[i] = Temp_Overlap[i];
}
//.....
m=8;
clean_EEG[m--]= cnt & 0X00FF;
clean_EEG[m]= (cnt>>8) & 0X00FF;
cnt++;

for(i=0;i<537;i++)
UART_PutChar(clean_EEG[i]); //sending header[9] + Raw Data[176] +
Clean EEG []

}
}
}

```