A New Adaptive Filtering Algorithm for Signal Enhancement Problem with Hardware Implementation Using Arduino

Sameer AbdulKadhim Khudhaiyr
Electrical Engineering Department, Collage of Engineering, University of Babylon
samirabdcathem@yahoo.com
Awwab Qasim Jumaah Althahab
Electrical Engineering Department, Collage of Engineering, University of Babylon
eng.awwab.qasim@uobabylon.edu.iq

Abstract

One of the most important issues in the area of digital signal processing is a signal noise cancellation that has been studied and given a considerable attention. In recent few years, adaptive filters have been used to cancel out different kinds of noise from analog signals based on many adaptive algorithms. The aim of this paper is to study and implement a real-time line enhancement problem (denoising) based on using Recursive Least Squares (RLS) and the modified version of RLS algorithms, which is proposed as a new adaptive algorithm (mRLS). Both are carried out in the microcontroller (Arduino). The main advantage of those algorithms is that they update the coefficient values of adaptive filters every single iteration until convergence is occurring. The performance analysis of the algorithms is assessed by showing the output of adaptive filters (denoised signals) and calculating the power signal to noise ratio (SNR). To verify their robustness, different values of random noise power are taken. It is observed and proved from the results, based on the above performance parameters, that the (mRLS) algorithm performs better than the (RLS) algorithm in terms of noise cancellation.

Keywords: Signal Enhancement, Denoising, Adaptive Filters, Adaptive Algorithms, Arduino, RLS, and mRLS.

الخلاصة

واحدة من أهم القضايا في مجال معالجة الإشارات الرقمية هو إلغاء اشارة الضوضاء والتي تم دراستها وإعطائها اهتماما كبيرا. في السنوات القليلة الماضية، تم استخدام المرشحات التكيفية لالغاء أنواع مختلفة من الضوضاء من الإشارات التناظرية بناء على العديد من الخوارزميات التكيفية. الهدف من هذا البحث هو دراسة وتنفيذ مشكلة تعزيز الاشارة في الوقت الحقيقي (تقليل الضوضاء) يقوم على استخدام خوارزمية المربع الاقل التكراري (RLS)، ونسخة معدلة من خوارزمية RLS، التي اقترحت في هذا البحث كخوارزمية جديدة. تم تنفيذ كلاهما في المتحكم الدقيق (اردوينو). والميزة الرئيسية لهذه الخوارزميات هي تحديث القيم لمعاملات المرشحات التكيفية كل تكرار واحد حتى حدوث التقارب. يتم تقييم تحليل أداء الخوارزميات بإظهار الناتج من المرشحات التكيفية (إشارات محسنة) واحتساب نسبة قدرة الإشارة إلى نسبة الضوضاء (SNR). للتحقق من متانة الخوارزميات، تؤخذ قيم مختلفة من قدرة الضوضاء. لوحظ وثبت من النتائج، استنادا إلى معايير الأداء المذكورة أعلاه، أن اداء خوارزمية (mRLS) أفضل من خوارزمية (RLS) من حيث إلغاء الضوضاء.

الكلمات المفتاحية: تعزيز الاشارات، تقليل الضوضاء، المرشحات التكيفية، الخوارزميات التكيفية، المتحكم الدقيق اردوينو، RLS ،mRLS.

1. Introduction

Recently, one of interesting problems in digital signal processing used in our lives is the problem of signal enhancement that has been paid a huge concern. Due to the severe effect of interference noise in signals, it is necessary to extract the useful information from noise corrupted signals to be then analyzed and processed. Noise exists everywhere and generates randomly from many sources, including but not limited to, traffic, crowds, ventilation equipments, echoes, reverberation, or produces from electronic devices such as thermal noise, distortion products and tape hiss (Ritika *et. al*., 2014; Ben *et. al*., 2011). All of these kinds of noise can distort or mask the quality of useful signals.

The most common way to attenuate noise from signals is applying linear filtering to do single trial analysis. However, low signal to noise ratio (SNR) is a major drawback of linear filtering since the noise power of corrupted signals varies along the time. Thus, the concept of adaptive filtering has been grown (Simon, 2001). The coefficients of adaptive filters change as a function of the noise power contained in the signals. In other words, the noise power variation is taken into consideration to correct the filter parameters (Lalith and Soundara, 2012). Adaptive filter is now defined as a digital filter that adapts itself to changes in its input signal automatically based on adaptive algorithms, which is used to modify the filter coefficients according to given criteria. Probably, an error signal is a common criterion given to improve the performance of adaptive filters (Ritika *et. al*., 2014; Juan *et. al*., 2009). In the literature, there are diverse range of applications that adaptive filters are used as a solution to remove noise or enhance signals such as telephone echo cancellation, radar signal processing, biomedical signal enhancement, equalization and estimation of communication channels (Simon, 2001; Allam *et. al*., 2011;Colin, 2002). In (Rahul *et. al*.,2015),the authors presented a comparative study of adaptive filtering algorithms includes Normalized Least Mean Squared (NLMS) and (RLS) in acoustic noise cancellation. It is shown that the (RLS) performs better as per the SNR change and faster the convergence rate than (NLMS). (Jay and Nitin,2014) described the performance analysis of the (RLS) and the family of (LMS) algorithm for mean square error (MSE) comparison, and all results demonstrated the (RLS) algorithm outperforms the family of (LMS) algorithms in terms of convergence rate and (MSE). This explains why the family of (LMS) algorithm is not taken into consideration in this paper.

The purpose of this paper is to investigate removing random noise from analog corrupted signals (Solving Line Enhancement Problem) using two adaptive filter algorithms implemented in the microcontroller (Arduino) operating in real time. The first optimization algorithm that will be utilized and implemented as a minimization technique is Recursive Least Squares (RLS), which is introduced in (Rahul *et. al.,* 2015). The second algorithm is a modified version of (RLS) that we propose to adaptively estimate the coefficients of adaptive filter to attenuate noise in corrupted signals.

The paper is organized as follows: in section (2) an overview of a line enhancement problem is given and a set of adaptive algorithms previously mentioned is presented. Section (3) describes the implementation of the methods in the DSP microchip (Arduino). In section (4), a set of experimental measurements carries out to analyze the performance of the system, including the output of adaptive filters (denoised signals) with different values of random noise power and SNR. Finally, section (5) gives some concluding remarks of the work.

2. Adaptive Line Enhancement

The basic structure of adaptive line enhancement is shown in figure (1). A system of this type has been studied by (Chun-Tang *et al.,* 2014). The input $s(n)$ is a useful signal (unknown) added to the unknown random noise $v(n)$ that produce the corrupted signal $d(n)$, which is measured and defined in equation (1).

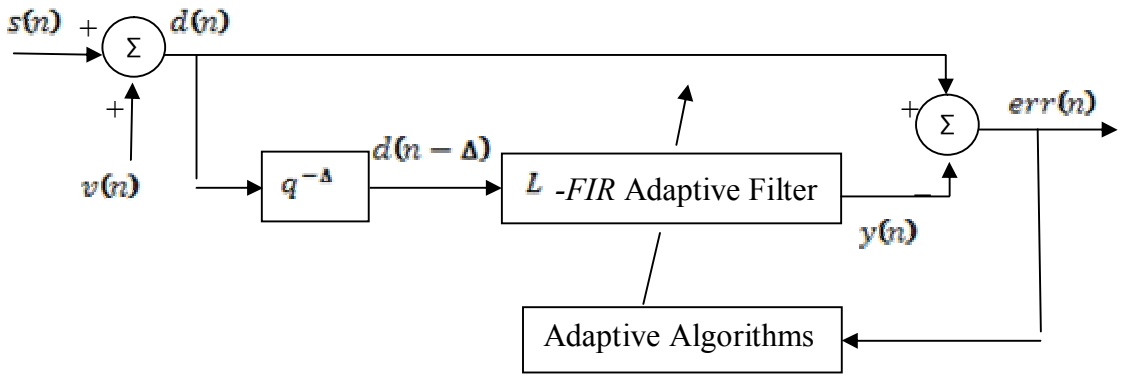


Figure 1: Basic Structure of Adaptive Line Enhancement.

The configuration consists of L-taps Finite Impulse Response (L -*FIR*) adaptive filter whose input is the corrupted signal $d(n)$ delayed with the variable value (Δ), which is the predicted distance of the adaptive filter in terms of the sampling period. The output of the filter is $y(n)$ that is obtained based on the measurement of time instant ($n-1$) and is defined in equation (2). The (L -*FIR*) filter coefficients can be adjusted by the error ($err(n)$) that is generated by comparing filter output $y(n)$ with the corrupted signal $d(n)$ and it is well defined in equation (3).

$$d(n) = s(n) + v(n) \tag{1}$$

$$y(n) = \sum_{l=0}^{L-1} h(l)\, d(n-l-\Delta) \tag{2}$$

and

$$err(n) = d(n) - y(n) \tag{3}$$

where $h_l(n)$ is the (*FIR*) filter coefficients. The error should converge to the unknown random noise $v(n)$. For computational simplicity, vector form is used to simplify equations (1), (2) and (3); yields

$$y[n] = [h(0) \quad h(1) \quad \cdots \quad h(L-1)] \begin{bmatrix} d(n-\Delta) \\ d(n-1-\Delta) \\ \vdots \\ d(n-L-\Delta+1) \end{bmatrix} \tag{4}$$

Let now define $H^T(n)$ to be the filter coefficients vector such that $H^T(n) = [h(0) \quad h(1) \quad \cdots \quad h(L-1)]$, and define $X(n)$ to be the input signal vector. Thus, (4) is now rewritten as:

$$y[n] = H^T(n).X(n) \tag{5}$$

Substituting (5) into (3), the error is written as:

$$err[n] = d(n) - H^T(n).X(n) \quad (6)$$

The optimization problem aims to find an estimate of the filter coefficients $H^T(n)$ from a set of observed real new samples $X(n)$. In other words, the filter parameters adapt in response to the error for each coming new sample (Chun-Tang *al.,* 2014).The adaptive algorithms are used to update the filter parameters. The adaptation equation for the Recursive Least Squares (RLS) (Rahul *et al.,* 2015) is described in equation (7) whereas the adaptation equation for the proposed algorithm (mRLS) is defined in (8).

$$H(n) = H(n-1) + p(n)X(n)\{d(n) - H^T(n-1)X(n)\} \quad (7)$$

and

$$H(n) = H(n-1) + p(n)X(n-1)\{d^*(n) - X^*(n-1)H(n-1)\} + p(n)\sigma_v^2\{(n-1)H(n-1) - (n-2)H(n-2)\} \quad (8)$$

where

$$p(n) = p(n-1) - \frac{p(n-1)X(n-1)X^*(n-1)p(n-1)}{1 + X^*(n-1)p(n-1)X(n-1)} \quad (9)$$

and σ_v^2 is the variance of random noise. $p(0)$ is an arbitrary positive definite matrix, typically $p(0) = p_0 I$ is chosen, where p_0 is a positive scalar and I is the identity matrix.

3. Hardware System

Arduino UNO is a flexible programmable hardware platform. This microcontroller comes from a company called ATMEL and the chip is known as an AVR (ATmega328P). It is running at 16MHz with an 8-bit core and has memory with 32KB of storage and 2KB of random access memory. The Arduino UNO board contains a 6 channel 10-bit analog to digital converter. This means that it will map input voltages (between 0 and 5 volts) into integer values (between 0 and 1023). This yields a resolution between readings of (5 volts / 1024) units or (4.9 mV) per unit. It takes about 100 microseconds to read an analog input, so the maximum reading rate is about 10,000 times a second(Simon, 2010).

The main goal of this paper is to present an implementation of adaptive denoising algorithms, which were previously mentioned in **Section 2**, using Arduino UNO microcontroller, and the system must operate in real time. The system is tested using electronic equipments in the Communications Lab of Electrical Engineering Department at the University of Babylon. We use main unit to generate information signal (6Vpp triangle with 50Hz frequency). Also, noise signal is generated with different voltage values at constant value of frequency (100 KHz). Then we use Module KL-93007 kit to add these two signals as shown in figure (2).

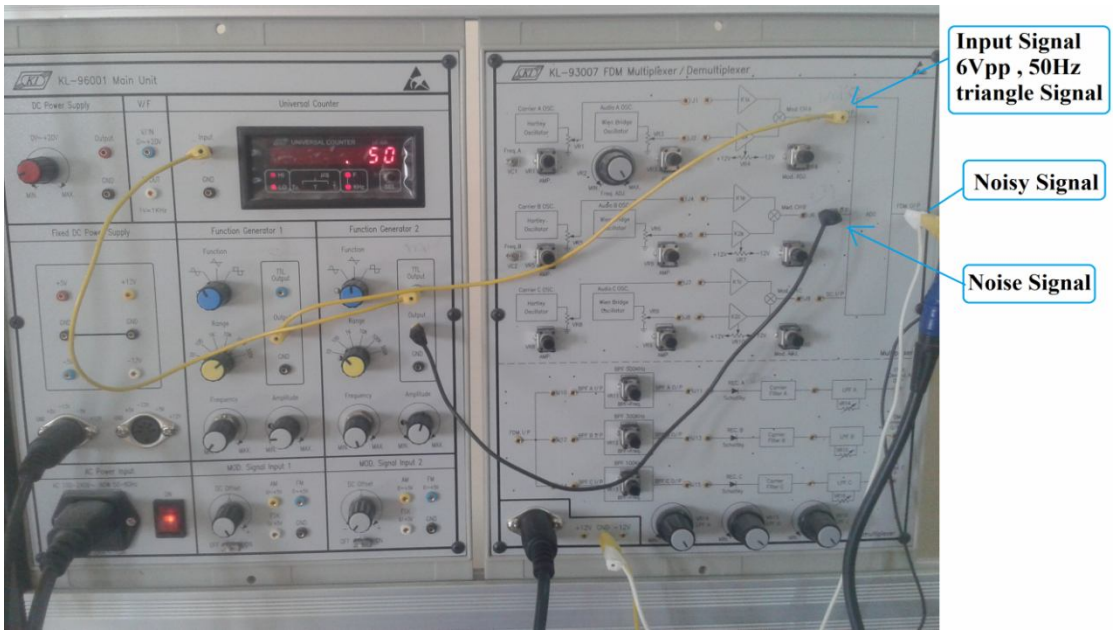


Figure 2: Generation Noisy Signal in Communication Lab.

The noise cancelling circuit is composed of three stages as shown in figure (3).

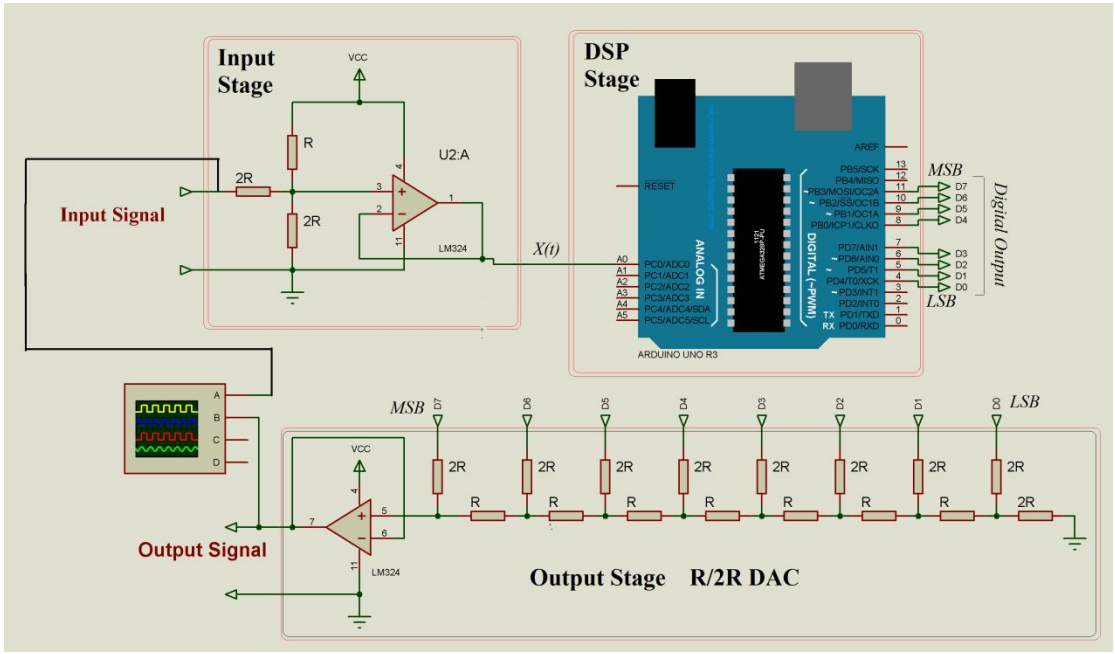


Figure 3: Stages of Hardware Noise Cancellation Scheme.

3.1 The First Stage

It is the input stage. The aim of this stage is to adjust the corrupted signal voltages for the ADC of microcontroller to be between 0 to 5 volts. This stage simply takes the noisy signal from Module KL-93007 kit and attenuates it by 0.25; then adding about 2.5 DC volts to shift the signal up. Finally, unity gain buffer preamplifier is used to protect Ardunio from high input voltage. If the input noisy signal is $S(t)$, the output signal of this stage, which is the input signal to DSP stage, will be:

$$x(t) = 0.25\ S(t) + 2.5 \qquad (10)$$

That made maximum allowable input signal voltage to the DSP stage about 10Vpp.

3.2 The Second Stage

It is a digital signal processing stage. First, ADC0 is used to read input sample with 10KHz sampling frequency and 10bits/sample resolution; then store it in value between [0 → 1023]. The pre-processing step will be:

1. Shift the contain of FIR filter with one step X[i-1]=X[i] where i= , , …2.1.
2. Convert input value from ADC0 into its equivalent real voltage (-2.5 → 2.5 volts), and the result will store in first memory location of FIR filter X[0].
3. Update the coefficient weight using one of previous algorithms.
4. Output sample will be Y= sum of(H[i]× X[i]) where i= 0, 1,….., $L-1$.
5. Divide Y by normalized factor N_f and add 2.5, the result should be positive number less than 5. (Note we use N_f=10 for L =11)
6. Finally map the result into 8bit decimal number where (0→ 0, 5→255); then 8 bit send into digital bins.
7. Go to step 1.

3.3 The Third Stage

It is the output stage. Since Ardunio has no analog output, we are obliged to use 8 bits R/2R DAC followed by unity gain buffer to convert output of Ardunio into equivalent analogue sample as shown in figure (4).

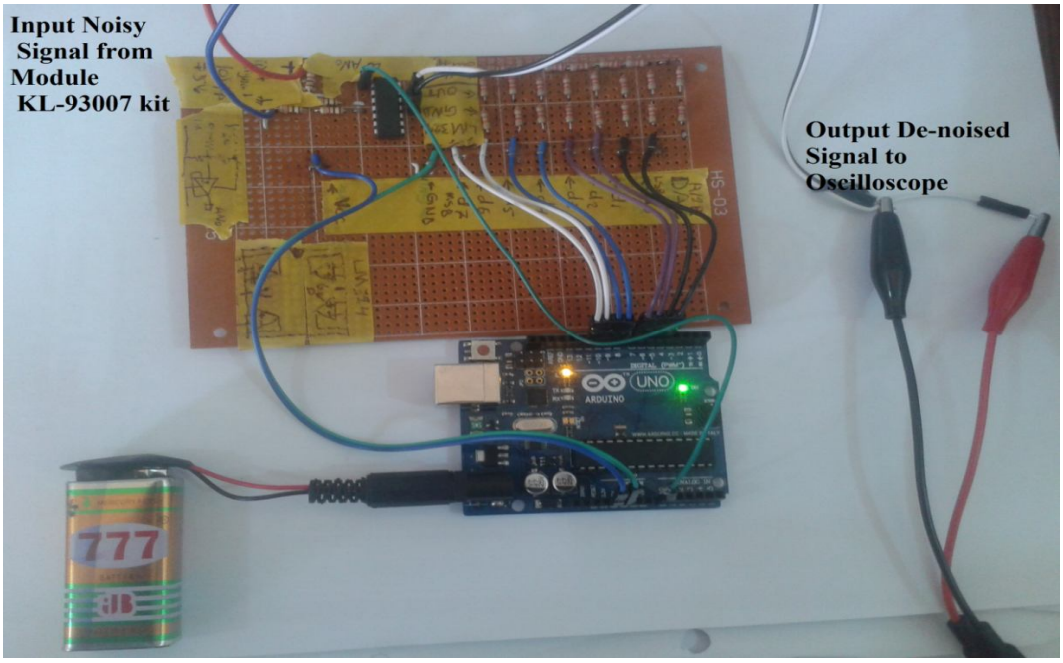


Figure 4: Actual Hardware Implementation.

Finally, the oscilloscope is used to compare between input noisy signal and output denoised signal.

4. Performance Analysis of The Results

Once the system was implemented, several measurements were carried out in order to analyze their performance. Thus, RLS and modified RLS (mRLS) algorithms presented in **Section 2** were tested with the following parameter:

1- Filter Length (L) =11 tab.
2- Delay Δ=10 tab.

For each implementation, oscilloscope was used to observe denoised signal with different amount of additive noise. Input SNRI can be calculated using:

$$SNRI = 10log_{10}\left(\frac{P}{N}\right) \quad (11)$$

where N is the noise power, and P is the input signal power. Triangle signal was used with the peak value A_S=3 V; then $P = \frac{(A_s)^2}{3} = \frac{(3)^2}{3} = 3$ Watt (for unit resistance load). If we consider noise signal is sine wave with A_n amplitude,

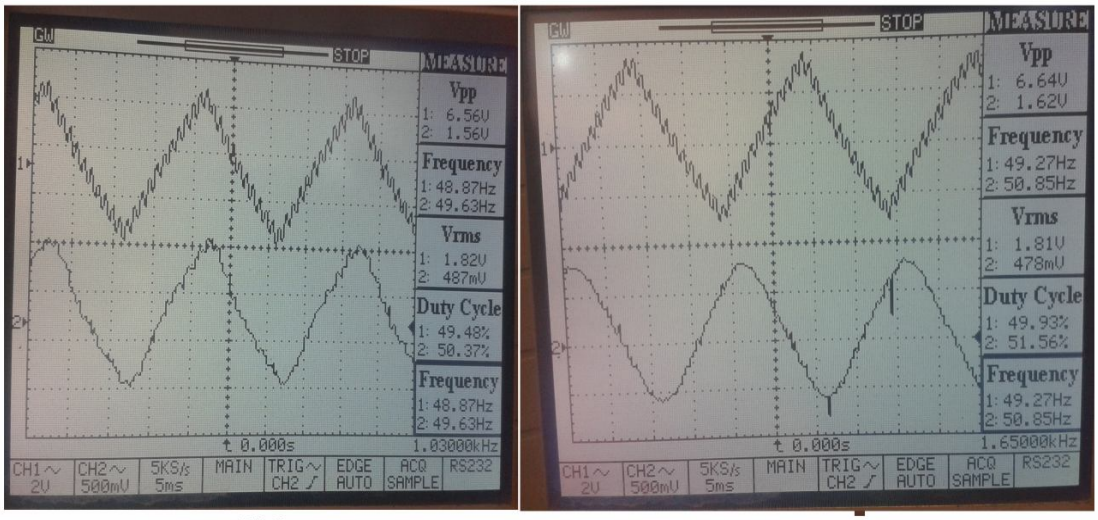
then $N = \frac{(A_n)^2}{2}$. Equation (11) can be rewritten as:

$$SNRI = 10log_{10}\left(\frac{6}{A_n^2}\right) \quad (12)$$

Four experimental tests were done in communication lab with different amount of noise voltages as given:

Test 1: Small Additive Noise Value

We set noise signal 100KHz, 1Vpp ($A_in = 0.5volt$) that made SNRI= 13.8 dB. The practical result for RLS and mRLS is shown in figure (5).



RLS mRLS

Figure 5: Output of Adaptive Filter Using (RLS) and (mRLS) Algorithms with 0.125 Watt of Noise Power.

Test 2: Medium Additive Noise Value

We set noise signal 100KHz, 2Vpp ($A_in = 1volt$) that made SNRI= 7.7 dB. The practical result for RLS and mRLS is shown in figure (6).

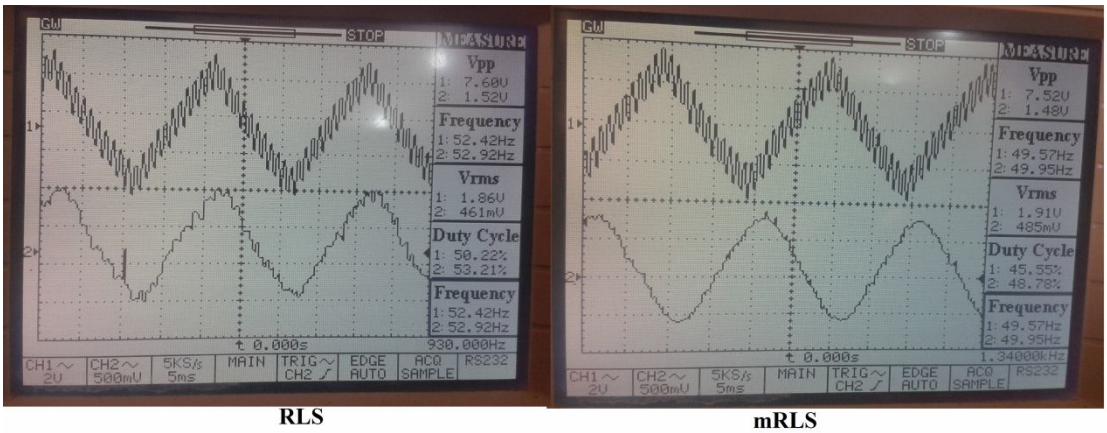


Figure 6: Output of Adaptive Filter Using (RLS) and (mRLS) Algorithms with 0.5 Watt of Noise Power.

Test 3: Large Additive Noise Value

We set noise signal 100KHz, 3Vpp ($A_n = 1.5volt$) that made SNRI= 4.25 dB. The practical result for RLS and mRLS is shown in figure (7).

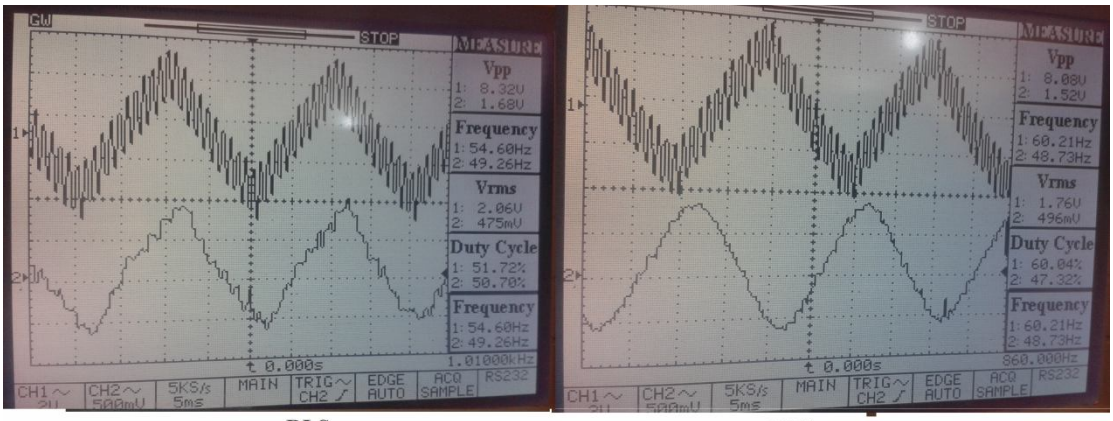


Figure 7: Output of Adaptive Filter Using (RLS) and (mRLS) Algorithms with 1.125 Watt of Noise Power.

Test 4: Extreme Additive Noise Value

We set noise signal 100KHz, 4Vpp ($A_n = 2volt$) that made SNRI= 1.7 dB. The practical result for RLS and mRLS is shown in figure (8).

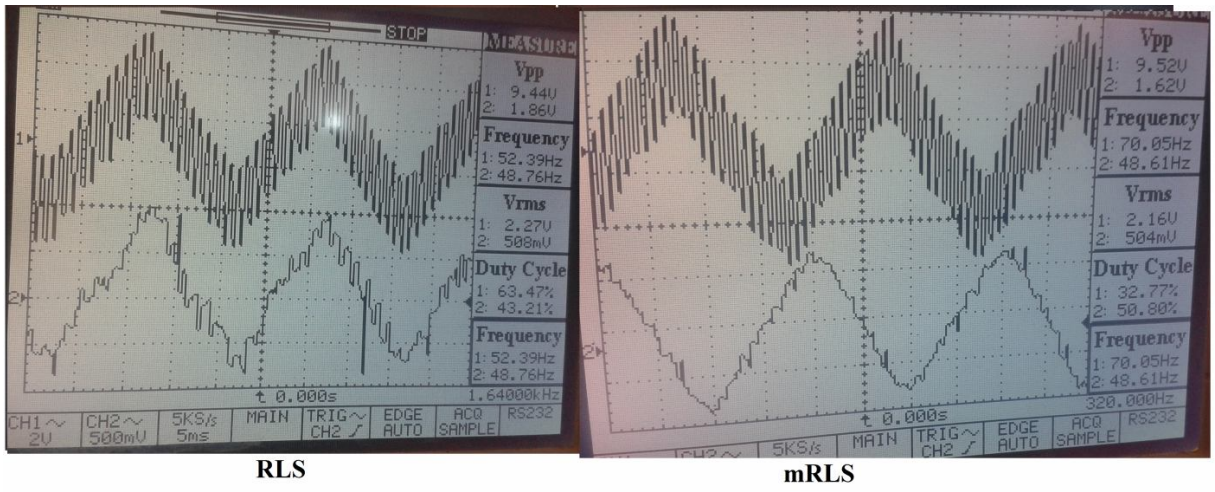


Figure 8: Output of Adaptive Filter Using (RLS) and (mRLS) Algorithms with 2 Watt of Noise Power.

5. Conclusion

This paper conducted a study of implementation real-time noise cancellation algorithms in a low-cost DSP using Ardunio UNO. In order to reduce computations and memory usage of the proposed algorithm, only 11 tap FIR filter were used. Note that this system requires 11 multiplies and 10 adds to implement. It also requires a total of 32 memory locations to store the 11 input signal samples, the 10 delay samples and 11 coefficient values, respectively. Each value in floating point format 4 byte each. It also takes about 100 microseconds to read an analog input value using ADC0 plus 40 microseconds to produce 8-bits output sample (5 microseconds for each bit). Based on the measurements, to make the system operates in real-time, we found that this system can be used only with low frequency signal. Thus, a 50Hz triangle signal as a test signal was used.

Comparisons mentioned in figures (5, 6, 7 and 8) show that the mRLS achieves higher quality in the noise signal cancellation, but at the same time standard RLS algorithm is simpler in evaluation process.

The large numerical calculation of mRLS algorithm obliged two substantial weak points. The first point is that this algorithm required large processing time that made denoised signal delay about (8 ~ 10 ms) from the original signal. The second point occurs when high order FIR filter is used, obliged us to use more powerful microcontroller (or microprocessor) and this made Ardunio UNO not suitable for this mission.

References

Allam Mousa, Marwa Qadus and Sherin Bader, 2011, "***Adaptive Noise Cancellation Algorithms Sensitivity to Parameters***" The 2nd International Conference on Multimedia Computing and Systems (ICMCS'11), IEEE, pp. 1-5.

Ben Gold, Nelson Morgan and Dan Ellis, 2011,"***Speech and Audio Signal Processing***" John Wiley and Sons, Inc, 2nd Edition.

Chun-Tang Chao Nopadon Maneetien, Chi-Jo Wang and Juing-Shian Chiou, 2014, "***Performance Evaluation of Heart Sound Cancellation in FPGA Hardware Implementation for Electronic Stethoscope***" Hindawi Publishing Corporation, The Scientific World Journal Volume 2014, Article ID 587238, 7 pages.

Colin H. Hansen, 2002, "***Understanding Active Noise Cancellation***" Taylor and Francis Inc., 1st Edition.

Jay Prakash Vijay and Nitin Kumar Sharma,2014,"***Performance Analysis of RLS Over LMS Algorithm for MSE In Adaptive Filters***" International Journal of Technology Enhancements and Emerging Engineering Research, Vol. 2, Issue 4, pp. 40-44.

Juan Gorriz *et. al*.,2009,"***A Novel LMS Algorithm Applied to Adaptive Noise Cancellation***" IEEE Signal Processing Letters, Vol. 16, No. 1, pp. 34-37.

Lalith kumar and Soundara Rajan,2012,"***Noise Suppression in Speech Signals Using Adaptive Algorithms***" International Journal of Engineering Research and Applications, Vol. 2, Issue 1, pp. 718-721.

Rahul Abhishek and V. Ramesh, 2015,"***Acoustic Noise Cancellation by NLMS and RLS Algorithms of Adaptive Filter***" International Journal of Science and Research, Vol. 4, Issue 5, pp. 198-202.

Ritika Thakur, Papiya Dutta and G. C. Manna,2014, "***Analysis and Comparison of Evolutionary Algorithms Applied to Adaptive Noise Cancellation for Speech Signal***" International Journal of Recent Development in Engineering and Technology, Vol. 3, Issue 1, pp. 172-178.

Simon Haykin, 2001,"***Adaptive Filter Theory***" Prentice Hall, 4th Edition.

Simon Monk,2010,***"30 Arduino Projects for the Evil Genius***" The McGraw-Hill Companies.