Implementation Of Least Mean Square Algorithm Using Arduino & Simulink

Mohcin Mekhfioui, Rachid Elgouri, Amal Satif, Meryem Moumouh, Laamari Hlou

Abstract: In this work, the least mean square (LMS) filter module is modeled, implemented and verified on a low-cost microcontroller to eliminate acoustic noise, which is a problem in voice communications. The main objective of this paper is to implement the module on an autonomous Arduino Due board in real-time, taking advantage of the low computational cost and ease of implementation through Matlab/Simulink programming. In the experimental application, the results of the implementation phase verify that the behavior of the implemented module is similar to the Simulink model.

Index Terms: Least Mean Square, Arduino Due, Matlab/Simulink, Real Time Workshop.

1. INTRODUCTION

Adaptive methods in signal processing aim at the automatic adaptation of processing operators to the statistical properties of signals and systems, as well as the adaptation to their variations over time. It is, therefore, a well-weighted mix between stationarity, which, thanks to the permanence in time of statistical properties, makes it possible to get rid of, or at least to reduce, purely random fluctuations, and non-stationarity, i.e. the "slow" variation over time of these properties, without which there would be no need for adaptive methods: it would suffice to calculate once and for all the "optimal filter" and then to put it online[1]. These methods have experienced considerable growth since the 1960s, due to the development of digital processing and the constant increase in the power of processing processors (DSP, Digital Signal Processors), allowing the implementation in real-time of increasingly sophisticated algorithms, at increasingly fast rates[2]. They have reached a certain maturity both in terms of the development and implementation of algorithms and in terms of theoretical tools for studying performance. This chapter aims to give a synthetic view, not exhaustive but sufficient, to allow the reader to quickly find the tools and results that interest him, and possibly references to works that allow to deepen specific aspects[3]. In this work, we implement a least mean square (LMS) algorithm in the Arduino board as free and low-cost hardware, to test its performance in real-time.This work is structured as follows: Section II presents Adaptive Filtering, Algorithm LMS, the Arduino Due board, Section III describes the implementation and the results, and finally, Section IV concludes this document.

- *M. MEKHFIOUI, Laboratory of Electrical Engineering and Energy Systems, Ibn Tofail University, Kenitra, Morocco. Email: mohcin.mekhfioui@uit.ac.ma*
- *R. ELGOURI, Laboratory of Electrical Engineering and Telecommunications Systems, ENSA, Ibn Tofail University, Kenitra, Morocco. Email: rachid.elgouri@uit.ac.ma*
- *SATIF, Laboratory of Electrical Engineering and Telecommunications Systems, ENSA, Ibn Tofail University, Kenitra, Morocco. Email: amal.satif@uit.ac.ma*
- *M. Moumouh, Laboratory of Electrical Engineering and Energy Systems, Ibn Tofail University, Kenitra, Morocco. Email: meryem.moumouh@gmail.com*
- *L. HLOU, Laboratory of Electrical Engineering and Energy Systems, Ibn Tofail University, Kenitra, Morocco. Email: laamari.hlou@uit.ac.ma*

2. MATERIAL AND METHODOLOGY

2.1 About the Adaptive Filter

An adaptive filter is, by definition, a digital filter whose coefficients, estimated according to a given criterion (generally of the least squares type), adapt to variations in the received signals. Usually, an input vector and a desired response are used to define an error vector which then controls the evolution of the parameters of the adaptive filter[4], [5]. The objective of adaptive filters is to approximate unknown transfer functions by "learning" the characteristics of the signals as they occur. They consist of two parts: a digital filter to filter and an algorithm to adjust the coefficients of this filter. In the following figure we can see a simplified diagram of an adaptive filterFig.1.

Where:

$d(n)$: represents the desired signal,

$y(n)$:The output of the digital filter $y(n) = x(n) * h(n)$

$e(n)$: The error signal.

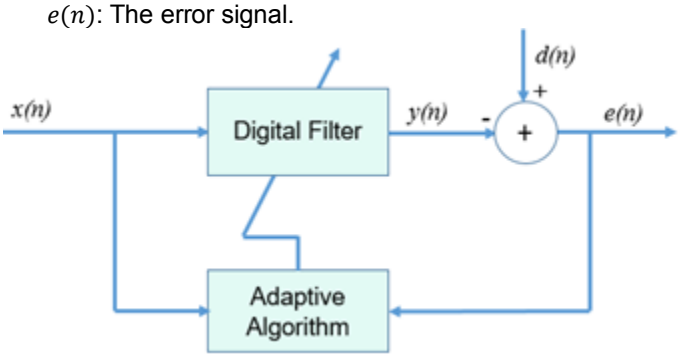

***Fig. 1:** diagram ofAdaptive Filter*

Digital adaptive filters can be classified according to their impulse responses:

- ✓ Finite, and in this case we speak of a finite impulse response FIR filter.
- ✓ Infinite, and in this case we speak of an infinite impulse response filter IIR.

The choice of the adaptive filter algorithm is based on several criteria, the most important of which are as follows:

- ✓ The optimization criteria: speed of convergence, complexity, robustness to noise ...
- ✓ The structure of the programmable filter.
- ✓ The type of signal processed: mono or multidimensional, real or complex ...
- ✓ The stability of the filter.

In the following, we will choose to work with the LMS algorithm based on their simplicity and performance.

2.2 About LMS Algorithm

The LMSalgorithm ("Least Mean Squares"), designed in 1959, is the most widely used adaptive algorithm[6]. It is based on the gradient method which calculates and updates the weights recursively[7]. It is shown that the error is a quadratic form of the weightings and, intuitively, the optimal solution is obtained by correcting the weighting vector step by step in the direction of the minimum[8]. In the following, we can see the standard LMS algorithm, where μ between 0 and 1 represents the adaptation step; the smaller μ is, the slower the convergence but the smaller the excess MSE; and vice versa.

Initialise by setting:

h(0)=0

For each time sample n=1, 2, … do the following:

$$\hat{x}(n) = h^T(n-1)\, y(n)$$

$$e(n) = x(n) - \hat{x}(n)$$

$$h(n) = h(n-1) + 2\,\mu\, y(n)\, e(n)$$

2.3 Arduino Due card

The Arduino Due is a programmable card based on a low-cost ARM Cortex-M3 microcontroller. It contains 54 digital pins (including 12 PWM), an 84 MHz clock, 4 UARTs, 12 analog inputs, a JTAG header Fig.2. [9]

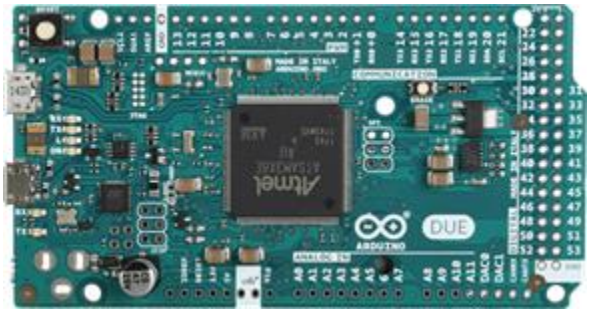

Fig. 2: *Schematic diagram of an Arduino Due card*

The analog input ports of this Arduino have a resolution of 10 bits (values in the range 0 to 1023), implying that an input voltage of 0V is as value 0, and an input voltage of 3.3V is represented as a value 1023. On the other hand, the DAC analog outputs have a resolution of 8 bits, with the option of operating with a 10 bit resolution the value of 0 creates an output potential of 0.55v and the value 1023 is transformed into an output of 2.77v Table 1.

Table1: *Technical specification of the Arduino Due*

Microcontroller	AT91SAM3X8E
Clock Speed	84 MHz
SRAM	96 KB
Flash Memory	512 KB
Digital I/O Pins	54
Analog Output Pins	2
Analog Input Pins	12
Operating Voltage	3.3V
Input Voltage (limits)	6-16V
Input Voltage (recommended)	7-12V
DC Current for 5V Pin	800 mA
DC Current for 3.3V Pin	800 mA
DC Output Current	130 mA

3 RESULTS AND DISCUSSION

3.1 Software Simulation

In this study, we used Matlab/Simulink to simulate the application of noise suppression in adaptive filtering fig. 3. The advantages of the proposed LMS algorithm are verified by simulation analysis. In the simulation, the original signal is in the form of a sine signal with a frequency of 500 Hz, and white Gaussian noise with an average value of 0 and an SNR of 5dB is input to the LMS adaptive filtering system. The results of the simulation are presented in Fig. 4below.

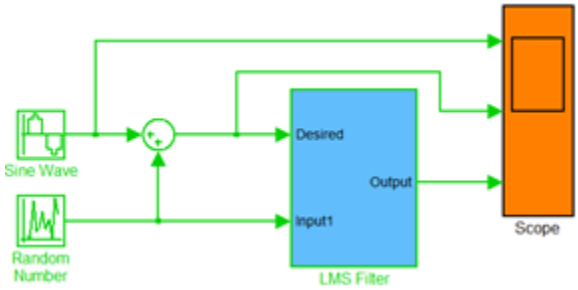

Fig. 3*: Adaptive filter LMS using Matlab function*

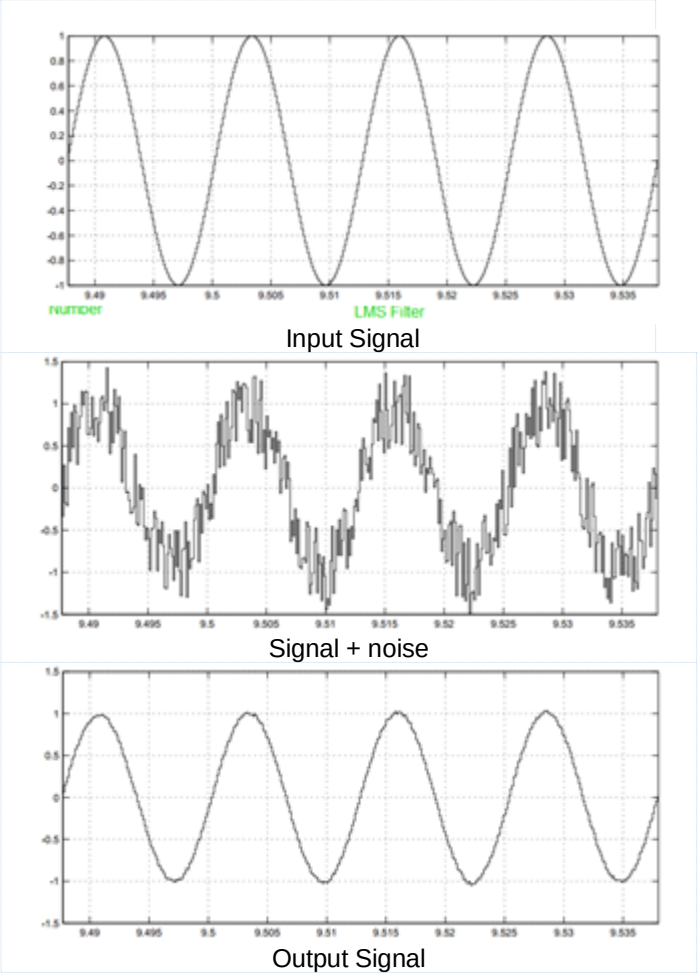

Fig. 4: *Simulation results of LMS algorithm*

3.2 Implementation and Experimental Results

In this part, the adaptive LMS algorithm will be implemented in an Arduino Due board. To test the implementation, a GBF generator has been used to generate the input signal x(n), an oscilloscope to display the output signals y(n) and e(n) Fig.5.

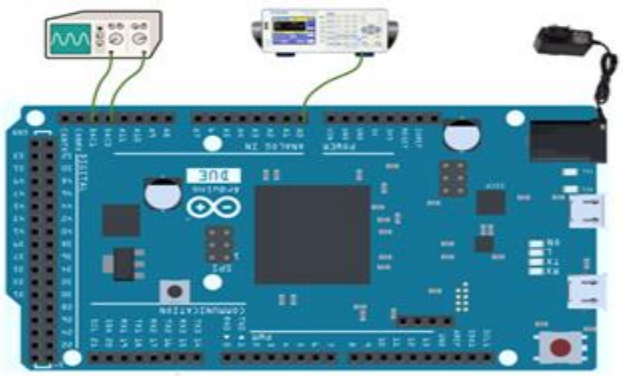

Fig. 5: *Hardware setup for Adaptive LMS algorithm*

To use the Arduino block in Matlab, Simulink Support Package Library for Arduino hardware must be installed,it can be obtained and installed by clicking on the Add-On on Matlab. When the installation is complete, the library can be added to the Simulink Library Browser as a Simulink support package for Arduino. Simulink needs to be configured and operated with Arduino Due. In the simulation options, the external model is enabled to run in real-time with the external hardware. When execution is requested, RTW converts the Simulink block diagram into C/C++ code, in order to compile and download the code into the microcontroller. The code is run in real-time with the chosen sampling time in the blocks and the selected signals are sent to Simulink via USB interface. Figure 6 show the diagram of the Real-Time Workshop [10],[11].

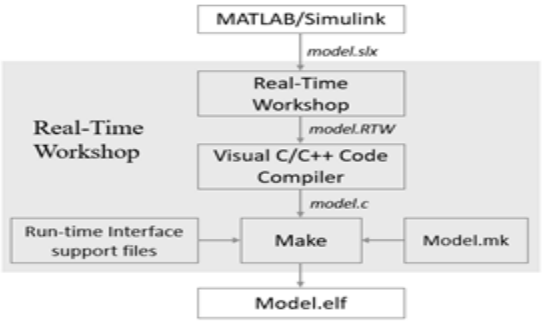

Fig. 6*: Flow diagram of the procedures for implementingthe model on Arduino Due*

Fig. 7 shows an adaptive real-time LMS algorithm design using the Arduino Due board with Matlab / Simulink.

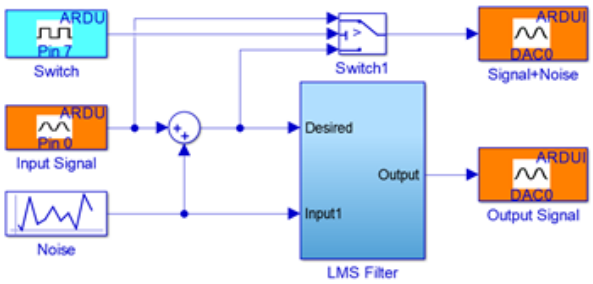

Fig. 7: *Model Real-time filtering system*

Our module realized on Simulink has been implemented in the Arduino Due board and their results are not shown in figure 8. Once the system was implemented, several measurements were made to analyze their performance. We note that this module is a real-time process and that the graphics are similar to those generated by the simulation in Matlab/Simulink.

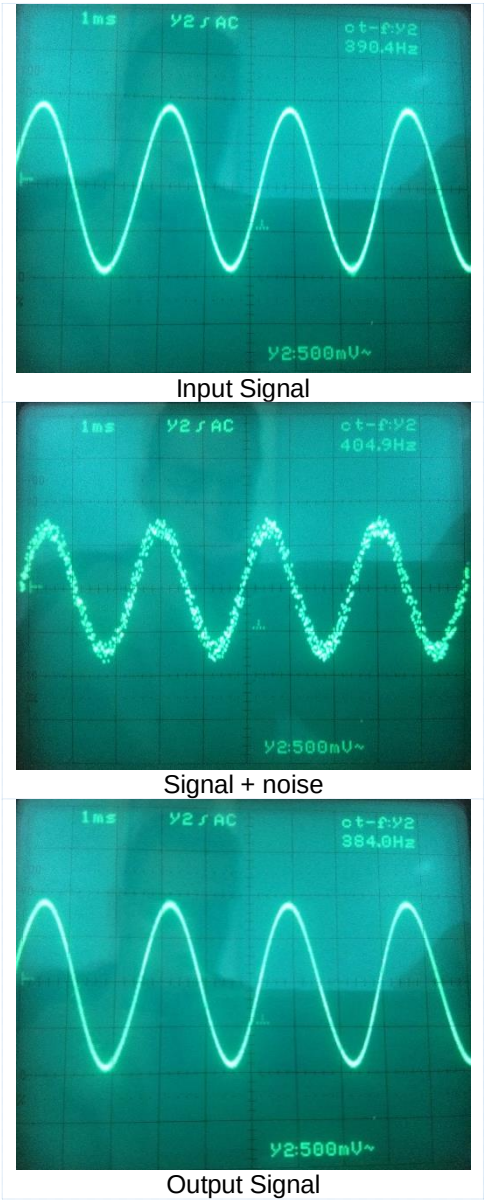

Fig. 8*: Result of Real time implementation using LMS algorithm*

4 CONCLUSION

Filters are an important part of a large number of projects. The adaptive filter is also widely used, so it is necessary to improve its adaptive algorithm. In this article, we have implemented a real-time adaptive LMS filter module in free and low-cost Arduino Due hardware, using a generator and an oscilloscope, we have obtained accurate results similar to those of Matlab. Due to the large numerical computation of the LMS algorithm, there are two major weaknesses. The first point is that this algorithm demanded a large processing time which caused the debounced signal to be delayed by about (5~ 7 ms) compared to the original signal. The second point is when the high-order FIR filter is used, which forced us to use a microcontroller (or microprocessor).

REFERENCES

[1] P. Regalia, Adaptive IIR Filtering in Signal Processing and Control. Routledge, 2018.

[2] C. Paleologu, J. Benesty, and S. Ciochină, “Adaptive filtering for the identification of bilinear forms,” Digital Signal Processing, vol. 75, pp. 153–167, Apr. 2018, doi: 10.1016/j.dsp.2018.01.010.

[3] J. Dhiman, S. Ahmad, and K. Gulia, “Comparison between Adaptive filter Algorithms (LMS, NLMS and RLS),” 2013.

[4] P. Kumar, H. S. Bhadauriya, A. R. Verma, and Y. Kumar, “Design Spline Adaptive Filter with Fractional Order Adaptive Technique for ECG Signal Enhancement,” Augment Hum Res, vol. 5, no. 1, p. 4, Oct. 2019, doi: 10.1007/s41133-019-0022-5.

[5] C. Venkatesan, P. Karthigaikumar, and R. Varatharajan, “FPGA implementation of modified error normalized LMS adaptive filter for ECG noise removal,” Cluster Comput, vol. 22, no. 5, pp. 12233–12241, Sep. 2019, doi: 10.1007/s10586-017-1602-0.

[6] D. Esposito, D. De Caro, G. Di Meo, E. Napoli, and A. G. M. Strollo, “Low-Power Hardware Implementation of Least-Mean-Square Adaptive Filters Using Approximate Arithmetic,” Circuits Syst Signal Process, vol. 38, no. 12, pp. 5606–5622, Dec. 2019, doi: 10.1007/s00034-019-01132-y.

[7] S. Dixit and D. Nagaria, “Hardware Reduction in Cascaded LMS Adaptive Filter for Noise Cancellation Using Feedback,” Circuits Syst Signal Process, vol. 38, no. 2, pp. 930–945, Feb. 2019, doi: 10.1007/s00034-018-0896-3.

[8] D. K. Gupta, V. K. Gupta, M. Chandra, A. N. Mishra, and P. K. Srivastava, “Hardware Co-Simulation of Adaptive Noise Cancellation System using LMS and Leaky LMS Algorithms,” in 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1–6, doi: 10.1109/IoT-SIU.2019.8777658.

[9] “Arduino Due | Arduino Official Store.” [Online]. Available: https://store.arduino.cc/arduino-due. [Accessed: 16-Feb-2020].

[10] “Arduino Support from MATLAB.” [Online]. Available: https://fr.mathworks.com/hardware-support/arduino-matlab.html. [Accessed: 16-Feb-2020].

[11] M. Mekhfioui, R. Elgouri, A. Satif and L. Hlou, « Arduino Due Implementation of an Algorithm for Blind Source Separation using Matlab Simulink», International Journal of Innovative Technology and Exploring Engineerin, vol. 9, no 2, p. 3692-3696, dec. 2019, doi: 10.35940/ijitee.B7385.129219.