1. INTRODUCTION

Blind Signal Separation (BSS) is a swiftly developing discipline in the signal processing community that gives an answer for isolating blended signals in the course of their propagation. The BSS, therefore, is composed in estimating a hard and fast of unknown source alerts from a fixed of located indicators which are mixtures of these source signals. With no additional hypothesis, the problem accomplished by the BSS is an incorrect problem posed. This is why all BSS strategies have assumptions about each source and mixing [1], [2]. Source separation strategies were carried out in numerous medical and technological fields, which include telecommunications, acoustics, imaging and biomedical sign processing. Previously, few studies are carried out on the real-time implementation of blind source separation. Yiu and Yong Low [3] proposed a real-time signal blindness noise reduction system exploiting the spatial diversity of source mixtures received by different sensors and kurtosis measurements. In 2019 Mekhfioui and all [4] implemented the SOBI algorithm on a DSP using MATLAB Simulink and TMS320C6713 DSK to validate its performance in real-time. This paper presents a new efficient algorithm to implement real-time blind source separation on the Arduino Due board without requiring detailed knowledge of programming languages. The concept is to use the Simulink Support Package Library for Arduino hardware, which integrates with the MATLAB-based graphical environment and does away with the need for programming. This work is structured as follows: Section II presents the principle of the blind source separation BSS and its mathematical model, the Arduino Due card, Section III presents the proposed work and the signals used. Section IV describes the implementation and results, and finally, Section IV concludes this paper.

2 MATERIAL AND METHODOLOGY

2.1 About Blind Source Separation

The Basic Source Separation (BSS) is based totally on estimating unknown source signals from their located combos when there is very little information to be had on the mixing model[5].

![Fig. 1: Principle of the separation of sources](image)

The basic model of the blind source separation (BSS) depends on the assumed mixing model, in the general case the equation of the instantaneous linear mixing is written as follows:

\[ x_i(t) = \sum_{j=1}^{n} a_{ij} s_j(t) + v_i(t) \]  

(1)

Where

\[ x(t) = [x_1(t),...,x_m(t)]^T \]

is the vector containing the observations;

\[ a_{ij} \]

is the mixing system;

\[ s(t) = [s_1(t),...,s_n(t)]^T \]

is the vector containing N signals emitted by N unknown sources;

\[ v(t) \]

is an additive noise vector we consider it negligible in the rest of the calculations.

The matrix form of equation (1) is written as follows:

\[ X(t) = HS(t) \]  

(2)

Where H is a mixture matrix of size \( n \times m \)

It is about finding in the ideal case the matrix W of size \( n \times m \) which inverts the mixture and provides the output vector:

\[ y(k) = Wx(k) = WHs(k) = s(k) \]  

(3)

The estimated sources are given by the vector \( s(k) \) and their corresponding projections for the different microphones are given by the estimated matrix: \( H = W^{-1} \).

2.2 Evaluation criteria

The performance of the algorithms of the Blind Source Separation BSS is evaluated via calculation of the Signal-to-
Interference Ratio (SIR). It is expressed in dB is defined for each source $s_j$ by [4]:

$$\text{SIR}_j = \max \left\{ \frac{10 \log_{10} \left( \left( \frac{\sum_{i=1}^{N} g_{ij}^2}{\sum_{i=1}^{N} K_i} \right) \right)}{10} \right\} \quad \forall j \in [1, N]$$ (4)

Where the values $g_{ij}$ ($i, j = 1, \ldots, N$) are the coefficients of the matrix of performances $G$ equal to the product of the matrices of mixture $A$ and of separation $W$, $G = A \ast W$.

Knowing that each report $\text{SIR}_j$ quantify the quality of source separations $\mathcal{S}$. We define the SIR, which quantifies the performance quality of the BSS algorithm used to separate all sources by:

$$\text{SIR} = \frac{1}{N} \sum_{j=1}^{N} \text{SIR}_j \quad (5)$$

More SIR, the higher the separation is good and the BSS algorithm used is efficient [4].

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. [6]

![Fig. 2: Schematic diagram of an Arduino Due card](image)

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 representing a value 1023 as value 0 or vice versa. On the other hand, the DAC analog outputs have a resolution of 8 bits, with the option of representing a value 1023 as a value 0 or 1.

3.1 Signal model

In our study, we have used two generators (GPF) to generate...
two basic signals ($s_1$ and $s_2$) of the same shape and different frequencies (800Hz and 400 Hz). These signals were combined linearly by a mixing matrix $A$ to produce the input signals $x_1$ and $x_2$ (in this article, we use the same mixing matrix to compare the different algorithms).

$$A = \begin{pmatrix} 0.35 & 0.86 \\ 0.63 & 0.42 \end{pmatrix}$$

4 RESULTSANDDISCUSSION

4.1 Implementation and Experimental Results

In this part, our separation method has been implemented in an Arduino Due card. To test the implementation, two GBF generators have been used to generate the input signals $x_1$ and $x_2$, an oscilloscope to view the output signals, and LEDs to indicate the index of the most efficient algorithm Fig.4.

![Hardware setup for system separation](image1)

**Fig. 4**: Hardware setup for system separation

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[15]. 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 5 show the diagram of the Real-Time Workshop [16].

![Flow diagram of the procedures for implementing the model on Arduino Due](image2)

**Fig. 5**: Flow diagram of the procedures for implementing the model on Arduino Due

Fig. 6 shows a design of our real-time blind source separation approach using the Arduino Due Board based on Matlab / Simulink.

![Model Real-time Separation of two signals](image3)

**Fig. 6**: Model Real-time Separation of two signals
In a real application, if the number of outputs is more than two, we can use switches linked with the Arduino board to define the output to be displayed on the oscilloscope. With our workstation, we are able to separate the input signals in real-time using our algorithm designed on Matlab/Simulink and the Arduino Due board. To observe the different signals (source signals, mixed signals, and estimated signals), we have to change the binary value of the link switches with the Arduino (Fig. 7). In addition, we have used the link LEDs with the Arduino board to display the index of the most efficient algorithm to separate our signals fig. 8. The equivalence between the index and the algorithm is described in table 2 above.

The implementation results show a behavior similar to that of the results obtained by simulation on the ICALAB [17], which confirms the performance of our method, with no need to use a computer or separation software.

5 CONCLUSION
In this work, we used the Matlab/Simulink blocks dedicated to real-time signal processing, to separate two mixed signals linearly by a grid A, using an efficient method based on the calculation of the Signal to Interference Ratio (SIR) of a group of algorithms inside the Arduino board. To validate this performance in practice and in real time, we implemented our method in the Arduino Due board, and thanks to two generators and an oscilloscope, and LEDs we obtained results similarly to those of Matlab.

REFERENCES


