Digital Filter Design Based on ARDUINO and Its Applications

Muharrem Çelebi [1]
[1] Kartepe Vocational and Technical Anatolian High School, Kartepe, Kocaeli.
muharrem.celebi@kocaeli.edu.tr

***Abstract*—Nowadays, due to the rapid development of digital platforms, digital signal processing has become realizable into small size processors. The purpose of this study is to test the filtering process on both PC-based platform and embedded system. The first goal of this study is to compare the results of the filtering process obtained in MATLAB and ARDUINO environments. For the second purpose, filtering process will be tried in real-time on ARDUINO platform with the produced filter coefficients. Findings obtained as a result of the study are presented and discussed.**

Keywords— FIR; IIR; filtering; MATLAB; ARDUINO

I. INTRODUCTION

The purpose of digital signal processing is to perform on digital systems instead of operations with analog circuits. In this way, it performs the same task based on software, without the need for electronic materials or replacement. This method ensures that the design is simple and upgradeable. For example, an analog filter made with RC elements, it is necessary to change the materials to change the filter's cutoff frequency. But if the same operation is to be done in digital filtering, only the filter coefficient should be changed. Despite the advantages of digital filters, analog filters are used in many areas, such as cross-over circuits in speaker cabinets.

The filters are systems where the desired signals are transferred to the output and undesired signals are suppressed. In filter design, if the materials used are made with elements such as R, C or L, it is defined as an analog filter. If filtering is done in a digital environment such as PC or microprocessor, it is called digital filter.

Various filter designs were realized. In design, the keyword is the cut-off frequency which means is decision point. Low-pass filters (LPF) are filters that allow the frequency values below the cut frequency to pass, and suppress the frequency above the cut frequency. High-pass filters (HPF) are filters that allow frequencies above the cut frequency to be transferred to the output, and not frequencies below the cut frequency. The band-pass filter (BPF) has two cut-off points. This filter transfers the frequency values at between two cut-off frequencies to output and other frequencies are damped. The band-stop filters (BSF) block the frequencies between the two cut-off frequency and transfer the other frequencies to the output. It is called as notch filter and used to suppress mains noise [1].

Digital filters are divided into two groups according to the impulse response. These are finite impulse response (FIR) and infinite impulse response (IIR). FIR filters have a linear phase response, so the group delay is the same for each input frequency. IIR filters have a nonlinear phase response and hence the group delay varies according to the applied frequency. Since FIR filters are always stable, it is possible to use them as an adaptive filter. The filter coefficients of FIR filters are many, so the process loads increase. However, IIR filters have a few filter coefficients, so, they have process loads low level [1].

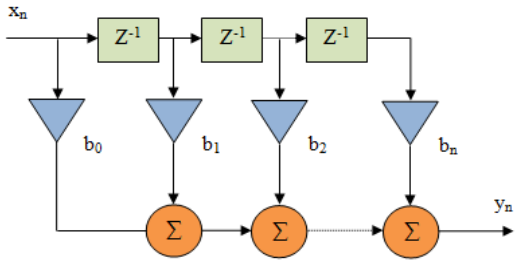

Fig. 1. FIR filter block diagram

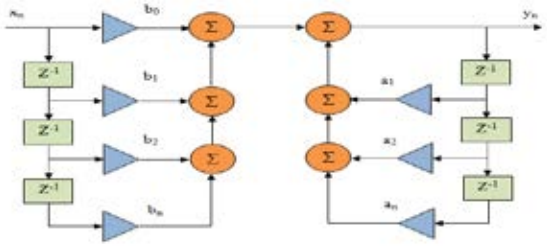

Fig. 2. IIR filter block diagram

In the Fig. 1 and Fig. 2 above, the x_n and y_n values represent the input signals and the output signals in a discrete domain. b_n and a_n represent filter coefficients. As can be seen in Fig. 1, the FIR filter multiplies at the past values of input signal multiplied by b_n filter coefficients then the result of each multiplication is added. In Fig. 2, the left side of the block diagrams is similar to the FIR filter, in addition to,

978-1-7281-8073-1/20/$31.00 ©2020 IEEE

multiplied at the past values of output signal by a_n filter coefficients then the result of each multiplication is added.

$$y(n) = \sum_{k=0}^{N} b(k) * x(n-k) \quad (1)$$

$$y(n) = \sum_{k=0}^{N} b(k) * x(n-k) + \sum_{j=0}^{N} a(j) * y(n-j) \quad (2)$$

Singh et al. [2], used the IIR filter to filter the ECG signal and remove the noise. Filter coefficients were produced using Matlab's FDA Tool and used the ATmega16A processor as the processor platform. Rahmatillah and Karim [3], designed a IIR Notch filter to reject 50 Hz power line noise. The algorithm were embedded to ARDUINO Due board. Al-Busaidi and Khriji [4], have builded a circuit that measures the ECG signal. Low-pass, notch ve band-pass filters were implemented in this structure. It was used ATmega32 processor. Tan and Jiang [5], presented teaching material for university students. In this material, for digital filter implementatirons, they underlined, sampling time, ADC tehcnique. they used 68HC12 microcontroller and assembley and C languages. Varshney and Tiwari [6], they designed finite impulse response (FIR) filter and implemented it in hardware as ATmega32 processor. Vostrukhin and Vakhtina [7], performed moving avarage filter using MATLAB and ARDUINO. Chiagunye et al. [8], in their study, low-pass, high-pass and band-pass digital filter were designed a circuit which is based on PIC16F877A microcontroller. Sharifi et al. [9], in their paper, designed an algorithm, for speech recognition based on AVR microcontroller. Runge et al. [10], in their work, they improved a system using FPGA for FIR filter. Mohammed and Ishak [11], created digital filter based on dsPIC30F6010a for motor driver. Koswatta and Karmakar [12], have builded a moving average filter work on PIC18F452 microcontrroler. It was used filtering RFID tag reader. Gargava et al. [13], established a system of Brain Computer Interfaces (BCI). In this system, EEG signals were filtered, then extracted the features. Pujari et al. [14], compared the performance of FIR filter onto two digital systems. This systems are microcontroller and FPGA.

II. Material and Methods

In this study, FIR_LPF, IIR_LPF, FIR_HPF, IIR_HPF experiments were carried out for 4 filters. These filters were calculated by the MATLAB program's filterDesigner tool. With the export command, b_n and a_n coefficients are produced and transferred to the ARDUINO environment. MATLAB-ARDUINO can easily communicate with a single cable. This allows data and results to be transferred quickly.

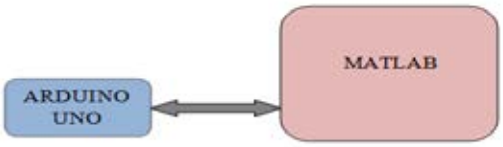

Fig. 3. Experiment connection - 1

In this study, two experimental connection are used. The first experiment is offline filtering and the second experiment is real-time filtering. Fig. 3 shows the structure of the experimental connection - 1. First, a signal is artificially generated in the MATLAB environment. The sampling period is chosen as fs = 3000 Hz. The components of this signal have been added to 500Hz, 1000Hz and noise. 300 points of this artificial signal produced are embedded in ARDUINO. After this process, the filter coefficients are embedded in the ARDUINO. All codes and coefficients are executed in ARDUINO board and the results are sent to MATLAB environment.

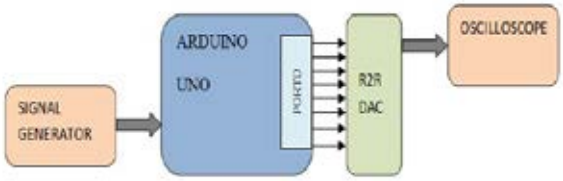

Fig. 4. Experiment connection - 2

Fig. 4 shows the structure of the experimental connection - 2. The ARDUINO has an ADC inbuilt, however it doesn't has DAC module inside. That is why, DAC circuit made using R2R resistor is soldered and connected with PORTD. In the input circuit, sampling is done using the A0 pin. At the output, an 8-bit output value is generated via PORTD. Pull-up and pull-down connections are performed on the A0 pin with two 10Kohm resistors. In this way, the input signal is kept constant at 2.5V DC level. In addition, with the 1uF capacitor, the ARDUINO is insulated with the signal generator. In sampling, it is carried out in 10-bit resolution. However, filtering analysis and DAC operations are performed at 8-bit resolution.

Each FIR filter and IIR filter are designed with a 5th degree filter. Sampling frequency is decided as 3000Hz. The parameter values of the FIR filter are fpass = 750Hz, fstop = 1000 Hz. For the IIR filter, fc = 750Hz is tuned. The response of the filters obtained in accordance with these values are presented in Fig. 5.

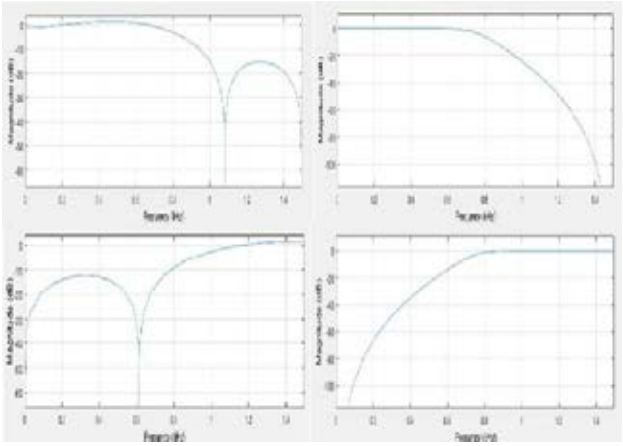

Fig. 5. FIR and IIR filter response

III. Experimental Results

Fig. 6 depicts the result graphics obtained after the first experiment. In MATLAB environment, a test signal has been artificially generated. Then, this sign was first filtered in MATLAB environment and illustrated sequentially in the right column. After this process, the test signal and filter coefficients are embedded in ARDUINO. The results obtained after this process are sent and illustrated in MATLAB environment. When the filtering results are compared with, the results of the operations performed in both MATLAB and ARDUINO environment give similar results.

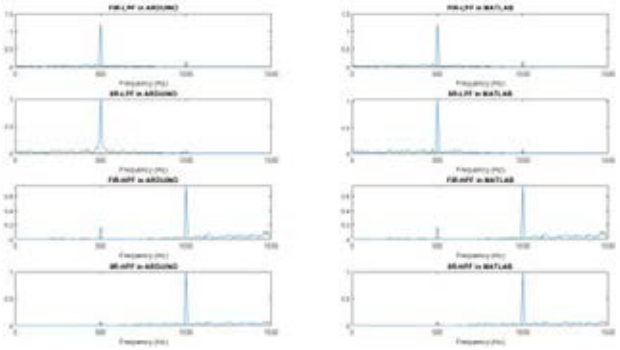

Fig. 6. Experiment - 1 filtering results

In the second stage, real-time filtering process is started. One of the most important problems here is the correct adjustment of the fs value of the processor. In the ARDUINO command list, there is a command that directly adjusts the fs value. With ADCSRA register, the sampling rate can be adjusted. As a starting point, the sampling frequency values of this register should be determined. Therefore, 100 sampling were performed on a 100-element array. Before and after this sampling process, time measurement was done using the micros () command. Then, the actual fs value was calculated by measuring the elapsed time in between two time points. This process was calculated for ADPS0, ADPS1, ADPS2 prescaler values into ADCSRA registers. The measured fs values are presented in Fig. 7.

```
                                                    // Maximum Sampling Frequency
ADCSRA |= bit (ADPS0);                              //       333333  Hz
ADCSRA |= bit (ADPS1);                              //       200000 Hz
ADCSRA |= bit (ADPS0) | bit (ADPS1);                //       125000 Hz
ADCSRA |= bit (ADPS2);                              //        66666 Hz
ADCSRA |= bit (ADPS0) | bit (ADPS2);                //        35714 Hz
ADCSRA |= bit (ADPS1) | bit (ADPS2);                //        17857 Hz
ADCSRA |= bit (ADPS0) | bit (ADPS1) | bit (ADPS2);  //         8928 Hz
```

Fig. 7. Maximum sampling frequency

When above Fig. 7 is examined, the minimum and maximum sampling intervals of ARDUINO are displayed. It can sample between 333kHz and at least 8.9kHz. To provide the fs = 3000 Hz we need, the delay cycle is performed with the while loop when each sample is obtained. The codes used for this process are displayed in Fig. 8.

```
for(int i=0; i<SAMPLES; i++)
{
  start_time = micros();
  analog_value[i] = analogRead(analog_pin);
  while(micros() < (start_time + sampling_time*9)){
    //wait to equal the Fs period.
  }
}
```

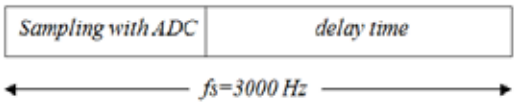

Fig. 8. Sampling and delay code

Another important point is that the processing time of the FIR and IIR library used should be calculated. Improved ARDUINO-Filters library [15] was used for filtering. The results of these measurement times are presented in Table 1.

TABLE I. Processing Times of Filters

FIR_LPF	IIR_LPF	FIR_HPF	IIR_HPF
145 µs	251 µs	146 µs	249 µs

If the filter degree is increased, the processing time to the filter increases. This increases the total Ts duration and the Fs value decreases. The filtering process is insufficient in reducing the degree of filter. Therefore, in this study, 5th order filters were designed for each filtering process.

Table I. When examined, the highest processing time is about 250 µs IIR filter. In addition to this time, the sampling process, output_PORTD and delay_time must be included in processes. For this reason, the total processing time is about 300 µs. When this time is reversed, the maximum real-time fs is 3300 Hz. As a result, fs = 3000 Hz was chosen.

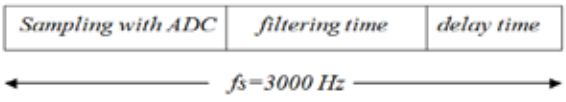

Fig. 9. Total processing time

However, as seen in Fig. 7, although the sampling values are high, the fs value decreases due to the inclusion of other processing times (filtering time, output_PORTD etc.).

The sine signals, whose amplitude is 4.2Vpp and the frequency values are 500Hz, 1000Hz and 1400Hz respectively, are produced from the signal generator. These signals are applied to the A0 pin, respectively. The filtered sign obtained was exported by PORTD with DAC circuit. Both input signs and filtered signs are displayed in Fig. 10.

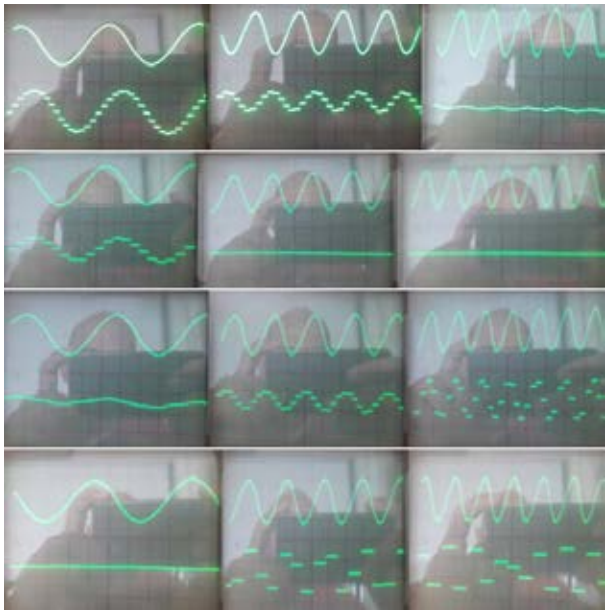

Fig. 10. Oscilloscope images of signals

12 pictures seen in Fig. 10 have been added. In each picture, the input signal and output signal are drawn. The top signal is the input sign, the bottom signal is the output sign. 500 Hz, 1000Hz and 1400Hz filtered signs are displayed from left to right. FIR_LPF, IIR_LPF, FIR_HPF, IIR_HPF results are listed top to below in this field.

When examining Fig. 10, in LPF processes, when it will go towards high frequency, the output amplitude value decreases. Similarly, in HPF processes, the output amplitude decreases as we go towards low frequencies. When FIR and IIR filters are compared among themselves, they are 5th degree in both filters. However, the ratio of the IIR filter to the compression of the signals is better.

IV. Conclusion

In filter design, the exact determination of the Fs value is important for the accuracy of the processes. However, in a real-time process, setting the Fs value incorrectly, or not being able to set it, will result in false results. There is no need for such an adjustment in the experiments carried out in the PC environment.

The results obtained in this study showed that when MATLAB and ARDUINO results are compared, they give similar results. As a result of the results obtained in the second part, the digital filters has been successfully realized in real-time. High sampling frequency can be reached with faster ARDUINO platforms.

References

[1] S. Ertürk, "Digital Signal Processing", Birsen Publishing, Istanbul, Turkey, 2005.

[2] N. Singh, S. Ayub and J. P. Saini, " Design of digital IIR filter for noise reduction in ECG signal", 5th International Conference and Computational Intelligence and Communication Networks IEEE, pp. 171-176, 2013.

[3] A. Rahmatillah and Ataulkairm, " IIR digital filter design for powerline noise cancellation of ECG signal using arduino platform", International Conference on Physical Instrumentation and Advanced Materials, Journal of Physics, 2017.

[4] A. M. Al-Busaidi and L. Khriji, "Digitally filtered ECG signal using low-cost microcontroller", International Conference on Control, Decision and Information Technologies (CoDIT), 2013, pp.258-263.

[5] L. Tan, J. Jiang, "Teaching Digital Filter Implementations Using the 68HC12 Microcontroller", American Society for Engineering Education (ASEE), 2011.

[6] V. Varshney and M. Tiwari, "Realization of an FIR filter using ATMEGA32 microcontroller", IEEE International Conference on Emerging Trends in Computing and Communication Technologies (ICETCCT), 2017.

[7] A. Vosrukhin and E. Vakhtine, " Studying Digital Signal Processing on Arduino Based platform", Proceedings of the 15th International Scientific Conference on Engineering for Rural Development, Jelgava, Latvia, 2016.

[8] T. Chiagunye, I. Somtoochukw and E. A. Okereke, "Designing a Microcontroller-Based Low-Pass, High-Pass and Band-Pass Digital Filter with Op-Amp", Journal of Multidisciplinary Engineering Science and Technology (JMEST), Vol. 2, issue 5, 2015.

[9] O. Sharifi, A. Zaeri and E. Ghafarioun, "Design and Implementation of Simple Speech Recognition System Based-on Digital Filters in AVR Microcontroller", Majlesi Journal of Multimedia Processing, vol. 1, 2012.

[10] C. R. Runge, M. Cerqueira and F. J. Arnold, " Introducing programmable logic devices in physics laboratories: a practical guide for the implementation of experiments", Revista Brasileira de Ensino de Fisica, 2020.

[11] M. F. Mohammed and D. Ishak," Improved BLDC motor performance with digitally filtering back-EMF using dsPIC30F microcontroller", IEEE Student Conference on Research and Development (SCOReD), 2009.

[12] R. Koswatta and N. C. Karmakar, " Moving average filtering technique for signal processing in digital section of UWB chipless RFID reader", Asia-Pacific Microwave Conference, 2010.

[13] P. Gargava, K. Sindwani and S. Soman, "Controlling an arduino robot using Brain Computer Interface",3rd International Conference on Reliability, Infocom Technologies and Optimization, 2014.

[14] S. Pujari, A. Yeotkar, V. Shingare, S. Monin and B. Kokare, "Performance analysis of microcontroller and FPGA based Signal Processing a case study on FIR filter design and implementation", International Conference on Industrial Instrumentation and Control (ICIC), 2015.

[15] website : https://github.com/tttapa/Filters