Rapid Prototyping of DSP Code for PC-Based Multi-Function Oscilloscopes

Fabrizio Russo and Francesco Travain

Abstract — This paper proposes a new approach to the implementation of digital signal processing (DSP) algorithms for personal computer (PC)-based multi-function oscilloscopes. These recently introduced devices are compact and low-cost measurement systems that typically combine a multi-channel data acquisition unit, an arbitrary waveform generator and a USB interface to the PC. Many measurement applications can be addressed with the development of specific software. The target of our approach is the user who has only a basic knowledge of computer programming and does not want to spend time on a long and expensive training with dedicated development environments. To address this issue, a novel software shell is proposed where user interface and basic functions (for signal acquisition, processing and display) are already built-in and specific DSP functions can be easily added. Application examples are reported to show the advantages of the proposed approach.

Keywords— Measurement systems, digital oscilloscopes, virtual instruments, digital signal processing, signal prefiltering, bilateral filter.

I. Introduction

Nowadays, digital signal processing (DSP) plays a role of paramount importance in electrical and electronic engineering. For instance, digital filters for data denoising are widely adopted in consumer electronics, medical imaging, remote sensing, electronic instrumentation etc. [1-6]. In the field of measurement systems (that is the subject of this paper), DSP techniques have rapidly become the core of modern instrumentation [7-19]. In this framework, personal computer (PC)-based instruments are very likely to represent the most versatile architecture for implementing and experimenting DSP algorithms in a measurement system. Multi-function USB PC oscilloscopes are an interesting example of this class of instruments, where the measurement hardware (connected to the PC by means of a USB interface) includes a multi-channel data acquisition unit and an arbitrary waveform generator. A typical block diagram is shown in Fig.1. After amplification/attenuation (upper blocks), the input signals are processed by subsequent modules including multiplexer, low-pass (LP) antialiasing filter, sample/hold (S/H) circuit, analog to digital (A/D) converter and a dedicated memory (MEM) for storing the converted data. The signal generation unit (lower blocks) typically includes waveform memory, digital to analog converter (D/A), low-pass filter and output circuits. All the operations are controlled by a microcomputer/microcontroller unit (μC) that also manages the USB interface. Due to their ability to generate and acquire test signals, multi-function USB PC oscilloscopes offer a very compact and effective solution to many different measurement problems. For this purpose, manufacturers generally provide software libraries and/or development kits for controlling the measurement hardware. Clearly, producing specific software for this class of instruments does not represent a problem for developers with expertise in computer programming. Conversely, this is a very critical issue for users having a limited knowledge of programming languages and that cannot spend time (and money) on a long training with dedicated development environments.

In order to address this issue, a novel user-friendly approach is presented in this paper. The approach consists in the development of a software shell where user interface and basic functions (for signal acquisition, processing and display) are already built-in and specific DSP functions can be easily added by non-expert users. We chose the Picoscope 2204A [20-21] to test our approach, because performance, flexibility and low-cost make this device an ideal candidate for education applications that are an important target of our work. The proposed software shell, called *DSPrototyper*, is written in C++ using Microsoft Visual Studio. It includes all the necessary files to automatically manage the graphical user interface commonly adopted in Windows-based applications. The final user is only requested to have a minimal knowledge of C language and to write the source code of the desired DSP algorithm in the (empty) body of a pre-defined C function.

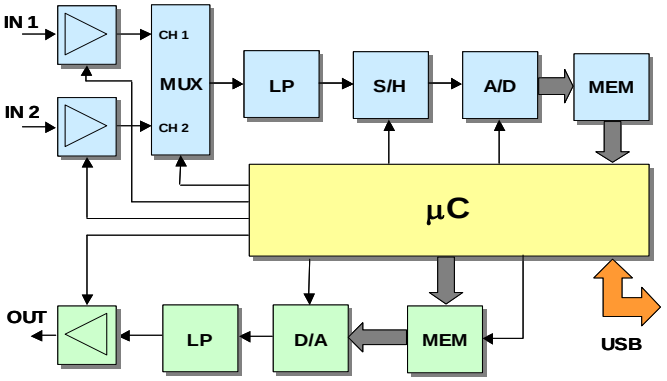


Fig.1 – Example of block diagram of a two-inputs multi-function USB PC oscilloscope.

F. Russo and F. Travain are with the Department of Engineering and Architecture, University of Trieste, Via A. Valerio 10, I-34127 Trieste, Italy (e-mail of the corresponding author (F. Russo): rusfab@univ.trieste.it).

Fig.2 – Development environment running on a notebook interfaced to a multi-function oscilloscope. (The output of the signal generator is connected to the input channel A).

Each function can be easily called by pressing the corresponding button provided in the soft-panel. Application examples are reported in the paper to show the advantages of the proposed approach. The rest of the paper is organized as follows. Section II describes the proposed software shell, Section III presents some application examples, and finally, Section IV reports the conclusions.

II. THE PROPOSED SOFTWARE SHELL

Fig.2 shows an example of experimental set-up including the Picoscope 2204A connected to a Windows-based notebook where Visual Studio is running. The main operations of *DSPrototyper* are listed in Fig.3. They include basic functions (for signal generation, acquisition and display), built-in DSP functions and specific user-defined DSP functions that represent the key feature of the proposed approach. The soft-panel representing the user-interface is depicted in Fig.4. The largest area (left-middle part) is devoted to the graphical representation of waveforms. All commands are located in the right side of the panel. The settings for signal acquisition address channel selection, range, timebase and AC/DC coupling. The settings for signal generation include the choice of the fundamental frequency, the waveform type (sine, square, triangle, ramp), amplitude and offset. A special option (arbitrary waveform) allows the user to define a periodic signal by choosing the fundamental frequency, order and relative amplitude of additional harmonics. The first group of built-in DSP functions (*measurements*) yield traditional measures of frequency, period, average and peak-to-peak values of the acquired signal The second group of built-in DSP functions is devoted to FFT-based spectral analysis. In addition to the rectangular window, other choices are available for reducing the spectral leakage such as the well-known Hanning, Hamming and Blackman-Harris windows [22]. The third group of pre-defined DSP functions deals with noise filtering.

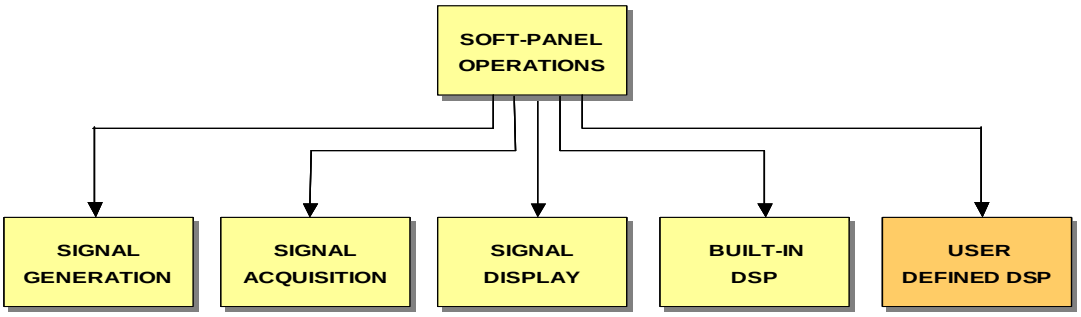


Fig.3 - The proposed software shell includes basic functions (for signal generation, acquisition and display), built-in DSP functions and special user-defined DSP functions.

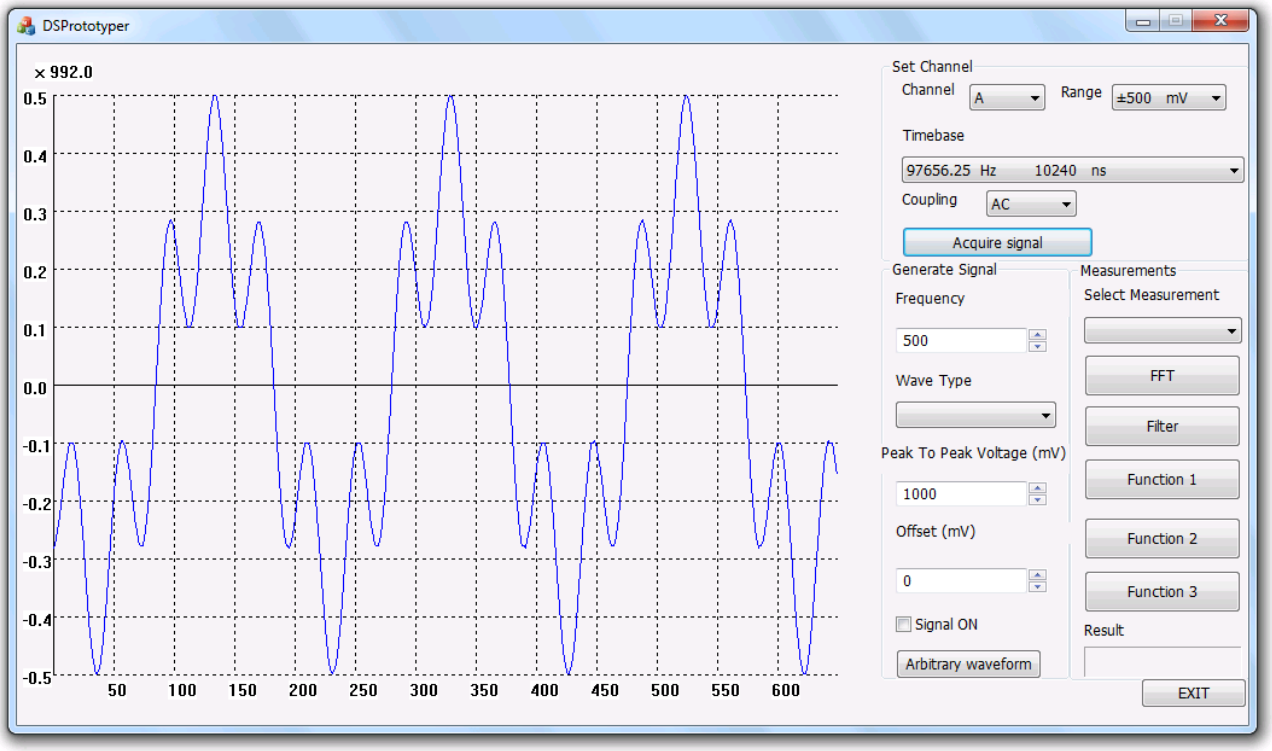


Fig.4 – Soft-panel of the proposed DSPrototyper showing the waveform of an electrical signal (fundamental and fifth harmonic) generated and then acquired by the multi-function oscilloscope.

The available methods include linear and nonlinear operators for the removal of Gaussian and impulse noise such as the moving average, the sigma filter, the median and the alpha-trimmed mean filter [23]. Due to the availability of the mentioned functions, the developer can directly focus on the experimentation of novel algorithms without being deceived by problems of user-interface and hardware control. The already implemented DSP functions can also offer a first comparison for testing new methods. The user-defined DSP algorithms are executed by pressing the appropriate software buttons. In the present release, three specific functions are provided, namely Function 1, Function 2 and Function 3 (Fig.4). The procedure for implementing new algorithms is very simple. Only a basic knowledge of C programming language is needed. As shown in Fig.5, the software shell is a Visual Studio project composed of many computer files. Most source files are devoted to the graphical user interface that is based on dialog boxes and the related controls. In our approach, we dedicated only one source file (MOD_DSP.cpp) to the implementation of DSP algorithms. This file groups all the mentioned built-in DSP functions and the user-defined functions as well. Although MOD_DSP.cpp is formally a C++ file, it can contain ANSI C source code, due to the well-known compatibility between C and C++. It is worth pointing out that the user is requested to interact with this file only. He/she should write the plain C source code of the desired algorithm in the empty body of one of three predefined functions included in this file and then simply recompile/rebuild the overall project without taking care of all remaining files. The procedure for passing the parameter values is also very simple. A specific function ("InputParameterWithName") has been provided to make this operation quick and user-friendly, as it will be shown in the next section.

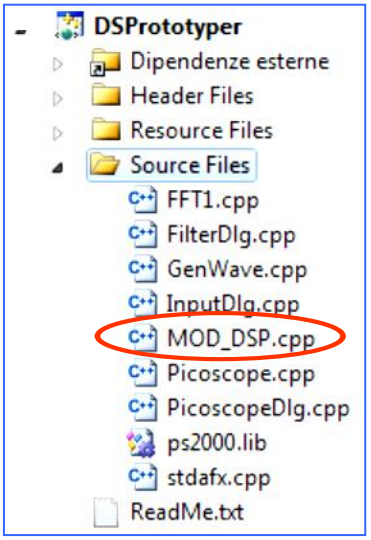


Fig.5 – Source file structure of the software shell: MOD_DSP is the file that includes all the DSP code.

```
void Bilateral(int N, double Sr, double Sd,int len,double input[],double output[]){
    int i,j;
    double qd,qr,nval,cval,wd,wr,sum,wsum;

    for(i=N;i<len-N;i++){
        cval=input[i];
        sum=0.0;
        wsum=0.0;
        for(j=-N;j<=N;j++){
            nval=input[i+j];
            qd=j*j;
            wd=exp(-qd/(2.0*Sd*Sd));
            qr=(cval-nval)*(cval-nval);
            wr=exp(-qr/(2.0*Sr*Sr));
            sum=sum+wr*wd*nval;
            wsum=wsum+wr*wd;
        }
        output[i]=sum/wsum;
    }
    for(i=N;i<len-N;i++)input[i]=output[i];
}

 ////////////////////////////// SPECIAL FUNCTIONS ////////////////////////////////
double Function1(){
    double funpar1=InputParameterWithName("BILATERAL - choose N ");
    double funpar2=InputParameterWithName("BILATERAL - choose Sr");
    double funpar3=InputParameterWithName("BILATERAL - choose Sd");
    Bilateral(funpar1,funpar2,funpar3,sigsize,isig,osig);
    return 1;}
```

Fig.6 – C source code defining and calling the bilateral filter.

III. SOME APPLICATION EXAMPLES

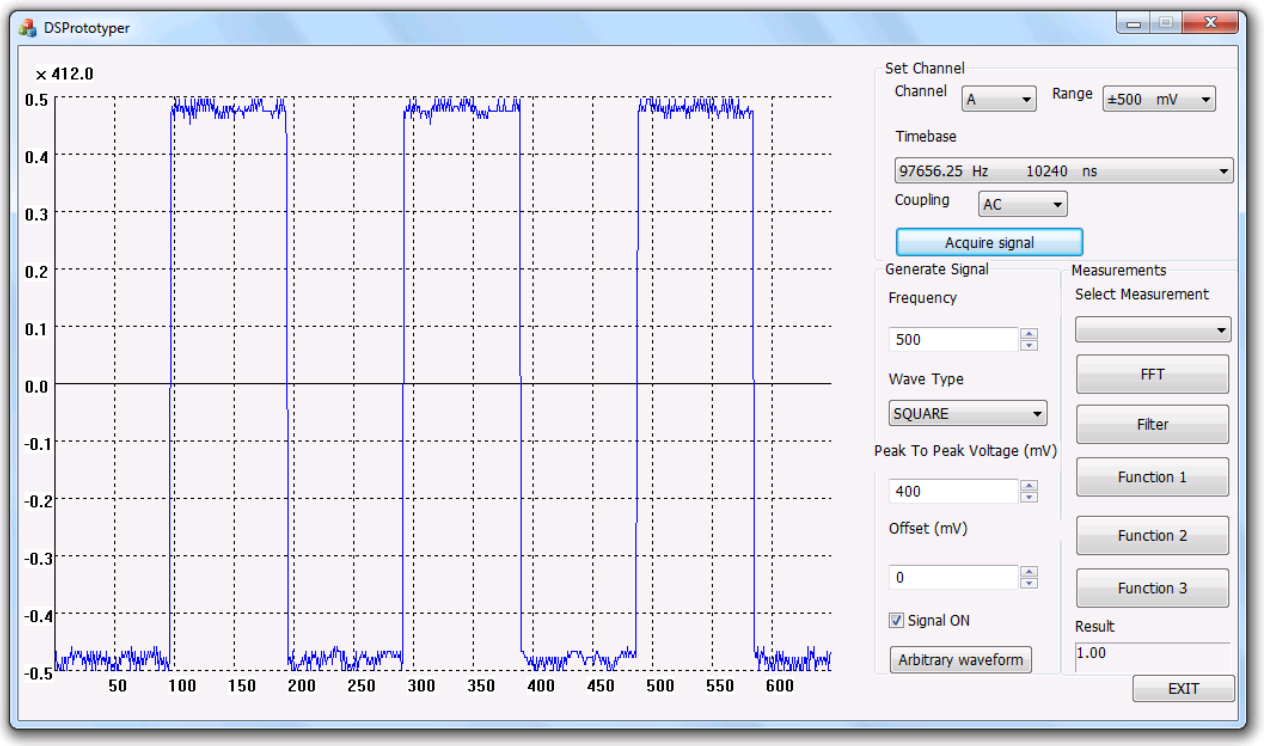
In this section we shall describe the step-by-step procedure for implementing a new pre-filtering function. For this purpose we chose the bilateral filter [24]. It is a well-known nonlinear operator that is specifically designed to reduce short-tailed noise distributions (e.g., Gaussian and uniform noise) while preserving the signal information. A study of its accuracy has been recently presented in [25] for the case of 2D signals. Formally, let x(n) be the sample at time n in the (noisy) input signal. Let us consider the neighboring samples belonging to a time window of 2N+1 elements centered on x(n). The bilateral filter is a nonlinear weighted average operator whose weights depend upon the time distance and intensity distance with respect to the central sample. The output y(n) of this filter is defined by the following relationships:

$$y(n) = \frac{\sum_{k=-N}^{N} w_d(k)\, w_r(n,k)\, x(n-k)}{\sum_{k=-N}^{N} w_d(k)\, w_r(n,k)} \qquad (1)$$

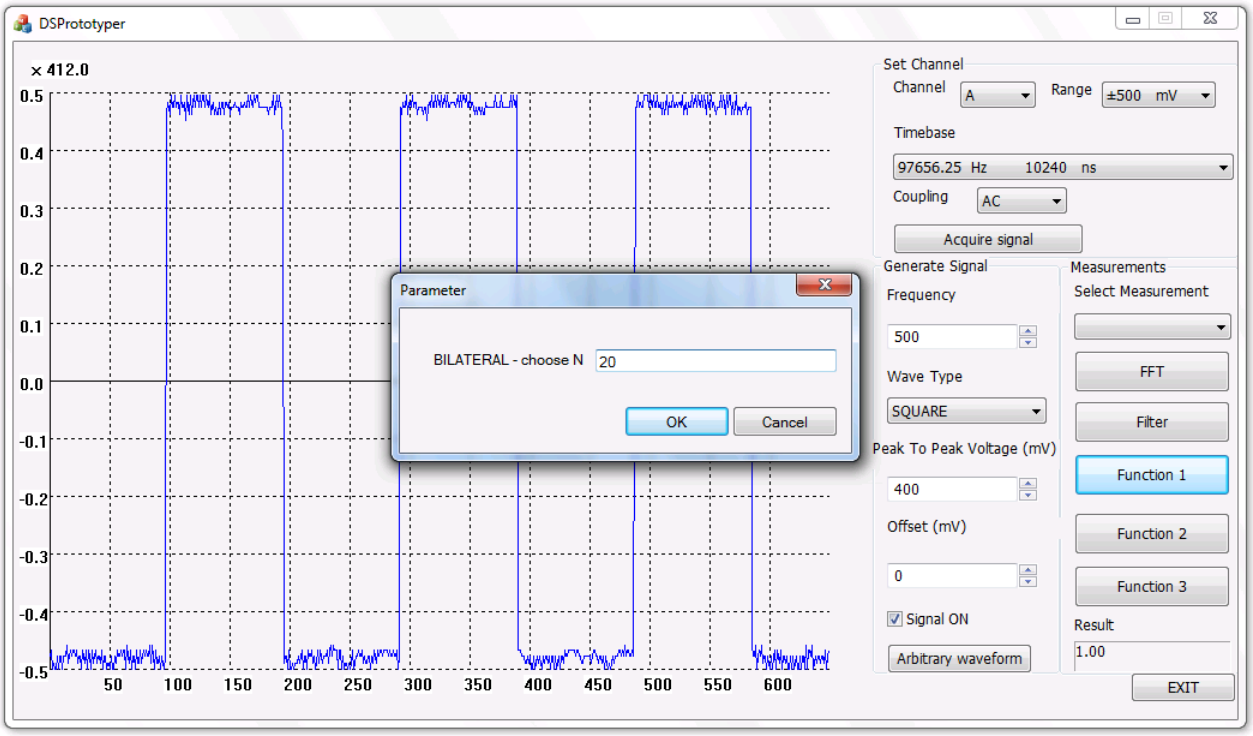
$$w_d(k) = e^{-\frac{k^2}{2S_d^2}} \qquad (2)$$

$$w_r(n,k) = e^{-\frac{[x(n-k)-x(n)]^2}{2S_r^2}} \qquad (3)$$

The filtering behavior is controlled by the parameters S_d and S_r. The source code of the corresponding C function "Bilateral" can be written in the MOD_DSP.cpp file (Fig.6), where "len" is the number of samples to be processed, "input[]"and "output[]" denote C arrays. If we want to activate the filtering by pressing the button "Function 1", we only need to insert a few statements in the (originally empty) body of the specialized C function "Function1", as shown in Fig.6. The first three operations ask the user for the parameter values, namely funpar1, funpar2 and funpar3. Then, the previously defined function "Bilateral" is called. At this point the overall project is recompiled and nothing else is needed. The request for a parameter value is performed by the special function "InputParameterWithName": it automatically opens a dialog box (printing the parameter name) and waits for the input value from the user. An application example is proposed in



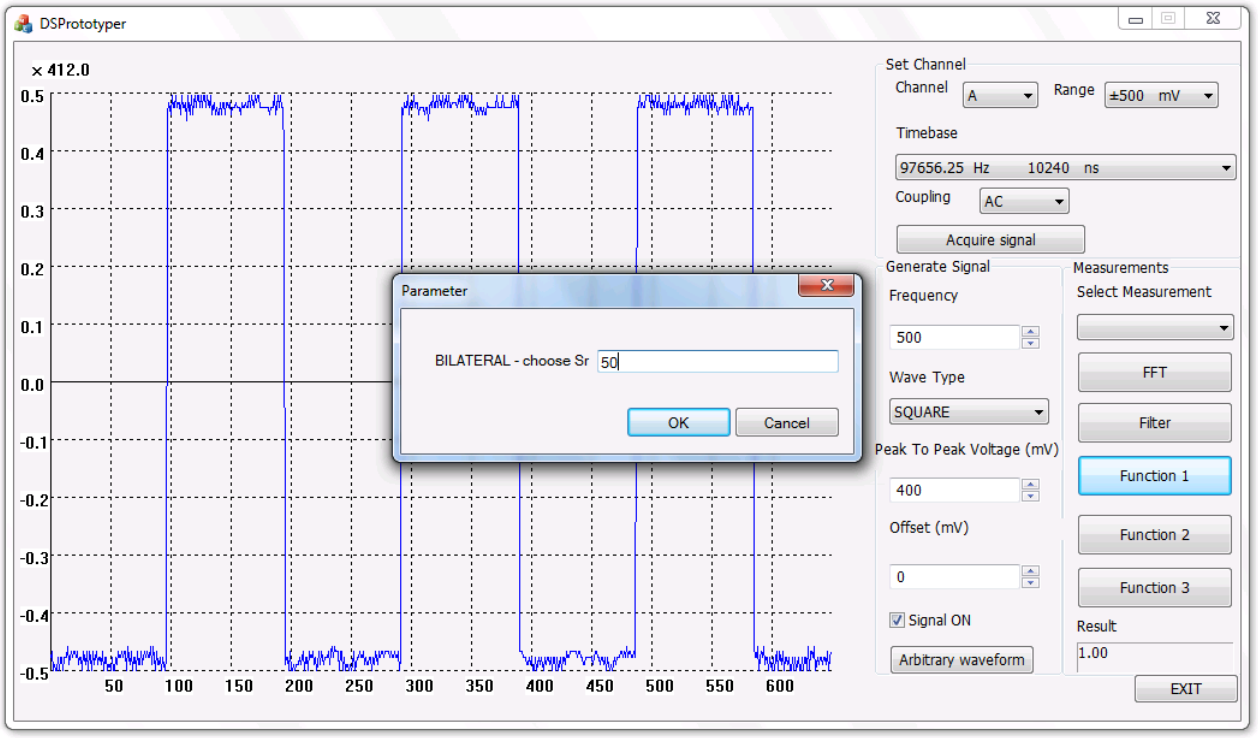
(a)



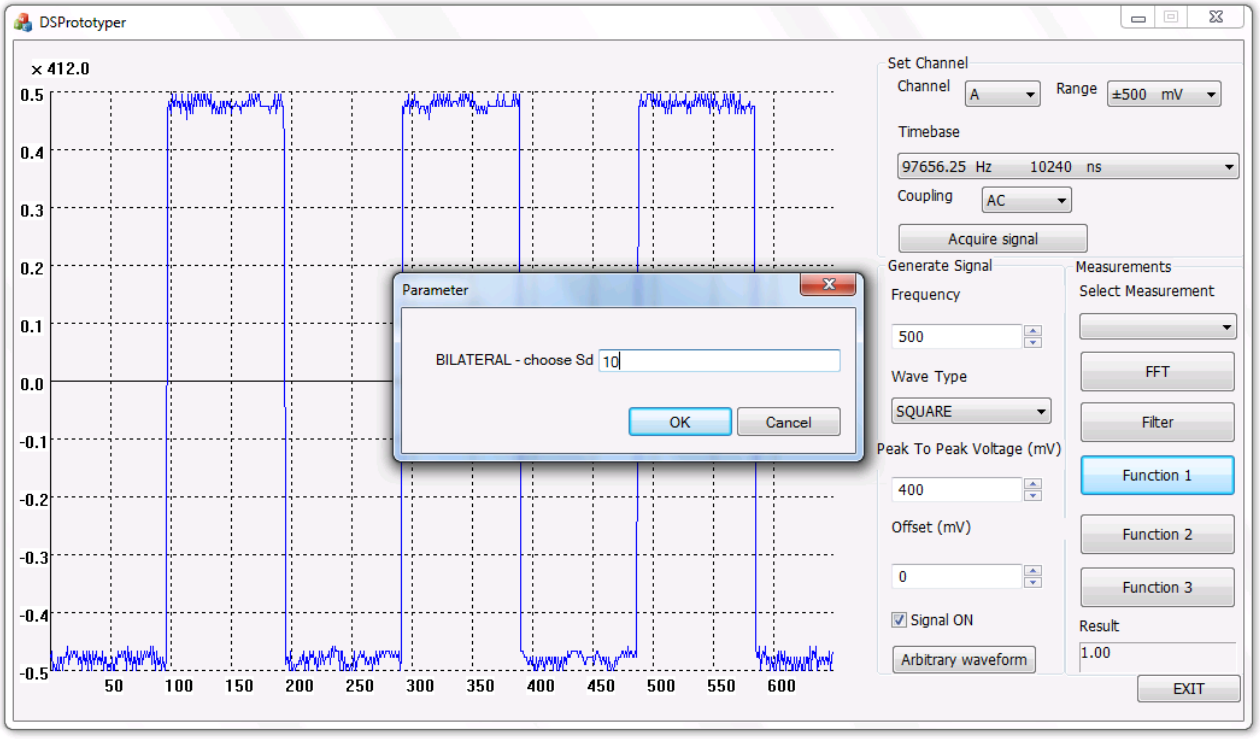
(b)

Fig.7 – Example of nonlinear filtering of a noisy input signal: acquisition (a) and choice of the parameter N (b).

Fig.7a that shows the acquisition of a noisy square wave. (In this case we did not use a shielded cable: we injected some noise on the wire that connects the output of the signal generator to the input channel A of the oscilloscope). By pressing the soft button “Function 1” the filtering procedure is executed. First, three parameter values are requested (Fig7b, 8a and 8b). Then, the bilateral filter is applied to the noisy data. The resulting waveform is depicted in Fig.9. We can



(a)



(b)

Fig.8 – Example of nonlinear filtering of a noisy input signal: choice of the parameters S_r (a) and S_d (b).

notice that the noise has been significantly reduced whereas rising and falling edges of the waveform have been satisfactorily preserved. In this example, the new function needed some parameter settings without returning any value. In case the implemented function should return a value, the following procedure must be adopted. Let us suppose we want to measure the root mean squared (RMS) value of a periodic signal. A (digital) measurement procedure could encompass

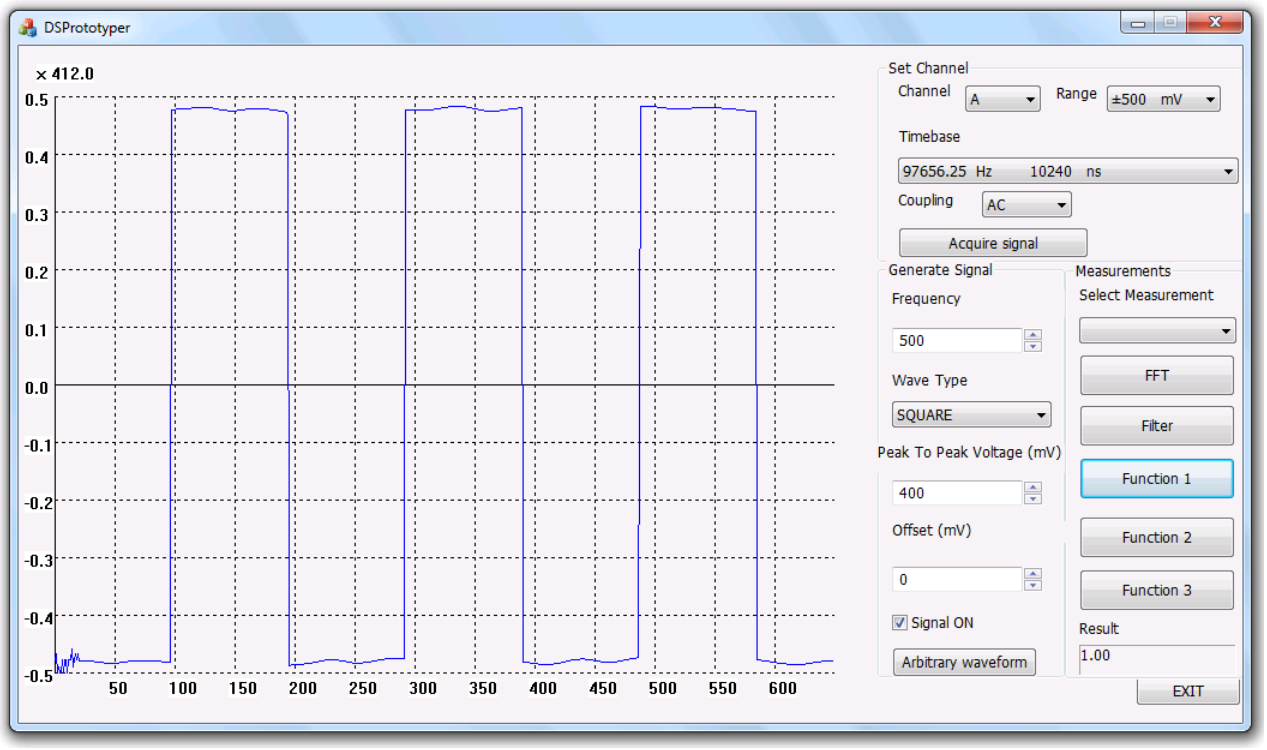


Fig.9 – Filtered signal.

some preliminary operations such as the estimation of the period. Let "Measure_RMSvalue" be the C function that performs the overall task (a detailed description of the algorithm is out of the scope of this work). The source code for calling this function has been added to the (originally empty) body of the specialized "Function2" (Fig.10). This function returns the result of the measure and automatically plots it in the pre-defined rectangular area ("Result") of the soft-panel designed to display any output data (right-bottom corner of the window). For instance, let us generate a sine wave with frequency of 500 Hz and peak-to-peak amplitude of 1V. The acquired signal is shown in Fig.11. By pressing the button "Function 2", the RMS value is first computed and then shown in the output box: the result is 349.55 mV. This measure is in good agreement with the theoretical value (353.55 mV): the relative error is 1.1%. Clearly, more sophisticated measurements could be performed. In any case, the built-in functions for parameter input/output allow the user to focus on the DSP problem without worrying about the graphical interface.

IV. Conclusions

A novel user-friendly approach to rapid prototyping of DSP code for multi-function PC oscilloscopes has been presented. The proposed approach is dedicated to users that are not willing to invest time and money for a long training with development environments dedicated to instrumentation.

```
////////////////////////////// SPECIAL FUNCTIONS ////////////////////////////////
double Function1(){
    double funpar1=InputParameterWithName("BILATERAL - choose N ");
    double funpar2=InputParameterWithName("BILATERAL - choose Sr");
    double funpar3=InputParameterWithName("BILATERAL - choose Sd");
    Bilateral(funpar1,funpar2,funpar3,sigsize,isig,osig);
    return 1;}

double Function2(){
    return Measure_RMSvalue(isig);
}
```

Fig.10 – Addition of the C source code calling the function for RMS evaluation.

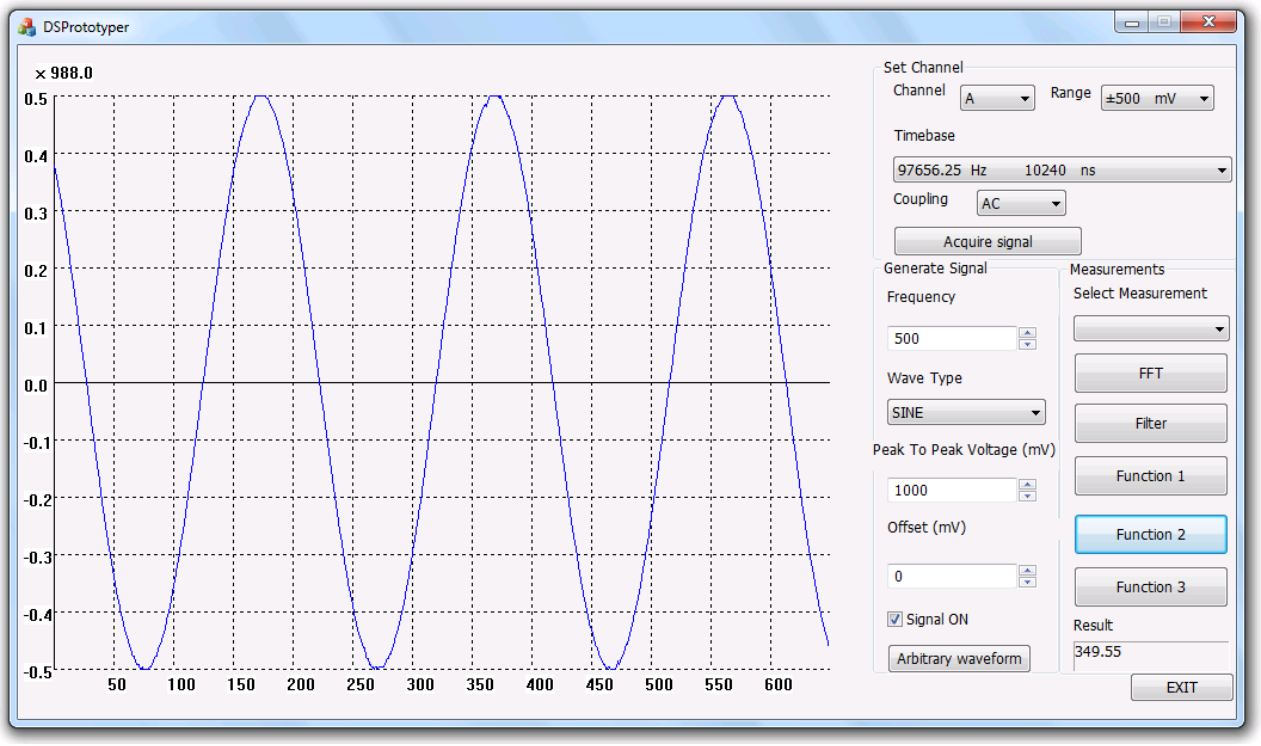


Fig.11 – Measurement of the RMS value of a sine wave generated and acquired by the multi-function oscilloscope.

Indeed, user interface and basic functions for signal acquisition, processing and display are already built-in in the proposed software shell. From the point of view of computer programming, only a basic knowledge of the widespread used C language is requested. Hence, the user can focus on the specific DSP algorithm, test the result using real (i.e., not simulated) electrical signals and possibly improve the method without being involved in problems of graphical user interface. The reported application examples have shown that the implementation of new DSP methods is quick and easy as possible. One important target of this work is education. Students of Industrial Engineering need to learn (and mainly to understand) how modern instrumentation works although they have a limited knowledge of computer programming and cannot spend time with complicated development environments. The proposed software shell could represent a useful and cost-effective tool to achieve this goal. (We plan to adopt it for first and second-level courses on Electrical Measurements at our University).

REFERENCES

[1] F. Russo, "New Tools for Classification and Evaluation of Filtering Errors in Color Image Denoising", *International Journal of Circuits, Systems and Signal Processing*, Vol.10, 2016, pp.178-189, ISSN 1998-4464.

[2] K. Kal Vin Toh, N. Ashidi Mat Isa, "Cluster-Based Adaptive Fuzzy Switching Median Filter for Universal Impulse Noise Reduction", *IEEE Transactions on Consumer Electronics*, Vol.56, n. 4 , pp.2560-2568, 2010.

[3] F. Russo, "Validation of Denoising Algorithms for Medical Imaging", in "Advances in Biomedical Sensing, Measurements, Instrumentation and Systems", S.C. Mukhopadhyay and A. Lay-Ekuakille (Eds.), Springer-Verlag, 2010, pp. 93-105.

[4] D. Potin, E. Duflos, P. Vanheeghe, "Landmines Ground-Penetrating Radar Signal Enhancement by Digital Filtering", *IEEE Transactions on Geoscience and Remote Sensing*, Vol.44, n.9, pp.2393-2406, 2006.

[5] D. Letexier; S. Bourennane, "Noise Removal From Hyperspectral Images by Multidimensional Filtering", *IEEE Transactions on Geoscience and Remote Sensing*, Vol.46, n.7, pp.2061–2069, 2008.

[6] C. Haritha, M. Ganesan, E. P. Sumesh, "A Survey on Modern Trends in ECG Noise Removal Techniques", Proc. IEEE 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Kumaracoli, India, March 18-19 2016, pp.1-7.

[7] D.N. Vizireanu, "Quantised Sine Signals Estimation Algorithm for Portable Digital Signal Processor Based Instrumentation", *International Journal of Electronics*, vol.96, n.11, pp. 1175-1181, 2009.

[8] A. Lakshmikanth and M. M. Morcos, "A Power Quality Monitoring System: a Case Study in DSP-based Solutions for Power Electronics", *IEEE Transactions on Instrumentation and Measurement*, Vol.50, n.3, pp.724-731, 2001.

[9] M. B. Yeary, R. J. Fink, D. Beck, D. W. Guidry, M. Burns, "A DSP-based mixed-signal waveform generator", *IEEE Transactions on Instrumentation and Measurement*, Vol.53, n.3 , pp.665-671, 2004.

[10] J. Mindykowski and T. Tarasiuk, "Development of DSP-based Instrumentation for Power Quality Monitoring on Ships", *Measurement* (Journal of the International Measurement Confederation), Elsevier, Vol.43, n.8, pp.1012-1020, 2010.

[11] O. Märtens et al., "DSP-Based Power-Quality Monitoring Device", Proc. 2007 IEEE International Symposium on Intelligent Signal Processing, WISP, 2007.

[12] M. Artioli, G. Pasini, L. Peretto, R. Sasdelli, F. Filippetti, "Low-cost DSP-based Equipment for the Real-time Detection of Transients in Power Systems, *IEEE Transactions on Instrumentation and Measurement*, Vol.53, n.4, 2004, pp.933-939.

[13] M. Brando et al. "A PC-Based Instrument for Automatic Monitoring and Control of a CPVT Power Plant", Proc. 20th IMEKO TC4

International Symposium, Benevento, Italy, September 15-17, 2014, pp. 40-44.
[14] F. Corrêa Alegria, E. Molino-Minero-Re, S. Shariat-Panahi, "Evaluation of a Four-point Sine Wave Frequency Estimator for Portable DSP based Instrumentation", *Measurement* (Journal of the International Measurement Confederation). Elsevier, Vol.45, n.7, 2012, pp.1866-1871.
[15] F. Adamo, G. Cavone, A. Di Nisio, A.M.L. Lanzolla, M. Spadavecchia, "A Proposal for an Open Source Energy Meter", Proc. IEEE Instrumentation and Measurement Technology Conference, I2MTC, Minneapolis, MN, USA, May 6-9, 2013, pp. 488-492.
[16] F. Adamo, F. Attivissimo, A. Di Nisio, M. Savino, M. Spadavecchia, "A Spectral Estimation Method for Nonstationary Signals Analysis with Application to Power Systems", *Measurement* (Journal of the International Measurement Confederation) Elsevier, Vol.73, n.11, 2015, pp.247-261.
[17] D. Bellan, "A Discrete Model of Jitter for Coarsely Quantized Waveforms", *International Journal of Circuits, Systems and Signal Processing*, Vol.10, 2016, pp. 269-274, ISSN 1998-4464.
[18] D. Bellan, "Estimation of Harmonic Power Components Affected by Frequency Instability", *International Journal of Circuits, Systems and Signal Processing*, Vol.10, 2016, pp. 275-280, ISSN 1998-4464.
[19] F. Adamo, F. Attivissimo, G. Cavone, C. Guarnieri Calò Carducci, A. M. L. Lanzolla, "New Technologies and Perspectives for Laboratory Practices in Measurement Science", Proc. 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), May 11-14, 2015, Pisa, Italy, 2015.
[20] https://www.picotech.com/oscilloscope/2000/picoscope-2000-overview
[21] PicoScope 2000 Series Programmer's Guide: https://www.picotech.com/oscilloscope/2000/picoscope-2000-manuals
[22] S. Rapuano and F. J. Harris, "An Introduction to FFT and Time Domain Windows", *IEEE Instrumentation & Measurement Magazine*, December 2007, pp. 32-44.
[23] I. Pitas and A. N. Venetsanopoulos, "Nonlinear Digital Filters: Principles and Applications", Springer-Verlag, 1990.
[24] B. K. Gunturk, "Bilateral Filter: Theory and Application" in "Computational Photography: Methods and Applications", Lukac R. (Ed.),CRC Press, 2011, pp.339-366.
[25] F. Russo, "Study of the Accuracy of the Color Peak Signal-to-Blur Ratio (CPSBR)", *International Journal of Circuits, Systems and Signal Processing*, Vol.10, 2016, pp. 242-253, ISSN 1998-4464.

F. Russo is currently Associate Professor of electrical and electronic measurements in the Department of Engineering and Architecture of the University of Trieste, Italy. He is author/coauthor of more than 100 papers in international journals, textbooks, and conference proceedings including the Wiley Encyclopedia of Electrical and Electronics Engineering, (J. G. Webster ed.). His research interests presently include intelligent measurement systems, virtual instruments, nonlinear and fuzzy techniques for image denoising and image quality measurement. He was one of the organizers of the 2004 and 2005 IEEE International Workshops on Imaging Systems and Techniques. He served as Technical Program Co-Chairman of the 2006 IEEE Instrumentation and Measurement Technology Conference and as Co-Guest Editor for the Special Issue of the IEEE Transactions on instrumentation published in August 2007. He was invited speaker of the plenary session "Fuzzy Models for Low-level Computer Vision: a Comprehensive Approach" at the IEEE Symposium on Intelligent Signal Processing (WISP 2007). He has served as session chairman/co-chairman in many conferences organized by IEEE, IMEKO, WSEAS and NAUN.

F. Travain received the BS degree in Industrial Engineering from the University of Trieste, Trieste, Italy. He is currently pursuing the MS degree in Electrical Energy and Systems Engineering at the Department of Engineering and Architecture, University of Trieste. His research interests include measurement systems, virtual instruments, digital signal processing and renewable energy sources.