UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"

POLO DELLE SCIENZE E DELLE TECNOLOGIE

FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA ELETTRICA

A Remotely Configurable and Programmable Measurement Laboratory

Annalisa Liccardo

TESI DI DOTTORATO DI RICERCA IN INGEGNERIA ELETTRICA

XIX CICLO

(coordinatore: prof. Guido Carpinelli)

TUTOR

PROF. NELLO POLESE

(UNIV."FEDERICO II", NAPOLI - DIEL)

CO-TUTOR

PROF. CLAUDIO DE CAPUA

(UNIV. "MEDITERRANEA",

REGGIO CALABRIA - DIMET)

DIPARTIMENTO DI INGEGNERIA ELETTRICA - VIA CLAUDIO 21 – 80125 NAPOLI

<u>*s*</u>

To my mother,
my best friend for ever

CONTENTS

INTRODUCTION

Remote laboratories are solutions which allow users from any location, through Internet network, to perform laboratory experiments as they were in an actual laboratory. Many researchers have recently focused their attention on remote laboratories applications in educational area. Students number at University, in fact, is hugely increasing, such that it is very difficult the arrangement of suitable laboratory bench. Remote laboratories could thus be a useful support to provide students for experimental activities, for a better understanding of topics learned during class lectures, with the aim of reducing problems such as low availability of expert teachers, technical personnel or suitable classrooms. In engineering disciplines, laboratory experiences are necessary to apply the studied theory and observe the differences between studied model and real world. In particular, for an Electrical and Electronic Measurements course, laboratory experiences are needed to fix the learned measurement methodologies and have a hands-on contact with instrumentation examined during class lectures. The experiments usually consists of control of laboratory instrumentation and conduction of live measurements. It is clear, hence, that remote laboratories for this kind of disciplines require additional features such as students' opportunity of interacting with laboratory equipment

A typical remote laboratory architecture consists of a server connected to the measurement instrumentation according to a defined communication protocol, which accepts the incoming client requests and forward them to the measurement instruments; on the other hand, the response of the measurement instruments are returned to the client according to HTTP protocol. As regard the software implementation, i.e. the software environment through which the application running on the server is developed, different solution are at the present available. The research activity has been initially focused on the exploration of a software solution allowing: *(i)* students access to the laboratory without downloading further software; *(ii)* easy integration of GPIB routines for the communication with instruments; *(iii)* interoperability with other software, in order to obtain flexibility in developing the client application.

A solution based on .NET Web Services has proved to be the most suitable for these purposes. Web Services, in fact, allow publishing defined function on Internet; these functions communicate through a protocol (SOAP) based on XML language, which is a language well understood by every software supporting communication on Internet. Client application, thus, can be developed in various software environments able to manipulate strings for constructing the SOAP request and transmitting it through TCP/IP. The main part of the research work has been devoted, then, to the development of the Web Service for the communication with measurement instruments. When different stations have been set up, another service able to manage multiple client connections has been implemented. Finally the client application has been developed in LabVIEW^{TM} environment, even though any other software could theoretically be adopted.

Another problem faced during these years has dealt with the possibility offered to the students of remotely operating on the analog circuit connected to the measurement station. Typically, in fact, the test circuit electrically wired to the instrumentation, is made of analog components not capable of exchanging communication data. The measurement station, then, has to be wired by technicians at the university where the physical laboratory is located, according to the experiment that students have to remotely conduct. It is clear, so, that each desired experiment requires a proper measurement station; that means to employ a server, a signal generator, multimeters or oscilloscopes, test circuit, electrical cables, GPIB and Ethernet board and cables. Since, in order to reduce the queue of requests to the laboratory, a number of identical stations could be desirable, laboratory resources and space are inexorably reduced.

In order to overcome these limitations, the research activity has been focused on the realization of a test circuit able to be programmed. At this aim, innovative integrated devices recently present on the market, namely Field Programmable Analog Arrays (FPAA), have been employed. FPAA are configurable through RS232 communication and are able to realize the transfer function of different analog circuits. It is, so, possible to implement a "universal" measurement station, where students select the analog circuit on which perform the experiment and control the instruments to execute the measurement.

So the remote laboratory has been completed by adding the functions for the remote FPAA configuration in both the server Web Service and the client program.

Finally, the realized laboratory has been tested. The remote measurements have been used also for characterizing the analog circuit emulated by the FPAA. In particular the differences between the expected output and that obtained for three different circuits (amplifier, filter and integrator) have been evaluated.

The research activity of the last months has concerned with the inclusion in the remote laboratory of smart sensor remotely configurable. FPAAs offer the opportunity of dynamically changing the characteristics of the realized analog circuit, configured in the primary configuration, that is a very attractive features in conditioning measurement signals. In particular the FPAA has been employed to realize the conditioning section of a distance sensor based on the estimation of Time Of Flight (TOF) of ultrasonic signals. The effectiveness of distance evaluation based on ultrasonic Time of Flight (TOF) measurements is mainly characterized by the accuracy gained to detect the exact arrival time of the returned echo. Usually, the time of echo onset is recognized when the received signal crosses a fixed threshold. As expected, the main factor degrading the measurement accuracy in estimating the onset is the noise level superimposed to the echo signal. The echo profile is indeed distorted by a number of factors such as geometrical and mechanical properties of the reflecting surface, position and orientation of the target, transmission paths, environmental inference factors (temperature, pressure, humidity), and so on.

Thanks to FPAAs flexibility, a conditioning block able to compensate the echo attenuation and shape distortion has been realized. Remote users are able to dynamically modify the circuit parameters, in order to tailor the conditioning block to the characteristics of the signal received by the transducer.

After the performances of the measurement technique have been assessed by laboratory users, a prototype of sensor, based on a DSC has been realized. The DSC starts the measurement process properly stimulating the piezoelectric transducer; when the FPAA output, i.e. the echo envelope, is received, the DSC controls the echo amplitude, in order to properly reconfigure the FPAA until the optimal echo shape is obtained. The performance of the realized prototype have, then, been assessed. Performance comparable with commercial sensors, but characterized by lower costs, have been experienced.

Chapter I - STATE OF THE ART ABOUT REMOTE TEACHING LABORATORIES

I.1 Introduction

Thanks to diffusion of Internet, high speed of networking and growing number of people enabled to be connected to the network, development of remote teaching laboratory has been becoming particularly attractive. Remote laboratories are solutions which allow users from any location, through Internet network, to perform laboratory experiments as they were in an actual laboratory. Many researchers have recently focused their attention on remote laboratories applications in educational area. Students number at University, in fact, is hugely increasing, such that it is very difficult the arrangement of suitable laboratory bench. Remote laboratories could thus be a useful support to provide students for experimental activities, for a better understanding of topics learned during class lectures, with the aim of reducing problems such as low availability of expert teachers, technical personnel or suitable classrooms [1]. In engineering disciplines, laboratory experiences are necessary to apply the studied theory and observe the differences between studied model and real world. In particular, for an Electrical and Electronic Measurements course, laboratory experiences are needed to fix the learned measurement methodologies and have a hands-on contact with instrumentation examined during class lectures. The experiments usually consists of control of laboratory instrumentation and conduction of live measurements. It is clear, hence, that remote laboratories for this kind of disciplines require additional features such as students' opportunity of interacting with laboratory equipment [2]. Sometimes, the course experimental session is provided by means of tests conducted on simulated experiments. In authors' opinion, this teaching technique is inadequate, since: *(i)* students could hardly understand concept as measurement uncertainty, randomness and noise; *(ii)* laboratory experiences aim just at highlighting the differences between theoretical model and real world; and *(iii)* simulations could cause lack of critical thinking in setting the instrumentation because simulated experiments do not suffer from problems related to incorrect configuration of instruments. A comparison between different teaching methods, with regards to teaching effectiveness and time and cost per students, is schematized in Fig. I. 1.

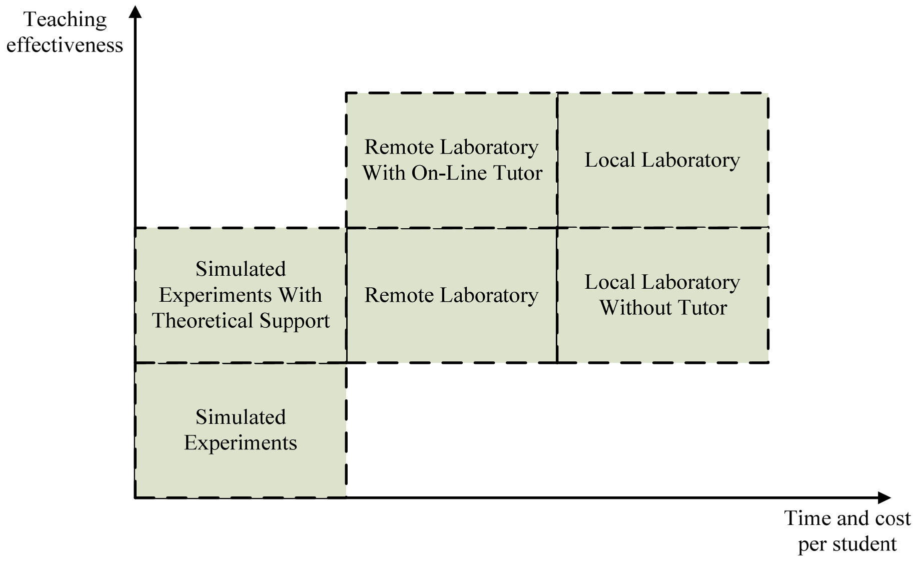

Teaching effectiveness
Remote Laboratory With On-Line Tutor
Local Laboratory
Simulated Experiments With Theoretical Support
Remote Laboratory
Local Laboratory Without Tutor
Simulated Experiments
Time and cost per student

Fig. I. 1 - Comparison between different teaching strategies.

It is undoubted that remote laboratories are not able to replace traditional face-to-face laboratory lessons, but, as enlisted in Tab. I. 1, they present advantages, especially in courses

Tab. I. 1 - Advantages of remote and physical laboratories.

Remote Laboratory	Physical Laboratory
Access at any time from any place	Students wire the test circuits
Experiments can be repeated	Students handle instruments
Students can dispose of measurement stations for more time	Students have teacher at their disposal for any question
No need of tutor presence	

characterized by intense laboratory activities, as useful as students' number increase [3].

I.2 Software solutions for remote control of measurement instrumentation

While implementing a remote laboratory, two critical actions have to be completed: *(i)* making clients capable of connecting to measurement equipment through Internet; and *(ii)* adapting previously realized experiments to be remotely controlled through Internet.

The former task is always fulfilled by inserting a server between measurement instruments and Internet network; the server should be capable of communicating with instrumentation through a standard protocol in order to act as a translator between HTTP and the instrument communication protocol.

Therefore, the common configuration of a remote laboratory is depicted in Fig. I. 2. The typical operating steps are:

1. Client accesses a dedicated Server through a defined connection (HTTP, Data Socket).
2. The server decodes client's request and sends the proper commands to the measurement instrumentation by means both of a standard communication protocol (IEEE 488, RS 232, USB) and a software running on it. The software has to be capable of managing the communication according to the standard.
3. The server receives the measurement results.
4. The server finally forwards results to the client.

The latter task can be fulfilled by implementing different solutions, each of which characterized by its advantages and drawbacks [4].

In the following sections an overview of software solutions available in literature will be discussed, highlighting advantages and peculiarities of each technology. A scheme of the

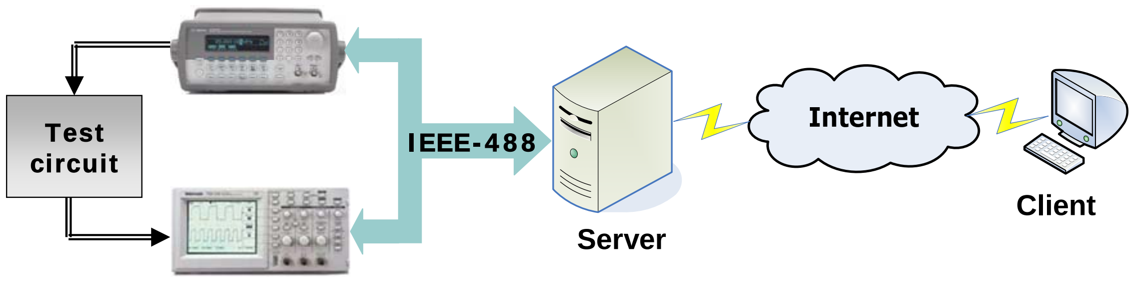


Fig. I. 2 - Scheme of a remote laboratory arrangement.

suitable solution according to developer's needs is shown in Fig. I. 3.

I.2.1 Solutions exploiting Client-Server software

In this section, particular attention is paid on National Instrument LabVIEW^{TM} software, a de facto standard in developing measurement applications. A similar analysis can be conducted on different software environments that, in their latest releases, allow communication with instrumentation and provide server tools (as an example, Matlab by Mathworks or LabWindows/CVI by National Instrument).

The techniques presented in this section are usually indicated as VI-based solutions, since both the server and client applications are developed in LabVIEW^{TM} environment. VI-based techniques allow an easier implementation and more flexibility, but impose a specific version of LabVIEW^{TM} Run Time engine (LRT) to be installed on client computer [3]. The available techniques make use of some tools comprised in LabVIEW^{TM} environment that are Data Socket, VI Server, and TCP VIs.

DataSocket is a technology greatly facilitating live data exchange between LabVIEW^{TM}

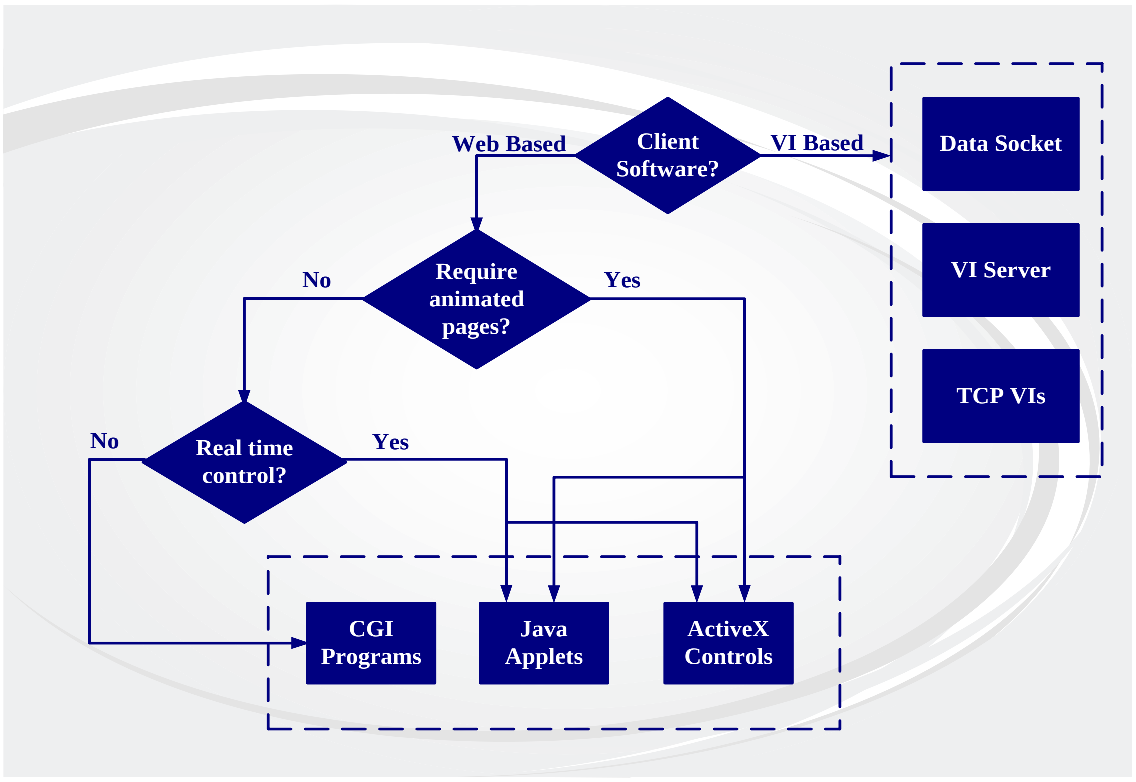


Fig. I. 3 - Brief representation of typical solutions for remote control of measurement instrumentation.

based applications over the Internet/Web without any TCP/IP programming [6]. This technology includes two components: DataSocket server application and DataSocket client application. DataSocket server publishes data and manages client connections. DataSocket client application is used to read, write, and manipulate published data.

VI server is a LabVIEW^{TM} add-on that allows the easy building of client-server applications over TCP/IP for networked computers. The client-server applications on networked computers are operative when VI server is enabled. A dialog box of VI server configuration allows the developer to enable the server and set its properties such as port number and TCP/IP access privileges needed for security.

TCP Vis are programs that exploit subroutine supporting TCP/IP communication protocol over the network. Developer need knowledge about TCP/IP protocol and standard. Two programs running respectively on server and client have, in fact, to be implemented. Client and server can reliably exchange data and results only if a proper synchronization has been realized between their programs [7].

The main advantage of all VI-based solutions is the reusability of pre-existent programs. If TCP VIs or DataSocket are used, developer just needs to replace the traditional routines for instruments communication with those peculiar to TCP/IP or DataSocket transmission, respectively. Whereas VI server is adopted, it is enough to properly configure the server, by adding the desired VI in the published VI class. Another benefit is the security of client-server sessions; generally only authorized users have, in fact, client applications and host name or IP address of server machine. Moreover, these technologies allow exploiting communication ports different from port 80, employed by HTTP communication (communication port in client and server program has obviously to be the same). The use of VI Server allows to still improve communication security; it is in fact possible to compile, in the cited server dialog box, a list of authorized and denied users. It is worth noticing that user's accessibility could be simplified by hiding the server IP address and port in client program. This way clients can access the laboratory without knowing this kind of information.

In conclusion, all described solutions allow exchange of live data between client and server; this feature makes them particularly attractive for real-time remote monitoring [8]. The main drawback lays on the maintenance cost. As stated above, each change in server program requires related modifications of client program that has, hence, to be distributed again to the users. The accessibility is lowered too, since users have to install LRT software on their computer. Some problems are also related to management of concurrent server accesses; no facilities are provided by the environment and developer has to design server program in such a way that it can handle queues.

I.2.2 Solutions developed through Web-Oriented software

Web-based solutions, with respect to the previous approach, require that developer has knowledge about Internet programming. On the contrary, client side is facilitated because users only need a standard browser to access the instrumentation control.

- **Java applets**

Java is a pure object-oriented programming language for distributed applications running on ‘thin clients’, namely on low-cost client computers. Java applications, or *Applets*, are different from ordinary applications because they reside on Internet in centralized servers, and may run on client machines. Web developers can realize applications in such a way that Internet can deliver the applications or applets to users’ computers when they request them. In general, there are two ways to implement systems using Java technology on Internet: *(i)* computational processes running on client machine, and *(ii)* computational processes residing at the server side. In the first case, users download Java applets, embedded in HTML documents, that are locally executed [9]. In the second case, users download Java applets acting as Graphic User (GU) interfaces for exchanging data input, pre and post-processing modules. It is worth noting that in both cases the main computational part, which can be a Java application or an application written in any object-oriented language, resides at server side. The running applet is able to link back to either the same server from which it has been downloaded or any other Web server existing anywhere on Internet. In remote laboratories, Java applets are used to create HTML pages through which users can interface with LabVIEW^{TM} [10] or other instrumentation control software, without needing to download this software. Advantages of using Java client-based applications are: *(i)* HTML pages at user’s side are “dynamic”; *(ii)* applets run on client and do not overload the server; and *(iii)* application written in Java is, in principle, platform independent.

The most important drawback associated with the use of Java consists in the need of redesigning and completely rewriting the existing software in the Java language. The usability is jeopardized too, since users have to download and install Java Virtual Machine (JVM) at the first connection. The process takes up to few minutes through a 56kbps connection.

In implementing remote laboratory with Java, Nacimiento software, Appletview [11] is particularly adopted. This software allows developing Java Applets whose plug in is a LabVIEW^{TM} VI. Besides keeping the advantages previously listed of Java Applets, Appletview provides the opportunity of simply reusing the old programs developed in LabVIEW^{TM} environment.

- **ActiveX controls**

ActiveX is a set of core technologies that provides cross-platform, component interoperability across Internet. ActiveX is, in fact, a brand name for Microsoft Component Object Model (COM). ActiveX includes COM to enable communication between client components (models and modelling systems), and Distributed COM (DCOM) to integrate components across the network [12].

The ActiveX Controls are interactive objects in a Web page that provide interactive and user-controllable functions. ActiveX is supported by many different languages and tools (open technology), and can be developed in various environment as Delphi, Microsoft's Visual Basic, and Visual C. In some remote laboratory applications, a proper collection of ActiveX is adopted to provide user with something similar to the front panel of the controlled instrument. This way, when the user operates on a control, he changes the value of the corresponding control in the software of the server connected to instrumentation. In this section it is worth citing recent applications exploiting ActiveX joint to multi-agent technologies [13].

Advantages of using ActiveX are: *(i)* existing applications written in C++, Visual Basic, or LabVIEW^{TM} languages can be easily transformed into ActiveX components by changing the controls class; *(ii)* server is not overloaded by running multiple applications; *(iii)* ActiveX allow live data exchange; and *(iv)* ActiveX components are currently faster than Java applets. Drawbacks associated with ActiveX technology are: *(i)* size of ActiveX components could be considerable (up to few Megabytes) with a consequent, long download time; *(ii)* ActiveX are less secure for clients since they receive access to all computer resources; and *(iii)* ActiveX components run currently only under Microsoft Windows.

- **CGI programs**

Common Gateway Interface (CGI) programs work on all browsers and are easy to be realized and interfaced with HTML forms.

These programs gather the inputs sent by users through an HTML page (as shown in Fig. 3), execute some code and provide the outputs to user displaying them in an HTML answer page. CGI scripts can be written in almost every language native to the Web server, including Perl, C, C++, Basic, Pascal, Windows CGI, or LabVIEW™. Solutions often realized for remote laboratories require that a CGI program receives the user's inputs and executes a LabVIEW™ program in order to carry out the measurement process [14].

Advantages of CGI approach are: *(i)* the main body of software runs on server and only input and output forms are sent across the Web; *(ii)* user does not have to download any kind of software; and *(iii)* there is no need to make controlling software platform-independent, since the software itself runs only on server.

Drawbacks associated with use of CGI programs are: *(i)* there are not features for connections management; developer himself has to implement programs for handling multiple connections; *(ii)* existing software must be redesigned; *(iii)* server can easily be overloaded by multiple users accessing its resources; and *(iv)* attention should be paid on security issues on server side, since CGI scripts may have access to the vital resources of the server.

- **LabVIEW™ Web Server**

It is a tool provided with LabVIEW™ environment; it allows, by means of Web Publishing Tool, to build HTML pages with embedded dynamic images (plug-in) of VIs. The Web Server has obviously to be enabled and properly configured. The required information consists of the list of allowed users, the list of denied users and the visible VIs. When client accesses the HTML page, he sees the front panel of the VI during its execution and can require the VI control in order to change program inputs. If another client is controlling the VI at the same time, user is queued; if this is the case, the client operating the VI maintains its control only for a fixed time, set by Web Server developer. Advantages of using Web Server are: *(i)* published pages are dynamic; *(ii)* Web Server includes itself a queue procedure; *(iii)* Web Server is the most straightforward way to publish previously developed LabVIEW™ applications [15].

Drawbacks associated with Web Server are: *(i)* users have to download LabVIEW^{TM} Run Time (LRT) Engine in order to see the plug in; since LRT size is about 20 Megabytes, it leads to waste of time; *(ii)* security issues are not provided; *(iii)* LabVIEW^{TM} Web Server is not compatible with other server software.

A summary of the characteristics of the examined techniques is shown in Tab. I. 2.

Tab. I. 2 - Comparison between different technologies.

	Security	**Software required on client side**	**Development**	**Reuse of existent programs**	**Interoperability**
TCP VIs	Excellent	LabVIEW	Medium	Easy	Not supported
VI Server	Good	LabVIEW	Easy	Easy	Not supported
DataSocket	Excellent	LabVIEW	Easy	Easy	Not supported
Java	Poor	JVM	Difficult	Easy	Good
Active X	Poor	Active X	Difficult	Easy	Not supported
CGI	Sufficient	Nothing	Medium	Difficult	Not supported
LV Web Server	Good	LV RTE	Easy	Very easy	Not supported

I.3 Lack of manual handling of the test circuits

Architectural scheme illustrated in Fig. I. 1, shows that, at the present, remote users are not capable of operating on the test circuit. Users, in fact, can communicate only with measurement instruments which are equipped with a proper communication board, whereas they cannot modify electrical wiring of the circuit. This is a not negligible constraint of remote laboratory because of the following reasons: *(i)* laboratory resources, in terms of space and instrumentation, are hugely reduced, since a suitable measurement station is required for each experiment available for remote connections; *(ii)* students are able to perform only the experiment according to which the measurement circuit, in the physical laboratory has been wired. For each new experiment, a technician changing the electrical connection is required; *(iii)* without knowledge about the circuit under test, students could consider the laboratory experiment as a simulated experiment, becoming bored and less interested [16].

Recently, solutions making use of a switching matrix board, shown in Fig. I. 4, have been proposed. Users can select a particular test circuit among three or four available configurations [17]. In this case, however, two problems have to be taken into account: *(i)* when the switches connections are changed, the end of electrical transient has to be waited before operating on the circuit; *(ii)* due to the low power involved, switches resistance cannot be neglected if compared with circuit impedance.

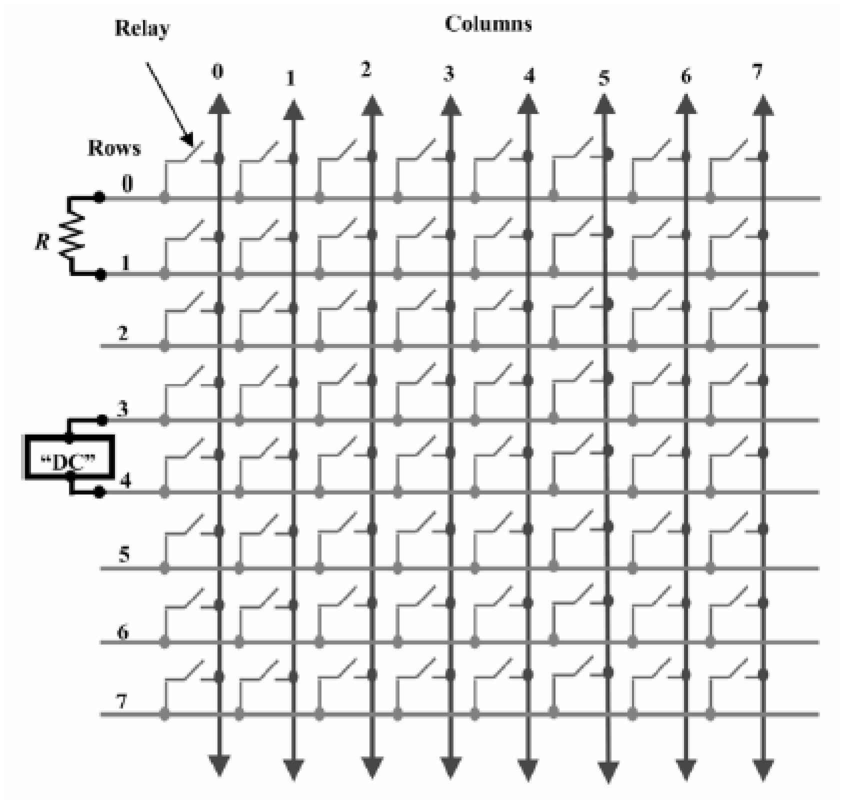


Fig. I. 4 – Switching matrix board for remote selection of test circuit.

References

[1] M. J. Korczynski, A. Hetman, A. Hlobaz, "Virtual Laboratory a Key for Teaching Principles of Digital Signal Processing", *Proc. Of IEEE Techn. Conf. on Instr. And Meas.*, Vol.2, pp. 1222-1226, Ottawa, Canada, 2005.

[2] P. Arpaia, A. Baccigalupi, F. Cennamo, P. Daponte, "A measurement laboratory on geographic network for remote test experiments", *IEEE Trans. on Instrum. and Meas.*, Vol.49, No.5, pp. 992-997, 2000.

[3] C. Landi, A. Liccardo, N.Polese, "Remote Laboratory Activities to Support Experimental Session for Undergraduate Course of Measurements for Diagnostics and Quality", *Proc. of IEEE Instrum. and Meas. Tech. Conf.*, pp. 851-856, Sorrento, Italy, 24-27 Aprile 2006.

[4] A. Baccigalupi, C. De Capua, A. Liccardo, “Overview on the Development of Remote Didactical Laboratories”, *Proc. of IEEE Instrum. and Meas. Tech. Conf.*, pp. 217-222, Sorrento (NA), Italia, 24-27 Aprile 2006.

[5] P. Arpaia, F. Cennamo, P. Daponte, M. Savastano, “A distributed laboratory based on object-oriented systems”, Measurement, vol. 19, pp. 207-215, 1996.

[6] National Instruments, "Integrating the Internet into Your Measurement System: DataSocket Technical Overview," available at *www.ni.com/Internet.*

[7] A. Gupta, A. Gabr, V. C. Matzen. “Alternatives in the Implementation of Internet-Enabled Laboratory Experiments in Undergraduate Civil Engineering Courses”, *Proc. of American Society for Engineering Education Annual Conference,* Salt Lake City, UT, 2004.

[8] C. Steidley, R. Bachnak, “Software and Hardware for Web-based Education”, *Proceedings of the 2004 American Society for Engineering Education Annual Conference,* Salt Lake City, UT, 2004.

[9] G. Canfora, P. Daponte, S. Rapuano, “Remotely accessible laboratory for electronic measurement teaching”, *Computer Standards and Interfaces*, vol.26, No.6, 2004, pp.489-499.

[10] A. Ferrero, S. Salicone, C. Bonora, M. Parmigiani, “ReMLab, “a Java-based remote, didactic measurement laboratory”, *IEEE Trans. on Instrum. and Meas.*,Vol.52, pp. 710-715, 2003.

[11] www.nacimiento.com.

[12] A. A. Siddiqui, Y. I. Zia, M. Aamir, S. E. Haque, “Virtual Remote Electronic Laboratory”, *Proc. of IEEE Students Conference on Emerging Technologies*, Vol. 1, pp. 94–98, 2002.

[13] F. Ponci, A. Deshmukh, A. Monti, L. Cristaldi, R. Ottoboni, “Interface for Multi-Agent Platform Systems”, *Proc. Of IEEE Techn. Conf. on Instr. And Meas.*, Vol. 3, pp. 2226-2231, Ottawa, Canada, 2005.

[14] C. De Capua, A. Liccardo, “An E-Learning portal for Electrical and Electronic Measurement courses employing remote instrument control”, *IADAT Journal on Advanced Technology*, Vol. 1, n. 4, pp. 174-176, 2005.

[15] M. Naghedolfeizi, S. Arora, and S. Garcia, “Survey of LabVIEW Technologies for Building Web/Internet-Enabled Experimental Setups”, *Proc. of American Society for Engineering Education Annual Conference,* Montreal, Quebec, Canada, 2002.

[16] I. Gustavsson, **“**Remote laboratory experiments in electrical engineering education”, *Proc. of IEEE Intern. Conf. on Devices, Circuits and Systems,* pp. 1021-1025, Caracas, Venezuela, 2002.

[17] J. A. Asumadu, R. Tanner, J. Fitzmaurice, M. Kelly, H. Ogunleye, J. Belter, S. Chin Koh, “A Web-Based Electrical and Electronics Remote Wiring and Measurement Laboratory (*RwmLAB*) Instrument”, ”, *IEEE Trans. on Instrum. and Meas.*,Vol.54, pp. 38-44, 2005.

Chapter II - Remote Laboratory Based on .Net Web Services

II.1 Introduction

In this section the software solution realized during the research activity to solve problems affecting the available technologies is presented. The developed remote laboratory is based on Web Services realized in Visual Basic .Net environment.

Solutions presented in the previous section have a common characteristic: when a software environment has been selected on the server, developers have to adopt a specific software to realize all client interfaces[1]. This feature is not suitable for distributed remote laboratory, where different physical laboratory take part to a unique remote laboratory [2]. A very important characteristic of a remote laboratory is the scalability; hence, it would be desirable including other applications without forcing developers with difference knowledge to learn and use a new software environment.

Web Services are application making particular functions, called Web Methods, available on Internet network. They communicates according to SOAP (Simple Object Access Protocol), that is based on XML (eXtended Mark up Language) [3]. The peculiarity of XML relies on its interoperability with other software; in practice, every software supporting communication on Internet is able to understand XML and, so, communicates according to SOAP [4]. Then, when a Web Method has been developed, in every software environment it is possible to build an application invoking the Web Service and use its methods as they were functions declared in the program lines [5].

During the research activity, then, a Web Services library for the control of laboratory instrumentation has been build up. Then client application, invoking these functions, have been developed in different software environments, according to the required characteristics of the experiment.

Obviously, due to the large number of connections expected, the issue of managing concurrent connections has been considered.

II.2 XML Web Services

An XML Web Service is a programmable application providing particular functions which are accessible by a wide variety of systems making use of universal Internet standards as XML and HTTP [6].

A Web Service can be employed by an internal application or can be published on Internet, and invoked by an indefinite number of remote applications. Thanks to its standard interface a Web Service allows different systems exchanging message and cooperating, independently on how they have been developed or where they are located [7].

There are different Web Services definitions, but they all highlight these characteristics:

- Web Services offer functionalities by means of a web standard protocol; usually the protocol is SOAP (Simple Object Access Protocol).
- Web Services interface is accurately described so that users can implement applications able to communicate with them. The description is an XML document, called WSDL (Web Service Description Language) [8].
- Web Services are registered in public lists in order to be easily discovered; This is done through UDDI (Unversal Description Discovery and Integration).

The most important advantage of Web Services technology relies on the offered opportunity of a standard communication between programs developed in different software environment and on different software platform, by means of standard web protocol as XML, HTTP and TCP/IP. In the following the realized applications based on Web Services will be described; in particular, a measurement station can be seen as a Service itself, available for various developers [9]. If it is desirable to share a function among applications of different companies, different platforms based on different components (as an example J2EE, .NET, CCM), thanks to Web Service technology a standard language is adopted, where specific applications and data type are encapsulated within standard interfaces.

In fact, Web Services are defined in a standard language W3C, the WSDL, are made available on Internet by registering them on public repository (UDDI), and are invoked through a light middleware protocol as SOAP.

Fig. II. 1 shows the typical operating steps when a Web Service is invoked [10]:

- The Web Service is searched in UDDI repository;
- Its interface its gathered by the Web Service consumer through the WSDL file;
- The Web Service is invoked by means of a SOAP call.

So Web Service consumer and Web Service provider have to share just the Web Service description in WSDL format; the only features required for a Web Service invocation is an Internet access and a SOAP support, usually furnished by every software platform.

II.2.1 Universal Description Discovery and Integration

As for all the Internet applications, a tool for searching the resource on the network is necessary. XML Web Services can be inserted, together with their main information, in public registries so that potential users can easily find them. In fact various IT societies, as Microsoft or IBM, put at programmer disposal free UDDI registries where they can insert their Web Service [11].

UDDI specifications define the standard method to publish the Service information. Web Service's research can be conducted according to three criteria: by searching the name of the Web Service's owner (registered in the white pages), by searching the Web Service's category or geographical location (registered in the yellow pages), by searching technical details, called tModels, on the message's exchange (registered in the green pages).

II.2.2 Discovery

Web Service discovery is the operation through which documents related to a service description are obtained. By means of this process, potential users know that a particular service exists and how to find documents describing it. In particular, an XML file with DISCO extension allows to find the other service descriptor. The DISCO file, in fact, contains the links to all the other documents regarding the Web Service.

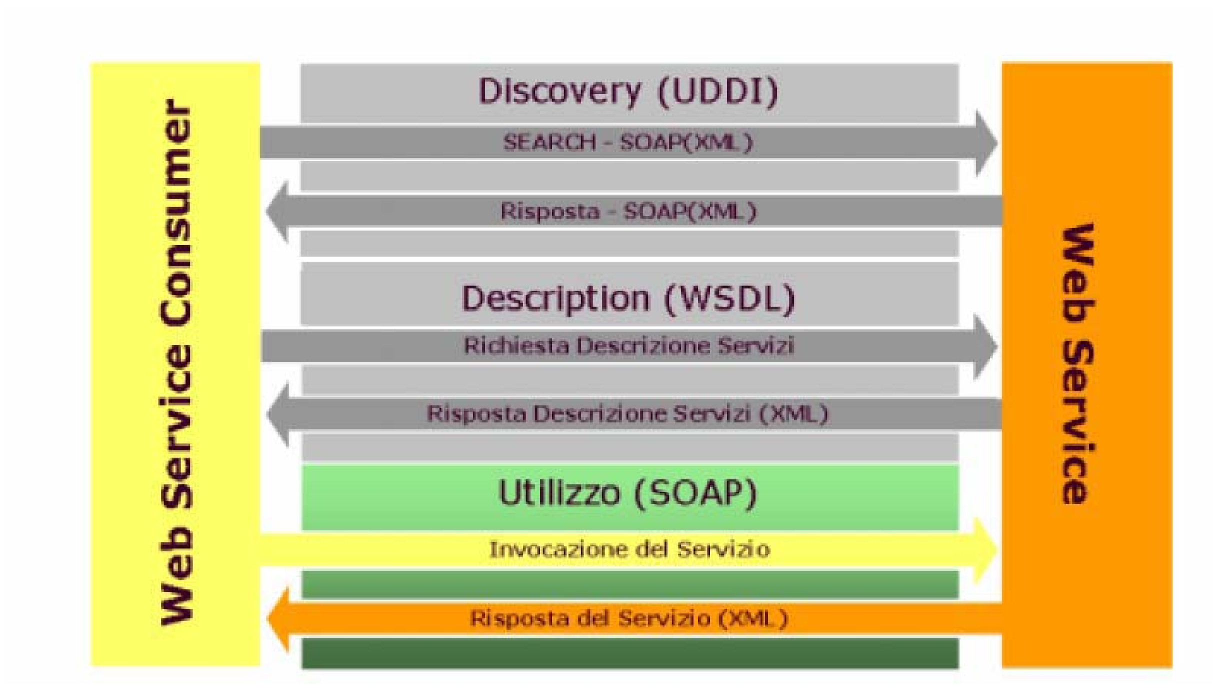


Fig. II. 1 – Typical operating steps for a Web Service invocation.

As an example, it is shown the structure of the DISCO file of the Web Service realize for the remote control of a measurement station. It is possible to detect the URL of the Service described, together with the position of its WSDL file.

```
<?xml version="1.0" encoding="utf-8" ?>
<discovery xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://schemas.xmlsoap.org/disco/">
    <contractRef
ref="http://143.225.116.110/WebServiceBANCO/banco.asmx?wsdl"
    docRef="http://143.225.116.110/banco.asmx"
    xmlns="http://schemas.xmlsoap.org/disco/scl/" />
    <soap address="http://143.225.116.110/WebServiceBANCO/banco.asmx"
    xmlns:q1="MISURE/BANCO"
    binding="q1:CounterSoap"
    xmlns="http://schemas.xmlsoap.org/disco/soap/" />
</discovery>
```

However, if a web site implements a Web Service, the description can be located also on another server, with, for example, a web site acting as Web Services list. Otherwise the description can be also not present, if the service has been developed for private applications.

II.2.3 Web Service Description Language

Message exchange with Web Services is based on the communication through XML messages as stated by the description of the published service. Service description is an XML document, typed in an XML grammar called WSDL (Web Services Description Language) which defines the message format that will be correctly recognized by the Web Service. Then the description is an agreement specifying the service behaviour and provide to users information about how it has to be invoked.

Web Service behaviour is defined through models which indicate to consumer what the service do if it receives a correct message. In particular, the request/answer model illustrates how to build a request message and describes the format of the service answer.

II.2.4 Simple Object Access Protocol

SOAP is an XML-based protocol, easy and light, to exchange complex information. This standard protocol assures easy of use and modularity because Web Service characteristics are described in a standard universal architecture, whereas information about the rules peculiar of the software platform through which the service has been realized are not necessary. SOAP specifications comprise four main parts:

- The envelope is the only necessary part. It represent the basic exchange unit between SOAP processors and encapsulate the whole SOAP request.
- The second part defines the rules to decode data in order to format them according to the developed application.
- The third section defines the RPC (Remote Procedure Call), i.e. a request/answer model.
- The last part defines the relationship between SOAP and HTTP (Hyper Text Transfer Protocol), even though SOAP can be used also with other protocols.

An example of SOAP request and response is reported in the following, where the explained above sections are highlighted.

```
        POST /WebServiceSCPI/servizioGPIB.asmx HTTP/1.1
        Host: 143.225.116.110
        Content-Type: text/xml; charset=utf-8
        Content-Length: 385
        SOAPAction:
    “http://localhost/WebServiceSCPI/servizioGPIB/Readstring”
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
xmlns:xsd=”http://www.w3.org/2001/XMLSchema”
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
       <soap:Body>
              <Readstring
       xmlns="http://localhost//WebServiceSCPI/servizioGPIB">
                     <BUS>0</BUS>
                     <ADDRESS>1</ADDRESS>
              </Readstring>
       </soap:Body>
</soap:Envelope>



HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 420

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
       <soap:Body>
              <readstringResponse
              xmlns="http://localhost/WebServiceSCPI/ServizioGPIB">
                     <readstringResult>eventuale
              risposta</readstringResult>
              </readstringResponse>
       </soap:Body>
</soap:Envelope>
```

II.2.5 Web Service advantages

The advantages related to the use of Web Services are:

- Platform independence: partners communicating through Web Services can choose different software platform without compatibility problem.
- Recycle of infrastructures: standard protocol as HTTP or SMPT can be used, so it is not necessary to purchase new components as proxy server, router, etc..
- Recycle of software: the extension of the previously realized applications into web-based applications is very easy.

- Firewall freedom; since it is based on HTTP, potential firewalls do not impede SOAP message exchange.

II.3 Realized Web Service

In this section the realized XML Web Service, on which the whole laboratory is based, is presented. The Web Service aims at publishing on the web the functions to communicate with the measurement instruments. Every application able to build SOAP message is capable, then, of remotely using the measurement station.

The adopted software environment has been Microsoft Visual Studio .NET® which offers at developer's disposal various functionalities and has proved to be very straightforward for Web Services programming. In particular, Visual Basic .NET® has been preferred. The software Measurement Studio® 7.0, by National Instruments, has been employed too, for implementing the routines for communication with measurement instruments. This way, functionalities to exchange messages with instrumentation, according to a specific protocol, can be imported and used as classes in Visual Studio environment.

The class *device* of Measurement Studio has been the most employed class; it provides, in fact, the methods *read* and *write*, for, respectively, receiving and sending messages to a measurement instrument identified by its own GPIB address.

In the following sections the implementation of the software solution in order to obtain the architecture shown in Fig. II. 2 will be presented. In particular the completed instruments library will be described; it is composed by various classes, each one related to a particular object-instrument. Then data exchange between the Web Service and the library in order to communicate with the different laboratory instruments will be described.

II.4 Namespace Instrument_Library

In this library the objects-instruments are defined; the library, then contains the data that will be managed by the Web Services. In particular, the service *Measurement Station* gets data from the library in order to send commands to the instrument according to SCPI format.

For a better readability, the library is divided in two namespace: *Instrument* and *Command*. In the namespace *Instrument* the definitions related to the particular instruments of the measurement station are listed. The classes of the namespace *Command* provide, once an instrument has been selected, the proper SCPI command to be sent.

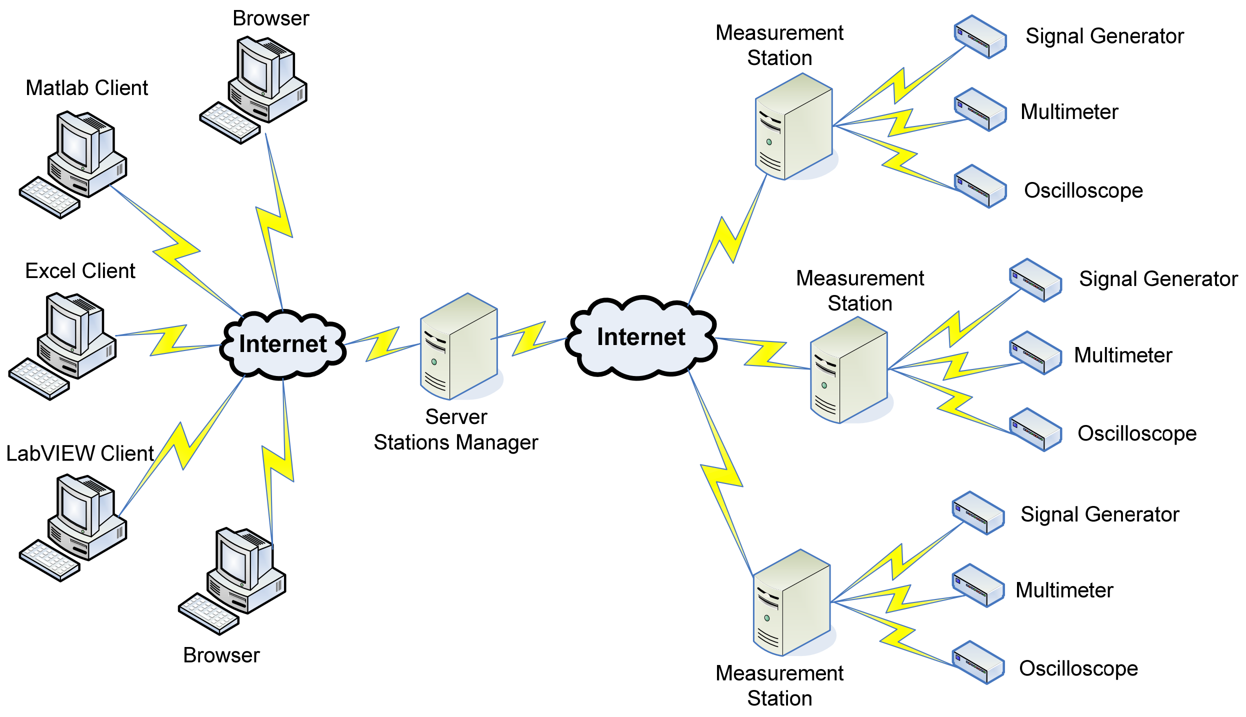


Fig. II. 2 – Architecture of the realized remote laboratory.

II.5 Namespace Instrument

In this namespace the objects employed for transferring the configuration parameters (from the user to the instrument) and receiving the measurement results (from the instrument to the user) are declared.

Three namespaces are here defined, according to the three instruments put at students disposal in the measurement station:

- Signal generator
- Multimeter
- Oscilloscope.

II.5.1 Signal generator

The most important class in this namespace is *GeneratorAgilent33120A*, defining the instrument Agilent 33120A. Besides typical parameters of a signal generator as the function to be generated or the output impedance, two particular objects are declared: *Delay* and

Buffer. *Delay* is a time delay used after the configuration of the generator; it is, in fact, necessary to wait that the output signal is in steady state condition. *Buffer* is an object used to record the samples string when user selects the generation of an arbitrary waveform. The objects declared are listed in Fig. II. 3.

II.5.2 Multimeter

The main class in this namespace is *MultimeterHP34401A*. This class contains enumerative variable configuring typical multimenter parameters, as measurement to perform, coupling and nplc; moreover two additional types are defined: *Delay*, which is a time delay inserted before executing a measurement; *Buffer*, where the measures read by the instrument are recorded; and *nummeas*, to which the desired measurement number is associated. The maximum number of measurement is 512.

II.5.3 Oscilloscope

The main class in the namespace is *OscilloscopeTDS210*. Here the subclasses *Channel*, *Trigger*, *Horizontal* and *Acquisition* are declared. As for the namespaces Signal_Generator and Multimeter the type enumerator has been largely employed. The enumerator type can be very useful, in fact, when user has to select a parameter value among a finite number of alternatives. For example, when users have to configure the horizontal scale of the oscilloscope, the sec/div variable can assume different particular values; the type enumerator is the most suitable in this instance, so that it is not necessary to verify that the settled parameter is correct.

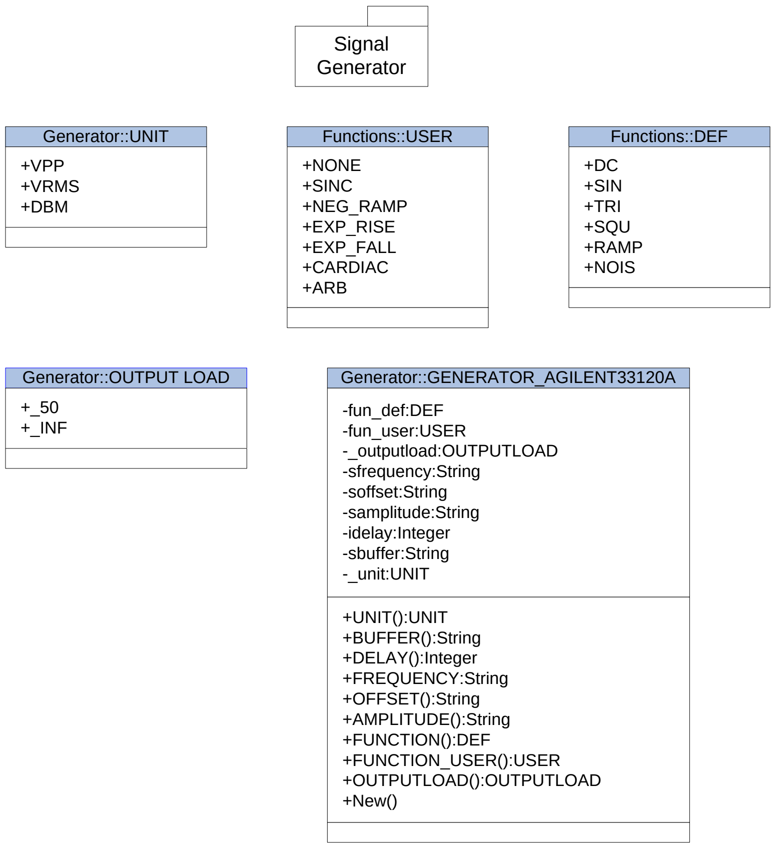


Fig. II. 3 – Namespace Signal Generator.

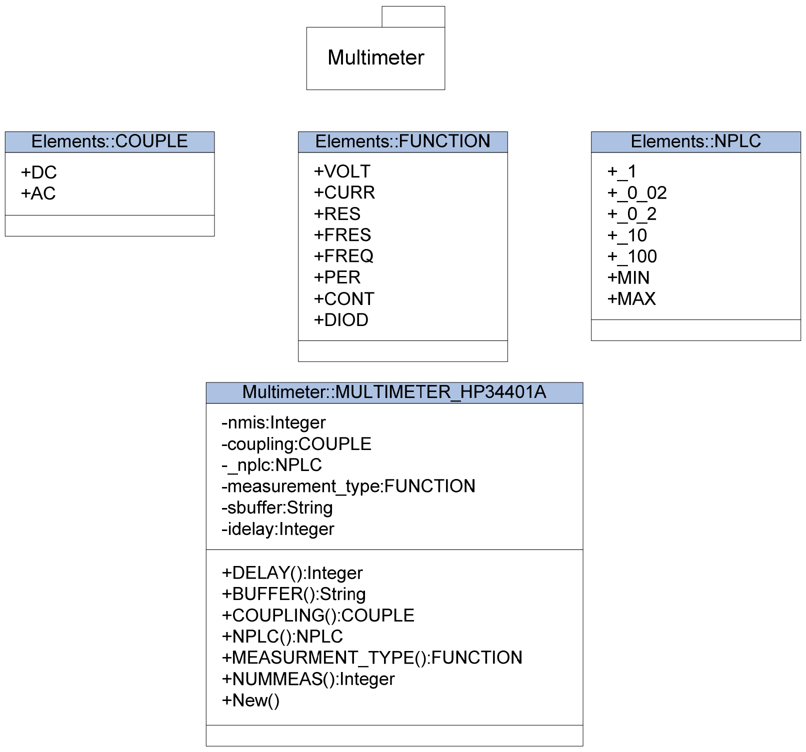


Fig. II. 4 – Namespace Multimeter.

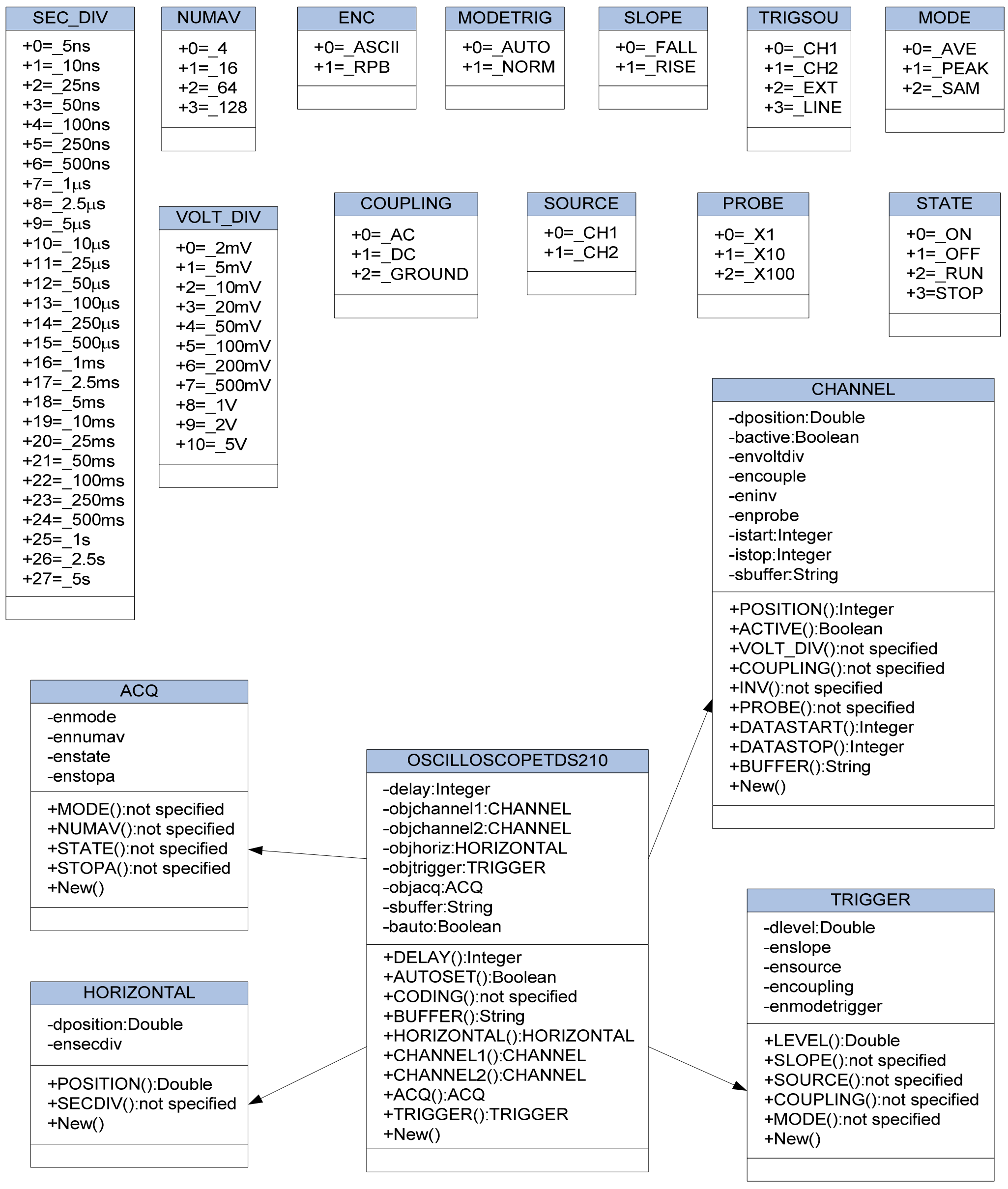


Fig. II. 5 – Namespace Oscilloscope.

II.6 Namespace Command

In this namespace the classes for creating the messages according to syntax SCPI are defined. Obviously, these classes are strictly dependent on the particular instrument and on the command strings reported in the instrument user's manual.

The main classes are:

- SCPI_Generator;
- SCPI_Multimeter;
- SCPI_Oscilloscope.

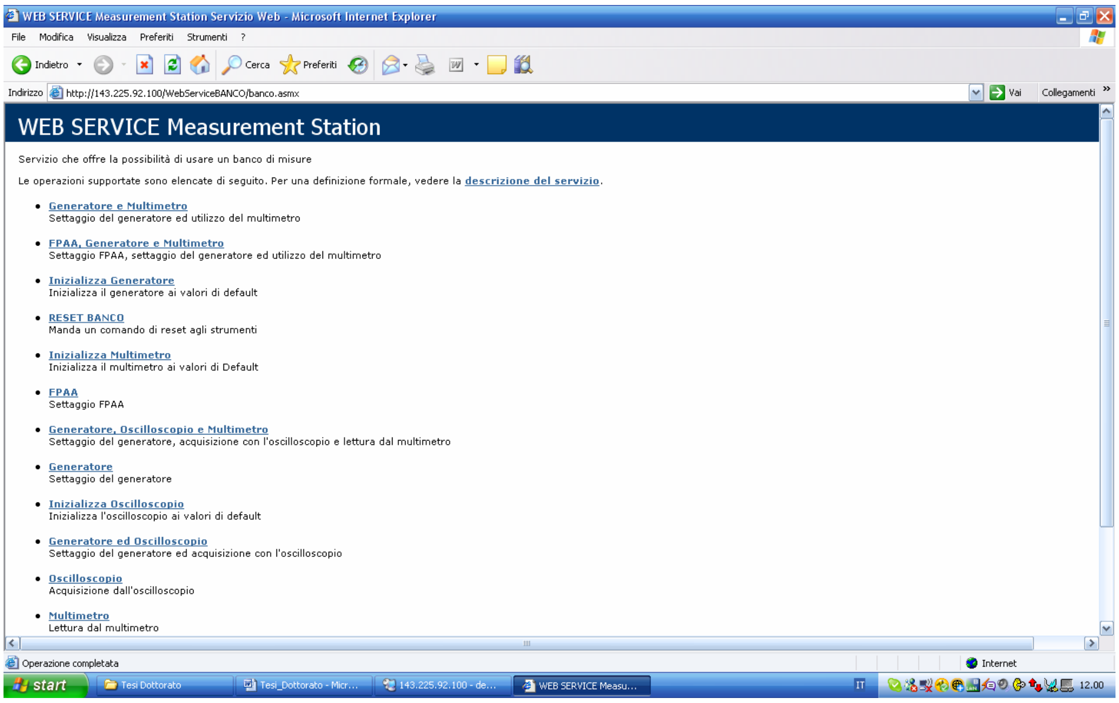
The SCPI_Generator class comprises only the command for setting the generator output (whose parameters are contained in the variables of Generator class in Instruments namespace). For the oscilloscope and the multimeter a command string and a query string, for attaining the measurement result, are considered.

II.7 Web Service Measurement_Station

As aforementioned, a Web Service is an application which publishes functions declared as Web Methods. Fig. II. 6 shows how the realized Web Service, for controlling the measurement instruments appears in a Web Browser to connecting users.

The service is a file with .asmx extension, that a browser can navigate, but it is different from a web page having .aspx extension. Web Services are, in fact, sets of available procedures, rather than complete applications; they, thus, are not characterized by their own interface. The page shown in Fig. II. 6 has the only purpose of exposing the offered service together with its methods and functionalities.

If user clicks the "service description" link, the page shown in Fig. II. 7 is displayed. It represents the WSDL file for invoking the service; whereas by clicking on a particular method, a page similar to that shown in Fig. II. 8 appears, where more specific information about the method are given. In particular, it is exposed the SOAP string that has to be sent for



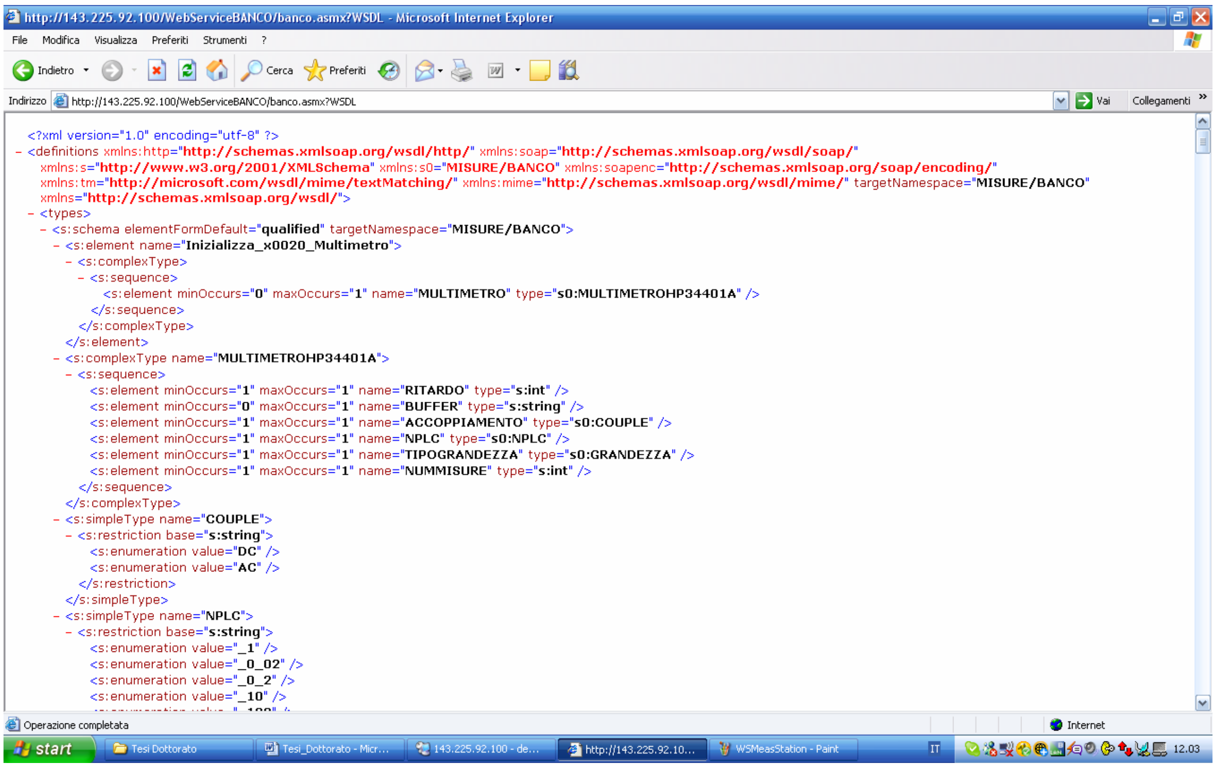
WEB SERVICE Measurement Station

Servizio che offre la possibilità di usare un banco di misure

Le operazioni supportate sono elencate di seguito. Per una definizione formale, vedere la descrizione del servizio.

- Generatore e Multimetro
  Settaggio del generatore ed utilizzo del multimetro
- FPAA, Generatore e Multimetro
  Settaggio FPAA, settaggio del generatore ed utilizzo del multimetro
- Inizializza Generatore
  Inizializza il generatore ai valori di default
- RESET BANCO
  Manda un comando di reset agli strumenti
- Inizializza Multimetro
  Inizializza il multimetro ai valori di Default
- FPAA
  Settaggio FPAA
- Generatore, Oscilloscopio e Multimetro
  Settaggio del generatore, acquisizione con l'oscilloscopio e lettura dal multimetro
- Generatore
  Settaggio del generatore
- Inizializza Oscilloscopio
  Inizializza l'oscilloscopio ai valori di default
- Generatore ed Oscilloscopio
  Settaggio del generatore ed acquisizione con l'oscilloscopio
- Oscilloscopio
  Acquisizione dall'oscilloscopio
- Multimetro
  Lettura dal multimetro

Fig. II. 6 – Web page of the service Measurement Station.


http://143.225.92.100/WebServiceBANCO/banco.asmx?WSDL - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro Cerca Preferiti
Indirizzo http://143.225.92.100/WebServiceBANCO/banco.asmx?WSDL
Vai Collegamenti
Operazione completata
Internet
start Tesi Dottorato Tesi_Dottorato - Micr... 143.225.92.100 - de... http://143.225.92.10... WSMeasStation - Paint
IT 12.03

```
  <?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:s0="MISURE/BANCO" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="MISURE/BANCO"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
  - <types>
    - <s:schema elementFormDefault="qualified" targetNamespace="MISURE/BANCO">
      - <s:element name="Inizializza_x0020_Multimetro">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="MULTIMETRO" type="s0:MULTIMETROHP34401A" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:complexType name="MULTIMETROHP34401A">
        - <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="RITARDO" type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="BUFFER" type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="ACCOPPIAMENTO" type="s0:COUPLE" />
            <s:element minOccurs="1" maxOccurs="1" name="NPLC" type="s0:NPLC" />
            <s:element minOccurs="1" maxOccurs="1" name="TIPOGRANDEZZA" type="s0:GRANDEZZA" />
            <s:element minOccurs="1" maxOccurs="1" name="NUMMISURE" type="s:int" />
          </s:sequence>
        </s:complexType>
      - <s:simpleType name="COUPLE">
        - <s:restriction base="s:string">
            <s:enumeration value="DC" />
            <s:enumeration value="AC" />
          </s:restriction>
        </s:simpleType>
      - <s:simpleType name="NPLC">
        - <s:restriction base="s:string">
            <s:enumeration value="_1" />
            <s:enumeration value="_0_02" />
            <s:enumeration value="_0_2" />
            <s:enumeration value="_10" />
```

Fig. II. 7 – WSDL file of the Web Service.

remotely using the method.

The main issues considered during the code development have been:

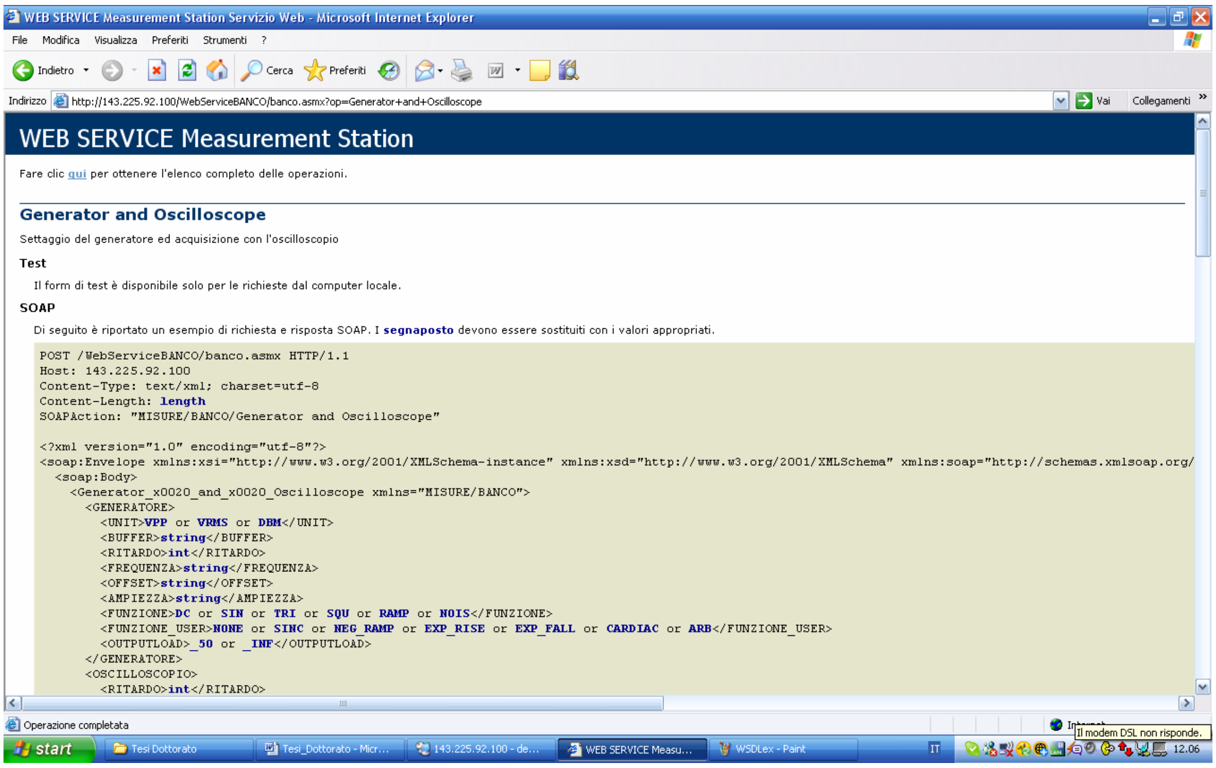
- Management of input/output parameters;
- Synchronization of the commands;
- Management of instruments status;
- Management of multiple accesses;
- Service life-time.

II.7.1 Management of input/output parameters

The first considered problem has concerned with the transmission of configuration parameters to the measurement instruments. The number of setting parameters, in fact, can be quite great for an oscilloscope, whereas it is reduced for the signal generator and the multimeter.

In order to adopt the same method for the configuration of both easy and more complicated instruments, a technique based on the exchange of proper objects containing the control parameters has been preferred. In particular, web methods receive instances of the classes of the namespace *Instruments*: object-*generator*, object-*multimeter* and object-*oscilloscope*.

The object are passed by reference, thus the invoked web method can modify them; in particular it writes in the variable buffer, inserting the values measured by the instrument. When user receives the method response, then, he obtains the sent object with the buffer filled by the measurement results.


WEB SERVICE Measurement Station Servizio Web - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro Cerca Preferiti
Indirizzo http://143.225.92.100/WebServiceBANCO/banco.asmx?op=Generator+and+Oscilloscope Vai Collegamenti
Operazione completata
Il modem DSL non risponde.
start Tesi Dottorato Tesi_Dottorato - Micr... 143.225.92.100 - de... WEB SERVICE Measu... WSDLex - Paint IT 12.06

WEB SERVICE Measurement Station

Fare clic qui per ottenere l'elenco completo delle operazioni.

Generator and Oscilloscope

Settaggio del generatore ed acquisizione con l'oscilloscopio

Test

Il form di test è disponibile solo per le richieste dal computer locale.

SOAP

Di seguito è riportato un esempio di richiesta e risposta SOAP. I **segnaposto** devono essere sostituiti con i valori appropriati.

```
POST /WebServiceBANCO/banco.asmx HTTP/1.1
Host: 143.225.92.100
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "MISURE/BANCO/Generator and Oscilloscope"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/
  <soap:Body>
    <Generator_x0020_and_x0020_Oscilloscope xmlns="MISURE/BANCO">
      <GENERATORE>
        <UNIT>VPP or VRMS or DBM</UNIT>
        <BUFFER>string</BUFFER>
        <RITARDO>int</RITARDO>
        <FREQUENZA>string</FREQUENZA>
        <OFFSET>string</OFFSET>
        <AMPIEZZA>string</AMPIEZZA>
        <FUNZIONE>DC or SIN or TRI or SQU or RAMP or NOIS</FUNZIONE>
        <FUNZIONE_USER>NONE or SINC or NEG_RAMP or EXP_RISE or EXP_FALL or CARDIAC or ARB</FUNZIONE_USER>
        <OUTPUTLOAD>_50 or _INF</OUTPUTLOAD>
      </GENERATORE>
      <OSCILLOSCOPIO>
        <RITARDO>int</RITARDO>
```

Fig. II. 8 – Method description of the service.

II.7.2 Synchronization of the commands

In order to successfully communicate with an instrument, each method has to comply with the correct commands sequence and timing. At this aim the Visual Basic .NET function *Sleep* has been employed. The function accepts as input the time (as number of milliseconds) during which the thread has to be stopped; this allows, so, to wait a settable time interval after the instrument configuration and before executing the measurement (for example, in order to wait the end of a transient).

The input variable of the Sleep method is contained in the *Delay* property of the object sent to the web method. The command sequence, for a multimeter, will be the following:

```
Multimeter.Write(commandmul) ‘configuration command is sent to the multimeter
Thread.Sleep(multimeter.DELAY)      ‘a settable time is waited
Multimeter.Write(querymultimeter)   ‘the query is sent
Multimeter.BUFFER=Multimeter.ReadString(16*Multimeter.NUMMEAS) ‘acquisition.
```

II.7.3 Management of multiple accesses

Another issue to be considered during the development of remote experiments is the correct management of simultaneous requests from different users. When a student is connected to the measurement station, he should have at his disposal all the resources and the instrument could not change their status because of requests sent by other clients. For example, if a user configures the signal generator and execute a measurement with the multimeter, modifications of generator parameter, due to another user's request, have to be avoided.

Unfortunately, web services are multithread, then more users invoking the same service are simultaneously handled. The generated threads are concurrent, with the same priority; so they could access to the same shared resource, i.e. the measurement station.

In order to avoid interferences between multiple requests, it has been preferred to reserve to the incoming user the whole GPIB bus, rather than the instruments. At this aim, the structure *SyncLock*, available in Visual Basic .Net, has been employed every time the user has to operate on a real peripheral [12]. Command sequences for measurement instruments, then, have been made thread-safe by encapsulating them into the structure:

```
Synclock
.
End Synclock
```

The code in the structure is entirely executed by one thread even if there are other threads generated by other requests. As an example, a code fragment assuring the control of the whole measurement station is reported:

```
    <WebMethod (Messagename:="Generator and Multimeter",
    Description:="Configuration of Generator and measurement with a Multimeter")>
    Public Function measurement(ByRef GENERATOR As GENERATORAGILENT33120A, ByRef
    MULTIMETER As MULTIMETERHP34401A, ByVal user As String) As Boolean
‘Syncronization of operations with Lock
Dim station As String = “locked station”
SyncLock measurement_station
Try
```

```
devgen.Write(commandgen)     'Configuration of generator
Catch
Return False
Finally
GENERATORE = gen
End Try
Thread.Sleep(gen.DELAY)
Try
devmul.Write(comandomul)     'Configuration of multimeter
Thread.Sleep(mul.DELAY)      'a delay is waited
devmul.Write(querymul)       'Query to the multimeter
mul.BUFFER = devmul.ReadString(16 * mul.NUMMEAS)
Catch
Return False
Finally
devgen.Dispose()
devmul.Dispose()
MULTIMETER = mul
End Try
End SyncLock    'The code fragment returns available for another thread
End Function
```

The method operates on a signal generator and a multimeter; the generator is configured and, after the configuration of the multimeter, the query to make the multimeter execute the measurement is sent. As shown, the SyncLock instruction has to be associated to a variable shared between all the threads. In the code this variable is *Station*, that, when the station is reserved to a unique user, has the value "locked station".

II.7.4 Management of instruments status

ASP application, then also Web Services, are stateless: the status of variables is lost between two different service invocations. Usually this problem is solved by means of the management of sessions through cookies. The solution implemented during the research work solves this restraint by defining the services lifetime on the basis of a configuration and an acquisition. Since parameters are exchanged by reference, they are both input and output parameters; they can be used, then, to keep the status of an instrument and to record the last configuration. In practice: *(i)* user sends a configuration by sending variables with proper value; *(ii)* when the web service response is received, the variable contains the values previously sent and the updated value; *(iii)* when, finally, the user execute the query for a measurement, he sent the same variable with the same values he has received. This way the instrument configuration is recorded by the client.

II.7.5 Service life-time

The service life-time starts when the configuration parameters are sent and it ends when measurement results are returned (usually few seconds). This way, if an user is operating on the measurement station, another user has to wait a very short time. The different users, in fact, can alternatively access the measurement instruments without interfering each others. The faster the service execution, the lower the probability of bottle neck in the presence of high number of requests.

User is unaware of this management of multiple accesses, since he considers the measurement station always available, but with variable response time.

II.8 Description of Web Methods

In Fig. II. 6 the service *Measurement Station*, has been shown, with its declared web methods, that in the following will be explained in detail.

II.8.1 The method ExecuteMeasurement

Actually there are six methods with the name *ExecuteMeasurement*, accepting different input parameters (this characteristic of Visual Basic .NET functions is known as overloading). According to the input parameters the correct method will be used.

The method receives, as inputs, the object-*instrument*, passed by reference, and a parameter identifying the user, passed by value. It extracts the configuration to be sent to the real instruments and fills the buffer with the instrument response.

As example, it is reported the code of the method configuring the signal generator and the oscilloscope and returning the oscilloscope measurements on both the channels.

```
<WebMethod(Messagename:="Generator ed Oscilloscope",
  Description:="Configuration of the signal generator and acquisition with
  oscilloscope")>
Public Function Execute_Measurement (ByRef GENERATOR As
  GENERATOR.GENERATORAGILENT33120A, ByRef OSCILLOSCOPE As
  OSCILLOSCOPE.OSCILLOSCOPETDS210) As Boolean
```

```
 ‘declaration of the object-instruments
Dim gen As New GENERATOR.GENERATORAGILENT33120A
Dim osci As New OSCILLOSCOPE.OSCILLOSCOPETDS210
Dim devgen As New LOGDEVICE(Application.Get("ibus"),
 Application.Get("iaddressgen"))
Dim devosci As New LOGDEVICE(Application.Get("ibus"),
 Application.Get("iaddressosci"))
‘declaration of the object containing the SCPI commands for the instruments
Dim commandSCPIgen As New COMMAND.SCPI_GENERATOR
Dim comandoSCPIosc As New COMMAND.SCPI_OSCILLOSCOPE
Dim commandgen, commandosci, comCh1, comCh2, comtrg As String
‘the length of the oscilloscope channels is declared
Dim len1 As Integer
Dim len2 As Integer
'a variable for the oscilloscope autoset is declared
Dim bauto As Boolean
gen = GENERATORE
osci = OSCILLOSCOPIO
len1 = osci.CHANNEL1.DATASTOP - osci.CHANNEL1.DATASTART + 8
len2 = osci.CHANNEL2.DATASTOP - osci.CHANNEL2.DATASTART + 8
devgen.INSTRUMENT = "GENERATOR"
devosci.INSTRUMENT = "OSCILLOSCOPE"
devgen.LOGATTIVO=Application.Get("blogattivo") ‘the access log is activated
devosci.LOGATTIVO=Application.Get("blogattivo")
'a name is assigned to the user for the log
devgen.USER = user
devosci.USER = user
'the SCPI command to be sent to the instrument is constructed
commandgen = commandSCPIgen.GPIBcommandGen(gen)
commandSCPIosc.oscilloscope = osci
bauto = osci.AUTOSET
commandosci = commandSCPIosc.GPIBCommandOsci()
comCh1 = commandSCPIosc.queryChannel1()
comCh2 = commandSCPIosc.queryChannel2()
comtrg = commandSCPIosc.commandTrigger
Try
SyncLock banco 'the application is locked
Try 'invio il comando al generatore
devgen.Write("*CLS")
If gen.FUNCTION_USER = INSTRUMENT.GENERATOR.FUNCTION.USER.ARB Then
devgen.IOTimeout = TimeoutValue.None
End If
devgen.Write(commandgen)
Catch
Return False
```

```
Finally
GENERATOR = gen
End Try
Thread.Sleep(gen.RITARDO)
Try 'invio dei comandi e lettura dall'oscilloscopio
Try 'settaggio generale
devosci.Write(commandosci)
If Not bauto Then 'if autoset is true, command trigger is not sent
devosci.Write(comtrg)
osci.BUFFER = " "
Else ‘if autoset is true, oscilloscope configuration is read
devosci.Write("*LRN?")
osci.BUFFER = devosci.ReadString(1500)
End If
Catch ex As Exception
Return False
End Try
Thread.Sleep(osci.DELAY)
If osci.CHANNEL1.ACTIVE Then 'eventual command for channel1
Try
devosci.Write(comCh1)
devosci.Write("CURVE?")
Select Case osci.CODING
Case INSTRUMENT.OSCILLOSCOPE.STRUCTUREBASE.ENC.ASCII
osci.CHANNEL1.BUFFER = devosci.ReadString(len1 * 5)
Case INSTRUMENT.OSCILLOSCOPE.STRUCTUREBASE.ENC.RPB
osci.CHANNEL1.BUFFER = devosci.ReadByteArray()
End Select
Catch
Return False
End Try
Else
osci.CHANNEL1.BUFFER = " "
End If
If osci.CHANNEL2.ACTIVE Then 'eventual command for channel2
Try
devosci.Write(comCh2)
devosci.Write("CURVE?")
Select Case osci.CODING
Case INSTRUMENT.OSCILLOSCOPE.STRUCTUREBASE.ENC.ASCII
osci.CHANNEL2.BUFFER = devosci.ReadString(len2 * 5)
Case INSTRUMENT.OSCILLOSCOPE.STRUCTUREBASE.ENC.RPB
osci.CHANNEL2.BUFFER = devosci.ReadByteArray()
End Select
Catch
```

```
85 Return False
86 End Try
87 Else
88 osci.CANALE2.BUFFER = " "
89 End If
90 Try 'trigger mode is set to auto again
91 devosci.Write("TRIG:MAI:MOD AUTO;TYP EDGE")
92 Catch ex As Exception
93 End Try
94 Catch ex As Exception
95 Finally
96 OSCILLOSCOPE = osci
97 End Try
98 End SyncLock 'unlock of the resources
99 Catch ex As Exception
100 Return False
101 Finally
102 devgen.Dispose()
103 devosci.Dispose()
104 End Try
105 Return True
106 End Function
```

As stated above, the function is declared as Web Method to be remotely invoked (row 1); the method will be invoked from Internet by using the name defined in the *messagename* attribute. Whereas the description defines what will appears to users in the help page. The objects of type *Instrument*, containing the instruments configuration, and *LOGDEVICE*, containing the information to be written in a log file, are declared (rows 3 and 4). The objects of type *Command* (rows 7 and 8) manipulate the object-*instrument* for creating the SCPI command strings.

From row 30 to 98 the instructions sequence thread-safe has been inserted; the structure Try…Catch…EndTry is used for capturing eventual errors due to a bad communication with instruments (for example if an instrument is not connected to the server or not supplied). If an error occurs, the method ends and returns false, without affecting the whole application.

If user wants to generate an arbitrary function, the timeout of GPIB bus is disabled, since the transmission of 16000 samples can take more time (rows 34-36). From row 41 to 55 the configuration procedure is synchronized: *(i)* the generator is configured; *(ii)* a time delay is waited; *(iii)* the oscilloscope is configured; and *(iv)* a second time delay is waited.

The communication with the oscilloscope is between the rows 43 to 92. If the autoset function is active, the trigger command is not sent, whereas oscilloscope configuration is read and collected in the buffer. Otherwise the trigger command is transmitted and the acquisition is performed according to the number of channels activated (rows 56-83).

After the acquisition, the oscilloscope is configured again by activating the autoset (row 91). The objects, completed with the measurement results, are finally returned to user (row 96).

II.8.2 The method Reset

Although the system has proved to be quite robust, accidental failures due to the transmission of conflicting configurations have to be considered. It has been noticed, for example, that if an oscilloscope receives incorrect commands, it stops its functioning, refusing other command on GPIB line. In order to remotely overcome these conditions, a procedure for the instruments reset has been developed. The realized method calls an executable driver, *RESET.EXE*, implemented in National Instruments LabVIEW^{TM} environment, which sends the reset commands to instruments through GPIB bus.

II.8.3 The method Station_Test

This method queries the instruments with “*IDN?” command in order to test the instruments status without asking them for a measurement. In the following the method code is reported.

```
<WebMethod(Messagename:="STATION TEST", Description:="Executes a test on the
instruments of the measurement station")> _
Public Function STATION_TEST() As String
Dim devgen As New LOGDEVICE(Application.Get("ibus"),
Application.Get("iaddressgen"))
Dim devmul As New LOGDEVICE(Application.Get("ibus"),
Application.Get("iaddressmul"))
Dim devosci As New LOGDEVICE(Application.Get("ibus"),
Application.Get("iaddressosci"))
Dim qtest As String = "*IDN?" 'Test query to be sent
Dim err As String
Dim answer As String
devmul.INSTRUMENT="MULTIMETER"
devosci.INSTRUMENT="OSCILLOSCOPE"
```

```
devgen.INSTRUMENT="GENERATOR"
devgen.LOGACTIVE=Application.Get("blogactive")
devmul.LOGACTIVE=Application.Get("blogactive")
devosci.LOGACTIVE=Application.Get("blogactive")
devgen.USER = user
devmul.USER = user
devosci.USER = user
Dim idgen, idmul, idosc, idstation As String
idstation = Application.Get("stationname")
Try
SyncLock station
Try
devgen.Write(qtest)
idgen = "GEN :" & devgen.ReadString(1000)
Catch ex As Exception
idgen = "GEN :error for generator"
End Try
Try
devmul.Write(qtest)
idmul = "MUL :" & devmul.ReadString(1000)
Catch ex As Exception
idmul = "MUL :error for multimeter"
End Try
Try
devosci.Write(qtest)
idosc = "OSC :" & devosci.ReadString(1000)
Catch ex As Exception
idosc = "OSC :error for oscilloscope"
End Try
answer = "TEST OK ; STATION " & idstation & " AVAILABLE; INSTRUMENTS: " & idgen
& ";" & vbCrLf & idmul & ";" & vbCrLf & idosc & vbCrLf
End SyncLock
Catch ex As Exception
answer = "TEST ERR ; STATION " & idstation & " : STATION NOT AVAILABLE; "
End Try
Return answer
End Function
```

As illustrated, the test query is sent to all the instrument; if a peripheral does not respond, an error is caught. The code aims at the construction of the response string, which contains an OK or ERROR message for the entire measurement station and for each instrument.

II.8.4 Traffic control: the LOGDEVICE class

The data transmitted through GPIB bus are recorded in a log file. At this aim a new class *LOGDEVICE*, schematized in Fig. II. 9, has been developed.

It derives from the *Device* class, but the methods read and write have been slightly modified. The methods, in fact, apart from sending the relative commands to the instruments and receiving data from them, writes the data in a file (if the property *LOGACTIVE* is true) as shown in the following code section.

```
Public Shadows Function Write(ByVal command As String) As Boolean
MyBase.Write(command)
  If bactive Then
  LOG(command, "WRITE")
  End If
End Function
  . . .
Private Sub LOG(ByVal text As String, ByVal command As String)
'this function writes the text and the command
Try 'if the file already exist, data has to be appended
If myfile.Exists(spathfile & snamefile) Then
mywriter = myfile.AppendText(spathfile & snamefile)
Else 'otherwise another file is created
mywriter = myfile.CreateText(spathfile & snamefile)
End If 'write file
      mywriter.WriteLine(Now.ToString & vbTab & suser & vbTab & snamestr &
  vbTab & command & ": " & vbTab & text)
Catch ex As Exception
Finally
```

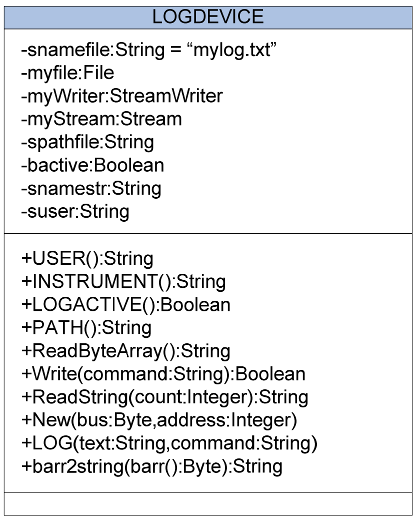


Fig. II. 9. Methods of the LogDevice class.

```
mywriter.Close()
End Try
   End Sub
   ...
```

The keyword *Shadows* makes the function not visible to users. In order to write a text file, in Visual Basic .NET environment two object have to be created: *file* and *stream*. The file object is necessary for opening or closing the file, whereas data to be written have to be inserted in the stream object.

It is worth noticing the use of the *user* attribute, which allows to register who accessed the station and the operations that he made.

In Fig. II. 10 it is finally shown the architecture of the whole Web Service *Measurement Station*.

II.9 Web Service Stations Management

The laboratory architecture has been realized so that a personal computer with a GPIB board connected to the instrumentation and with the web service *Measurement Station* installed is an instrumentation server. The laboratory is, so, flexible and scalable since it is very easy to add a new measurement station.

At the higher hierarchical level, a management server is necessary, for directing the incoming requests towards the measurement stations, due to the following reasons:

- Management of different requests: the various stations can be connected to different instruments; a management service, knowing the instruments composing each station, forwards the requests to the proper *Measurement Station* service.
- Authentication: this service should assure the login only of the authorized users and the logging operations.
- Security: only the management service communicates with the different *Measurement Station* services, which are not directly accessible by the clients.

The web service mandated to the accesses management, is *Stations Management*. Fig. II. 11 schematizes the so obtained architecture.

Fig. II. 12 shows the web service *Station Management*. The exposed functions are very similar to the methods of *Measurement Station* service, but they receive as input, apart from the objects-instruments, an object-user. The functions, successively, invoke the methods with their same name belonging to a particular service *Measurement Station*. The object-user is

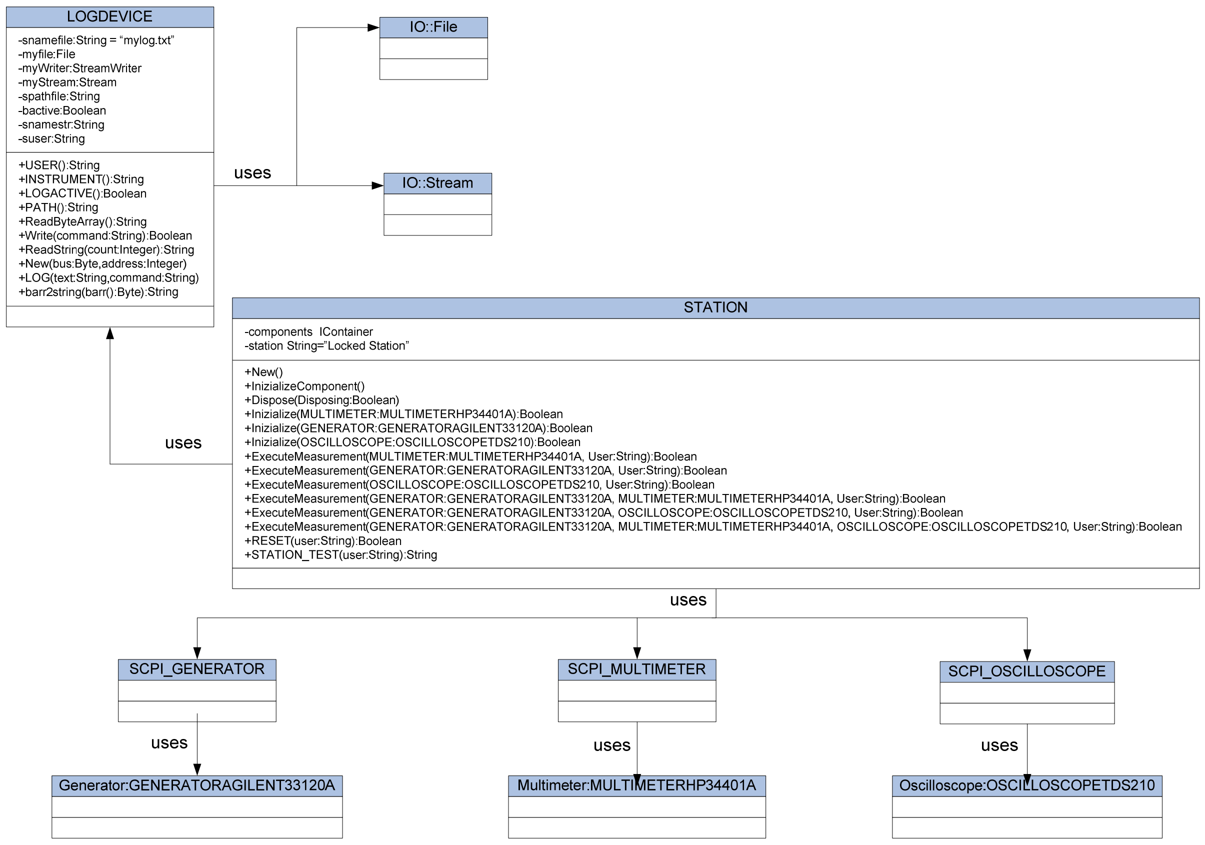

LOGDEVICE
-snamefile:String = “mylog.txt”
-myfile:File
-myWriter:StreamWriter
-myStream:Stream
-spathfile:String
-bactive:Boolean
-snamestr:String
-suser:String
+USER():String
+INSTRUMENT():String
+LOGACTIVE():Boolean
+PATH():String
+ReadByteArray():String
+Write(command:String):Boolean
+ReadString(count:Integer):String
+New(bus:Byte,address:Integer)
+LOG(text:String,command:String)
+barr2string(barr():Byte):String
uses
IO::File
IO::Stream
uses
STATION
-components IContainer
-station String=”Locked Station”
+New()
+InizializeComponent()
+Dispose(Disposing:Boolean)
+Inizialize(MULTIMETER:MULTIMETERHP34401A):Boolean
+Inizialize(GENERATOR:GENERATORAGILENT33120A):Boolean
+Inizialize(OSCILLOSCOPE:OSCILLOSCOPETDS210):Boolean
+ExecuteMeasurement(MULTIMETER:MULTIMETERHP34401A, User:String):Boolean
+ExecuteMeasurement(GENERATOR:GENERATORAGILENT33120A, User:String):Boolean
+ExecuteMeasurement(OSCILLOSCOPE:OSCILLOSCOPETDS210, User:String):Boolean
+ExecuteMeasurement(GENERATOR:GENERATORAGILENT33120A, MULTIMETER:MULTIMETERHP34401A, User:String):Boolean
+ExecuteMeasurement(GENERATOR:GENERATORAGILENT33120A, OSCILLOSCOPE:OSCILLOSCOPETDS210, User:String):Boolean
+ExecuteMeasurement(GENERATOR:GENERATORAGILENT33120A, MULTIMETER:MULTIMETERHP34401A, OSCILLOSCOPE:OSCILLOSCOPETDS210, User:String):Boolean
+RESET(user:String):Boolean
+STATION_TEST(user:String):String
uses
SCPI_GENERATOR
SCPI_MULTIMETER
SCPI_OSCILLOSCOPE
uses
uses
uses
Generator:GENERATORAGILENT33120A
Multimeter:MULTIMETERHP34401A
Oscilloscope:OSCILLOSCOPETDS210

Fig. II. 10 – Architecture of the entire Measurement Station service.

characterized by: *(i)* a login and *(ii)* a password identifying the client; *(iii)* a request about the station to be used; and *(iv)* a message, where the measurement result or necessary information will be written.

So the service *Station Management* invokes the different *Measurement Station* services on the various laboratory stations. At this aim, the proxy classes of the *Measurement Station*

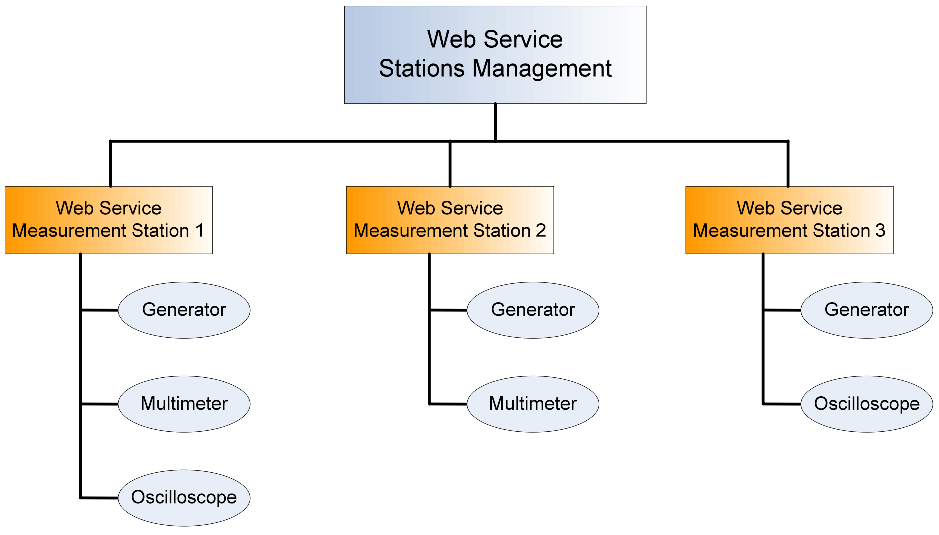
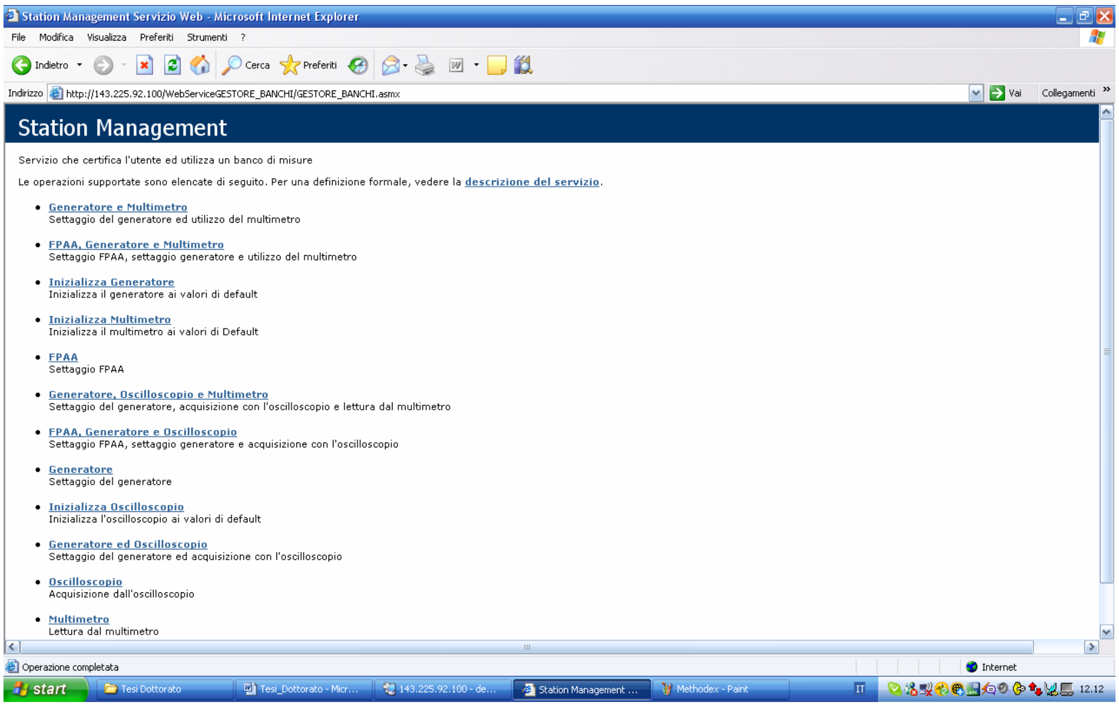


Fig. II. 11 – Scheme of the Web Service Station Management.

services have been included into the service *Station Management*. The proxy class is employed as a typical Visual Basic .NET class; the property used, in particular, is the URL, which represents the URL of the *Measurement Station* service to be invoked.

Particular attention has been paid to the function *Authenticate*, which has two tasks: (i) to verify if the user is authorized to access the remote laboratory and (ii) to obtain the address of the measurement station to which the user's request will be sent. At this aim, the object OleDBConnection is employed, in order to obtain from the database where information about registered users and measurement stations. First, users login and password are searched among the allowed users in the table *Authorized Users*. Finally, a measurement station able to satisfy user's request is searched; in particular, the service obtains from the database the URL of the service *Measurement Station* associated to the selected laboratory physical station.


Station Management Servizio Web - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro Cerca Preferiti
Indirizzo http://143.225.92.100/WebServiceGESTORE_BANCHI/GESTORE_BANCHI.asmx
Vai Collegamenti
Operazione completata
Internet
start
Tesi Dottorato
Tesi_Dottorato - Micr...
143.225.92.100 - de...
Station Management ...
Methodex - Paint
IT
12.12

Station Management

Servizio che certifica l'utente ed utilizza un banco di misure

Le operazioni supportate sono elencate di seguito. Per una definizione formale, vedere la descrizione del servizio.

- Generatore e Multimetro
  Settaggio del generatore ed utilizzo del multimetro
- FPAA, Generatore e Multimetro
  Settaggio FPAA, settaggio generatore e utilizzo del multimetro
- Inizializza Generatore
  Inizializza il generatore ai valori di default
- Inizializza Multimetro
  Inizializza il multimetro ai valori di Default
- FPAA
  Settaggio FPAA
- Generatore, Oscilloscopio e Multimetro
  Settaggio del generatore, acquisizione con l'oscilloscopio e lettura dal multimetro
- FPAA, Generatore e Oscilloscopio
  Settaggio FPAA, settaggio generatore e acquisizione con l'oscilloscopio
- Generatore
  Settaggio del generatore
- Inizializza Oscilloscopio
  Inizializza l'oscilloscopio ai valori di default
- Generatore ed Oscilloscopio
  Settaggio del generatore ed acquisizione con l'oscilloscopio
- Oscilloscopio
  Acquisizione dall'oscilloscopio
- Multimetro
  Lettura dal multimetro

Fig. II. 12 – Web page of the service Station Management.

II.10 Example of Client applications in LabVIEW^{TM}

The main advantage of solutions based on Web Services relies on the interoperability. The services explained above can be, thus, invoked by programs developed in a wide variety of software environment. In this section an application realized in National Instruments LabVIEW^{TM} software is shown.

LabVIEW 7.0 and successive releases provide a specific tool, the WSbrowser, which allows to easily invoke a web service if the .NET Framework is installed on client computer [13]. By inserting the URL of the desired service, the WSbrowser builds the code containing the service proxy class and, so, a dll file, which has to be saved in the same directory of the LabVIEW client program. The client application will contain a "constructor node", which makes use of the created dll file, and an "invoke node" method, which automatically shows as input the web service input.

Actually, WSbrowser and .NET Framework are not indispensable to invoke web service; they just send an HTTP request containing the correct SOAP message allowing the communication with the service without knowing low level data exchange. Then, another solution, which does not force users to download other software, has been developed.

The typical format of SOAP request and service response is displayed in the web page of the method description; then the HTTP request can be built by the client program properly concatenating strings and can be transmitted through TCP/IP protocol to the service. The facilities required to a software environment for developing the client program, then are only two: (i) strings manipulation for building the SOAP request; and (ii) use of TCP/IP protocol.

The algorithm executed by the program client has to follow these operating steps:

- Conversion of data in string format.
- Translation of strings to XML language
- Creation of SOAP request, properly formatting the strings.
- Transmission of SOAP request to the server through TCP (port 80).
- Waiting for the SOAP response from the server.
- Conversion of XML data to strings.
- Conversion of strings in local data.

The main section of the VI client is the SubVI *VI_Station*, whose inputs are four clusters, containing the instruments setting and user information: 1) Generator, 2) Multimeter, 3) Oscilloscope and 4) User. The function outputs are a cluster, namely Oscilloscope, and a string containing potential message to the user. The input clusters are sets of configuration parameters, with variables of different type (enumerative, string, number). The *VI_Station* has to serialize, according to XML language, each parameter before sending the request to the web service. At this aim, a proper serialization VI (*XMLElem*) has been developed, whose block diagram is shown in Fig. II. 13; from the variables value, name and namespace the VI aims at building the XML element:

<NAME xmlns="NAMESPACE">VALUE</NAME>

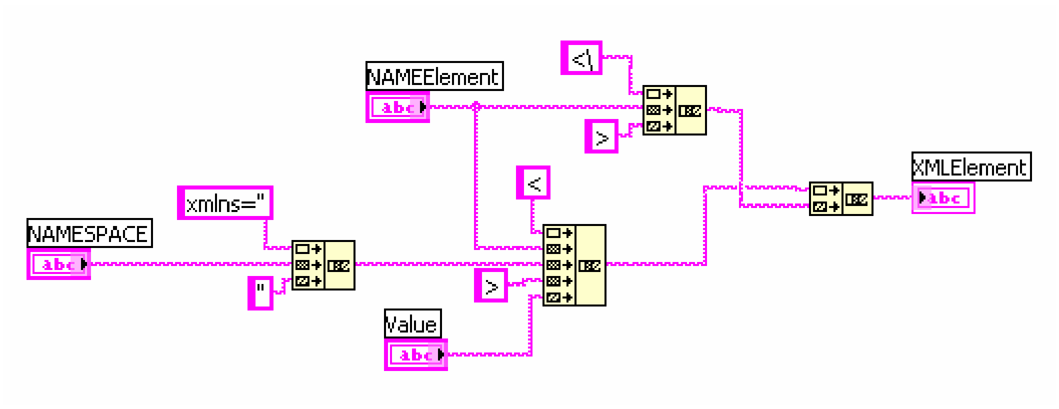


Fig. II. 13 – Block diagram of XMLElem VI

Then, for each cluster, a VI, making use of *XMLElem* has to serialize the whole cluster. As an example, Fig. II. 14 shows the VI *XMLSerUser* for the serialization of the variables encapsulated in the User cluster.

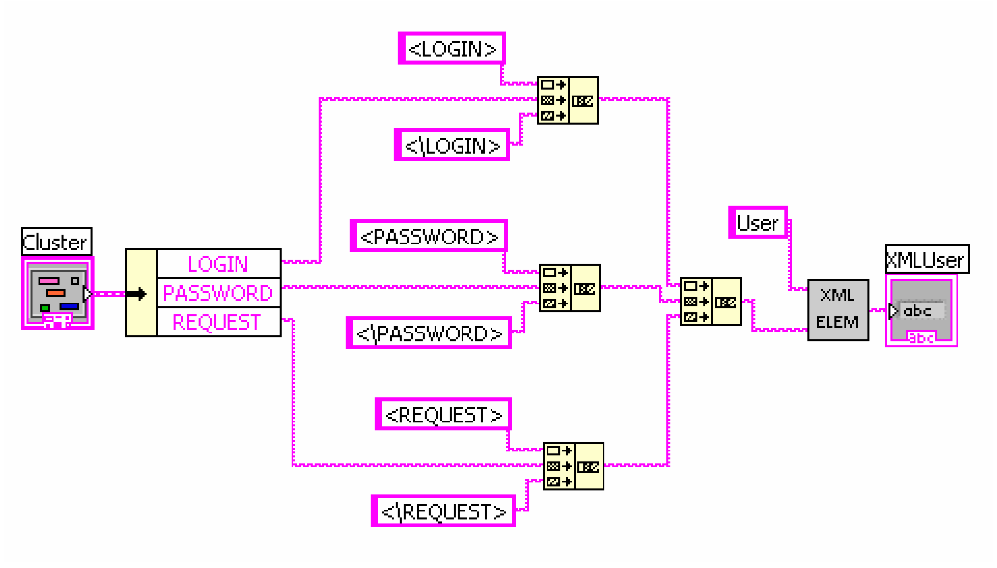


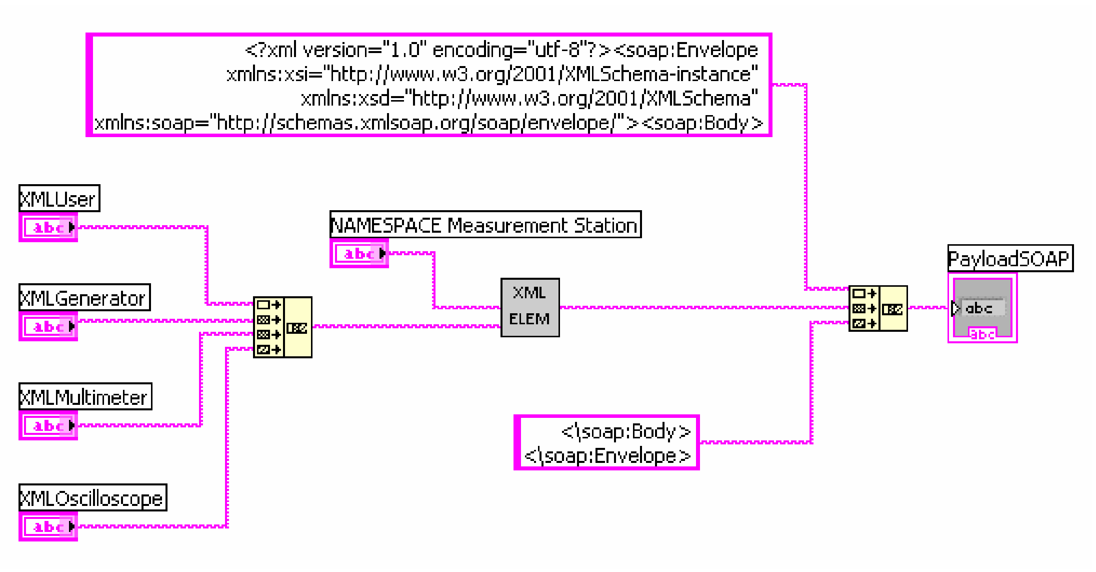
Fig. II. 14 – Block Diagram of XMLSerUser VI.

The VI output is a string containing user's information formatted according to XML language:

<LOGIN>Username<\LOGIN>
<PASSWORD>Password<\PASSWORD>
<REQUEST>User's Request<\REQUEST>

In a similar way, other VIs for the serialization of generator, multimeter and oscilloscope data have been implemented.

Fig. II. 15 illustrates the VI *Payload*, which, starting from the XML data, builds up the string containing the whole SOAP envelope.



<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body>
XMLUser
XMLGenerator
XMLMultimeter
XMLOscilloscope
NAMESPACE Measurement Station
XML ELEM
PayloadSOAP
<\soap:Body> <\soap:Envelope>

Fig. II. 15 – Block Diagram of Payload VI.

Finally, the HTTP request has to be formatted. At this aim, the VI *HTTPReq* (shown in Fig. II. 16) has been developed. It concatenates to the SOAP string the header necessary to execute an HTTP POST request.

The VI *SendRequest* (shown in Fig. II. 17) opens aTCP connection and executes a TCP write on port 80 of the server on which is located the service; successively, it waits for the

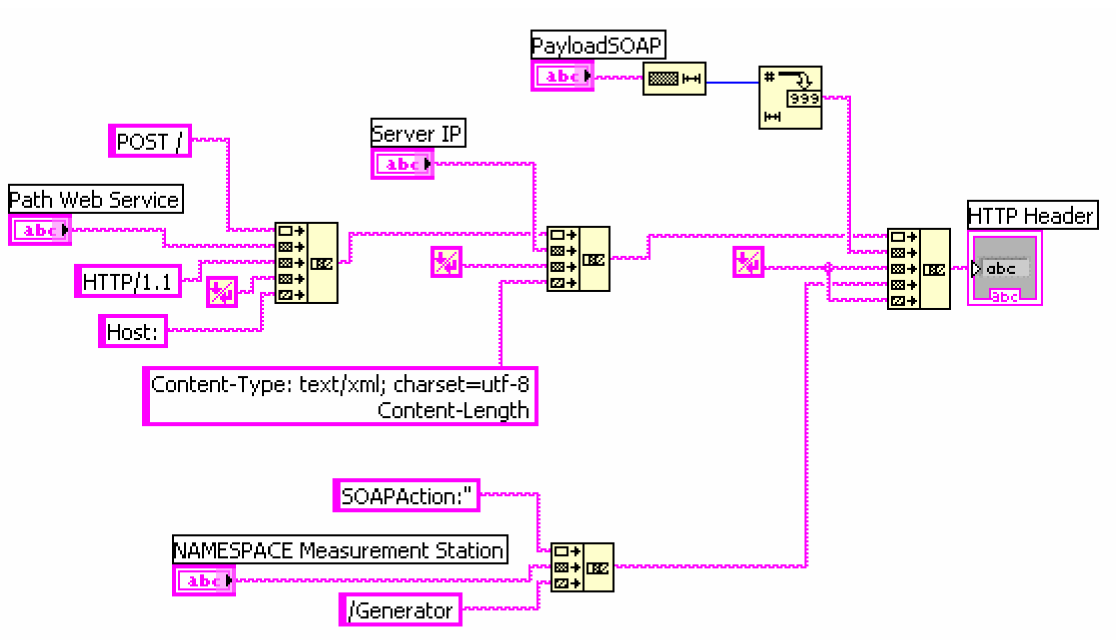


Fig. II. 16 – Block diagram of HTTPReq VI.

service response on port 80 of client computer. When 200ms have elapsed and data have not received, the read function returns a timeout error; the error is erased and the loop while is repeated. Otherwise, 512 bytes are read and concatenated to the previously received byte and a new iteration starts. The cycle ends when the entire message has been received or when an error different from timeout has occurred. The VI output is a string containing the SOAP response of the service, which has to be properly divided. This operation is performed by the OutputBuffer VI, which scans the string searching for the desired variables (in particular the oscilloscope channels and the multimeter response).

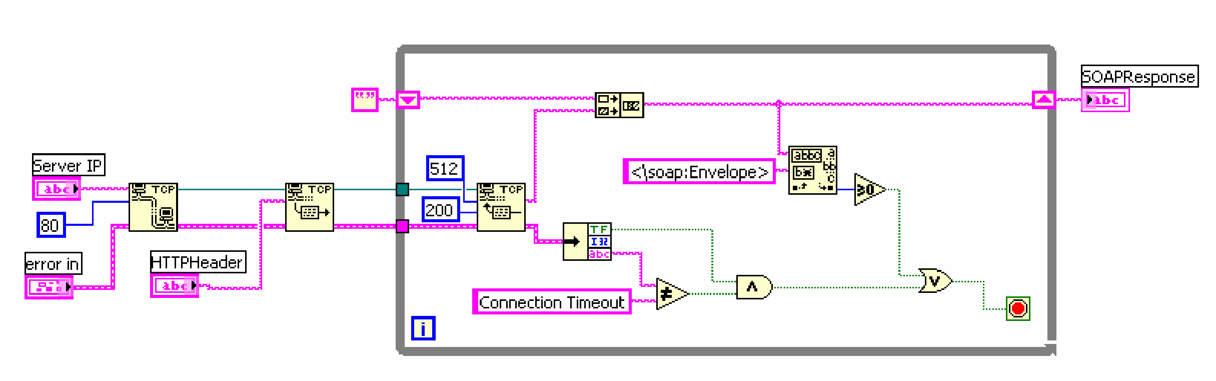


Fig. II. 17 – Block diagram of SendRequest VI.

II.11 Example of Application in Macromedia Flash

In this section an example developed in Measurement for Diagnostics and Quality (MDQ) course is presented. In this instance laboratory experiences are necessary, as students have to collect an amount of data coming from a process under control and evaluate some statistical features of the process itself. The course, in fact, deals with quality survey of industrial processes in order to verify if the process is compliant to existing quality Standards [14]. Typical experimental activities in MDQ course and subsequent calculations operate for performing the following analysis: (i) Normality Test; (ii) Time variation of quality indexes [15] through the examination of control charts; (iii) Hypothesis Test; (iv) Analysis of Variance [16]; and (v) Development of Design Of Experiments for parametrically optimizing experiments [17]. One of the primary tools for quality management is statistical process control (SPC), a methodology that involves a statistical sampling of measurable parameters of a process output. To this aim, a parameter is cyclically sampled at equal time interval and time evolution is plotted. The analysis is performed through the Control Charts, by collecting a specified number of samples of the selected parameter in repeated time intervals (each hour, as an example) and plotting (i) the means of the samples acquired each hour (X chart); (ii) the maximum difference between the highest and the lowest parameter value in each hour (R chart); and (iii) the sums, for each hour, of the differences between the means of values in an hour and the mean of all the acquired samples (Cusum chart). So a second laboratory session is required, to make this amount of data available. Then students have to evaluate the dependence of the monitored variate of the process on the changing of a parameter P1. In particular, they carry out a first measurement, setting the parameter to a value P1a; successively, they sample the variate of interest again, setting P1 to the value P1b. From these two sets of data, students, by implementing a Hypothesis Test, have to estimate with what confidence level they can assume that a variation of the parameter P1 causes a variation of the probability distribution of the controlled variate. Another statistical analysis often used, in studying the evolution of industrial processes, is the Analysis of Variance (ANOVA). This tool allows recognizing the variability of the monitored variate as due to random factors or to the variation of a parameter influencing the process. In particular, students are asked to implement a two ways ANOVA, employed to estimate how the variability of a physical quantity is related to the variation of two independent parameters or their interaction. A further laboratory experience, thus, is required, since students have to collect four set of data; if P1 and P2 are the parameters on which the variate of interest X is supposed to depend, undergraduates have to measure X setting to parameter to the values: (i) P1=P1a and P2=P2a; (ii) P1=P1a and P2=P2b; (iii) P1=P1b and P2=P2b; and (iv) P1=P1b and P2=P2a. The test intends to assess the hypotheses that the parameter is influenced by P1, by P2, or by the interaction between P1 and P2. It is conducted by evaluating the variances of the parameter for each value of P1 and P2; these variances are divided by the variance of the error that takes into account the randomness of the process. Once a confidence level has been fixed, comparing the obtained ratio with the related value of the F variate (Fisher's test), it is possible to decide if the aforementioned hypotheses can be accepted or not.

It is so clear that a valuable section of the course is hence dedicated to laboratory experiences as support to traditional lectures. A test circuit, representing an industrial process is shown and explained to students; they have to properly set up the measurement instrumentation, carry out the measurement process and collect data to be statistically treated. At least four laboratory lessons are required to make at students' disposal a real system for applying the studied theory, collecting data to process, having a better comprehension of the

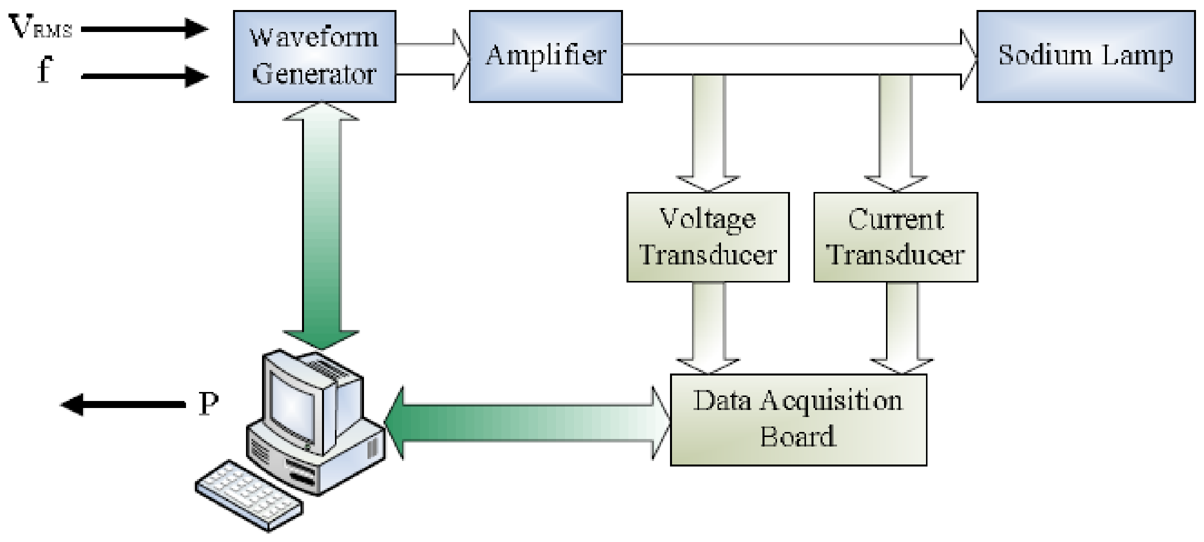


Fig. II. 17 - Scheme of the station for power measurement.

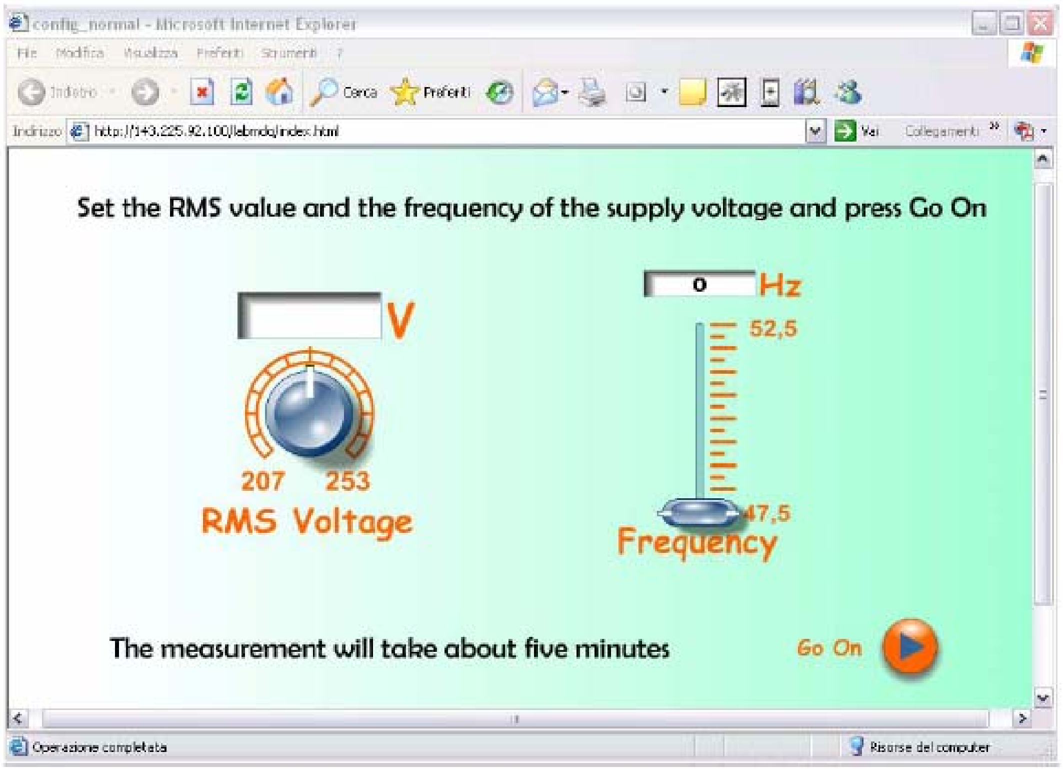
statistical tools. Since in last years about three hundreds students have pursued the course, the arrangement of the laboratory sessions has proved to be unreasonable. Starting from these considerations, it has been employed the developed remote laboratory.

Since students' knowledge of LabVIEW and instrument control is not required, it has been implemented a client application exploiting the graphical capability of Macromedia Flash, which allows building up interface pages particularly "user-friendly". The experiment concerns electrical power measurement (Fig. II. 17). The measurand is the active power absorbed by a high pressure sodium lamp characterized by a nominal power of 250W; the parameter P1 and P2 influencing the process are the frequency and the RMS value of the supply voltage. RMS voltage and frequency are set by controlling the waveform generator to deliver the voltage having the selected characteristics. Voltage and current samples, that have to be processed in order to evaluate the active power absorbed by the lamp, are collected through the oscilloscope [18]. After user's authentication, by inserting a valid user name and password, the students access the web page shown in Fig. II. 18. They can select the VRMS supplying the sodium lamp and the voltage frequency. Once pushed the "Go On" button, the code in the Flash page invokes the *Generator and Oscilloscope* method of the *Measurement Station* Web Service.

Therefore, students can collect data over a number of different operating conditions and plot curves of the results. The measurement process lasts five minutes, at the end of which the software is programmed to gather the power data and create a text file of the data versus elapsed time. Users see the measurement results in tabular and graphical format and can choose to download them. Students have performed a preliminary measurement, in order to estimate, from the numerical data, the time duration of the electrical transient and have to detect the time instant from which the system can be considered in a steady-state condition. In fact, the successive statistical analysis can be conducted only with data acquired from a system in steady-state condition.

Therefore, data of the transitory interval have to be discarded. Then undergraduates have accessed the remote laboratory to gather data for executing the Normality Test. An example of histogram realized with acquired data is depicted in Fig. II. 19. It is clearly noticeable that data constitute a probability distribution that fits a Gaussian distribution with a high confidence level (95%). The test is executed by comparing the histogram of the observed occurrence frequencies with the one obtained by a Gaussian distribution having the same mean and standard deviation of the acquired distribution. It is so calculated a factor F.M. that is the sum of squared differences, for each histogram class, between the observed frequencies and the expected ones. If F.M. is lower than the statistical variate $\chi 2$ associated to the degree of freedom and to the fixed confidence level, it can be assumed that the collected sample comes from a Gaussian distribution.

Students have so assumed that the measurement process has not been influenced by exceptional factors. Subsequently, data have been acquired each thirty minutes to elaborate


Set the RMS value and the frequency of the supply voltage and press Go On

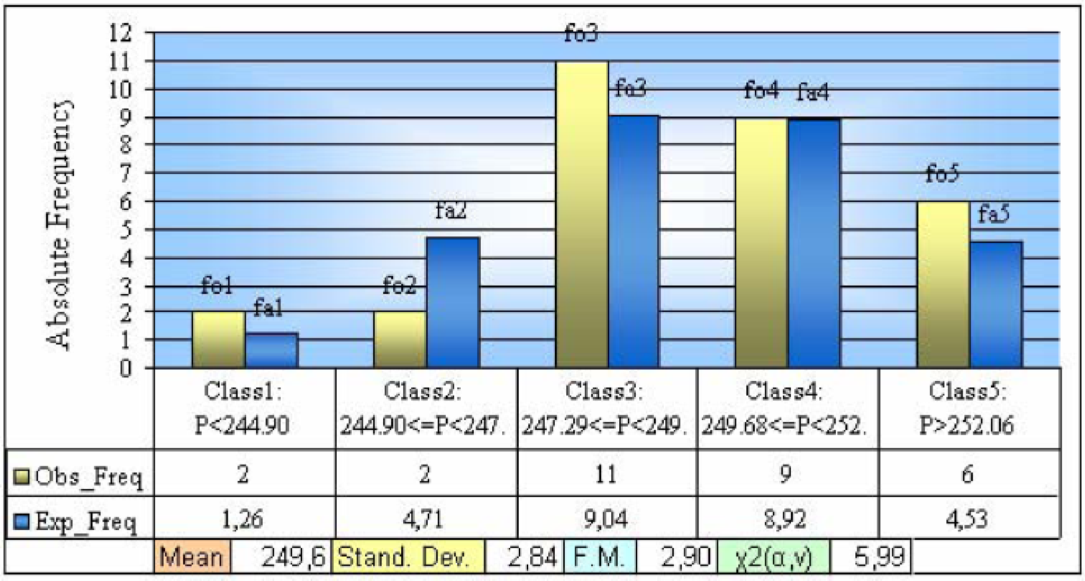
The measurement will take about five minutes

config_normal - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro
Cerca
Preferiti
Indirizzo http://143.225.92.100/labmdq/index.html
Vai
Collegamenti
0
Hz
V
52,5
47,5
207
253
RMS Voltage
Frequency
Go On
Operazione completata
Risorse del computer

Fig. II. 18 - Client application in Flash environment.

the control charts. An example is illustrated in Fig. II. 20, where it can be observed that the measurement process can be assumed under control and cannot be found particular events that could lead the process out of control. The Upper and Lower Control Limits (UCL and LCL) are derived from the samples distribution as: UCL = $\mu+3\sigma$ LCL=$\mu-3\sigma$ Once all the students have executed the measurement for the χ square test and the control charts, they have been empowered to perform a second series of measurement. This time they have changed one of the two control parameter (frequency or RMS voltage) and recorded a new set of data.

By combining the new data with the one previously acquired, students, hence, have implemented the Hypothesis Test. In Fig. II. 21 it is shown an example where it is possible to see that the power supply frequency variations strongly influence the power absorbed by the lamp. In fact, with a high degree of confidence it can be assumed that the power data sampled by supplying the lamp with a different power frequency f2 approximate a normal distribution with a mean $\mu 2$ different from the mean $\mu 1$ of the distribution fitted by the first set of data, obtained with f=f1. In the final part of the course, then, undergraduates have accessed the laboratory to execute the measurement for obtaining data for ANOVA test. In particular, they have changed the RMS voltage to the value V2 and have gathered two sets of data: the first by setting the frequency equal to f1 and the second with the frequency f2. Mixing this data to the one acquired with V=V1, students have been capable of perform the ANOVA test. As expected, absorbed power results to depend on both the power frequency and VRMS.



	Class1: P<244.90	Class2: 244.90<=P<247.	Class3: 247.29<=P<249.	Class4: 249.68<=P<252.	Class5: P>252.06
Obs_Freq	2	2	11	9	6
Exp_Freq	1,26	4,71	9,04	8,92	4,53

Mean	249,6	Stand. Dev.	2,84	F.M.	2,90	$\chi 2(\alpha,v)$	5,99

Fig. II. 19 - Comparison between observed and expected frequencies in normality test.

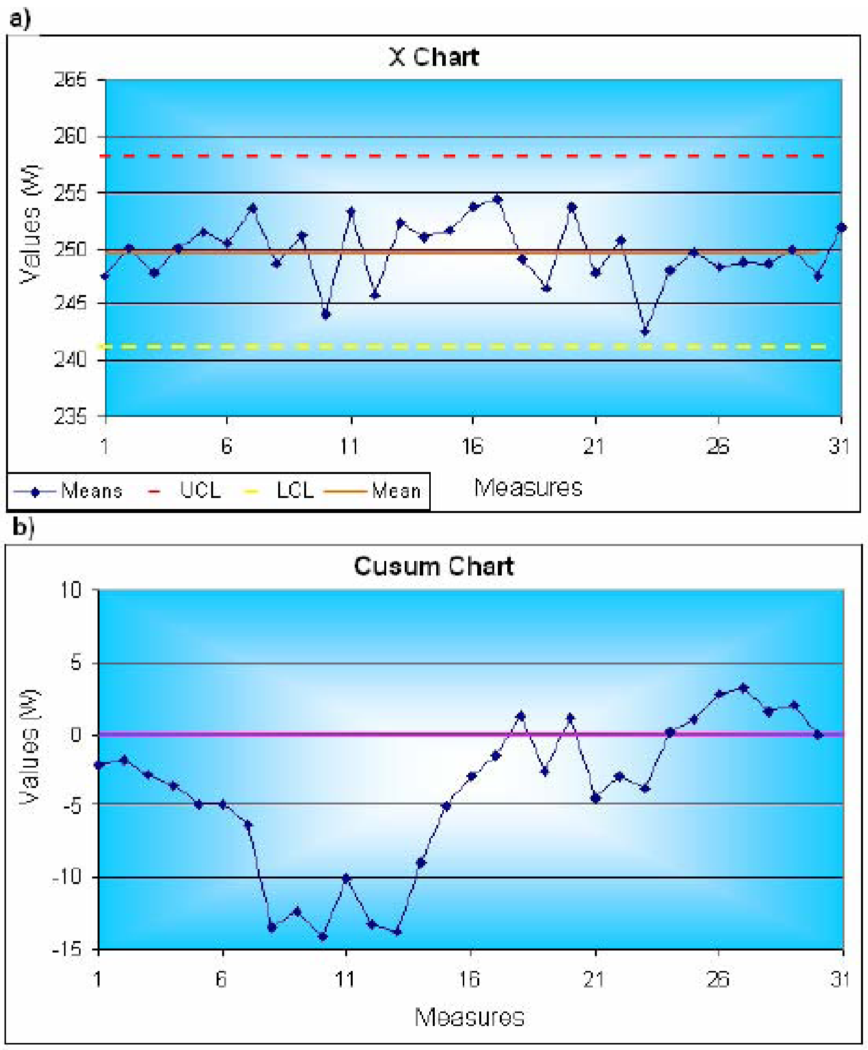


Fig. II. 20 - Control charts: a) X chart; b) Cusum chart.

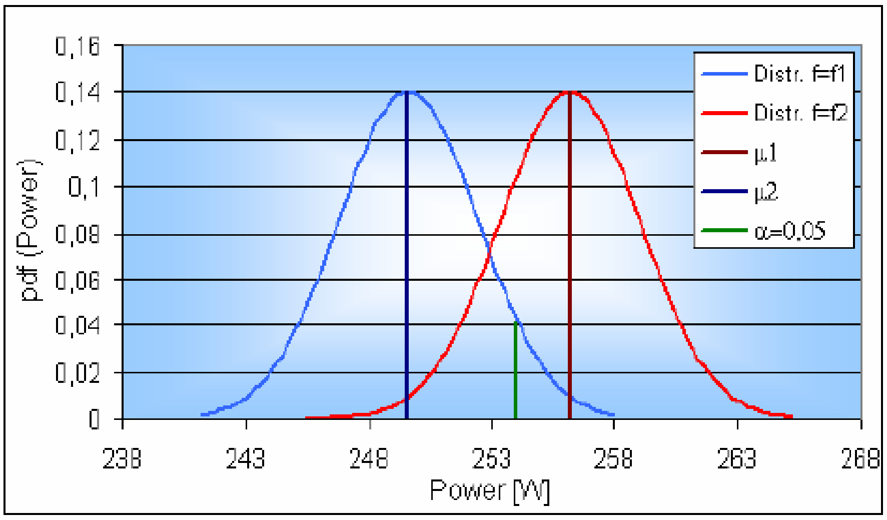


Fig. II. 21 - Data distributions with two different power frequencies.

References

[1] C. De Capua, A. Liccardo, "An E-Learning portal for Electrical and Electronic Measurement courses employing remote instrument control", *IADAT Journal on Advanced Technology*, Vol. 1, n. 4, pp. 174-176, Luglio 2005.

[2] G. Andria, A. Baccigalupi, M. Borsic, P. Carbone, P. Daponte, C. De Capua, A. Ferrero, D. Grimaldi, A. Liccardo, N. Locci, A. M. L. Lanzolla, D. Macii, C. Muscas, L. Peretto, D. Petri, S. Rapuano, M. Riccio, S. Salicone, F. Stefani, "Remote Didactic Laboratory "G. Savastano": the Italian Experience for the E-learning at the Technical Universities in the Field of the Electrical and Electronic Measurements, Overview on Didactic Experiments", *Proc. of IEEE Instrum. and Meas. Tech. Conf.*, pp. 1537-1542, 24-27 Aprile 2006, Sorrento, Italy, 24-27 April 2006.

[3] URL: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[4] URL: *www.w3.org/XML*, "Extensible Markup Language (XML)".

[5] G. Bucci, F. Ciancetta, E. Fiorucci, D. Gallo, C. Landi, "A Low Cost Embedded Web Services for Measurements on Power System", Proc. of IEEE Virtual Envir., human-Computer Interf and Meas. Systems, Giardini-Naxos, Italy, pp. 7-12, 2005.

[6] F. Balena. Visual Basic .NET ASP.NET & WEB SERVICES. Mondatori Informatica, 2003.

[7] H.M. Deitel, P.J. Deitel, T.R. Nieto. *Visual Basic.NET Programmazione avanzata e Web Services* Milano:Apogeo, 2003.

[8] *IEEE Computer*, Special Issue on Web Services, 2003.

[9] P. Daponte, C. De Capua, A. Liccardo, "A technique for remote management of instrumentation based on Web Service", *Proc. Of IMEKO 13th Symposium on Measurements for Research and Industry Applications*, Athens, Greece, pp. 687-692, Sept. 2004.

[10] C. De Capua, A. Liccardo, R. Morello, "On the Web Service-based remote didactical laboratory: further developments", *Proc. of IEEE Instrum. and Meas. Tech. Conf.*, Vol.3, pp. 1692-1696, Ottawa, Canada, 16-19 Maggio 2005.

[11] URL: http://www.uddi.org.

[12] MSDN Library di Visual Studio.NET 2003

[13] URL: *http://zone.ni.com*, "Building a Web Service-based application in LabVIEW^{TM} 7.0".

[14] CEI EN ISO 9001, Quality management systems – Requirements, 2004.

[15] G. Taguchi, S. Chowdhury, Taguchi's Quality Engineering Handbook, John Wiley & Sons, 2004.

[16] W. J. Diamond, Practical Experimental Designs, Second Ed., Van Nostrand Reinhold, NY, 1989.

[17] D. C. Montgomery, Design and Analysis of Experiments, Fifth Edition, John Wiley & Sons, New York, NY, 2000.

[18] C. Landi, A. Liccardo, N. Polese, "Remote Laboratory Activities to Support Experimental Session for Undergraduate Course of Measurements for Diagnostics and Quality", *Proc. of IEEE Instrum. and Meas. Tech. Conf.*, pp. 851-855, Sorrento, Italy, 24-27 Aprile 2006

Chapter III - DEVICE UNDER TEST REMOTELY CONFIGURABLE

III.1 Introduction

In this section the innovations produced on the hardware architecture of a remote laboratory during the research activity will be described. As mentioned in Chapter I, at the present, users accessing a remote laboratory are able to control only the measurement instruments, which are equipped with a proper communication interface board. Typically, the test circuit electrically wired to the instrumentation, is made of analog components not capable of receiving communication data.

This limitation of remote laboratory could lead to a lack of effectiveness of teaching. Students have information about the analog test circuit, but cannot change anything of it; they, thus, could think that the multimeter or oscilloscope results are provided by a simulated experiment, becoming bored and less interested.

The measurement station, then, has to be wired by technicians at the university where the physical laboratory is located, according to the experiment that students have to remotely perform. It is clear, so, that for each desired experiment, a proper measurement station is required; that means to employ a server, a signal generator, multimeters, oscilloscopes, test circuit, electrical cables, GPIB and Ethernet board and cables. Since, in order to reduce the queue of requests to the laboratory, a number of identical stations could be desirable, laboratory resources and space are inexorably reduced.

Moreover, students are able to perform a limited number of experiment, whereas, as it is well known, in most applied engineering courses such as in electrical and electronics circuit theory, different topics are covered in rapid succession; it is important that students grasp these concepts very quickly.

In order to overcome these limitation, the research activity has been focused on the realization of a test circuit able to be programmed. At this aim, innovative integrated devices recently present on the market, namely Field Programmable Analog Arrays (FPAA), have been employed. FPAA are configurable through RS232 communication and are able to realize the transfer function of different analog circuits. It is, so, possible to implement a “universal” measurement station, where students select the analog circuit on which perform the experiment and control the instrument to execute the measurement.

III.2 Field Programmable Analog Array Technology

By the end of the last decade a new type of analog circuit, based on configurable blocks, the Field Programmable Analog Arrays (FPAAs) was introduced to the market. The FPAA is a single chip, based upon switched capacitor circuit technology that can be easily configured and dynamically reconfigured, in order to implement a variety of analog functions, called Configurable Analog Modules (CAMs): sum, amplification, differentiation, integration, filtering [1].

The internal architecture of the FPAA used during the research activity is shown in Fig. III. 1; it is an AN221E04, a component of the second generation family of FPAAs by Anadigm Company [2]. Analog resources are contained in multiple identical Configurable Analog Blocks (CAB) which incorporate operational amplifiers, capacitor banks, CMOS switches and SRAM. Analog functions such as programmable gain stages, adders, rectifiers, sample and hold circuits, and first order filters can be implemented in a single CAB. Higher level functions, such as biquadratic filters, level detectors, and so on, can be implemented using two or more cells [3].

Configuration data, when are received, are stored in a configuration Shadow RAM; when data have been entirely received and their correctness has been verified, data block is transferred from the Shadow RAM to the configuration RAM [4]. Then, after only five clock cycle, the FPAA is configured. Fig. III. 2 shows the scheme of a CAB. Among the many analog switches within the CAB, some are static and determine things like the general CAB

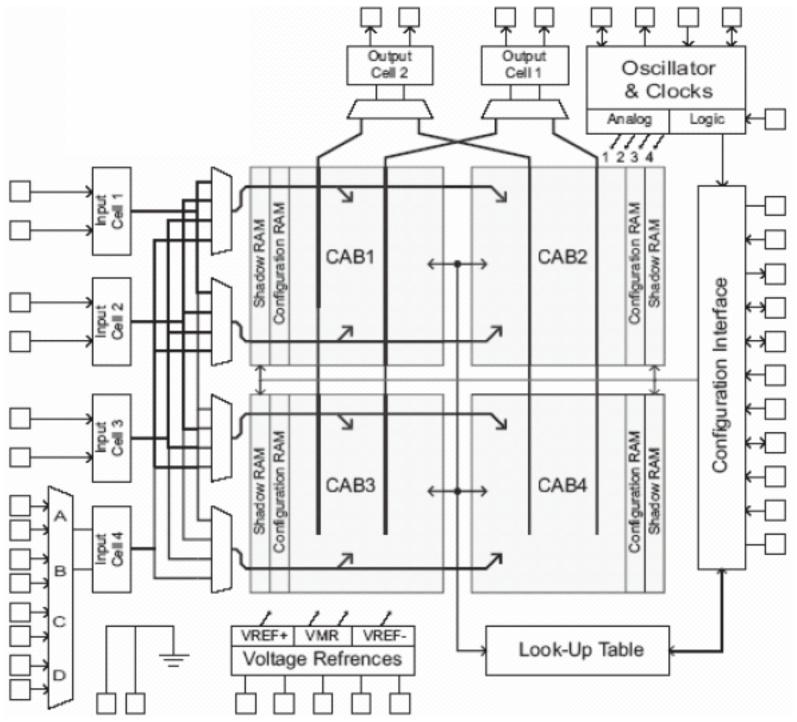


Fig. III. 1 – Architectural scheme of FPAA.

circuit connections, capacitor values, and which input is active. Other switches are dynamic and can change under control according to the phase of the clock selected [5]. Whether static or dynamic, all of the switches are controlled by the Configuration SRAM.

Next it is a bank of 8 programmable capacitors. Each of these 8 capacitors is actually a very large bank of very small but equally sized capacitors. Each of these 8 programmable

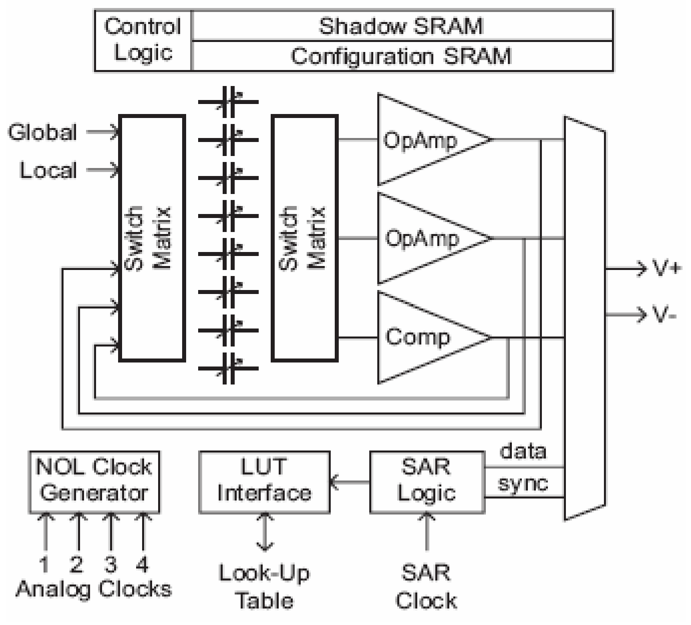


Fig. III. 2 – Scheme of a Configurable Analog Block.

capacitors can take on a relative value between 0 and 255 units of capacitance.

Obviously, limiting factors are present in any analog circuit, as a limited bandwidth and/or non-linearity. So, also FPAA technology is expected to exploit such unwanted characteristics. Furthermore, capacitors in FPAA can assume only integer values, so the effect of quantization errors should be considered. As a consequence of the aforementioned factors, the FPAA output will be somewhat different from the expected one [6]. Thus, FPAAs to be used in measurement applications require an accurate metrological characterization of the implemented blocks in order to allow the estimation of the involved uncertainty.

III.3 Switched Capacitor Technique

The Switched Capacitor (SC) technique is based on the consideration that a capacitor switched periodically between two circuit nodes is equivalent to a resistor connecting these nodes if the average value of current (over a period of time exceeding a number of times the switching period) is considered [7].

A circuit diagram for this basic SC circuit is shown in Fig. III. 3. During the time when the switch S1 is closed and S2 is opened, the capacitor C is charged to the voltage applied to the input, V1. So the total charge on the capacitor C, in steady state conditions, is: $Q_1 = C \cdot V_1$.

When the switches position changes, i.e. S1 is opened and S2 is closed, C is charged or discharged depending on the applied voltage V2; however, in steady state, the total charge on the capacitor is: $Q_2 = C \cdot V_2$.

The net charge transferred from the input to the output during one switching period is then:

$$\Delta Q = Q_1 - Q_2 = C(V_1 - V_2) \tag{1}$$

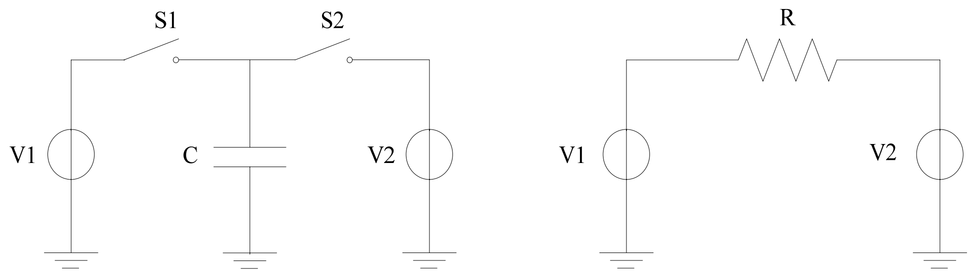


Fig. III. 3 – SC circuit implementing a resistance.

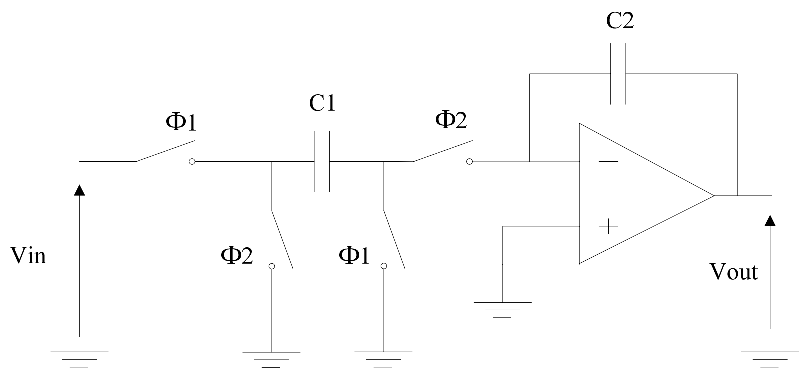


Fig. III. 4 - SC circuit equivalent to an integrator.

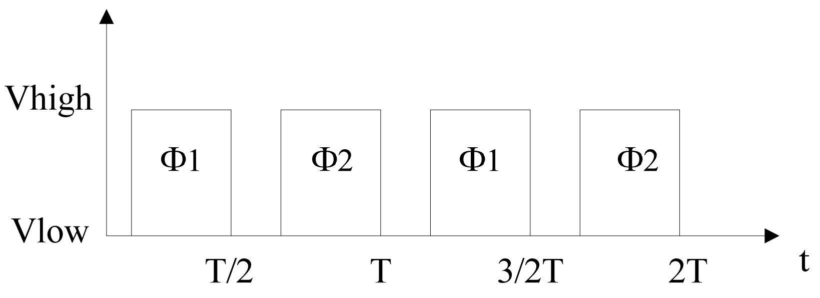


Fig. III. 5 - Switching clock scheme for an integrator.

The mean value of the current flowing from the input to the output is:

$$I = \frac{\Delta Q}{T} = \frac{C(V_1 - V_2)}{T} \tag{2}$$

This is equivalent to a current *I* flowing in a resistor whose value can be easily calculated:

$$I = \frac{V_1 - V_2}{R} \Rightarrow R = \frac{T}{C} \tag{3}$$

By properly modifying the switching period T and the capacitance value C, a wide range of resistances can be obtained. As stated above, by adopting only capacitors, it is possible to obtain a circuit behaving as a resistance, but with better accuracy, low power consumption and, so, reduced dimensions [8].

Combining the scheme in Fig. III. 3, with op amps, various analog functions can be achieved [9]. As an example, in Fig. III. 4 an integrator circuit, realized through switched capacitors is shown. Φ1 and Φ2 represent the phases during which a group of switches is closed, according to the clock scheme depicted in Fig. III. 5; T is the circuit clock, called master clock.

III.4 Local FPAA Configuration

In the following the operations required for the AN221E04 FPAA configuration are described. A section of the research activity has been based on the comprehension of the configuration technique, in order to implement the circuit programming independently from the software tool furnished by the manufacturer.

The values of the programmable components of the CAB cannot be changed directly by the user, since the programming and constructive details are left unknown by Anadigm. In order to configure the desired circuit into the FPAA, the provided development design environment, Anadigm Designer 2® software, along with a set of pre-built modules [10], has to be used. The user can link these modules and set parameters like block gain, central frequency of filters, integration constants and thresholds of comparators. As an example, in

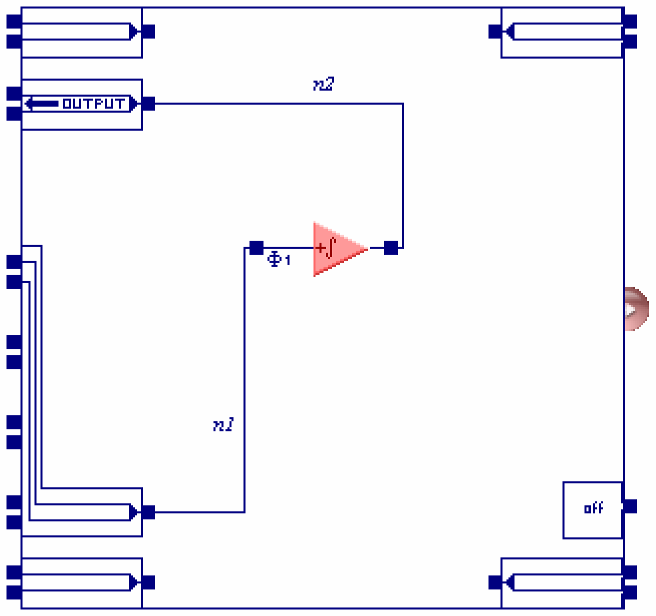


Fig. III. 6 – Implementation of an integrator in Anadigm Designer.

Fig. III. 6, an integrator block has been used and connected to FPAA input/output. Once the circuit has been designed, Anadigm software programs and configures the FPAA device as requested building the block configuration data and transmitting it through serial communication. This operation is called Primary configuration and it aims at the definition of the analog circuit topology.

FPAAs offer, moreover, the opportunity of rapidly modifying the characteristics of the analog circuit programmed with the primary configuration [11]. As an example, user can change the integration constant of the circuit in Fig. III. 6, without resetting the device. This operation, known as on-the-fly reconfiguration, just modify the value of the proper capacitance in the device RAM. Unfortunately, Anadigm Designer software does not provides facilities for the device reconfiguration; then the proper string has to be assembled by the developer. In fact, the device documentation provides the relations between the analog circuit parameters and the device capacitances; once the capacitances have been obtained, a reconfiguration string, according to the FPAA specifications, has to be constructed.

III.4.1 Configuration string structure

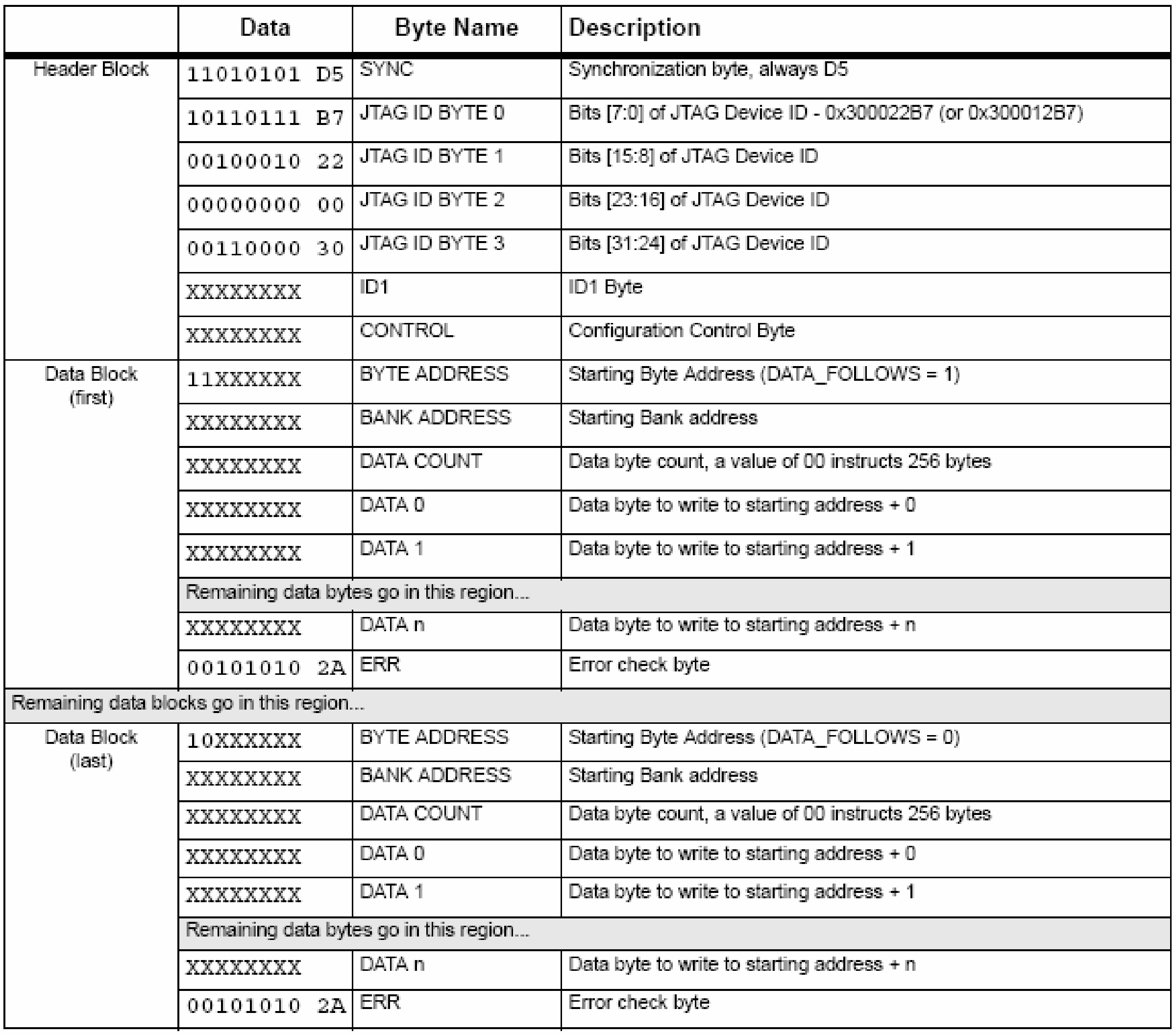
In Fig. III. 7 the data format of both the primary configuration and the dynamic reconfiguration as expected from the FPAA device is shown. The header block comprises seven bytes, particular for each FPAA model, in order to synchronize the communication. Each data block is made of:

- Byte Address: the memory address of the byte from which data have to be written; one bit contains the information “data follows” that is zero when the last data block is transmitted.
- Bank Address: the memory address of the bank where data have to be written.
- Data Count: length, in bytes, of the data block.
- Data: data to be recorded.
- Error Check: a control byte sent when the entire data block has been transmitted.

III.4.2 Development board

The manufacturer provides the FPAA AN221E04 mounted on a development board, the AN221K04, shown in Fig. III. 8. The developed board is equipped with various components:

- A microcontroller PIC 16F876 by Microchip which receives the data bytes from RS232, verifies that data are organized according to the format expected by FPAA and forwards them to the device digital lines. The PIC manages also the device reset, in order to erase the RAM content, when a new Primary Configuration is received.
- An RS232 port for data exchange connected, through a voltage level adapter (MAX232), to the microcontroller.
- A 16MHz oscillator, providing the clock signal to the FPAA; from it, four different clock signals (whose frequency is equal to the oscillator frequency divided by a



	Data	Byte Name	Description
Header Block	11010101 D5	SYNC	Synchronization byte, always D5
	10110111 B7	JTAG ID BYTE 0	Bits [7:0] of JTAG Device ID - 0x300022B7 (or 0x300012B7)
	00100010 22	JTAG ID BYTE 1	Bits [15:8] of JTAG Device ID
	00000000 00	JTAG ID BYTE 2	Bits [23:16] of JTAG Device ID
	00110000 30	JTAG ID BYTE 3	Bits [31:24] of JTAG Device ID
	XXXXXXXX	ID1	ID1 Byte
	XXXXXXXX	CONTROL	Configuration Control Byte
Data Block (first)	11XXXXXX	BYTE ADDRESS	Starting Byte Address (DATA_FOLLOWS = 1)
	XXXXXXXX	BANK ADDRESS	Starting Bank address
	XXXXXXXX	DATA COUNT	Data byte count, a value of 00 instructs 256 bytes
	XXXXXXXX	DATA 0	Data byte to write to starting address + 0
	XXXXXXXX	DATA 1	Data byte to write to starting address + 1
	Remaining data bytes go in this region...		
	XXXXXXXX	DATA n	Data byte to write to starting address + n
	00101010 2A	ERR	Error check byte
Remaining data blocks go in this region...			
Data Block (last)	10XXXXXX	BYTE ADDRESS	Starting Byte Address (DATA_FOLLOWS = 0)
	XXXXXXXX	BANK ADDRESS	Starting Bank address
	XXXXXXXX	DATA COUNT	Data byte count, a value of 00 instructs 256 bytes
	XXXXXXXX	DATA 0	Data byte to write to starting address + 0
	XXXXXXXX	DATA 1	Data byte to write to starting address + 1
	Remaining data bytes go in this region...		
	XXXXXXXX	DATA n	Data byte to write to starting address + n
	00101010 2A	ERR	Error check byte

Fig. III. 7 – Primary configuration data stream structure.

power of two) can be derived, in dependence on the desired switching frequency of FPAA components.

- Four analog input/output interface block, whose circuit is in Fig. III. 9. These four blocks are each connected to both analog input and output pin of the AN221E04 FPAA device. This allows all 4 blocks to be used as either an input or output interface. The selection is made by the developer, creating a short circuit between the proper pins, by means of four jumpers. Whereas, by positioning other jumpers it is possible to set differential as single-ended input.

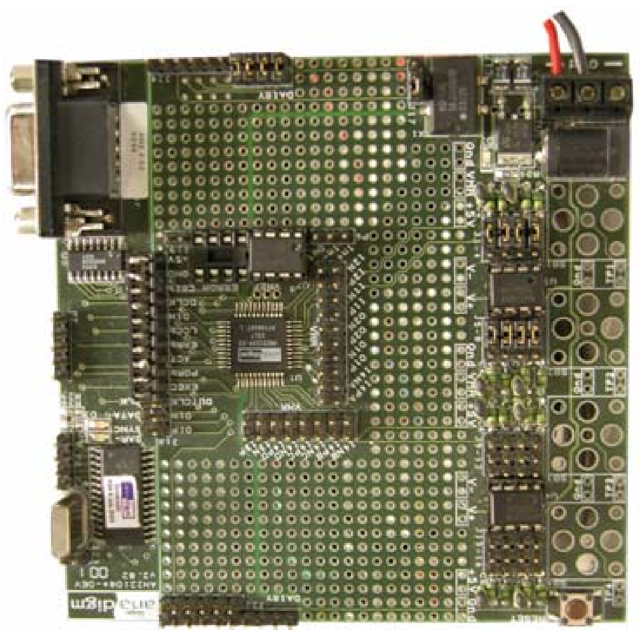


Fig. III. 8 – Development board AnadigmAN221K04.

III.4.3 Transmission of configuration data

As stated above, the microcontroller received the primary configuration from RS232 port and transmit it to the digital lines of FPAA, which are listed in Fig. III. 10. The PIC configures FPAA, implementing a synchronous serial protocol, through its I/O lines which control:

- Power On Reset (PORb) signal, whose rising edge carries the FPAA in its reset state (operation which takes about 30ms).
- Data In (DIN) signal, used by FPAA to read data transmitted by DSC.
- Digital Clock (DCLK) signal, whose rising edges latch valid data on DIN line.

FPAA primary configuration is performed through the operating steps sketched in

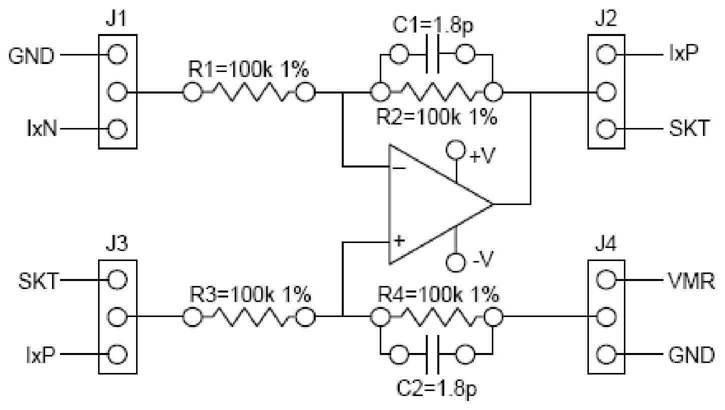


Fig. III. 9 – Circuit of analog interface block of the board.

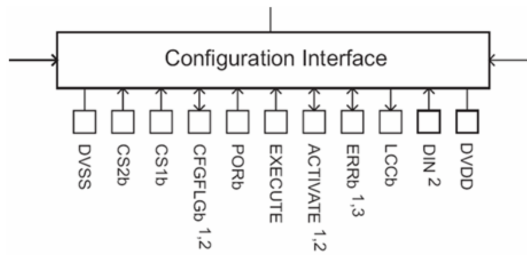


Fig. III. 10 – FPAA digital interface lines.

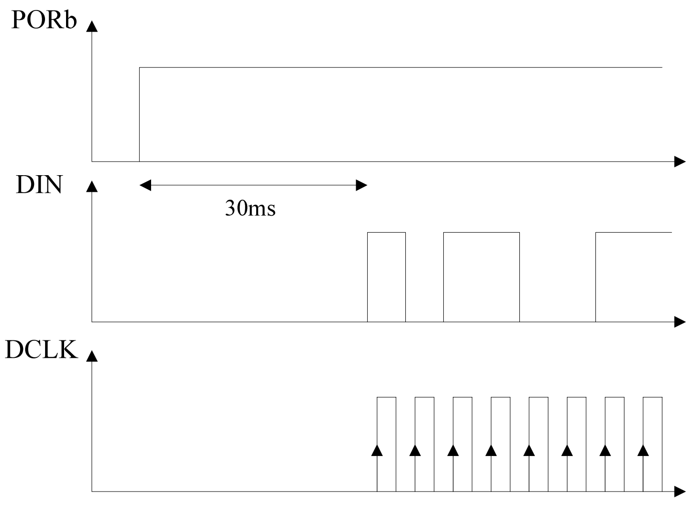


Fig. III. 11 - Byte transmission to FPAA.

Fig. III. 11: *(i)* when the device is reset, PORb line is set high, so, before any further operation can go on, 30 ms must be waited for; *(ii)* the microcontroller sets DIN line high or low according to the data bit to be transmitted; *(iii)* microcontroller pulses DCLK line in order to latch the data bit; steps *(ii)* and *(iii)* are iterated until the whole primary configuration data have been sent.

III.5 Integration of FPAA in the remote laboratory

In this section the experimental activity aiming at the integration of the FPAA in the remote laboratory is presented. The laboratory hardware architecture is modified as in Fig. III. 12. The FPAA is connected to a signal generator, from which it receives the input signal, and provides the output to an oscilloscope or a multimeter [12]. This electrical connection are fixed for every experiment, since the analog circuit is changed by configuring the FPAA. With this station, thus, students are able to perform a wide variety of experiment without waiting for local personnel wiring the circuit.

The successive step has concerned with the implementation of the software for the remote FPAA configuration, with the same Web Service technology employed for the remote laboratory development.

III.5.1 Collection of Primary Configurations

When an analog circuit has been designed, Anadigm Designer software allows saving the associated Primary Configuration in a text file, rather than sending it to the FPAA device. An application for configuring the FPAA, then, should read the contents of a text file and

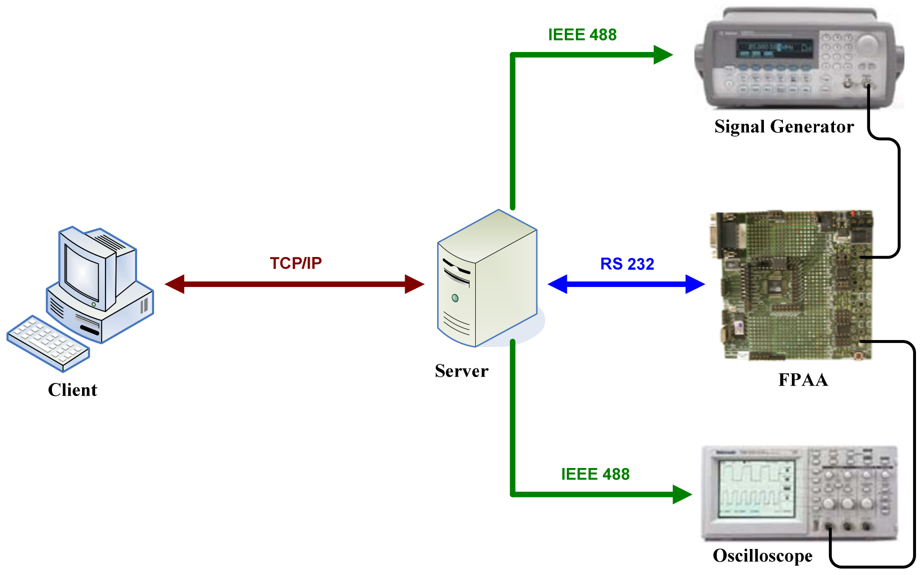


Fig. III. 12 – Architecture of the remote laboratory with FPAA.

transmit it on the serial port. On the other hand, in order to remotely configure the FPAA device, the client application should send to the measurement station server a string containing the Primary Configuration data. The server, then, should forward the string to the serial port to which the FPAA is connected. This implies two possible solutions: *(i)* clients need Anadigm software to design the test circuit and obtain the configuration file to successively send to the server; *(ii)* clients need to copy the configuration file on their computer.

Actually, to make the data exchange between client and server as fast as possible, and the client operating action easier another solution has been preferred. In Anadigm Designer environment, a defined number of test circuits has been collected; the files associated to each Primary Configuration have been saved on the server managing the measurement station. The client application allows user to select one of the available test circuits and send, through TCP/IP a message to the server only consisting of a string associated with the particular file

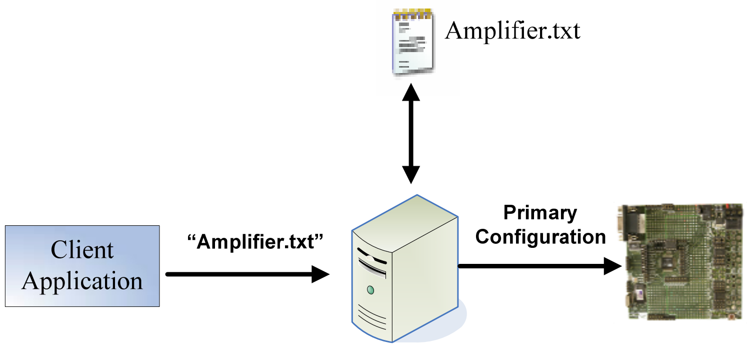


Fig. III. 13 – Remote selection of circuit under test.

name. The server, then, finds the file in memory associated with the received file name, reads its content, and transmit it to the FPAA. The procedure for the primary configuration is sketched in Fig. III. 13.

III.5.2 The Web Method FPAA Configuration

As described in the previous chapter, the communication between the client computer and the remote measurement station is based on a .NET Web Service publishing a set of web methods running on the server that exchange messages, through IEEE 488 protocol, with the signal generator, the oscilloscope and the multimeter. Another method, for transmitting the configuration string to FPAA by means of RS232 protocol has been included in the service. In particular, the method for configuring the FPAA has been developed and included in the web service; successively the pre-existent methods have been updated, in order to implement methods able to manage the combined communication with FPAA and measurement instruments.

In the following, the function for reading the content of a file and transmitting it on serial port is shown:

```
<WebMethod>
Public Function WriteSerial(ByVal port As String, ByVal filepath As String,
ByVal update As String, ByVal UpdateFlag As Boolean)
        Dim ConfigurationFile As New System.IO.StreamReader(filepath)
        Dim config As String = GoFile.ReadToEnd()
        ConfigurationFile.Close()
        SerialCommunication = ResourceManager.GetLocalManager.Open(port)
        SerialCommunication.BaudRate = 57600
        SerialCommunication.Parity = Parity.None
        SerialCommunication.DataBits = 8
        SerialCommunication.FlowControl = FlowControlTypes.None
        SerialCommunication.TerminationCharacterEnabled = False
        If upflag = False Then
           SerialCommunication.Write(\0x02 & config & \0x03)
        Else
           SerialCommunication.Write(\0x02 & config & \0x03 & \0x02 & update
& \0x03)
        End If
End Function
```

The function assigns to the variable *config* the content of the text file whose name has been received by the client application. Then the parameters (baud rate, flow control, etc) of the serial session are configured, and the string associated to *config* is transmitted to the serial port. As requested by the Anadigm protocol, the configuration string is completed with the characters Start of Transmission (STX) and End of Transmission (ETX); for the sake of clarity, in the provided example STX and ETX have been replaced by their hexadecimal ASCII expression.

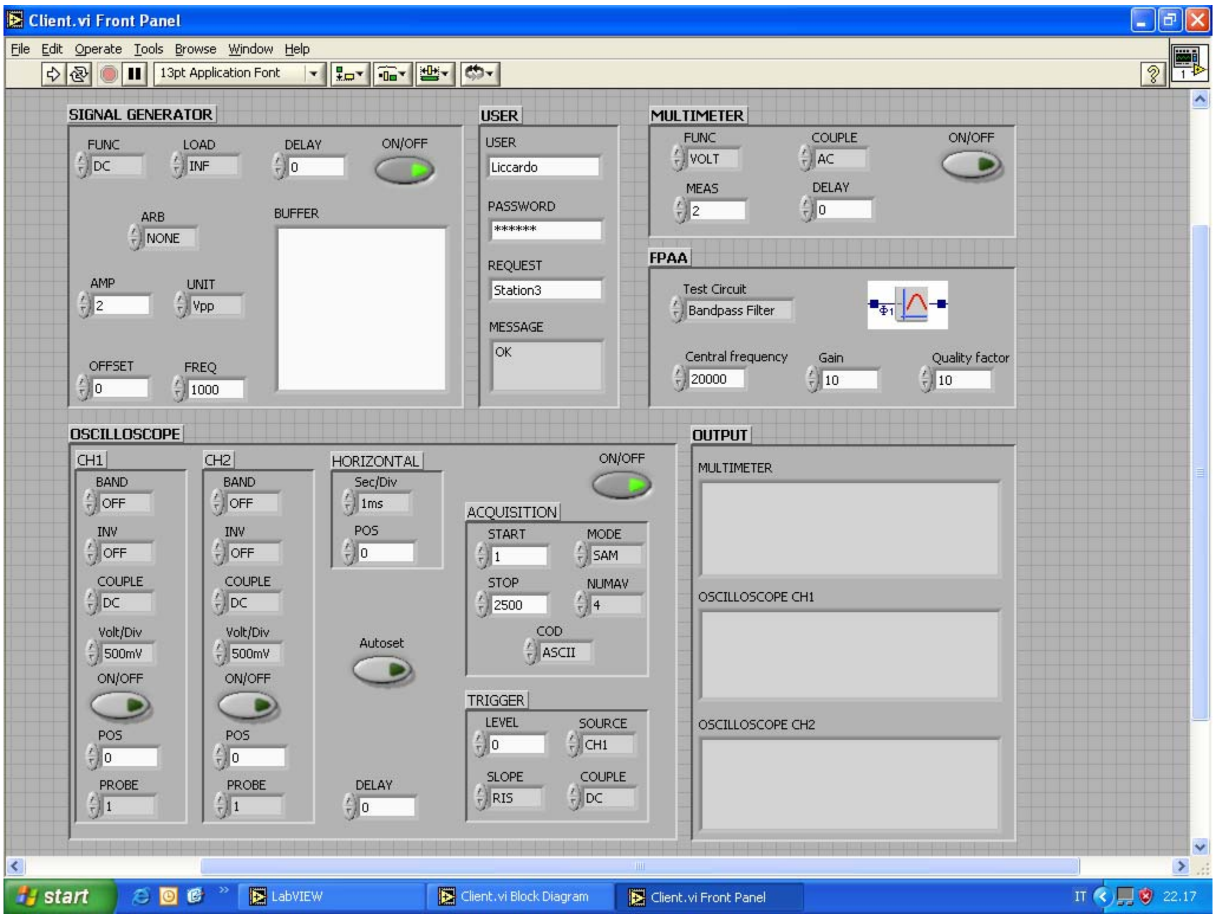
As shown in the example, another expected input of the function is the string *update*. The primary configuration, in fact, programs the FPAA to emulate an analog circuit with particular parameters; for example, if users select as test circuit an amplifier, FPAA will be configured to realize an amplifier with gain equal to 5. In order to make this parameters selectable by the client, the opportunity of dynamically reconfiguring the FPAA device has been exploited. If the user has modified the parameter of the selected analog circuit, an *update* string, containing the reconfiguration string is received by the method, together with a boolean variable *UpdateFlag* set true. In this instance the method transmits to FPAA the variable *config* concatenated with the content of *update* variable.

The service lifetime has been modified too for avoiding configuration conflict between concurrent access; the measurement station, in fact, has to be locked in order to allow a client executing the following operation: *(i)* selection of the test circuit, by programming the FPAA; *(ii)* configuration of the signal generator; *(iii)* configuration of the multimeter and/or the oscilloscope; and *(iv)* data acquisition from measurement instruments. So every web method described in Chapter II has been modified, by inserting into the Sync Lock structure also the utility configuring the FPAA. Moreover the input FPAA, containing information about the device configuration, has been added to the methods of the Measurement Station web service

III.5.3 Change in client application

The client application illustrated in Chapter II has been updated, since the SOAP string to be sent to the web service has to include the information about the FPAA. Fig. III 14 shows the front panel of the client application. In the section FPAA, user can select the analog test circuit from a defined list and its main parameters; for example, in Fig. III. 14, a bandpass filter has been selected; user can set the filter parameters as central frequency, gain and quality factor. There is another section, for configuring the signal generator, in order to control the input signal of the selected analog circuit. Then, the sections multimeter and oscilloscope allows to configure these instruments and perform measurement on the analog circuit output.

The SubVI shown in Fig. III. 15 extracts, according to the selected circuit, the path of the file on the server containing the primary configuration. Whereas in Fig. III. 16 it is depicted



SIGNAL GENERATOR
FUNC DC
LOAD INF
DELAY 0
ON/OFF
ARB NONE
BUFFER
AMP 2
UNIT Vpp
OFFSET 0
FREQ 1000

USER
USER Liccardo
PASSWORD ******
REQUEST Station3
MESSAGE OK

MULTIMETER
FUNC VOLT
COUPLE AC
ON/OFF
MEAS 2
DELAY 0

FPAA
Test Circuit Bandpass Filter
Central frequency 20000
Gain 10
Quality factor 10

OSCILLOSCOPE
CH1
BAND OFF
INV OFF
COUPLE DC
Volt/Div 500mV
ON/OFF
POS 0
PROBE 1

CH2
BAND OFF
INV OFF
COUPLE DC
Volt/Div 500mV
ON/OFF
POS 0
PROBE 1

HORIZONTAL
Sec/Div 1ms
POS 0
Autoset
DELAY 0

ON/OFF
ACQUISITION
START 1
MODE SAM
STOP 2500
NUMAV 4
COD ASCII

TRIGGER
LEVEL 0
SOURCE CH1
SLOPE RIS
COUPLE DC

OUTPUT
MULTIMETER
OSCILLOSCOPE CH1
OSCILLOSCOPE CH2

Fig. III. 14 – Front panel of the implemented client application.

the SubVI which constructs the reconfiguration string according to the circuit parameters set by the user. Finally Fig. III. 17 shows the VI serializing in XML language the primary configuration and the reconfiguration. The result string is, then, encapsulated in the SOAP request as explained in Chapter II.

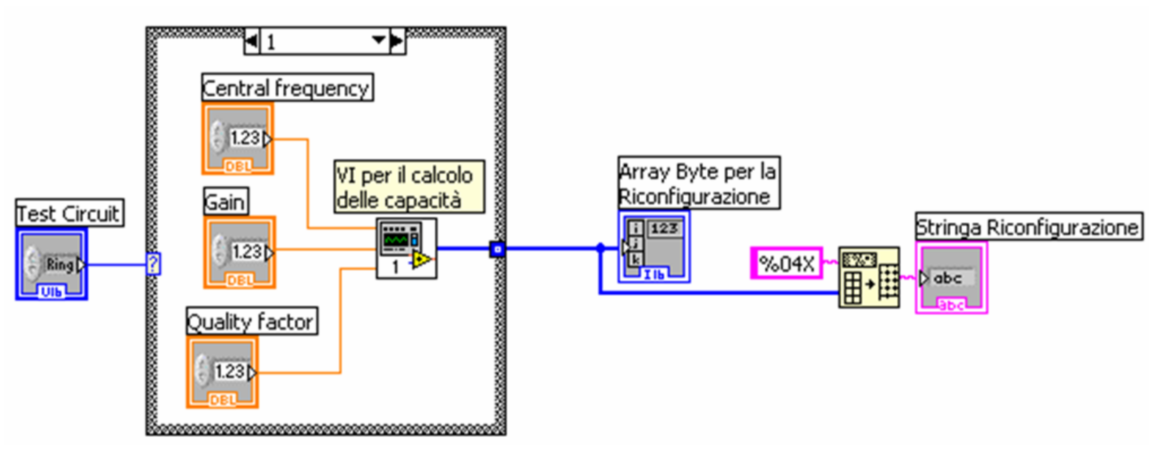


Fig. III. 16 – SubVI for the construction of the reconfiguration string.

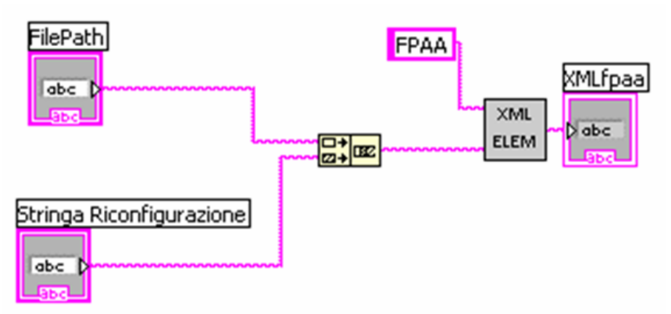


Fig. III. 17 – SubVI for serialize FPAA data.

III.6 Experimental assessment

A considerable part of the research activity has been focused on the experimental assessment of the realized laboratory. First the correct execution of the implemented software has been tested. Then the remote laboratory has been employed in order to characterize the

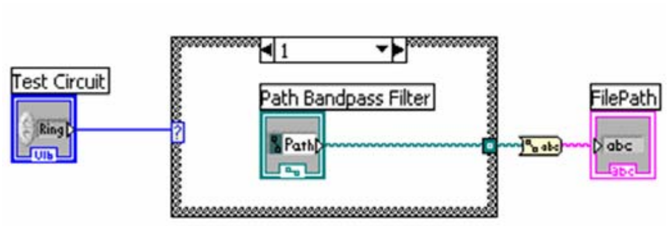


Fig. III. 15 – SubVI extracting the Primary Configuration.

performance of the analog circuits emulated by the FPAA. Obviously, limiting factors are present in any analog circuit, as a limited bandwidth and/or non-linearity. So, also FPAA technology is expected to exploit such unwanted characteristics [13]. Furthermore, capacitors in FPAA can assume only integer values, so the effect of quantization errors should be considered. As a consequence of the aforementioned factors, the FPAA output will be somewhat different from the expected one. Thus, before applying FPAAs in measurement applications an accurate metrological characterization of the implemented blocks in order to allow the estimation of the involved uncertainty has been considered necessary [14].

The test have been conducted employing one of the measurement station of the remote laboratory, composed by: *(i)* a development board including the FPAA under test AN221E04; *(ii)* a signal generator Agilent 33220A (14 bits resolution, 20MHz maximum generation frequency, 20V peak-to-peak maximum voltage); *(iii)* a digital oscilloscope Tektronix TDS 210 (8 bits resolution, 60 MHz bandwidth, 1GS/s maximum sample rate).

In the following the main results obtained from analog blocks widely used in electronic circuits are reported.

III.6.1 Remote measurement on an amplifier

The amplifier ideal transfer function is:

$$\frac{V_{out}(s)}{V_{in}(s)} = G \tag{4}$$

The SC circuit realizing this CAM is shown in Fig. III. 18. The capacitor values are

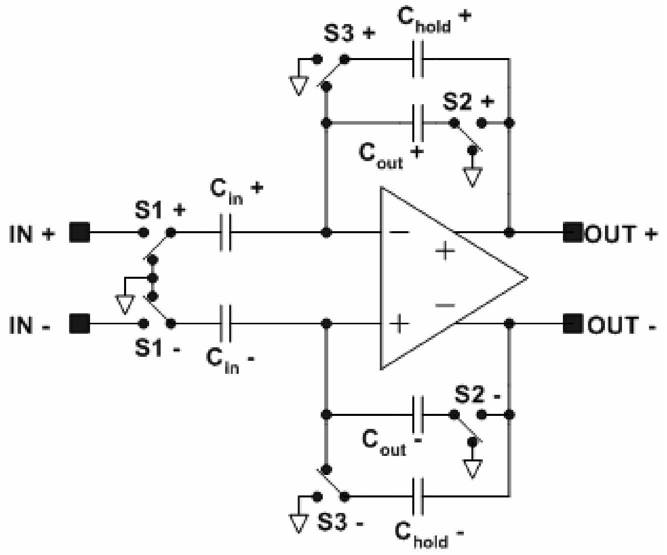


Fig. III. 18 – SC circuit emulating an amplifier.

chosen based on the best ratios of the capacitors satisfying the following relations:

$$G = \frac{C_{in}}{C_{out}} \tag{5}$$

Obviously errors due to finite bandwidth, finite input impedance and finite gain of op amps and quantization cannot be neglected. With the aim of investigating the actual characteristics in comparison with theory, several parameters have been investigated.

The amplifier actual transfer function has been obtained by selecting a DC input within the range -5V - +5V, with a 50mV step. Results, for a gain equal to 1 are reported in Fig. III. 19. The saturation limits of the device (declared as ±2V) seem to be asymmetrical, probably due to the supply circuit of the evaluation board, which does not provide a perfectly symmetrical voltage.

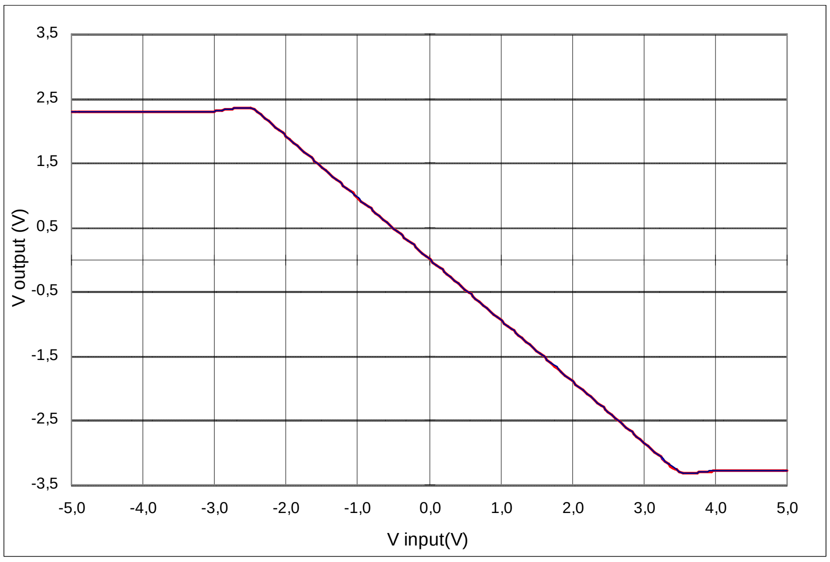


Fig. III. 19 – Actual transfer function of the amplifier.

The slope of the transfer function provides the actual gain, estimated as 0.95. As for the other gain values, significant differences between the nominal set gain and the measured one has been experienced; in particular, the highest the desired gain, the worst the obtained difference. The discrepancies seem not to depend by the capacitances accuracy but they are mainly due to the saturation of the output amplifier. In fact, the same discrepancies have been observed by fixing the gain and by varying the input amplitude so to observe the same output voltage.

Fig. III. 20 shows the amplifier Integral Non Linearity (INL) in the input range -2V – +2V; it has been evaluated as the differences between the transfer function and its linear interpolation.

INL is limited within -2mV and 3mV, so the transfer function can be considered linear with

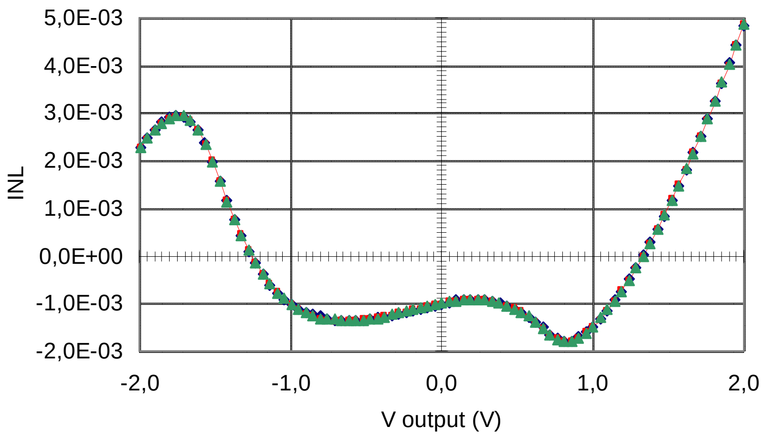


Fig. III. 20 – Amplifier INL.

an error equal to 0.25%.

Experimental results obtained about the amplifier frequency response are reported in Fig. III. 21.

The input was a sine wave with logarithmic frequency sweep in the range from 100 Hz to 15MHz. Actual gain and phase delay of the output are plotted versus frequency for different values of the nominal gain.

The experience puts in evidence that the frequency response is quite flat inside the 100 kHz bandwidth declared by Anadigm. Furthermore, the flatness remains unaffected by the value of the settled gain.

The amplifier phase delay is practically negligible up to 20 kHz, independently from the settled gain and then increases rapidly with frequency and gain.

Amplifier step response, due to a 20mV input step, is presented in Fig. III. 22. Though, in theory, this should be a zero-order system, the device output evidences a slight overshoot in low gain configurations.

In Tab. 1, the most significant parameters measured at the output of the device supplied with the 20 mV step are reported versus gain variations. In particular, response time and settling time arise along with the gain, whereas the slew rate decreases.

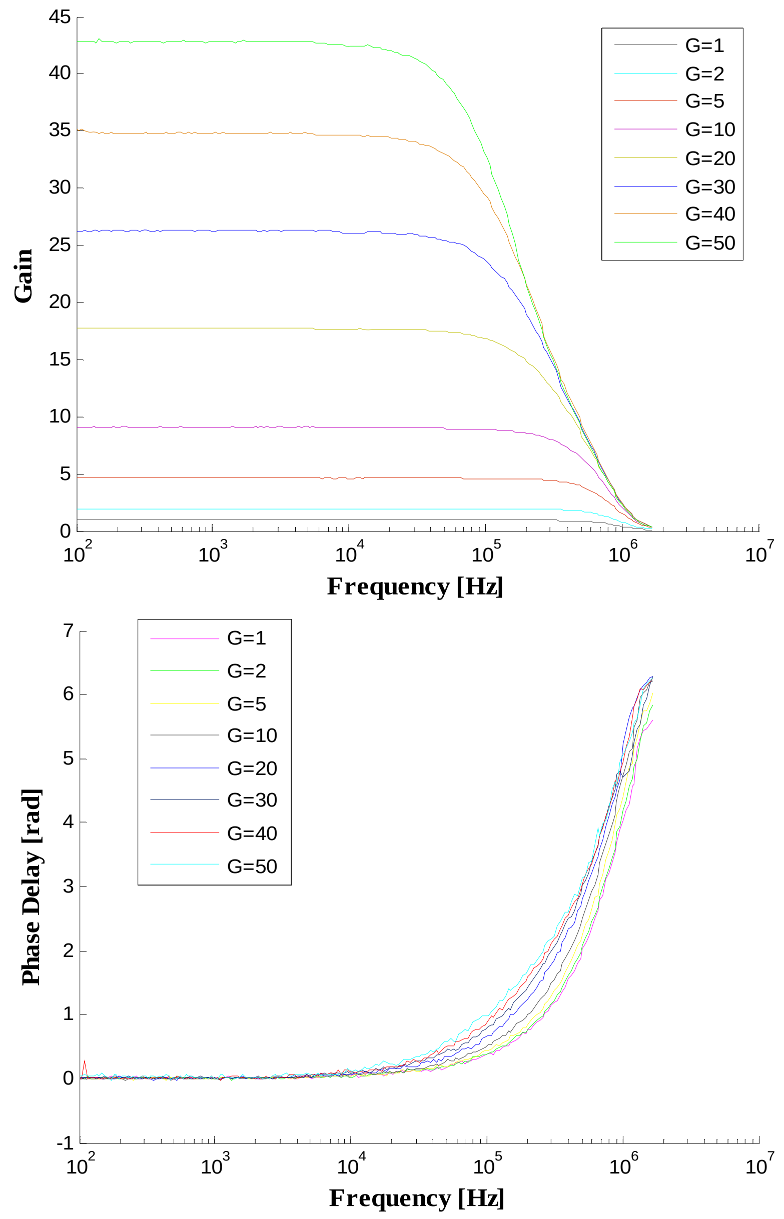


Fig. III. 21 - Amplifier frequency response.

Tab. III. 1 - Step response parameter of the amplifier.

Gain	Slew Rate (V/µs)	Response Time (µs)	Settling Time (µs)	Overshoot
1	3.3109	0.429	0.832	0.0579
2	3.2148	0.437	0.833	0.0623
5	2.9420	0.474	0.905	0.0431
10	2.3738	0.572	0.706	0.0037
20	1.4241	0.816	1.540	0.0030
30	0.9965	1.088	1.962	0.0015
40	0.7231	1.432	2.907	0.0006
50	0.6337	1.701	3.458	0.0001

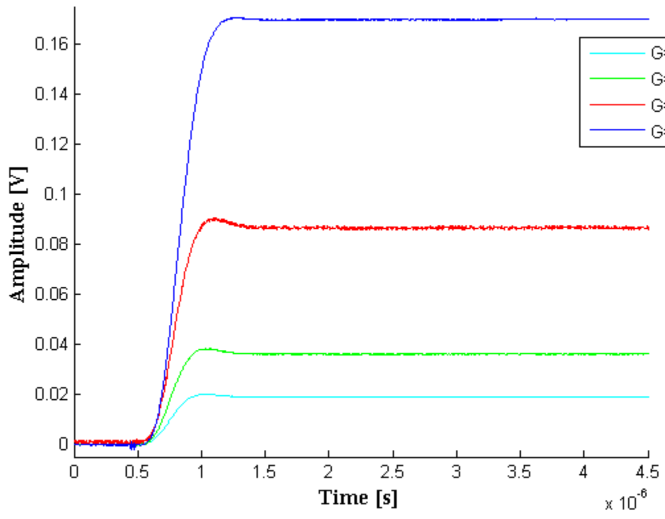


Fig. III. 22 - Amplifier step response.

III.6.2 Remote measurement on a bandpass filter

A similar analysis has been performed on analog filters. As an example, the results obtained in experimental tests conducted on biquadratic bandpass filter are given in the following. Fig. III. 23 shows the switched capacitor (SC) circuit implemented in the device under test to realize a second order bandpass filter.

The ideal transfer function is:

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{-2\pi f_0 \frac{G}{Q} \cdot s}{s^2 + \frac{2\pi f_0}{Q} \cdot s + 4\pi^2 f_0^2} \tag{6}$$

Capacitors size is chosen according to the following equations:

$$f_0 \cong \frac{f_c}{2\pi}\sqrt{\frac{C_2 C_3}{C_A C_B}} \tag{7}$$

$$G \cong \frac{C_p}{C_4} \tag{8}$$

$$Q \cong \frac{1}{C_4}\sqrt{\frac{C_2 C_A C_B}{C_3}} \tag{9}$$

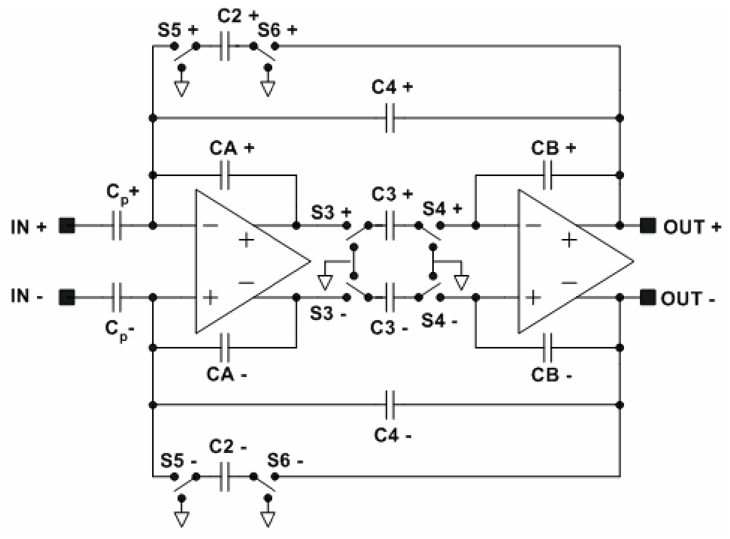


Fig. III. 23 - SC circuit operating as a bandpass filter.

As aforementioned, the user selection of the central frequency, gain and quality factor of the filter are achieved by properly varying the capacitance values as stated form equations (7), (8), and (9).

Filter performance in frequency domain has been assessed by evaluating its frequency response in the presence of three different values of gain (1, 5, and 10), quality factor (5, 10, and 15) and central frequency (1kHz, 50kHz, and 400kHz). For each experimental configuration, actual gain, upper and lower passband frequencies, upper and lower stopband frequency and quality factor have been evaluated. The obtained results have been compared with the characteristics obtained by means of two models developed in Matlab® software through the ideal transfer function given in (6) with *(i)* the desired parameters and *(ii)* the "quantized" parameters approximating that desired, but obtained from equations (7), (8) and (9). By comparing results of the two models implemented according to the ideal transfer function, the quantization error has been estimated as

$$\Delta G_q = G_{quantized} - G_{theoretical} \tag{10}$$

Where $G_{theoretical}$ is the gain simulated by Matlab transfer function with desired parameters, whereas $G_{quantized}$ is simulated by the transfer function with quantized parameters.

By comparing the characteristics obtained by the transfer unction with the quantized parameters with that measured on the actual circuit the actual error has been evaluated as:

$$\Delta G = G_{actual} - G_{quantized} \tag{11}$$

Where G_{actual} is the actual gain measured at output of the device under test.

As an example, Fig. III. 24 plots the filter output along with the Matlab theoretical and quantized responses corresponding to *(i)* central frequency of 1 kHz, *(ii)* unity gain and *(iii)* quality factors of 5, 10, 15.

At the bottom of the same figure, the errors expressed by equations (10) and (11) are reported.

As shown in figure, both the errors ΔG_q and ΔG are negligible within the considered frequency range, except for the frequencies close to the desired filter central frequency where

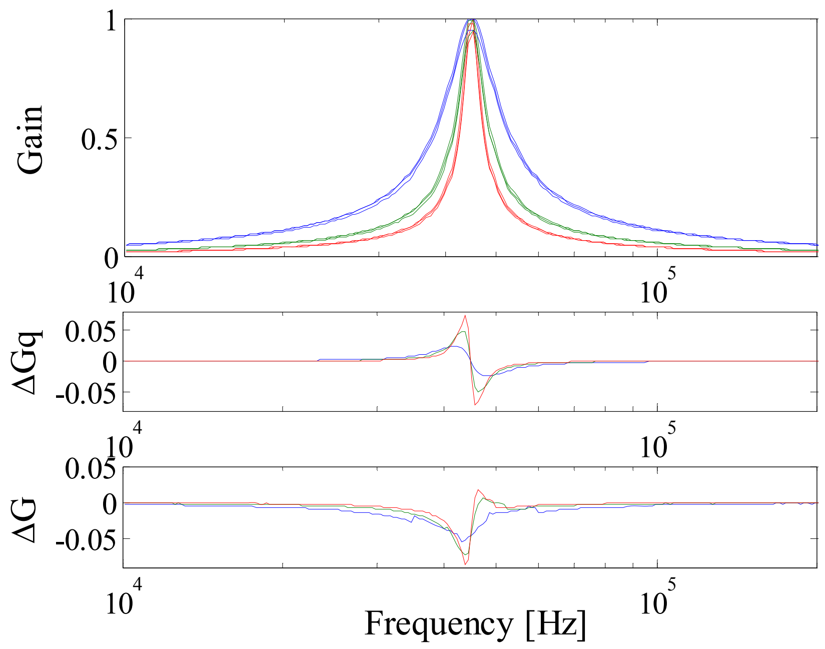


Fig. III. 24 - Comparison between filter responses with ideal parameters, quantized parameters, actual parameters.

the worst error is experienced. This behaviour of the gain error reveals that the actual filter is centred on a frequency slightly different from that selected and higher the selected quality factor, greater the errors.

In order to reach a better understanding of the phenomena, the filter central frequency has been changed up to 400 kHz. It has been observed that, remaining inside the 100 kHz bandwidth, the filter response shows a shape similar to Fig. III. 22, with the same worst error at the central frequency. By measuring the actual central frequency, the experiments confirmed the aforementioned hypothesis; in fact, an error affecting the filter resonance frequency contained in 1% of the expected value has been observed. Tab. III. 2 shows, as an example, the differences between the desired central frequency (F_{cs}) and the measured frequency associated to the maximum gain, for different Q values.

Tab. III. 2 – Differences between nominal Fc and maximum gain frequency

Q	$F_{MAX\ GAIN}$	$F_{MAX\ GAIN} - F_{cs}$
0,707	39,9 kHz	- 0,100 kHz
1,42	39,8 kHz	- 0,200 kHz
5,65	39,8 kHz	- 0,200 kHz
14,1	39,82 kHz	- 0,180 kHz
70,7	39,760 kHz	- 0,240 kHz
99,7	39,712 kHz	- 0,288 kHz

Filter behaviour in time domain has also been investigated by means of its step response. In Fig. III. 25 the step response of the simulated model and the step response obtained by experimental tests are matched. Clearly, apart from an offset experienced in the actual response, the filter real step response agrees very well with the ideal response. The offset could represent a problem when the bandpass filter belong to a conditioning chain including other blocks operating on signal amplitude, such as amplifiers. This drawback can, in fact, result in a saturated output of the FPAA with the loss of information of the input signal.

Other time domain parameters have been measured and are reported in Tab. III. 3.

As expected, step response behaves as a second order narrow bandwidth system. Time constant and, consequently, settling time are, in fact, rather high. Moreover, ringing frequency has been found equal to the natural frequency of the filter, i.e. its center frequency.

Tab. III. 3 - Step response parameter of the filter.

Gain	Damping Time Constant (μs)	Ringing Frequency (Hz)
5	36	50000
QF	Settling Time (μs)	Overshoot (V)
5	139	0.85

III.6.3 Remote measurement on an integrator

Finally, the analysis has been conducted on an integrator, whose SC circuit is shown in

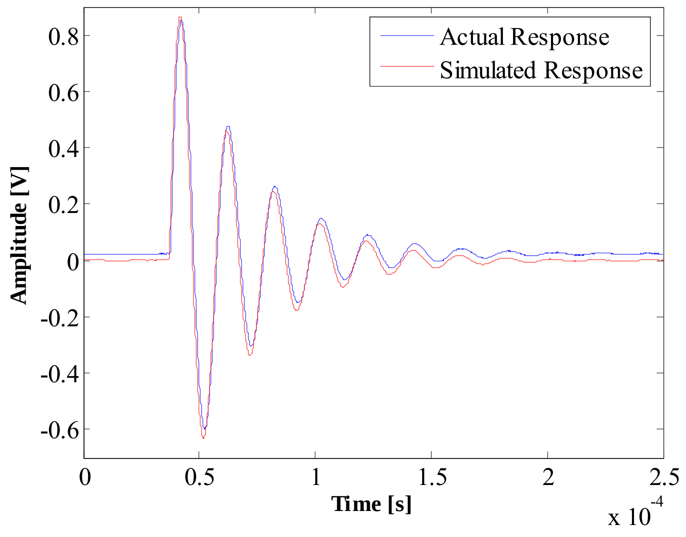


Fig. III. 25 - Filter step response.

Fig. III. 26. The ideal transfer function is:

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{K}{s} \tag{III.12}$$

In Fig. III. 27, as an example, the input and output of an integrator with integration constant equal to 0.1μs^{-1} are reported.

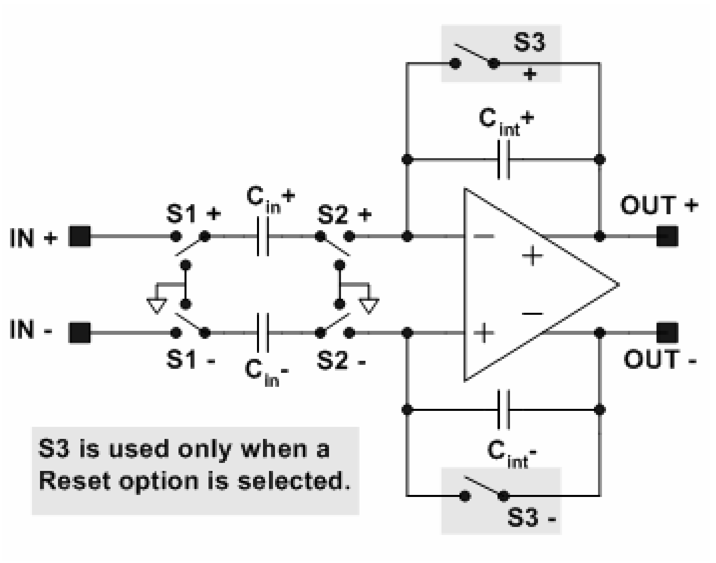


Fig. III. 26 – SC circuit of an integrator.

With a sine wave input, the output presents a phase delay of 90° as expected. Actually the output shows a phase displacement if compared with the ideal cosine waveform, because of the time delay due to signal propagation in the FPAA device. This phase displacement has been experienced lower than the 2% of the input signal period.

The actual integration constant has been evaluated as the ratio between the input/output negative peak values.

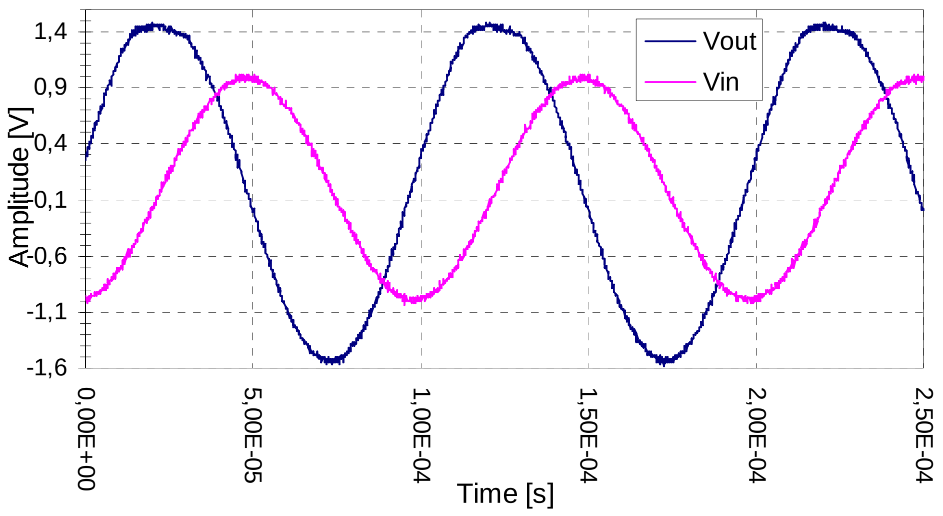


Fig. III. 27 – Voltage input and output of the integrator CAM.

$$\frac{V_{outP-}}{V_{inP-}} = \frac{K_i}{2\pi f_S} = K_f \qquad (13)$$

The measurement have been performed by fixing Kf and varying the input frequency. The differences between the expected Kfi and the actual Kfr reported in Fig. III. 28.

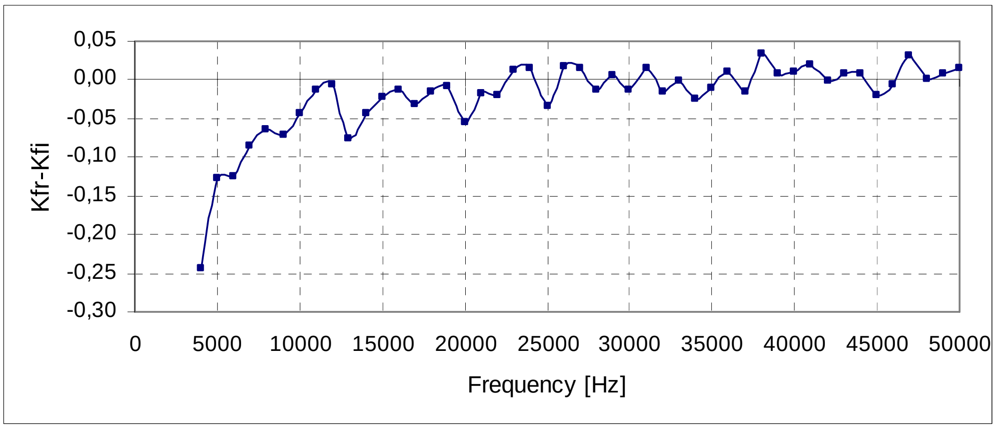


Fig. III. 28 – Differences between expected and actual integration constant.

References

[1] R. Naiknaware, T. S. Fiez, "Process-insensitive low-power design of switched-capacitor integrators", IEEE Transactions on Circuits and Systems, Vol. 51, Issue 10, pp. 1940 – 1952, Oct. 2004.

[2] Anadigm Company, "Anadigm AN221E04 Datasheet", 2003, www.anadigm.com.

[3] P. I. Yakimov, E. D. Manolov, M. H. Hristov, "Design and implementation of a V-f converter using FPAA", 27th International Spring Seminar on Electronics Technology: Meeting the Challenges of Electronics Technology Progress, Vol. 1, pp. 126-129, 2004.

[4] M. B. Halima, M. Fakhfakh, M. Loulou, "A switched current based FPAA macrocell for mixed mode signal processing systems", IEEE Workshop on Signal Processing System Design and Implementation, pp. 143-147, 2005.

[5] C. A. Looby, C. Lyden, "Op-amp based CMOS field-programmable analogue array", IEEE Transaction on Circuits, Devices and Systems, Vol. 147, Issue 2, April 2000, pp. 93 - 99.

[6] T. S. Hall, C. M. Twigg, P. Hasler, D. V. Anderson, “Application performance of elements in a floating-gate FPAA”, Proc. of International Symposium on Circuits and Systems, 2004, Vol. 2, pp. 589-592.

[7] J. Silva-Martinez, E. Sánchez-Sinencio, “Switched Capacitor Filters”, In Handbook of Circuits and Filters, CRC Press, 2003.

[8] P. Hasler, “Low-power programmable signal processing”, Proc. of . Fifth International Workshop on System-on-Chip for Real-Time Applications, 2005. pp. 413 - 418.

[9] G. Merendino, S. Callegari, A. Golfarelli, M. Zagnoni, M. Tartagni, “Signal conditioning for capacitive sensors with field programmable analog arrays”, Proc. of International Symposium on Signals, Circuits and Systems, 2005, Vol. 1, pp. 139 - 142.

[10] Anadigm Company, “Anadigm Designer IP Module Manual”, 2003, www.anadigm.com.

[11] A. Baccigalupi, A. Liccardo, “Condizionamento dei segnali ecografici mediante Field Programmable Analog Arrays”, Relazione ad invito in *Atti del Congresso Gruppo Misure Elettriche ed Elettroniche*, pp. 337-346, L’Aquila, Italia, 11-13 Settembre 2006.

[12] A. Baccigalupi, A Liccardo, “Programmazione via Internet di dispositivi a capacità commutate”, *Atti del Convegno Nazionale AEIT*, Capri, Italia,16-20 Settembre 2006.

[13] A. Laknaur, H. Wang, “A methodology to perform online self-testing for field-programmable analog array circuits”, IEEE Transactions on Instrumentation and Measurement, Vol. 54, Issue 5, Oct. 2005, pp. 1751 - 1760.

[14] A. Andrade, G. Vieira, T. R. Balen, M. Lubaszewski, F. Azaïs, “Built-in self-test of global interconnects of field programmable analog arrays”, Microelectronics Journal, Vol. 36, Issue 12, December, 2005, pp. 1112-1123.

Chapter IV - Remotely Programmable Smart Sensors

IV.1 Introduction

The recent research work has been focused on the employment of FPAA for including in the laboratory smart sensors remotely programmable. Despite a large part of electronic circuitry now consists of digital logic, in several systems, analog parts have still critical importance. As an example, in most cases, they are used to condition analog signals coming from sensors and actuators so that they can successfully be converted in digital form by analog-to-digital converters. Moreover, in some applications, even though powerful digital signal processors could perform the desired functions, it turns to be more convenient to exploit the same functionality, at a fraction of the application cost, with analog solutions. In conclusion, it is always a good design strategy to tailor the final application as the result of a correct balance between analog and digital load.

The most important feature of FPAAs is the provided opportunity of dynamically reconfiguring the analog blocks, in order to properly tailor the characteristics of conditioning section to the signal to be acquired [1]. This is a particularly attractive characteristic in sensor applications, where the possibility of tuning some parameter, allows an improvement of the sensor metrological characteristics. Moreover, in contrast with architectures built with dissipative components (i.e. resistors), the capacitor-based technology used in FPAA turns into high accuracy, very low power dissipation and, consequently, low dimensions.

In particular, FPAA has been employed to realize the conditioning section of a distance sensor based on the estimation of the Time Of Flight (TOF) of ultrasonic signals. The measurement principle is shown in Fig. IV. 1:the transducer emits an ultrasonic signal that reaches a target; when the reflected echo is received, by knowing the velocity of sound and the time elapsed between the signal generation and the echo onset it is possible to estimate the target distance from the transducer [2].

The effectiveness of distance evaluation based on ultrasonic Time of Flight (TOF) measurements is mainly characterized by the accuracy gained to detect the exact arrival time of the returned echo. Usually, the time of echo onset is recognized when the received signal crosses a fixed threshold. As expected, the main factor degrading the measurement accuracy in estimating onset is the noise level superimposed to the echo signal. Actually, in open-space measurements, Signal-to-Noise-Ratio (SNR) down to few decibels is experimented for a target distance of 1÷2 meters. Another degrading factor of TOF measurement uncertainty is the echo shape distortion which, in turns, depends on the target distance [3]. The echo profile is indeed distorted by a number of factors such as geometrical and mechanical properties of the reflecting surface, position and orientation of the target, transmission paths, environmental inference factors (temperature, pressure, humidity), and so on [4].

Many solutions are suggested in literature for improving echo SNR and, consequently, the related measurement uncertainty [5-9]. Among them, good effectiveness is generally presented by those lying on analog signal pre-processing (amplification, filtering, etc.). They, actually, can increase SNR so reducing the aforementioned difficulty to detect the exact time of echo onset. These techniques are further improved by adopting analog blocks with time-varying characteristics, or by using a proper digital signal processing algorithm.

For instance, an analog block with time-varying characteristics can be achieved by a

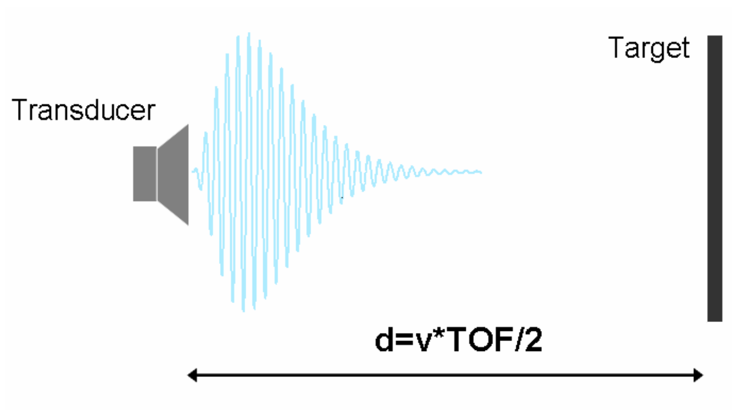


Fig. VI. 1 – Ultrasonics-based distance measurement.

programmable amplifier whose gain is changed versus the target distance. Programmable amplifiers, however, cannot compensate echo shape distortion and, as a consequence, threshold methods can suffer from the dependence of rising edge slope on the target distance. Conversely, techniques based on real time digital signal processing of the returned echo, surely characterized by a greater flexibility, require a heavy computational burden as well as the use of performing processors.

In the following a distance sensor exploiting a conditioning block based on FPAA device is presented. It is featured by: *(i)* normalized echo envelope amplitude without requiring any further processing; *(ii)* echo shape independence from target distance and *(iii)* a satisfactory signal-to-noise ratio at any distance inside the measurement range.

Remote users are able to acquire the echo envelope and to reconfigure the FPAA in order to obtain the optimal echo shape.

IV.2 FPAA as Programmable Conditioning Section

Apart from the Primary configuration, which allows to program the FPAA to obtained a defined analog circuit, FPAAs offer the opportunity of dynamically changing the characteristics of the implemented analog circuit. Once the application circuit topology has been fixed, capacitance values can be modified by the user altering their values in the RAM array so that the capacitance values inside FPAAs can be varied *on-the-fly* in order to dynamically change the characteristics of the implemented analog blocks CAMs, without resetting the device and transmitting a lower number of bytes [10].

As a consequence, analog blocks can be tuned according to the operating and environmental conditions, so that optimal signals can be anytime furnished to the next processing blocks.

In the proposed application, an FPAA has been used to realize an adaptive conditioning system of an ultrasonic sensor, whose main building blocks are: *(i)* a bandpass filter, whose 45 kHz central frequency is tuned to the ultrasonic sensor natural frequency with the aim of enhancing echo SNR; *(ii)* a full wave rectifier followed by a peak detector in order to extract the echo envelope; *(iii)* a low pass filter with 1kHz corner frequency used to smooth the output signal; *(iv)* an output amplifier employed to increase the level of the conditioned signal. The conditioning block, as realized in Anadigm environment, is shown if Fig. IV. 2 [11].

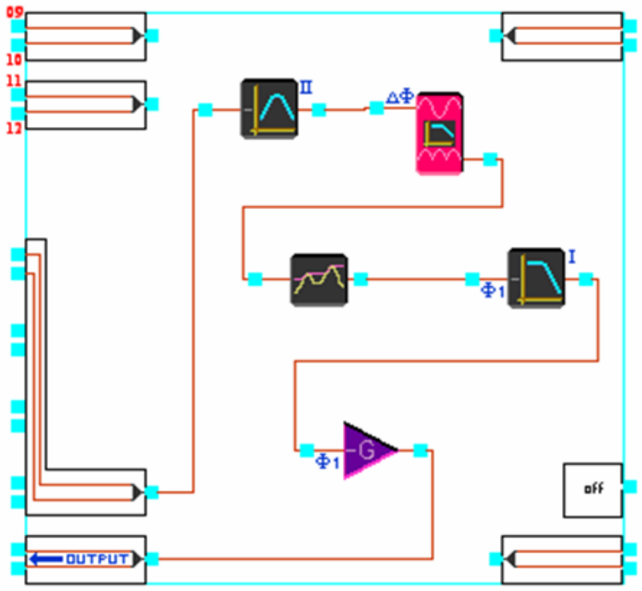


Fig. VI. 2 – Conditioning circuit in Anadigm Designer

Main output waveforms are plotted in Fig. IV. 3.

The primary configuration cannot be changed, whereas users are able to operate on the building blocks parameters, in order to adapt the gain and the quality factor of the bandpass input filter to the target distance.

The circuit topology along with the default set of parameters of each configurable module

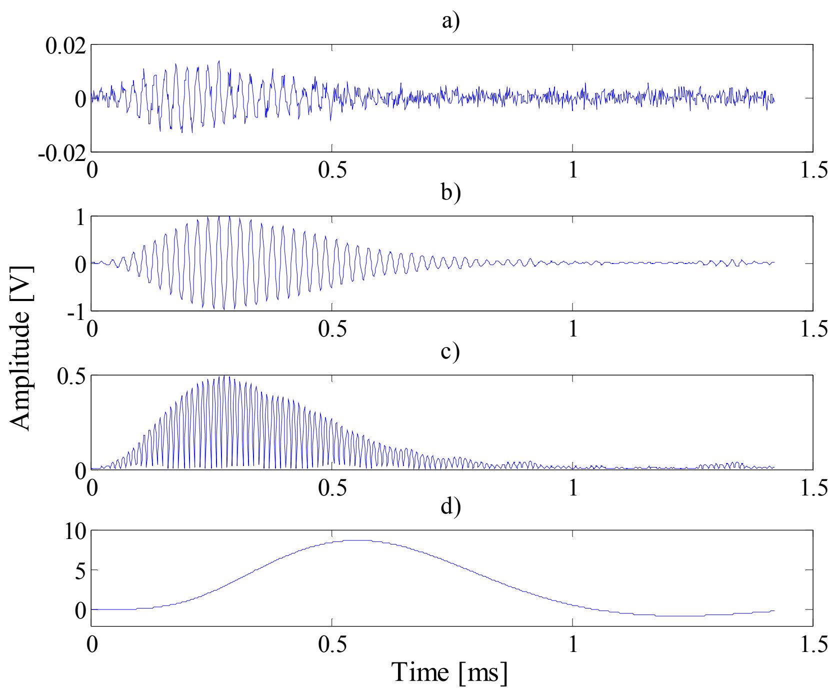


Fig. IV. 3 - a) Echo at FPAA input; b) Bandpass filter output; c) Rectifier output; d) Lowpass filter output.

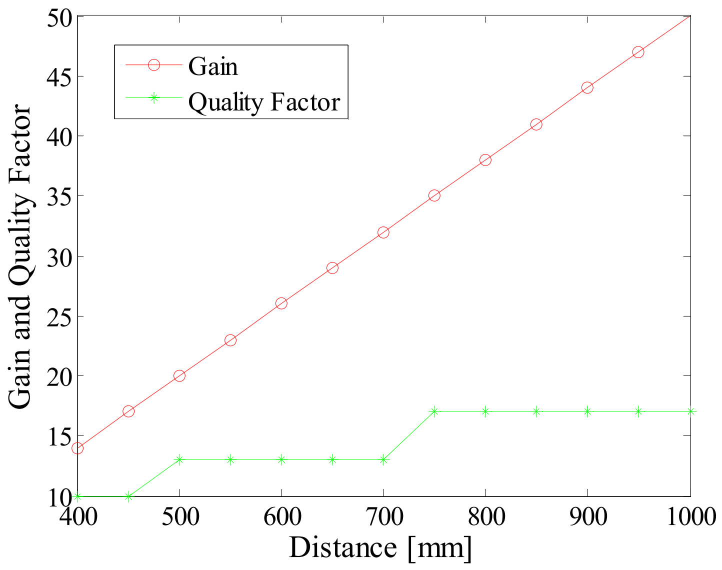


Fig. IV. 4 - Parameters of bandpass filter versus target distance.

(the Primary Configuration) is sent to the FPAA from a proper web method each time that the method is invoked. It accepts as input also the string containing the FPAA reconfiguration sent by user application. Clients access the measurement station in order to build the configuration associated to the optimal echo envelope reducing the uncertainty of onset location. Students goal is the implementation of an algorithm able to find the optimal circuit parameters. As an example, in Fig. IV. 4, the optimal parameters, experimentally found ranging from 400 mm up to 1000 mm, are drawn versus the target distance. The best choice of the bandpass filter gain proved to be linearly dependent on the target distance, whereas the strategy of varying the filter quality factor has based on the technique to constrain the non linearity measurement error inside predefined thresholds.

IV.3 Remote Configuration of the Sensor

The developed measurement station is shown in Fig. IV. 5. The signal generator provides to the piezoelectric transducer the stimulus signal, properly amplified by the front end. When the echo is received, it is conditioned by the FPAA; the resulting envelope is acquired by the oscilloscope. The server is moreover able to change the parameter of the conditioning circuit via RS 232.

The developed web method has to perform the following operation: *(i)* to send the reconfiguration string received by client program to the FPAA; *(ii)* to configure the signal generator for obtaining the pulse train according to the transducer central frequency; *(iii)* to configure the oscilloscope for acquiring the echo envelope; *(iv)* to receive samples data from

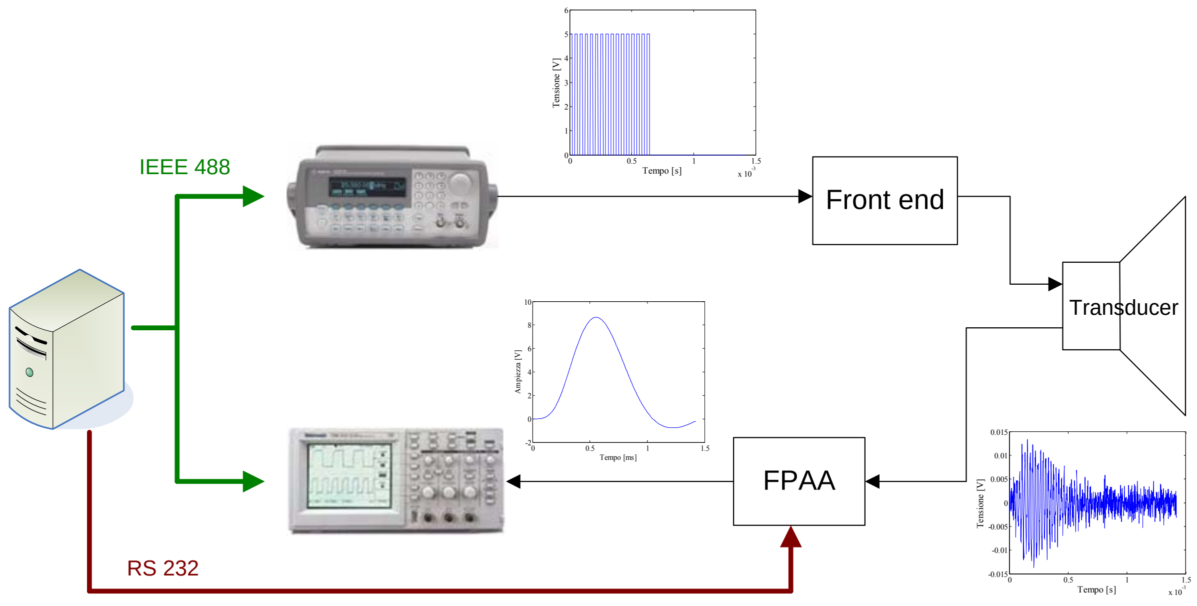


Fig. IV. 5 – Scheme of the remote sensor.

the oscilloscope and send them to the client program. The method has been derived by the *Generator, Oscilloscope and FPAA* method of the *Measurement Station* Web Service; but the only variable configurable by users is the FPAA reconfiguration string.

The client application is developed by students, who have to implement in LabVIEW environment the entire measurement and optimization algorithm. Students receive, with the software environment, a subVIs library for the serialization and transmission of data according to SOAP protocol. The block diagram of these SubVIs is hidden, so that students use them as the GPIB write and read functions.

As an example, in Fig. IV. 6 a LabVIEW^{TM} application realized by students of last semester is shown. The research of the optimal parameter is performed by comparing the echo amplitude with a tolerance range between 1.7 and 1.9 V. If the amplitude is lower than 1.7 V, than the filter gain is increased; otherwise, if echo amplitude is greater than 1.9 V the gain is lowered. In both cases, however, the gain increment is divided by two. The SubVI filter provides, then, the filter quality factor related to the selected gain. The SubVI communication, instead, builds the reconfiguration string and invoke the method to transmit it to FPAA and repeat the echo acquisition.

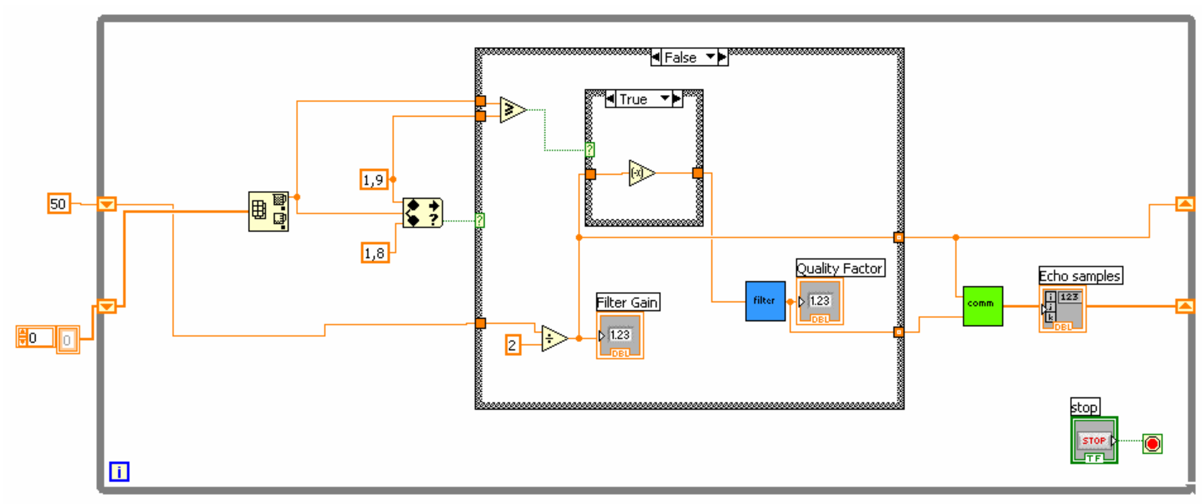


Fig. IV. 6 – Example of LabVIEW program for remotely configure the sensor.

IV.4 Prototype of Smart Distance Sensor

On the basis of the sensor assessment performed by students and the conceived algorithms, a prototype of smart distance sensor has been developed.

The FPAA used in this work, an AN221E04 by Anadigm®, is usually configured by means of a proper software environment, the Anadigm Designer 2, whose graphical interface allows to quickly and easily construct analog circuits by selecting, placing and wiring together selected CAMs. The main drawback of using Anadigm Designer 2 software is that it does not allow dynamic reconfiguration [12].

Anyway, the Anadigm® software provides a C code function library that enables the reconfiguration of FPAA through the use of a host processor. At this aim, authors designed the architecture shown in Fig. IV. 7. The main element is the Digital Signal Controller (DSC), *Microchip® dsPIC30F3013* (120 MHz maximum clock frequency, 30 MIPs, 24 KB on-chip Flash program space, 2 KB on-chip data RAM, 1 KB non-volatile data EEPROM, 12-bit analog-to-digital converter, 3 independent Timers, each of which can be used as interrupt source).

The ultrasonic transducer is an industrial piezoelectric sensor, whose natural frequency is 45 kHz. In order to drive the sensing element and, in particular, to adapt the stimulus signal level to the piezoelectric sensor, a proper front-end has been realized. At the measurement beginning, eight rectangular pulses, generated through a DSC I/O pin, are used to switch on and off the power transistor base. An high-frequency transformer, HFT, *6500 ranging transformer*, (5.0 ± 0.1 V_{DC} pulsed primary excitation, 580 ± 30 V peak-to-peak output

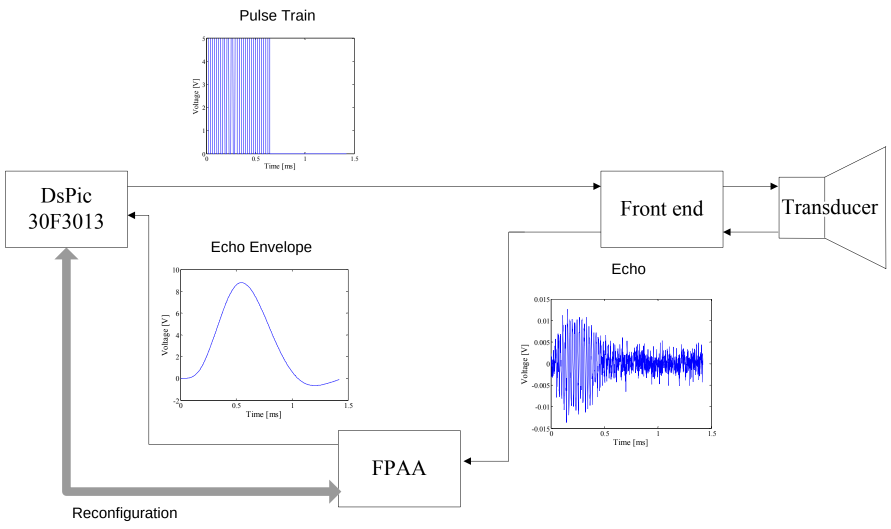


Fig. IV. 7 - Block diagram of realized prototype.

voltage amplitude) amplifies the voltage signal up to the value needed to drive the piezoelectric transducer, *Polaroid 9000 Series*, (17°x35° beam angle, (45±2)kHz operating frequency, 108 dB transmitting sensitivity and -78 dB receiving sensitivity). In particular, the electrical stimulus is characterized by frequency value and peak-to-peak voltage amplitude equal respectively to 45 kHz and 40 V.

In order to avoid FPAA saturation, signal is clipped inside ±1V and, furthermore, it is cleaned form any DC component by a coupling capacitor. Echo then passes through the FPAA where it is treated by the analog blocks configured as previously described.

DSC configures FPAA, implementing a synchronous serial protocol, through its I/O lines which control: *(i)* Power On Reset (PORb) signal, whose rising edge carries the FPAA in its reset state (operation which takes some 30ms); *(ii)* Data In (DIN) signal, used by FPAA to read data transmitted by DSC; *(iii)* Digital Clock (DCLK) signal, whose rising edges latch valid data on DIN line.

FPAA primary configuration is performed through the operating steps sketched in Fig. IV. 8: *(i)* when the device is reset, PORb line is set high, so, before any further operation can go on, 30 ms must be waited for; *(ii)* DSC set DIN line (high or low according to the data bit to be transmitted); *(iii)* DCS pulses DCLK line for 3μs in order to latch the data bit; steps *(ii)* and *(iii)* are iterated until the whole primary configuration data have been sent.

On-the-fly reconfiguration, as aforementioned, allows to change the parameters of implemented analog blocks, without modifying circuit topology. The Anadigm Designer software provides the analytical relationship between CAMs characteristics and involved capacitances, i.e., the equations needed to compute the capacitance values in order to obtain the desired circuit behaviour. Then, data blocks are organized according to FPAA AN221E04 reconfiguration protocol. Each block starts with a synch byte used for synchronizing the communication. Successive two bytes are FPAA's JTAG Identification Bytes. Then five bytes are sent for each capacitance value to be configured; their meaning is: *(i)* capacitor memory address; *(ii)* capacitor memory bank; *(iii)* byte count, i.e. how many bytes contain the information of the capacitance value; *(iv)* capacitance value (in the range 1÷255); *(v)* an error check byte [13].

Obviously, compared with primary configuration, the on the fly reconfiguration is faster to be transmitted, since a lower amount of data is required. Moreover, the reconfiguration does not need resetting PORb line in order to start the configuration protocol, but only DIN and DCLK lines have to be controlled as described above [14].

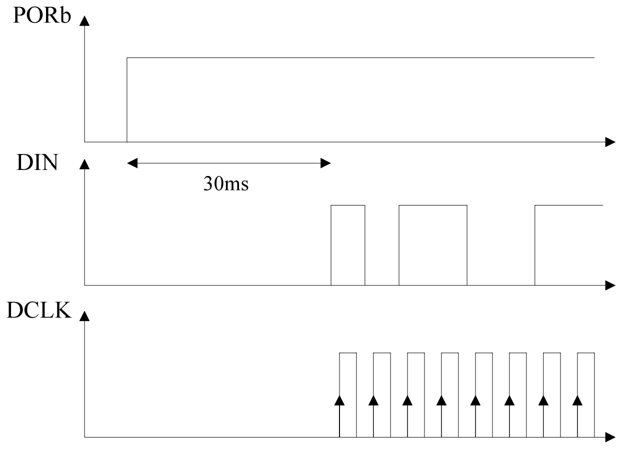


Fig. IV. 8 - Byte transmission to FPAA.

IV.5 Software Strategy

Efficiency considerations suggested to develop the control routines mandated to supervise both the digitization phase and the dynamic FPAA configuration in assembly language. On the contrary, measurement algorithms, mandated to process acquired signals and to calculate the optimal FPAA parameters have been written in an ANSI-x3.159-1989-compliant C language.

As illustrated in Fig. IV. 9, the measurement procedure, implemented by the proposed software architecture, consists of different stages, details of which are given in the following.

IV.5.1 Initialization

The first stage of software architecture is mandated to the initialization of DSC and the primary configuration of FPAA.

In this section, DSC timers are initialized and 12-bit analog-to-digital converter is configured. A specific choice of ADC configuration registers causes the ADC to start a new sampling phase anytime a conversion is terminated on that channel.

FPAA is reset, by setting the DSC line (pin RC15), associated to FPAA input PORb, at logical level high. Timer 1 is employed, in order to wait at least 30ms, required for the FPAA reset routine. DSC, then, transmits the primary configuration to the FPAA, through port lines (RC13 and RC14) linked to FPAA pins DCLK and DIN.

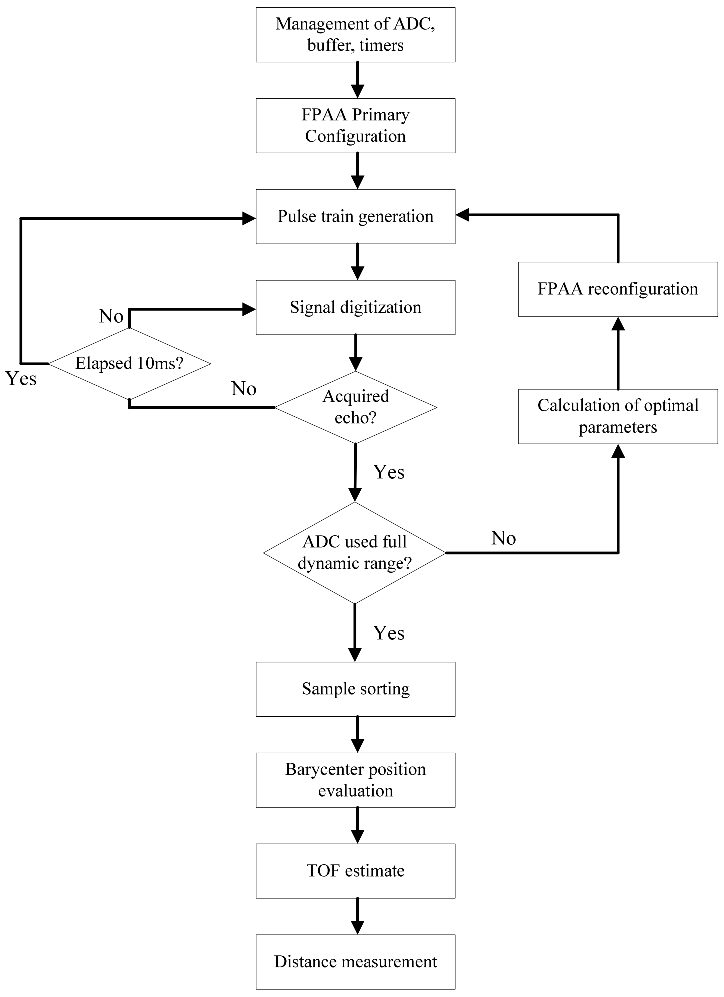

Management of ADC, buffer, timers
FPAA Primary Configuration
Pulse train generation
Signal digitization
FPAA reconfiguration
No
Elapsed 10ms?
Yes
No
Acquired echo?
Calculation of optimal parameters
Yes
ADC used full dynamic range?
No
Yes
Sample sorting
Barycenter position evaluation
TOF estimate
Distance measurement

Fig. IV. 9 - Flow diagram of measurement process.

IV.5.2 Generation of pulse train

When measurement starts, the electrical stimulus for the ultrasonic transducer is generated by alternatively toggling the logic state of an output DSC line (RD9) with generation time managed by the DSC internal timer Timer1. In order to generate a pulse train with 45 kHz frequency, RD9 state has to be toggled each 11.1 μs. Then, Timer1 is employed to set a flag when this time is elapsed.

During this stage, the ADC module has to be turned off in order to avoid the digitization of the signal portion accounting for ultrasonic generation and acoustic ringing. At this aim, Timer1 is reconfigured to wait a time interval that through experimental test has been estimated equal to 2.5 ms. This time establishes the lower limit of the sensor measurement range to 450 mm. For smaller distances, in fact, echo is received before 2.5 ms are expired and, thus, cannot be digitized.

IV.5.3 Ultrasonic signal digitization

Once 2.5 ms time has expired, the ADC module is turned on. The memory buffer reserved to collect the digitized samples is managed through a first-in-first-out strategy. The maximum buffer dimension allowed by employed DSC is 320 data samples. Since the sampling frequency f_s is equal to 140 kHz, the time window that can be saved in memory is about 2.3 ms. This time is sufficient to include the received echo, but it is not enough large to cover the entire TOF. In order to have information about the elapsed time before echo detection, then, a counter variable *C* has been necessary. *C* is increased each time the buffer size is overflowed and echo has not still occurred.

Digitization ends if one of the following conditions is met: (*i*) an ultrasonic echo has been detected and received, or (*ii*) measurement time has expired (time out condition).

As for (*i*), an interrupt service routine is executed to identify the occurrence of the received echo. In particular, microcontroller is configured to set the ADC flag high each time 8 successive samples containing an ultrasonic carrier period are acquired. Samples are acquired according to a double buffer technique; while eight samples are digitized, DSC compares the other eight samples with a fixed threshold to find out the echo start. If a sample is greater than the set threshold, a trigger condition occurs, a software flag is set and the current trigger position is saved. From now on, signal conversion continues for the next 280 samples. In this way, the digitized signal consists of 40 pre-trigger and 280 post-trigger samples. When all the samples have been acquired, they are analyzed according to the saved trigger position.

The time out condition (*ii*) is handled by reading (through Timer1) a time interval equal to 10 ms (corresponding to a full-scale distance of 2000 mm at a temperature of 24°C). The range overflow evidenced by the time out condition stops the measurement and set an error condition.

IV.5.4 FPAA reconfiguration

Once echo envelope has been reconstructed, a control on the sample range is executed. FPAA is reconfigured if: *(i)* distance is too low, signal amplitude is greater than ADC full dynamic range; *(ii)* distance is too high, then echo attenuation does not allow exploiting the desired ADC dynamic range. If one of these events occurs, DSC re-executes the algorithm for computing a new optimal set of filter parameters (as explained in Section IV.2). When the new capacitance values have been computed, the reconfiguration block is built and sent on the fly to the FPAA.

IV.5.5 Measurement algorithm

Thanks to FPAA characteristics discussed in the previous sections, measurement algorithm is based on the evaluation of parameters related to echo envelope waveform. In particular, TOF measurement is executed by estimating the time location of the envelope barycenter, computed as the average of sample positions, weighted by their amplitudes:

$$i_B = \frac{\sum_{k=1}^{N} i_k \cdot V_k}{\sum_{k=1}^{N} V_k} \qquad \text{(IV.1)}$$

The result i_B is the position of the barycenter in the array of 320 samples acquired when the echo has been detected.

TOF measurement must be compensated for systematic errors imposed by two main reasons: *(i)* the barycenter position, as expected, is delayed from the instant of echo onset; *(ii)* FPAA analog filters produce a significant time shift on the echo envelope (as clearly evidenced in Fig. IV. 8). Both effects *(i)* and *(ii)* can be easily removed by a compensation numerical technique. At this concern, preliminary tests have been performed in order to asses that the above mentioned systematic delay is really independent on the target distance. Thus, the compensation delay has been evaluated by executing averaged measurements, for different nominal distances, in the total measuring range. Obtained echo envelopes, for different distances, are shown in Fig. IV. 10. Time axis starts from the pulse train generation; stars account for the barycenter position for each distance, whereas stems indicate the nominal onset time. Experimental results prove that the offset is practically independent on the nominal distance and its mean value is about 510 ns.

In conclusion, the evaluated correction parameter, namely *offset*, has been introduced in the DSC algorithm to produce the compensated measurement of *Time-Of-Flight*.

TOF, is then obtained by adding the three times as follows: *(i)* the barycenter time delay, *(ii)* the time elapsed before echo detection and *(iii)* the time waited after pulse train generation:

$$TOF = \left(i_B + C\right) \cdot T_s + 2.5ms - offset \qquad \text{(IV.2)}$$

In eq. (2) *Ts* means *sampling period.*

Equation (IV.2), if compared with threshold methods, allows increasing measurement resolution, but it does require a computational burden much lower than that needed in methods based on the interpolation of echo rising edge.

Finally, the distance *d* is easily obtained from the TOF according to the equation:

$$d = \frac{TOF \cdot v_s}{2} \qquad \text{(IV.3)}$$

Where v_s is the velocity of sound in the propagation medium. The velocity of sound is not constant, but it is affected by the environmental temperature according to the well known formula:

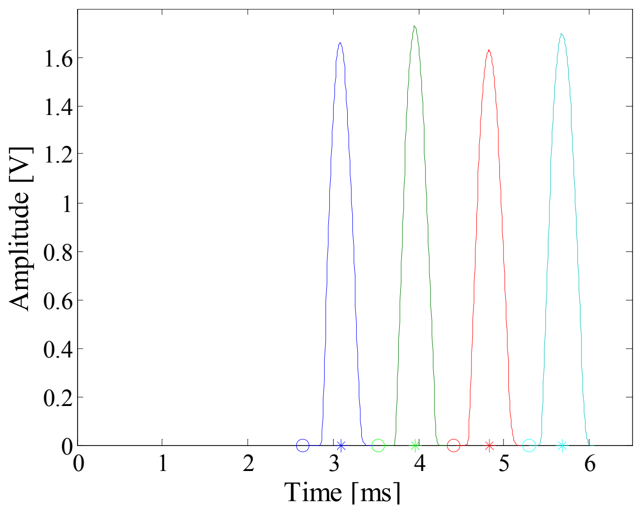


Fig. IV. 10 - Echo envelopes at 450, 600, 750 and 900 mm.

$$v_s = 20.05\sqrt{T} \qquad \text{(IV.4)}$$

Where T is the temperature expressed in Kelvin.

During the research activity, two feasible solutions have been considered: *(i)* to measure the propagation medium temperature in order to directly apply equation (IV.4); *(ii)* to deduce the actual sound velocity by measuring the time of flight of the echo returned from a reference target placed at calibrated distance. At the present, the solution *(ii)* has been realized.

IV.6 Experimental Assessment of the Sensor

A number of experimental tests in free-space propagation have been conducted to assess the performance of both the sensor prototype and measurement algorithm. In particular, the capability of measuring the distance of a reflecting plane characterized by orthogonal alignment with the transmitted ultrasonic signal has been investigated.

At this aim, the piezoelectric transducer has been mounted on a movable plane sliding on a calibrated distance slide and different distances ranging inside the interval 450÷1150 mm have been examined. For each distance, one hundred measurements have been iterated.

Fig. IV. 11 shows results obtained for experimental standard deviation and Integral Non-Linearity (INL), calculated as

$$INL\% = \frac{d_{meas} - d_{nom}}{d_{nom}} \cdot 100 \qquad \text{(IV.4)}$$

where d_{meas} is the distance measured by the sensor (after systematic offset compensation) and d_{nom} is the nominal value at each distance. The INL has been compared with that obtained from a sensor having the conditioning block with constant parameters.

As regards the conditioning with fixed parameters, since it has been designed in order to provide the optimal SNR in the center of the measurement range (800mm), it is possible to notice that the INL% hugely increases as the target distance is far from 800mm. With the adjustable conditioning section based on FPAA, instead, conditioning parameters can be adjusted so to obtain similar echo envelope at every target distance, thus, reducing the effect of envelope modification on measurement accuracy. It is possible to observe that the INL% is limited within the range -0.1%-0.1%.

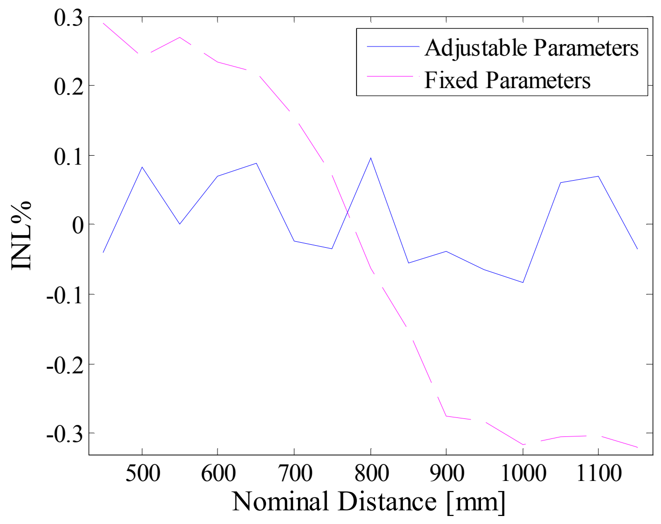


Fig. VI. 11 – INL versus nominal distance.

In Fig. IV. 12 information about sensor repeatability is given. Advantages of FPAA adaptability also highlighted by the obtained repeatability; type A uncertainty, evaluated with coverage factor equal to 3, always results lower than 400 µm. Whereas the uncertainty proves to get worst when the distance increase when a conditioning block with fixed parameters is employed.

By combining the obtained INL and standard deviation, resolution values of 1 mm have been attained.

In conclusion, the prototype grants similar or superior performance with the respect to other solutions already available on the market [15-16], though it stands out for its lower cost.

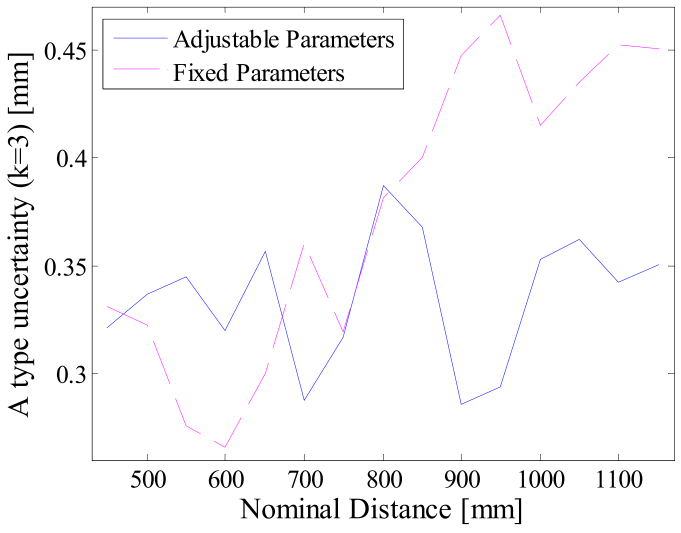


Fig. IV. 12 - Expanded uncertainty with coverage factor 3, versus nominal distance.

References

[1] J. Silva-Martinez, E. Sánchez-Sinencio, "Switched Capacitor Filters", In *Handbook of Circuits and Filters*, CRC Press, 2003.

[2] D. N. Cheeke, Fundamentals and Applications of Ultrasonic Waves, CRC Press, Florida, 2002.

[3] A. M. Sabatini, "A digital signal-processing technique for compensating ultrasonic sensors," *IEEE Trans. on Instr. and Meas.*, Vol. 44, No.4, pp.869-874, August 1995.

[4] V.Magori, "Ultrasonic sensors in air," *Proc. Of IEEE Ultrasonics Symposium*, 1994, Vol.1, pp. 471-481.

[5] L. Angrisani, A. Baccigalupi, R. Schiano Lo Moriello, "A measurement method based on Kalman filtering for ultrasonic time of flight estimation", *IEEE Trans. on Instr. and Meas.*, Vol. 55, No.2, pp. 442-448, April 2006.

[6] G. Bucci, C. Landi, "Numerical Method for Transit Time Measurement in Ultrasonic Sensor Application", I*EEE Trans. on Instr. And Meas.*, Vol. 46, No.6, pp.1241-1246, December 1997.

[7] H.Eriksson, P.O.Börjesson, P.Ödling, N.G.Holmer, "A robust correlation receiver for distance estimation," *IEEE Trans. on Ultras., Ferr., and Freq. Contr.*, vol.42, No.5, pp.592-601, 1993.

[8] P.Ramuhalli, J.Kim, L.Udpa, S.S.Udpa, "Multichannel signal processing methods for ultrasonic nondestructive evaluation," *Proc. of Sensor Array and Multichannel Signal Processing Workshop*, pp.229-233, Aug. 2002.

[9] A. Püttmer, J. Nowottnick, P. Hauptmann, "High-accuracy measurement of pulse amplitudes for new applications of ultrasonic sensors", *Sensors and Actuators A: Physical,* Vol. 68, Issue 1-3, A, June 15, 1998, pp. 454-459.

[10] G. Merendino, S. Callegari, A. Golfarelli, M. Zagnoni, M. Tartagni, "Signal conditioning for capacitive sensors with field programmable analog arrays", *Proc. of International Symposium on Signals, Circuits and Systems*, 2005, Vol. 1, pp. 139 - 142.

[11] A. Baccigalupi, A. Liccardo, "Condizionamento dei segnali ecografici mediante Field Programmable Analog Arrays", Relazione ad invito in *Atti del Congresso Gruppo Misure Elettriche ed Elettroniche*, pp. 337-346, L'Aquila, Italia, 11-13 Settembre 2006.

[12] Anadigm Company, "Anadigm Designer IP Module Manual", 2003, www.anadigm.com.

[13] Anadigm Company, "Anadigm AN221E04 Datasheet", 2003, www.anadigm.com.

[14] P. I. Yakimov, E. D. Manolov, M. H. Hristov, "Design and implementation of a V-f converter using FPAA", 27th International Spring Seminar on Electronics Technology: Meeting the Challenges of Electronics Technology Progress, 2004, Vol. 1, pp. 126-129.

[15] MS6502 specifications, http://www.sensocomp.com.

[16] Acu-trac™ specifications, http://www.ssitechnologies.com.

CONCLUSIONS AND FUTURE DEVELOPMENTS

The presented research work has concerned with the realization of a measurement laboratory configurable and programmable by remote users.

The research activity has been initially focused on the exploration of a software solution allowing: *(i)* students access to the laboratory without downloading further software; *(ii)* easy integration of GPIB routines for the communication with instruments; *(iii)* interoperability with other software, in order to obtain flexibility in developing the client application. At this aim, after a deep study of the solutions available in literature, a solution based on .NET Web Services has proved to be the most suitable for these purposes.

A Web Service exposing the functions for communicating with different measurement instruments, then, has been developed. In particular different classes for each instrument connected to the measurement station have been implemented. The classes define object employed for transferring the configuration parameters (from the user to the instrument) and receiving the measurement results (from the instrument to the user). Moreover, these classes are employed for creating the messages for instruments according to syntax SCPI. Successively, the web methods making use of this classes have been developed; the methods receives, as inputs, the object-*instrument*, passed by reference, and a parameter identifying the user, passed by value. They extract, then, the configuration to be sent to the real instruments and fill the buffer with the instrument response.

Then clients application have been set up. Two examples, respectively in LabVIEW^{TM} and Macromedia Flash environment, have been presented. The client program, according to user's selection, has to build the SOAP message, in order to properly invoke the Web Service and transmit it through TCP/IP protocol; it has, then, to extract the measurement results from the SOAP response received by the server.

Another problem faced during these years has dealt with the possibility offered to the students of remotely operating on the analog circuit connected to the measurement station. Typically, in fact, the test circuit electrically wired to the instrumentation, is made of analog components not capable of exchanging communication data. The research activity has been focused, then, on the introduction of programmable devices as test circuit in the laboratory. In particular, Field Programmable Analog Arrays (FPAA), have been employed. FPAA can be configured through serial communication and can realize the transfer function of various analog circuits. Another web method have been realized and the Web Service has been update; this way students are able to select the analog circuit on which to perform the experiment and to control the instruments for executing the measurement.

Finally, the realized laboratory has been tested. The remote measurements have been used also for characterizing the analog circuit emulated by the FPAA. In particular the differences between the expected output and that obtained for three different circuits (amplifier, filter and integrator) have been evaluated. Principal blocks, such as amplifiers and filters, have been considered.

As far as it concerns the amplifier, the following conclusions can be summarized: in the gain range 1÷50, the discrepancy between the actual in band gain and designed one have been measured and never exceed ±0.5 dB. The observed discrepancy is expected to depend mainly on the saturation effect of the output amplifier. In particular, we suggest to maintain the output swing inside ±1V in order to obtain an accuracy of some 1%. The effect of the quantization error introduced by the programmable capacitances that can assume only quantized values (0÷255 capacitance units) is, instead, negligible. Good results have been experienced also in the amplifier frequency response, where the actual -3dB bandwidth reproduces with high fidelity the 0÷100kHz range declared by Anadigm. Also satisfactory is the response in terms of in band ripple.

As for the filter characterization, different tests have been carried out in order to verify how faithfully the filter response matches the one expected by its mathematical model. In particular, parameter such as gain, passband and stopband frequencies have been taken into account. The achieved results show a slight difference between the amplification nominal set value and the measured one. Also in this case the output swing is a limiting factor.

In conclusion, even though some discrepancies between design parameters and actual measurements have been evidenced, FPAA devices are of sure interest in the realization of conditioning systems where the flexibility is a major constrain compared with parameter accuracy.

The last section of the work has been dedicated to the implementation of a remote configurable sensor. FPAA has been employed as the adaptive conditioning block of a distance sensor based on the estimate of time of flight of ultrasonic signals. The conditioning circuit is composed by: *(i)* a bandpass filter; *(ii)* a full wave rectifier; *(iii)* a peak detector; *(iv)* a lowpass filter; *(v)* an amplifier. By dynamically varying gain and quality factor of the bandpass filter, shape and amplitude of echo envelope independent on the target distance are obtained. So a web method for the dynamic reconfiguration of the bandpass filter parameter has been developed. Students implement in LabVIEW environment their own algorithm for researching the circuit parameter providing the optimal echo shape in every measurement condition.

On the basis of students assessment of the sensor, a prototype based on a DSC has been set up. The DSC provides the stimulus signal for the piezoelectric transducer and acquire the echo envelope conditioned by the FPAA. Echo amplitude is, then, compared with two threshold; if it is too attenuate or too amplified, the DSC executes the algorithm for detecting the optimal parameters and sends the proper reconfiguration to the FPAA. Several experimental tests in free-space propagation have been conducted to assess the performance of the sensor prototype. As performance factors, (*i*) the Integral Non-Linearity (INL) expressed in percentage form, *(ii)* the experimental standard deviation, *(iii)* the sensitivity, and *(iv)* the resolution have been taken into account. Experimental results prove the advantages associated to a dynamically reconfigurable conditioning block. Sensor shows an INL% always lower than 0.1%: changing filter parameters, in fact, made it possible to make INL% values stay within a fixed tolerance range. Moreover, an experimental standard deviation lower than 400μm, sensitivity very close to 1 and resolution of about 1mm have been appreciated. The prototype grants similar or superior performance compared with other solutions already available on the market, though it stands out for its lower cost.

A wide variety of developed solutions has been included in the Remote Didactical Laboratory "G. Savastano", whereas the laboratory implemented at the University of Naples has been widely exploited for measurement courses as Measurement for Diagnostics and Quality and Sensors and Transducers.

Future developments aim at the production of the laboratory experiments library, particular for different measurement disciplines. Then, when other analog circuits will be employed, their characterization, as it has been done for the amplifier and the bandpass filter, will be required.

LIST OF FIGURE CAPTIONS

LIST OF TABLES

RINGRAZIAMENTI

Voglio cogliere l'occasione per ringraziare le persone con le quali ho condiviso la mia vita negli ultimi 3 anni. So bene che non riuscirei a dire loro ciò che penso senza commuovermi, quindi spero che abbiano occasione di leggere queste poche righe.

Innanzitutto ringrazio quel santo uomo che mi sopporta ventiquattro ore al giorno; che mi fa da collega, amico, confidente, compagno… Grazie Rhos, per aver cambiato la mia vita e perché, ogni giorno, mi dai la gioia di averti ancora accanto.

Ringrazio Antonio, Mauro, Michele, Nicola, Umberto, per avermi offerto, in ogni, momento, il loro aiuto ed appoggio. Ma più di tutto mi hanno offerto la loro amicizia; ho avuto la fortuna di trovare persone con le quali non ci si annoia mai e la stanchezza, anche dopo 10 ore davanti ad un monitor, non si fa mai sentire. Loro non sanno che la mia decisione di continuare il dottorato è stata dettata dalla convinzione che difficilmente avrei trovato di nuovo un ambiente tanto piacevole ed avrei avuto la fortuna di collaborare con delle persone così.

Inoltre vorrei ringraziare le persone che, in questi anni, hanno assunto per me il ruolo di guida e dalle quali ho avuto modo di imparare tanto.

Ringrazio il Prof. Claudio De Capua, che mi ha iniziato a questo bellissimo mestiere; lo ringrazio molto per l'aiuto che mi ha dato, per l'affetto e l'umanità che mi ha sempre dimostrato e per i suoi innumerevoli "non mollare".

Ringrazio il Prof. Nello Polese, per i preziosi consigli, per la fiducia che mi ha sempre riservato e per le possibilità che mi ha offerto.

Grazie al Prof. Carmine Landi per i tutti i suoi insegnamenti. Mi ha incitato ad affrontare gli argomenti con maggiore spirito critico e a “lavorare bene e seriamente”. Lo ringrazio, inoltre, per aver creduto in me; in pochi mesi ho visto cambiare tante cose e so che, per questo, gli devo tanto.

Ringrazio il Prof. Aldo Baccigalupi per tutto ciò che mi ha trasmesso nell’ultimo anno. La sua passione per la ricerca e la dedizione per l’insegnamento mi hanno permesso di imparare molto più di quanto avessi potuto sperare. La grande competenza, unita alle innumerevoli qualità umane hanno reso piacevoli le tante ore di laboratorio ed hanno decisamente cambiato la qualità del mio lavoro.