Remote Laboratory Design and Implementation as a Measurement and Automation Experiential Learning Opportunity

Francesco Bonavolontà, Mauro D'Arco, Annalisa Liccardo and Oscar Tamburis

Designing and implementing a remote laboratory has long been a professional job, carried out by experts working in synergy with teachers. Nowadays, thanks to the availability of advanced tools, the implementation can be assigned as a student project related to advanced measurement and automation classes. The development of the project happens to have an important educational validity, since it allows the student to get familiar with principles and technologies that characterize intelligent manufacturing and smart environments, as central elements of Industry 4.0. In particular, the student has the opportunity to increase knowledge of Internet technologies that more and more often are substituting for the proprietary solutions, previously adopted in the industry for process monitoring and automation.

The primary focus of the work presented in this paper is the educational validity of designing and implementing a remote laboratory and only secondarily the possibilities related to its use. The authors present a straightforward solution to guide the students throughout the realization of a remote laboratory that functions to accomplish several complementary goals of courses dealing with remote process control and automation.

Remote Laboratories Assets

Remote laboratories represent nowadays a consolidated useful educational tool in a wide range of secondary school and university programs, especially concerned with physics and engineering courses [1]. They allow users to perform experiments and laboratory tasks through an interaction that takes place at a distance with the assistance of the remote infrastructure [2].

Fruitful debates about web-based education have already addressed a large variety of topics [3], [4]. Differences between a direct interaction with visual and/or audio feedback and a remote one have been deeply analyzed. Benefits related to convenience, cost-efficiency, and e-learning possibilities of a remote laboratory, which is available 24 hours per day from any Internet-connected location, have been discussed. In particular, it has been highlighted that the laboratory demand can be satisfied with less equipment, and that expensive instruments can be shared and utilized more effectively. Remote laboratories offer opportunities for students, unable to attend classroom interactions, to not forsake laboratory activities. Nonetheless, remote laboratories grant superior security and safety: experiments that involve risk hazards to personal safety can be carried out with less concern, and equipment are more preserved since hazardous areas can be interdicted to public access [5].

The design and implementation of a remote laboratory has long been a professional job, carried out by experts of Internet and measurement-and-automation technologies working in synergy with teachers [6]-[8]. The job requires, in fact, a preliminary understanding of the educational purposes, which are specified by the teacher, and follows a thorough analysis of the suitability of the experiment to the remote use, which is assessed by both teachers and technology experts, and, finally, the engineering development, which mainly commits the technology expert. More specifically, a candidate experiment must be analyzed to determine the ways in which the user interacts with apparatus to affect learning outcomes and ensure that these can be conveyed effectively by the remote infrastructure. This impacts the design of the user interface, which has to convey the physical aspects of the experiment [9]. The expert technician has to evaluate Internet connection bandwidth and client software/hardware prerequisites. If video and audio monitoring are required, the user has to rely on a broadband Internet connection. The software to be installed on the user's system could require a given operating system and be affected by operating system software obsolescence [10]. Besides, the technician has to set up a fault tolerant solution that prevents individual malfunctions that may shut down the entire laboratory operation [11].

At present, thanks to the availability of advanced tools, setting up a remote laboratory requires fewer efforts and can be rapidly afforded. It can even be assigned as a student project related to advanced measurement and automation courses, as those typically taught in electrical engineering master degree paths.

1094-6969/19/$25.00©2019IEEE

Remote Laboratories Technology

Remote laboratory technologies are typically based on server/client software applications. The server application runs on a machine situated in the physical laboratory and directly interfaced to the equipment. The client application is installed on the user machine, which is required to have Internet access, and lets the user specify a measurement and automation process, eventually made up of several individual tasks to be executed in a temporized sequence [12]-[14].

Server and client applications can be easily programmed using different advanced software tools that include functions to communicate via Transmission Control Protocol/ Internet Protocol (TCP/IP) protocol to remote machines. These functions are nowadays available as portable executables in a variety of software environments.

The server application essentially has to get from clients directives related to the control of local instruments and actuators. These directives can be represented in terms of structured data. For instance, in the proposed approach, the server gets an array of elements, each one consisting of a data structure detailing a task. The presence of more elements in the array means that the client is requesting a sequence of tasks to be executed in the received order. The server then executes the directives by locally controlling instruments and actuators.

The client application allows the user to specify the sequence of tasks by configuring the array of data structures and collect some results at the completion of the tasks.

Server Application

The server application creates a listener for a Transmission Control Protocol (TCP) network connection on a given port input; to this end, suitable library functions are exploited. These functions allow the programmer to select which network to listen to, i.e., which one of the Ethernet cards, if there are more than one, and the port number on which to listen for a connection. The port number should be chosen among valid values, which are in the range between 49152 and 65535, as specified by the Internet Assigned Numbers Authority (IANA). Unfortunately, not all operating systems follow the IANA standard; for example, Windows allows the use of ports between 1024 and 5000, which are a portion of registered port numbers, ranging from 1024 up to 49151, and the programmer should be aware that such a choice could lead to resource conflicts. The functions typically return a network connection identifier for the listener and an error message containing error information, if any.

The creation of a TCP network connection is related to the configuration of the resources but has to be followed by an explicit instruction to definitely set the server to wait for a TCP connection. This instruction relies on the execution of another function that, according to the input parameters, gets the server to operate either indefinitely or for a limited time interval and eventually exit with a timeout message if no connections occur. At any connection, the function acquires the IP address and the port number used by the remote client. It also produces a single identifier for that TCP connection, to uniquely refer to it afterward. In common paradigms related to server applications programming, the function is configured to wait indefinitely for a client connection, and it is inserted into a repeating structure, in order to set again the server to wait for a subsequent connection after any reception.

For active connections, data exchange between clients and server are controlled by means of TCP read and TCP write library functions.

The server uses the TCP read function configured to wait until all requested bytes arrive, unless the time specified in timeout runs out. At any TCP connection it expects to receive a short binary string, made up of 8 bytes, that is interpreted as two consecutive 4-byte integers. The first integer is used to implement an authentication scheme, according to which the server utilization can be either granted to authorized users or denied to those who are unauthorized. The second integer provides the size, in terms of number of bytes, of the message to come immediately after, referred to as payload.

It is opportune to agree on an upper bound for the size of the payload to avoid errors in data exchange caused by limited memory buffer availability. The payload contains all directives to the instrument and an actuator controller to perform a process as a sequence of tasks. The TCP read function first acquires from the connecting client the 8-byte header and distinguishes the two integers, namely the authentication key and the size of the payload.

If authentication fails, the unauthorized TCP connection is straightforwardly closed. It is worth noticing that, different from a client application of an authorized user, which is programmed to send a string containing data related to the sequence of tasks immediately after a connection acknowledgement, an unauthorized user could make the server stuck by not sending any other byte. This is the reason why the TCP read function operates on a timeout basis; if no byte is received after a reasonable limited time, the TCP connection is closed. Nonetheless, the use of the header to declare the length of the payload allows the server to configure the TCP read function and escape from waiting on absent bytes.

If authentication succeeds, the server uses again the TCP read function to acquire the payload, which is a string that can represent a highly-structured data type. The string, in fact, is produced at the client side through conversion operations and in its final form contains header descriptors before each component of the data structure: descriptors declare size and data type for forefront and nested elements and allow the data to reconvert to their native structure.

The payload, introduced as an array of structured data describing a sequence of tasks, is then passed as a parameter to a subroutine that plays the role of instrument and actuator controller. The subroutine returns the results obtained by executing all of the tasks.

Results are less structured data, typically consisting of an array of strings. Each string is related to the results of a task specified by the client, in the corresponding order.

The server converts the array of strings to a unique string by introducing header descriptors, and it is sent by means of

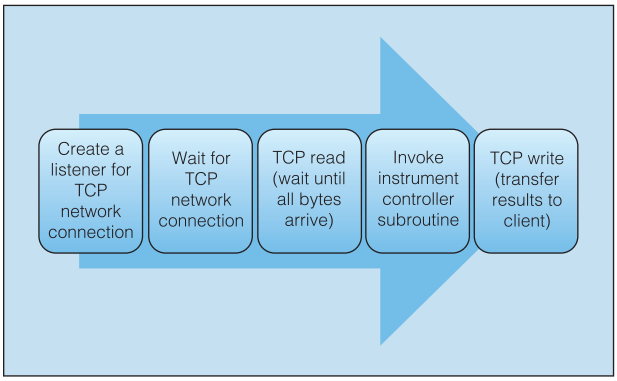

Fig. 1. Conceptual diagram highlighting the main operations of the server application.

a TCP write function as payload to the client. To this end, the server uses the identifier for that TCP connection, produced at the very beginning of the client connection. Nonetheless, it also measures the length of the payload in terms of number of bytes and conveys this information with a 4-byte integer header added to the string sent to the client. Notice that the protocol adopted in client-to-server data transfer is asymmetrical with respect to the server-to-client one, since the latter does not require an authentication procedure.

Finally, the server uses for the last time the identifier of the TCP network connection to close it. The main operations performed by the server application are summarized in Fig. 1.

Instrument and Actuator Controller

The instrument and actuator controller uses one or more interfaces to send configuration commands and acquire measurement results from peripheral instruments. As for TCP communication, software environments contain suitable library functions to exploit the capabilities of several standard interfaces; nonetheless, they can be integrated with additional portable executables to cope with interfaces not included in the standard list.

Commands can be related both to the interface and peripheral-dependent functionalities. They consist of messages, typically represented by short byte sequences, and their effect is explained in the documentation of the equipment, namely the interface and peripherals programmer manuals.

The controller of the proposed remote laboratory scans the input array, singling out its elements in the given order. Each element contains a data structure describing a task to be performed, and each task concerns the configuration of a single peripheral. The data structure utilized to describe the task contains four mandatory fields: ID identifying the peripheral, code to select between two basic operations, message to specify commands to be sent or bytes to be collected, and a timing directive to control the execution of the task operation.

Depending on the adopted interface, the ID received by the controller can be an abstract ID, linked to a local ID and managed by the controller.

The basic operations consist of: byte transfer from controller to peripheral for configuration purposes (writing

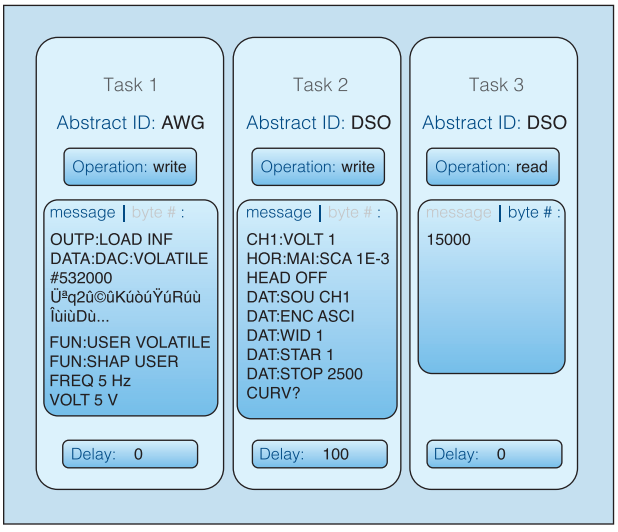

Fig. 2. Instance of a sequence of tasks specified by a remote user in a simple application aimed at verifying the linearity of a DUT.

operation) or byte transfer from peripheral to controller for measurement results acquisition or status verification purposes (reading operation). The code identifying the basic operations allows the controller to invoke the right interface function; the message in the structure is just an input parameter for the interface function.

The timing directive allows the user to delay the execution of an operation: it is useful anytime there is a need in the process to wait for settling phenomena before measuring a quantity or maneuvering an actuator.

As an example, an instance of a sequence of tasks specified by a remote user in a simple application, aimed at verifying the linearity of a device under test (DUT) in terms of third order intercept (TOI), is given in Fig. 2. In this application the input port of the DUT is connected to an arbitrary waveform generator (AWG) and the output port to a digital storage oscilloscope (DSO).

The first task in the sequence configures the AWG to output a dual-tone test signal. To this end, the signal is synthesized in digital form by the user, formatted as a binary block, and included as a parameter in a command string to be downloaded in the volatile waveform memory of the AWG. The reproduction in analog form of the test signal is then activated with additional configuration commands appended to the same task instance. Also, additional commands are required to control the burst reproduction rate and amplitude features.

The second task in the sequence configures the DSO to acquire the output of the DUT. The samples stored into the DSO acquisition memory are retrieved with the execution of the third task, where the controller is asked to read 15000 bytes through the interface from the addressed DSO.

The controller executes all of the tasks in the given order and collects, at the completion of each of them, the data produced by the query. The first and third tasks are executed immediately, i.e., with 0 delay from the reception of the message, whereas the second one is executed with a delay of 100

ms, to wait for the expiration of settling effects characterizing the AWG operation. At the completion of each task, the controller inserts a string in an array; empty strings are inserted for those tasks that do not query for instrument outputs. The array returned to the server by the controller contains as many strings as the tasks required by the client.

The client will finally retrieve through the server the DUT output acquired by the DSO and exploit a digital signal processing approach to perform spectral analysis and estimate the selected figure of merit (TOI).

Client Application

The client application has to open a TCP network connection with the remote server identified by the IP address and port number on which the laboratory service is made available. To this end, software environments offer library functions with an IP address, remote port number, time out, and local port number as configurable input parameters. The time out parameter is to complete the function and return an error if the connection is not established in a specified time, as could occur in case of server unavailability. The local port number parameter differs from the remote port number of the server application and allows the user to explicitly choose the port number utilized by the client in the TCP connection. The programmer can often rely on the operating system to allocate an unused port to the purpose, but if the server only allows connections to clients that use local port numbers within a specific range, it is necessary to have control of this parameter. Opening a TCP connection returns a single identifier for the TCP connection as well as an error message when failing to connect.

A successful TCP connection permits the client to exploit the remote server and hence, to direct the instrument and actuator controller.

Directives are included in the message the client transfers to the server. The message has to comply with the protocol defined by the programmer. In the proposed approach, the message has to contain an 8-byte header for both authentication (4 bytes) and declaration of the length of the payload of the message (4 bytes) expressed in terms of number of bytes. The original payload consists of an array that typically contains structured data elements to represent the demanded tasks. In order to be transferred through the TCP connection, it is converted to a string: the use of header descriptors will let the server application re-convert the payload to its native data type.

The client concatenates the header and the payload strings and uses a TCP write function to transfer the resulting string message to the remote server. The function requires two input parameters: the single identifier for the TCP connection and the message. It operates on a timeout basis, so that if it does not complete after a reasonable limited time, it returns an error. In the presence of a timeout error, the client application closes the TCP connection, popping up a dialog message to the user and encouraging retrying later.

If the TCP write function completes with no errors, the client can pause for a while to let the remote server execute the demanded tasks: the duration of the pause has to be calibrated according to the sequence of tasks.

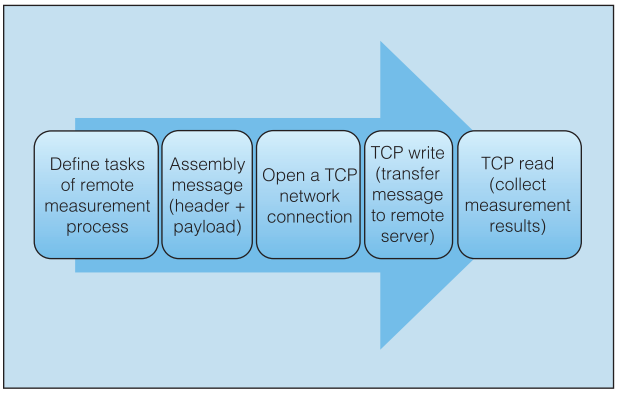

Fig. 3. Conceptual diagram highlighting the main operations of the client application.

After the pause the client uses a TCP read function, configured to wait until all requested bytes arrive, to collect the results. To this end, it first invokes the function to read a short string made up of 4 bytes representing an integer number and get acquainted with the length of the payload. Then, it invokes the function again to read the remaining part of the message, i.e., the payload. In both cases, it provides two input parameters to the TCP read function: the identifier of the TCP connection and the number of bytes to read. Remember that the protocol adopted in server-to-client data transfer is asymmetrical with respect to the client-to-server one, since the former does not require an authentication procedure.

Finally, the client uses for the last time the identifier of the TCP network connection to close it. The main operations performed by the client application are summarized in Fig. 3.

The client application cannot jeopardize the user interface, which according to the goals of the remote laboratory can be characterized by different degrees of abstraction. Namely, if the final users of the laboratory are neophytes, such as students of secondary schools or bachelor degree courses, it is convenient not to worry them with issues related to the syntax of the commands of programmable devices. To this end, the user panel can be designed to include controls and indicators that ease the configuration and results inspection tasks. In this case, the educational goal could require complementary real-time video and audio monitoring. Differently, for advanced courses, the issues related to the use of standard commands for programmable instrumentation (SCPI), adopted by several standard interfaces such as IEEE-488, RS-232, RS-422, Ethernet, USB, VXIbus, or HiSLIP, are significant. The user interface should be designed to allow more visibility of the available interfaces, in order to teach the student to arrange the command list of each task of the process and use the alternative data transfer options in remote controller-device communication.

Supplementary Design Issues

In the previous sections, some important aspects related to error condition and multiple connections handling have not been discussed. These aspects impact the complexity of the

design and implementation of the remote laboratory. They represent more advanced issues to be highlighted in the educational path, as in the following.

The remote laboratory should be programmed to have fault tolerance, so that an error related to a single user will not prevent the laboratory from being used by other clients. To this end, the server application should be capable of skipping operations and take forward client connection closure in case of fatal errors. This can be achieved by programming critical functions such that they:

- accept as an input parameter a list of errors related to the operations executed before by the same client;
- recognize errors occurred during their own run; and
- provide as an additional output parameter the updated list of errors.

The described mechanism allows keeping memory of the errors of the whole critical functions chain related to the laboratory session of a client. At any moment of the laboratory session, it is possible for the current function to judge, on the base of the errors description, if the current operation is meaningful or not; in the latter case, the function is programmed to skip any operations and propagate the error information. The client application gains in efficiency by implementing this error handling strategy, rather than relying uniquely on timeout mechanisms to get rid of stuck conditions.

As for concurrent client connections, these can be managed in different ways, according to the number of users and the probability of concurrence. The simplest way consists of queuing clients according to their order of arrival. A large timeout value, given as input to the TCP open connection function by a client, is often enough to wait for a sufficient amount of time to be admitted to the server utilization soon after the completion of the tasks invoked by the preceding client. Thus, clients can get access to the remote resources without incurring in a fail-to-connect event, only occasionally experiencing a longer wait at connection opening.

Alternatively, the remote equipment in the laboratory must be replicated to allow contemporaneous operation to more clients. For the server to accept more clients at one time, it is necessary that the server application be able to run with some execution parallelism. To this end, the server application has to be realized using a multi-thread programming paradigm. At present, several software environments bring users the advantages of powerful multithreading technology in a simple straightforward way.

In the proposed approach, a viable scheme to assure parallelism consists in framing the server operations with different threads, which can respectively involve acceptance, processing, and disposal of the client request, as schematized in Fig. 4.

In particular, the acceptance thread returns an identifier for the client connection to refer to it afterward, acquires the payload of the message, and invokes a processing thread, providing it with the client ID and related message. After that, the thread can enter a sleep mode, waking up only at a new client connection request.

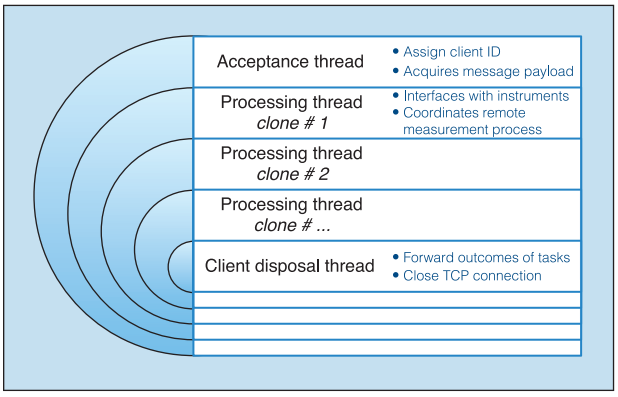

Fig. 4. Server application allowing execution parallelism of more individual threads.

The processing thread can be carried out with the subroutine that implements the controller actions, interfacing with instruments and actuators. The processing thread needs to be cloned in order to have one subroutine for each client active on the remote laboratory and implement parallel operations. Cloning can be prepared statically or even attained dynamically; static cloning is preferable since dynamical creation of clones slows down server operations.

Finally, a disposal thread has to send the message received by a processing thread to the client and perform the close connection. The message received by any processing threads contains both a client identifier and an array of strings with the outcomes of the tasks. As for the acceptance thread, this is a unique thread that interacts with the processing thread and the remote clients via TCP connection.

Conclusions

An educational approach to the implementation of a remote laboratory has been presented. Addressing this theme should get the student familiar with Internet technologies that more and more often are substituting for proprietary solutions, previously adopted in the industry for process monitoring and automation as well as, more in general, with the exigencies of Industry 4.0 program framework. The subject has been developed to provide straightforward guidelines to set-up a remote laboratory, to meet the goals of a course dealing with remote process control and automation. More advanced issues concerning error handling and operation parallelism have also been presented. Finally, a viable scheme to frame the server application in multiple parallel threads has been highlighted.

References

[1] G. Andria *et al.*, "Remote didactic laboratory 'G. Savastano,' the Italian experience for e-learning at the technical universities in the field of electrical and electronic measurement: architecture and optimization of the communication performance based on thin client technology," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 4, pp. 1124-1134, 2007.

[2] A. Baccigalupi, U. Cesaro, M. D'Arco, and A. Liccardo, "Web-based networking protocol for expanding IEEE-488 ATE

capabilities," in *Proc. IEEE Int. Workshop Measurements and Networking (M&N)*, pp.100-104, 2011.

[3] J. M. G. Palop and J. M. A. Teruel, "Virtual work bench for electronic instrumentation teaching," *IEEE Trans. Educ.*, vol. 43, no. 1, pp. 15-18, 2000.

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

[5] L. Benetazzo, M. Bertocco, F. Ferraris, A. Ferrero, C. Offelli, M. Parvis, and V. Piuri, "A web based, distributed virtual educational laboratory," *IEEE Trans. Instrum. Meas.*, vol. 49, no. 2, pp. 349-356, 2000.

[6] P. Arpaia, F. Cennamo, P. Daponte, and M. Savastano, "A distributed laboratory based on object-oriented measurement systems," *Measurement*, vol. 19, issues 3-4, pp. 207-215, 1996.

[7] P. Arpaia, A. Baccigalupi, F. Cennamo, and P. Daponte, "A remote measurement laboratory for educational experiments," *Measurement*, vol. 21, no. 4, pp. 157-169, 1997.

[8] D. Grimaldi and S. Rapuano, "Hardware and software to design virtual laboratory for education in instrumentation and measurement," *Measurement*, vol. 42, no. 4, pp. 485-493, 2009.

[9] M. Corrado, L. De Vito, H. Ramos, and J. Saliga, "Hardware and software platform for ADCWAN remote laboratory," *Measurement*, vol. 45, no. 4, pp. 795-807, 2012.

[10] U. Hernandez-Jayo and J. Garcia-Zubia, "Remote measurement and instrumentation laboratory for training in real analog electronic experiments," *Measurement*, vol. 82, pp. 123-134, Mar. 2016.

[11] N. Wang, X. Chen, Q. Lan, G. Song, H. R. Parsaei and S. C. Ho, "A novel wiki-based remote laboratory platform for engineering education," *IEEE Trans. Learning Technologies*, vol. 10, no. 3, pp. 331-341, 2017.

[12] M. Kalúz, J. García-Zubía, M. Fikar and L. Čirka, "A flexible and configurable architecture for automatic control remote laboratories," *IEEE Trans. Learning Technologies*, vol. 8, no. 3, pp. 299-310, 2015.

[13] A. Chevalier, C. Copot, C. Ionescu and R. De Keyser, "A three-year feedback study of a remote laboratory used in control engineering studies," *IEEE Trans. Education*, vol. 60, no. 2, pp. 127-133, May 2017.

[14] U. Lichtenthaler, "Open innovation in practice. an analysis of strategic approaches to technology transactions," *IEEE Trans.*, vol. 55, no. 1, pp. 148-157, 2008.

Francesco Bonavolontà, Ph.D., is a Research Fellow in the Department of Electrical and Information Technologies at University of Naples Federico II, Italy. His research activity is centered in the area of instrumentation and measurement and is mainly focused on developing of innovative DAS systems based on compressive sampling techniques, also including Internet of Things devices for smart applications.

Mauro D'Arco, (mauro.darco@unina.it), Ph.D., is Associate Professor with the University of Naples Federico II, Italy. His current research interests include remote measurement and automation, data acquisition systems characterized by high bandwidth and high sample rates, arbitrary waveform generators, and digital signal processing themes.

Annalisa Liccardo is Associate Professor with the Department of Electrical Engineering and Information Technology of the University of Naples Federico II, Italy after being a tenure track Researcher from 2013 to 2016. Her current research interests involve the measurements for monitoring and protection of electrical power systems, IoT sensors for electrical measurements, and distributed measurement networks.

Oscar Tamburis, Ph.D., is Assistant Professor of processing information systems in the Department of Veterinary Medicine and Animal Production at University of Naples Federico II, Italy. His research interests include innovation in the eHealth sector, process modeling (UML), DES simulation in health, and implementation of advanced analytics for data-mining.