Work-in-progress on a thin IEEE1451.0-architecture to implement reconfigurable weblab infrastructures

Ricardo J. Costa[12], Gustavo R. Alves[1], Mário Zenha-Rela[2]
ISEP/CIETI/LABORIS[1], FCTUC/CISUC[2]
rjc@isep.ipp.pt; gca@isep.ipp.pt; mzrela@dei.uc.pt

Abstract- **Institutions have been creating their own specific weblab infrastructures. Usually, they use distinct software and hardware architectures comprehending instruments and modules (I&M) able to be parameterized but difficult to be shared. These aspects are impairing their widespread in education, since collaboration between institutions, in developing and sharing resources, is still low. To handle both aspects, this paper proposes the adoption of the IEEE1451.0 Std. with FPGA technology for creating reconfigurable weblab infrastructures. It is suggested the adoption of an IEEE1451.0 infrastructure with compatible instruments, described in Hardware Description Languages (HDL), to be reconfigured in FPGA-based boards. Besides an overview of the IEEE1451.0 Std., this paper presents a solution currently under development which seeks to enable the reconfiguration and the remote control of weblab infrastructures using a set of IEEE1451.0 HTTP commands.**

Keywords: Weblabs, Remote laboratories, IEEE1451.0 Std., Reconfiguration, FPGAs.

I. INTRODUCTION

Typical classrooms are being complemented and, in some cases, replaced by new technology-based tools. This is obvious in faculties, justified by the appearance of the internet and associated services that give students easy access to several educational contents and teachers the adoption of web tools to facilitate the teaching/learning processes. Weblabs are examples of technological appliance in education that allow the access to real laboratories using a device connected to the internet (e.g. a PC) for conducting experiments. Currently, many institutions are applying them in their courses, some available through the library of labs portal [1]. However, there is still a lack of standardization in weblab infrastructures, difficulting the development and the share of resources. These aspects impair institutional collaboration, since the adopted I&M do not use standardized interfaces, despite some proposals both at software and hardware levels [2][3][4][5][6][7]. Furthermore, typical weblab infrastructures use different platforms based on computers and independent instruments, increasing costs and contributing for low reconfiguration capabilities.

It is precisely in this scenario, characterized by current limitations, that the IEEE1451.0 Std. and FPGA technology are seen as possible solutions to create reconfigurable weblab infrastructures. The supporting facts are: the IEEE1451.0 Std. describes hardware and software layers to control and network-interface transducers (which can also form the I&M used in weblabs), and FPGAs can be easily reconfigured with different embedded IEEE1451.0-compatible instruments described in standard HDL.

In this paper, section II briefly presents the main issues of the IEEE1451.0 Std.. Section III points recent applications of the standard, and justifies its adoption together with FPGAs for creating reconfigurable weblab infrastructures, referring previous work made by the authors. Section IV presents the work under development. It describes the implementation of a generic IEEE1451.0 infrastructure in an FPGA (currently interfacing two digital I/O modules), and some guidelines for future developments. Section V concludes the paper.

II. IEEE1451.0 STD. OVERVIEW

The IEEE1451.0 Std. [8] aims to network-interface transducers (sensors and actuators) and defines a set of operating modes, based on specifications provided by Transducer Electronic Data Sheets (TEDS). Defined in 2007, this standard is the basis for forthcoming and previous members of the IEEE1451.x family so they can operate together. The operating modes defined by the standard are controlled using low-level commands that can be applied using a set of APIs.

The standard defines an architecture based on two modules that should be interconnected using an interface protocol: the Transducer Interface Module (TIM) and the Network Capable Application Processor (NCAP). Each module is connected through an interface defined by another standard of the IEEE1451.x family, some already specified according to the IEEE1451.0 Std. (e.g. the IEEEp1451.6 Std. for the CANopen interface) and others intended to be modified in the future (e.g. IEEE1451.2 Std. which defines point-to-point interface). Figure 1 illustrates the IEEE1451.0 Std. main modules, and associated layers.

Each TIM may contain up to 32767 (2^15) Transducer Channels (TC), controlled individually or in groups when they simultaneously control/monitor a specific physical phenomenon or transducer. The TIM and TC behaviors are specified by different TEDS and follow two operation state diagrams and two trigger state diagrams, one for sensors and another for actuators. Each state represents a transducer operating/trigger state that changes according to the defined operation mode. TEDS are implemented in 8 bit (octet) words inside the TIM, or can be remotely located (named virtual

TEDS). Some of them are required to be implemented and others are optional, being the most important and required the Meta-TEDS (defines general TIM features) and the TC-TEDS (defines the behavior of each TC within the TIM).

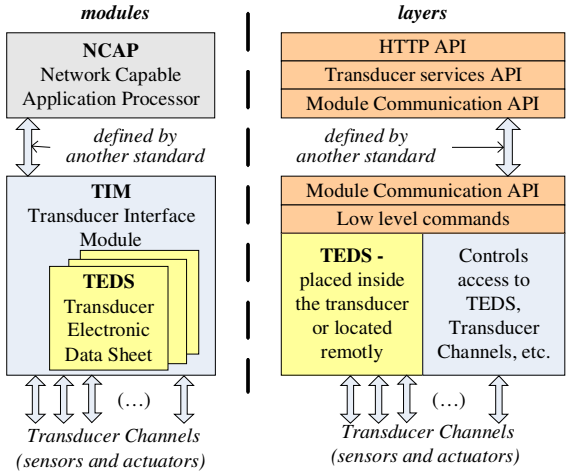

Fig. 1. Main modules and associated layers of the IEEE1451.0 Std..

Controlling the behavior of a transducer is basically done using low-level commands. The IEEE1451.0 Std. defines standard commands and allows manufacturers to define their own commands. The standard commands comprehend common commands (e.g. read/write TEDS, control transducers' behaviors, etc.) and other groups of commands that allow controlling the TIM and TC operation. A command is specified by two octets. The first octet comprehends the class and the second one the function within the class. Some commands have octets (data fields) containing associated data (e.g. for writing TEDS, contents should be specified by these last octets). There are commands that may be applied to the TIM and/or to specific TC, depending on the current transducer operating state.

To monitor the operations of a transducer, namely if a specific command was correctly applied, if there was a change in a TEDS, etc., the IEEE1451.0 also implements a 32 bit status register for each TC and another for the TIM.

A software application is required for accessing (remotely or not) a specific transducer. For this purpose, the IEEE1451.0 Std. specifies a set of 3 APIs:

- Module Application API: implemented both in TIM and NCAP, provides functions to transfer data and commands between these two modules;
- Transducer Services API: implemented in the NCAP, provides the interface between the applications running in the NCAP and the low-level commands defined in TIM;
- HTTP API: implemented in NCAP, uses the HTTP protocol and basically defines a communication message format for accessing the Transducer Services API.

The use of this standard for creating reconfigurable weblab infrastructures is motivated by its acceptance in industry and in the research community, and also by previous work made by authors, as described in the next section.

III. MOTIVATION AND PREVIOUS WORK

The acceptance of the IEEE1451.x Stds. is evident as proved by the many solutions already developed by companies [9][10][11][12] and others by universities [13]. Additionally, recent research shows the interest on using the IEEE1451.0 Std., as proved by some publications that suggest adapting previous standards with it [14], contributions for implementing web services [15], and also recent developments using FPGA technology [16]. In addition, the adoption of FPGAs to accommodate I&M used by weblabs was already referred in [17], where two infrastructural solutions were presented, namely: i) SoC. approach: a webserver plus I&M are implemented inside the FPGA, and ii) hybrid approach, where I&M are implemented inside the FPGA, while the webserver is implemented by an external device. This last approach was seen as the preferable one, and an instrument, commonly used in electronic experiments named Function Generator (FG), was synthesized into an FPGA able to remotely control through a MicroWebserver from Lantronix [18]. However, this solution did not follow any standard, which created several problems during developments, since tasks were conducted by two different institutions, Instituto Superior de Engenharia do Porto Portugal (ISEP) and Herriot Watt University - Scotland (HW). ISEP, developed and implemented the infrastructure with the FG, while the web interface, that allows its remote control, was developed in HW. Since no standard protocol was adopted for controlling the FG, it was defined a specific one at ISEP, which posed some problems, since HW developers had to learn all protocol details. This delayed developments and alerted for problems that come up if no standard solution is adopted.

Therefore, taking into consideration the characteristics of the IEEE1451.0 Std., its acceptation by the industry, and the advantages of creating low-cost, standard and reconfigurable weblabs, lead us to considerate the adoption of the IEEE1451.0 Std., together with FPGA technology, as a possible solution for creating reconfigurable weblab infrastructures. These aspects lead us to start the implementation of an IEEE1451.0 infrastructure into an FPGA-based board able to accommodate several IEEE1451-compatible I&M that may be adopted, in the future, by any weblab infrastructure.

Next sections present the proposed architecture focusing on the TIM module currently under development. It also depicts the way users can, in the future, attach new I&M and remote control them using a thin architecture based on the HTTP API defined by the IEEE1451.0 Std..

IV. SOLUTION UNDER DEVELOPMENT

A. Architecture

Supported on the IEEE1451.0 Std. specification, it was selected an hybrid architecture composed by two modules: a TIM and a NCAP. As illustrated in figure 2, both are interconnected through a serial RS232 interface to send/receive messages, as defined by the standard. NCAP acts as a webserver implementing some services and allowing the remote interaction with the TIM. This way, users are able to send commands to interact with the TIM module, and receive replies and/or messages initiated by the TIM that may be caused by an external event (e.g. a change in a digital input sensor).

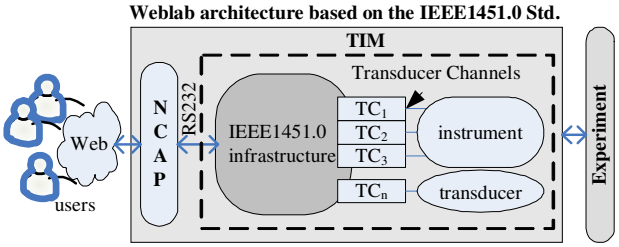

Fig. 2. Hybrid architecture implemented.

Currently, the NCAP module is being implemented in a small device named Arduino [19] that enables the control of a serial interface and an Ethernet bus for remote accessing its services. To satisfy throughput requirements of every I&M, that can form simple transducers or instruments, and to provide reconfiguration capability for the weblab infrastructure, the TIM module, namely the IEEE1451.0 infrastructure, transducers and/or IEEE1451.0-compatible instruments, are being implemented in an FPGA-based board, namely the Spartan-3E Starter Kit from Xilinx [20].

B. IEEE1451.0 infrastructure

This module is being described using a standard HDL named Verilog. This way, all internal modules are independent of the FPGA manufacturer since they can be synthesized to several platforms currently available, and to others that may appear in the future. As illustrated in figure 3, internally, the infrastructure comprehends a set of 4 generic modules, a configuration file, plus specific instruments or simple transducers (also described in Verilog) attached to several TCs:

1. TEDS controller - contains all TEDS and controls read, write and update TEDS operations;
2. UART - interfaces the TIM with the NCAP using an RS232 connection. It also implements a verification mechanism to guarantee that transferred data is in accordance with the message structures defined by the IEEE1451.0 Std.;
3. Status/state - controls the operating and trigger states, and read/write actions on the status registers of each TC and TIM;
4. Decoder/controller - controls all other modules, decodes instructions, handle errors, attends and interfaces TCs;
5. Configuration file - contains a set of initial values that describes the implemented infrastructure namely: number and length of adopted TEDS, transmission baud rate, number of implemented TCs, etc..
6. Instruments/transducers - are the instruments or transducers adopted by the weblab infrastructure.

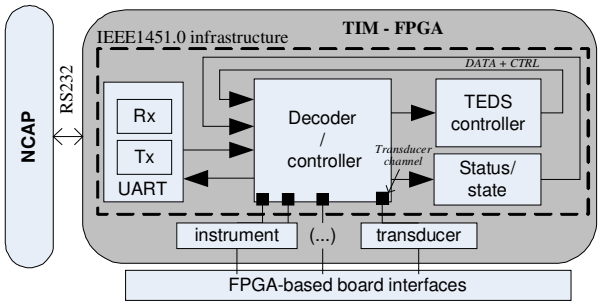

Fig. 3. Implemented HDL modules.

Presently, we are implementing a second version of the IEEE1451.0 infrastructure, since the first one required a lot of FPGA resources. This delayed developments, but at the same time gave us more expertise for future improvements. The TEDS, Status/state controllers and the UART are already finished, although not all instructions are implemented by the decoder/controller module. To validate the implemented architecture, two digital I/O transducers, namely an 8 digital input and a 4 digital output, were connected to the decoder/controller module through two independent TC. Only a few parameterizations in the IEEE1451.0 infrastructure were required: i) change some values in the text definition file (number of implemented TC, number of adopted TEDS, and number of status/state registers); ii) add, for each TC, internal tasks and a set of wires to connect the digital I/O transducers, and iii) define and connect the required TEDS within the TEDS controller.

As defined by the standard, each I/O is characterized upon TEDS contents. This way, besides the Meta-TEDS that describes general TIM features, including access and control information, for each I/O it was also defined two other TEDS: i) TC-TEDS (indicating all information regarding TCs operations) and ii) the user's transducer name TEDS (indicating the name of each transducer). These TEDS have specific fields required to be defined in order to characterize the behavior of the whole TIM and their TCs. Table I exemplifies two of those TEDS, namely the TC-TEDS, together with some information describing the fields. Some fields are omitted because either they are not required or they are optional.

All HDL modules were simulated using the Xilinx® ISim simulator from the ISE WebPACK [21]. To exemplify the

TIM operation, figure 4 presents the simulation of a *write status-event protocol* instruction, together with an event change in the digital input transducer. Since the *status-event protocol* (defined by the IEEE1451.0 Std.) was activated for the adopted TC, besides transmitting through Tx the data read from the digital input sensor, a simulated change in its inputs generated a TIM message with the status register contents.

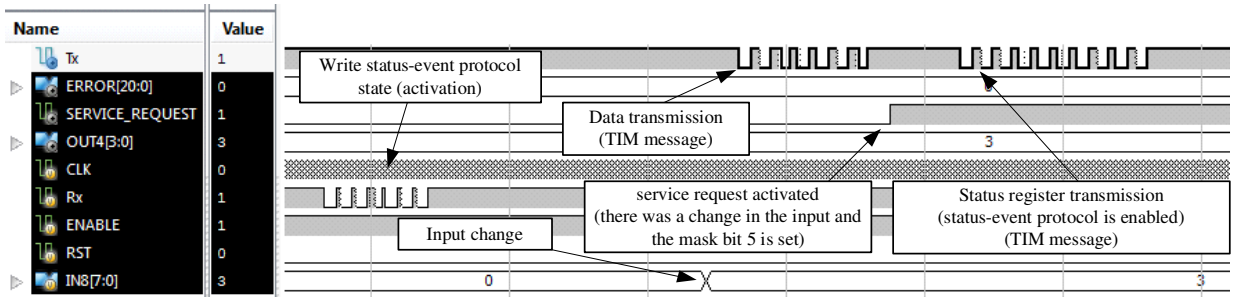

Fig. 4. Simulation of the *write status-event protocol state* command with a change in the digital input of the transducer.

TABLE I
INPUT AND OUTPUT TC-TEDS DATA BLOCK CONTENTS

Field type	Field name	Input (8 inputs)	Output (4 outputs)
	length	**00.00.00.43**	**00.00.00.49**
		Length of the TC-TEDS data block.	
3	TEDSID	**03.04.00.03.01.01**	
		TEDS identification.	
10	CalKey	**0A.01.00**	
		Calibration capability (no calibration, set to 0).	
11	ChanType	**0B.01.02**	**0B.01.01**
		Type: Event Sensor.	Type:Actuator.
12	PhyUnits	**0C.01.04**	
		SI units for the physical quantity (set to digital data)	
13	LowLimit	**0D.04.7F.FF.FF.FF**	
		Low range (not implemented, NaN = 7F.FF.FF.FF).	
14	High limit	**0E.04.7F.FF.FF.FF**	
		High range (not implemented, NaN = 7F.FF.FF.FF).	
15	Oerror	**0F.04.00.00.00.00**	
		Uncertainty in TC's output over the worst-case combination like environment and power supply variations (TCs are switches set to 0).	
16	SelfTestKey	**10.01.00**	
		Self test capability (no self test, set to 0).	
18	Sample	**12.09.28.01.04. 29.01.01.2A.01.08**	**12.09.28.01.04. 29.01.01.2A.01.04**
		Definition of data (28.01.04 - bit sequence, 29.01.01 - 8 bits length, 2A.01.08/04 - 8 or 4 significant bits).	
20	update time	**14.04.00.00.00.01**	
		Max. time between a trigger event and the latching of the first sample in a data set (it was set to 1 second).	
21	Wsetup	Omitted	**15.04.00.00.00.00**
		Minimum time between the end of a write data and the application of a trigger (required for actuators but not relevant for the implemented TC).	
23	Speriod	**17.04.32.AB.CC.77**	
		Minimum sampling period for read/write operations (depends on the adopted technology. it was set to 20ns because the FPGA has an 50MHz oscillator).	
24	WarmUp	**18.04.00.00.00.01**	
		Period of time in which a TC stabilizes its performance to predefined tolerances (set to 1 sec.).	
	checksum	**F3.32**	**F3.18**
		F3.32 = (FF.FF – 0C.CD) / F3.18 = (FF.FF – 0C.E7) hash sum to detect errors in the TEDS	

After simulating all implemented commands, the HDL code was synthesized using the Xilinx ISE Design Suite 12.2 from the ISE WebPACK [21] and a bitstream file was created to configure the FPGA-based board. For validating the control of the whole infrastructure, the FPGA-based board was connected to a serial communication tool named Comm Operator Pal [22] able to send/receive IEEE1451.0 low-level commands.

Figure 5 illustrates the way a TIM can be controlled by sending the low-level commands in hexadecimal format to the TC 1 attached to the digital input sensor. It exemplifies the test of two commands. The first is the *Read TEDS segment* of the TC-TEDS. This command comprehends a general structure with the TC number, command class and function, and the length of the associated data, namely the TEDS access code number 3 (representing the access to the TC-TEDS) and an offset representing the address relative to the beginning of TEDS at which the block of data shall be read. The reply to this command was a *reply error message*, since the appliance of this command requires the TC to be in a state named *operation*, which is not the case after an initialization (it is in an *idle state*). For enabling its correct appliance, another command, named *TransducerChannel Operate*, was sent to change the TC state from *idle* to *operation* state. Only after this transition the *Read TEDS segment* command was correctly applied. It was sent two times, the first with an offset of 1 octet and the second with an offset of 16 octets, which originates successful replies containing the TC-TEDS contents for both situations.

After concluding the implementation of all instructions in the decode/controller module, and defining a methodology for configuring the general IEEE1451-infrastrcuture, two issues will be considered for future developments: i) develop a set of IEEE1451.0-compatible instruments, and ii) develop the NCAP module by implementing a thin IEEE1451.0 architecture.

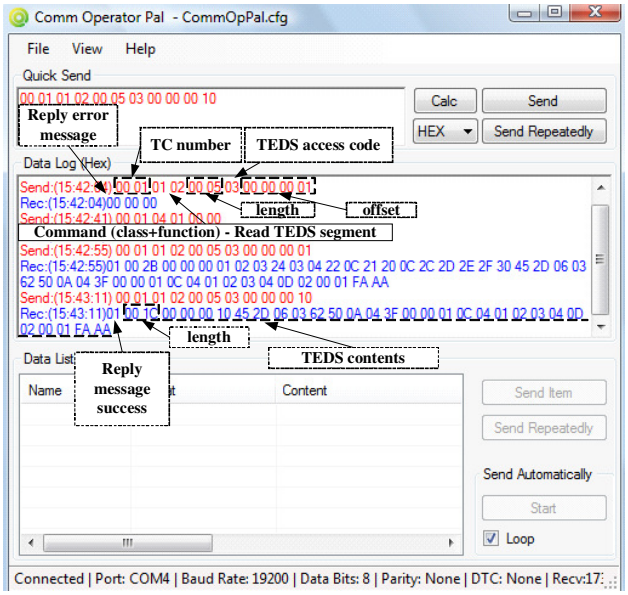

Fig. 5. Example of commands sent to the IEEE1451.0 infrastructure.

C. *IEEE1451.0-compatible instruments*

Every instrument should be implemented by one or more synthesizable HDL modules, and developed taking into consideration the implemented IEEE1451.0 infrastructure, namely the decoder/controller module. The digital I/O transducers are controlled using independent TC through low-level IEEE1451.0-commands. If a more complex instrument is attached, several TC may be required to access the instrument. They should be defined as a group in the Meta-TEDS, and specified by individual TC-TEDS to provide information regarding their characteristics. For example, the implementation of a FG, similar to the one referred in section IV, requires 3 grouped TC for controlling 3 variables: 1) the waveform type, 2) the time scale and 3) the scale value.

Furthermore, reusing the IEEE1451.0 infrastructure by different instruments requires the specification of a common interface between the infrastructure and each instrument. This solution will be considered in future specifications, since it will facilitate the reconfiguration of weblab infrastructures. This way, different institutions may develop their own instruments and interface them with the general IEEE1451.0 infrastructure. The instruments will be easily sharable and reconfigured in FPGA-based boards, and they may be accessed with standard IEEE1451 instructions, increasing collaboration between institutions for creating weblab infrastructures.

D. *Remote access*

The remote access to the IEEE1451.0 infrastructure is made through the NCAP module that is being implemented in an Arduino device. Besides a set of services that it must provide (e.g. monitoring instruments' operation, controlling trigger capabilities, etc.), it was decided to simplify the IEEE1451 architecture by mapping the low-level commands with the HTTP API functions, i.e. suppressing the use of the module communication and services APIs. This decision was made taking into consideration the main objective of creating a weblab infrastructure able to be accessed using standard functions, which is already fulfilled by the HTTP API. Furthermore, the suggested architecture focus on a single TIM attached to a single NCAP, i.e. not requiring any network communication methods, only a single point-to-point connection. This requirement may suggest the use of the IEEE1451.2 Std. (point-to-point communication between NCAP - TIM). However this standard is not yet compatible with the IEEE1451.0 Std., despite some suggestions [14], and it mainly defines time constrains and interface signals that can be easily handled using a simplified RS232 connection. Therefore, after analyzing the HTTP API functions, and the low-level commands, we are currently establishing a cross map, which suggests the use of a thin architecture for implementing weblab infrastructures.

V. CONCLUSIONS

The adoption of the IEEE1451.0 Std. and FPGAs may be a solution to create reconfigurable weblab infrastructures. The capacity of reconfiguring FPGAs with several modules described in standard HDL, allows instruments and transducers to be easily shared. However, a standardized solution is required so those instruments can be interfaced and accessed. Supported on this request, the IEEE1451.0 Std. was seen as a possible solution, since it describes a set of layers that allow network-interfacing transducers, which can form most of the instrumentation required in every weblab infrastructure. Two main advantages may be pointed out when using this approach: i) better control, characterization and easy interface among all transducers/instruments using network resources and standardized functions; and ii) easy developments, since the std. provides several layers, namely commands and APIs that facilitate a task division during developments, increasing collaboration among institutions.

A solution is currently under development and it focus in two main issues: i) create a general synthesizable IEEE1451.0-infrastrucure able to interface IEEE1451.0-compatible instruments, and ii) enable that those same instruments can be easily attached to the infrastructure and accessed using standard HTTP functions.

REFERENCES

[1] LiLa - Library of Labs [2009 - 2011], *http://www.lila-project.org/content/index.html*, 2011. [Online]. [Accessed: 21-Feb-2011].

[2] J. García-Zubia et al., "Towards a Distributed Architecture for Remote Laboratories," *International Journal of Online Engineering (iJOE)*, vol. 4, 2008.

[3] J. García-Zubía et al., "Towards a canonical software architecture for multi-device WebLabs," *Industrial Electronics Society IECON 2005. 31st Annual Conference of IEEE*, p. 6, 2005.

[4] V. J. Harward et al., "The iLab Shared Architecture: AWeb Services Infrastructure to Build Communities of Internet Accessible

Laboratories," *Proceedings of the IEEE*, vol. 96, no. 6, p. 20, Jun. 2008.
[5] I. Gustavsson et al., "The VISIR project – an Open Sources Software Initiative for Distributed Online Laboratories," *Conference on Remote Engineering and Virtual Instrumentation (REV'07)*, p. 6, Porto - Portugal. 2007.
[6] I. Gustavsson et al., "A Flexible Electronics Laboratory with Local and Remote Workbenches in a Grid," *International Journal of Online Engineering (iJOE)*, vol. 2, no. 3, 2008.
[7] Salaheddin Odeh, "Building Reusable Remote Labs with Adaptable Client User-Interfaces," *Journal of Computer Science and Technology*, vol. 25, no. 5, pp. 999-1015, Sep. 2010.
[8] IEEE 1451.0 Std., "Standard for a Smart Transducer Interface for Sensors and Actuators - Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats," *The Institute of Electrical and Electronics Engineers, Inc.*, p. 335, 2007.
[9] Esensors - Esensors specializes in designing and producing high quality networked sensors, sensor data acquisition, and toxic gas monitors., [Online]. Available: http://www.eesensors.com/ieee1451.html. [Accessed: 25-Jan-2011].
[10] Smart Sensor Systems - Develops products and services to help customers make physical measurements economically using IEEE 1451- compliant smart sensors., [Online]. Available: http://www.smartsensorsystems.com/. [Accessed: 25-Jan-2011].
[11] SENIT - Technology Solutions, [Online]. Available: http://www.senit.biz/. [Accessed: 25-Jan-2011].
[12] National Instruments - Sensors Plug&Play with Transducer Electronic Data Sheets (TEDS), [Online]. Available: http://www.ni.com/teds/. [Accessed: 25-Jan-2011].
[13] Huang Qi et al., "Design and Implementation of a Power System Sensor Network for a Wide-Area Measurement," *2008 International Symposium on Computer Science and Computational Technology Shanghai, China*, pp. 796-799, Dec. 2008.
[14] H. Ramos, "A contribution to the IEEE STD. 1451.2-1997 revision and update," presented at the AFRICON 2007, pp. 1-7, 2007.
[15] Eugene S. and Kang L., "Sensor Network based on IEEE 1451.0 and IEEE p1451.2-RS232," *IEEE Instrumentation and Measurement Technology Conference Proceedings*, pp. 1728-1733, 2008.
[16] J. Kamala and B. Umamaheswari, "IEEE 1451.0-2007 Compatible Smart Sensor Readout with Error Compensation Using FPGA," *Sensors & Transducers Journal*, vol. 102, no. 3, Mar. 2009.
[17] R. J. Costa et al., "FPGA-based Weblab Infrastructures - Guidelines and a prototype implementation example," *3rd IEEE International Conference on e-Learning in Industrial Electronics (ICELIE'2009), Porto - Portugal*, p. 7, Nov. 2009.
[18] Lantronix, 2011. [Online]. Available: http://www.lantronix.com/device-networking/embedded-device-servers/. [Accessed: 23-Feb-2011].
[19] Arduino, [Online]. Available: http://www.arduino.cc/. [Accessed: 21-Feb-2011].
[20] Spartan-3E Xilinx Starter Kit, [Online]. Available: http://www.xilinx.com/products/devkits/HW-SPAR3E-SK-US-G.htm. [Accessed: 23-Feb-2011].
[21] ISE WebPACK Design Software, [Online]. Available: http://www.xilinx.com/tools/webpack.htm. [Accessed: 23-Feb-2011].
[22] Comm Operator Pal, [Online]. Available: http://www.serialporttool.com/CommPalInfo.htm. [Accessed: 21-Feb-2011].