Extending the IEEE1451.0 Std. to serve distributed weblab architectures

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 - **The appliance of the IEEE1451.0 Std. into the remote experimentation domain may be an interesting solution not only to develop reconfigurable weblab infrastructures, but also to improve the way infrastructures, and their experiments, may be shared. Therefore, this paper proposes a distributed weblab architecture supported on a IEEE1451 concept named Transducer Electronic Data Sheet (TEDS). It is suggested the use of a new TEDS, named LabTEDS, to provide information about weblab infrastructures namely, web location, technical resources and type of experiments described according a metadata model specification defined by the Lab2go project. The access to the architecture is made through the IEEE1451.0 HTTP API extended with new functions. At the end of the paper a thin implementation of the architecture is presented, supported on a cross-mapping established between the HTTP functions and the low-level commands, which are used to control the weblabs.**

Keywords - Weblabs, Weblab architectures, IEEE1451.0 Std., Remote Experimentation.

I. INTRODUCTION

Changes in education were promoted by the appearance of new technological resources. One of those resources is the Internet and associated services that allow remotely accessing educational contents like real experiments, which are fundamental for fulfilling the learning objectives in engineering education [1][2]. In the last 2 decades, several institutions have being creating weblab infrastructures providing remote access to experiments that were previously only accessible using traditional laboratories, e.g. [3]. However, those infrastructures are generally developed following different architectures, which difficult their development and sharing because they use different Application Programming Interfaces (APIs) to access their features and, in many situations, they are not able to be easily found in the World Wide Web. Many experts are aware of these limitations, which led to the creation of the Global Online Laboratory Consortium (GOLC) [4] that, among other objectives, focus on researching ways to improve collaboration among institutions, and to facilitate users' (students & teachers) access to every weblab by using standards. Therefore, this paper suggests the use of the IEEE1451.0 Std. to serve distributed weblab architectures. Supported on its characteristics, that focus on implementing and network-interfacing transducers (sensors and actuators), it is proposed some extensions to facilitate the dissemination and the access to weblab infrastructures and experiments they may handle. So readers may understand the proposed IEEE1451.0 Std. extensions, section II provides a brief overview of the standard, focusing on Transducer Electronic Data Sheets (TEDSs) that specifie technical characteristics of transducers, and on functions provided by the IEEE1451.0 HTTP API, that allow remotely accessing those same transducers. Section III describes the proposed architecture, and section IV presents a new TEDS, named LabTEDS, to be used for describing weblab infrastructures. The operational sequence for accessing weblab infrastructures is specified in section V, with a detailed description of the registering, discovering and accessing processes. Before concluding, section VI presents a thin implementation of the IEEE1451.0-architecture, that focus on interfacing the IEEE1451.0 HTTP functions to a set of low-level commands for simplifying developments and accesses to weblab infrastructures.

II. IEEE1451.0 STD. OVERVIEW

Defined in 2007, the IEEE1451.0 Std. [5] aims to network-interface transducers and defines a set of operating modes, based on specifications provided by TEDSs. It is the basis for forthcoming and previous members of the IEEE1451.x family, so they can operate together according to operating modes controlled through low-level commands that can be applied using a set of APIs. The standard defines an architecture based on two modules: the Transducer Interface Module (TIM) that controls Transducer Channels (TCs), and the Network Capable Application Processor (NCAP) that provides network access to those TCs. 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).

To control the behaviour of a transducer, the IEEE1451.0 Std. provides low-level commands to read/write TEDSs, control transducers, etc., and allows manufacturers to define their own commands. To monitor the operations of a transducer, the IEEE1451.0 Std. implements a 32 bit status register for each TC and another for the TIM. To access (remotely or not) a specific transducer 3 APIs are specified:

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

Figure 1 illustrates the IEEE1451.0 Std. main modules and associated layers, also providing an indication of the chapters where they are described by the standard.

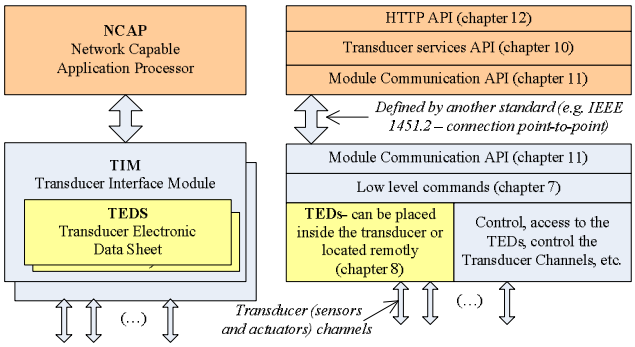

Figure 1. Main modules an layers of the IEEE1451.0 Std..

The proposed architecture suggests extending the IEEE1451.0 Std. with a new TEDS named LabTEDS, the addition of IEEE1451.0 HTTP functions, and presents an alternative thin implementation for accessing weblabs. Therefore, before describing these suggestions, next subsections present the TEDSs and the IEEE1451.0 HTTP API with its data structures for better understanding the proposed extensions.

1) *TEDSs*

TEDSs are pieces of information that describe the structure and behaviour of the entire TIM module and of each TC. The standard describes 15 TEDSs, some required and others optional, all identified by an access code. They are usually placed inside the TIM but they can also be remotely located (named virtual TEDS). The information within each TEDS may either be binary or a text-based XML data format using a structure named TLV (Type/Length/Value), which is defined in the standard according to each TEDS. As illustrated in figure 2, TEDSs are divided in several octets. The first 4 octets indicate the length, and the remaining provide the data block and the checksum with the one's complement of the sum of all previous octets.

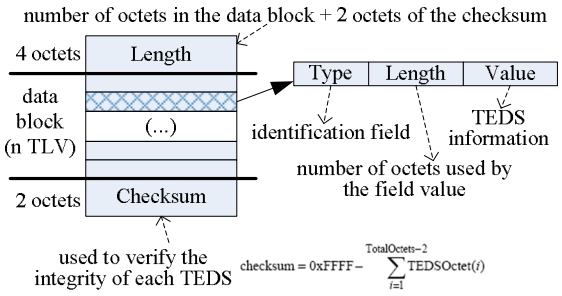

Figure 2. Generic TEDS format.

2) *HTTP API*

The IEEE1451.0 Std. provides an API with a set of HTTP functions for accessing through the web a specific transducer. All functions are handled by the NCAP module that receives requests and provides responses using the HTTP protocol. The NCAP works as a common webserver that communicates with all clients according to common client-server architectures. Each IEEE1451.0 HTTP function, described in table I, belongs to a specific interface, namely:

- Discovery: Discover IEEE1451.x communications modules, TIMs and TCs;
- TransducerAccess: Read and write TCs;
- TEDSManager: Read and write TEDSs and manage NCAP-side TEDS cached information.
- TransducerManager: Provides control functions over TIM accesses, e.g., to lock the TIM for exclusive use and to send arbitrary commands to it.

All functions are sent to the NCAP using an HTTP message format (*http://<host>:<port>/<path>?<parameters>*), and replies can either be in XML, HTML or text formats.

TABLE I. IEEE1451.0 HTTP API.

API	Function	Path
Discovery	TIM Discovery	1451/Discovery/TIMDiscovery
	Transducer Discovery	1451/Discovery/TransducerDiscovery
Transducer Access	ReadData	1451/TransducerAccess/ReadData
	StartReadData	1451/TransducerAccess/StartReadData
	Measurement Update	1451/TransducerAccess/MeasurementUpdate
	WriteData	1451/TransducerAccess/WriteData
	StartWriteData	1451/TransducerAccess/StartWriteData
TEDS Manager	ReadTeds	1451/TEDSManager/ReadTeds
	ReadRawTeds	1451/TEDSManager/ReadRawTeds
	WriteTeds	1451/TEDSManager/WriteTeds
	WriteRawTeds	1451/TEDSManager/WriteRawTeds
	UpdateTeds Cache	1451/TEDSManager/UpdateTedsCache
Transducer Manager	SendCommand	1451/TransducerManager/SendCommand
	StartCommand	1451/TransducerManager/StartCommand
	Command Complete	1451/TransducerManager/CommandComplete
	Trigger	1451/TransducerManager/Trigger
	StartTrigger	1451/TransducerManager/StartTrigger

Input and output parameters are defined according to each function, and use data types specified in the Args module described in chapter 9 of the standard.

III. ARCHITECTURE

The proposed architecture seeks to spread and share weblab infrastructures and experiments. It intends to be used for accommodating weblabs created according to the IEEE1451.0 Std. and to be integrated with other solutions already available like the: iLABs architecture [6] for users' management and the Lila portal to accommodate pedagogical contents [7]. Furthermore, it adopts the metadata model specification named

Online Laboratory Metadata - Reference Model Specification described by the Lab2go project [8][9].

The architecture gathers information into a LabServer regarding the available weblab infrastructures and experiments they may handle. This server may be located anywhere and it should operate as a central provider for all weblabs developed according the IEEE1451.0 Std.. As illustrated in figure 3, the architecture allows remote accessing Units Under Test (UUT), which form the experiments, through weblab infrastructures that may integrate simple TIM-NCAP modules or structured ones that use more than one TIM. These infrastructures are connected through the Internet to one or more LabServers that provide the Internet location of each weblab infrastructure. These same infrastructures also provide other relevant information about their technical characteristics and the experiments they may handle. When a specific weblab is implemented through FPGA-based boards and is able to be reconfigured according to the idea presented in [10], the LabServer may also provide Instruments and Modules (I&M) described in Hardware Description Languages (HDL), and associated web interfaces that may be used to remotely control and reconfigure weblab infrastructures. This reconfiguration depends on the technical characteristics of the weblab infrastructure (e.g. processing power, interface ports, etc.) that need to be provided, so users may decide if the infrastructure is appropriated to accommodate the I&M required for running a specific experiment.

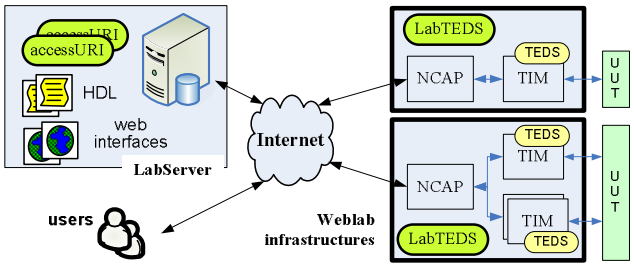

Figure 3. Distributed weblab architecture based on the IEEE1451.0 Std..

As presented in the previous sections, the IEEE1451.0 Std. specifies a set of TEDSs that may be used to define the behaviour of an adopted I&M in a specific weblab infrastructure. However, all specified TEDSs do not consider a centralized approach that would allow: i) users to find weblab infrastructures and experiments, and ii) institutions to create a network with several weblab infrastructures. Therefore, for implementing the architecture, it is suggested the use of a new TEDS, named LabTEDS, that provides information about the weblab infrastructure, namely the web location specified through an Unified Resource Identifier (URI), a description of the available experiments, technical features of a specific infrastructure, among other information. Each weblab infrastructure should have one LabTEDS located at the NCAP side. During a process called registration, each URI defined in the LabTEDS will be send to the LabServer, and an application will read all LabTEDS and provide information for users, so they can choose one weblab infrastructure or experiment to access to, discovering it using the IEEE1451.0 HTTP API. After this process, users can access the weblab infrastructures and/or the associated experiment(s), and all data transferred during the conduction of a specific experiment between a weblab and users can be automatically monitored for students' assessment purposes.

Next sections present the LabTEDS and the operational sequence for registering, discovering and accessing a specific weblab infrastructure and/or experiment, using new IEEE1451.0 HTTP API functions and interfaces, namely:

- NCAPRegister, to register or unregister NCAPs (new Register API interface);
- NCAPDiscovery, to discovery NCAPs (Discovery API interface);
- ReadLabTeds and WriteLabTeds to read and write LabTEDS (TEDS manager API interface);
- ReadTIM and WriteTIM to reconfigure weblab infrastructures (new Reconfiguration API interface) and;
- ReadLog and WriteLog to read and write a log file for assessment purposes (new Log access API interface).

IV. LABTEDS

Following the same structure defined for all TEDSs, LabTEDS seeks to establish a standardized way to: i) disseminate and share weblab infrastructures and experiments, and ii) specify all infrastructural resources (e.g. power processing capabilities, memory space, etc.). The main idea is to use and provide information about those infrastructures and experiments using new fields and a set of fields defined in a text-based TEDS format to accommodate the metadata model specification illustrated in figure 4, that corresponds to the Reference Model Specification described by the Lab2go project [8][9].

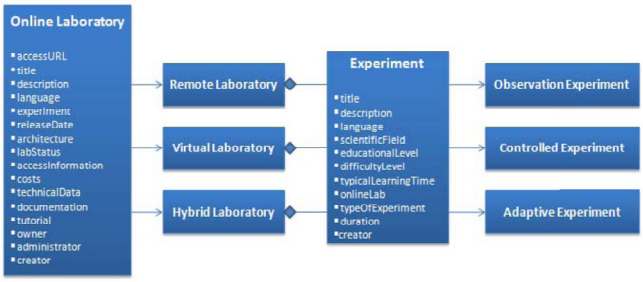

Figure 4. Lab2go Metadata - Reference Model Specification.

This way, all weblab infrastructures and/or experiments can be developed according to the IEEE1451.0 Std. and described using a TEDS. Table II presents the LabTEDS structure and a detailed description of each field. Only new fields are described, since the others are in accordance with the current IEEE1451.0 Std..

- Field 3 (TEDSID) [required] - TEDS Identification Header: Uses the same format specified for all other TEDSs and the access code is the number 16.

- Field 10 (numLabs) [optional] - Number of weblabs: This field seeks to group all weblab infrastructures required for implementing a specific experiment. When a specific experiment requires the use of more than one weblab infrastructure, this field must indicate that requirement. The remaining fields should provide the web location and technical resources of each weblab infrastructure and the fields grouped in the related information should specify the experiment. In other words, if an experiment needs two weblab infrastructures, eventually located in different places, it should be indicated by filling in the number 2 in this field. In this case, the remaining LabTEDS fields should specify both weblabs infrastructures and the experiment(s) they handle, and should also be located in both NCAP modules. If this field is omitted, users must consider that the current weblab infrastructure does not belong to any group since it contains all resources to run the experiment(s) detailed in the remaining fields.

TABLE II. LABTEDS FIELDS.

Field Type / Name	Description	Data type	#octets
-	TEDS length	UInt32	4
0-2/-	Reserved	-	-
3/TEDSID	TEDS Identification Header	UInt8	4
4-9/-	Reserved		
10/numLabs	Number of weblabs	Uint8	1
Weblab infrastructure related information (repeated for each weblab)			
Web Location			
11/accessURI	Weblab URI [IP addr. (first 4 octets) + port num (last octet)]	Uint8	5
12/logURI	Log file URI [IP addr. (first 4 octets) + port num (last octet)]	Uint8	5
Technical resources			
13/implType	Implementation type (thin ≠ 0 (true), standard = 0 (false))	Boolean	1
14/numTIMs	Number of TIMs connected to the NCAP module	Uint8	1
Related information (should be repeated for each supported language)			
15/numLang	The number N of different language blocks in this TEDS	UInt8	1
16/dirBlock	Language block description. This block is repeated N times	-	-
20/langCode	Language code from ISO 639	UInt8	2
21/offset	Language offset	UInt32	4
22/length	Language length = LL	UInt32	4
23/compression	Enumeration identifying the compression technique used	UInt8	1
17/subSum	Nondisplayable data checksum	UInt16	2
-/XMLText	XML-based text block (semantics of the Lab2go project)	text	LL-2
-/XMLSum	Text block checksum	UInt16	2
18-19/24-127	Reserved		
128–255/	Open to manufacturers	-	-
-	Checksum	UInt16	2

- Field 11 (accessURI) [required] - Location of the weblab infrastructure: Represents the IP address and port number of a specific Weblab infrastructure. If the accessURL element of the Lab2go metadata model specification is defined in the XML-based text block, this field should have the same value.
- Field 12 (logURI) [optional] - Location of the log file: Represents the IP address and port number of the log file used to log all data transferred between users and the weblab infrastructure. If this field does not exist, it means the current weblab does not implement the logging process.
- Field 13 (implType) [required] - Implementation type: Specifies if the weblab infrastructure followed a thin (≠ 0 (true)), or standard (= 0 (false)) implementation (these aspects are further referred in section VI of this paper).
- Field 14 (NumTIMs) [required] - Number of TIM modules connected to the NCAP module: Indicates the number of TIM modules that comprehend the weblab infrastructure. This is a required field since some weblab infrastructures may use more than one resource (e.g. more than one FPGA or more than one PC). Technical data of each TIM should be provided in text format through the remaining fields using the metadata model specification described by the Lab2go project. Is up to the developer to describe the technical data according to each TIM specification.
- Fields 15, 16 and 17 [optional]: gather data information according to the Lab2go metadata model specification following the IEEE1451.0 Std. text-based TEDS. Information presented in this model that was already defined in previous fields, e.g. the accessURI, should be repeated.

V. OPERATIONAL SEQUENCE

The operational sequence for accessing the weblab architecture includes three processes, as illustrated in figure 5:

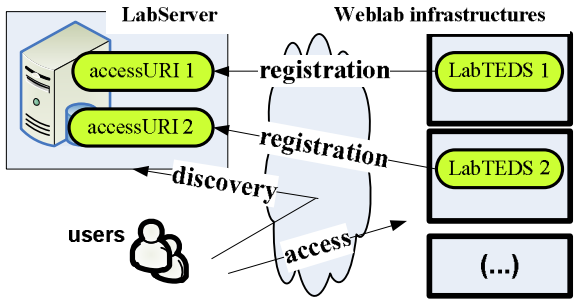

Figure 5. Operational sequence for accessing weblab infrastructures.

- Registration: When a specific weblab infrastructure is created using the IEEE1451.0 Std., it should be registered in the LabServer, placed in a known web location. For this purpose, the IP addresses and port numbers included in the accessURI field of each LabTEDS will be copied into the LabServer, registering the weblab infrastructure in the network.
- Discovery: Users initiate a discovery process to find the appropriate weblab they want to use, requesting a list of weblabs and experiments available (the ones that were already registered in the LabServer). This list may be dynamically created by the use of the NCAPDiscovery function. As described in the following subsections, the NCAPDiscovery mainly gets the URI of all registered weblab infrastructures. The response of this function should

be used to access each LabTEDS through the readLabTEDS function to create, for example, a webpage describing weblabs and experiments they may handle. Using this page, users may choose the appropriate weblab infrastructure and/or experiments they want to access to.

- Access: Once connected, users should be able to access weblab infrastructures using the functions available in the IEEE1451.0 HTTP API. Three sub processes are provided to: i) control experiments; ii) reconfigure weblab infrastructures with I&M (when they provide that feature); and iii) monitor all data transferred between users and weblab infrastructures. This last sub process is relevant for assessment purposes, since all data transferred between users and weblab infrastructures follow standardize functions that can be logged into a file for future teachers' analysis or for automatic analysis made by intelligent tutoring systems [11].

Below, these processes are detailed according the new proposed IEEE1451.0 HTTP functions.

1) Registration

The registration process should be automatically made after connecting a weblab infrastructure to the Internet. It uses the NCAPRegistration function to send the URI to the LabServer, so it can create a map table with all registered weblabs. Is up to the LabServer to periodically query if all weblabs are still running, for example, using a ping command to check if the destination IP address is available. If the response to this command indicates the inexistence of the target IP, it means that the correspondent weblab infrastructure is not available anymore, and its URI should be deleted from the map table, unregistering the weblab infrastructure. A weblab infrastructure may also unregister itself without being disconnected from the Internet, using the same NCAPRegistration function. Figure 6 illustrates the register/unregister process.

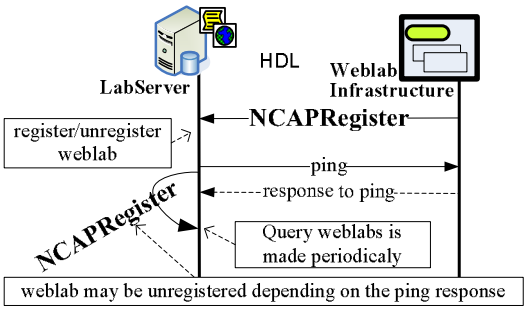

Figure 6. Process of register/unregister weblab infrastructures.

2) Discovery

For the discovery process the NCAPDiscovery and ReadLabTeds functions should be used. The NCAPDiscovery follows a similar approach provided by the TIMDiscovery and TransducerDiscovery functions, and intends to discover the location of all registered NCAP (or weblabs infrastructures) by retrieving their IP addresses and port numbers provided by each accessURI field of a LabTEDS.

The ReadLabTeds follows a similar approach provided by the functions that belong to the TEDS Manager API type, namely the ReadRawTeds and ReadTeds. The IEEE1451.0 std. adopted these two functions because it suggests caching the same TEDS available in TIM inside the NCAP module, to improve speed and security by implementing redundant information. ReadRawTeds reads a TEDS available in TIM modules, and ReadTeds may read this last TEDS only if it is not available in the NCAP module. However, this situation is not considered for reading the proposed LabTEDS, since it should be always implemented in the NCAP module or remotely located. No redundant information is required for implementing the architecture, despite developers may implement a mechanism to replicate LabTEDS in more than one location.

As illustrated in figure 7, the use of these new functions (NCAPDiscovery and ReadLabTeds) can be done by an application running in the user side or in the LabServer side.

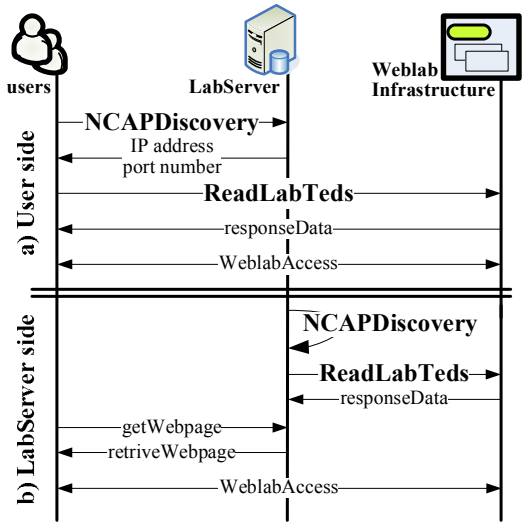

Figure 7. Using NCAPDiscovery and ReadLabTeds functions to access registered weblabs infrastructures.

In both situations, the NCAPDiscovery function should retrieve the accessURI of a specific weblab infrastructure. Using that location, the ReadLabTeds is able to read all information within each LabTEDS, so it can be used to create, for example, a webpage listing all available weblab infrastructures and experiments they may provide.

In the first solution (user side) users start sending the NCAPDiscovery to get an array with all accessURIs of all registered weblab infrastructures. Based on this information, users can read all features of each weblab infrastructures and/or experiments they handle, using the ReadLabTeds. Supported on the retrieved information from the LabTEDS, users may create a list of all available weblab infrastructures and/or experiments.

In the second solution (LabServer side) both functions, NCAPDiscovery and ReadLabTeds, are applied by the LabServer itself. An application should be constantly using the NCADiscovery function to get the location of all registered weblab infrastructures. Using the accessURIs retrieved from that function, the LabServer should consult each LabTEDS using the ReadLabTeds, in the same way as described for the first solution. Unlike the first solution, where all processing is made in the users' side, this second solution requires a specific application inside the LabServer to handle the information retrieved from each weblab infrastructure or experiment.

3) Access process (reconfiguration and logging)

The access process is divided into three sub processes: i) control, which allows users to interact with weblab infrastructures, i.e. the experiments, ii) reconfiguration and iii) logging. While the first sub process is already covered by current functions provided by the IEEE1451.0 HTTP API, the other two (reconfiguration and logging) require some new functions, as detailed in the following topics.

a) Reconfiguration

The idea already described in [10], suggests the use of FPGA-based boards and the IEEE1451.0 Std. to accommodate different I&M developed according the IEEE1451.0 Std.. These weblabs adopt the same platform to accommodate several I&M able to be easily shared by several experiments that may (or may not) run in different institutions. Developing specific I&M, using HDL, allows easily sharing them by simple downloading them from the LabServer to reconfigure the FPGA-based board. Besides each I&M described in HDL, the LabServer should also provide the interfaces that allow their remote control.

Therefore, to remote reconfigure similar reconfigurable weblab infrastructures, even if they were created using other technology than FPGAs, new IEEE1451.0 HTTP API functions are suggested, namely the ReadTIM and WriteTIM. These new functions should be handled by the NCAP module and their usage depends on the technological architecture of each TIM. While the ReadTIM may be used without a preceding function, the WriteTIM should only be applied after reading the technical characteristics of the target TIM. It will be necessary to use the ReadLabTeds to get the text field tag named 'technical data' of the metadata defined in the XML text field, so users can evaluate if the TIM is capable of accommodating a specific I&M. Figure 8 illustrates the sequence for using the WriteTIM and ReadTIM functions.

b) Logging

The logging process intends to keep tracking of all actions made by users when they interact with a specific weblab infrastructure. The objective is to provide a mechanism for assessment purposes, so teachers may consult a specific log file to evaluate students' behaviour during the conduction of an experiment. Field 12 (logURI) in LabTeds indicates if a specific weblab infrastructure has the logging activated, by indicating the URI location of the log file. When active (i.e. the logURI is defined), all data used to access the weblab will be logged into that file that can be located in the NCAP or remotely located, e.g. in the LabServer. The file, e.g. a database table, should keep tracking of all data exchanged between the weblab and the users' device used to access the weblab infrastructure, namely: the title of the experiment (expTitle), user's identification (userID), a date indicating when a specific action was applied (date), and the standardize functions (function) with associated parameters (data). This file should gather all data according to the XML schema format presented in table III.

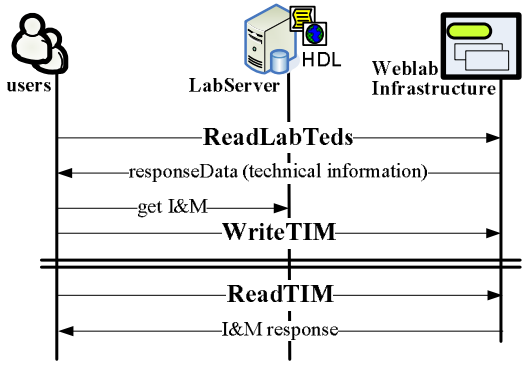

Figure 8. Using WriteTIM and ReadTIM for reconfiguring weblab infrastructures.

TABLE III. LOG FILE XML SCHEMA CONTENTS FORMAT.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
xmlns:stml=http://grouper.ieee.org/groups/1451/0/1451HTTPAPI
<xs:element name="LogFile">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="expTitle"      type=" stml: _String"/>
   <xs:element name="userID"        type=" stml: _String"/>
   <xs:element name="date"          type=" stml: timeInstance "/>
   <xs:element name="function"      type=" stml: _String "/>
   <xs:element name="data"          type=" stml: StringArray "/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

The WriteLabTeds function should be used to activate the logging session by writing the URI location of the log file in field 12 of LabTEDS. Once defined, teachers may read all actions performed by students, that are described in the log file, using a function named ReadLogFile. To clean or update that same log file (i.e. change its contents) users may use a function name WriteLogFile. Both functions (WriteLogFile and ReadLogFile) are new suggestions for the IEEE1451.0 HTTP API. It is up to the developer to establish some constrains on the use of these functions, since, in most situations, both should only be accessible for teachers.

Before concluding this paper, next section suggests the use of a thin implementation of the IEEE1451.0-architecture that cross-maps the IEEE1451.0 HTTP functions and the low-level commands used to control TIM modules.

VI. THIN IMPLEMENTATION

As described by the IEEE1451.0 Std., NCAP and TIM modules should be connected through specific protocols (e.g.

Bluetooth) following another IEEE1451.x Std.. Despite the IEEE1451.0 Std. intends to form the basis for future and previous IEEE1451.x Stds., some of those are not yet compatible. Furthermore, the NCAP-TIM connection requires using two additional APIs (transducer services and module communication) that will overload developments not bringing any added value for weblab infrastructures that use single NCAP-TIM connections. Therefore, it is proposed a thin implementation of the architecture that suggests cross-mapping the IEEE1451.0 HTTP API functions with the low-level commands used to control TIM modules. This is an alternative solution that should be indicated in the field 13 (implType) of each LabTEDS to simplify developments and to avoid overloading the adopted devices with additional computational tasks. After analyzing the current and the suggested IEEE1451.0 HTTP functions, it was noticed that not all functions were required to be cross-mapped with low-level commands. Table IV presents the established cross-map, and provides an indication about the section where each function and low-level command is described by the IEEE1451.0 Std..

TABLE IV. CROSS-MAP FUNCTIONS WITH LOW-LEVEL COMMANDS.

	Low-level Commands
Registration API	No map
NCAPRegistration (new)	
Discovery API	No map
NCAPDiscovery (new), TIMDiscovery (sec. 12.2.1) and TransducerDiscovery (sec. 12.2.1.4)	
Transducer Access API	
ReadData, StartReadData, MeasurementUpdate (sec. 12.3.1.1/2/3)	Sampling mode (sec. 7.1.2.4) and Read Transducer Channel data-set segment (sec. 7.1.3.1)
WriteData (sec. 12.3.2.1)	Sampling mode (sec. 7.1.2.4) and Write Transducer Channel data-set segment (sec. 7.1.3.2)
StartWriteData (sec. 12.3.2.2)	Write Transducer Channel data-set segment (sec. 7.1.3.2)
TEDS Manager API:	
ReadTeds, ReadRawTeds, UpdateTedsCache (sec. 12.4.1/2/5)	Reads TEDS segment (sec. 7.1.1.2)
WriteTeds, WriteRawTeds (sec. 12.4.3./4)	Write TEDS segment (sec. 7.1.1.3)
ReadLabTeds,WriteLabTeds (new)	No map
Transducer Manager API:	
SendCommand and StartCommand and CommandComplete (sec. 12.5.1/2/3)	may apply any command
Trigger or StartTrigger (sec. 12.5.4/5)	Read TEDS segment (sec. 7.1.1.2) and Sampling Mode (sec. 7.1.2.4) and Trigger Command (sec. 7.1.3.3)
Reconfiguration API	No map
WriteTIM or ReadTIM (new)	
Log access API	No map
WriteLog or ReadLog (new)	

VII. CONCLUSIONS

The IEEE1451.0 Std. was considered as a possible solution for implementing standard weblab architectures so they may be easily developed and shared. For providing and managing accesses to weblab infrastructures, it was suggested some extensions to the IEEE1451.0 Std., namely the use of a new TEDS to describe weblab infrastructures, named LabTEDS, and new IEEE1451.0 HTTP functions, to handle the processes of registering, discovering and accessing weblab infrastructures and experiments they may handle. It was also suggested a simplified architecture for implementing weblab infrastructures, that cross-maps IEEE1451.0 HTTP functions and low-level commands described by the standard. Despite the detailed description presented in this paper, some considerations should also be analyzed in the future, namely the way the architecture can be integrated with others that already implement scalable and secure access mechanisms. Therefore, it is our intention to implement and adapt the architecture with other solutions, and test it in a real scenario to prove the feasibility of using the IEEE1451.0 Std. with the suggested extensions, to create standardized and easy accessible weblab infrastructures.

VIII. REFERENCES

[1] Lyle D. Feisel and Albert J. Rosa, "A Colloquy on Learning Objectives For Engineering Education Laboratories," *American Society for Engineering Education Annual Conference*, 2002.

[2] Lyle D. Feisel and George D. Peterson, "Learning objectives for engineering education labs," *32nd Annual Frontiers in Education (FIE'02) Boston, MA*, Nov. 2002.

[3] 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.

[4] GOLC - Global Online Lab. Consortium, 2011. [Online]. Available: http://online-lab.org/. [Accessed: 14-Jun-2011].

[5] 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.

[6] 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.

[7] LiLa - Library of Labs [2009 - 2011], 2011. [Online]. Available: http://www.lila-project.org/content/index.html. [Accessed: 15-Jun-2011].

[8] Christian Maier and Michael Niederstätter, "Lab2go – A Repository to Locate Online Laboratories," *International Journal of Online Engineering (http://www.i-joe.org/)*, vol. 6, no. 1, 2010.

[9] Lab2go Web portal, 2011. [Online]. Available: http://www.lab2go.net/. [Accessed: 15-Jun-2011].

[10] R. J. Costa et al., "Reconfigurable weblabs based on the IEEE1451 Std.," *1st IEEE Engineering Education 2010 – The Future of Global Learning in Engineering Education (EDUCON'2010), Madrid - Spain*, p. 8, Apr. 2010.

[11] Albert T. Corbett et al., "Intelligent Tutoring Systems," Handbook of Human-Computer Interaction, Second, Completely Revised Edition in M. Helander, T. K. Landauer, P. Prabhu (Eds), Elsevier Science B. V., no. 37, 1997.