Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

OPC-DB Link for the Management of New Systems in a Remote Laboratory

**Manuel Domínguez, Serafín Alonso, Juan J. Fuertes
Miguel A. Prada, Antonio Morán, Pablo Barrientos**

*SUPPRESS research group, Esc. de Ing. Industrial e Informática,
Campus de Vegazana s/n, 24071, León, Spain
(e-mail: author@unileon.es).*

Abstract: The remote laboratories are proven tools for technological training. In these laboratories, the students interact with a real system through the Internet, as if they were physically in front of the system. When a remote laboratory is developed, many technical difficulties arefound, mainly with respect to the links between the different elements such as physical system, database and clients. In this sense, it is necessary to make an effort to standardize the implementation of the links. In this paper, we propose a standard application to communicate the physical systems and the database. This middleware, called OPC-DB, uses OPC (OLE for Process Control) for communication with control systems and has been developed in LabVIEW. The software can be easily reused in different laboratories by means of a database.

Keywords: Web-based laboratories, remote monitoring, control education, laboratory management system, data exchange, OPC-DB link, OPC protocol.

1. INTRODUCTION

A complete learning experience in engineering education should include experimentation with systems to introduce professional practice and skills, support analytical concepts and increase the involvement of the students (Lindsay and Good, 2005). Remote laboratories have proven to be effective educational tools in engineering, with comparable results to traditional on-site labs (Nickerson et al., 2007). Indeed, they provide additional advantages, such as their flexibility, which is appealing for the students. The remote laboratories also enable sharing the scarcely available equipment. This is a relevant issue because education on technological subjects often requires the use of complex and expensive equipment. As a result, the development of virtual and remote laboratories in the field of automatic control has advanced significantly in the last years (Guzmán et al., 2010; Leva and Donida, 2008). The Remote Laboratory of Automatic Control of the University of León (LRA-ULE) (Domínguez et al., 2011)is a laboratory oriented to research and education with a focus on real industrial equipment.

The experience has shown that, in order to provide educational value with a reasonable use of resources, a remote laboratory needs to be more than a working prototype and ensure certain standards in manageability, extensibility, security and accessibility (García-Zubia et al., 2009). It is necessary to recognize the importance of software to provide the performance and scalability needed to ensure its educational usefulness. Convenient approaches will also alleviate the workload needed to create new experiments or set up new physical devices. For that reason, research in the field of remote laboratories is progressively focusing on hardware and software solutions that address usual problems or provide general frameworks.

The traditional architecture of remote laboratories is N-tier, also a usual structure in general web development (Eckerson, 1995). Between the client application that runs in the computer of the user and the physical systems, there is a number of layers that provide different services such as authentication, scheduling, data logging, additional educational content or security. The communication between adjacent layers is a key issue because its implementation should balance effectiveness, flexibility and rapid development. Usually, these aims is met through the provision of interface libraries that can be used by the applications to communicate transparently. These application programming interfaces (API) often make use of state-of-the-art technologies such as web services.

In the field of industrial automation, there is a standard called OPC (Object Linking and Embedding for Process Control) for the communication of real-time plant data between control devices from different manufacturers. This paper proposes leveraging the OPC standard to communicate between the industrial plants and the upper management layers. It presents a middleware based on OPC that acts as interface between the industrial equipment and the middle layer of the LRA-ULE platform. Section 2 describes the state of the art regarding the architecture of remote laboratories and the solutions proposed to communicate between their layers. Section 3 presents the proposed approach and its implementation. After that, section 4 explains the usage of the middleware in the LRA-ULE platform. Finally, the conclusions are discussed in 5.

2. ARCHITECTURES AND COMMUNICATION APPROACHES IN REMOTE LABORATORIES

The field of remote laboratories has achieved a certain level of maturity. Nowadays, several platforms provide

much more than the simple remote control/monitoring of a physical system through the Internet. They provide resource-efficient solutions oriented to achieve aims such as scalability, platform independence, security and accessibility. There are nevertheless two main goals that any effective remote laboratory must address. One is the ability of connecting heterogeneous physical systems easily. The other one is to alleviate the workload of the website administrators, who are sometimes faculty and need to spend their time developing educational content instead of performing tasks such as user management, new systems configuration, etc. In order to provide the required level of manageability, the approaches use multi-tier architectures, centralize some core services and use standardized interfaces for communication.

In this section, some relevant examples of remote laboratories, which are not constrained to the field of control engineering, are presented. It can be seen that some of them manage the communication with the system completely, storing data and controlling the interaction, whereas other platforms simply set up a connection and do not participate in the data exchange. We are more interested in the platforms that support completely managed experiments.

A platform that supports managed and unmanaged experiments is the WebLab-Deusto remote laboratory. This remote laboratory is a framework developed to decouple infrastructure development and experiment development (García-Zubia et al., 2009). Through plug-ins and APIs available in different client-side and server-side technologies, they achieve certain aims such as scalability or distribution. Regarding infrastructure, the architecture uses a login server for authentication and core server to manage scheduling, storage, requests, etc. The laboratory and experiment servers are used to manage the laboratory. The unmanaged experiments can use, for instance, virtual machines. In the experiments that are managed by the platform, the communication is made through a set of commands using web services. The clients need to use the client library, with methods for submitting a command, whereas the experiment servers can implement the server library or use XML-RPC directly.

On the other hand, the University of Technology, Sydney (UTS) Remote Laboratory follows an unmanaged approach. The architecture of the UTS remote laboratory is developed to be flexible, extensible and able to manage multiple sets of equipment (Lowe et al., 2009). The user needs a web browser and a remote desktop client. The user stars session on a web browser, the arbitrator allocates equipment to the the student, provides visual monitoring of the experiment and boots a virtual machine on a master server to enable control of the system. To control the system, the user creates a remote desktop connection to this virtual machine. The arbitrator supports access queues and management of access to multiple identical systems.

Another example of comprehensive remote laboratory is the iLab Shared Architecture (ISA) of the Massachusetts Institute of Technology (MIT), which follows a typical three-tier structure. This distributed software toolkit pursues scalability, platform independence and efficient management (Hardison et al., 2008). The first tier is the user application, available in a web browser. The second tier is called the service broker, which provides authentication and manages access to the experiments. The third tier comprises the lab servers that are connected to the physical devices and controls the experiment. The lab client and the lab server only communicate directly to the service broker. For that purpose, two APIs exposed as web services with pass-through methods are available. Each experiment needs a lab server. The clients must be able to produce and interpret XML documents in compliance with a lab client/server communication schema. The lab servers also need to interpret and produce those documents and transform experiment parameters to physical commands. The architecture can be shared between two or more institutions, the two first tiers are located in an institution and the broker server communicates with the lab server on charge of the experiment.

As can be seen from the overview of available platforms, there are 3 key issues that need to be addressed in the design of a remote laboratory. First, it is necessary to decide how to implement the client-side communication, because this choice influences the development of the clients. Second, we need to establish the functionality and structure of the core server, sometimes also known as arbitrator or broker. This server, or set of servers, is in charge of authentication, user and session management, resource allocation, etc. and therefore it is crucial in the operation of the laboratory. Third, the communication interface with the control systems of the physical systems must be designed. This interface must enable easy configuration of existing and new equipment in the platform.

The LRA-ULE remote laboratory addresses client-side communication by providing a library for client development, available both in Java and JavaScript. This library provides a simple interface for system interrogation and I/O operations. The interface communicates with a PHP module in the server-side, running in a Drupal content management system (CMS). The set of CMS, specific LRA-ULE modules developed in PHP and proxy server performs equivalent tasks to the broker/arbitrator servers available in other platforms (Domínguez et al., 2012). This is a similar approach to the one proposed in (Mejías et al., 2013), where user management is performed by a plug-in of a content management system and another component called Remote Access System for Laboratories (RASLAB) is used to set up the needed links between the user interface and the experimental system without compromising security. Apart from managing the communication between a client in a public network and the experimental system in the intranet, RASLAB can also handle the power supply of that system.

Several approaches have been proposed to alleviate the work necessary to make existing local systems accessible through the web. For instance, a middleware application (JIL) is proposed in (Vargas et al., 2009) to enable communication between clients in the common form of Java applets and a platform widely used in the field of instrumentation, control and acquisition systems such as LabVIEW. This tool have been tested by several Spanish universities with Easy Java Simulation applets in the client-side and LabVIEW in the server side.

Regarding the communication with the end systems, the approach taken by the LRA-ULE remote laboratory should address its main goal: to connect industrial-oriented equipment. For that reason, it is convenient to reuse common procedures in industry for the communication and control of the physical system. This way, the work and knowledge required to configure or add a system will be minimized. For that reason, contrary to other approaches, the proposed mechanism avoids the implementation of an ad-hoc application, using a certain API, in the computer associated to the physical system. Instead, it uses the OPC Data Access standard. OPC provides a common bridge between PC applications and process control hardware, with consistent access methods to field data regardless of the control system device. Thus, general OPC clients can communicate with any hardware as long as an OPC server is available. Therefore, there is no need to re-implement the interface for new devices in a SCADA/HMI.

This paper presents a middleware (OPC-DB) that communicates the industrial systems with a standard database associated to the system. Since the middleware acts as an OPC client, it is only necessary to install and configure the OPC server that is compatible with the control system in the end computer. The OPC server is usually provided by the manufacturer and its configuration procedure is common knowledge among control engineers. Therefore, this approach enables easy and fast configuration of new systems and, in turn, fewer problems and increased flexibility. The main drawbacks are that the acquisition of OPC servers will increase the cost of the solution and that some low-end devices might not be OPC-enabled.

3. THE OPC-DB LINK

In this section, the proposed middleware, OPC-DB link, is described. This middleware connects the physical system with the intermediate layer of the remote laboratory by acting as a link between the OPC server associated to the system and the database of the remote laboratory.

3.1 Description and purpose of the OPC-DB link

The management system of a remote laboratory generally relies on a database that, apart from other information, deals with data sampled from the systems and actions submitted by the users. The purpose of the OPC-DB link is to read the status from the input variables of a system and store them in the database, and to retrieve the actions from the database and write them to the corresponding output variables of the system. Fig. 1 summarizes this operation.

As discussed in the previous section, communication with the system is performed through the OPC Data Access standard. The middleware works as an OPC client that obtains from a database the information about which OPC server and variables it needs to connect to. The communication with the database is performed by means of a standard ODBC driver. This way, the middleware preserves its modularity and independence from the type of system, so that the integration of new equipment in an existing laboratory is easier. The database needs to be integrated in the operation of the remote laboratory. It is common that interactions in the client applications

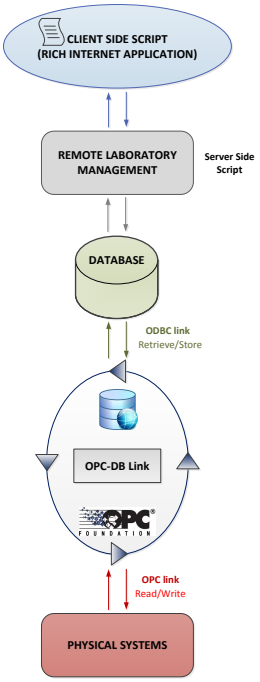

Fig. 1. General structure of the OPC-DB link

result in the execution of server-side scripts that update and query the database by means of stored procedures. In that case, the requirements to apply the proposed middleware would just be the creation of some tables with the distinctive parameters of the system (variables, data types, OPC tags, etc.). It is also possible to add/erase system variables by simply adding/removing rows of the table. The OPC server of the real system must also be configured and activated.

3.2 Functionality of the OPC-DB link

The OPC-DB link carries out several tasks to accomplish its functionality (see Fig. 2). A call to the OPC-DB link needs two parameters: the name of the database associated with the system and the run or stop command. When a running command is sent, the connection with the corresponding database is opened and the event and reference to the application instance are registered in the log table. After that, the configuration parameters associated to the system (OPC name, sampling period, lifespan limit, etc.), the system variables and their data type and the OPC tags are retrieved from the database. The data types configured in the database must match the ones from the OPC server.

Once all the information is known, the connection to the OPC server is established and the acquisition loop starts. In each iteration of the loop, the values of each of the OPC tags corresponding to the system input variables are read and stored into a table of the database. This way, the remote laboratory services can provide real-time data of the operation of the system to their clients. The data (actions) sent by the remote clients are updated in the database and, in turn, written to the OPC tags that correspond to the system output variables. However, for efficiency purposes, data are only written when the values in the table differ from the ones in the last iteration. In

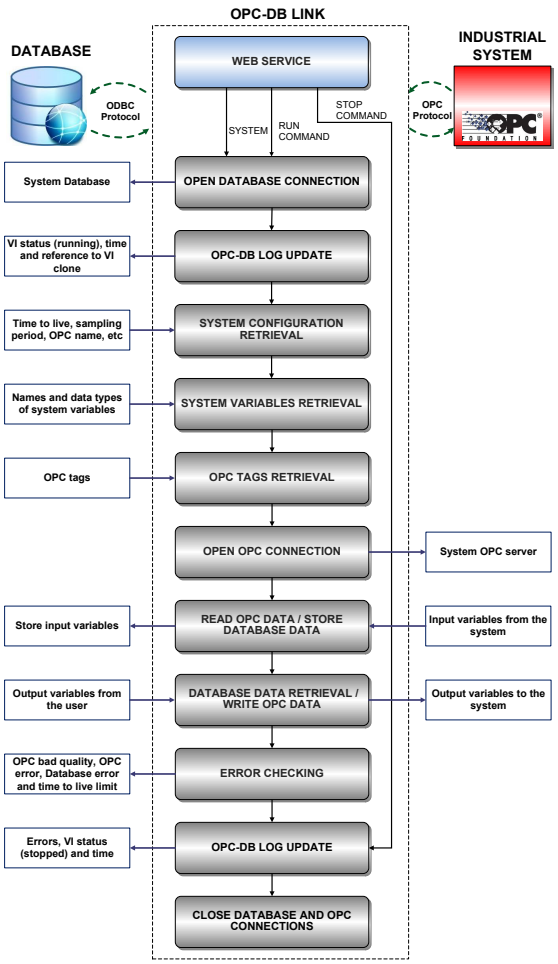

Fig. 2. Functionality of the OPC-DB link

each loop iteration, possible errors in the connection with the OPC or the database are checked. The middleware stops automatically whenever an error is detected and writes the details in the log. Besides, the quality of the data provided by the OPC server for each tag is checked and an error, which stops the middleware, is reported whenever the quality of any variable is found to be bad for ten consecutive iterations.

Finally, when the OPC-DB link receives the stop command, it is logged into the database and the connections with both the OPC and the database are closed. The OPC-DB link also stops whenever it detects inactivity by the users, i.e., when a time longer than the lifespan limit has passed since the last update of the output variables. The inactivity control avoids the undefined link execution if the remote laboratory platform fails to detect the user exit.

3.3 Programming and deployment of the OPC-DB link

The OPC-DB link has been programmed in LabVIEW [1], a graphical programming language developed by National Instruments for scientists and engineers, which uses intuitive graphical blocks instead of written code for the development of applications involving acquisition, control, analysis and data visualization (National Instruments, 2003). LabVIEW includes extensive libraries of basic and

[1] http://www.ni.com/labview/esa/

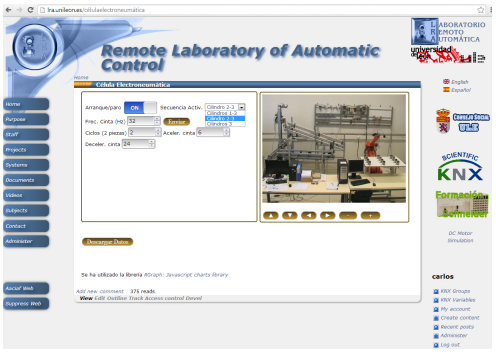

Fig. 3. Remote client of the electro-pneumatic cell

specific functions for data acquisition, instrumentation control, communication and data storage, as well as a web server which lets us publish remote front panels, web services, etc. Standard communication protocols such as serial, GPIB, Modbus, OPC, etc. are also available. Due to its extensive functionality and connectivity, LabVIEW programming language has been chosen.

LabVIEW programs are called *Virtual Instruments* (VIs) and consist of two parts: the front panel (graphical user interface) and the block diagram (source code). In this case, the design of front panel is not important since the OPC-DB link runs transparent to the user. The LabVIEW project of the OPC-DB link has been structured in two VIs: the main VI implements the complete functionality of the OPC-DB link whereas the command VI is only used to command (run/stop) the main one. The LabVIEW VI server is used to communicate between both VIs. The main VI is defined as re-entrant to allow that multiple instances of itself can run in parallel without interfering with each other. Therefore, the OPC-DB link allows multiple calls so that many users can use different physical systems (different variables, parameters, etc.) at the same time in the remote laboratory. For each different system, a different instance is running. On the other hand, the command VI is deployed as web service to facilitate the integration with the management of the remote laboratory. It requires that the LabVIEW web server is active.

4. USAGE OF THE OPC-DB LINK IN THE LRA-ULE REMOTE LABORATORY

As stated before, the LRA-ULE remote laboratory is a platform used for education and research with a focus on industrial-oriented devices. Therefore, the architecture has to be flexible enough to be able to integrate new equipment easily. The OPC-DB link is well adapted to this structure, because it is based on open technologies and facilitate the communication between the database and a physical controller regardless of the manufacturer. In this section, the physical systems that have been connected to the platform through the OPC-DB link application are described in detail.

The electro-pneumatic classification cell is a physical system that aims to classify steel pieces ($6 \times 6 \times 8$ cm) in three categories This system includes many different industrial equipment such us PLCs, drives, sensors, actuators or a six degrees or freedom robot. For the user interaction with the electro-pneumatic classification cell, an HTML5/AJAX

Table 1. Variables of the system included in the remote laboratory.

ID	System output variables
1	System activation signal
2	Activation sequence of the cylinders in the conveyor belt
3	Number of cycles performed by the system
4	Frequency of the drive
5	Acceleration curve of the frequency drive
6	Deceleration curve of the frequency drive
ID	**System input variables**
1	Frequency of drive 1
2	Power of drive 1
3	Speed of drive 1
4	Frequency of drive 2
5	Power of drive 2
6	Speed of drive 2
7	System status
8	Rail 1 activation
9	Rail 2 activation
10	Rail 3 activation
11	Start signal for code recognition
12	Status signal for code recognition

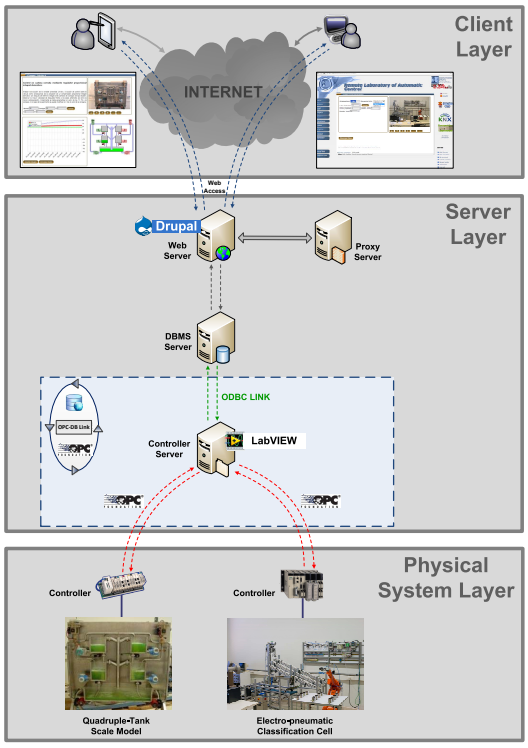

Fig. 4. OPC-DB link application in the LRA platform

client (see Fig. 3) is used. It allows to read and modify some of the physical system variables, which are shown in Table 1.

Figure 4 shows the architecture to integrate the electro-pneumatic cell into the LRA platform by means of the OPC-DB link. As discussed in section 2, this architecture is three-tier. The first layer is formed by the physical equipment and their control and acquisition systems. In the intermediate layer, there are the servers that manage most of the aspects needed in a remote laboratory. The content management system, web server, security and communication manager and database are found in this layer.

The purpose of this work is to include the OPC-DB link middleware in this intermediate layer to manage the communication between the database management system, where the data of the remote laboratory are stored, and the physical system. The database enables a seamless integration of the OPC-DB link in the platform, which previously used other non-standard ad-hoc solutions to provide that functionality. This is achieved by a minor adaptation of the stored procedures to work with the predefined structure of the tables needed to use the middleware. It needs to be noted that those tables store not only values of the variables, but also configuration parameters of the link with the OPC server.

In more detail, a database suited to work with the OPC-DB middleware must contain the six tables listed in the Table 2. In the "Input variables" and "Output variables" tables, the user must create a row for each of the I/O system variables. These tables have three columns: identifier (ID), data type (analog or digital) and OPC tag. Each column of the "OPC data" and "OPC actions" tables corresponds to a variable of the system. The OPC-DB link application reads the values of the variables introduced in the "Input variables" table in order to insert them into the "OPC data" table. Apart from that, the middleware decides whether to update an output variable in the physical system by reading the current values in the "OPC actions" table and looks up the "Output variables" table to get the OPC tags where the values need to be updated. The "Logging" table is used to record the events and errors of the middleware and the "Config" table includes the address of the OPC server, the desired sampling period and the desired maximum idle lifespan. The server-side scripts call directly some stored procedures for reading/writing data and starting/stopping the acquisition. In order to integrate the middleware, those store procedures have been adapted.

On the other hand, it is necessary to configure the OPC server, which in the particular case of the electro-pneumatic cell is a Scheider Electric OPC Factory server. This is achieved by generating a symbol table from the system control strategy, in this case implemented in a Premium TSX PLC using Unity Pro. The symbol table includes all the variables of the strategy, but only the variables in the "Input variables" and "Output variables" are used by the remote laboratory.

The top layer includes the client application, implemented with HTML5 and AJAX, which only communicates with the server-side script through the library implemented for that purpose. Therefore, the use of the OPC-DB link will be transparent to the user.

In addition to the electro-pneumatic cell, the OPC-DB link application has been used for the communication with another physical system, the quadruple-tank model. The steps to include the system are equivalent to those presented for the electro-pneumatic cell. A database with the aforementioned structure needs to be created for this particular system. The necessary stored procedures will be exactly the same than in the previous case, so there is no need to write any further code. The input variables in this system are basically the levels of the four tanks and the status of the actuators, whereas the output variables comprise the speed of the pumps, the openings of the

Table 2. Structure of the database tables

Name	Description
OPC actions	Output var. sent by Drupal & data types
OPC data	Input var. and data types
Config	Sampling time, OPC server address & lifespan
Logging	System errors and events
Input var.	Name, ID, type & OPC tag of input var.
Output var.	Name, ID, type & OPC tag of output var.

valves and the status of the digital valves but also the gains of the PID controller that is implemented in the strategy. In this case, the OPC Server is an Opto OPCServer because the strategy was developed in PAC Control Basic for an OPTO SNAP Ultimate I/O controller

The OPC-DB link has represented a clear advance regarding the workload needed to configure or add a device, when it uses a control or acquisition system that has an OPC server available. Several instances of the middleware can run simultaneously for different systems. The existence of a single piece of software in the server-side and no ad-hoc applications running in the computers associated with the systems is also and advantage with regard to maintenance and reusability.

5. CONCLUSION

The remote laboratories provide web environments to interact with physical systems. From a technical point, designing a remote laboratory is a complex task. In contrast to the traditional laboratories, where it is just necessary to select the physical system and develop the educational contents, a technological platform to manage student access has to be developed. This platform is composed of several layers such as the local connection to the physical system, the database and core management system, or the client interface. A software solution is needed to communicate these layers. In the last years, alternatives have been proposed to implement those communication in standard and reusable applications, but these approaches usually focus on the usage of ad-hoc libraries. To let faculty focus on the development of educational contents, it is necessary to enable connection of new hardware without much need of programming or technical tasks.

For that reason, we proposed a standard solution to link industrial-oriented physical systems with a database in remote laboratories. A middleware has been programmed in LabVIEW, so that the link uses the OPC standard on the side of the physical system and the ODBC driver on the side of the database. If administrators want to use this standard solution to include a system in their remote laboratory, they only need to install and configure the OPC server associated to the control system (usually provided by any manufacturer of control equipment) and create the associated tables in the database. The proposed link has been used to integrate two physical systems in the remote laboratory LRA-ULE at the University of León: an electro-pneumatic cell and a 4-tank process. The steps needed to incorporate these systems into the remote laboratory have been explained in detail. This approach decreases the workload and knowledge needed to add or configure physical systems.

REFERENCES

Domínguez, M., Fuertes, J.J., Prada, M.A., Alonso, S., and Morán, A. (2011). Remote laboratory of a quadruple tank process for learning in control engineering using different industrial controllers. *Computer Applications in Engineering Education*, n/a–n/a. doi: 10.1002/cae.20562.

Domínguez, M., Prada, M., Morán, A., Alonso, S., and Barrientos, P. (2012). Improving user interaction in remote laboratories through html5/ajax. In *Advances in Control Education*, volume 9, 282–287. doi: 10.3182/20120619-3-RU-2024.00036.

Eckerson, W.W. (1995). Three tier client/server architectures: Achieving scalability, performance, and efficiency in client/server applications. *Open Information Systems*, 3(20), 46–50.

García-Zubia, J., Orduña, P., López-de Ipiña, D., and Alves, G.R. (2009). Addressing software impact in the design of remote laboratories. *Industrial Electronics, IEEE Transactions on*, 56(12), 4757–4767. doi: 10.1109/TIE.2009.2026368.

Guzmán, J.L., Domínguez, M., Berenguel, M., Fuertes, J.J., Rodríguez, F., and Reguera, P. (2010). Entornos de experimentación para la enseñanza de conceptos básicos de modelado y control. *Revista Iberoamericana de Automática e Informática Industrial*, 7(1), 10–22. doi: 10.4995/RIAI.2010.01.01.

Hardison, J., DeLong, K., Bailey, P., and Harward, V. (2008). Deploying interactive remote labs using the ilab shared architecture. In *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*, S2A–1–S2A–6. doi:10.1109/FIE.2008.4720536.

Leva, A. and Donida, F. (2008). Multifunctional remote laboratory for education in automatic control: The CrAutoLab experience. *Industrial Electronics, IEEE Transactions on*, 55(6), 2376–2385. doi: 10.1109/TIE.2008.922590.

Lindsay, E. and Good, M. (2005). Effects of laboratory access modes upon learning outcomes. *Education, IEEE Transactions on*, 48(4), 619–631. doi: 10.1109/TE.2005.852591.

Lowe, D., Murray, S., Lindsay, E., and Liu, D. (2009). Evolving remote laboratory architectures to leverage emerging internet technologies. *Learning Technologies, IEEE Transactions on*, 2(4), 289 –294. doi: 10.1109/TLT.2009.33.

Mejías, A., Márquez, M.J., Andújar, J.M., and Sánchez, M.R. (2013). A complete solution for developing remote labs. In *10th IFAC Symposium Advances in Control Education*, 96–101. IFAC. doi:10.3182/20130828-3-UK-2039.00057.

National Instruments (2003). *LabVIEW user manual, no. 320999E-01*. URL `http://www.ni.com`.

Nickerson, J.V., Corter, J.E., Esche, S.K., and Chassapis, C. (2007). A model for evaluating the effectiveness of remote engineering laboratories and simulations in education. *Computers and Education*, 49(3), 708–725. doi:doi:10.1016/j.compedu.2005.11.019.

Vargas, H., Sánchez-Moreno, J., Dormido, S., Salzmann, C., Gillet, D., and Esquembre, F. (2009). Web-enabled remote scientific environments. *Computing in Science & Engineering*, 11(3), 36–46.