

From Online Experiments to Smart Devices

Christophe Salzmann & Denis Gillet
École Polytechnique Fédérale de Lausanne, Switzerland

Abstract—Online experiments have been available for more than a decade. The integration of online experiments into collaborative environments is more recent. The wealth of client applications/environments, the versatility of possible interaction protocols/technologies and the needs for more autonomous actions impel the evolution of online experiments to the smart device concept. This paper reviews the evolution of an electrical drive experiment and presents the requirements for turning online experiments into smart devices.

Index Terms—Agent, collaborative environment, online experiment, smart device.

I. INTRODUCTION

Online experiments are typically used by brokers to provide distant users with remote experimentation. For example, the access to an online experiment during a live demonstration in a classroom. Online experiments are often used in control, robotic and mechatronic education for illustrating theoretical principles and deployment methodologies. The different control design and implementation steps taught to students in control courses (system identification, controller design, real-time control, performance validation, etc.) can be efficiently carried out remotely on mechatronic systems as they exhibit visually observable dynamical behaviors.

Remote experimentation solutions are based on a client-server approach where the server is connected to the physical equipment and the client application is connected to the server via the Internet. The user interface can be of various forms but it is generally proposed through a web browser. The aim of a remote experimentation solution is to make the student interaction with the distant system as close as possible to the actual work on the real equipment. Collaborative environments are proposed to support the distant user learning process. These environments integrate various services to streamline the user experience and to help the user environment appropriation. To provide a tight integration of remote equipments within collaborative environments, specific care has to be taken regarding the interface, the provided features and the communication protocol. This customization leads to a very specific solution that is difficult or impossible to integrate into another collaborative environment. The concepts of smart devices are used to expand the online experiment scheme such that the proposed solution is adaptive, autonomous, as envisioned in the Internet of Things realm.

This paper is organized as follows: the physical equipment locally controlled is first presented in section II. In section III, a communication component is added to permit remote access. The integration of online experiments into collaborative environments is presented

in Section IV. Section V presents the smart device concept applied to online experiments and its requirements. Section VI concludes by presenting hints about smart device evolution.

II. PHYSICAL EQUIPMENT AND LOCAL ACCESS

The physical equipments remotely controlled are often mechatronic devices with moving parts as they exhibit visually observable dynamical behaviors. For example the laboratory-scale electrical drive (Fig. 1) is used in many textbooks and courses to illustrate automatic control theory. This setup consists of a DC motor equipped with a digital encoder. The motor drives a brass disk acting as the load. The angular position is measured with a digital encoder connected to the motor axle. Along the same axle, an enlarged rotating disk permits an easy visualization of the motion. This enlarged disk and the rotating load motion are captured by a video camera. The whole hardware has been designed in such a way that it is fully controllable from the connected computer. Similarly, the hardware state can be diagnosed from the connected computer. For example, in addition to the required disk position and speed measurements, diagnostic signals informing about the power status can be read from the connected computer. Likewise, additional actuators have also been added. For example a second motor acting as a generator is placed along the main motor axle to generate a perturbation that can be controlled remotely by switching the generator load. Also, the main power can be switched *on* and *off* from the connected computer to save energy when not used.

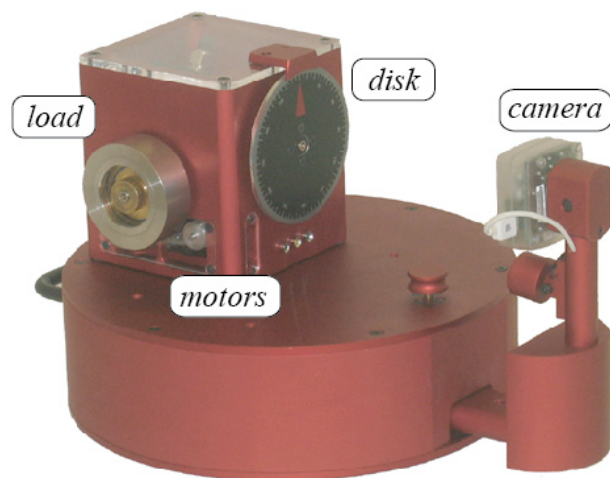


Figure 1. Laboratory-scale electrical drive

Depending on its complexity and computational resource requirements, the control of the equipment can also be performed via an on-board micro-controller. The former solution offers more flexibility than the latter.

The considered physical equipment is used by students to apply the control theory learned during ex-cathedra classes. For example, the experimentation protocol consists in choosing the right set of parameters to position the load or to impose a rotating speed according to some specifications. The PID controller is implemented as a real-time task that communicates with both the physical equipment and the graphical user interface (GUI) [1].

With the help of the graphical user interface, the users can experience the effect of the various controller parameters and see their effects on the physical equipment. Figure 2 shows the user interface of the application that locally controls the laboratory-scale electrical drive. It is written in LabVIEW [2]. The upper part displays the acquired measurement in a scope area. The lower part is intended for the user to modify the controller parameters as well as the reference signal.

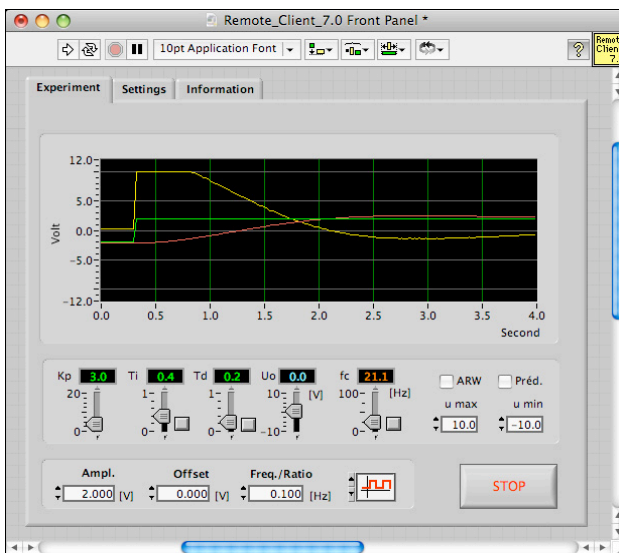


Figure 2. Local control of the electrical drive

III. FROM LOCAL TO REMOTE ACCESS

Providing remote access to physical equipments controlled locally by a dedicated computer is fairly straightforward assuming that the computer is connected to the Internet. Depending on the required degree of interaction between the distant user and the physical equipment both turnkey and customized solutions can be envisioned.

A. Turnkey solutions

Nowadays many turnkey solutions are proposed by commercial applications. The quickest solution is the use of applications that enable the sharing of a distant computer screen (VNC, Remote Desktop, Timbuktu, etc.). While easy to implement this solution generally suffers a high bandwidth usage. Another drawback is that the user cannot see the physical equipment. Visualizing the controlled equipment is considered a key feature to differentiate simulation from real experimentation. Also directly accessing the remote server may grant too much

rights to the distant user, permitting him/her to act on aspect he/she is not supposed to.

Another kind of turnkey solution relies on specific applications that are able to generate a remote view of the local GUI. This view is generally displayed in a web browser. The server application can either generate a dynamic web page or require specific client applications (LabVIEW remote panel). Both solutions can be enabled with only a few mouse clicks but suffer the same visualization drawback as the screen sharing solution. While a video stream coming from an IP camera may be embedded to the web page containing the departed view, the information synchronization between the sources of information is added to the potential bandwidth problem.

B. Custom-made solutions

Custom-made solutions permit a finer control of all aspects involved in remote experimentation. This at the cost of additional developments to create both the client application and to add the network communication layer to the local application [3].

Various technologies can be used to implement the client application. Web-based technologies (web 2.0, Java, Flash, Silverlight, QuickTime, ActiveX, etc.) are the most widely used since they are ubiquitous and often pre-installed within the web browser.

Depending on the client application requirements, the chosen technology should provide the following features:

- display GUI elements (button, etc.)
- ability to get user actions and events (mouse, keyboard)
- support communication protocol (TCP, UDP, HTTP, etc.)
- timing synchronization (threads, timers)

As of today Java is the most versatile option that permits the finest control of the client application. A solution that only uses standard web technologies (CSS, Ajax) without the help of plug-ins is possible provided that the web transmission protocol (HTTP/TCP) does not constrain the envisioned communication. Java provides the full control over the above features necessary to implement an advanced remote experimentation client. It especially permits to implement alternate transmission protocol (UDP) and allows a tight synchronization between the various flows of information (video, data, parameter).

Figure 3 presents a java applet that enables to control the distant laboratory-scale electrical drive. The provided interface permits the user to change the various parameters of the PID controller and to see its effects in real-time via the oscilloscope area and the video feedback.

The main adjunctions to the distant application that locally control the physical equipment (Fig. 2) are the video acquisition and transmission layer that allows remote clients to communicate with the online experiment. With these additions the local application becomes a server application. Additional requirements and best practices are presented in [4].



Figure 3. The client application implemented as a java applet

IV. WEB 2.0 SOCIAL SOFTWARE FOR COLLABORATIVE LEARNING AND INTERACTION WITH ONLINE EXPERIMENTS

The additional flexibility provided by remote experimentations is highly appreciated and permits distant users to manage the remote experimentation sessions at their own pace and from their own location [5]. One drawback is that the learning modalities often found on campus should be emulated. Collaborative learning support should be provided, as well as some forms of tutoring and assistance. Collaborative environments such as *emersion* [6] and *elogbook* [7] support the activities with online experiments by providing additional services such as shared spaces for saving measurements, discussion forums, live support, etc. The online experiments need to be specifically adapted to maximize the benefits offered by collaborative environments. Not only the client applications need to be adapted, but also the server application. For example authentication is often required by collaborative environments, thus the client and/or the server applications must be adapted to support authentication. Similarly, saving data in a shared space requires specific protocols that may not be present initially. When possible, some or all the required adaptations could be implemented by a proxy application that bridges both worlds. This translation is often done at the cost of performances. This proxy application bridging mechanism can be generalized as the concept of *agent* that works on the behalf of users, or in the presented case on the behalf of the online experiment server [8].

Figure 4 shows the remote experimentation agent working on the behalf of both the user *Chris* and the equipment *RT-201* within the *elogbook* collaborative environment. The measurements acquired with the help of the agent (center of Fig. 4) are directly saved within the shared space (right column of Fig. 4) and are visible for the members (left column of Fig. 4) of the given activity (top of Fig. 4). The agent has been granted the right to directly save measurements in the shared space after it authenticated using the provided *elogbook* mechanism.

The original online experiment does not know about the *elogbook*, it is only aware of the *elogbook* agent. Using the agent concept, the online experiment could be interfaced by various collaborative environments but would require a dedicated agent for each new

environment. To alleviate this restriction and to permit “any” client to access the online experiment, *all* agents should be hosted at the server side. While this is not directly feasible since *all agents* are not known beforehand, the structure to support multiple types of connections, protocols, modes of operation and interfaces in an autonomous and self-contained way should be implemented in the server.

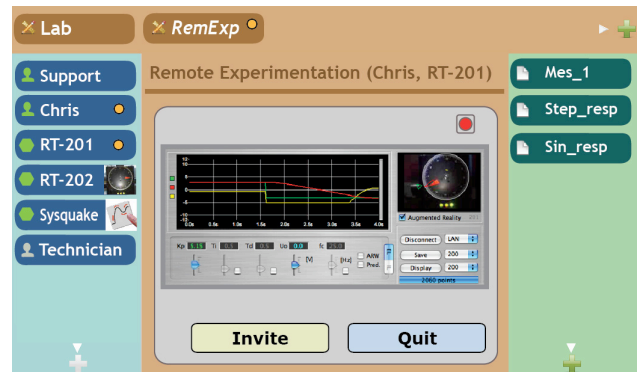


Figure 4. Remote experimentation agent in *elogbook*

A server connected to physical equipment with the above capabilities and the ability to interact autonomously with other machines is called a smart device. In the above example the communication between the applet agent and the *elogbook* is performed with the help of webservices. When running in the standalone mode (Fig.3) the applet communicates with the smart device via raw UDP blocks.

V. SMART DEVICES

Smart devices can be described as devices that have some autonomy to perform actions. These devices often have sensors and/or actuators and support communication with other devices. The interconnection of smart devices and other intelligent objects sketches the Internet of Things.

The Internet of Things is a metaphor that envisions the connection of all existing objects for the universality of communication processes, for the integration of any kind of digital data and content, for the unique identification of real or virtual objects and for architectures that provide the communicative glue among these components [9].

Thomson [10] suggested that smart devices need some or all of the following capabilities:

- i) communication
- ii) sensing and actuating
- iii) reasoning and learning
- iv) identity and kind
- v) memory and status tracking

A. Requirements for turning online equipment into smart devices

Physically, the considered smart device is made of the adjunction of the controlling computer -the server-connected to the physical equipment on one side and to the Internet on the other side. The capabilities required for smart devices controlling physical equipment are twofold.

The first set of requirements is related to the physical equipment. The physical equipment should be *identifiable* to define what kind of equipment is connected. Also the equipment should be fully *controllable* and *diagnostic-*

able by the controlling computer. Due to security reasons the full controllability of the physical equipment may not be exposed to the outside world. Controllability also implies that it is always possible to place the equipment in a known state. Other requirements such as *reliability* and *maintainability* should also be considered.

The second set of requirements is related to the interaction of the controlling computer and the outside world. This interaction implies that first the computer is *connected* to the Internet and capable of *understanding* incoming *requests* and to reply to them. It should also be capable of some *autonomy* to report for example alarms or its status. *Security* and *authentication* must also be provided. It should not be possible to temper with the physical equipment from the outside world. This is also true for the hosting computer and the engaged communications. In the considered case, the server may propose a graphical user interface to interact with the server.

The communication requirements suggest that the server is able to talk any low level protocols such as TCP or UDP to get the requests and send replies, but also high level protocols or technologies such as HTTP, XML, REST, FTP, XML-RPC, WSDL, POP, MAIL, RSS, etc. to understand the requests. It is definitely not possible to implement all possible protocols but the structure to handle new protocols should be in place to minimize the development effort. The next section presents the chosen set of protocols to be implemented and the rationale behind these choices.

B. Smart device example

The proposed smart device is an extension of the online experimentation server described in Section II. Initially, the sole task of the server was to control the physical equipment locally. Then a communication component was added to send measurements and to receive parameters from a home-built application using the UDP protocol. The access from a web browser required the writing of a Java applet and the associated server modifications. The integration of remote experimentations in collaborative environment implied the adjunction of new functionalities and new protocols and the use of the agent concepts. The global management of the available resources was also required to dynamically spread the load among the available online equipments leading to additional modifications at the server side.

The current server is a smart device with an evolving structure that guarantees compatibility with existing solutions while streamlining the addition of new ones. On the physical equipment side, the server only exposes a limited set of actions that are validated prior to its application on the physical device. Similarly the server only provides aggregated information regarding the physical equipment to the outside world.

The former client applications can interact with the smart device by sending UDP packets that contain parameters for the implemented controller. As a reply, client applications receive two UDP streams, one for the video feed and one for the measurements feed.

The current collaborative environments (*emersion* and *elogbook*) do not have specific interface to the smart device. This interface is provided by the smart device in the form of a Java applet. The smart device is also able to

handle the user credential to directly interact with the collaborative environment. This interface can also be used independently by web browsers without the need for a link to the collaborative environment. The interface functionalities will be adapted accordingly, for example user will be able to save measurements only on the user desktop and not in the collaborative environment shared space.

In addition, the server features awareness information to the outside world. These information covers, among other information, the server status, connections statistics and usage statistics. Client applications can get the above awareness information through various channels. For example by opening a raw TCP connection to the server. Alternatively the client application can use a provided web service using the XML-RPC protocol. An RSS feed with the above information is also provided (Fig. 5).

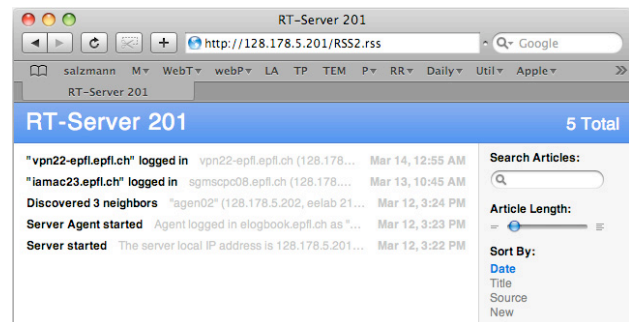


Figure 5. Smart device status information via RSS Feed

Last but not least, it is possible to send an email to the server to get the above information or to get for example measurements (Fig. 6). Interfacing programs via email is not new but has been forgotten over the years. However email support provides a simple and unobtrusive interface that is generally ubiquitously available [11].

The mechanism implemented for the awareness information is also available for the other streams of information (parameters, measurements and video).

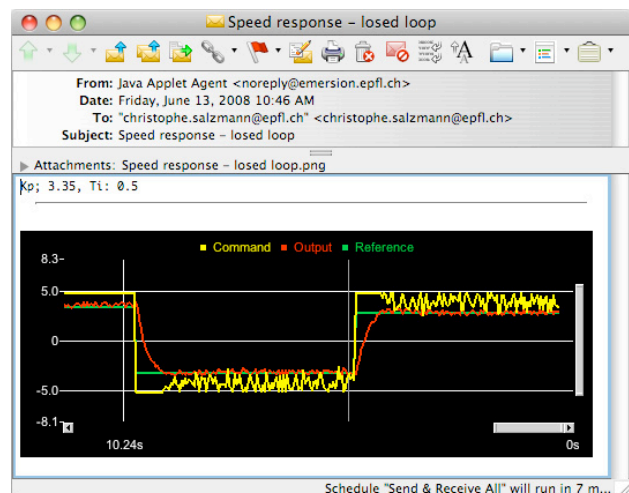


Figure 6. Measurements sent via email

The above information exchange relies on a request-answer mechanism. The smart device is also able to push information to given recipients. In case of self-diagnosed malfunctions the server can send an email and an SMS (Fig. 7). A typical malfunction is the physical equipment

main switch set to OFF. The power status of the equipment is regularly checked and appropriate actions are taken once discovered.

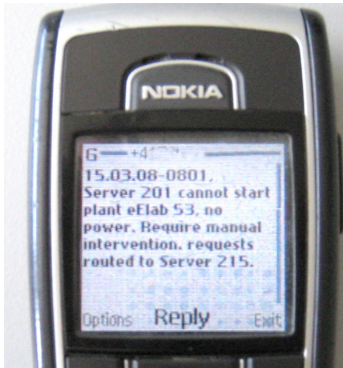


Figure 7. Alarm sent via SMS

The collaborative environment may deal with many devices. An auxiliary application dynamically redirects the collaborative environment agents to available devices. The smart device informs this auxiliary application about its status. If the auxiliary application is unavailable and if the smart device already is in use, the smart device is able to re-route the agent requests to neighbor equipments.

VI. CONCLUSIONS

This paper presents the evolution of an electrical drive that is initially controlled locally by a computer. Various components are added to permit a remote access. The online experiment is then integrated into collaborative environment. The concept of agents is used to permit the online experiment to work on the behalf of the user within the collaborative environment. The performed actions can be controlled when the user decides to save measurements at a given time or autonomously when sending an alarm. If the agent performing autonomous task is part of the server, the resulting tandem can be called a smart device. The requirements for turning online equipments into smart devices are then presented. An emphasis is placed on the communication side that should accommodate to many technologies and protocols. Finally an example depicts the interaction between a smart device and various client applications.

REFERENCES

- [1] Salzmann, C.; Gillet, D. ; Longchamp, R. ; Bonvin, D., "Framework for Fast Real-Time Applications in Automatic Control Education", *4th IFAC Symposium on Advances in Control Education*, p. 345-350, 1997
- [2] <http://www.ni.com/labview/>
- [3] Salzmann, C.; Gillet, D. ; Huguenin, P., "Introduction to Real-time Control using LabVIEW with an Application to Distance Learning", *International Journal of Engineering Education*, vol. 16, num. 3, 1999, p. 255-272
- [4] Salzmann, C., Gillet, D., "Challenges in Remote Laboratory Sustainability", *International Conference on Engineering Education*, ICEE 2007, Coimbra - Portugal, 3-7 September 2007.
- [5] Salzmann C., D. Gillet, P. Scott, and K. Quick, "Remote lab: Online Support and Awareness Analysis", 17th IFAC World Congress, Seoul, Korea, July 6-11, 2008.
- [6] Nguyen Ngoc, A. V. , Y. Rekik, and D. Gillet. "A framework for sustaining the continuity of interaction in Web-based learning environment for engineering education", *World Conference on Educational Multimedia, Hypermedia & Telecommunications ED-MEDIA 2005*.
- [7] <http://elogbook.epfl.ch>
- [8] Salzmann C., C.M. Yu, S. El Helou, and D. Gillet, "Live Interaction in Social Software with Application in Collaborative Learning", *3rd International Conference on Interactive Mobile and Computer Aided Learning*, Amman, Jordan, April 16-18, 2008.
- [9] Federal Ministry of Economics and Technology. 2007. European policy outlook RFID, draft version: Working document for the expert conference "RFID: Towards the Internet of Things". June 2007.
- [10] Thompson, C. W. 2005. Smart devices and soft controllers. *IEEE Internet Computing*, vol. 9, issue 1, Jan.-Feb. 2005, pp. 82-85.
- [11] D. Gillet, C.M. Yu, S. El Helou, A. Madina Berastegui, Ch. Salzmann, and Y. Rekik, "Tackling Acceptability Issues in Communities of Practice by Providing Lightweight Email-based Interface to the eLogbook, a Web 2.0 Collaborative Activity and Asset Management System", *2nd European Conference on Technology Enhanced Learning*, Crete, Greece, September 17-20, 2007.

AUTHORS

Ch. Salzmann is with the École Polytechnique Fédérale de Lausanne, Switzerland (e-mail: christophe.salzmann@epfl.ch).

D. Gillet is with the École Polytechnique Fédérale de Lausanne, Switzerland (e-mail: denis.gillet@epfl.ch).

This article was modified from a presentation at the REV2008 conference in Düsseldorf, Germany, June 2008. nuscript received 1 July 2008.