Laboratories over the network: from remote to mobile

Alessandro Vittorio Papadopoulos, Alberto Leva

Politecnico di Milano,
Dipartimento di Elettronica, Informazione e Bioingegneria
Piazza Leonardo Da Vinci, 32 - 20133 Milano, Italy
(e-mail: papadopoulos@elet.polimi.it, alberto.leva@polimi.it)

Abstract: Modern technologies allow to create a networked control system with off-the-shelf mobile devices. As such, there is the possibility of having the role of who offers and who uses a "remote" laboratory played by the same people. Extending recently published ideas, the paper presents a first nucleus of functionalities allowing one to create process simulators and controllers which run on a mobile application, and then share them with others. Some words are also spent on some of the possibilities opened by the proposal, sketching out some interesting didactic activities to propose to the students.

Keywords: Remote laboratories; mobile applications.

1. INTRODUCTION AND MOTIVATION

Remote laboratories for control education have been receiving a lot of attention in the last years, essentially as a viable and effective means to have students interact with physical experiments irrespective of their location, to share experiments among teaching and research institutions, and so forth, see for example Dormido et al. (2008) and the comprehensive collection presented in Gomes and García-Zubía (2007).

As a result of such a dominant perspective, a correspondingly dominant idea can be observed in the literature – of course, with some exceptions – on the structure, or in some sense the idea itself of "remote control laboratory": a server hosts some experiment, *one* client at a time allows a person or a group to act on that experiment, and possibly other clients allow other persons or groups to watch; examples are found in Murtra et al. (2007); Leva and Donida (2008a); Vicente et al. (2010); Djalic et al. (2012) and many other works.

A number of variants of such an idea exists, including for example reservation mechanisms, the possibility of handing over control from a client to another, the supervision by some "instructor" client, inter-client communication, and much more.

Nonetheless, the "traditional" view just sketched results in a crisp role distinction between the *creator* (and maintainer) of the laboratory on one side, and the users (students and instructors) on the other side.

There are of course many good motivation for such a *scenario*, two being in the authors' opinion the most important. One resides in the typically complex and expensive nature of the hardware/software architecture of a remote laboratory server. This is particularly true if physical experiments are involved, as witnessed by various examples like Guzmán et al. (2005), and is only partially mitigated in the case of simulated experiments, see, e.g., Sanchez et al. (2002), and from a more abstract and general point of view, the discussion reported in Gomes and Bogosyan (2009).

The other motivation, in some sense connected to the previous one, is the complexity of both server- and client-side applications, the development of which requires to join and coordinate control and software engineering expertise. This can be eased by the use of some technology addressing the "remote user interface" problem, such as the National Instruments LabVIEW [1] server. Anyway, the problem is not eliminated at all, especially if one wants clients to be able of changing the control law and not only its parameters – see Casini et al. (2005); Pastor et al. (2005) for an example – or even to host not just user interfaces, but also (part of) the controllers themselves.

However, the *panorama* is changing, at least potentially, as nowadays it is possible to create a networked control system – thus a remote laboratory – entirely with off-the-shelf devices and technologies. In fact, mobile applications can be developed and used to realise a "mobile laboratory", almost without any physical or computational constraint, since smartphones and tablets abundantly possess the necessary computational power and user interface richness.

By building on this idea, there is thus the possibility of having the role of who offers and who uses a "remote" laboratory played by the same people, which in turn opens a wealth of new didactic and pedagogical opportunities. This work, starting from some recently published ideas, presents – essentially in the form of a position paper – a first nucleus of functionalities, which enables a wide range of innovative teaching activities. The possibilities range from the use of the mobile application to make students to better understand concepts of basic control courses, to more advanced activities where students can create and test their own controllers or process simulators in an affordable way, and in a self-contained yet realistic simulation environment. A simple proof of concept is provided herein for the proposed ideas, and some words are spent on how it can be used to devise and deliver didactic activities at different levels.

[1] LabVIEW is a trademark of National Instruments Corp., Houston TX.

The paper is organised as follows: Section 2 reports a synthetic overview of the envisaged "mobile laboratory" architecture, while Section 3 describes the resulting structure of the typical application. Section 4 presents the basic types of activities that can be carried out by the students, Section 5 illustrates the envisaged developments, and finally Section 6 draws some conclusions and sketches out future work.

2. ARCHITECTURE OVERVIEW

The introductory considerations reported in Section 1 allow to summarise the needs as follows.

First, only off-the-shelf elements have to be used, and this applies both to the hardware devices and the software development environments; wherever possible, for apparent reasons, free tools are to be chosen.

Second, the architecture has to allow mobile nodes to interact both with other mobile nodes and/or a central server (generally not of the mobile type, since this situation is most likely to occur in the presence of physical experiments).

Third, although the presented activity lies in the domain of *control* education, it cannot avoid the necessity of some software development expertise. In fact, one of the more interesting (yet advanced) activities to propose to students is to modify existing simulators, or create from scratch their own controller/process one, and plug it into the application with a little (if any) integration effort with respect to the rest of the application. As a consequence, care has to be taken to make the unavoidable difficulties manageable by the "average" control student, i.e., comprehensible and solvable with the typical software engineering skills taught in (more or less) all control-centric curricula, by minimising and isolating as much as possible the fragments of code related to control and simulation topics.

After some considerations, the choice was made to ground all the parts of the architecture that result in mobile software on the Android operating system.

The main reason for that is the vast and increasing diffusion of Android in the mobile (particularly smartphone) market; this witnessed by the projected data of Figure 1, showing data published by Gartner, Inc. (2011).

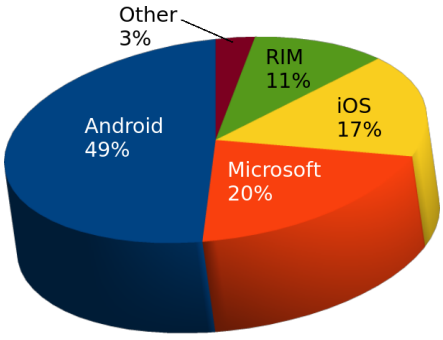

Fig. 1. Market shares for smartphone operating systems, 2015 projection.

A second one, somehow a consequence, is the expectable standardisation of development tools for Android, and also the creation of more user-friendly ones than those available to date.

Finally, the Android SDK (Software Development Kit) is based on Java with all the modularisation benefits brought with the object-oriented paradigm, allowing to isolate the control-relevant fragments of code as needed.

However, despite its apparent advantages, Android (but from this point of view virtually any mobile-targeted operating system) has also some pitfalls. Such systems are in fact endowed with quite complex an application interface, owing essentially to the necessity of maintaining prompt response to asynchronous events (like phone calls or messaging activities) while at the same time running interactive applications. The authors did not deem this difficulty strong enough to abandon the smartphone as the device of election, since it is sumptuously paid back by connectivity ease, but nonetheless the software development problems remain.

When non-mobile nodes are involved, of course, the operating system and development issues more or less vanish. However, given the expectedly increasing need for smartphone-to-base communication functionalities also in different domains with respect to control, it can be assumed that adopting Android will prove beneficial also from this viewpoint.

Coming to the development tools side, the decision was taken to adopt the standard Android SDK framework, which means developing everything in the Java programming language. At present, however, the major IDE (Integrated Development Environment) tools provide quite high-level debugging and device emulation functionalities, but for the same reasons above, are more targeted to the event-based programming typical of "standard" mobile applications than to the needs of control systems' implementation.

Incidentally, Android is quite widely used also in the context of computer education, including teaching operating systems (Andrus and Nieh, 2012), which means that quite a lot of didactic and support material is available, and more will be diffused in the future. In any case, in the application structure described in Section 3 as integral part of the proposal, care has been taken to properly isolate the simulation- and control-related code from the rest of the application, in a view to confronting control students with challenging but feasible objectives as sketched out above.

3. APPLICATION STRUCTURE

In this section we describe how the proposed mobile application is structured, and highlight the relevant parts for possible didactic activities.

The application can be structured as a classical client-server architecture. The process simulation is managed on the server-side, receiving control inputs by a client – the regulator – and communicating in turn its output (see Figure 2).

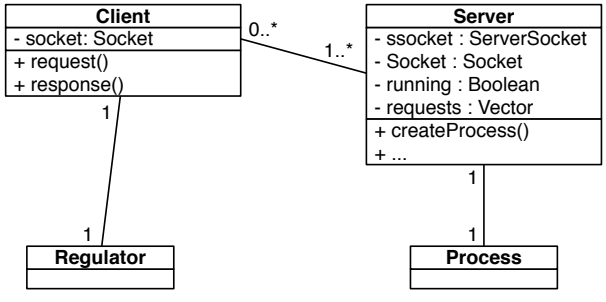

Fig. 2. Client-server architecture.

The advantage of using such kind of architecture is twofold. First, the process can have multiple outputs simulated in a centralised way by the server, and multiple control loops can be set up and managed by different controllers (clients) independently and in a distributed way. Second, the so designed architecture allows the client to connect to a *generic server* and receive some feedback measurements, in a totally transparent way, i.e., no matter of what is on the server side. In fact, the actor communicating with the client can be either another mobile device, or a web server simulating the behaviour of a physical system, or even a server connected to and acting on a real plant (see Figure 3).

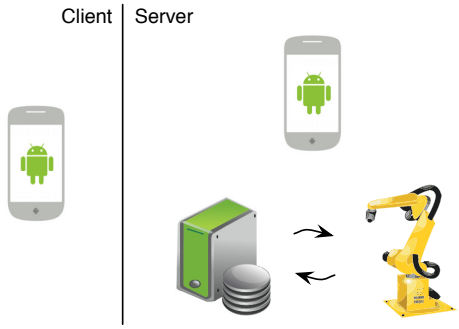

Fig. 3. Client-server interaction.

This is a first peculiarity of the proposed "mobile laboratory", since it opens different possibilities with a unique interface. Just to give an idea, considering the experiment categorisation proposed in Gomes and Bogosyan (2009), adopting this framework allows to cover 3 out of 4 possible experiments:

- the "local simulation", when two mobile devices are used for the simulation of the process-control interaction;
- the "remote simulation", when a mobile device connects to a remote server on which the plant behaviour is simulated;
- the "remote experiment", when a mobile device connects to a remote server which acts on a real plant,

and the only case that is left out is the "hands-on experiment", which, in some sense, stray from the concept of "mobile/remote laboratory".

Coming to some implementation details, the application can take a lot of advantage of the Object-Oriented approach, since there are essentially two entities into play: the process and the controller. Both of those entities are just "abstract" and depending on the user choices are instantiated accordingly. For example, the regulator can be a PI or a PID (see Figure 4), and the process can be a First Order Process (FOP), a First Order Plus Dead Time (FOPDT), or a Second Order Process (SOP), and so on (see Figure 5). However, this is just an example, and different types of controller and process can be implemented. In particular, the process is not constrained to be a transfer function, it can be a physical model with specified inputs and outputs of interest, encapsulating the simulation details in the class methods. If the simplification to simulate the model at fixed step is accepted – which is absolutely reasonable in the addressed context – the extensions needed by the class diagram of Figure 5 are straightforward.

Apparently, the sketched framework provides to date only a first nucleus of functionalities, but can be easily extended with more

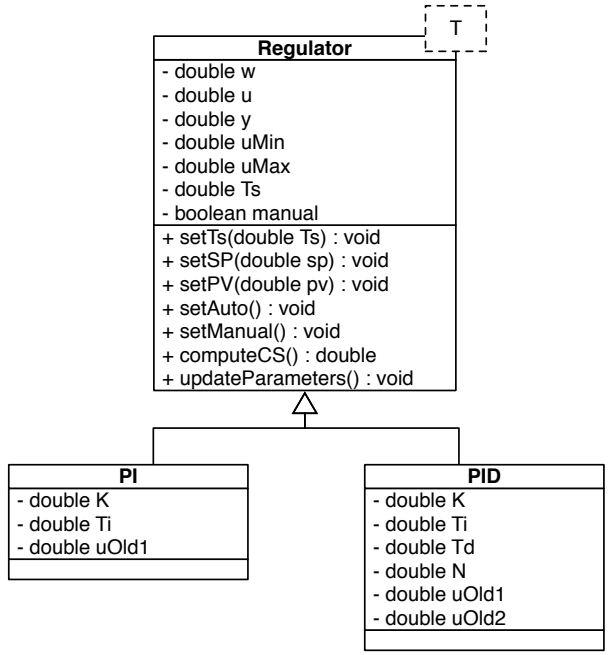

Fig. 4. Regulator class diagram.

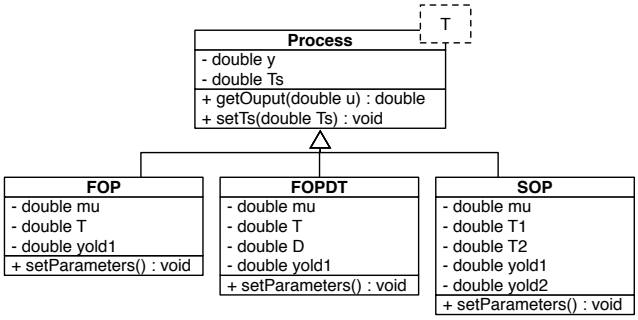

Fig. 5. Process class diagram.

classes, including more advanced control techniques, and more complex models.

3.1 Some use cases

In the development of the application, some simple, but significant use cases are considered, and here briefly presented.

The first use case is to use the mobile application as a remote regulator, and to control a process (simulated) on a different mobile device. A typical scenario is the following:

- The user can choose among different controller structures, e.g., PI or PID, and which server to connect via an IP selection.
- The user interface (Figure 6) allow the user to select the set point, and the regulator parameter values and to establish the connection through the "Start control" button.
- The user can see the control variable, and the measured output values by means of a some progress bars.
- The user can modify the parameter values online and see the consequences on the control variable and on the measured output.
- From the controller viewpoint other information is not relevant, thus not shown to the user.

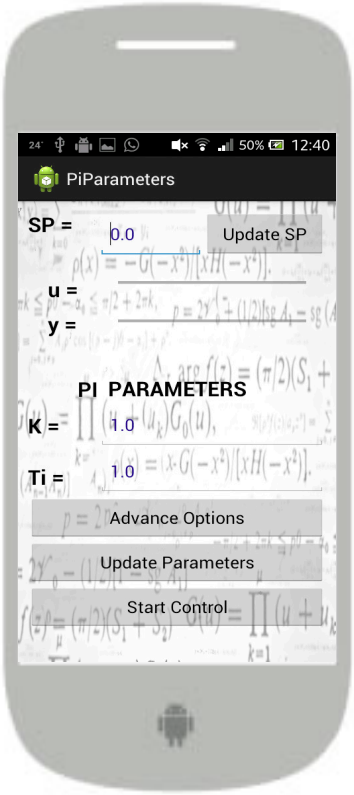

Fig. 6. User interface for a PI controller.

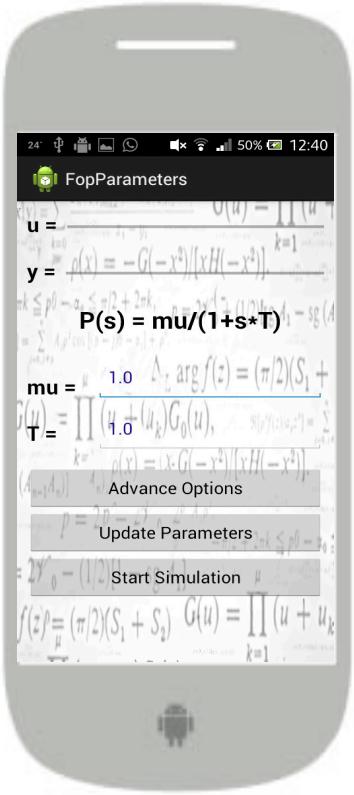

Fig. 7. Screenshot of the mobile application.

Another use case is the scenario in which the user selects the process instead of the regulator part on a mobile device (Figure 7).

- The user can choose among different process structures, e.g., FOP, FOPDT, SOP, and so on, depending on the implemented classes in the application.
- The user, pressing the "Start simulation" button, makes the application start waiting for a client request (i.e., the regulator request for the output value), and thus for control inputs.
- When the connection between the process and the regulator is established, the closed loop simulation starts.
- The user can change the model parameters, and reinitialise the process to the initial conditions.
- The user can stop the simulation at any time by pressing the "Stop simulation" button.

In both the presented cases, the students can get more insight on how the parameters influence the closed-loop behaviour, and its dynamic performance.

3.2 Isolating simulation and control code

As already stated, the application is written in Java, using the Android SDK, thus it is divided into classes that expose public methods (e.g., see again Figure 5) that can be modified without altering the rest of the project. From a teaching point of view, this is a fundamental aspect, since the students can modify very isolated parts of the application – the control-relevant ones – without taking care of the Android framework implementation details.

A first worth noticing example of exploiting this feature is to make students understand how to write the piece of code in charge of managing the control-loop signals (see Listing 1).

Listing 1. Managing the simulation.

```
thread = new Thread(
 new Runnable(){
  public void run(){
    double u;  // Control variable
    double y;  // Output variable
    try{
      while(true){
        // Read the input
        u = (double)uIn.getInput();
        // Update the output
        y = proc.getOutput(u);

        // Update the GUI
        progBar.setProgress((int)y);
        progBar.setText(
          Double.toString(y)
        );

        // Wait for the next sampling time
        Thread.sleep(
            (long)(proc.getTs()*1000)
        );
      }
    }catch(InterruptedException ex){
      return;
    }
  }
 }
);
```

This is a crucial skill to acquire, since a wrong implementation may lead to an unstable behaviour of the system, even if the control is designed correctly.

On the other hand, another important aspect is worth evidencing. In fact, in basic control courses, students are used with continuous-time systems, but are almost unaware of how those systems can be simulated. Listing 2 shows a very simple implementation of the Explicit Euler discretisation for a continuous-time First Order Process.

Listing 2. First Order Process class.

```
public class FOP extends Process {
        // Parameters
        private double mu;
        private double T;
        private double Ts;

        // Variables
        private double y;
  private double yo;

  /* ...
   * Other auxiliary methods omitted
   * ...
   */

  public double getOutput(double u) {
    double p = this.Ts/this.T;
    this.y = (1-p)*this.yo + this.mu*p*u;
    this.yo = this.y;  // save the state
    return y;
  }
}
```

The method `getOutput` can be easily modified with different discretisation rules, and make the students better understand the effect of this choice varying the sampling time T_s.

In both cases, this framework provides a "safe" hands-on experimental laboratory to make student practice coding techniques for control and simulation (Maggio and Leva, 2011), and see the effect of a their choices with the mobile application—in some sense covering also the last category of "experiment" presented in Gomes and Bogosyan (2009).

4. THE PROPOSED DIDACTIC ACTIVITIES

The activities that can be carried out by the students with the presented mobile laboratory architecture can be broadly divided into three categories.

The first and simplest category consists of just using some already developed experiments. In fact such an activity is very similar to many others already presented and illustrated in the literature, but with two differences. First, the use of a smartphone enhances flexibility in the fruition of the laboratory. Second and most important, a group of students can be given a complete laboratory, including both the experiment server side (in general simulated) and the client side. This allows for example to experiment in various conditions, experiencing connection and other environmental problems that more traditional laboratory installations are less keen to evidence. This activity can take place also in quite basic courses, and with a tendentiously high degree of student autonomy.

The second and more complex category consists of building a laboratory, by taking the modules already present in the library and composing them into new structures. This puts the students in direct contact with the software (and possibly hardware) architecture, so as to induce in future control experts also the knowledge of how solutions are realised in a mobile context—a potentially precious skill when it will be for them the time to interact with computer professionals. Also, this type of activity allows to build and experiment with articulated systems, possibly with more than one controller node acting on one plant node with multiple loops. This activity is suitable for more advanced (e.g., graduate) courses, and requires more strict a guidance (e.g., by first providing some examples and then periodically interacting with the students).

The third and most advanced category consists of actually participating to the development and enrichment of the software architecture. This can be substantiated for example into writing new modules, always taking care to ease the use of said modules on the part of people who will compose them into new applications. Doing so inherently leads to learn how to suitably enforce the encapsulation of android-specific (or more in general, architecture-specific) facts when seen by the developer. Such an activity is most likely to be carried out in the form of a thesis, as quite strict an interaction is necessary with the instructor, to actually grasp the involved interdisciplinary competences.

5. FUTURE DEVELOPMENTS

At present, the developed software is just a nucleus, since the choice was made to concentrate on the design of its architecture, of which a sample has been explained in Sections 2 and 3. A first implementation is released as free software within the terms of the GNU General Public License version 3 (GPLv3) [2], so as to share experience and ideas with the community *de facto* at the very beginning of the library realisation. Nonetheless, developments for the near future are already scheduled, and can be summarised as follows.

A first "reasonably complete" set of blocks has been envisaged. These include of course transfer functions to simulate simple processes, but also controllers realised with as "industrial" as possible an attitude, i.e., taking profit of the pursued control code isolation – see the discussion reported before – to include functionalities realised along the guidelines provided in Maggio and Leva (2011), and including advanced ones like the autotuning PIDs for use in remote laboratories described in (Leva and Donida, 2008b). Also, simulators of simple processes based on first-principle, and also nonlinear equations will be included, such as for example small thermo-hydraulic plants. This will enrich the students' experience and its realism, allowing also to experiment with control structures (feedforward compensation, cascade, decoupling, and so forth).

On a tightly related front, a set of user interface modules is being designed, to allow the developer of an application to easily realise such interfaces with a control-oriented look and feel (i.e., including graphs, trends, and the like). This is important to further help isolate control code and streamline the development of new applications, since Android has a number of useful features – whence its choice apart from the dominant

[2] It can be downloaded from `home.deib.polimi.it/leva/MobileLabs.html`

and increasing diffusion – but the widget sets available in the major IDEs for it, are not conceived for control applications.

Then, the possibility is being explored to have servers with physical experiments, both based on PCs and composed of very small, self-contained apparatuses that one may connect to a laptop or tablet. The goal is to have student groups who will run a complete remote laboratory, server included, to actually experience the encountered difficulties. Here too, the experience gathered in the past, and described in several of the quoted papers, will be exploited, together with exploring the possibilities nowadays offered by powerful and quite inexpensive development boards, like for example the Arduino or the Raspberry Pi. Using such boards will of course complicate the design (an activity at present not expected to be included in the didactic one presented herein, however) owing to the necessity of having them deal with the physical equipment on one side and interact with the Android node on the other. Nonetheless, the development ease and connectivity possibilities of such devices should make the idea feasible, and reasonably straightforward to realise.

Moreover, plans are to study possible integrations of the presented Android-based architecture with other frameworks for networked laboratories. In this respect Easy Java Simulations (Sánchez et al., 2005) is a natural candidate – on the server side, quite intuitively – since it is Java-based, but by designing and realising convenient wrappers to interface the architecture's Java code to heterogeneous modules, one might consider also the numerous other alternatives available in the literature.

Finally, documentation and didactic material is being prepared, and is expected to grow together with the library. Needless to say, the authors hope that once the tool is released, the community will help by providing cooperation, ideas, and criticisms to improve the overall environment and make its use most fruitful.

6. CONCLUSIONS

Building on some previously proposed ideas and results, a first nucleus was presented of an Android-based architecture that allows to create process simulators and controllers on mobile devices, and then share them with others to compose a complete, mobile "remote" laboratory.

After motivating the research, the architecture was described with quite a high-level attitude, and an example of how it can be used to design applications was sketched out.

Some possible activities were then devised, illustrating how they can address students at various levels of competence within a unitary pedagogical path, and pursuing goals suitable for all the mentioned levels.

The proposal is apparently just the beginning of a long-term work, that was also briefly illustrated. The authors hope that the reported ideas and considerations will stimulate the community to cooperate to what they believe to be a very promising topic in control education.

REFERENCES

Andrus, J. and Nieh, J. (2012). Teaching operating systems using android. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, 613–618. ACM, Raleigh, NC, USA.

Casini, M., Leva, A., and Schiavo, F. (2005). AIRES: a standard for web-based remote experiments. In *16th IFAC World Congress*. Prague, Czech Republic.

Djalic, V., Maric, P., Kosic, D., Samuelsen, D., Thyberg, B., and Graven, O. (2012). Remote laboratory for robotics and automation as a tool for remote access to learning content. In *Interactive Collaborative Learning (ICL), 2012 15th International Conference on*, 1–3. IEEE, Villach, Austria.

Dormido, S., Vargas, H., Sánchez, J., Dormido, R., Duro, N., Dormido-Canto, S., and Morilla, F. (2008). Developing and implementing virtual and remote labs for control education: The uned pilot experience. In *17th IFAC World Congress*, 8159–8164. Seoul, South Korea.

Gartner, Inc. (2011). Gartner says android to command nearly half of worldwide smartphone operating system market by year-end 2012. URL `http://www.gartner.com/newsroom/id/1622614`.

Gomes, L. and Bogosyan, S. (2009). Current trends in remote laboratories. *Industrial Electronics, IEEE Transactions on*, 56(12), 4744–4756. doi:10.1109/TIE.2009.2033293.

Gomes, L. and García-Zubía, J. (2007). *Advances on remote laboratories and e-learning experiences*. University of Deusto.

Guzmán, J.L., Berenguel, M., Rodríguez, F., and Dormido, S. (2005). Web-based remote control laboratory using a greenhouse scale model. *Computer Applications in Engineering Education*, 13(2), 111–124. doi:10.1002/cae.20035.

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

Leva, A. and Donida, F. (2008b). A remote laboratory on pid autotuning. In *17th IFAC World Congress*, volume 17, 8147–8152. Seoul, South Korea.

Maggio, M. and Leva, A. (2011). Teaching to write control code. In *18th IFAC World Congress*, 7292–7297. Milan, Italy. doi:10.3182/20110828-6-IT-1002.01026.

Murtra, M., Jansa, G., Martinez, H., Domingo, J., Gámiz, J., and Grau, A. (2007). A proposal of remote laboratory for distance training in robotic applications. In *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, 1180–1187. IEEE, Patras, Greece.

Pastor, R., Martín, C., Sánchez, J., and Dormido, S. (2005). Development of an xml-based lab for remote control experiments on a servo motor. *International journal of electrical engineering education*, 42(2), 173–184.

Sánchez, J., Esquembre, F., Martín, C., Dormido, S., Dormido-Canto, S., Canto, R., Pastor, R., and Urquia, A. (2005). Easy java simulations: an open-source tool to develop interactive virtual laboratories using matlab/simulink. *International Journal of Engineering Education*, 21(5), 798.

Sanchez, J., Morilla, F., Dormido, S., Aranda, J., and Ruiperez, P. (2002). Virtual and remote control labs using java: a qualitative approach. *Control Systems, IEEE*, 22(2), 8–20. doi:10.1109/37.993309.

Vicente, A.G., Munoz, I.B., Galilea, J.L.L., and del Toro, P.A.R. (2010). Remote automation laboratory using a cluster of virtual machines. *Industrial Electronics, IEEE Transactions on*, 57(10), 3276–3283.