IEEE Report
Open virtual and remote labs for Control Education

S. Dormido, J. Chacón, J. Saenz, L. de la Torre
Universidad Nacional de Educación a Distancia, Madrid, Spain.
e-mail: {sdormido,ldelatorre}@dia.uned.es,{jchacon,jsaenz}@bec.uned.es

March 2018

Contents

Part I
Report

1 Introduction

Control engineers need to have both a wide experience implementing solutions in real problems, plants and processes and a deep understanding of the mathematics and theory that lie behind these solutions. Therefore, reaching a balance between theoretical proofs and physical intuition is a major challenge in control education. Lab experimentation plays a key role as a way to connect theory and practice [7, 22]. Among others, lab experience serve as [2]:

- An introduction to real world modeling and/or control issues, such as uncertainties, saturation, noise, sensor/actuator dynamics, etc.
- A demonstration or validation of analytic and theoretical concepts.
- A way of providing facility with instrumentation and measurement tools.
- Activities for problem solving and team learning.
- A motivational activity.
- A way to develop professional practices, including maintaining engineering notebooks and report writing.

On the downside, traditional hands-on labs entail high costs related with space requirements, equipment, and maintenance staff [17]. For the last twenty years, there has been a line of research that looks for reducing lab costs by taking advantage of the Internet, i.e., by replacing hands-on labs with online ones. Depending on the nature of the resource (the plant/process) an online lab can be either virtual or remote [14]:

1. *A remote lab* is a real plant that can be accessed through the Internet. Students remotely operate and control a real plant through an experimentation interface.
2. *A virtual lab* is similar to the previous one, but it replaces the physical system with a mathematical model.

Usually, the approach when setting up a new remote laboratory on control is to start from an already working hands-on experiment, and then enable a way to accessed it from a remote computer: an attractive graphical user interface, security access control, etc. There are many examples of this approach in literature ([4, 12, 6, 13, 20, 15, 8, 27]). Commercial control experiment systems, however, are expensive, tend to be large and heavy, and in many cases are not very flexible. On the contrary, the teaching needs are subject to changes. For example, a teacher may want to use the system in a level different than the one it was originally designed for. There are many experimentation systems which lack of use after some years, despite the economical investment. Certainly, after investing hundreds (or thousands) of euros in equipment, it is not easy to discard it simply because it does not fit exactly with our needs.

One of the goals of this work is to provide a complete open-source and open-hardware remote lab solution that is low-cost and easy to replicate, so that anyone who intends to build it can use it either as a ready-to-use system or as a starting point to introduce custom modifications. For that purpose, there are two essential elements:

- Open-source/open-hardware tools.
- Rapid prototyping technologies. We have witnessed the rise of 3D printing or single board computers, technologies that fit like a glove to our purposes.

The designs of the 3D printed parts used for building the lab presented in this work are free to use and modify, so the costs of cloning most of the system's structural elements is marginal. Moreover, the designs use components that either can be gathered from old electronics devices or are cheap and easy to find. Building a control experiment system from scratch is a demanding and time-consuming task. Sometimes, the use of low-cost components such as sensors or actuators must be compensated with creativity, or loss of performance. However, looking at the success of many community-driven open-source projects (RepRap 3D printers (`http://reprap.org/`), PublicLab (`https://publiclab.org/`), Thingiverse (`http://www.thingiverse.com/`)), it is not unrealistic to think that collaborating between laboratory designers could really enhance the teaching experience in control engineering as well as in other areas.

Given the complementary uses of virtual and real experimentation [30, 1, 19], the authors have also developed a virtual version of the system. This work presents a low-cost virtual and remote lab implementation of an air levitation system, based on open solutions. It can be easily adopted to be used as both a remote lab and as a hands-on lab. Air levitation is the process by which an object is lifted without mechanical support in a stable position, by providing an upward force that counteracts the gravitational force exerted on the object. In addition to the study of air levitation physics, we found interesting to create a virtual and remote lab of the air levitation system for three reasons:

- The rapid dynamics makes it specially suitable for experiences with a control lab.
- Due to the system nature, the design can be optimized and, with some precision tradeoff, kept affordable both in cost and construction efforts.
- The realistic physics of the system are complex and the system is better modeled through an identification process. However, a simple physics model approximation can also be used and compared with the behavior of both the real system and the model obtained through the identification process.

1.1 Approach

Recently, low cost single board computers such as Raspberry Pi or Beaglebone Black, and 3D printing technologies, which allow for rapid prototyping of mechanical systems, have become pervasive. These tools provide an interesting framework that can assist the creation of remote labs. The hardware framework is complemented with reusable software components, a web-based architecture, and standard communication protocols to reduce the development cost and effort. Based on this paradigm, an easily replicable remote lab architecture is proposed, using 3D printed parts designs that have been open source licensed to allow for free use or modification, as well as software components that implement the different subsystems of the lab, and elements that either can be gathered from old electronics devices or are cheap and commonly used components.

Currently, a virtual and remote lab of an air flow levitation system has been built using the proposed methodology. It consists of a small object that has to be lifted using the air flow generated by a fan inside a cylinder. The position of the levitating object is measured with an infrared distance sensor and it is used to control the rotation speed of the fan. The prototype will be incorporated into a master degree course in control engineering. Some of the benefits expected from the experience are to provide students with a global insight of engineering processes, increase their motivation to research about different sensing technologies, and promote their creativity.

In recent times, it is not unusual that students, even of first courses of engineering, have at least basic knowledge of the mentioned development platforms, and a good predisposition to use them. In spite of that, the popularity of 3D printing technologies and the do-it-yourself (DIY) and the maker community can be an attractive way of drawing the students' attention. As an example of a similar approach, some universities already proposed robotic competitions where students are asked to solve some problems using basic construction kits. These activities, which have demonstrated to benefit students' development, are not very different in nature compared to the one proposed in this work. Therefore, it is expected to obtain great profit from the Air Levitation System not only as a hands-on lab or a remote lab, but also to have students built their owns, which is a useful experience in itself.

2 Design

The air levitation system presented in this work has a minimalist design, because any increase in complexity has an effect in terms of cost and effort and the system is meant to be affordable and easy to replicate. The system is robust enough to work as a remote lab. It is composed of a cylinder in which a forced air flow is used to lift a small object levitating on a desired position (Figure 1).

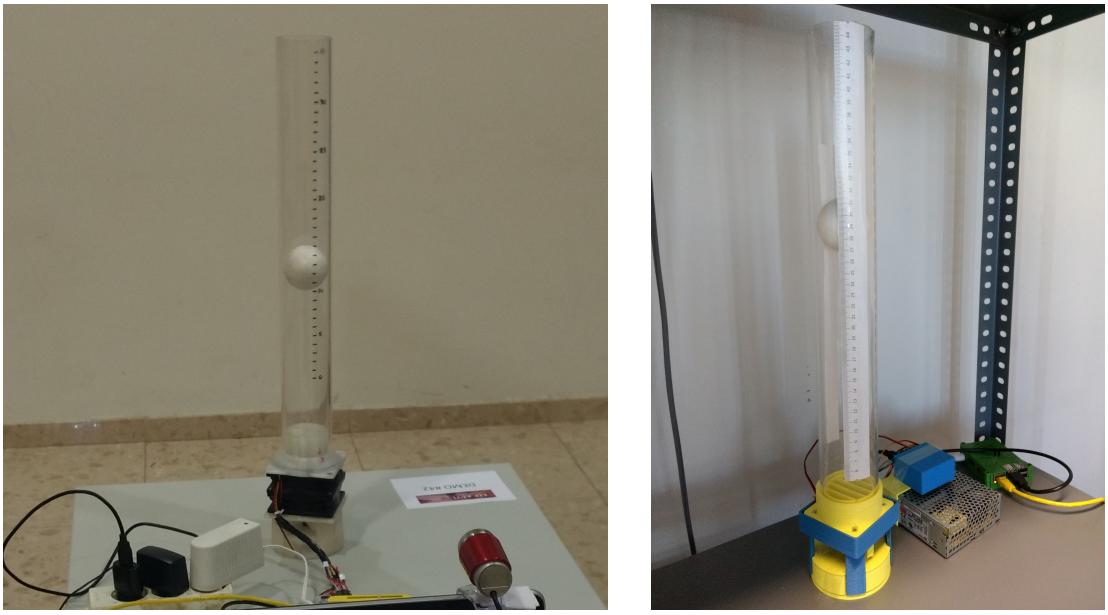
Before describing the design and construction of the plant, it is worth to mention that the low cost and optimized design of the system enables two different approaches. Usually, the laboratory creator/maintainer build the system, remote lab, and any other resource needed. Another approach is to let students build their own systems, so they can get a learn-by-doing understanding of a thorough engineering process. Converting an idea or a concept to a practical solution is essential in engineering.

The construction of the system has been decomposed in several tasks:

1. *Design of the experience*;
2. *Design and construction of the plant*;
3. *Design and construction of the server software*;
4. *Creation of the graphical user interface (GUI).*

The structure is simple on purpose, there are only a few elements: a methacrylate tube with a nozzle, at one end, coupled with a blower fan. Both elements are supported by an open and movable stand, which let the air flow into the fan. The system has been built using only the following components:

- A methacrylate tube.
- A small and light object.
- 3D printed parts.
- A single-board computer (Beaglebone Black)



(a) (b)

Figure 1: The Air Levitation System in a) a demo session at *exp@at'17*, and b) the authors' laboratory.

- An infrared distance measuring sensor (PIR).
- A PC fan.
- Some discrete electronic components and a printed circuit board (PCB).
- A webcam.

All the design files, documentation about the system and instructions about how to build and setup a new replica is freely available on `https://github.com/jcsombria/OpenHardwareLabs`.

2.1 Hardware

Since the release of the first Raspberry PI model, a bunch of single board computers have appeared intending to fit developers needs, which range from small do-it-yourself (DIY) projects such as home media centers or domotic appliances, to high performance research computing. Most of these boards are specifically focused on the maker community, students and educators, so they are fully open-source hardware.

An interesting feature of these single-board computers is their ability to run a complete operating system (OS). As an example, a *Raspberry PI* can run several *Linux* distros (*Raspbian*, *Ubuntu*, *LibreElec*, etc.), Windows 10 IOT Core, or RISC OS, immediately opening an universe of possible applications: it is easy to set up a web server, enable remote connections through Secure Shell (SSH) or even graphical sessions, or use many different programming languages to develop our project. Furthermore, the integrated input/output (IO) capabilities through general purpose input/output (GPIO)

pins: digital IO, interconnection protocols (SPI, I2C, etc), or AD converters makes easy (and affordable) to build electronics systems, even if not an expert in the subject.

These single board computers tend to provide similar performance. The most popular ones, like Raspberry PI or Beaglebone Black, are based on an ARM architecture, presenting differences like the RAM size, input/output capabilities or wireless connectivity. For example, Raspberry PI provides digital IO, SPI, UART, and I2C connectivity, but it does not have integrated analog IO. On the contrary, Beaglebone Black has analog IO but does not provide built-in WiFi or Bluetooth. Table 1 summarizes the capabilities of four representatives platforms, covering a wide range of costs and functionalities.

	Onion omega 2	Raspberry PI 3	Beaglebone Black	Intel Galileo (Gen 2)
Cost[1]	20€	35€	50€	75€
SoC	400 MHz MIPS 24Kc Big-Endian Processor	Broadcom BCM2837	ARM	Intel Quark SoC X1000
RAM	64MB DDR2 (400Mhz)	1GB LPDDR2 (900 MHz)	512MB DDR3L (800Mhz)	256MB DDR3 (800Mhz)
Wireless	WiFi	WiFi, Bluetooth	n/a[2]	n/a
GPIO	UART, SPI, I2C, PWM, digital IO	UART, SPI, I2C, PWM, digital IO	UART, SPI, I2C, CAN, PWM, digital IO, A/D inputs	UART, SPI, I2C, JTAG, PWM, digital IO
Ports	Universal Serial Bus (USB), WiFi, (more options availaible with expansion boards)	HDMI, 3.5mm analogue audio-video jack, 4xUSB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)	HDMI, USB, Ethernet	USB, PCIe, Ethernet

Table 1: Comparison of low-cost development platforms.

The air levitation system presented in this work is controlled by a Beaglebone Black board, running a GNU/Linux distribution. The choice of this board for developing the air levitation system was based on several reasons, namely:

- All boards ship with the Debian GNU/Linux image. This image comes with pre-installed software tools and, in particular, it provides the Node.js runtime and the Cloud9 IDE (Integrated Development Environment). Also, the bonescript library (included in the Node.js installation) provides an Arduino-like *application programming interface* (API) to access the GPIO, so any person with previous experience in Arduino finds a soft learning curve.
- The GPIO provides analog inputs that can be used to acquire the sensor measures, and PWM outputs to control the fan and the servo of the air levitation system.

- The board have and on-board eMMC memory, eliminating the need of an external SD card memory.
- There is an active development community. The Beaglebone boards have good hardware/software support and it is easy to find documentation, guides, etc.

It is important to highlight that, while the Beaglebone Black is used in our experimental setup, the other alternatives not only are also suitable but they are entirely compatible with the software and hardware described in the next sections. In fact, since the boards have USB ports, they even can be connected to Arduino boards or other peripherals to add compatibility, reuse other designs or extend the capabilities of the board.

The Beaglebone Black board provides built-in A/D converters to read analog signals. Since the range of the voltage signal provided by the sensor (PIR) lies outside the one admitted by the analog inputs of the board $(0, 1.8V)$, it must be adapted before being connected. Similarly, the actuators (fans) require voltages and currents that can not be directly handled by the board, so a signal conditioning circuit has to be used.

Most structural elements have been printed in a *Prusa Mendel i3* 3D printer, a very popular and affordable *RepRap* printer, available at the authors' department. The 3D parts has been modeled with FreeCAD.

2.2 Sensors

To measure the position of the ball, several possibilities were considered: visual recognition, ultrasonic sensors, and infrared sensors. While it is interesting to use a video cam to get the ball position, and even could be adequated for teaching in an image processing subject, the complexity and cost of the system would increase, so it was discarded. With respect to the ultrasonic distance sensors, they are a valid alternative as the infrared ones, similar in cost and complexity. However, the latter option was finally chosen. The position of the ball is measured with an infrared beam sensor, particularly a Sharp GP2Y0A21YK0F Analog Distance Sensor (Sharp Corporation), which can obtain measures between 10 and 80 cm. There are other similar models that are electrically compatible and have different ranges, as the GP2Y0A21YK0F (4-30 cm) and the GP2Y0A02YK0F (20-150 cm), so it can be chosen to adapt to different tube lengths. All the aforementioned sensors are analog, yielding a signal roughly in the range (0-5 V), which is proportional to the inverse of the distance measured. The sensor is composed of two IR LEDs, an emitter which projects a light beam, and a receiver that measures the bounce in the detected object. Since the sensor actually measures the light reflected by the object, it may be affected by the color, shape and movement of the object. Also, it has an update period of approximately 40 ms. These aspects must be taken into account to get a reliable measure. The BeagleBone Black Board has analog inputs which admit a value in the range (0-1.8 V), so the sensor output has to be adapted to that range, which can be done with an op.amp. based circuit. Figure 2 shows the calibration curve corresponding to the GP2Y0A21YK0F sensor.

Since, as mentioned before, the actual map between voltage and position depends on several factors, the sensor must be calibrated with the working conditions. The calibration process was:

- Fix:
  - Measurement range ($h_{min} = 20$ cm ¡ h ¡ $h_{max} = 40$ cm).
  - Measurement interval ($h = 1$ cm).

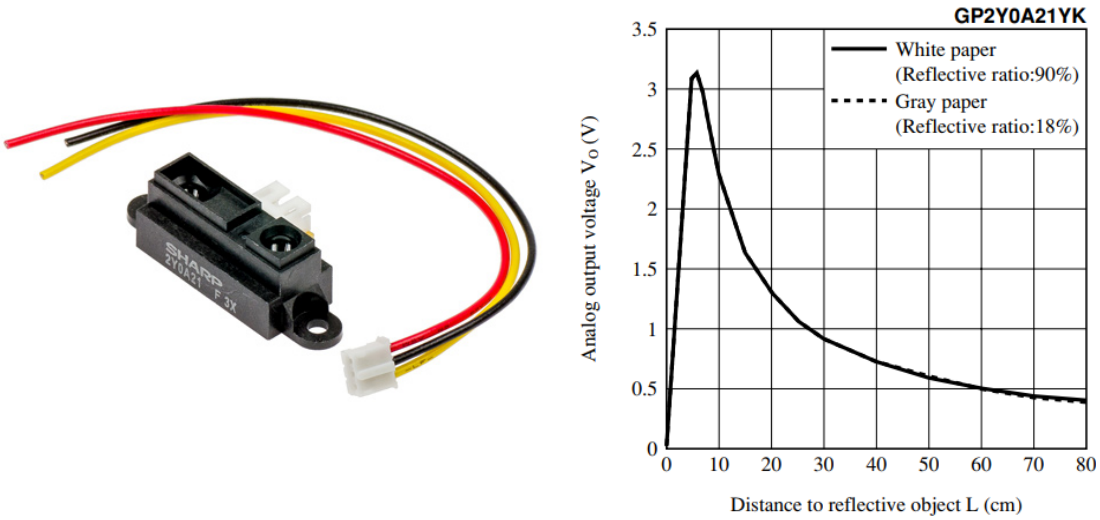


Figure 2: a) Sharp GPY2Y0A21YK infrared distance sensor and b) voltage vs. distance plot.

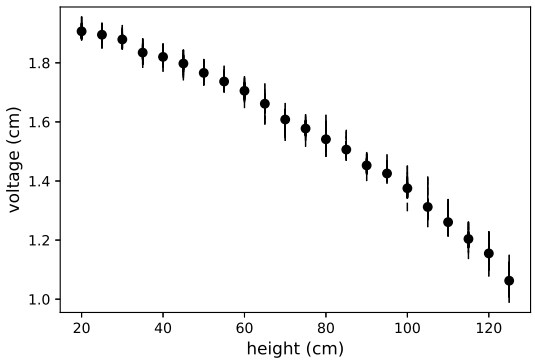
- Repeat, for each height (steps of 1 cm):
  - Put the ball fixed at a known level.
  - Record the sensor voltage for t seconds.
  - Calculate the mean voltage value and store $v_i \rightarrow h_i$.

In a first approximation, there was a problem with a local minimum. Looking at the curve of Figure 2b, it can be seen that for very short distances the voltage grows until around 3 V, and after that it monotonically decreases until the maximum distance. That was not the case of the measured response, which decreased at around 15 cm, after that increased until 20 cm, and finally it decreased again. It was a very problematic issue because, in order to avoid that unwanted behaviour, the operating would have to be drastically decreased. After detecting that the strange response was due to the reflection on the tube, the solution adopted was to add two coloured strips inside the tube. Figure 3a shows a plot with the measured voltage vs. distance, and the table in Figure 3b contains the voltage ranges corresponding to some distances.

2.3 CAD Software

Computer Aided Design (CAD) tools assist the designer to model the physical components which will be part of the system, in our case the structural parts and the electronics circuits. It is out of the scope of this work to discuss the pros and cons of the so many options available. However, it is worth to mention at least some of the most popular open-source alternatives that cover the lab needs: FreeCAD, OpenSCAD, KiCad EDA.

FreeCAD is an open-source 3D CAD software tool very popular among the 3D printing community. It has many features, parametric design, multiplatform (works on Linux, Windows and Mac), a fully



height	voltage
20 cm	1.92 ± 0.014 V
22 cm	1.88 ± 0.014 V
24 cm	1.82 ± 0.015 V
26 cm	1.77 ± 0.016 V
28 cm	1.71 ± 0.017 V
30 cm	1.61 ± 0.021 V
32 cm	1.54 ± 0.023 V
34 cm	1.45 ± 0.015 V
36 cm	1.38 ± 0.022 V
38 cm	1.26 ± 0.024 V
40 cm	1.15 ± 0.027 V

Figure 3: a) Measured voltage vs. distance, and b) measured voltage ranges for some distances.

customizable GUI, and native support for python scripting and extensions.

OpenSCAD is another popular tool, mostly used to design 3D printed parts. Unlike FreeCAD, it uses a non-graphical with a different modelling approach. It is based on a specific description language, so the creation process is more similar to traditional programming. One of the advantages of this approach is the flexibility to parameterize designs.

The electronic circuits and the PCBs has been created with *KiCad EDA* (`http://www.kicad-pcb.org`), a multiplatform and open-source tool that have the support of the *CERN*, which started the *KiCad EDA* project and have made important contributions to it as part of the *Open Hardware Initiative* (OHI) [3].

As in the case of 3D printing, there are many PCB manufacturers where you can send your circuit design and have your PCB with professional quality and a moderate cost or, following the maker paradigm, you can build your own circuit with a *computer numerical control* (CNC) PCB milling machine or a chemical etching process.

2.4 Design and construction of the server software

The software in the target computer must implement several capabilities, including: *Hardware interface*, *Datalogging*, and *Communication* and *Control* subsystems.

The overall picture of the system is represented in Figure 4a,b shows a detailed diagram with the RIP software architecture. The implementations of the aforementioned subsystems map to several Node.js objects, which interact to provide the desired functionality, as follows:

- The *hardware interface* is implemented in *Node.js* by the object *BoardInterface*, which provides a common interface to access the boards, and the boards objects actually implementing the low-level communication. At the moment, only the *Beaglebone Black* and *Arduino* boards have been implemented, but there will be support for other boards in the future. These objects provide several methods to work with the hardware.

- The *datalogging* is implemented by the *Node.js* singleton object *Datalogger*, which gathers the

[3] `https://home.cern/about/updates/2015/02/kicad-software-gets-cern-treatment`

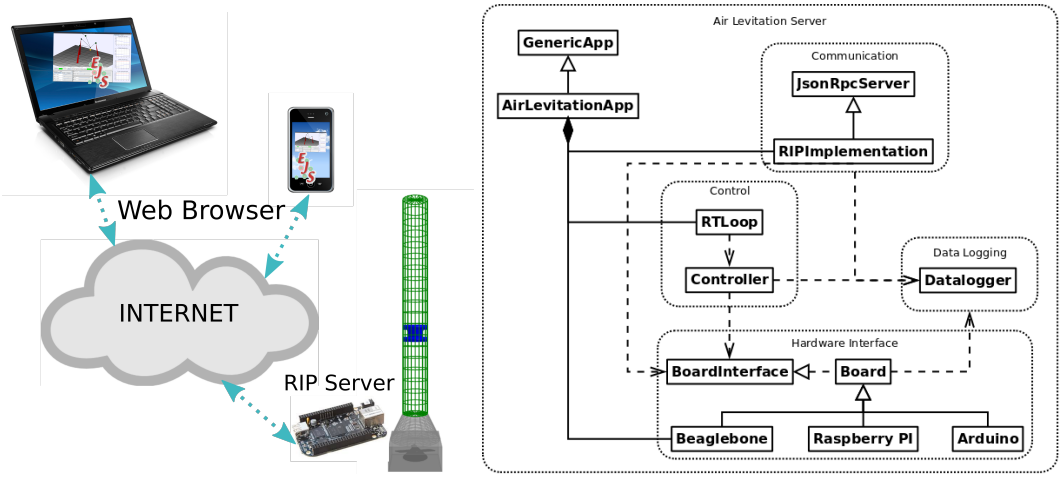


Figure 4: a) Overall view of the system, and b) RIP software architecture.

important data and sends it to the database server. Currently, the data can be logged to a local file or sent to an InfluxDB server.

- The *communication* is implemented in Node.js by the object *JsonRpcServer*, which provides basic functionality to create a JSON-RPC 2.0 server, and *RIPServer*, which makes use of the former to implements the API of the *Remote Interoperability Protocol* (RIP). These approach can also be used to easily define new protocols or adapt to other implementations.

- The *control* subsystem is implemented by the object *RealTimeLoop*, which defines the controller implementation.

Since the server software has been designed to be reconfigurable, there will be diferent implementations of some of the subsystems. In particular, the hardware interface is obviously highly dependent on the hardware, so in case a different board is used, a new implementation of that part would be needed. To specify which particular implementation should be used, there is a configuration file that allows to create the experiment by interconnecting components, defining the input and output, transport type, etc. As an example, the configuration of the Air Levitation experience is shown in Figure 5.

The flexibility of the software allows to interoperate with other solutions. One of the reasons to choose Node.js was because it can be easily extended to add new functionality. For example, there are library modules that implement solutions like Profibus, Modbus, MQTT, and many others protocols, and which could be incorporated to our architecture, thus expanding the interoperability.

2.4.1 Hardware interface

The *hardware interface* purpose is to read measures from the sensors, and send values to the actuators. Though it is obviously very platform dependent, it is a good practice to use standard libraries and protocols. For example, the Arduino API is widely used for its simplicity and it has been exported to other hardware, like the Beaglebone boards or Raspberry PI. The functionality to be covered can usually be reduced to read and write digital or analog input and outputs.

```
var conf = {
  server: {
    rip: 'RIPServer',
    transport: 'HttpServer',
  },
board: {
    require: 'BeagleBoneBlackBoard',
    name: 'Beaglebone Black Board',
    variables: [
      { 'name': 'ball_height', 'pin': 'P9_36', 'type': 'in' },
      { 'name': 'fan_control', 'pin': 'P9_14', 'type': 'out' },
      { 'name': 'servo_control', 'pin': 'P9_22', 'type': 'out' },
      { 'name': 'setpoint', 'type': 'in_out' },
      { 'name': 'kp', 'type': 'in_out' },
      { 'name': 'ki', 'type': 'in_out' },
      { 'name': 'kd', 'type': 'in_out' },
    ],
  },
}
var App = require('./GenericApp');
App.init(conf);
App.start();
```

Figure 5: Configuration File

In the Air Levitation System, the hardware interface task is accomplished using the *bonescript* library, which basically mimics the Arduino API to cope with Beaglebone and the GPIO pins of the board. There is a *real-time loop* implementing the time critical actions: read sensors, update the controller and write outputs. Technically, it is not actually real-time, because currently it is not supported by the *Node.js bonescript* library. But for the time scale of the system, which is sampled at a $100ms$ rate, it performs correctly. In case hard real-time is needed, there are other alternatives (such as C++) supported by the Beaglebone board.

2.4.2 Datalogging

Once the values have been acquired, it is needed to store them in order to be accessed whenever be required. For that purpose there are many options, but again it is recommended to use a standard solution. There are time series database systems (TSDB) that are specialized on time series management, such as *InfluxDB*, *graphite*, *OpenTSDB* or *RRDtool*. The *datalogging* capabilities have been separated into a low priority task that periodically dumps measures and control actions to a database, so the data is stored and can be accessed to perform off-line processing of past sessions.

2.4.3 Communication

The server software, running at the target platform (the single-board computer) must provide an API to interact with the system. The *Remote Interoperability Protocol* (RIP) has been proposed to interconnect engineering systems with user interfaces. It is a simple API based on the JSON-

RPC 2.0 protocol which is human-readable and can be easily integrated with *JavaScript* applications, as it uses the *JavaScript Object Notation* (JSON) to encapsulate *remote procedure calls* (RPC). The *communication* subsystem to make the server functionality accessible from outside of the lab computer implements the RIP [5], which provides a standard API to control and monitor the hardware. That basically means that any RIP enabled application can easily interconnect with the server to read and modify variables and plant parameters, so it is easy to decouple the GUI design from the rest of the system.

2.4.4 Control

The remote labs have a local controller implemented, which can be as simple or as sophisticated as needed. In the case of a control engineering lab, it must be a central part of the design, but even in other cases it is always needed to take some safety measures, to assure that the system cannot be harmed by accident or by a malicious user. The *control* subsystem implements a *proportional-integral-derivative PID* controller which parameters can be modified and tuned. The control subsystem is prepare to be extended with more sophisticated controllers without much development effort. More detailed information is provided in Section 3.

2.5 The Virtual and Remote Lab

EjsS is an open source authoring tool designed to easily create interactive simulations with a GUI for users with no programming skills. EjsS allows users to create applications in both Java or Javascript. Many virtual and remote lab (VRL) applications have been developed using EjsS [16, 4, 9, 10, 3, 6, 11, 24, 21, 18, 23] and some of them explicitly define it as a tool that facilitates the development of applications by researcher, teachers and students who want to focus in the simulation theory and not in the technical programming aspects [16, 4]. The use of interactive simulations and computer based modelling for teaching physics concepts is described in [10], and a complete discussion of remote labs and their benefits for teaching physics and enginering is available in [11]. [6] presents a virtual and remote laboratory of mobile robots where EjsS is used in combination with LabVIEW (National Instruments) and MATLAB (MathWorks), and in [16] a new approach to create interactive networked control labs is described. A remote control laboratory based on EjsS, Raspberry PI and Node.js is described in [3]. The authors of [24] present an ongoing schema to develop virtual models of physical setup equipment and their integration into the corresponding remote laboratory. In [21], students experiment with a set of hands-on exercises about Automatics and Robotics using *RobUALab*, a virtual and remote laboratory develoed in EjsS, firstly in face-to-face classes and afterwards accessing the online experimentation environment. [18] presents the design and implementation of a network for integrating Programmable Logic Controllers (PLC), the Object-Linking and Embedding for Process Control protocol (OPC) and EjsS, and [23] presents a set of open-source software tools and low-cost hardware architectures are proposed that allow an easy access to local or remote sensors and actuators integrated in EjsS. A systematic approach for developing web-based experimentation environments for control engineering education is presented in [29].

A nice teaching approach is to provide students with both the virtual and the remote version of an experiment. Both versions of the air levitation system presented here, share a simple and clean layout and most of the interface is similar. There is a view of the system on the left (consisting on a video stream obtained from the laboratory webcam in the remote version and on a 3D model in the virtual one). Graphs on the right show the evolution of the interesting variables (the height of the lifting object, the setpoint and the control signal sent to the fan). Finally, there is a control panel at

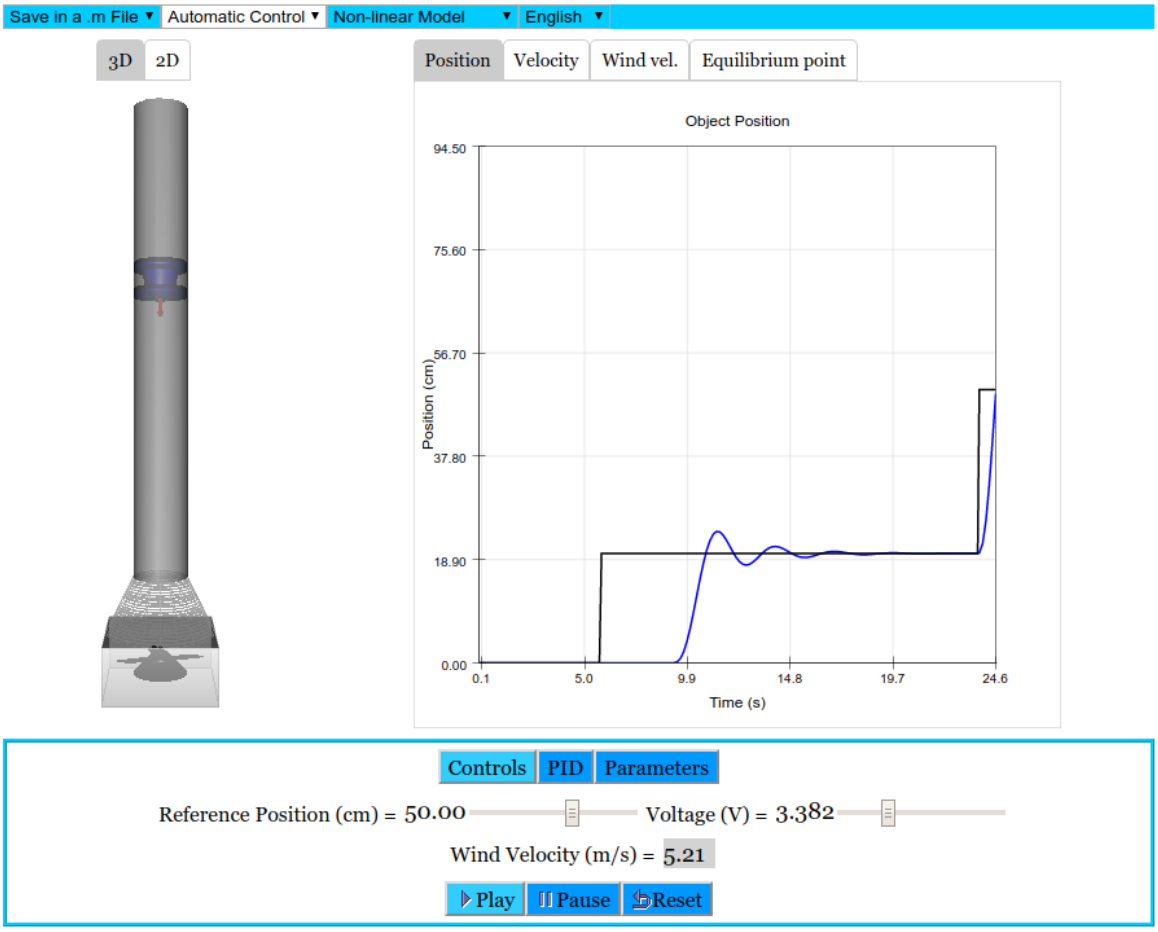


Figure 6: The virtual lab of the air levitation system.

the bottom, which allows modifying some system parameters, as the controller gains or the setpoint, and the connection buttons, in the remote lab, which are analogous to the simulation execution control ones, in the virtual lab. Figure 6 shows the web interface of the virtual laboratory, and Figure 7 of the remote laboratory.

The virtual and remote laboratory can be accessed in an open course in UNILabs (`http://unilabs.dia.uned.es`).

2.6 Discussion

Even when the knowledge, money, and time invested in the construction of a remote lab are certainly not negligible, it is an issue that is usually not addressed in literature. When faced with the decision to buy an educational system, several questions main arise, such as *why would I spent time and effort in building an experimentation system when I can buy a prebuilt system?* Well, there are several reasons

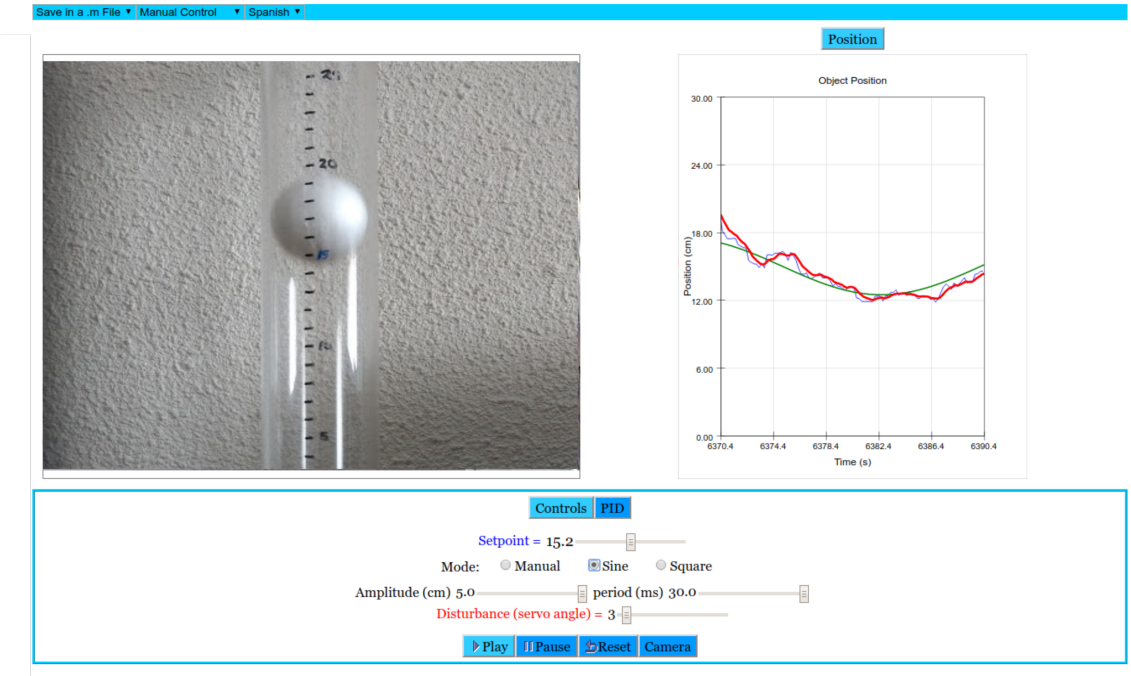


Figure 7: The remote lab of the air levitation system.

that can be argued to justify the decision:

- Commercial academic platforms are very expensive and sometimes can be less flexible. However, they are usually complemented with a curricula of activities, technical support, etc. Building your own system can be cost-effective and the final product is prone to be enhanced or adapted to experiences different from the originally thought.
- As mentioned before, the construction process itself is interesting from a didactic point of view. It can be proposed as an academic activity, if not from the scratch, dividing into smaller tasks or with some guidelines so students can addressed the activity.

In table 2, an estimation of the cost is provided in working time (hours) and money (€). The first group correspond to the inital design phase, which is the most difficult and laborious. Here, the results have been measured from the air levitation system developing process. The second group would be the effort needed to replicate and existing design and adapt to our specific needs. The time estimation was calculated as the mean time needed by 5 people which were given the design resources and told to build the system.

3 Activities

Up to this point, the platform is ready to experiment and carry on learning experiences. As a demonstration of capabilities, the next paragraphs present an activity which consists in the implementation, test, and comparison of different PI controllers.

	Time	Cost
Software	200h	n/a
Structural parts design	60h	n/a
Assembling	10h	<100€
Lab Software Design	40h-100h	n/a

Table 2: Cost & Time Estimation

Listing (1) PID controller implementation in Arduino

```
float PID::update(float y) {
  float kp = params[0], ki = params[1], kd =
       params[2];
  float I = state[0], e_prev = state[1];
  float e = sp - y;
  float P = kp*e;
  float I_prev = I;

  I += (ki*e) * (period/1000.0);
  float v = u0 + P + I;
  // Conditional integration
  float u = sat(v, 0.8, 1.0);
  if((u - v) * e < 0.0) {
    I = I_prev;
  }
  state[0] = I;
  state[1] = e;
  return u;
}
```

(a)

Listing (2) PICI controller implementation in Arduino

```
float PICI::update(float y) {
  float kp = params[0], ki = params[1], pr =
       params[2];
  float I = state[0], Ic =  state[1], e_prev =
        state[2];
  float e = sp - y;
  float P = kp*e;
  I += (ki*e) * (period/1000.0); // I
  Ic += (ki*e) * (period/1000.0); // Ic

  float v = u0 + P + (1-pr)*I + pr*Ic;
  // Conditional integration
  float u = sat(v, 0.8, 1.0);
  if((u - v) * e >= 0.0) {
    state[0] = I;
    state[1] = Ic;
  }
  // Reset
  if(e_prev * e <= 0) {
    state[1] = 0;
  }
  state[2] = e;
  return u;
}
```

(b)

Listing (3) PI2D controller implementation in Arduino.

```
float PID::update(float y) {
  // read the params
  float kp = params[0], ki = params[1], delta =
       params[2], delta_I = params[3],
      uff = params[4];

  // read the state of the previous iteration
  float I = state[0], e_last = state[1], I_last =
       state[2], ysp = state[3], uff_last = state[4];
  float e = sp - y;

  // feedforward: Apply u_ff whenever a new setpoint
       arrives and remove it after entering the
       deadband
  if(sp != ysp) {
    uff_last = (sp > ysp) ? u0 + uff : u0 - uff;
    ysp = sp;
  } else {
    if(abs(e) < delta) {
      uff_last = u0;
    }
  }

  // P Event
  if(abs(e - e_last) > delta) {
    e_last += (e > e_last) ? delta : -delta;
  }

  // I Event
  I += e * (period/1000.0f);
  if(abs(I - I_last) > delta_I) {
    I_last += (I > I_last) ? delta_I : -delta_I;
  }
  float v = uff_last + kp*e_last + ki*I_last;

  // Conditional integration
  float u = sat(v, 0.86, 1.0);
  if((u - v) * e >= 0.0) {
    state[0] = I;
  }

  // store the state for the next iteration
  state[0] = I;
  state[1] = e_last;
  state[2] = I_last;
  state[3] = ysp;
  state[4] = uff;

  return u;
}
```

(c)

Figure 8: Different controllers implemented in Arduino: (a) PI, (b) PI-CI, and (c) PI2D

3.1 Implementing the controller

The controller is deployed into an Arduino Nano. The software platform provides a template to implement generic controllers, so anyone who is familiarized with Processing can easily extend the system to experiment with new control laws. The communication and other implementation details are encapsulated into the provided code. The class Controller can be extended to implement the control law. In particular, the method update is invoked with a fixed period $T = 100ms$. This period is imposed by a physical limitation of the sensor, in order to provide stable measurements. The period of the controller can be chosen to a different value, though the measurement will be updated only every T seconds.

3.2 Control Strategies

Four control strategies have been implemented and tested, namely:

- PI: A classical PI controller.
- PI-CI: A PI controller with a Clegg's integrator .
- PI2D: An event-based PI controller with feedforward [26].
- Robust Adaptive Hybrid PI [28].

The approach chosen in this work to compare the performance of the different controllers is based on integral criteria. The following ones are commonly used to express the performance of a control system ([25]):

- $IE = \int_0^t e(t)dt$
- $IAE = \int_0^t |e(t)|dt$
- $ITAE = \int_0^t t|e(t)|dt$
- $ISE = \int_0^t e^2(t)dt$
- $QE = \int_0^t (e^2(t) + \rho u^2(t))dt$

To compare the performance of the different strategies, the closed loop is excited with step changes in the reference and external disturbances. The experiment are repeated several times for each implemented strategy and then the performance indexes are computed and averaged. To introduce an external disturbance, the plant provides a servo-mechanism that can modify the input air flow in a repeatable manner.

3.2.1 PI

The first one is a PI controller in parallel form $C(s) = (k_p + \frac{k_i}{s})e(s)$. This implementation is provided as a reference for the comparison. To cope with practical problems such as the saturation of the actuator, the controller implementation incorporate an antiwindup mechanism, based on a conditional integration. The code is shown in Figure 8(a), Listing 1.

3.2.2 PI-CI

The second implementation is a PI controller with a Clegg's integrator (PI-CI). The Clegg's integrator is a special kind of integrator which is based on resetting the state to zero whenever the input crosses by zero. It was shown by Clegg that the integrator with reset introduces a phase lag of $-38.1°$, as oppossed to the $-90°$ of the linear integrator. The PI-CI consist of a PI controller with a Clegg's integrator connected in parallel with the linear integrator, and a reset coefficient $\rho \in [0, 1)$ to weight the influence of each one. That means that for $\rho = 0$ the controller is equivalent to a PI controller, and $\rho = 1$ corresponds to a PI with a pure Clegg's integrator. This latter case is not recommended though: the effect of the integral term is lost and the resultant controller is not able to reject disturbances in steady state. With regard to the implementation in Arduino, the code is shown in Figure 8(b), Listing 2.

3.2.3 PI2D

The third implementation is an event-based PI with feedforward [26]. This controller consist of two parts: the feedforward block respond to changes in the setpoint, and generates a two-state control action that moves the output to the desired value, in absence of disturbances and assuming a perfect model of the system. The feedback block consist of a PI controller which has been modified to use a send-on-delta sampling strategy both in the proportional and integrated terms. To compute the feedforward action, in [26] the process is modeled as a First Order plus Time Delay (FOTD), yielding the two values of the control signal. However, since the model considered here contains a pure integrator, the second value is fixed to zero, and the first one can be modified as an extra degree of freedom. The feedback block is characterized by two params, δ_P and δ_I, which are the proportional and integral event thresholds, respectively. A new event is triggered every time the difference between the current value of the signal ($e(t)$ or $I_e(t)$) and the value at the last event is greater than the threshold, i.e. $|e(t) - e(t_k)| > \delta_P$ for the proportional term and $|I_e(t) - I_e(t_k)| > \delta_I$ for the integral term.

3.2.4 ROBUST ADAPTIVE HYBRID PI

The last implemented controller is described in [28]. It is a PI controller which resets its integrator's state if under the temporal regularization and the state vector of the closed loop belong to the jump set, D. Defining $e = r - y$, $\xi = x_I - x_{eq}$, the controller reset condition is $2e\xi + \xi^2\epsilon < 0, \tau > \rho$, and the state is reset to $e^+ = e$, $x_I^+ = x_I - \alpha\epsilon$, $\xi^+ = 0, \tau+ = 0$.

4 Conclusion and Future Work

A versatile low cost experimentation platform has been designed and built starting from the scratch. The use of rapid prototyping technologies in conjunction with open-source software and hardware makes the platform affordable for institutions with less resources. Moreover, the open and extensible design encourages the students to tinker with the system and boosts their creativity.

The Air Levitation System can be used as a standalone laboratory, connected directly to the student PC or laptop, or it can be combined with a Raspberry PI or another single-board computer to build a remote laboratory. The latter approach has been followed in the author's institution, incorporating several instances into the labs' network UNILabs.

Finally, the platform has been complemented with learning activities and guidelines to have a ready-to-use experimentation system. The activities include data-based model identification and the implementation of several PI controllers.

References

[1] Mahmoud Abdulwahed and Zoltan K. Nagy. The trilab, a novel ict based triple access mode laboratory education model. *Computers & Education.*, 56:262–274, 2011.

[2] Panos Antsaklis, Tamer Basar, Ray DeCarlo, Harris McClamroch, Mark Spong, and Stephen Yurkovich. NSF/CSS workshop on new directions in control engineering education. Technical report, University of Illinois at Urbana-Champaign:, 1998.

[3] J. Bermudez-Ortega, E. Besada-Portas, J.A. Lopez-Orozco, J.A. Bonache-Seco, and J.M. de la Cruz. Remote web-based control laboratory for mobile devices based on ejss, raspberry pi and node.js. In *In Proceedings of the 3rd IFAC Workshop on Internet Based Control Education, Brescia, Italy, 4-5 November, IFAC PapersOnLine*, volume 48, pages 158–163, 2015.

[4] J. Chacon, H. Vargas, G. Farias Castro, J. Sanchez Moreno, and S. Dormido. EJS, JIL Server and LabVIEW: An architecture for rapid developments of remote labs. *IEEE Transactions on Learning Technologies*, 8(4), 2015.

[5] Jesus Chacon, Gonzalo Farias, Hector Vargas, Antonio Visioli, and Sebastian Dormido. Remote interoperability protocol: A bridge between interactive interfaces and engineering systems. *IFAC-PapersOnLine*, 48(29):247 – 252, 2015.

[6] Dictino Chaos, Jesus Chacon, Jose Antonio Lopez-Orozco, and Sebastian Dormido. Virtual and remote robotic laboratory using ejs, matlab and labview. *Sensors*, 13(2):2595–2612, 2013.

[7] Xuemin Chen, Gangbing Song, and Yongpeng Zhang. Virtual and remote laboratory development: A review. In *12th International Conference on Engineering, Science, Construction, and Operations in Challenging Environments*, pages 3843–3852, Hawaii, USA, March 2010.

[8] Jennifer L. Chiu, Crystal J. DeJaegher, and Jie Chao. The effects of augmented virtual science laboratories on middle school students' understanding of gas properties. *Computers & Education*, 85:59 – 73, 2015.

[9] W. Christian and F. Esquembre. Modeling physics with easy java simulations. *The Physics Teacher*, 45:475–480, 2007.

[10] Wolfgang Christian, Francisco Esquembre, and Lyle Barbato. Open source physics. *Science*, 334(6059):1077–1078, 2011.

[11] L. de la Torre, M. Guinaldo, R. Heradio, and S. Dormido. The ball and beam system: A case study of virtual and remote lab enhancement with moodle. *IEEE Transactions on Industrial Informatics*, 11(4):934–945, Aug 2015.

[12] L. de la Torre, J. Sanchez, S. Dormido, J.P. Sanchez, M. Yuste, and C. Carreras. Two web-based laboratories of the fisl@bs network: Hooke's and snell's laws. *European Journal of Physics*, 32:571–584, 2011.

[13] L. de la Torre, J. P. Sanchez, and S. Dormido. What remote labs can do for you. *Physics Today*, 69:48–53, 2016.

[14] Sebastian Dormido. Control learning: present and future. *Annual Reviews in Control*, 28:115–135, 2004.

[15] Natividad Duro, Raquel Dormido, Hector Vargas, Sebastian Dormido-Canto, Jose Sanchez, Gonzalo Farias, Francisco Esquembre, and Sebastian Dormido. An integrated virtual and remote control lab: The three-tank system as a case study. *Computing in Science & Engineering*, 10(4):50–59, 2008.

[16] G. Farias, R. D. Keyser, S. Dormido, and F. Esquembre. Developing networked control labs a matlab and easy java simulations approach. *IEEE Transactions on Industrial Electronics*, 57:32663275, 2010.

[17] L. Gomes. Current trends in remote laboratories. *IEEE Transactions on Industrial Electronics*, 56:47444756, 2009.

[18] Isaías González, Antonio José Calderón, Andrés Mejías, and José Manuel Andújar. Novel networked remote laboratory architecture for open connectivity based on plc-opc-labview-ejs integration. application in remote fuzzy control and sensors data acquisition. *Sensors*, 16(11):1822, 2016.

[19] Ruben Heradio, Luis de la Torre, and Sebastián Dormido. Virtual and remote labs in control education: A survey. *Annual Reviews in Control*, 42:1–10, 2016.

[20] Clara M. Ionescu, Ernesto Fabregas, Stefana M. Cristescu, Sebastin Dormido, and Robin De Keyser. A remote laboratory as an innovative educational tool for practicing control engineering concepts. *IEEE Transactions on Education*, 56(4):436 – 442, November 2013.

[21] Carlos A. Jara, Francisco A. Candelas, Santiago T. Puente, and Fernando Torres. Hands-on experiences of undergraduate students in automatics and robotics using a virtual and remote laboratory. *Computers & Education*, 57:2451–2461, 2011.

[22] Jing Ma and Jeffrey V. Nickerson. Hands-on, simulated, and remote laboratories: A comparative literature review. *ACM Computing Surveys*, 38, 7, 2006.

[23] Andrés Mejías, Reyes S Herrera, Marco A Márquez, Antonio José Calderón, Isaías González, and José Manuel Andújar. Easy handling of sensors and actuators over tcp/ip networks by open source hardware/software. *Sensors*, 17(1):94, 2017.

[24] R. Pastor, J. Sanchez, and S. Dormido. Web-based virtual lab and remote experimentation using easy java simulations. In *In proceedings of the 16th IFAC World Congress, Prague, Czech Republic, 3-8 July*, 2005.

[25] Karl J. Åström and Tore Hägglund. *Advanced PID Control*. ISA-The Instrumentation, Systems, and Automation Society, 2005.

[26] Jose Sánchez, Antonio Visioli, and Sebastián Dormido. A two-degree-of-freedom pi controller based on events. *Journal of Process Control*, 21:639–651, 2011.

[27] Eileen Scanlon, Chetz Colwell, Martyn Cooper, and Terry Di Paolo. Remote experiments, re-versioning and re-thinking science learning. *Computers & Education*, 43(1-2):153–163, 2004.

[28] Ignacio Rubio Scola, Mariella M. Quadrios, and Valter J. S. Leite. Robust Hybrid PI Controller with a Simple Adaptation in the integrator reset state. In *IFAC PapersOnLine*, volume 50, 2017.

[29] H. Vargas, J. Sanchez, N. Duro, S. Dormido-Canto, G. Farias, S. Dormido, F. Esquembre, C. H. Salzmann, and D. Gillet. A systematic two-layer approach to develop web-based experimentation environments for control engineering education. *Intelligent Automation & Soft Computing*, 14(4):505–524, 2008.

[30] Z.C. Zacharia. Comparing and combining real and virtual experimentation: an effort to enhance students' conceptual understanding of electric circuits. *Journal of Computer Assisted Learning*, 23(2):120–132, 2007.