Customized Online Laboratory Experiments

A general tool and its application to the Furuta inverted pendulum

Daniel Galan, Dictino Chaos, Luis de la Torre,
Ernesto Aranda-Escolastico, and Ruben Heradio
POC: D. Galan (dgalan@dia.uned.es)

Thanks to online laboratories, students can perform experimental activities from their mobile devices and/or computers. This paper proposes an experimentation environment that extends the capabilities of interactive online labs with scripting language support. Thus, control engineering students can specify complex experiments, avoid routine tasks, and empirically test controllers made by themselves.

To put our work into context, let us introduce two popular taxonomies for labs and experiments. First, two criteria can be used to classify labs: (i) according to where they are accessed, labs can be local or remote; and (ii) according to their physical nature, labs can be real plants or computer simulations. Hence, these criteria can be combined in four different ways [1]:

1) *Local access & real resource:* ***Traditional hands-on labs***.
2) *Local access & simulated resource:* ***Local virtual labs***. The plant is substituted by a computer simulation.
3) *Online & real resource:* ***Online remote labs***. The plant is real, but accessed through the Internet.
4) *Online & simulated resource:* ***Online virtual labs***. The plant is replaced by a simulation that can be remotely and simultaneously accessed by many students, enabling virtual classrooms.

Our approach supports scripting experiments for online remote labs and local/online virtual labs. Harward et al. [2] propose the following classification regarding the type of experiments that can be undertaken on remote and virtual labs:

1) *Batched experiments.* The student specifies the experiment beforehand and submits it to the lab. When the experiment finishes, the student receives the results. No feedback is

provided during the experiment execution. The MIT Microelectronics WebLab [3] is a good example of this type of labs.

2) *Interactive experiments.* The student can manipulate lab parameters during the experiment execution, receiving immediate feedback through sensors, cameras and other devices. For example, with the Ball and Beam remote lab described in [4], students can adjust controller parameters in real-time to stabilize the ball in a specific position of the beam while they visualize the effects in the plant.
3) *Sensor Experiments.* The student only monitors or analyzes real-time data streams without influencing the lab execution. An example of this type of experiments is the MIT instrumented flagpole [5].

We propose the use of an Experimentation Environment (EE) with which students can perform interactive tasks with virtual and remote laboratories, but also, write their own code and develop more complex tasks. Our approach provides a general solution for all the previously defined kind of labs and also enables combining them. For instance, students can program an experiment in the EE. Then, they can monitor its execution, or even modify the experiment on-the-fly by directly manipulating the lab. The pedagogical benefits of this flexibility will be illustrated with two experiments on a Furuta inverted pendulum: (i) keeping the pendulum in the upright position, while its rotatory arm follows the reference, and (ii) implementing the swing up control. Moreover, this paper also reports the successful application of our approach in a distance-education university course. Check "This work at a glance" to see a brief summary and what we have achieved with this work.

Related Work

Current experimentation scripting solutions for control engineering education are ad-hoc tools that only work on specific virtual or remote labs. In contrast, our approach provides a general solution; EE enables scripting experiments on any lab developed with Easy Java/javascript Simulations (EJsS) [6], gathering and visualizing real-time data from them. Furthermore, EE provides a simple interface to the remaining labs created with any other technology; that is, to interconnect a non-EJsS lab with EE. The only requirement the lab must satisfy is implementing the reduced function list in the interface.

Table 1 includes some significant laboratories that use distinct programming languages to script experiments. Data have been obtained from literature reviews [7], [8], [9]. The first two columns are the name and the description of the laboratory or environment. The rest of the columns provide important information about the most relevant features general experimentation environments should have, such as the followings:

1) The grade of coupling between the experimentation tools and the plant that is either simulated or used in the real lab.
2) The usability of the system, the ease with which users can perform experiments using the available resources.
3) The reusability of the experimentation tools presented in the lab for other setups.
4) Whether the experimentation tools are embedded in the lab user interface or not.

Finally, Table 1 includes the name of the language to program the experiments, whether this language is easy to learn or not (learnability), the ability to perform complex experiments (power), and, if the language is a Domain-Specific Language (DSL) or General Purpose one (GPL). As discussed later in this paper, all these characteristics were assessed through a student's questionnaire. According to the table:

1) Coupling is very high between the labs and their experimentation scripting support, except for general purpose environments such as PLC programming or assembly code. Any change in the lab requires modifications in its experimentation scripting support. In contrast, EE has been specifically designed so that changes in the lab do not alter the functionality of the environment and thus do not require any extra changes.
2) When the experimentation scripting support is embedded into the lab, reusing it in other labs usually requires making profound changes in every particular lab. In contrast, EE is a general solution where any lab is easily deployed.
3) A DSL specifically designed to program experiments is usually easier to learn and use than a GPL created for any kind of task [10]. In addition, GPLs involve more risks and require the implementation of security layers in the labs. EE provides students with a Blockly DSL for most tasks. Nevertheless, for those special situations when the DSL is not enough, EE supports complementing the DSL code with general JavaScript code.

A Universal Experimentation Environment for Virtual and Remote labs

EE is open source, and it is distributed as part of *EJSApp*, which is a package to deploy EJsS labs on the Moodle Learning Management System (LMS) and it is freely available at: https://github.com/UNEDLabs/moodle-mod_ejsapp. EE supports:

1) *Programming experiments both graphically or textually.* EE aims to be useful not only for any type of online lab, but also for undergraduate and secondary students. For that reason, it offers both a visual and a textual programming language. The first one, Blockly, is especially indicated for students without an in-depth knowledge of programming, or who are interested in undertaking simple experiments. The second, JavaScript, allows the

user to perform more complex tasks, such as experiments that require complex equations. Figure 6 exemplifies this situation.

2) *Extracting and plotting data from the experiments.* EE allows students to choose which data to analyze and plot. They can choose from line charts: the X and Y-axis data, the sampling period and whether they want to display all the data or just the last "*n*" points. Students can collect data at any specific moment of the experiment. They can select which data to visualize and so, they are not constrained by the laboratory interface limits. In this way, they become aware of the importance of something they take for granted in traditional interactive virtual and remote laboratories.
3) *Real-time interaction among the lab, the experiment code and the user.* Students can run their experiments with just a mouse click. Since the experiment execution does not require compilation or waiting times, students can quickly make changes to correct errors or try different algorithms. The experiments have an immediate effect on the laboratory, reflecting all changes in real time at the Graphical User Interface (GUI).

User Interface Elements

Figure 1 shows the three main components of the EE user interface, which are detailed below:

1) **Experiment Editor (marked with a red rectangle at the bottom):** This is where the experiments are programmed. It consists of a menu and an editing area. The first one, located on the left side, is where the different blocks used to design the scripts are available and organized in categories. The experiment is defined by joining different blocks. Later on, EE will interpret and execute the experiment automatically. It does not require compilation or waiting times; the creation process is fast, dynamic and simple.
2) **Custom Charts Panel (blue box at the top right):** Students can define customized plots using EE. They decide which values or expressions to represent and how they want to visualize them. It is possible to set the sampling period, and the number of points to be plotted. The information is depicted in real time as the experiment progresses. EE enables students to create different types of graphs for the same experiment, or even compare the result of previous tests on the same chart.
3) **Online Laboratory (green box at the top left):** Online laboratories, whether they are virtual or remote (as their nature is transparent to the environment), do not need additional adaptations and are automatically included in EE.

Identifying and Separating User Roles

Along the online lab life cycle, three roles can be distinguished: the designer (the person who creates the laboratory), the instructor (who uses the lab with teaching purposes), and the students (who carry out experiments with learning purposes). The use of EE offers several advantages to each of them.

As EE can collect and plot any experimental data, the lab **designer** is freed from the need to consider what data and plots are significant for any potentially interesting experiment in order to implement the corresponding functionality in the lab.

Thanks to EE,**instructors** can propose more diverse tasks to their students. Also, the evaluation of the students' work can be enhanced thanks to the possibility of reviewing the students' code to see how they reached their lab results. In addition to being a practical exercise, the experiments can be used as a tutorial or as an example, so the instructor can distribute the laboratory along with experiments to better explain different phenomena.

Lastly, EE is a general-purpose tool for **students** to perform different lab tasks, including those that the designer and the instructor might not have contemplated. This allows research-based learning techniques to be used in any laboratory. The environment gives students freedom and precision, allowing them to modify variables that might not appear in the interface. Moreover, they can add actions to be performed when an event is triggered and automate repetitive tasks such as data collection. It also enables the custom design and analysis of charts. Experiments can be saved (so that students can continue unfinished work at any time) and shared (so they can study different approaches to accomplish the same task).

Figure 2 shows a clear work-flow of the work to be done by each of the roles involved in the laboratory life cycle. The tools cited as examples in the figure have been used in the final implementation of EE and are described throughout the paper.

Underlying EE Technologies

The following points summarize the technologies upon which EE is built:

1) *EjsS to develop online laboratories*. The creation of student-friendly and effective online interactive labs requires programming knowledge and development time. EjsS is a program generator that facilitates the design and distribution of these laboratories [11]. Users can focus on designing the lab user interface and, in the case of virtual labs, on defining their mathematical models. Then, EjsS automatically generates the Java or JavaScript code that

implements the lab. As nowadays most browsers do not support Java applets because of a number of security issues, it is preferable to work with the JavaScript generated code. This way, labs can be directly accessed through a web browser without installing any program. EjsS is part of Open Source Physics (OSP), which organizes and shares collections of open-source educational resources in Physics through the ComPADRE Digital Library [6]. In particular, OSP offers more than 500 EjsS labs [12], ready to be used with our EE.

2) *Blockly to graphically script experiments*. Blockly is an open source library created by Google in 2011 to develop visual programming environments and used in many applications [13], [14]. Scripts are generated by connecting blocks with different functionalities between them, as if it was a puzzle. The final script can be translated to JavaScript, Python, Dart, PHP or adapted to any user-defined language [15]. In the case of EJsS, since the labs will be preferably implemented in JavaScript, Blockly diagrams are translated into JavaScript. We have enriched the standard programming environment offered by Blockly with new blocks and functions to control every aspect required by online labs. Table 2 summarizes the new blocks.
3) *Chart.js to plot experimental data*. Chart.js is an open-source JavaScript library that enables the creation of HTML5 responsive charts [16].
4) *UNILabs to deploy EE and online labs*. UNILabs [17] is an international network of virtual and remote labs supported by different universities around the world [18]. As UNILabs is based on Moodle, the management and deployment of labs and experimentation resources are simple tasks for the instructors.

Use Examples of the EE in a Furuta Pendulum System

This section exemplifies the educational uses of EE by defining two experiments for the Furuta inverted pendulum. First, a virtual lab for the pendulum is presented; then, its remote lab is described. By using both, students can contrast the theoretical results, obtained from the virtual lab, against the real data, obtained from the remote lab. Both labs are open-source and freely available at: https://github.com/AnonymCSM/FurutaPendulum. There it is also possible to find all the information provided to students to work with the laboratories: a basic guide to the Furuta pendulum, the documentation relating to the lab interface and the task protocol.

Virtual Lab

Inverted pendulum systems and, particularly, the Furuta pendulum, have been used as a benchmark in control engineering for years because of their dual nature [19]: while they usually are easily describable systems, they also imply challenging problems from the point of view of non-linear dynamics and control theory.

A Furuta pendulum is formed by a rigid link attached to an arm which can rotate in a horizontal plane, perpendicular to the pendulum (see Figure 3). This rotary arm results in the main difference with other inverted pendulum systems, and adds additional complexities in the form of Coriolis forces and centrifugal torques. This provides not only a more involved mathematical model but also interesting behaviors to study [20], [21], [22].

The Furuta pendulum is schematically described in Figure 3. The system has two degrees of freedom, which are the angle described by the rotary arm ψ and the angle described by the pendulum θ. Rigorously, the configuration space is a 2-torus, that is, $\begin{pmatrix}\psi & \theta\end{pmatrix} \in \mathcal{S}^1 \times \mathcal{S}^1$, and the corresponding tangent space is $\begin{pmatrix}\psi & \theta & \dot{\psi} & \dot{\theta}\end{pmatrix} \in \mathcal{S}^1 \times \mathcal{S}^1 \times \mathbb{R}^2$. If we assume that both angles are increasing when the arm and the pendulum rotates counter-clockwise (as shown in Figure 3), their positions in a Cartesian coordinate system s_0 and s_1 can be expressed as follows

$$s_0 = l_0 \begin{pmatrix} \sin\psi \\ \cos\psi \\ 0 \end{pmatrix} \tag{1}$$

$$s_1 = \begin{pmatrix} L_0 \sin\psi + l_1 \sin\theta\cos\psi \\ L_0 \cos\psi - l_1 \sin\theta \sin\psi \\ l_1 \cos\theta \end{pmatrix}, \tag{2}$$

where l_0, l_1 are the length to their center of mass of the rotary arm and the pendulum, respectively and L_0 is the total length of the rotary arm. For the sake of brevity, the equations of motion of the inverted pendulum are not reproduced, but they can be derived from the Euler-Lagrange method as in [20], [21], [22].

The virtual laboratory consists of a simulation of the pendulum equations developed with EJsS. The simulation incorporates a 3D graphical representation of the pendulum and user inputs to interact with it, as shown in Figure 4. The user interface is designed as simple as possible to allow only the most basic actions: change the position of the pendulum, the reference, as well as stop, pause or resume the simulation.

Remote Lab

The remote laboratory is composed of two main components: 1) the server side, developed in LabVIEW, that accepts connections using an element called JIL [23] and 2) the view in EJsS, whose variables are linked to the ones in the remote laboratory. The implementation of the server in LabVIEW and the EJsS files that include JIL communication between the software and the hardware can also be found at https://github.com/AnonymCSM/FurutaPendulum.

The design of the remote laboratory is very close to the virtual one, replacing the 3D view with a webcam video streaming of the pendulum and eliminating the interaction with the variables that can not be directly manipulated on the real plant (angles θ and ψ, which in our virtual and remote labs are called α and β, respectively).

The virtual and remote laboratory user interfaces have been streamlined as much as possible, without losing usability. This is the crucial difference from previous designs that are bloated, and include lots of tools and visualizations, each focused on a limited set of very specific tasks.

A vital feature of the laboratory is that it enables the user to change the controller of the plant. This has been done in previous laboratories as well, but using a specific language to the domain of application, which needs a special server for its execution (see [24] for a previous successful implementation of this idea). In this work, this task has been addressed with a novel approach, being the controller defined in JavaScript. Hence, the same code can interact without modification with the simulation on the browser and with the real system. In the latter case, the JavaScript code is executed in a sandbox in LabVIEW.

Furthermore, the controller does not need "to be written" at all, and the EJsS lab does not have an editor or options to load and save files. Everything (from the controller to the visualization) can be constructed with our EE, as the next section illustrates. Thus, the user interface is general, simple and can be used for many different purposes and assignments without any modification.

Experiment 1: Keeping the Pendulum in the Upright Position While the Rotatory Arm Follows the Reference Changes

This activity aims to develop a state feedback linear control law capable of keeping the pendulum in a vertical and upward position while the base of the pendulum is following position setpoint changes.

For this remote laboratory, the controller has been implemented on the server side. Instead of implementing it on the client side, where the control signal should be sent periodically to the server. This is mandatory for the Furuta pendulum because it is a very fast plant (stability is lost above 30ms of sampling time or delay).

Without students having to indicate anything, the experimentation environment is capable of distinguishing whether the code fragments must be executed locally or remotely, depending on whether the laboratory is virtual or remote and the type of code the student wants to run. Figure 5 shows an experiment with local and remote code to control the pendulum and to visualize the outputs. Code fragments related to the visualization are executed locally. However, the code that substitutes the controller function (the one inside the red rectangle in the figure) is executed remotely.

At the top of Figure 5, the experiment designed by the student is shown. At the bottom, the server side is represented, showing the program implemented in LabVIEW, which is responsible for controlling the input and sending the output of the Furuta pendulum. The code that is executed in the server part is marked in a red rectangle. The controller designed using the EE is displayed above. In the lower part, the already translated code that is remotely executed on the server is shown. Although this controller statements are programmed on the client side in JavaScript language, LabVIEW has tools to translate and execute it, so the received code is evaluated at each execution step, transmitting the new control action value to the motor.

In the top part of Figure 5 (in the client side), there is a possible design for the proposed experiment. It consists of joining different blocks to form a script. The first of these blocks, called "*Create Chart*", situated at the top of the editing area, is used to define the graph for evaluating the control performance. It plots the reference for the angle described by the rotatory arm ("*betaRef*") and its real value ("*beta*"), obtained from the remote plant as a function of the time ("*t*"). A sample shall be taken every 100 milliseconds. After defining the visual elements, there is a set of blocks that describe the actual execution of the experiment. The first of these, the "*start data collection*" block, is used for the system to start taking the measures defined in the "*Create Chart*" block and plotting them in a graph. The first "*set*" block is used to tell the server that a controller designed by the student will be used. Finally, the "*replace function*" block is responsible for modifying the function implemented by the controller. This code fragment makes use of the following laboratory variables: "*beta*", "*betaRef*", "*alpha*", "*dbeta*" and "*dalpha*" referring to the real beta angle, the reference beta angle, the real alpha angle, the derivative of beta and the derivative of alpha, respectively. Finally, the result obtained from the controller code is passed to the variable "*u*", which refers to the motor voltage connected to the rigid arm. Once the experiment is executed, the system translates the code automatically into JavaScript and the statements inside the "*replace function*" block are sent to the server to be used as the controller for the pendulum system.

Figure 1 shows a screenshot of the experimentation environment after executing the experiment. At the top right part, it is the user-defined graph with the results. It shows the changes in the angle of reference corresponding to, at first, a step to 1.48 radians and, later, to -0.38 radians. These changes were made during the execution of the experiment using the interactive element of the user interface.

The performance of different controllers can be evaluated quickly and easily with this experiment. It can be observed how the parameters used are appropriate, since, in addition to keeping the pendulum in equilibrium, the system can follow the reference efficiently. Thanks to the IP-camera, which provides a front view of the plant, and to the user-defined graphs, both updated in real time, students can corroborate the results visually and empirically. In case students would like to try different parameters, they merely have to modify them in the experiment and relaunch it. Then, they could compare the previous graph with the new one to study the controller efficiency, or to see if the system becomes unstable and the pendulum falls.

Experiment 2: Implementing the Swing Up Control

It is possible to design much more complicated controllers maintaining the same structure in the design of the experiment used in the previous experience. The controller proposed in this experiment is based on the design of K. J. Åström and K. Furuta [25]. This controller is not only able to follow the rotating arm angle reference, but also to swing the pendulum upwards if it falls. This is something that might happen if the change in angle reference is very abrupt. In this way, if the pendulum is outside the attraction region, the controller will execute the code that will bring it back to the desired equilibrium position. Otherwise, the control will be the same as that studied in the previous experiment.

Since this is an advanced controller that requires more lines of code, there can be students to whom the use of blocks may impose an added difficulty in the creation process. The experimentation environment offers the possibility of evaluating code written directly in JavaScript to make the design task more accessible. This block can be activated or deactivated by the instructor according to the requirements of the activity to be performed.

To check the correct operation of this controller, it is necessary to start from an initial situation in which the pendulum is down. To do this, a possibility is to initially send a controller to the server where the control signal is 0 ($u = 0$), and then transmit the new controller and test whether or not the pendulum rises.

Figure 6 shows the script proposed for the experiment. Compared to the previous example, the main change is the controller code, which has been implemented in JavaScript directly. Instead of creating equations by verbosely combining blocks, the *"evaluate"* block is used. The lower part of the figure shows in detail the JavaScript code that has been used. As this implementation varies from the previous one in the elevation of the pendulum, to check if the script works correctly it is enough to see through the IP-camera if the pendulum really rises once the experiment is executed.

Empirical Assessment of the EE Pedagogical Value

The evaluation proposed in this section was undertaken for the course *Fundamentals of Automatic Control* to assess the EE educational utility. This course is taught at the National University of Distance Education in Spain, in the 4th year of the degree associated with the faculties of Physical Sciences and Computer Science. 23 students of Physical Sciences (from a total of 27) and 17 of Computer Science participated (of 22 students enrolled) in this process. The evaluation consisted of:

1) An initial interview to gather general information before the experiences were undertaken.
2) The first experience consisted of using just a virtual laboratory without the EE. Students had to study the Furuta pendulum model and the controller already implemented using thelab GUI. They had to modify parameters such as the initial state of the pendulum or the controller parameters to perform the experimentation tasks.
3) Students answered a questionnaire of 40 questions after working with the laboratory. These questions were categorized according to the aspect to be analyzed: (1) Environment, (2) Usability, (3) Learning and (4) Other aspects. The answers were supplemented with data automatically collected from the web environment. Interested readers can find the whole set of questions, tests, and the automatically obtained data in the public repository: https://github.com/AnonymCSM/FurutaPendulum/blob/master/Evaluation.pdf
4) In the second experience, students had to use the Furuta pendulum virtual lab through EE. The goal was to create a position controller with the swing up feature (Experiment 2).
5) Finally, students filled again the previous questionnaire to compare their answers and appraisals with both tools.

The purpose of this evaluation was to compare student performance with traditional online labs versus the EE. Students can interact with the first ones but not design their own controllers, something they can do if the lab is included in the EE. The complete evaluation protocol was held on the same day. Moreover, students did not have access to the answers of the work done for the first experience until they had completed the second questionnaire. This was intended to prevent students from having more knowledge of the concepts involved when they were facing the second evaluation in comparison to when they faced the first one.

Initial Interview Results

The aim was to obtain general information about the students' background, experiences and confidence with the concepts explained in the course. The results are summarized in Figure 7.

Confronted Results of the First and Second Evaluation

Results obtained from the students' evaluations after the first experience (only with the virtual laboratory) and the second (using the EE) are compared in this section.

About the Working Environment

Regarding the questions that assessed students' perception of the virtual laboratory and EE, the most notable results are found in Figure 8.

No significant differences were found regarding the ease of use of the virtual lab and EE; students did not have more problems working with one tool than the other. However, EE helped them identifying the underlying model and control concepts slightly better than the sole use of the virtual lab. Students believed that EE was more useful than the lab to help them solving their own doubts, to identify misconceptions, to make use of concepts learned in theory, and to perform more experiments than the requested ones.

About the Usage

Students needed more opportunities (+1346 executions) and time (+48 hours) to perform the experimentation tasks using the EE than the virtual laboratory. We consider reasonable this difference since the proposed tasks for the EE were more complex than for the virtual lab. The extra time was used to deepen their knowledge of the theory and to solve their doubts as shown by the results described above. Therefore, keeping more time working with the system helps them to learn more and with better quality.

Students participated more in the forum (+17 messages) when they were facing experimentation tasks for the EE than with the virtual laboratory. This fact favored the exchange of ideas and points of view among the students.

About the Learning Outcomes

Students took a 10-question test to assess what they had learned using both tools. They had to choose between 4 possible answers, and mistakes did not penalize. In the test, five questions were about the model of the Furuta pendulum and the other five about the controller. The number of students who answered the questions related to the controller correctly increased significantly thanks to the use of EE (in total, 21% more students passed the test once they used EE and 93% obtained better grades). Therefore, it can be concluded that allowing students to implement their own algorithms helps to improve the degree of learning of specific concepts.

About Other Aspects

- *The quantity and quality of information provided to students*: 34 of the 40 students considered that they received the appropriate information to perform the tasks for both tools (responding very much in agreement or in agreement).
- *Problems and difficulties during the performance of tasks*: The students had more challenges in performing the work with EE than with the virtual laboratory (24 with EE and 12 with the laboratory). We believe that documentation and clarity of tasks need to be improved.
- *Tool affinity*: Students found both tools useful for this and other subjects. 32 of them said they were very much in agreement or agreed on the use of laboratories and EE, 5 only to the use of laboratories, 2 just to EE and 1 to none. Likewise, 21 students said they strongly agreed or agreed about having fun using EE versus 26 with the virtual lab.
- *Negative feedback*: Most of the negative comments about EE are due to the amount of time students had to dedicate to carry out the experiences. However, as demonstrated by the marks obtained in the test, that time was invested in obtaining a more solid knowledge. So we consider that, despite the feedback from the students, it is beneficial for them. Nevertheless, we will try to make the activities more enjoyable and attractive in the future.

Conclusions

An online experimentation environment has been presented to extend current virtual and remote lab capabilities. With it, students can easily program experiments, test controllers designed by themselves, monitor the execution of experiments, and collect and plot real-time data streams.

Two experiments for the Furuta pendulum have been described to illustrate the proposed environment applicability to teach control engineering. First, a simple state feedback linear control law has been defined to keep the pendulum in a vertical and upward position while its base follows position set-point changes. Then, a more complex controller is designed, which not only follows the rotating arm angle reference but also swings the pendulum upwards if it falls.

Finally, a pedagogical evaluation of our approach has been reported, working with students from a distance-education university course, obtaining positive results regarding both learning outcomes and student acceptance.

Acknowledgments

This work has been supported by (i) the Spanish Ministry of Science, Innovation and Universities, under the project with reference DPI2016-77677-P; (ii) the Community of Madrid, under the research network RoboCity2030-DIH-CM P2018/NMT-4331; and (iii) the Spanish Ministry of Economy and Competitiveness under the project CICYT DPI2014-55932-C2-2-R.

References

[1] S. Dormido, "Control learning: present and future," *Annual Reviews in Control*, vol. 28, no. 1, pp. 115–136, 2004.

[2] V. J. Harward, J. A. Del Alamo, S. R. Lerman, P. H. Bailey, J. Carpenter, K. DeLong, C. Felknor, J. Hardison, B. Harrison, I. Jabbour *et al.*, "The ilab shared architecture: A web services infrastructure to build communities of internet accessible laboratories," *Proceedings of the IEEE*, vol. 96, no. 6, pp. 931–950, 2008.

[3] J. Hardison, D. Zych, J. Del Alamo, V. Harward, S. Lerman, S. Wang, K. Yehia, and C. Varadharajan, "The microelectronics weblab 6.0–an implementation using web services and the ilab shared architecture," in *International Conference on Engineering Education and Research*, Tainan, Taiwan, 2005, pp. 1–5.

[4] L. de la Torre *et al.*, "The ball and beam system: A case study of virtual and remote lab enhancement with moodle," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 4, pp. 934–945, Aug 2015.

[5] K. Amaratunga and R. Sudarshan, "A virtual laboratory for real-time monitoring of civil engineering infrastructure," in *International Conference on Engineering Education*. Manchester, U.K.: Citeseer, 2002, pp. 18–22.

[6] W. Christian, F. Esquembre, and L. Barbato, "Open source physics," *Science*, vol. 334, no. 6059, pp. 1077–1078, 2011.

[7] L. Gomes and S. Bogosyan, "Current trends in remote laboratories." *IEEE Transactions on Industrial Electronics*, vol. 56, no. 12, pp. 4744–4756, 2009.

[8] R. Heradio, L. de la Torre, and S. Dormido, "Virtual and remote labs in control education: A survey," *Annual Reviews in Control*, vol. 42, no. Supplement C, pp. 1 – 10, 2016.

[9] V. Potkonjak, M. Gardner, V. Callaghan, P. Mattila, C. Guetl, V. M. Petrović, and K. Jovanović, "Virtual laboratories for education in science, technology, and engineering: A review," *Computers & Education*, vol. 95, pp. 309–327, 2016.

[10] M. Fowler, *Domain-Specific Languages*. Addison-Wesley Professional, 2010.

[11] F. Esquembre, "Easy Java Simulations: A software tool to create scientific simulations in Java," *Computer Physics Communications*, vol. 156, no. 6, pp. 199–204, January 2004.

[12] "Open Source Physics webpage," http://www.opensourcephysics.org, available online (last accessed 09/12/2018).

[13] N. Fraser, "Google blockly-a visual programming editor," *Published. Google, Place*, 2014.

[14] J. Trower and J. Gray, "Blockly language creation and applications: Visual programming for media computation and bluetooth robotics control," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. Kansas City, Missouri, USA: ACM, 2015, pp. 5–5.

[15] J. Trower and I. Gray, "Creating new languages in Blockly: two case studies in media computation and robotics," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. Kansas City, Missouri, USA: ACM, 2015, pp. 677–677.

[16] "ChartJS webpage," http://www.chartjs.org/, available online (last accessed 09/12/2018).

[17] "UNILabs webpage," http://unilabs.dia.uned.es, available online (last accessed 09/12/2018).

[18] J. Sáenz, J. Chacón, L. De La Torre, A. Visioli, and S. Dormido, "Open and low-cost virtual and remote labs on control engineering," *IEEE Access*, vol. 3, pp. 805–814, 2015.

[19] O. Boubaker, "The inverted pendulum benchmark in nonlinear control theory: a survey," *International Journal of Advanced Robotic Systems*, vol. 10, no. 5, p. 233, 2013.

[20] E. Aranda-Escolástico, M. Guinaldo, M. Santos, and S. Dormido, "Control of a chain pendulum: A fuzzy logic approach," *International Journal of Computational Intelligence Systems*, vol. 9, no. 2, pp. 281–295, 2016.

[21] M. Ramírez-Neria, H. Sira-Ramírez, R. Garrido-Moctezuma, and A. Luviano-Juarez, "Linear active disturbance rejection control of underactuated systems: The case of the Furuta pendulum," *ISA Transactions*, vol. 53, no. 4, pp. 920–928, 2014.

[22] J. G. Gonzalez Fontanet, A. Lusson Cervantes, and I. Bausa Ortiz, "Alternatives of control for a Furuta's pendulum," *Revista Iberoamericana de Automatica e Informatica Industrial*, vol. 13, no. 4, pp. 410–420, 2016.

[23] J. Chacon, H. Vargas, G. Farias, J. Sanchez, and S. Dormido, "Ejs, jil server, and labview: An architecture for rapid development of remote labs," *IEEE Transactions on Learning Technologies*, vol. 8, no. 4, pp. 393–401, 2015.

[24] D. Chaos, J. Chacón, J. A. Lopez-Orozco, and S. Dormido, "Virtual and remote robotic laboratory using ejs, matlab and labview," *Sensors*, vol. 13, no. 2, pp. 2595–2612, 2013. [Online]. Available: http://www.mdpi.com/1424-8220/13/2/2595

[25] K. J. Åström and K. Furuta, "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, 2000.

[26] K. Collins, *PLC programming for industrial automation*. Exposure, 2007.

[27] E. Lindsay, D. Liu, S. Murray, and D. Lowe, “Remote laboratories in engineering education: Trends in students’ perceptions,” in *Proceedings of the 18th Conference of the Australasian Association for Engineering Education*. Melbourne, Australia: Australasian Association for Engineering Education, 2007.

[28] O. A. Herrera, G. R. Alves, D. Fuller, and R. G. Aldunate, “Remote lab experiments: Opening possibilities for distance learning in engineering fields,” in *Education for the 21st CenturyImpact of ICT and Digital Resources*. Springer, 2006, pp. 321–325.

[29] M. Casini, D. Prattichizzo, and A. Vicino, “The automatic control telelab: A user-friendly interface for distance learning,” *IEEE Transactions on Education*, vol. 46, no. 2, pp. 252–257, 2003.

[30] T. Mohsen-Torabzadeh, Z. M. Hossain, P. Fritzson, and T. Richter, “OMWeb-Virtual web-based remote laboratory for Modelica in engineering courses,” in *Proceedings of the 8th International Modelica Conference*. Dresden, Germany: Linköping University Electronic Press, 2011, pp. 153–159.

[31] B. Aktan, C. Bohus, L. Crowl, and M. Shor, “Distance learning applied to control engineering laboratories,” *IEEE Transactions on Education*, vol. 39, no. 3, pp. 320–326, Aug 1996.

[32] A. V. Parkhomenko, O. Gladkova, E. Ivanov, A. Sokolyanskii, and S. Kurson, “Development and application of remote laboratory for embedded systems design,” *International Journal of Online Engineering (iJOE)*, vol. 11, no. 3, pp. 27–31, 2015.

[33] I. Iturrate, G. Martín, J. García-Zubia, I. Angulo, O. Dziabenko, P. Orduña, G. Alves, and A. Fidalgo, “A mobile robot platform for open learning based on serious games and remote laboratories,” in *2013 1st International Conference of the Portuguese Society for Engineering Education (CISPEE)*. Porto, Portugal: IEEE, 2013, pp. 1–7.

[34] I. Santana, M. Ferre, E. Izaguirre, R. Aracil, and L. Hernandez, “Remote laboratories for education and research purposes in automatic control systems,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 547–556, 2013.

[35] D. Lowe, S. Murray, L. Weber, M. de la Villefromoy, A. Johnston, E. Lindsay, W. Nageswaran, and A. Nafalski, “LabShare: towards a national approach to laboratory sharing,” in *Proceedings of the 20th Annual Conference for the Australasian Association for Engineering Education*. Adelaide, Australia: The School of Mechanical Engineering, University of Adelaide, 2009, pp. 458–463.

Sidebar: This work at a glance

. Online labs are revolutionary tools that produce both economic and educational benefits. On the one hand, labs can be shared among institutions, drastically reducing traditional lab costs. On the other hand, students can access experimental resources 24/7 from anywhere through their electronic devices. This paper presents an open source online lab for the Furuta Pendulum, a very popular system in control engineering education that helps to study interesting problems related to non-linear dynamics and control theory.

Thanks to the experimentation environment described in this article, students can define new controllers and validate them, program non-trivial experiments, plot real-time data streams, and avoid repetitive work. The environment main component is a language designed specifically for programming experiments easily and visually. As the environment is online itself, it runs on any device with a web browser and avoids any software installation.

The environment is illustrated solving two practices on the Furuta Pendulum online lab: 1) defining a state feedback linear control law that keeps the pendulum in a vertical and upwards position while its base follows position set-point changes, and 2) designing a more complex controller that swings the pendulum upwards when it falls.

Author Biography

Daniel Galan received the M.Sc. degree in Automation and Robotics from the Polytechnic University of Madrid in 2013 and Ph.D. in Computer Engineering and Automatic Control in 2017 from UNED. Currently, he is working with the Department of Computer Science and Automatic Control of the Computer Engineering School of UNED. His research interests include robotics, virtual and remote labs, education technologies, interactive applications and virtual reality.

Dictino Chaos received M.Sc. Degree in Physics from the Complutense University of Madrid in 2004, and Ph.D. in Computer Engineering and Automatic Control in 2010 from UNED. Currently, Dr. Chaos holds an Assistant Professor position with the Department of Computer Science and Automatic Control of UNED. His research interests include virtual and remote labs, nonlinear control, tracking, point stabilization, and path following of under-actuated vehicles, the stability of switched systems and robotics.

Luis de la Torre received the M.Sc. degree in physics from the Complutense University of Madrid, Madrid, Spain, in 2008, and the Ph.D. degree in computer science from the Universidad Nacional de Educacion a Distancia (UNED), Madrid, in 2013. He is an Assistant Professor with the Department of Computer Sciences and Automatic Control, UNED. His research interests include virtual and remote labs, distance education, and http protocols and technologies.

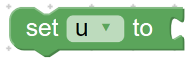
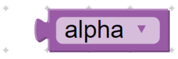
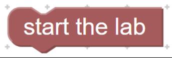
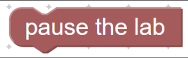
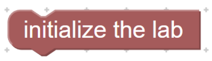
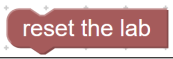
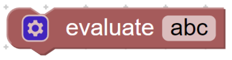
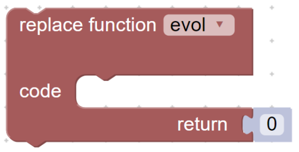
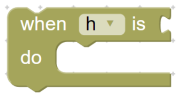
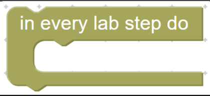
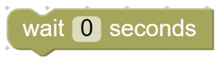
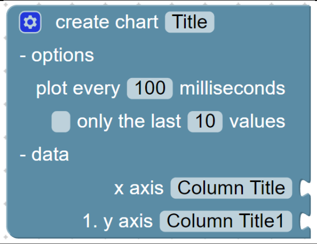
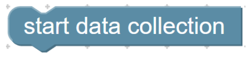
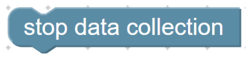
Ernesto Aranda-Escolastico received the M.S. degree in Physics from the Complutense University of Madrid, Spain, in 2013, the M.S. degree in Control Engineering from UNED, Spain, in 2014, and the Ph.D. degree in Computer Engineering and Automatic Control from UNED in 2018. His research interests include event-triggered control, networked control systems, multi-rate systems and nonlinear systems.

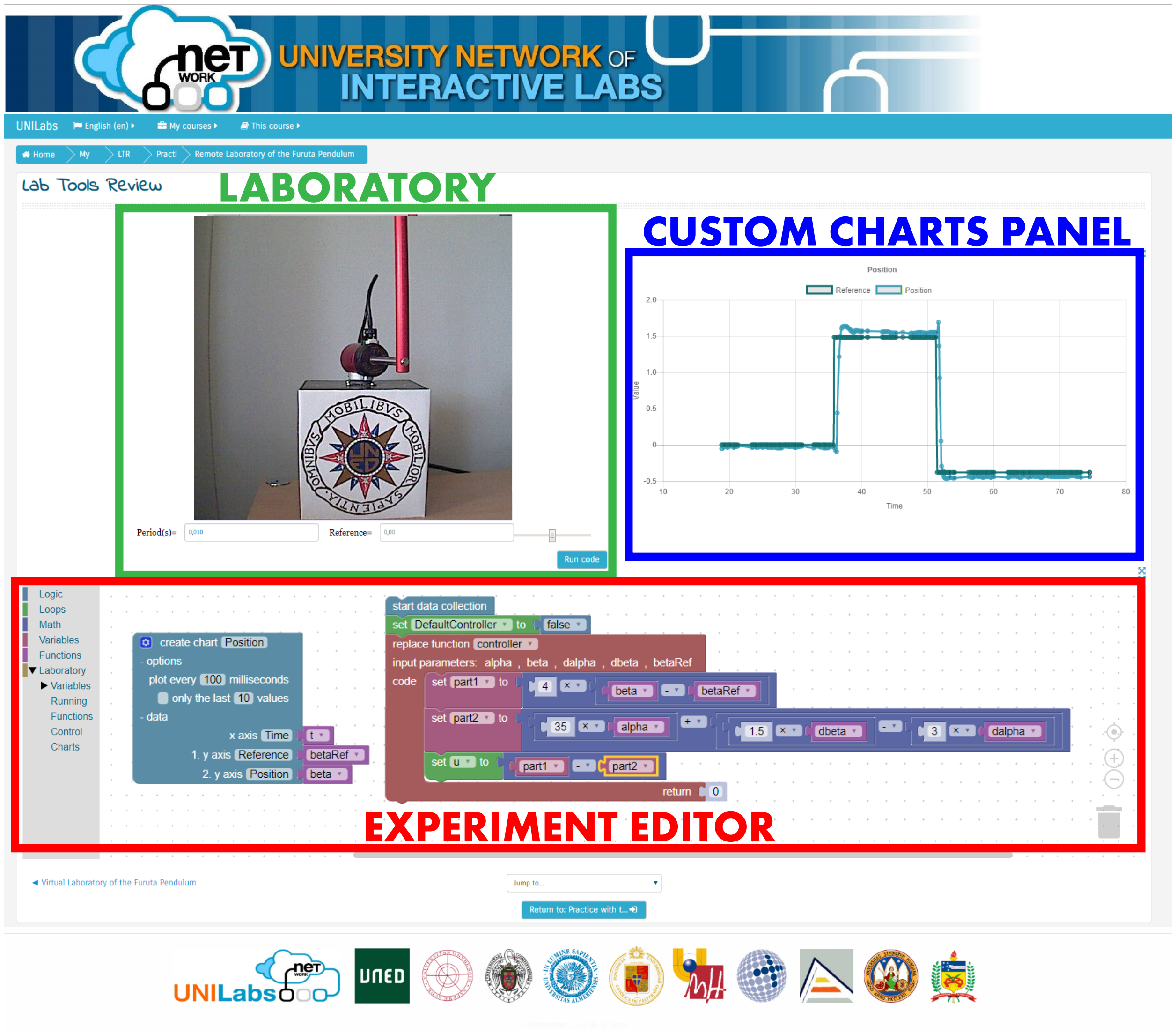
Ruben Heradio received the M.Sc. degree in Computer Science from the Polytechnic University of Madrid, Spain, in 2000; and the Ph.D. degree in Software Engineering and Computer Systems from UNED in 2007. Currently, Dr. Heradio is Associate Professor at the Software Engineering and Computer Systems Department of the UNED Computer Engineering School. His research and teaching interests include Software Engineering, Computational Logic, and e-Learning.

TABLE 1: Comparative summary of state-of-the-art experimentation scripting support for online labs. The last row summarizes the environment this paper presents

Name	**Description**	**General Features**				**Language Features**			
		Coupling	**Usability**	**Reusability**	**Embedded**	**Name**	**Learnability**	**Power**	**Domain**
TriLOGI [26]	It is a simulator to program PLCs with contact diagrams.	↓	-	↑	✗	PLC graphical scripts	↓	↓	DSL
UTS remote PLC [27]	It is a remote lab where students can write down PLC scripts to interact with pneumatically driven cylinder apparatus.	↑	↓	↑	✓	PLC instruction list scripts	↓	↑	GPL
Remote Lab Experiment [28]	It allows users to create and compile their own assembly code for a 8051 micro-controller.	↓	↓	↑	✗	Assembly code	↓	↑	GPL
Automatic Control Telelab [29]	It offers a set of different Control experiments where users are able to upload their own controller created with MatLab and Simulink.	↑	-	↓	✗	MatLab	↓	↑	GPL
OMWeb [30]	It is a web-based teaching environment where users can edit Open Modelica scripts to solve exercises proposed by the instructor.	↑	↓	↓	✓	Modelica	↓	↑	GPL
Second Best to Being There [31]	It is the first remote laboratory developed. Students were able to upload a controller coded with C.	↑	↓	↓	✗	C	↓	↑	GPL
RELDES [32]	It is an online remote laboratory where students can upload Arduino code to solve some proposed exercises.	↑	↓	↓	✗	Arduino	↓	↑	GPL
WebLab-Deusto [33]	An experience offered through WebLab-Deusto to control a mobile robot using Blockly.	↑	↑	↓	✓	Blockly	↑	↓	DSL
SLD [34]	Using Simulink files users can control systems modeled by the instructor.	↑	↑	↓	✗	Simulink	↑	↓	DSL
iLab [2]	Using Xilinx and VHDL schemes users are able to define specific procedures for some elements of the plant.	↑	↓	↓	✓	Xilinx-VHDL	↓	↑	GPL
LabShare Sahara [35]	Users are able to define Java algorithms to develop controllers that must fulfill certain predefined constraints.	↑	↓	↑	✓	Java	↓	↑	GPL
UniLabs [18]	It is possible to define a controller with JavaScript code for a two coupled electric drives system.	↑	↑	↑	✓	JavaScript	↓	↑	GPL
Experiment Editor (EE)	The environment proposed in this paper.	↓	↑	↑	✗	JavaScript & Blockly	↑	↑	GPL&DSL

TABLE 2: New blocks added to the Blockly base distribution to define experimentation scripts on online labs.

	Description
	It is used to set the selected laboratory variable to the desired value. In this case, users will set the value of the control signal “u” for the Furuta pendulum controller.
	It returns the value of the selected lab variable. In this case users will obtain the value of the angle “alpha” from the Furuta pendulum laboratory.
	It starts the evolution of the laboratory.
	It pauses the evolution of the laboratory.
	It sets all variables with the predefined initial value and restart the evolution of the laboratory.
	It resets the laboratory.
	It executes the “eval” JavaScript function with the desired code written in the block text-area.
	It replaces a laboratory function with the code specified inside the block. The parameters of the function will appear below the name.
	It adds a new event, the condition is an state variable from the evolution of the system compared with the desired value and the action is the code added inside the block.
	It adds the desired statements to the code executed every step of the laboratory evolution.
	It pauses the execution of the following blocks the amount of seconds desired.
 create chart Title - options plot every 100 milliseconds only the last 10 values - data x axis Column Title 1. y axis Column Title1	It is used to define the chart where the values are going to be represented.
	It starts the data representation in the chart.
	It stops the data collection.



```
create chart Position
- options
    plot every 100 milliseconds
    only the last 10 values
- data
                x axis Time t
          1. y axis Reference betaRef
          2. y axis Position beta
```

```
start data collection
set DefaultController to false
replace function controller
input parameters: alpha , beta , dalpha , dbeta , betaRef
code  set part1 to 4 × beta - betaRef
      set part2 to 35 × alpha + 1.5 × dbeta - 3 × dalpha
      set u to part1 - part2
                                     return 0
```

Figure 1: General view of the user interface elements of the proposed Experimentation Environment. The lab (in this case, the remote version of the Furuta pendulum) is shown in a green square. The experiment editor is marked in red, it can be seen an experiment coded with Blockly. The plot obtained from the execution of this experiment is shown in blue.

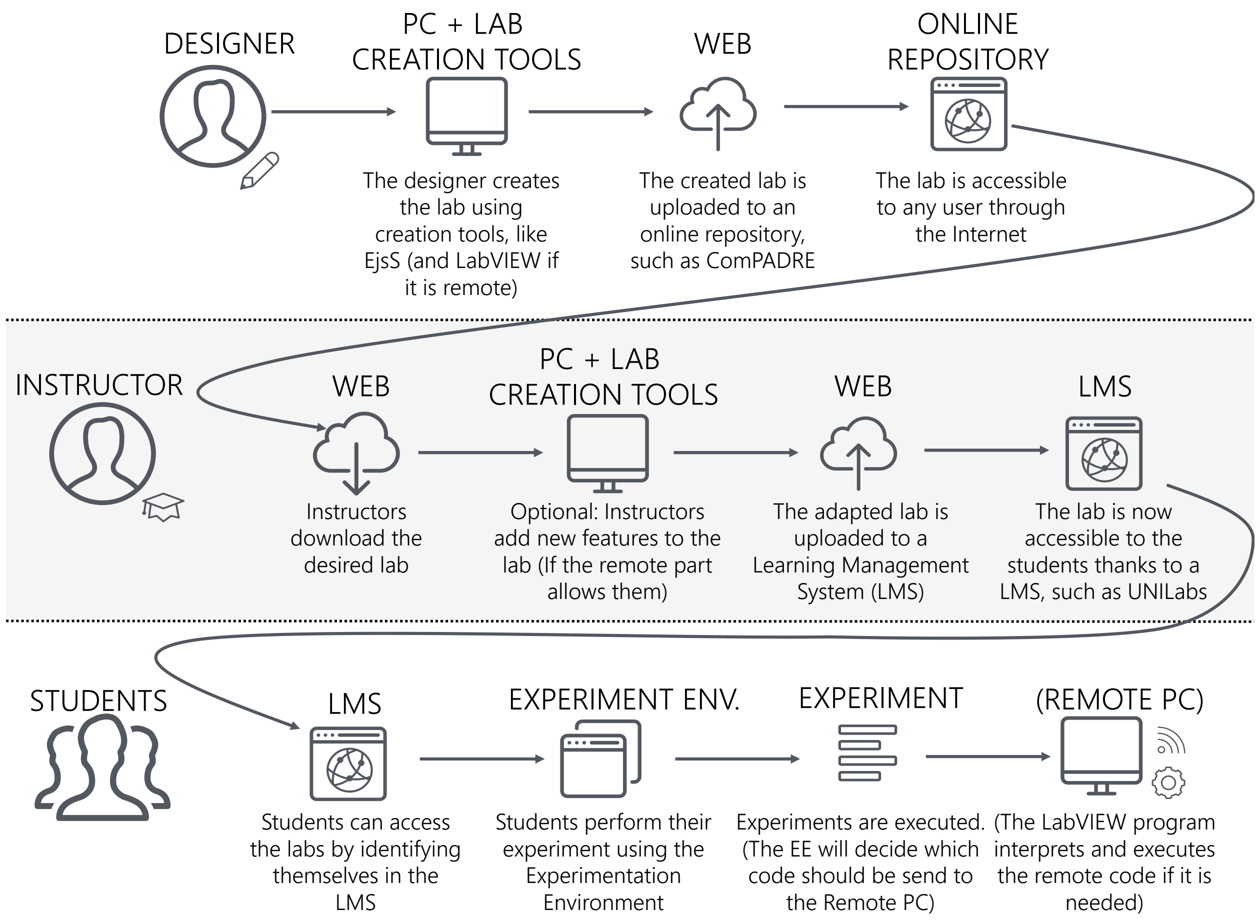

DESIGNER
PC + LAB CREATION TOOLS
WEB
ONLINE REPOSITORY
The designer creates the lab using creation tools, like EjsS (and LabVIEW if it is remote)
The created lab is uploaded to an online repository, such as ComPADRE
The lab is accessible to any user through the Internet
INSTRUCTOR
WEB
PC + LAB CREATION TOOLS
WEB
LMS
Instructors download the desired lab
Optional: Instructors add new features to the lab (If the remote part allows them)
The adapted lab is uploaded to a Learning Management System (LMS)
The lab is now accessible to the students thanks to a LMS, such as UNILabs
STUDENTS
LMS
EXPERIMENT ENV.
EXPERIMENT
(REMOTE PC)
Students can access the labs by identifying themselves in the LMS
Students perform their experiment using the Experimentation Environment
Experiments are executed. (The EE will decide which code should be send to the Remote PC)
(The LabVIEW program interprets and executes the remote code if it is needed)

Figure 2: Online lab life cycle: from its creation by the designer to its use by students going through its preparation as a learning object by the instructor. Steps related to the use of remote laboratories are shown in brackets.

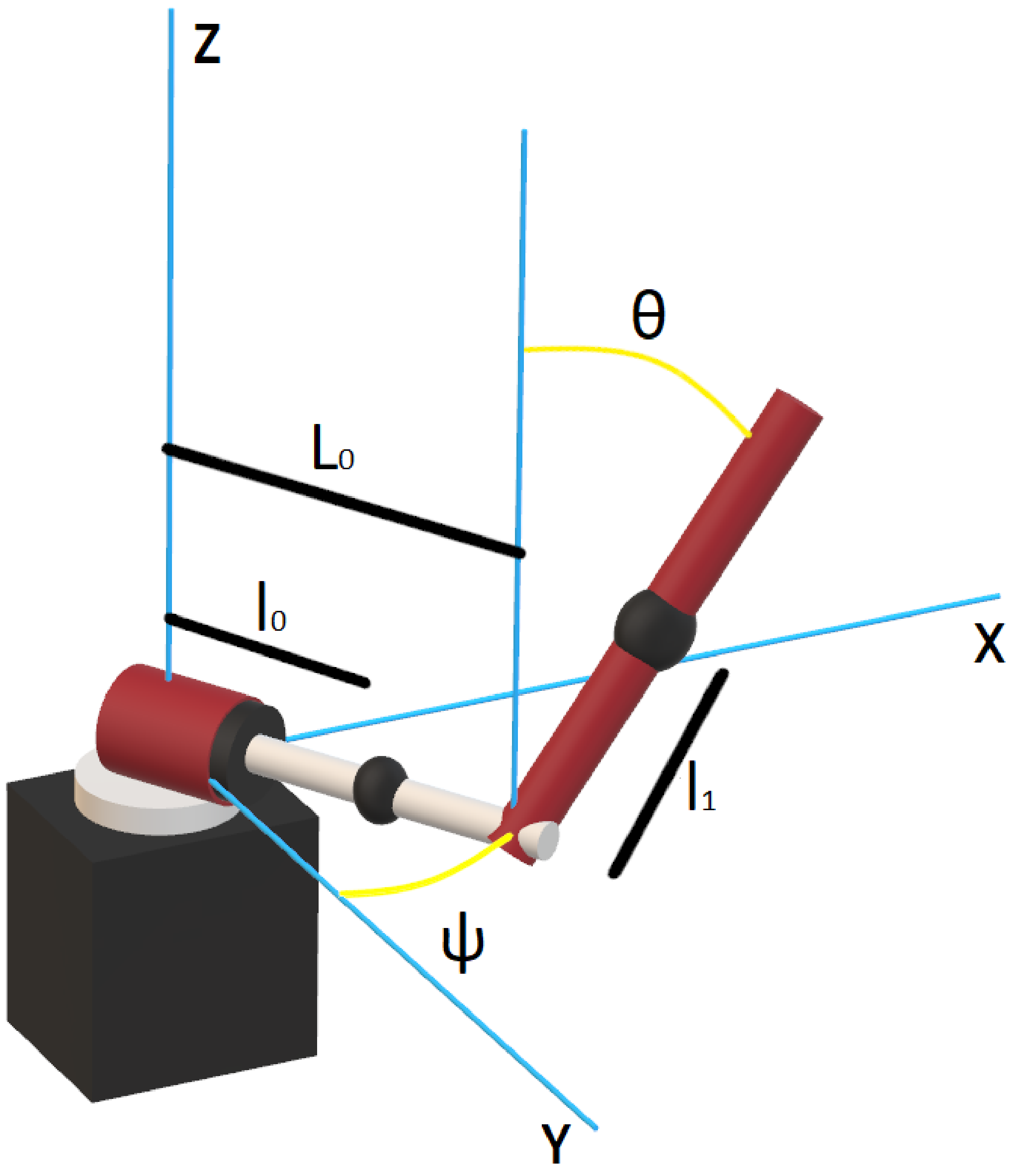
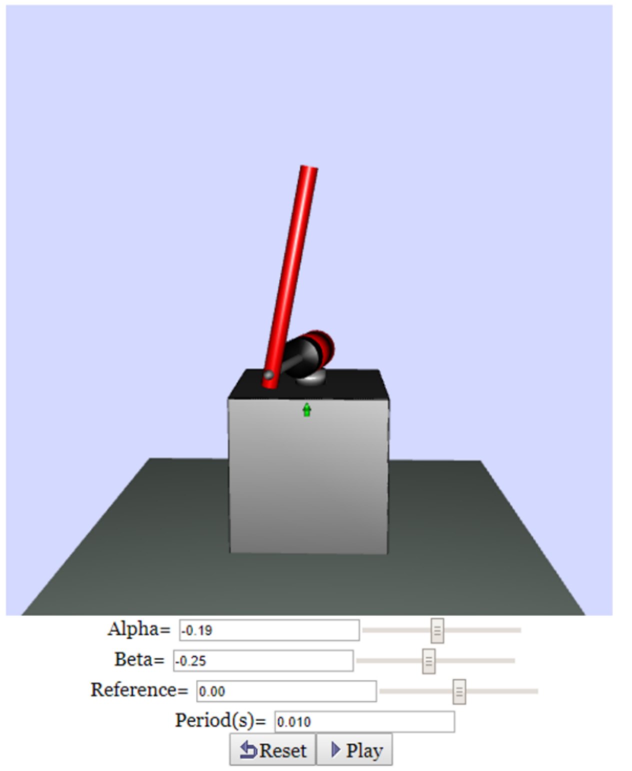


Figure 3: Schematic representation of the Furuta pendulum model. The imaginary axes (x, y, and z) are represented in blue. The angles (*theta* and *psi*) formed between the axes and the arms are painted in yellow, the measurements at the center of masses of each arm (l_0 and l_1) and the total length of the rigid link (L_0) are represented in black.



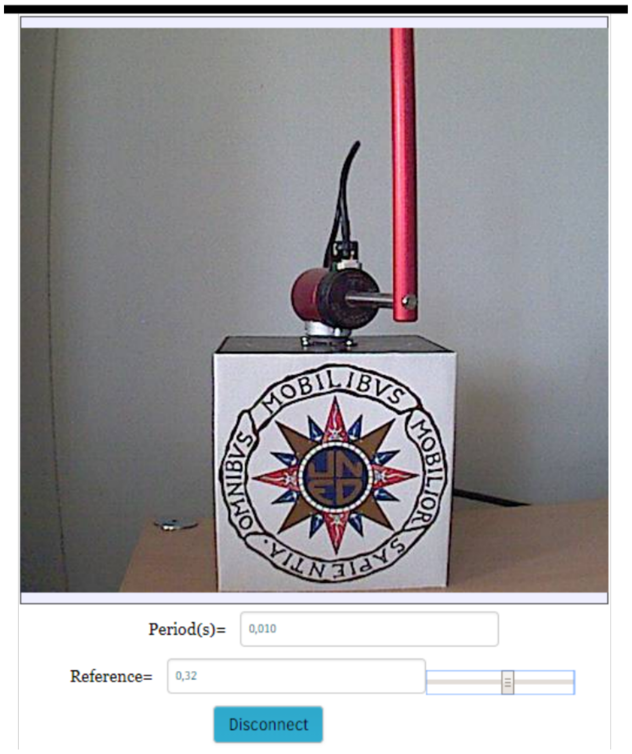
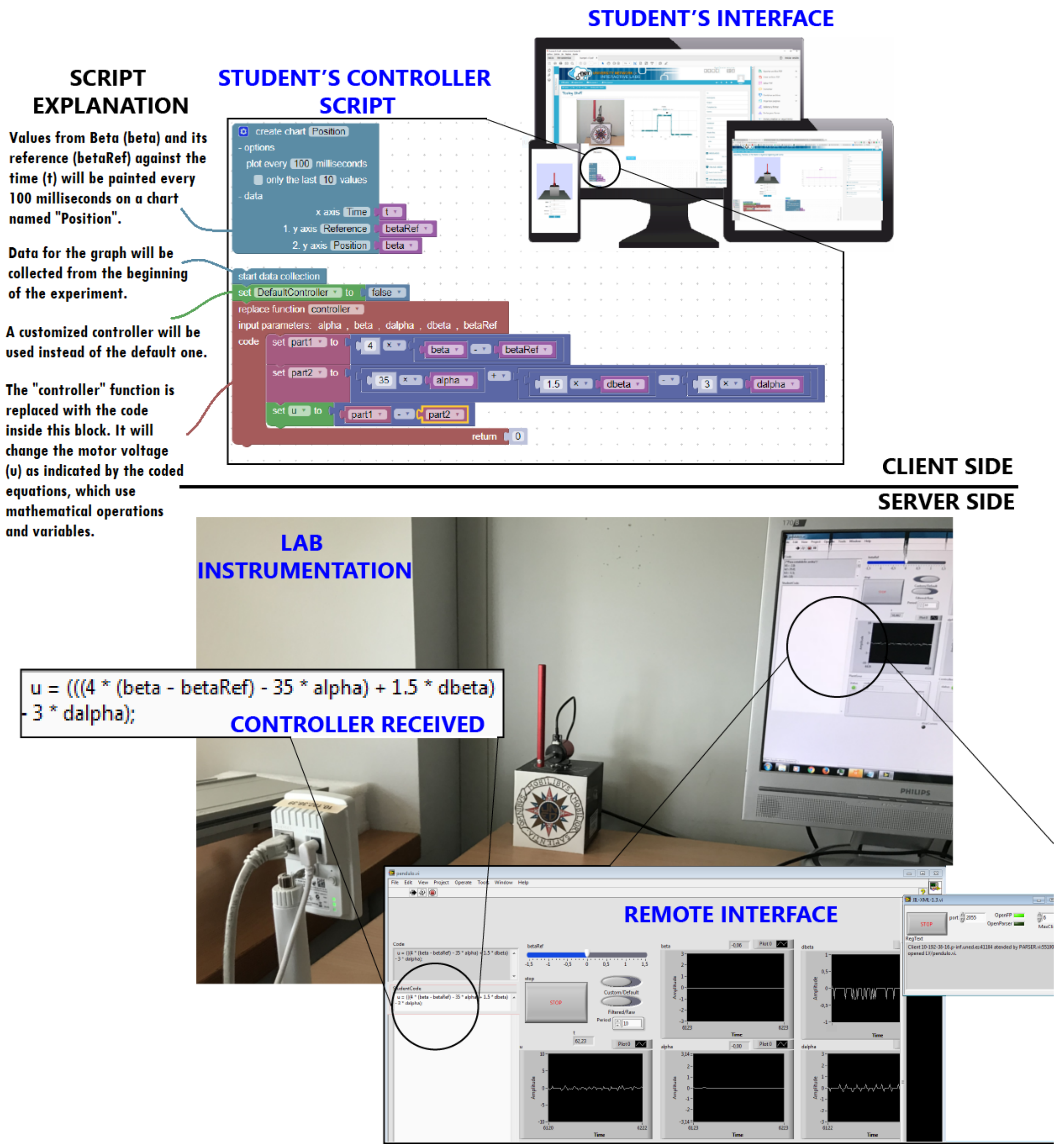


Figure 4: Virtual (up) and remote (down) labs for the Furuta pendulum. GUIs consist of a visual feedback area situated at the top, and the model parameters control area at the bottom. In the case of the remote lab, web-cam images are displayed, and any change in the GUI is automatically sent to the plant.



Values from Beta (beta) and its reference (betaRef) against the time (t) will be painted every 100 milliseconds on a chart named "Position".

Data for the graph will be collected from the beginning of the experiment.

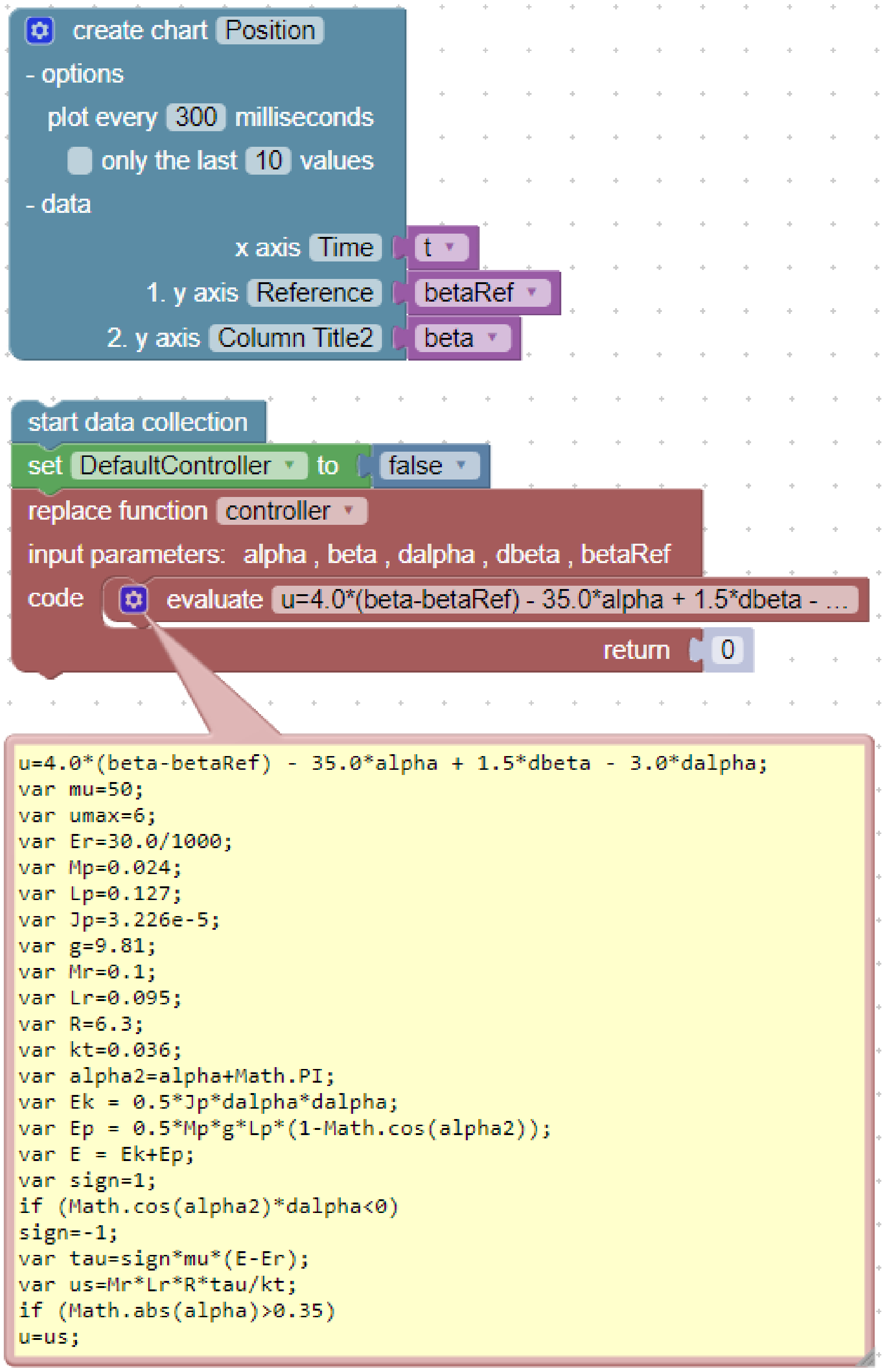
A customized controller will be used instead of the default one.

The "controller" function is replaced with the code inside this block. It will change the motor voltage (u) as indicated by the coded equations, which use mathematical operations and variables.

```
create chart Position
- options
    plot every 100 milliseconds
    only the last 10 values
- data
    x axis Time  t
    1. y axis Reference  betaRef
    2. y axis Position  beta
start data collection
set DefaultController to false
replace function controller
input parameters: alpha , beta , dalpha , dbeta , betaRef
code  set part1 to  4 × ( beta - betaRef )
      set part2 to  ( 35 × alpha ) + ( 1.5 × dbeta ) - ( 3 × dalpha )
      set u to  part1 - part2
return 0
```

```
u = (((4 * (beta - betaRef) - 35 * alpha) + 1.5 * dbeta)
- 3 * dalpha);
```

Figure 5: Client and server side of the Furuta remote lab when the position controller experiment is executed


create chart Position
- options
plot every 300 milliseconds
only the last 10 values
- data
x axis Time
t
1. y axis Reference
betaRef
2. y axis Column Title2
beta
start data collection
set DefaultController to false
replace function controller
input parameters: alpha , beta , dalpha , dbeta , betaRef
code
evaluate u=4.0*(beta-betaRef) - 35.0*alpha + 1.5*dbeta - ...
return 0

```
u=4.0*(beta-betaRef) - 35.0*alpha + 1.5*dbeta - 3.0*dalpha;
var mu=50;
var umax=6;
var Er=30.0/1000;
var Mp=0.024;
var Lp=0.127;
var Jp=3.226e-5;
var g=9.81;
var Mr=0.1;
var Lr=0.095;
var R=6.3;
var kt=0.036;
var alpha2=alpha+Math.PI;
var Ek = 0.5*Jp*dalpha*dalpha;
var Ep = 0.5*Mp*g*Lp*(1-Math.cos(alpha2));
var E = Ek+Ep;
var sign=1;
if (Math.cos(alpha2)*dalpha<0)
sign=-1;
var tau=sign*mu*(E-Er);
var us=Mr*Lr*R*tau/kt;
if (Math.abs(alpha)>0.35)
u=us;
```

Figure 6: Swing up and position control experiment

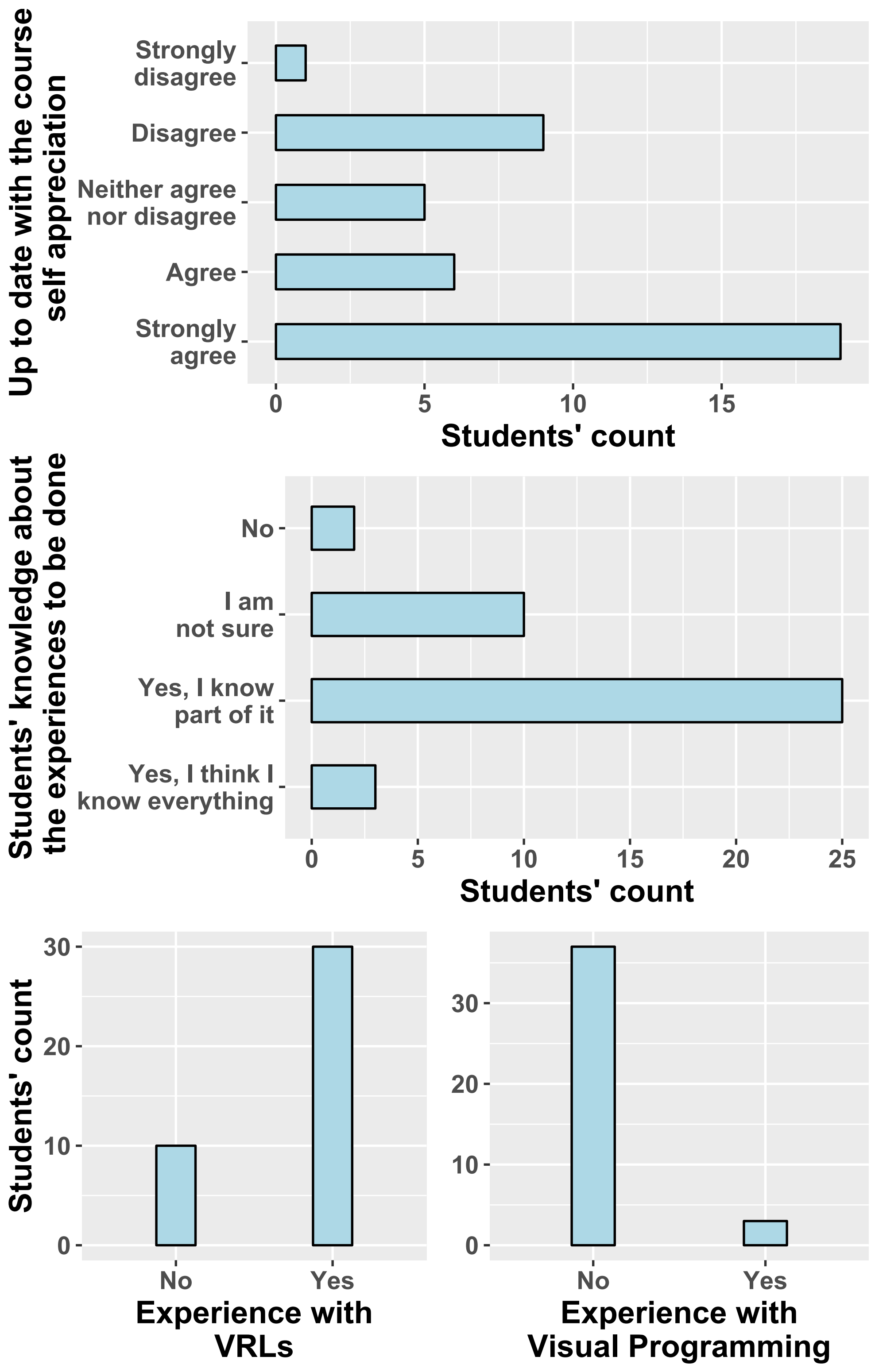


Figure 7: Results from the first students' evaluation phase to determine their background before undertaking the experiences.

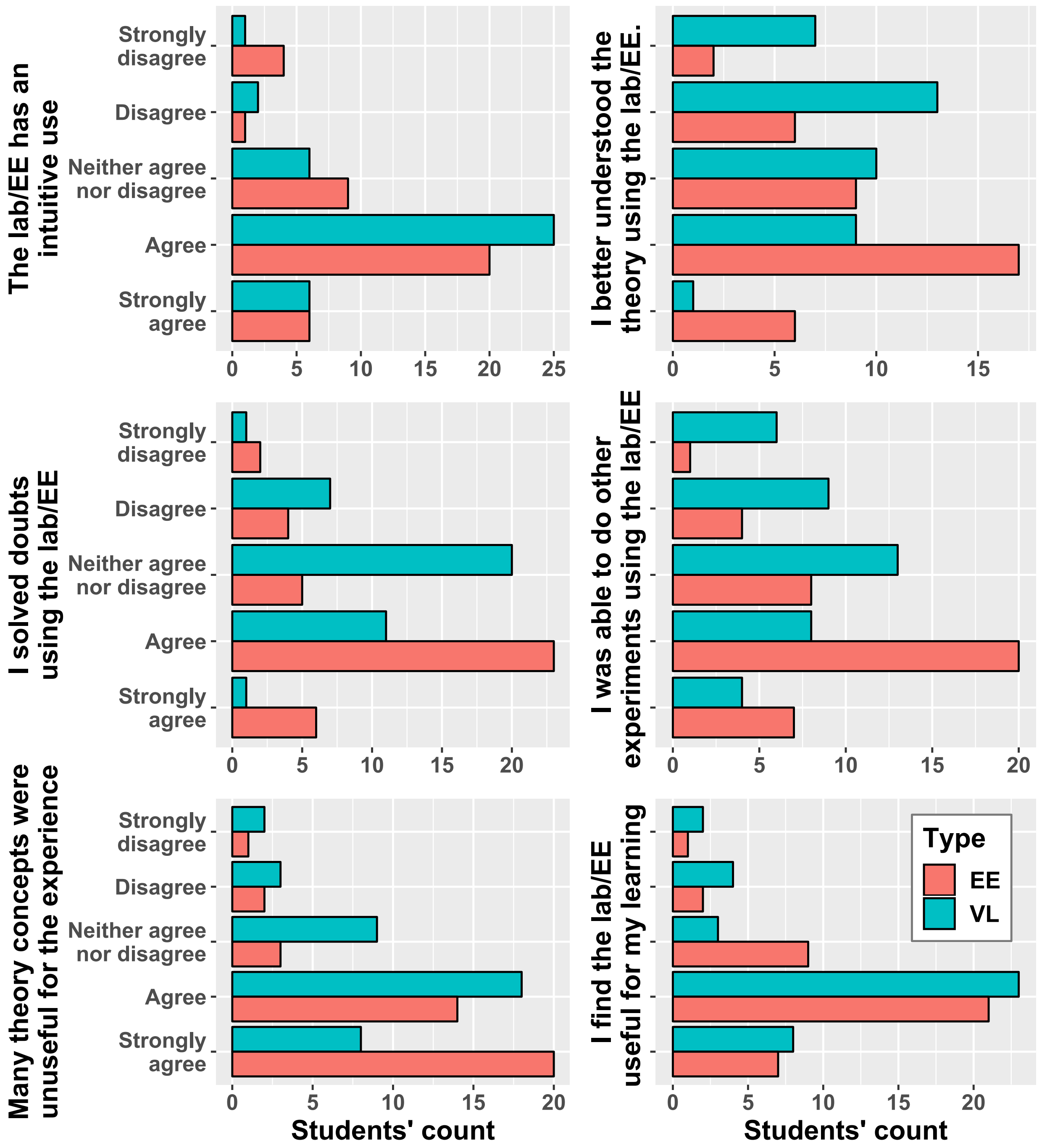


Figure 8: Students' perception of the most representative questions about the Furuta pendulum virtual lab lab and the EE.