Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-17 (2022) 326–331

Efficient deployment of remote laboratories with TwinCAT-PLCs and EjsS Plugins ⋆

Jesús Chacón * **Eva Besada-Portas** * **Lía García-Pérez** * **José A. López-Orozco** *

* *Department of Computer Architecture and Systems Engineering, Universidad Complutense of Madrid, Madrid, 28040 Spain (e-mails: jeschaco@ucm.es, liagar05@ucm.es, ebesada@ucm.es, jalopez@ucm.es).*

Abstract: This paper describes a new approach to streamline the development of Remote Laboratories (RL) for Control Education based on TwinCAT Programmable Logic Controllers (PLCs) and Easy JavaScript Simulations (EJsS). On one hand, the TwinCAT PLC is used to implement the laboratory back-end (responsible of closing the feedback loop over the plant under study) with industrial real-time automation methodologies. On the other hand, EJsS is used: 1) to define the RL Human-Machine Interface (HMI) front-end (used by the students to parametrize & observe the evolution of the PLC behavior and signals) and 2) *to smooth and centralize the majority of the tasks that the instructors have to perform to configure and deploy the RL*. This second utility of EJsS, a novelty of the RL presented in this paper, is supported by an EJsS *Plugin* that has been especially designed with that purpose. Besides, it is worth noting that our new RL is supported by an improved version of *ReNoLabs*, developed in 2016 to be a lightweight middle-ware of RLs with EJsS HMI webpages and different types of back-ends, whose Node.js-based server has been updated to backup the functionality of the EJsS *Plugin*. This paper also shows how to use our approach to setup a RL and describes the main characteristics of two RLs that have been updated with it.

Keywords: Virtual and Remote Labs; Internet-Based Control Education; EJsS; TwinCAT PLCs

1. INTRODUCTION

Laboratory practices allow students to face real problems that illustrate relevant aspects of the theory and to transmit useful working skills for their future (Ma and Nickerson, 2006). However, these benefits are often limited by the cost of replicating the lab hardware for the students and by the scarce hours of practical sessions (Gomes, 2009).

Remote Laboratories (RL), which give Internet access to the lab devices, mitigate these inconveniences by increasing the availability, accessibility and security of the experiences. In more detail, they allow students to interact with the system under study and to observe the evolution of its variables, from home and with personalized schedules. This fact has been exploited by the educational institutions during the COVID-19 restrictions, which have also boosted the development of RLs, although previous studies already showed that remote and face-to-face labs can provide similar educational benefits (Kostaras et al., 2011; de la Torre et al., 2016; Faulconer and Gruss, 2018).

RLs for Control Education usually implement a multi-layer architecture as Fig. 1 shows, where the back-end is connected/interacting to/with the system under study, the front-end provides the Human-Machine-Interface (HMI) used by the students to conduct the lab experience, and

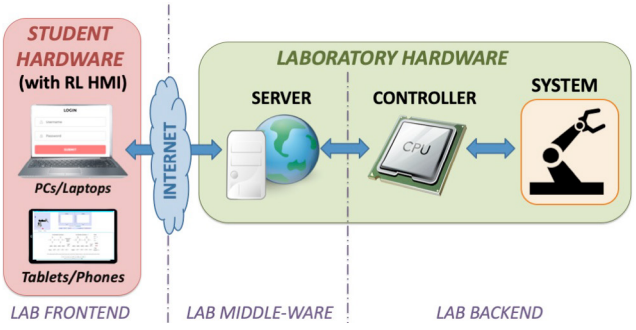

Fig. 1. Schema of the usual architecture of RLs.

the middle-ware provides different services, including the information exchange between the front and back ends. As the software/hardware that implements/runs each layer vary, different types of RLs (often supported by open-source software and low-cost computing devices) are developed and presented to the educational community (Zubía and Alves, 2011; Sáenz et al., 2015; Heradio et al., 2016).

A less usual alternative in RLs consists on using industrial programming units (such as Programmable Logic Controllers - PLCs- or Process Control Systems - PCSs-) in the back-end. The benefits of this choice are multiple: it lets the labs re-use elements already available for other subjects and it shortens the gap between the technology used in Academia and Industry (Leva et al., 2020). In spite

⋆ This work has been funded by the Educational Innovation program of the University Complutense de Madrid under the 2021-39 project.

10.1016/j.ifacol.2022.09.300

Table 1. Characterization of RLs with Industrial back-ends

Work	Back-end	Communication	Middle-ware	Front-end
Kalúz et al. (2015)	PLC		AJAX Web application	HTML+JavaScript webpage
Bellmunt et al. (2006)	PLC	Modbus TCP/IP	——	PLC-proprietary software
Sánchez-Herrera et al. (2020)	PLC, others	Modbus TCP/IP	SARLAB Server	EJsS webpage
Klein and Wozny (2006)	PCS	OPC	Multiple layers	Java applet
Prada et al. (2015)	PLC	OPC	Multiple servers (data, web, proxy)	Web interface
González et al. (2016)	PLC	OPC	Multilayer server (OPC, LabView & Java)	EJS applet
Besada-Portas et al. (2012, 2013b)	TwinCAT PLC	ADS	Java server	EJS applet
Bermudez-Ortega et al. (2016)	TwinCAT PLC, others	ADS	Node.js multipurpose (data, access, web) server	EJsS HTML+JavaScript webpages

of the scarcity of these labs, they can also be characterized, as the summary in Table 1 shows [1], according to the type of industrial back-end [2], the communication protocol between the PLC and other elements [3], the middle-ware architecture and software [4], and the HMI of the front-end [5]. Table 1 shows that there is also variability on the technologies involved in these RLs, although PLCs, industrial communication protocols, and Easy Java/JavaScript Simulations (EJS/EJsS) applets/webpages [6] are often used.

The RL presented in this paper is an extension of the one characterized in the last row of the table that takes advantage of a renewed server and the capability of enhancing the functionality and Graphical User Interface (GUI) of EJsS by means of *Plugins* (Chacón et al., 2021), in order to streamline the deployment of the RL. In more detail, although the RL in Bermudez-Ortega et al. (2016) was completely operational from the point of view of the students, the instructors in charge of its development needed to re-code part of the server, manually modify the HTML+JavaScript webpage generated in EJsS, and edit multiple files in the computer where the server was running. In the current version of the RL, all these steps are smoothed and centralized from EJsS, which was already the tool that the RL developers use to design the webpages of the experiences.

The remaining of the paper is organized as follows. Section 2 presents an overview of the whole RL and of its elements. Section 3 describes how to configure and deploy a new RL with our approach. Section 4 shows two RLs that have been updated with it. Section 5 draws the conclusions and presents some future lines of research.

2. LABORATORY OVERVIEW

The RL presented in this paper is an improved version of *ReNoLabs*, whose initial architecture, introducted in Besada-Portas et al. (2016) and particularized for PLCs in Bermudez-Ortega et al. (2016), constitutes a Remote Laboratory Manament System (RLMS) with: 1) different types of back-ends (e.g. PLCs, LabVIEW virtual instruments or applications programmed with generic languages); 2) a Node.js web-server middle-ware with multiple purposes (including user-access-control and data-logging); and 3) an interactive HMI for the experiences developed in EJsS.

Our current RLMS makes use of a EJsS *Plugin* and a renewed Node.js server that have been especially designed with the purpose of centralizing from EJsS, as far as possible, the configuration and deployment of our RLs.

In the following, we briefly present the different software elements involved in the RL and their main features.

2.1 TwinCAT PLCs

The back-end of our RLs are implemented with a TwinCAT [7] PLC that is connected to the system under study through an EtherCAT bus and different terminal cards (e.g. incremental encoder, and analog to/from digital converters). Its functionality is programmed in Structured Text [8] (ST) and consists on implementing the main loop of a controller (i.e. measure the system outputs, obtain the control signals, and apply them to the system inputs). Its variables are accessible from other PLCs or applications (e.g the web-server of our RL) using the ADS protocol, which manages the distribution of information between different devices connected to a TCP/IP network.

Finally, it is worth noting that the PLC configuration and programming has to be performed with the software provided by TwinCAT. Moreover, as this part of the RLMS remains unchanged, readers are referred to Bermudez-Ortega et al. (2016) for further information.

2.2 Node.js ReNoLabs Server

ReNoLabs server is the middle-ware of our RLMS architecture and includes the capabilities that are required to put our RLs into production. Its features can be divided into two sets, as Fig. 2 represents. The first group (at the top of

[1] Some additional works can be found in Zubía and Alves (2011).
[2] The label *others* in the second column of the table means that the approach does not only support industrial back-ends.
[3] *OPC* stands for Object-linking and embedding for Process Control, while *ADS* stands for the Automation Device Specification.
[4] This layer complexity depends on the functionality of the RL and its sofware is restricted by the one used at its extreme-ends.
[5] The second work on the table use the PLC-proprietary software to let students perform the same types of experiences as in the face-to-face lab. The remaining works have a HMI that is especially designed for the experiences available in the RL.
[6] Web-browsers stopped supporting applets in 2015. Hence, EJsS HTML+JavaScript webpages became the HMI of the newest RLs.
[7] `https://www.beckhoff.com/en-en/products/automation/`. TwinCAT is a powerful and well-known tool to develop industrial controllers, run them on a Windows-adapted-for-real-time PC, and observe/modify the values of their variables during its execution.
[8] To this end, other PLC-programming languages, supported by TwinCAT, could also be used.

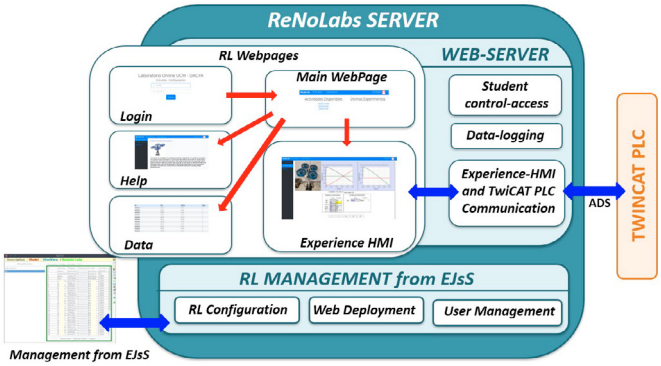

Fig. 2. Server main functionality.

the figure) includes the functionality required by the students to perform the experiments through the web-pages of the RL. The second set (at the bottom) implements the utilities that are used by the lab instructors to configure and deploy the lab from EJsS.

In more detail, from the point of view of the students, *ReNoLabs* acts as a server of the following webpages: available experiences and experience login, help, data and interactive HMI. The functionality underneath controls the access of the students (only to their available not-in-use experiences), logs the experimental data (consistent on the evolution of the variables of interest of each experience) and communicates the interactive HMI of each experience with its corresponding TwinCAT PLC. All the functionality except the last one has been developed for *ReNoLabs* RL with different types of back-ends (Besada-Portas et al., 2016; Bermúdez-Ortega et al., 2017). The last functionality relies on the new *TwinCAT Hardware Adapter for ReNoLabs*, which encapsulates and handles the communications between the server and the PLC by means of the ADS protocol [9] and the `node-ads` package [10]. Moreover, in order to let the server know the target PLC and which of its variables read or write during the experience, the lab instructors will need to provide information about 1) the connecting elements (server and PLC) according to the ADS protocol, and about 2) the variables to synchronize. Further details on this point will be provided in Section 3.

On the other side, *ReNoLabs* acts as a RLMS for the instructors and the server provides the functionality that allows them to use EJsS and its *ReNoLabs Plugin* to: 1) configure the communications between the server and the PLC (indicating the ADS connection and the PLC variables of interest), 2) define and deploy the help webpage and interactive HMI of the experience, and 3) manage the user-access permissions. In other words, in order to let EJsS modify/adapt the behavior of the web-server of *ReNoLabs*, the functionalities incorporated into EJsS by *ReNoLabs Plugin* have had to be backed up by new functionalities in the management side of *ReNoLabs*.

Finally, it is worth noting that *ReNoLabs* has be implemented in Node.js [11], because 1) it was already used in Besada-Portas et al. (2016) and Bermudez-Ortega et al. (2016), and 2) it keeps on being excellent for building light-weight data- intensive real-time network applications (such as our server) deployable in any computer and operating-system (including the Windows PC that is running the PLC, or a low-cost platform such as the Raspberry Pi).

2.3 EJsS Experience-HMI and RL-Management

Easy JavaScript Simulations [12] is an open source tool initially developed to help users with limited programming skills to create interactive physical simulations. Its users only have to define the Model and the graphical View of the simulation, and let the tool automatically generate its corresponding webpage. EJsS is also frequently employed in Control Education, using the View to define the GUI of the interactive HMI of the RL and substituting the Model by calls to routines/methods that support the communications between the interactive webpage defined within EJsS and the RL server.

Furthermore, the functionality and GUI of version 6.0 of EJsS can be adapted by means of *Plugins* (Chacón et al., 2021). Exploiting this property, we have developed a *Plugin* that centralizes, in EJsS, the configuration and management of different aspects of *ReNoLabs*. More specifically, the *Plugin* adds the *Remote laboratory* tab to EJsS main menu, which gives access to different panels to specify the connection parameters (IP and access port) of the server, select the controller back-end (*TwinCAT Hardware Adapter*, in this case), indicate the PLC and the variables to interact, or manage the students-access permissions. The *Plugin* also incorporates a `Lab` class that lets the experience HMI that is being developed with EJsS access (read/write) the values of the variables that the server is synchronizing with the PLC. Moreover, it incorporates into EJsS a new panel with Graphical Elements ready to be used in the View of the HMI to automatically build the elements of its GUI that allow students to run/stop/reset the PLC or access the contents of some of its variables. Finally, the *Plugin* also adds a new item to EJsS button bar, which is used to deploy the Description and HTML+JavaScript webpage defined in EJsS, and use them as the Help webpage and interactive HMI of the experience. Next section details the process to follow to set up our RLs from EJsS.

3. LAB CONFIGURATION AND DEPLOYMENT

This section explains the set of parameters/settings that characterize how *ReNoLabs* and the students interact with the experimental hardware, as well as the process followed to define them [13].

3.1 Software Installation & Running

We start installing the software that support our RLMS: Node.js and *ReNoLabs* server in the platform where the server will run, and EJsS and *ReNoLabs Plugin* in the computer that the instructors use to set up the RL [14].

[9] Using this protocol has a side benefit: our server can connect to any back-end (and not only TwinCAT PLCs) that adheres to it.
[10] `https://www.npmjs.com/package/node-ads`
[11] `https://nodejs.org/en/`
[12] `https://www.um.es/fem/EjsWiki/`
[13] This section assumes that TwinCAT is installed, and that the PLC is programmed, running, and accessible from *ReNoLab* server.
[14] *ReNoLab* server and plugin will be available in `https://gitlab.com/jcsombria/pimcd-2020-2021`.

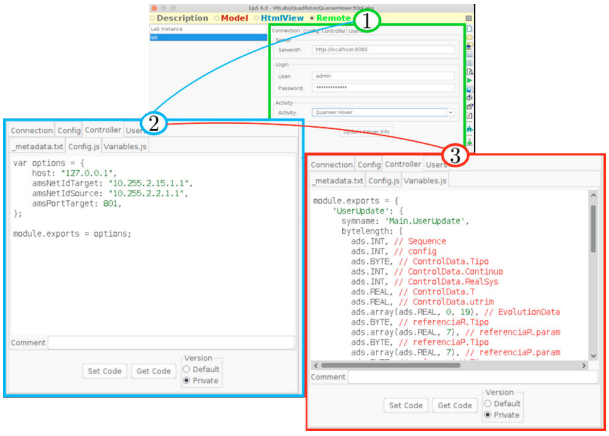

Fig. 3. Connecting EJsS to the server and configuring the PLC-Server Communication.

Next, we start the RL server running `node server.js` in the command line of the server computer, launch EJsS in the instructors computer, connect EJsS to the server introducing the server address and instructor credentials in the *Remote Lab* editor (highlighted in green in screenshot (1) of Fig. 3), and load `ejss` file of the experience of the RL that is stored in the server.

3.2 PLC-Server Communication Configuration

In order to let *ReNoLab* server access the PLC, it needs to know the TCP/IP address of the computer hosting the PLC, the ADS Network Identifiers of the target and source computers (i.e. where the PLC and server are running), and the ADS port of the PLC. This information is packed in the following structure, which can be down/up-loaded from/to the server and modified using the *Config.js* tab of the *Config* editor (displayed in screenshot (2) of Fig. 3).

```
module.exports.options = {
    host: "192.168.56.101",
    amsNetIdTarget: "10.255.2.15.1.1",
    amsNetIdSource: "10.0.2.15.1.1",
    amsPortTarget: 801    };
```

The variables to synchronize must be declared too [15], using a syntax similar to the one provided by `node-ads` module: for each variable to synchronize, we provide its PLC identifier in `synname` (prepending the program name to the variable name [16]), the number of bytes in `bytelength`, the name of the variable in Node.js in `propname`, and, optionally, the number of elements of this variable in `elements`. The information of multiple variable is packed in the following structure, which can be down/up-loaded from/to the server and modified in the *Variable.js* tab of the *Config* editor (highlighted in screenshot (3) of Fig. 3).

```
module.exports.variables = {
  'config': {
    symname: 'Main.ControlState',
    bytelength: [ads.INT],
    propname: ['value'],},
```

[15] Although this step depends on the variables defined in the PLC, if they are systematically named, we can reuse the configuration from one PLC to another.

[16] For example, `Main.ControlState` stands for the `ControlState` variable which is defined in the `Main` program of the PLC.

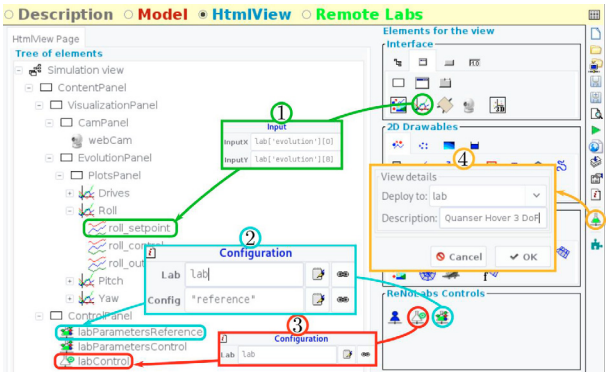

Fig. 4. Composing the *View* of the HMI of the experience.

```
  'evolution': {
    symname: 'Main.evolution',
    bytelength: [ads.REAL],
    propname: ['values'],
    elements: [14],},
  ...
}
```

3.3 Defining the help and HMI of the experience

Next, we build the *View* of the experience. To do it, we add and configure the usual EJsS visual elements to plot the evolution of the variables of interest in the experience (whose values are accessible, as screenshot (1) of Fig. 4 and its associated green-framed elements show, with the methods of the `Lab` object that is accessible since the initial connection of EJsS to the server) and show some images of the system under study (adding EJsS `WebCam view` elements). Afterwards, we add the *ReNoLabs* visual elements that automatize the generation of the remaining elements of the HMI of the experience: we use the new `LabParameter` widget to build the panels that let the students modify parameters of the PLC (e.g. reference signals and controllers), and the new `LabControl` widget to add the panel row of (run/stop/reset) buttons that control the PLC status. Both types of elements are parameterized with the `Lab` object, as screenshots (2) and (3) of Fig. 4, and their corresponding cyan/red-framed elements, show.

We also have to write in EJsS the *Description* of the experience to prepare the help webpage of the RL.

Finally, we press the `ReNoLabs` item of EJsS buttom bar, framed with an orange circle in Fig. 4, to upload to the server the EJsS file that contains the new/updated *View* and *Description* of the experience.

3.4 Putting the RL into production

At this point, the RL is almost ready to be used by the students. However, two steps remain. The first one is updating the user-access permission file, which is also editable from EJsS. The second one consists on enabling an activity (which is the combination of a EJsS *View* & *Description* with a TwinCAT PLC) in the RL activity webpage. This final step allows to make multiple activities available from the same *ReNoLabs* server, as well as to use the same PLC with different HMI to let the students focus on the variables/parameters of interest of each activity.

a) Quanser 3-DOF Hover b) Feedback 33-100 Unit

Fig. 5. Systems under study in the 2 RLs.

4. LABORATORY EXAMPLES

This section presents some details of the experimental setup, activities and experience-HMI of two RLs that use the current version of *ReNoLabs* with PLC back-ends.

4.1 Quanser 3-DOF Hover

The first RL lets students learn how to stabilize a quadrotor in a controlled laboratory environment. As previous versions of this RL have been presented in Besada-Portas et al. (2013a) and Bermudez-Ortega et al. (2016), this section is focused on outlining those aspects that help readers to understand how this system is controlled from the new RLMS.

The experimental setup. The Hover of 3 Degrees of Freedom (3-DOF) of Quanser [17], shown in Fig. 5, is a system consistent of a planar frame mounted in a pivot joint that allows the frame to rotate freely around the roll, pitch and yaw axes. The angle of each axis is measured with a high resolution optical encoder that belongs to the family of Beckhoff EtherCAT terminal-cards, and controlled with 4 DC-motors that drive its corresponding propellers and that are connected to a Beckhoff EtherCAT analog-output terminal-card. The system is controlled from a TwinCAT PLC programmed in ST.

The activities. This setup is complex enough to be used by students of different courses and help them to learn basic control concepts, such as Proporcional-Integral-Differencial (PID) regulators, in undergraduate degrees; or more advanced topics, such as Sliding Mode (SM) or Feedback Linearization (FL) control, in Master degrees (de la Torre et al., 2020). Throughout the different courses, the following activities are proposed: 1) identifying the system model observing the time-response to the reference signals, and 2) tuning & testing the PID, SM and FL controllers that are implemented in the PLC.

The experience-HMI webpage. To support the previous activities, the instructors have developed in EJsS the experience webpage presented in Fig. 6. At the top, the students can directly observe a image taken by an IP camera during the experience, as well as four plots: one shows the 4 voltages that are applied to the DC-motors/drives, while the 3 remaining ones show the reference signal and encoder

[17] `http://www.quanser.com/products/3dofhover`

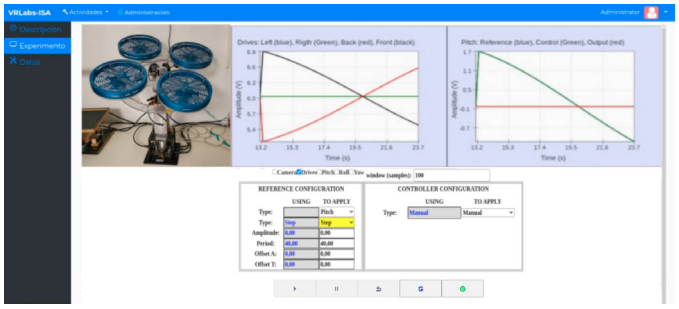

Fig. 6. Experience webpage of Quanser 3-DOF Hover RL.

measurement of each of the 3 DOF. With the selectors underneath, the students can enable/hide each image/plot to adapt the interface to their interest [18]. In the center, there are two parameterization panels, automatically built with the information provided by the server. The one at the left lets students select different reference signals, while the one at the right lets them choose the controller. Finally, at the bottom, there is a row of buttons, associated to *ReNoLabs* `LabControl` widget, that lets students start/stop/reset the PLC and, therefore, each experience.

4.2 Feedback 33-100 DC Motor

The second RL lets students learn different topics of Single Input Single Output (SISO) linear systems. Again, the section is focused on some aspects of the RL, as previous versions of it have has been presented in Besada-Portas et al. (2013b) and Bermudez-Ortega et al. (2016).

The experimental setup. The Feedback [19] mechanical unit 33-100, shown in Fig. 5, is a system consistent of a DC motor, an analog tachogenerator, analog input and output potentiometers, absolute and incremental digital encoders and a magnetic break. Although it has multiple outputs, we use it as a SISO system that amplifies the input power (provided by a Beckhoff EtherCAT analog-output terminal-card) to drive its DC motor and measure its angular position (with a Beckhoff EtherCAT analog-input terminal-card). Again, the system is controlled from a TwinCAT PLC programmed in ST.

The activities. The system allows to teach introductory linear systems and control topics such as system identification, and PID & state feedback regulation. Hence, we propose again activities related to system modelling, and control tuning & testing.

The experience-HMI webpage. For this RL, the instructors have developed the experience webpage shown in Fig. 7, which is similar, although simpler, to the one of the Hover RL. The main differences are that only 1 plot is needed to show the reference, control and system-output signals of this SISO system, and that the controller parametrization panel lets students select other regulators and parameters. Nevertheless, the similarities in the experience-HMI webpage of both RLs highlight an additional benefit of our RLMS: the `LabParameter` and `LabControl` widgets are useful to speed up the development and unify the HMI of RL for Control Education.

[18] In Fig. 6 only the camera, drive and pith graphics are enabled.

[19] `http://www.feedback-instruments.com`

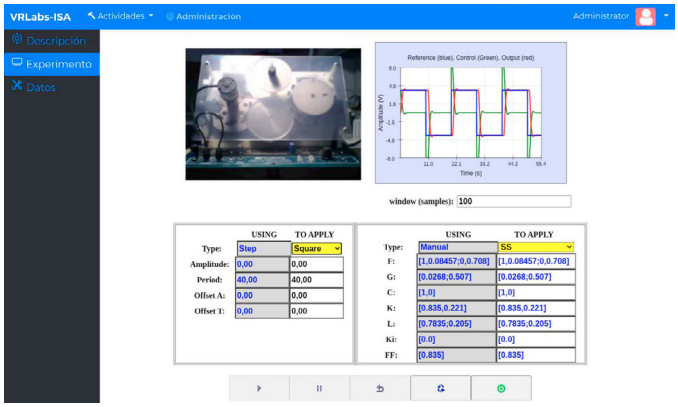

Fig. 7. Experience webpage of Feedback 33-100 unit RL.

5. CONCLUSIONS

This work presents a new RLMS to efficiently develop, configure and deploy remote laboratories for Control Education which close the feedback loop over the system under study using a TwinCAT PLC and which provide the students with HMI that are generated with EJsS. This last tool, enhanced with *ReNoLabs Plugin* and backed-up by the renewed *ReNoLabs* server, is also used to streamline and speed up the tasks that the operators perform to configure and deploy the RL. The paper also illustrate how to use our RLMS and presents the main features of two RLs already put into production with it. In spite of the current versatility of the approach, which can also be used with other types of back-ends, we are planning to improve it further. In this regard, we want to extend the capabilities related to the automatic generation of the interface and to provide support for collaborative learning activities, for saving the data in external storage API applications (e.g. Dropbox or Moodle LMS) and for gathering user interaction data.

REFERENCES

Bellmunt, O.G., Montesinos-Miracle, D., Galceran-Arellano, S., Sumper, A., and Sudrià-Andreu, A. (2006). A distance plc programming course employing a remote laboratory based on a flexible manufacturing cell. *IEEE Transactions on Education*, 49(2).

Bermudez-Ortega, J., Besada-Portas, E., Lopez-Orozco, J.A., Chacon, J., and de la Cruz, J. (2016). Developing web & TwinCAT PLC-based remote control laboratories for modern web-browsers or mobile devices. In *2016 IEEE Conference on Control Applications*.

Bermúdez-Ortega, J., Besada-Portas, E., López-Orozco, J.A., and de la Cruz, J.M. (2017). A new open-source and smart-device accessible remote control laboratory. In *4th International Conference Experiment@*.

Besada-Portas, E., Bermudez-Ortega, J., de la Torre, L., López-Orozco, J.A., and de la Cruz, J.M. (2016). Lightweight Node.js EJsS-based web server for remote control laboratories. In *11th IFAC Symposium on Advances in Control Education*.

Besada-Portas, E., Lopez-Orozco, J.A., Aranda, J., and de la Cruz, J.M. (2013a). Virtual and remote practices for learning control topics with a 3DOF quadrotor. In *10th IFAC Symposium Advances in Control Education*.

Besada-Portas, E., Lopez-Orozco, J.A., de la Torre, L., and de la Cruz, J.M. (2012). EasyJava Simulations meets TwinCAT: Remote real-time control experiments using programmable logic controllers. In *9th IFAC Symposium Advances in Control Education*.

Besada-Portas, E., Lopez-Orozco, J.A., de la Torre, L., and de la Cruz, J.M. (2013b). Remote control laboratory using EJS applets and TwinCAT programmable logic controllers. *IEEE Transactions on Education*, 36(2).

Chacón, J., Besada-Portas, E., Carazo-Barbero, G., and López-Orozco, J.A. (2021). Enhancing ejss with extension plugins. *Electronics*, 10(3).

de la Torre, L., Saenz, J., Chaos, D., Dormido, S., and Sanchez, J. (2020). A master course on automatic control with remote labs. In *21st IFAC World Congress*.

de la Torre, L., Sanchez, J., and Dormido, S. (2016). What remote labs can do for you. *Physics Today*, 69.

Faulconer, E. and Gruss, A. (2018). A review to weigh the pros and cons of online, remote, and distance science laboratory experiences. *The International Review of Research in Open and Distributed Learning*, 19(2).

Gomes, L. (2009). Current trends in remote laboratories. *IEEE Transactions on Industrial Electronics*, 56.

González, I., Calderón, A.J., Mejias, A., and Andujar, J.M. (2016). Novel networked remote laboratory architecture for open connectivity based on plc-opc-labview-ejs integration. application to remote fuzzy control and sensors data acquisition. *Sensors*, 16.

Heradio, R., de la Torre, L., Galan, D., Cabrerizo, F.J., Herrera-Viedma, E., and Dormido, S. (2016). Virtual and remote labs in education: A bibliometric analysis. *Computers Education*, 98.

Kalúz, M., García-Zubía, J., Fikar, M., and Cirka, L. (2015). A flexible and configurable architecture for automatic control remote laboratories. *IEEE Transactions on Learning Technologies*, 18(3).

Klein, A. and Wozny, G. (2006). Web based remote experiments for chemical engineering education. the online distillation column. *Trans IChemE, Part D*, 1.

Kostaras, N., Xenos, M., and Skodras, A. (2011). Evaluating usability in a distance digital systems laboratory class. *IEEE Transactions on Education*, 54(2).

Leva, A., Cimino, C., and S.Seva (2020). A control education software suite to bridge methodological and engineering aspects. In *21st IFAC World Congress*.

Ma, J. and Nickerson, J. (2006). Hands-on, simulated, and remote laboratories: A comparative literature review. *ACM Computing Surveys*, 38(7).

Prada, M.A., Fuertes, J.J., Alonso, S., García, S., and Domínguez, M. (2015). Challenges and solutions in remote laboratories. application to a remote laboratory of an electro-pneumatic classification cell. *Computers Education*, 85, 180–190.

Sáenz, J., Chacón, J., De La Torre, L., Visioli, A., and Dormido, S. (2015). Open and low-cost virtual and remote labs on control engineering. *IEEE Access*, 3.

Sánchez-Herrera, R., Marquez, M.A., , and Andújar, J.M. (2020). Easy and secure handling of sensors and actuators as cloud-based service. *IEEE Access*, 8.

Zubía, J.G. and Alves, G.R. (2011). *Using Remote Labs in Education. Two Little Ducks in Remote Experimentation*. DeustoDigital.