2016 IEEE Conference on Control Applications (CCA)
Part of 2016 IEEE Multi-Conference on Systems and Control
September 19-22, 2016. Buenos Aires, Argentina

Developing web & TwinCAT PLC-based remote Control laboratories for modern web-browsers or mobile devices*

Julian Bermudez-Ortega[1], Eva Besada-Portas[1], Jose A. Lopez-Orozco[1], Jesus Chacon[2] and Jesus M. de la Cruz[1]

Abstract— **This paper describes a new approach to develop remote Control laboratories accessible from modern web-browsers and student devices (PCs, laptops, tablets and smart-phones) based on TwinCAT Programmable Controllers (PLCs), Easy JavaScript Simulations (EJsS) webpages, and a Node.js laboratory web-server. On the one hand, implementing the laboratory back-end application (responsible of closing the feed-back loop over the plant under study) using a TwinCAT PLC provides the laboratory controllers with standard/industrial automation methodologies, real-time support and connectivity to a wide range of input/output signals. On the other hand, defining the controller front-end (graphical/interactive interface used by the students to parametrize the PLC behavior and observe the evolution of the plant and PLC signals) with EJsS facilitates the organization of its visual/interactive elements and allows the generation of a JavaScript and HTML5 webpage that is accessible from modern web-browsers and various types of students devices running different operating systems. Last, but not least, developing the laboratory web-server (in charge of managing the student access to the lab and of hosting its different webpages, including the controller front-end) within the JavaScript development and runtime platform Node.js ensures a lightweight behavior of the remote lab, provides a robust connectivity to the students, and supports efficient (real-time) communication between the controller back & front-ends. This paper also shows how the new strategy is used to update and overcome the current accessibility limitations of some existing (and published) experiences with Proportional/Integral/Differential (PIDs) and Estimator and State Feedback (ESF) Controllers based on TwinCAT PLCs and Easy Java Simulations (EJS) applets front-ends.**

I. INTRODUCTION

Remote web-based laboratories are technological platforms that provide access to industrial, research or educational lab equipment through the internet, making it operable and observable by faraway users. Their development for educational purposes in the last 20 years provides multiple benefits ([1], [2], [3]). On the one hand, they let students 1) conduct, at any time and from anywhere, a bigger amount of experiments than during the traditional sessions in hands-on laboratories; and 2) interact with real equipment and observe their real noisy and rich data (which presents phenomena not always shown during simulations). On the other hand, they can provide safe access to expensive or dangerous equipment, as well as increment its usage and decrease its spoiling risk. These characteristics make them also ideal as educational tools for Control Engineering, as existing remote labs show (e.g. to control the water levels of multi-tank systems in [4] and [5]; the temperature of a heat exchanger in [6]; the position/velocity of servo/dc motors in [7] and [8]; or the attitude of a fix-located quadrotor in [9]).

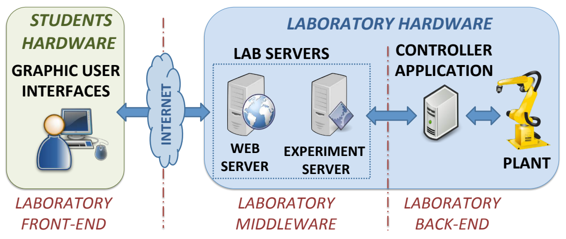


Fig. 1. Structure of a generic web-based laboratory. Its main elements are labelled in black, the layers used to distribute them are highlighted in red, and the exchange of data among them is represented by the blue arrows.

The functionality of these laboratories is often distributed over the different elements/layers that appear in the structure displayed in Fig. 1. At the laboratory back-end, the controller application is responsible of closing the feedback loop over the plant under study: it measures the plant outputs, calculates the outputs of a parametrizable controller, and applies them to the plant. At the front-end, the Graphic User Interfaces (GUIs) let the students 1) log-in the lab, 2) access information about the experiences, and 3) interact with the lab by parametrizing the behavior of the back-end controller, and by observing the evolution of its signals and of real-time images of the plant. Finally, the servers in the middleware are in charge of handling the students access to the lab and the data flow between the other two layers.

The software tools that support each layer often differ from one lab to other and many times show the software preferences of its developers. The variability that appears in the controller application[1] is closely related to the software availability in the lab. The diversity that appears in the GUIs[2]

*This work was supported by the Spanish National Research Project DPI2013-46665-C2-R1.

[1]Julian Bermudez-Ortega, Eva Besada-Portas, Jose A. Lopez-Orozco and Jesus M. de la Cruz are with the Department of Computer Architecture and Systems Engineering of Universidad Complutense de Madrid, Madrid, Spain {juliberm,ebesada,jalo,jmcruz}@ucm.es

[2]Jesus Chacon is with the Computer Science and Automation Department, Universidad Nacional de Educacion a Distancia, Madrid, Spain, jchacon@bec.uned.es

[1]Examples range from robust automation/industrial software tools (e.g. LabVIEW in [4] and [10], or TwinCAT in [8]) to generic programming languages and compilers (e.g. C in [11] or Python in [12]).

[2]Examples vary from Java applets (e.g. those developed with Easy Java Simulations in [4], [8] and [10]) to AJAX GUIs in [13] or mashups in [14].

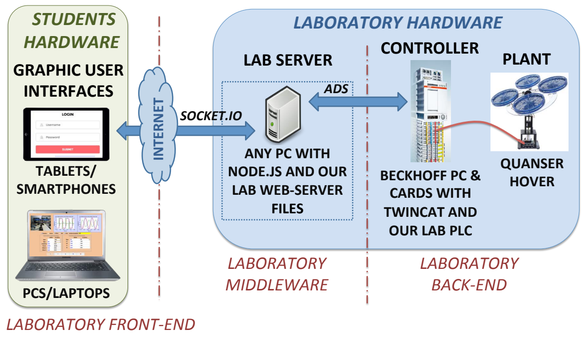


Fig. 2. Schema of the new web-based laboratory. The labelled elements are explained through the paper. The student tablet shows the login webpage while the student PC shows the webpage of the controller for Quanser Hover.

and in the servers[3] is driven by the continuous evolution of the internet, a fact that simultaneously 1) brings challenges to existing labs (e.g. a sudden lack of support of the GUIs) and 2) opens new possibilities of development (e.g. software tools) and of access (e.g. from mobile devices). Both matters (challenges and possibilities) are the origin of the new approach presented in this paper and schematized in Fig. 2 to develop remote web-based laboratories whose back-end, front-end and middleware are respectively based on Twin-CAT[4] Programmable Logic Controllers (PLCs), HTLM5 and JavaScript webpages developed with Easy JavaScript Simulations (EJsS[5]), and a Node.js[6] lab server.

Our lab uses PLCs to exploit the versatility they bring to a big variety of industrial applications[7], and adopts TwinCAT to develop/deploy/run the PLCs and to support the access to their variables from an external application (the lab web-server). It employs HTLM5 + JavaScript webpages to make the lab GUIs accessible from modern web-browsers of computers and mobile devices, and exploits EJsS to facilitate/accelerate its creation. Finally, the JavaScript-based development and runtime environment Node.js makes our lab-server a robust lightweight asynchronous event-driven application, capable of hosting the lab GUIs and of efficiently exchanging data between them and the controller. In short, all these tools allow us to create a versatile robust web-based lab that allows students a friendly real-time remote access to our PLCs from their home computers and mobile devices.

It is worth noting that the supporting tools of our lab have already been used in other remote web-based laboratories. However, the approach in [8] and [9], consistent on combining TwinCAT PLCs with Easy Java Simulations (EJS) applets is becoming obsolete as the last are not accessible from new versions of Google Chrome[8] or from the web browsers of iOs/Android/WindowsMobile-based mobile devices. The same problem makes [18] develop a methodology to connect an accessible EJsS JavaScript and HTML5 front-end to an out-of-date EJS applet (e.g. the front-end of an existing remote laboratory) and [11] use Node.js to communicate an EJsS front-end with a C-programmed controller. Besides, the labs in [14] and [19] only use Node.js for setting up the data communication between their GUIs (a mashup and EJS applet) and laboratory back-ends (LabVIEW controller and general purpose input/output pins); while [20] introduces the idea of using Node.js to support the whole webpage of a remote lab connected to either a C-programmed, LabVIEW or TwinCAT controller. So, this work extends: [11], [14] and [19] by using Node.js to support all the webpages of the lab; [20] by detailing the set up of new laboratories based on TwinCAT PLCs, a Node.js web server and EJsS GUIs; and [8] and [9] by upgrading the front-end and middleware layers of their labs with up-to-date internet technologies.

II. SOFTWARE TOOLS

This section explains the main features and functionalities of the software used to support our new lab, highlighting those that are especially useful for their development.

A. TwinCAT PLCs and Runtime for the Back-end Layer

TwinCAT is a powerful and widely used tool to develop different types of controllers (including PLCs) for the industry, run them on Windows PCs connected to the plants through different buses (e.g. EtherCAT and CAN) and terminal/cards (e.g analog to/from digital converters and incremental encoders), and observe/modify the values of their variables during its execution. To that end, it converts the deploying PC into a real time system that runs the controllers; offers visual tools to graphically represent online the evolution of some of their variables; and provides data communication mechanisms and libraries to be able to exchange data between the controllers and other applications.

All these properties are supported by several complex tools, welcome in the industry for developing/performing a wide spectrum of tasks, but overwhelming for students of the introductory subjects of Systems Engineering and Automation.

Therefore, we use TwinCAT[9] to program ourselves, as lab instructors, the functionality of the lab back-end PLCs; to support its robust and real-time execution; and to permit the access of the front-end GUIs (through the lab web-server) of those variables that the students have to observe/modify to analyze/parametrize the behavior of the plant and controller.

B. EJsS for the HTML5 + JavaScript Front-end Layer

Easy JavaScript Simulations is a freeware open-source graphical tool that helps users with low programming skills to create discrete computer simulations runnable as apps in mobile devices or as webpages in web-browsers. For the

[3]Examples range from basic ad-hoc applications to communicate the terminal layers (e.g a Java program in [8]) to multiple tools for hosting the laboratory webpage, connecting the GUI and the controller, and handling collaborative/exclusive access to the laboratory resources (e.g. the SARLAB server and Moodle learning platform in [15] and [16]).

[4]TwinCAT: http://www.beckhoff.es/english.asp?twincat/default.htm

[5]EJsS: http://www.um.es/fem/EjsWiki/pmwiki.php

[6]Node.js: https://nodejs.org/en/

[7]Examples such as water management, manufacturing, and plants control and monitoring are described in [17].

[8]Java applet support was interrupted in September 2015.

[9]In particular, we are using TwinCAT 2.11.

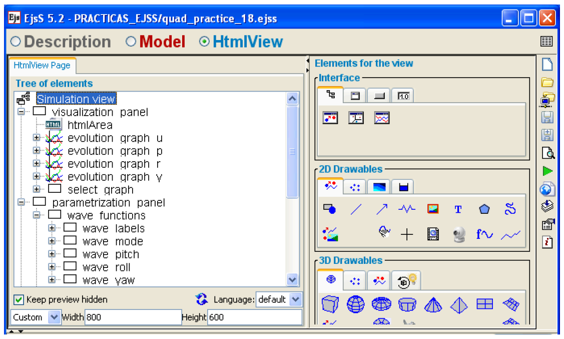


Fig. 3. EJsS user interface, displaying the HtmlView information of the Quanser Hover (3 DOF Quadrotor) experiment.

last purpose, it creates a HTML5 + JavaScript webpage that 1) is generally accessible from modern web-browsers of computers and mobile devices, and that 2) encodes the GUI and simulation behavior defined within EJsS.

We only[10] exploit the displaying and interactive elements of the HtmlView panel of EJsS (see Fig. 3) to define the front-end GUIs of our lab controllers, and the programming possibilities within its Model panel to encode the functionality associated to the interactions with the GUI[11] and to the asynchronous reception of data from the server[12].

Finally, it is worth noting that EJsS is the natural substitution of its predecessor EJS (which was previously used in [8] and [9] as the lab front-end of TwinCAT PLCs), as both follow the same programming Model-View-Controller paradigm [21]. Besides EJsS creates HTML5 + JavaScript webpages, generally reachable from modern web-browsers; while EJS created Java applets, unsupported by some present web-browsers, including those used in mobile devices.

Therefore, EJsS[13] facilitates the creation of the GUIs of the experiments and the students access from all their devices.

C. Node.js for the Middleware Layer

Node.js is a multi-platform open-source JavaScript-based development and runtime environment, built on an event-driven non-blocking I/O architecture, to implement scalable and data-intensive real-time network applications.

We develop a Node.js web-server for our lab to *exploit the computational efficiency* delivered by its underlying Google V8 JavaScript engine[14], *the robustness* tested by its big community of users[15], the general *web-browser accessibility* of its JavaScript-encoded webpages, and the added functionality provided by *the extension modules* by other users.

Among the extension modules employed by our server, it is worth highlighting: *Express* and *Forever* for managing the webpages; *Express-session*, *Cookie-parser* and *Body-parser* for coping with the simultaneous sessions of different students; *Passport*, *Passport-local* and *connect-ensure-login* for certifying only the access of registered students; and *Fs* and *Dateformat* for storing, in different text files, information about the students experiments. Besides, our server also uses *Socket.io* and *ADS*, the first for supporting the communications between the web-server and the GUIs[16] and the second for exchanging data between the web-server and the PLC[17].

In short, Node.js[18] and the selected extension modules make our lab web-server a robust lightweight application that handles the access of registered students[19] to the lab and efficiently exchanges data between its terminal layers.

III. LABORATORY SETUP OVERVIEW

This section introduces the steps required to set up our new web & TwinCAT PLC-based remote Control lab. Besides, it highlights the requirements that we have included in its different layers in order to standardize their behavior and communications.

A. Back-end Layer: TwinCAT PLCs

The high-level behavior of the PLCs at the back-end of our lab is implemented as a state-machine whose transitions are governed by the value of a variable which is modified by the lab web-server as a consequence of the student interactions with the elements at the bottom of the controller GUI. Besides, the configuration of the reference signals and of the specific controllers implemented in each PLC is defined in several variables that the server modifies after the students 1) input their values in the controller GUI and 2) request an update of the controller back-end. Finally, the values of the selected signals of the PLC are internally stored in an evolution array-variable that is periodically accessed from the server to send its information, with the data of the configuration of the reference signals and of the controller, to the front-end GUI of the controller.

[10]We do not require its simulation capabilities, because our lab focuses on remote experiences (connected to real hardware). However, they could be used in our lab to incorporate virtual practices (simulations) if desired.

[11]An example is the submission to the server of the reference signals selected by the students.

[12]An example is the evolution data that the lab server sends to the controller GUI, after reading it periodically from the back-end PLC.

[13]In practice, we are using EJsS 5.2.

[14]This engine makes Node.js compile the JavaScript code of our web-server into machine code and optimize its execution.

[15]They include firms such as LinkedIn, eBay or Paypal.

[16]In particular, Socket.io supports, when it is possible in both sides, a full-duplex Websocket communication based of JSON messages between our lab web-server and the controller GUI.

[17]The *ADS* extension module (https://www.npmjs.com/package/ads) permits our lab web-server to use the TwinCAT Automation Specification System (ADS) to access the contents of the variables of a PLC. In practice, it is used by our server to establish the ADS connection with the PLC, read/write the contents of the desired PLC variables, and set up a TwinCAT cyclic notification to awaken the lab web-server periodically in order to make it read the evolution data stored in some of the variables of the PLC. In spite of the versatility of the *ADS* extension module, we have to extend its functionality for our lab to let it support an efficient exchange of *vectorial/matrix* information, a quality which is especially useful to represent the polynomial or matrixes used to define some types of controllers. Finally, it is worth noting that we have preferred adapting *ADS* for our purposes than using *Tame4*, a JavaScript library to access TwinCAT PLCs that already supports the exchange of vectorial and matrix information, because *Tame4* requires to set up an additional software layer (a TwinCAT ADS WebService) that is not required by *ADS*. A similar reason, as well as the requirement of using TwinCAT 3.0, has also made us turn down TwinCAT TcAdsWebService.js JavaScript library.

[18]In particular, we are using Node.js 4.2.4.

[19]At present, our lab-server ensures the exclusive access of a single student to the controller webpage and the access of multiple students to the others (login, menu, help and data webpages). Besides, all have to be pre-registered by the lab instructors within a database of the lab server.

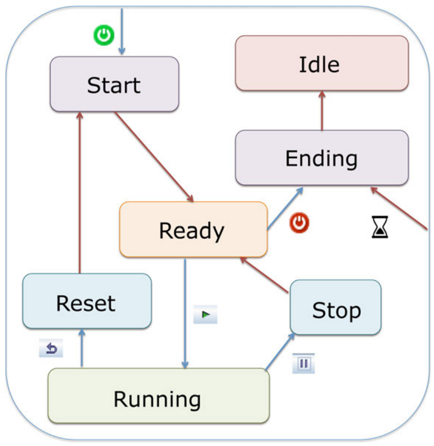


Fig. 4. High-level state machine of our PLCs

In order to develop the PLC of a new experiment, the lab instructors can adapt any of the existing ones[20], because they already implement the required state machine and transitions, which are represented in the schema shown in Fig. 4. Therefore, they have to distribute the functionality required in the PLC among the different states following the information presented in Table I and create/use the variables required to exchange data with the GUI in order to parametrize the reference signals and controllers, and access the evolution of the selected signals. Besides, they also have to define and use the input/output variables that the PLC uses to interact with the plant and configure its relationship with the input/output cards of the PC where the PLC is deployed and run.

B. *Front-end Layer: EJsS GUIs*

The GUIs of the controllers (see Fig. 5 and 6) share a common structure, displaying: 1) at the top, a real-time image of the plant and several graphics with the evolution of some signals of the PLC; 2) in the middle, the parametrization data of the reference signals (left) and of the controllers (right); and 3) at the bottom, the control buttons used to indirectly manipulate the state machine of the back-end PLC.

They also implement the generic high-level behavior that already communicates with the server when the control buttons at their bottom are used to perform the actions summarized in Table II. Besides, they already 1) incorporate the code required to send the parametrization information provided by the student to the server, and 2) set up the event that automatically receives from the server any message with data (of the evolution or parametrization values of the variables in the PLC) and transfer its information to the variables used by the GUIs for displaying purposes (e.g. in graphics and in parametrization fields[21]).

In order to develop the GUI of a new experiment, the lab instructors can again adapt any of the existing ones. In this

TABLE I
INTENDED FUNCTIONALITY WITHIN EACH STATE OF THE PLC

State	**Intended Operations**
Start	Functionality required to set up the controllers
Ready	Code that prepares the controller to start the experiment
Running	Usual steps of a feedback loop (measure plant outputs, calculate control signals & apply them to the plant)
Stop	Code to obtain the control signal to apply while plant is halted
Reset	Any operation required from the *Running* state before re-starting the experiment
Ending	Steps to be performed at the end of the experiment

TABLE II
INTENDED FUNCTIONALITY OF THE GUI CONTROL BUTTONS

Buttons	**Intended Operations**
▶	Run the feedback loop
❚❚	Stop the feedback loop
↺	Reset the PLC
🔄	Update the parametrization variables of the PLC with the values in the GUI (TO APPLY column in Fig. 5 and 6)
⏻ / ⏻	Connect/disconnect the server from the PLC

case, they should first focus on defining the variables used by the GUI to store 1) the evolution and parametrization information which is provided by the web-server after reading their values in the PLC, and 2) the parametrization information manipulable by the students in the GUIs[22]. Afterwards, they should define the graphics and fields used to display the information and link them with the corresponding variables. Finally, they have to wrap (build) the EJsS project, unzip it, extract the files named **_javascript_simulation.xhtml* and *ejsS.v1.min.js*, slightly modify the first[23], and deploy both of them within the lab web-server folders.

C. *Middleware Layer: A Node.js web-server*

Not only is our lab web-server responsible of displaying the controller webpage and of supporting the communication between it and the PLC, but it is also in charge of managing: 1) the login webpage that restricts the access to the lab to registered students; 2) the menu webpage that permits the students access the controller, help and data webpages; 3) the help webpage that provides information about the experiments in the lab; and 4) the data webpage that lets the students access the data files that are automatically stored during their own experiments with the controllers.

All this functionality is programmed in a JavaScript file that specifies, according with the event-driven Node.js philosophy and the approach introduced in [20], which actions to perform when a student: 1) writes the lab connection

[20]They will be available after the conference, as the rest of the code, at http://www.dacya.ucm.es/isalab/index.html

[21]We use two types of fields to display/modify the parametrization information: a numeric box for single-variable parameters and a text box for vector/matrix parameters. The last should be written column-wise, separating each column with a comma and each row with a semi-colon.

[22]Note that our GUIs use two different variables for each set of parameters: the first to store the information used by the PLC and the second to let the students manipulate their values. Therefore, it is capable of simultaneously displaying both types of data: the first in the non-editable fields under the *USING* columns of the GUIs and the second in the editable fields under the *TO APPLY* tabs.

[23]We automatically uncomment a few lines related with the communication functionality, also supported by Socket.io, of our GUIs, and make its background and header equal to the one used in all the lab webpages.

address in a web-browser, 2) provides his/her login information, 3) selects an option in the menu webpage, and 4) chooses a file from the student data webpage. Besides, this file also specifies 5) how to redirect the data that the GUI wants the server to transmit to the PLC (after the student interactions with the controller GUI control buttons) and 6) what information to read from the PLC and to transmit to the GUI after a periodical awakening.

In order to develop the Node.js web-server of a new experience, the lab instructors can again adapt any of the existing ones. Moreover, they can maintain the functionality associated to the first 4 actions enumerated in the last paragraph and focus on modifying the code associated to the remaining two actions (i.e. the ones related with the exchange of data between the GUI and PLC). On the one hand, they should 1) replicate in the JavaScript file of the lab web-server the variables used within EJsS to send information of the PLC to the server and 2) define the *ADS* variables that the server will use to access the variables of the PLC that store its evolution and parametrization information. On the other, they have to transfer the information among the variables used to receive/access the information of any of the two terminal layers and re-direct it to the opposite one.

IV. APPLICATION EXAMPLES

In this section, we shortly describe the first two examples that have been adapted to this new approach and that are currently tested in the laboratories of the Control subjects of the University Complutense of Madrid.

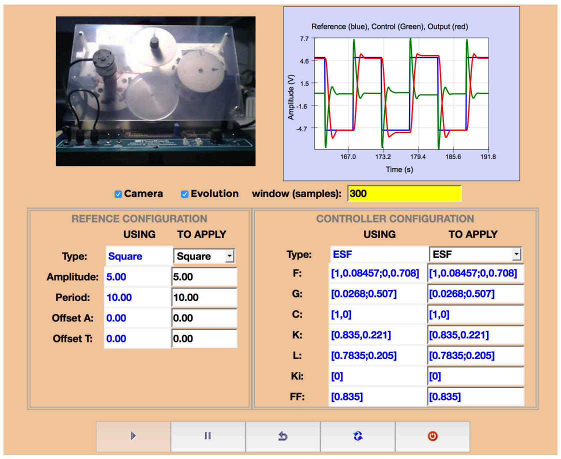
A. *Controlling a Feedback 33-100 DC Motor*

The Feedback[24] mechanical unit 33-100 is an electromechanical system that consists of a DC motor, an analog tacho-generator, analog input and output potentiometers, absolute and incremental digital encoders and a magnetic break. It is a Single Input Multiple Output (SIMO) system that amplifies the input power to drive its DC motor and that provides information of its rotation speed and angular position.

It is a well-tested and robust platform ideal for introductory topics in Control, where the students can test and tune different types of linear controllers. For that reason, we already used it in the remote laboratory presented in [8] to make our students understand different tuning strategies for: Estimator and State Feedback (ESF) controllers with feed-forward and integral action; a 2-Degrees-Of-Freedom (2-DOF) controller; and a Proportional/Integral/Differential (PID) controller.

However, the new trends of web-browsers, which are stopping to support the execution of Java applets within a webpage, decrement the chances of accessing this platform from the student PCs. Besides, they are not either available from their mobile devices.

Therefore, we have decided to adapt this experience to our new approach, maintaining the PLC that implements the functionality of the 3 controllers under test (ESF, 2-DOF and PID) and reimplementing within EJsS a GUI (see



☑ Camera ☑ Evolution window (samples): 300

REFENCE CONFIGURATION	USING	TO APPLY
Type:	Square	Square
Amplitude:	5.00	5.00
Period:	10.00	10.00
Offset A:	0.00	0.00
Offset T:	0.00	0.00

CONTROLLER CONFIGURATION	USING	TO APPLY
Type:	ESF	ESF
F:	[1,0.08457;0,0.708]	[1,0.08457;0,0.708]
G:	[0.0268;0.507]	[0.0268;0.507]
C:	[1,0]	[1,0]
K:	[0.835,0.221]	[0.835,0.221]
L:	[0.7835;0.205]	[0.7835;0.205]
Ki:	[0]	[0]
FF:	[0.835]	[0.835]

Fig. 5. Front-end of the controller of the FeedBack 33-100 DC Motor lab. In the captured image, the students have chosen to apply a square reference signal and a ESF controller, and have selected the numeric parameters of the first and the matrix parameters of the second. The elements at the top show the actual image of the plant and the evolution of the reference (blue), control signal (green) and angular position of the motor (red).

Fig. 5) that permits the students to 1) select the reference (of the angular position) of the motor; 2) parametrize the behavior of the 3 controllers; and 3) observe the evolution of the actual plant through the images provided in real time by a camera and of the reference, the control signal and angular position of the motor. Besides, we have adapted the Node.js server introduced in [20] to make its help webpage provide information about these experiences and to connect the existing PLC with the new controller GUI HTML5+EJS webpage. It is worth noting, that this lab has to use several controllers with matrix and vectorial parameterizations.

B. *Controlling a Quanser Hover (3DOF QuadRotor)*

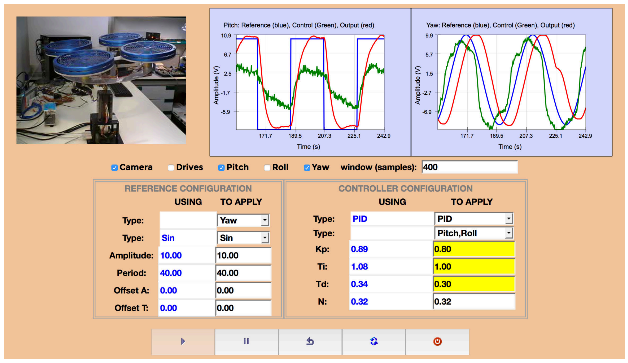
The 3DOF Hover by Quanser[25] is a Multiple Input Multiple Output (MIMO) system designed to study the behavior of a fixed-located quadrotor using a platform consistent of a frame with 4 propellers mounted on a 3 DOF (Degrees Of Freedom) pivot joint. Each propeller is driven by a DC motor and the joint permits the frame to roll, pitch and yaw. These angular displacements can be controlled by the lift force and the torque generated by the propellers. Information about the frame orientation is provided by the optical encoders mounted over each axis and each DC motor is independently controlled using an analog signal.

The current interest on these types of systems makes them an attractive choice to learn control topics. However, its high price reduces the number of plants of this type that a laboratory can have and makes it an ideal candidate to develop remote control experiences. Both reasons drove us to develop the controllers and experiences presented in [9], which are no longer accessible for all our students due to the current lack of Java applet support of several web-browsers.

Therefore, we have decided to maintain the functionality of its PLC, and reimplement within EJsS a GUI (see Fig. 6) that permits the students to 1) select a different angular

[24]http://www.feedback-instruments.com

[25]http://www.quanser.com/products/3dof_hover


Pitch: Reference (blue), Control (Green), Output (red)
Yaw: Reference (blue), Control (Green), Output (red)
Amplitude (V)
Time (s)
10.9
6.7
2.5
-1.7
-5.9
9.9
5.7
1.5
-2.7
-6.9
171.7
189.5
207.3
225.1
242.9

☑ Camera ☐ Drives ☑ Pitch ☐ Roll ☑ Yaw window (samples): 400

REFERENCE CONFIGURATION

	USING	TO APPLY
Type:		Yaw
Type:	Sin	Sin
Amplitude:	10.00	10.00
Period:	40.00	40.00
Offset A:	0.00	0.00
Offset T:	0.00	0.00

CONTROLLER CONFIGURATION

	USING	TO APPLY
Type:	PID	PID
Type:		Pitch,Roll
Kp:	0.89	0.80
Ti:	1.08	1.00
Td:	0.34	0.30
N:	0.32	0.32

Fig. 6. Front-end of the Quanser Hover lab. In the captured image, the students have chosen to apply a square reference signal to the pitch and a sinusoidal curve to the yaw, a PID controller for each of the angular displacements, and the numeric parameters of all of them. The elements at the top show the actual image of the plant and the evolution curves of the pitch and yaw references/control signals/outputs.

reference for each angular displacement; 2) parametrize a different PID for the yaw, the pitch and the roll, or a feedback linearization or sliding control for the whole system; and 3) observe the evolution of the actual quadrotor, of the 4 motor signals, and of each angular reference, control and output signal. Besides, we have adapted the Node.js web-server developed for the Feedback 33-100 DC motor lab to make its help webpage provide information about the Quanser Hover experiments and to connect the existing PLC with the new controller GUI HTML5+EJS webpage.

V. Conclusions

This paper presents a new approach to develop remote laboratories that uses TwinCAT PLCs for implementing the feedback loop over the plant, EJsS for defining the GUIs of the controller, and Node.js to set up the lab web-server. The selected elements create a robust lightweight Node.js web-server that manages the access to the webpages of the lab and that exchanges information between the controller GUI (a HTLM5+JavaScript webpage developed in EJsS) and the TwinCAT PLC. Besides, our new labs fulfill the requirements of modern browsers, letting the students access the lab from their home PCs and mobile devices. Finally, the paper also shows the process of setting up the labs and describes two previously existing labs that have been adapted to take advantage of the benefits provided by our new approach.

We are exploring the possibility of automating the tasks required to setup a new lab (e.g. by automatically creating in the lab web-server the variables required to communicate the controller GUI with the PLC) and of improving the lab web-server functionality (e.g. by letting students enroll in the lab, by providing support for collaborative access to the PLCs, or by considering the suggestions of the students that are using the labs during this academic year in some Control subjects of the Universidad Complutense of Madrid).

Acknowledgment

The authors thank D. Sanchez Foces and F. Marquez Vidal for their support in setting up the experiments.

References

[1] B. Alhalabi, D. Marcovitz, K. Hamza, and S. Hsu, "Remote labs: an innovative leap in the world of distance education," in *Proc. of the 5th IFAC Symp. on Advances in Control Education*, 2000.
[2] C. Salzmann and D. Gillet, "Challenges in remote laboratory sustainability," in *International Conference on Engineering Education*, Coimbra, Portugal, 2007.
[3] L. de la Torre, J. P. Sanchez, and S. Dormido, "What remote labs can do for you," *Physics Today*, vol. 69, no. 4, pp. 1185 – 1205, April 2016.
[4] R. Dormido, H. Vargas, N. Duro, J. Sanchez, , S. Dormido-Canto, G. Farias, F. Esquembre, and R. Dormido, "Development of a web-based control laboratory for automation technicians: The three-tank system," *IEEE Trans. on Education*, vol. 51, pp. 35–44, 2008.
[5] M. Stefanovic, V. Cvijetkovic, M. Matijevic, and V. Simic, "A LabVIEW-based remote laboratory experiments for control engineering education," *Computer Applications in Engineering Education*, vol. 19, pp. 538–549, 2011.
[6] C. Lazar and S. Carari, "A remote-control engineering laboratory," *IEEE Trans. on Industrial Electronics*, vol. 55, pp. 2368–2375, 2008.
[7] H. Vargas, J. Sanchez, N. Duro, R. Dormido, S. Dormido-Canto, G. Farias, and S. Dormido, "A systematic two-layer approach to develop web-based experimentation environments for control engineering education," *Intelligent Automation and Soft Computing*, vol. 14, pp. 505–524, 2008.
[8] E. Besada-Portas, J. Lopez-Orozco, L. de la Torre, and J. de la Cruz, "Remote control laboratory using EJS applets and TwinCAT programmable logic controllers," *IEEE Trans. on Education*, vol. 56, no. 2, pp. 156–164, 2013.
[9] E. Besada-Portas, J. A. Lopez-Orozco, J. Aranda, and J. M. de la Cruz, "Virtual and remote practices for leaning control topics with a 3DOF quadrotor," in *10th IFAC Symp. on Advances in Control Education*, 2013.
[10] J. Chacon, H. Vargas, G. Farias, J. Sanchez, and S. Dormido, "EJS, JiL and LabVIEW: How to build a remote lab in the blink of an eye," *IEEE Trans. on Learning Technologies*, vol. 8, no. 4, pp. 393–401, Oct 2015.
[11] J. Bermudez-Ortega, E. Besada-Portas, J. A. Lopez-Orozco, J. A. Bonache, and J. M. de la Cruz, "Remote web-based control laboratory for mobile devices based on EJsS, Raspberry Pi and Node.js," in *IFAC Workshop on Internet Based Control Education*, 2015.
[12] A. Hoyo, J. L. Guzman, J. C. Moreno, and M. Berenguel, "Teaching control engineering concepts using open source tools on a Raspberry Pi board," in *IFAC Workshop on Internet Based Control Education*, 2015.
[13] S. Dutta, S. Prakash, and D. Estrada, "A web service and interface for remote electronic device characterization," *IEEE Trans. Educ*, vol. 54, no. 4, pp. 646–651, Nov 2011.
[14] X. Chen, D. Osakue, N. Wang, H. Parsaei, and G. Song, "Development of a remote experiment under a unified remote laboratory framework," in *2013 World Congress on Engineering Education*, 2013.
[15] L. de la Torre, R. Heradio, C. A. Jara, J. Sanchez, S. Dormido, F. Torres, and F. A. Candelas, "Providing collaborative support to virtual and remote laboratories," *IEEE Trans. on Learning Technologies*, vol. 6, no. 4, 2013.
[16] 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 Trans. on Industrial Informatics*, vol. 11, no. 4, pp. 934–945, Aug 2015.
[17] E. R. Alphonsus and M. O. Abdullah, "A review on the applications of programmable logic controllers (PLCs)," *Renewable and Sustainable Energy Reviews*, vol. 60, pp. 1185 – 1205, 2016.
[18] J. Saenz, F. Esquembre, F. J. Garcia, L. de la Torre, and S. Dormido, "An architecture to use easy java-javascript simulations in new devices," in *IFAC Workshop on Internet Based Control Education*, 2015.
[19] J. Saenz, J. Chacon, L. De La Torre, A. Visioli, and S. Dormido, "Open and low-cost virtual and remote labs on control engineering," *Access, IEEE*, vol. 3, pp. 805–814, 2015.
[20] E. Besada-Portas, J. Bermudez-Ortega, L. de la Torre, J. A. Lopez-Orozco, and J. M. de la Cruz, "Lightweight Node.js & EJsS-based web server for remote control laboratories," in *11th IFAC Symp. on Advances in Control Education*, 2016.
[21] 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.