Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 13474–13479

Conducting Online Lab Experiments with Blockly

Daniel Galan * **Ruben Heradio** ** **Luis de la Torre** *
Sebastian Dormido * **Francisco Esquembre** ***

* *Dept. of Computer Science and Automatic Control, Universidad Nacional de Educación a Distancia, Juan del Rosal 16, E–28040, Madrid, Spain (e-mail: dgalan@dia.uned.es, ldelatorre@dia.uned.es, sdormido@dia.uned.es).*
** *Dept. of Software Engineering and Computer Systems, Universidad Nacional de Educación a Distancia, Juan del Rosal 16, E–28040, Madrid, Spain (e-mail: rheradio@issi.uned.es).*
*** *Dept. of Mathematics, Universidad de Murcia, Campus de Espinardo, E–30071, Murcia, Spain (e-mail: fem@um.es).*

Abstract: Laboratory experimentation plays an essential role in control education. To reduce the high costs of maintaining apparatus in traditional labs and to support distance and blended learning, online laboratories are used as a possible alternative to conventional hands-on labs. In these labs it is often desirable to allow students to define their own experiments. This paper presents the definition and implementation of a generic experimentation language for conducting automatic experiments on existing online laboratories. The main objective is to use an online lab, created independently, as a component in which users can perform experiments. To achieve it, authors present the Experiment Application. It is composed by Blockly, to define and design the experiments, and Google Chart, review and visualize the experiment results. This tool offers benefits to students, teachers and, even, lab designers. For the moment, it can be used with any existing lab or simulation created with the authoring tool Easy Java(script) Simulations. Since there are repositories with hundreds of free available labs created with this tool, the potential applicability of the described tool is considerable. To illustrate the utility of the Experiment Application a very well-known system is used: the Simple Harmonic Oscillator system.

Keywords: Experimentation Language; Experiments; Virtual Laboratories; Remote Laboratories; Easy java(script) Simulations; JavaScript; Blockly.

asa

1. INTRODUCTION

The use of virtual and remote laboratories (VRLs) for control engineering has increased significantly given the rise of distance, online and blended learning. The benefits of VRLs are well known, including: (1) cost savings in equipment, space and maintenance staff, (2) possibility to study phenomena which would not be possible to investigate in a traditional hands-on laboratory, and (3) to make a great number of simulations without any restriction, Heradio et al. (2016).

A laboratory is meant to offer experimentation possibilities. Experimentation can be defined as the process of extracting data from a system by exerting it, not only through its inputs, but also through the model parameters. Traditionally, users of VRLs were expected to perform experiments by interacting with the applications' graphical user interface (GUI). For this reason, visualization and interactivity are features of special importance for laboratories used to teach control engineering, Heck (1999); Esquembre (2004). The greatest proof of this statement is the existence of hundreds of simulations used for pedagogical purposes on the internet, specially, in the field of control engineering.

The use of images or animations is highly recommended in order to help users to understand more easily the system under study. Current developments in interactivity allows users to visualize the response of the system to any external or internal change, Dormido et al. (2005); Sánchez et al. (2002). These features, rich visual contents and the possibility of an instantaneous visualization of the system response make VRLs a human-friendly tool to learn, helping users to achieve practical experience into engineering control systems.

Despite all these improvements, there are certain limitations that have to be solved. Certain actions performed during experimentation activities need to be accurate and time controlled, something nearly impossible by just interacting with the GUI (for example, pausing the evolution of the lab at the exact moment an event occurs, changing a static value of the model, compute the elapsed time between two actions). Some other actions add no educational value to the students and take considerable time, such as repeating a process tens of times with different parameter values to evaluate the obtained results. In contrast, it

10.1016/j.ifacol.2017.08.2323

would be preferable to code the experiment by using a flexible, intuitive and user-friendly experimentation language to automatically run it. This way, the VRL is considered a complete system and all variables, including the execution of the experiment, are controllable and observable.

Authors' main goal is to enrich existing VRLs with an application to create and perform automated experiments. Authors' proposal aims to provide an online application with which users are able to define and perform their own experiments with any online VRL. In order to achieve this objective, a new Application Programming Interface (API), a set of functions which VRLs should conform to in order to provide the desired experimentation capabilities, has been designed.

Experiments are scripts coded with Blockly, Marron et al. (2012), an easy and intuitive graphical programming language, which is independent of the one in which the VRLs are coded. Some of the existing modeling environments (ACSL, EcosimPro, Dymola) include certain scripting facilities that enable the user to run experiments, Elmqvist et al. (1998). As an example, Dymola's manual states that "...there is a script facility that makes it possible to load model libraries, set parameters, set start values, simulate, and plot variables by executing scripts". Based on the study of these modeling environments, authors have added new capabilities to the general general specifications obtained from the study of these environments to achieve a more global fulfill specification that provides flexible features.

To test the viability of the proposed experimentation application, authors' implementation uses JavaScript labs developed with the modeling tool Easy Java(script) Simulations (EjsS), designed to make the creation of computer simulations easy and accessible to teachers. EjsS allows saving time when creating VRLs because teachers do not need to create them from scratch. Another important feature about EjsS is that it is free. EjsS is part of the Open Source Physics (OSP), which provides free online resource collections through the ComPADRE digital library, supporting students and teachers. Among these resources, users can find more than 500 applications created with EjsS.

Despite the huge advantages and great utilities offered by the two previous tools and resources (EjsS and the OSP digital library, respectively), there was not a way to use them to allow the creation of simulation experiments by users. This limitation is not restricted to EjsS and the OSP digital library. PhET simulations, Wieman et al. (2008), also available to download for free, present the same problem too.

The paper is organized as follows. Section II presents the Experiment Application and its benefits. Section III discusses the implementation of the language and the blocks needed to represent experiments. Section IV shows the experimentation language in practice. Finally, the conclusion and some pointers to further works are described in Section V.

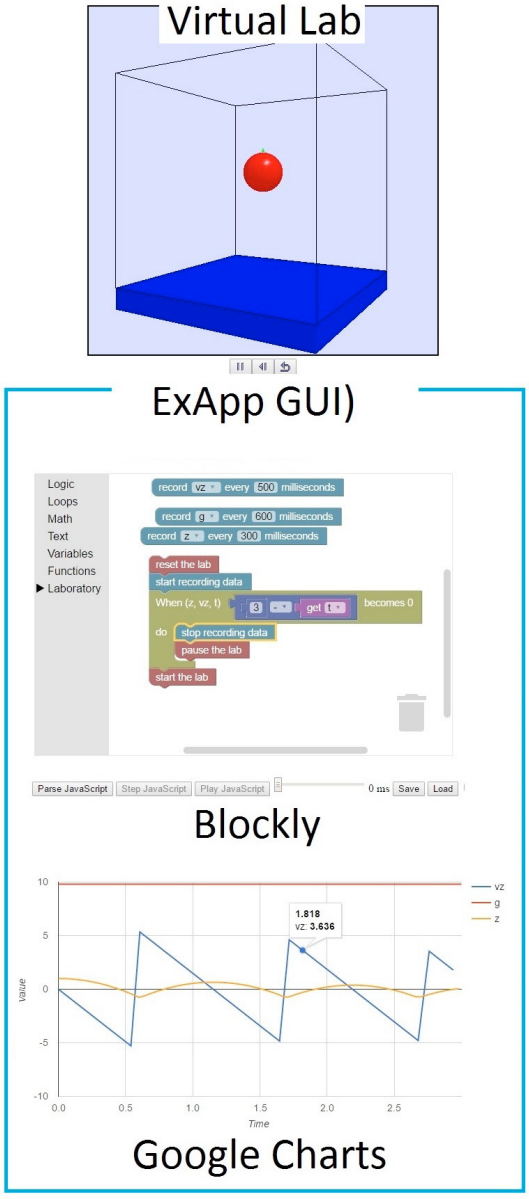

Fig. 1. ExApp GUI (Blockly Code and Google Charts) with a virtual lab modeling a bouncing ball

2. THE EXPERIMENT APPLICATION

The Experiment Application (ExApp) is composed of four elements: (1) Blockly Editor to design the experiment, (2) Google Charts visualization to review the experiment results, (3) the API to share information between the VRL and the experiment and (4) the experimentation language. The first two elements (Blockly and Google Charts), which comprise the application GUI (see Figure 1), are explained in the following lines. The API and the experimentation language, which are closely related, will be explained in Section 3. As has been mentioned in the introduction, the ExApp is entirely independent of the lab used. If the lab, whether a virtual lab, a remote lab, an hybrid lab or a simulation, implements the API proposed by the authors the ExApp can be used.

The experiment designer is developed using Blockly. It is a free and open source library that adds a visual code

editor to web and Android apps. The Blockly editor uses interlocking, graphical blocks to represent code concepts like variables, logic expressions, loops, and more. It allows users to apply programming principles without having to worry about syntax or the laboratory structure. Blockly is used in lots of learning applications as: Blockly Games (a set of educational games that teach programming concepts), MIT's App Inventor (to create applications for Android), Code.org (to teach introductory programming to millions of students in their Hour of Code program), Wonder Workshop (to control their Dot and Dash educational robots), the Open Roberta project (to program Lego Mindstorms EV3 robots), or ScratchyCAD (a web based parametric 3D modeling tool which allows users to create 3D objects). Using Blockly to create experiments for VRLs rather than other programming languages is a valuable asset from the experience of the authors. As VRLs can be used by any person, with or without programming skills, Blockly is the easiest way to start creating small algorithms to conduct experiments. Furthermore, this code editor offers interesting features that favor the web use, maintaining the power of traditional languages:

(1) Implemented with JavaScript.
(2) Support for many programmatic constructs including variables, functions, arrays.
(3) Minimal type checking supported.
(4) Easy to extend with custom blocks.
(5) Localized into 50+ languages.

In the last stage of the experiment, the data analysis is presented to the user using Google Charts, Zhu (2012). This library is used to visualize data on a website. As Blockly, it is free and open source. Google Charts provides a large number of ready-to-use chart types. It is able to represent from simple line charts to complex hierarchical tree maps. It is highly customizable and supports dynamic data and controls to create interactive dashboards. It also offers functions to import and export data to other formats, so it is possible to use them with R, Microsoft Excel or other common analysis tools. It is a JavaScript library so its incorporation to an online tool is simple and clean.

2.1 Benefits from using ExApp

ExApp users can be distinguished in three groups according to their role in a teaching experience: (1) lab designers, (2) teachers and (3) students. Each of them benefits from this tool in different ways.

Lab designers are in charge of creating the Lab. Usually they have to create the model, the view, decide which variables are going to be visualized in charts and add some interactive elements to control the execution of the lab by changing some variables or internal functions. If designers use a tool that implements the API proposed in this paper (EjsS at the moment), they will not need to change anything in their lab implementation to use ExApp. Furthermore, they will not need to create charts or any interactive element to control the lab. As the ExApp has access to every variable, the final user can decide the way to work with the lab and the variables that are important to show in the charts. The designer will focus only in the model definition and its view. This means that the designer will need less time to create the lab and the experiences or assignments that can be proposed are not limited in the design. As an example, in a bouncing ball simulation, if the designer do not add a way to change the gravity variable, students will not be able to study its implication in the model.

Teachers have to define the lab experiences for the students. If the lab is open and not restricted by the designer pretensions, the teacher will have plenty of possibilities and experiences to offer to the students. From simple algorithms to discover the important variables of a system, to create from zero the PID level controller of a water tank. To deploy the ExApp and the lab on a web page is as simple as preparing a HTML with the two elements. Authors' next step is to include the ExApp as a Moodle plugin. In this way, the lab, the ExApp, the experiment files and the results are manage by Moodle. The correction of these type of interactive experiments using Blockly is as easy as running the student file and evaluating the results obtained. The time needed by teachers to explain how to use the tool is extremely short comparing with other simulation tools that allow the creation of experiment scripts. About evaluating the assignments, teachers may give value to whether the correct result is obtained as well as how the student reached to that solution. Teachers have the possibility to assess the experiments structure, to study the algorithms used and to perform the students experiments as many times as needed just with one click. Even more, Blockly and other similar tools as Scratch are currently being used in elementary schools. This means, near future users will not need any extra explanation about how to used it, because students will be familiar with these tools.

Students are the final users of the lab and the ExApp. Nowadays, Blockly is the first step to start learning programming skills, so even students with no programming knowledge will find ExApp an easy tool to code their scripts. Blockly offers visualization features as highlighting blocks which are executed at a certain time so it is very easy to follow the execution flow of the experiment and correct possible mistakes. For the same lab, students can face different assignments depending on their skills which promote, among others, imagination to solve the assignments, learning interest, critical thinking, being challenged and inquiry-base learning. Also, by scripting the experiment, students avoid tedious or repetitive tasks without any educational value. They are able to exchange experiments with the teacher or other students, comparing results and confronting different approaches. Visualizing, collecting and assessing results is easier thanks to Google Charts.

3. IMPLEMENTATION

To achieve the objective of controlling every aspect of a VRL, an interface between the ExApp and the VRL is needed. The API should then contain the following elements:

(1) Elements to initialize and configure the ExApp.
(2) Elements to access variables.
(3) Elements to specify algorithms.
(4) Elements to control the execution of the lab.

(5) Elements to review the results.

These elements, how they conform the experimentation language and how are implemented in the ExApp are described in the following subsections.

3.1 Elements to initialize and configure the ExApp

An initialize process is needed to configure correctly the ExApp and to link it to a lab. This means that ExApp has to receive the object that contains the variables and the lab functionality (In EjsS labs, the *model* variable). Once the ExApp and the lab are linked, all variables from the lab system are classified by type and prepared for their use in the code. Optionally, an XML file can be configured to show more or less Blockly blocks in order to create from the most standard to the most complex algorithm. The XML is configured with all the blocks by default.

3.2 Elements to access variables

The lab has to implement two functions to set and get the variables of the model. EjsS labs, for example, implements these functions using a JSON Object, *model.userUnserialize({variable,value})* to set and the function *model.userSerialize()* to get the values of the variables.

The experimentation language implement two blocks using the two functions seen in Figure 2. The one on top will show a message with value of the selected variable (t) when executed and the one in the bottom will set the value of the statement linked to it, in this case, it will set the variable g with the value *3*. Model variables will appear in the variable chooser of the block automatically, as it is possible to see in Figure 2.

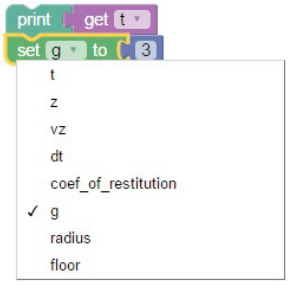

Fig. 2. How to set and get variables from the lab

3.3 Elements to specify algorithms

Experiments not only require specific algorithms that use the variables from the lab, but also, additional functions and variables defined by users, so the API should allow it. Thanks to the JavaScript features this is easy to implement.

To do this, the experimentation language must provide different blocks to create standard algorithmic constructions to allow users to write complex algorithms, if required. Figure 3 shows declaration of a new function in the code, how to call it and few blocks to create different types of statements.

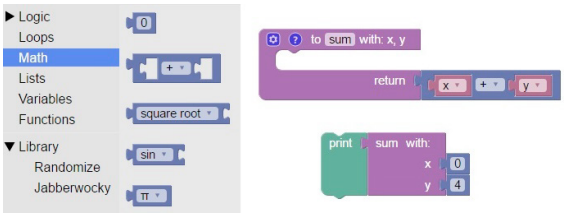

Fig. 3. How to define and call a function using Blockly

3.4 Elements to control the execution of the lab

The API should implement different ways to control the lab execution. If it is a lab whose evolution depends on time, instructions to start, pause or stop the lab are necessary. Also, more complex functions are needed, like *events* (do something when a given condition is met). For example, "run the simulation until the level of the tank is greater than 10" or "run the simulation increasing the set point by 50% when t = 10". The lab should implement the function *model.addEVent(conditionCode, actionCode)* in order to allow these type of statements.

Figure 4 shows how the experimentation language implements this function to add events to the lab. First of all, the lab is reset and then an event is added. The condition is 3 minus the variable t from the lab. And the action consist in pausing the lab. After this, the lab is started. When the variable t gets to 3 the lab will pause.

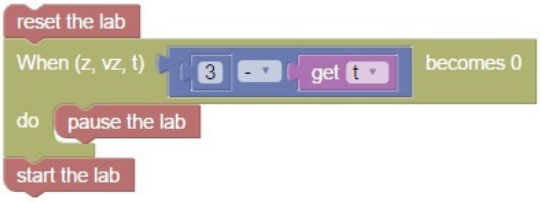

Fig. 4. How to control the lab execution using Blockly

3.5 Elements to review the results

The API should provide functions to review and assess obtained results from the experiment by comparing output data with visual elements as plots or graphs. Google Charts is the tool used to visualize data. Data are registered in tables with the values of the selected variable in one column and the time variable of the model in another column.

The experimentation language has three blocks to implement this functionality. Figure 5 shows , on the left, the three of them and the obtained chart on the right. The first block is to declare which model variable is going to be recorded and the sample period for it. It is possible to declare as many variables as needed. Once the recording variables are declared, it is only precise to use the *start recording* and the *stop recording* blocks to select the time bands to track those variables. Once the experiment starts, the chart will visualize the selected variables.

4. EXAMPLE OF USE

This section presents an example that shows the usefulness of the ExApp and the advantages described throughout

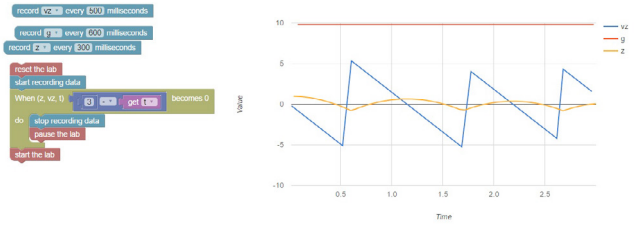

Fig. 5. How to review data with Blockly

the paper. The presented simulation, Simple Harmonic Oscillator (see figure 6), was taken from the ComPADRE Digital Library. While this simulation is a great tool by itself, it is a bit limited in terms of visualization and interaction with the model. With the ExApp, however, a student can use the simulation to conduct as interesting experiments as the ones showed in the following subsections.

4.1 Simple Harmonic Oscillator

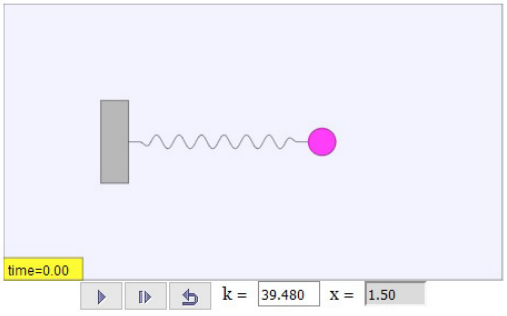

Fig. 6. Simulation of the Simple Harmonic Oscillator

The Simple Harmonic Oscillator Model illustrates the motion in the horizontal dimension of a mass m situated at the end of a spring of length l. The model assumes that the reaction of the spring to a displacement dx from the equilibrium point can be modeled using Hooke's Law ($F(dx) = -k\ dx$) where k is the spring constant. Thus, applying Newton's Second Law, it is obtained the following equation:

$$\frac{d^2x}{dt^2} = \frac{-k}{m}(x - l) \tag{1}$$

The system of coordinates has its x-axis along the spring and the origin at the spring's fixed end. The particle is located at x and its displacement from equilibrium $\delta x = x - l$ is zero when $x = l$.

The spring is initially stretched and the ball has zero initial velocity. With the GUI is possible to change the initial position of the ball, the spring constant value and to play, pause and reset the simulation. Notice that using the ExApp none of these features are necessary to interact with the model.

4.2 Example 1: Visualizing variables involved in the system evolution

Identifying the important variables of the system is the first step in every experiment. It is easy to understand, seeing Equation 1, that the variables that determine the system evolution are the velocity and the position of the free end of the spring. But, what else could be done if the system is unknown or the equation is not so simple. With the ExApp is very easy to visualize the system variables so with a quick glance is possible to see how variables evolve. Figure 7 shows an experiment (left part) where all the variables of the lab are visualized in a chart (right part) for 5 seconds.

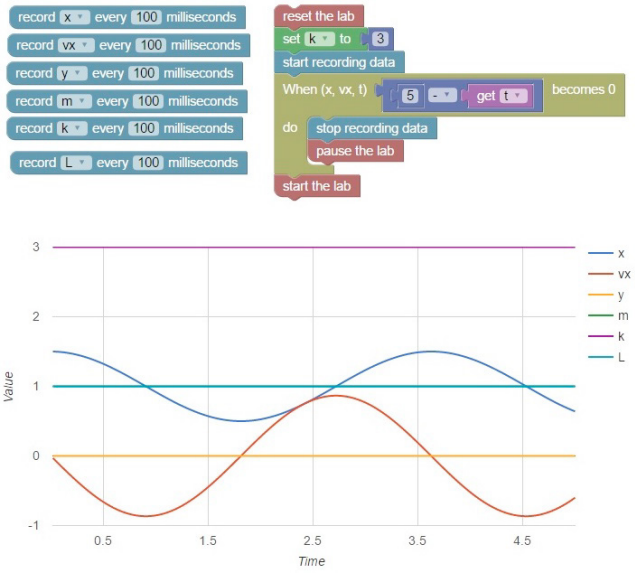

Fig. 7. Example 1: Visualizing variables

4.3 Example 2: Study how the value of the spring constant affects the velocity

Interacting with the model variables is easy using ExApp. Thus, a good assignment would be to study the relation between those variables to discover the equations that model the system. ExApp allows users to vary one variable and see the effect in another one using the events and the chart. The experiment starts resetting the lab, and setting the spring constant to *1*. After that, three events are added, two to change the spring constant to *10* at 2 seconds and another one to set it to *20* after 4 seconds. Finally, the execution finishes after 6 seconds. Figure 8 shows the blocks used to create this experiment and the chart with the results. It is easy to see the proportional relation between the spring constant and the velocity seeing the chart and the Hooke's Law.

4.4 Example 3: Study the relation between the mass and the velocity of the oscillator

This example is similar to the Example 2 but the way of solving it is different. Instead of creating events and changing the desired variable, it is possible to run the experiment several times with different initial conditions and visualize the outputs overlapped in the chart. Figure 9 represent the code of one of the experiments, in this case, the user should change the mass every time that runs the experiment, in the first experiment was set to *1*, in the second one to *3* and finally to *5*. The chart shows overlapped the velocity for every experiment. It is simple to see that the frequencies of the oscillating velocities are

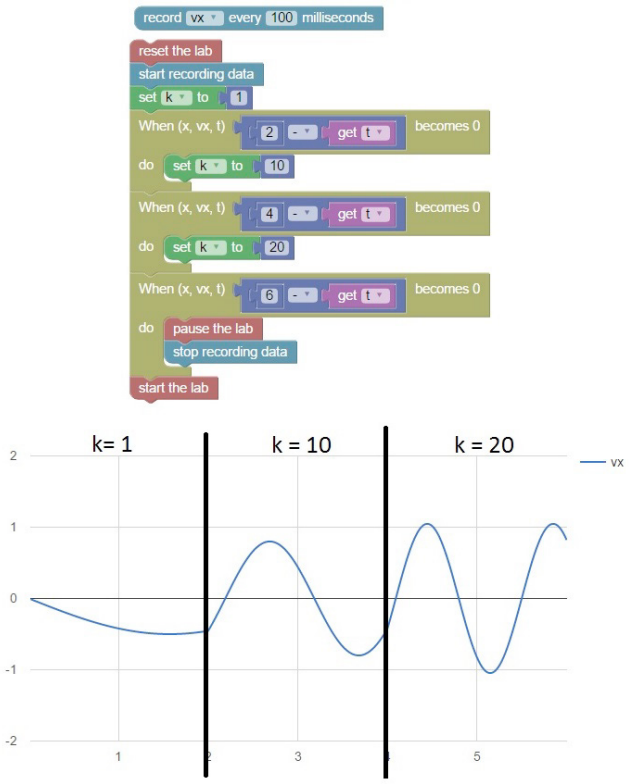

Fig. 8. Example 2: Relation between the spring constant and the velocity

inversely proportional to the mass values by just looking the chart and having in mind the Newton's Second Law.

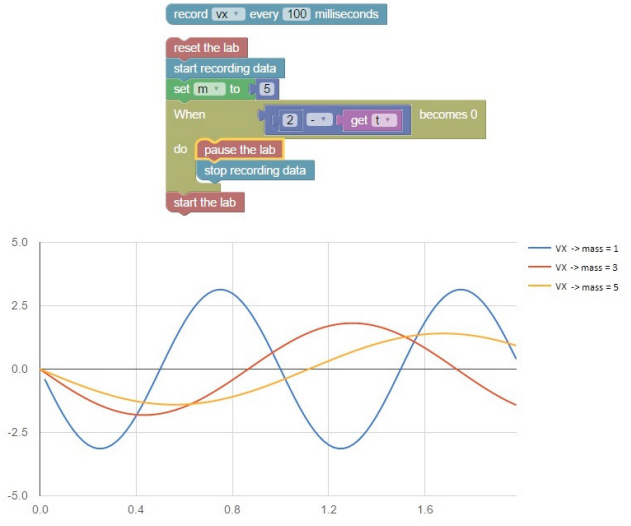

Fig. 9. Example 3: Different velocity by changing the mass

5. CONCLUSION

The first version of this web application is ready to use. It is possible to: access and modify all the variables from the laboratory, create algorithms and functions, and control the execution of the experiment. Additionally, users can execute the experiment step by step or run the whole script with a modifiable interval of time between code sentences.

Authors are currently working in the implementation of a Moodle plugin to use the tool easily. Once developed, everyone who wants to use this tool in his/her lab will only have to add the plugin to Moodle and select the desired features of the tool to use in each EjsS lab (or any other lab that implements the proposed API). From the authors point of view, these type of plugins can change the way of performing experiments, creating new experiences in Learning Management Systems (LMS).

Authors are in the process of testing the initial design creating different types of experiments of practical use in teaching Automatic Control and Physics. In this way, the assessment and tests of the tool are currently in process with different groups of students. Early results show that this implementation is both simple and flexible, supplying users a great deal of control of the running simulation. The combination of JavaScript and Blockly has been crucial in making the proposed implementation very fresh and clear. The way EjsS implements the VRLs allows external applications to access easily all its variables without any required modification in the lab applications already developed. The benefits of this application are studied in an example of implementation using the Simple Harmonic Oscillator system.

In a more general context, authors believe the API proposed in this work can effortlessly be adapted to different lab implementations or to any future standard protocol.

REFERENCES

Dormido, S., Dormido-Canto, S., Dormido, R., Sánchez, J., and Duro, N. (2005). The role of interactivity in control learning. *International Journal of Engineering Education*, 21(6), 1122.

Elmqvist, H., Mattsson, S.E., and Otter, M. (1998). Modelica: The new object-oriented modeling language. In *12th European Simulation Multiconference, Manchester, UK*.

Esquembre, F. (2004). Adding interactivity to existing simulink models using easy java simulations. *Computer Physics Communication*, 156, 199–204.

Heck, B.S. (1999). Future directions in control education [guest editorial]. *IEEE Control Systems*, 19(5), 36–37.

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, 14–38.

Marron, A., Weiss, G., and Wiener, G. (2012). A decentralized approach for programming interactive applications with javascript and blockly. In *Proceedings of the 2nd edition on Programming systems, languages and applications based on actors, agents, and decentralized control abstractions*, 59–70. ACM.

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

Wieman, C.E., Adams, W.K., and Perkins, K.K. (2008). Phet: Simulations that enhance learning. *Science*, 322(5902), 682–683.

Zhu, Y. (2012). Introducing google chart tools and google maps api in data visualization courses. *IEEE computer graphics and applications*, 32(6), 6.