1/10/2021

REIABL

IO 6: WEBLAB TUTORIAL OF TECHNICAL DESIGN AND IMPLEMENTATION – INTRODUCTION

Co-funded by
the European Union

1. INTRODUCTION

Web Laboratories (weblabs) have proven to be a fantastic tool for teaching and learning, especially, as a complement to traditional hands-on labs in technical, engineering and science fields. Unfortunately, designing and implementing a weblab is usually not an easy nor a quick task, making this kind of resources difficult to find in many university courses. Therefore, an effort to educate our educators on how to create weblabs, so they can add them to their courses, is required. The RELAB project is aware of this lack, and its Intellectual Output 6 (to which this document belongs) addresses this problem.

1.1. Objective

This manual is part of a series of documents that were created with the intention to give some very specific instructions on how to create a weblab and deploy it into an online course to make it accessible for the students. This way, our intention and hope is that more teachers in Europe and around the world would feel comfortable to develop their own weblabs and use them in their courses in order to improve their teaching.

1.2. Target audience

The target audience of these series of manuals are teachers of technical, engineering and science subjects who would like to include more experimentation activities in their courses without the restrictions that hands-on labs impose, but do not have a clear idea on how or where to start.

An assumption made in this document is that the reader already knows what a web-based lab, online lab, or remote lab, is, and has a slight idea on what are the parts that form part of it.

1.3. Scope

While weblabs can be developed using many different technologies, tools, and approaches (many of which are equally valid in many aspects), these manuals were conceived to be as short and as simple as possible to try to introduce the development of weblabs in a friendly way.

The tools proposed in these documents for developing weblabs, and for which specific instructions of use can be found, are all free and open source, following the European directives and spirit and facilitating their access, adoption, and adaptability.

2. GENERAL VIEW AND TOOLS

A weblab is usually comprised of a client, web-based application and some assets on the server side. Whether these assets are a computer running a simulation software or one connected to a real plant (like the example in Figure 1), some tasks are always performed in a remote laboratory (the server side). Therefore, normally at least two software tools or technologies are required to prepare a weblab.

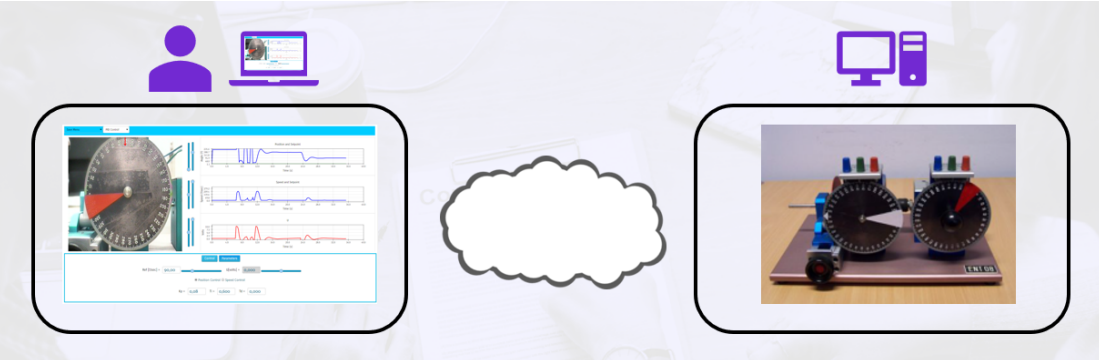

Figure 1. Basic architecture of a weblab

For the sake of simplicity, these manuals are focused only on the use of the following software tools.

For the client side:

- Easy Java/Javascript Simulations (EjsS) – It is a free and open-source tool especially created to make the task of developing simulations and web applications for weblabs as easy as it can get. The result of an EjsS project is an HTML5 web application (like the one shown in the previous figure) that can be easily integrated into online courses.

For the server side:

- LabVIEW – This software from National Instruments is one of the most common ones used in universities and research centers to operate and control lab physical devices, systems and experimentation plants. As good as it is, this is a commercial software that requires paying a license, and not all groups or institutions can afford it.
- Python – This is a well-known and powerful free programming language that allows connections with many hardware devices too, such as Arduino, Raspberry, and many more. Since these manuals present guides for designing and implementing weblabs both with LabVIEW and Python, users will always have an available choice at their reach to follow it.

The following documents focusing on the use of each of these tools for their corresponding part in the development and implementation of weblabs provide the links to download the necessary resources and software that is required.

THE FOUR STEPS

There are four main steps related to weblabs design, development and use. This document in general, and this section in particular, only addresses them in a general way, giving a short introduction to each of these steps. However, four more manuals are part of this series of documents, each of them tackling with each of these steps in more detail, and giving a step-by-step guide on how to address these tasks with the proposed tools. Before reading them, it is recommended to first go through the lines here to get a general idea and overview of the whole process.

The four aforementioned steps are: 1) the server-side development and implementation, 2) the client-side development and implementation, 3) the deployment of the weblab app into an online course and 4) security and communication issues. The next sections introduce each of them and contextualize the rest of the manuals in this series within their particular scope.

2.1. Server-side development & implementation

The server-side development for a weblab usually has to solve connecting a computer with real hardware, in the form of actuators, sensors, etc. This can be done through a variety of hardware (Arduino, Raspberry, Data Acquisition Cards, PLCs, etc) and software (LabVIEW, Beckhoff, Matlab, Python, etc.) solutions.

These manuals do not enter on the hardware part, as this is extremely dependent on the type of experiment you are preparing and the resources you might have available or at your reach. For the software part, the documents in this series focus on two solutions (LabVIEW and Python), both equally valid and easy to connect to a web-app generated with EjsS.

Those familiar with LabVIEW and who can use the software, would probably prefer this option, as there are good chances the lab hardware they need to connect already has a LabVIEW interface and/or API. Any other person is recommended to use the Python solution, as it presents a high degree of interoperability and connectivity, making it ideal for most applications.

2.2. Client-side development & implementation

The client-side development includes creating the HTML5 web application that would allow students to interact with the remote lab assets. This usually includes buttons, sliders, graphs, input fields and/or a video stream of what is going on in the lab room.

Any HTML + Javascript + CSS programming could give you a weblab HTML5 app as a result, but the amount of work to program this from scratch is considerable. Other solutions, such as using Unity or similar web-friendly engines, are also possible, but the same problem remains. These manuals will focus on the use of EjsS for this task, as it is especially designed for this and anyone can download it and use it for free.

2.3. Deployment

While the work required on the server side is a must in order to set up a weblab, it is the client web application the one that needs to get deployed in an online course to make it accessible to the students and enable them to perform their lab tasks.

Again, many solutions are possible for this, but these manuals focus in a particular one with the idea of not being general and vague but specific and clear instead. The guide you will find in the manual for the third step takes Moodle as the target LMS in which to deploy the web HTML5 applications created with EjsS, and uses the EJSApp set of plugins to facilitate the labor of integrating these apps in a Moodle course.

2.4. Security & communications

Usually neglected, security and communications issues with weblabs are extremely important, and, sometimes, difficult to address. A complete manual on how to solve these issues is out of the scope of this series of manuals, but the last document does provide a brief overview of the problem and some possible solutions.