

SPANISH OPEN UNIVERSITY (UNED)

**New Generation Virtual and
Remote Laboratories: Integration
Into Web Environments 2.0 With
Learning Management Systems**

by

Luis de la Torre Cubillo

Supervisors:

José Sánchez Moreno and Sebastián Dormido Bencomo

School of Computer Sciences

Department of Computer Sciences and Automatic Control

May 6, 2013

Declaration of Authorship

I, Luis de la Torre, declare that this thesis titled, 'New Generation Virtual and Remote Laboratories: Integration Into Web Environments 2.0 With Learning Management Systems' and the work presented in it are my own. I confirm that:

- This work was done wholly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Many that live deserve death. And some that die deserve life. Can you give it to them? Then do not be too eager to deal out death in judgment. For even the very wise cannot see all ends."

Gandalf the Grey.

J.R.R. Tolkien, *The Fellowship of the Ring*.

SPANISH OPEN UNIVERSITY (UNED)

Abstract

School of Computer Sciences

Department of Computer Sciences and Automatic Control

Doctor of Philosophy

by Luis de la Torre Cubillo

Learning Management Systems (LMS) are software for web applications oriented for the administration, documentation, tracking, and reporting of e-learning programs. Moodle is a free source LMS with more than 63 million users, which makes it the most used LMS around the world. Like some other LMS, the stated philosophy of Moodle includes a constructivist and social constructionist approach to education, emphasizing that learners (and not just teachers) can contribute to the educational experience. This is the web 2.0 applied to education.

Easy Java Simulations (EJS) is an authoring tool written in Java that helps to create interactive simulations in Java, mainly for teaching and learning purposes. By means of this tool, instructors can easily create virtual and/or (if they also use the appropriate additional software) remote laboratories. While virtual laboratories are based on mathematical models, remote ones use real equipment and so, the experiments are carried out in the reality.

Virtual and remote laboratories (created with EJS, for example) as well as LMS (Moodle, for example) offer different but fundamental educational tools to both teachers and students. Since these resources are complementary (and not mutually exclusive), e-learning programs should offer both kinds of tools to be considered

a complete experience for students. However, the integration between these two different resources is still an open issue that must be addressed.

This thesis gathers together the two previous resources and provides the necessary tools and methodology for developing web experimentation portals (such as UNEDLabs, also presented here) that can offer e-learning programs based on: 1) experimentation (thanks to the use of the virtual and remote laboratories) and 2) theory documentation provision, social interactivity and easy management (thanks to the use of a LMS). Moreover, the presented tools and methodology allow people not specialized in these particular topics (such as they are the vast majority of the teachers) to easily create these kind of experimentation portals and populate them with virtual and/or remote laboratories (VRLs). Four new Moodle plug-ins facilitate this task: EJSApp, EJSApp Booking System, EJSApp Files Browser and EJSApp Collab Sessions.

Nowadays, UNEDLabs holds two courses (among others); one in Control Engineering and another one in Physics. Both of them offer three experiments which are all available in the two possible versions: virtual and remote. The Physics course is still growing and new experiments are added every year. Right now, the three available ones are a motorized rotatory laser for studying the light in isotropic media, a motorized optical bench for determining the focal length of a thin lens, and an experiment with three springs and related to Hookes law.

Thanks to the tools presented in this work, not only all the EJS laboratories in UNEDLabs are added and integrated into the Moodle web portal in a very easy and natural way, but they also acquire several special functionalities they lack when they are used outside this LMS. The first of these functionalities is the capability of saving and/or loading files to/from the private files repository in Moodle. The second one is the possibility to create collaborative experimental sessions with other users of Moodle who are enrolled to the course the EJS lab belongs to. The

third one is an automatic integration with a booking system specifically designed for managing users access to the remote laboratories. Finally, EJS labs in Moodle can be administrated exactly in the same way as any other Moodle resource or activity, meaning they can be updated or deleted; their access can be restricted to a certain group of users or to users that have previously fulfilled some steps or conditions; security copies are automatically performed during Moodles system backups, etc.

Los Sistemas de Gestión del Aprendizaje (LMS, por sus siglas en inglés) son herramientas software para crear aplicaciones web orientadas a las administración, documentación, seguimiento y reporte de programas de aprendizaje en la red o aprendizaje digital. Moodle es un LMS de código abierto y gratuito que tiene más de 63 millones de usuarios, lo que lo convierte en el LMS más utilizado en todo el mundo. Al igual que hacen otros LMS, la filosofía sobre la que Moodle se asienta incluye una visión constructivista y de construcción social de la educación, poniendo énfasis en que los estudiantes (y no sólo los profesores) puedan contribuir a la experiencia y el proceso educacional. Esto es, la web 2.0 llevada a la educación.

Easy Java Simulations (EJS) es una herramienta gratuita y de código abierto escrita en Java que permite crear simulaciones interactivas en Java de manera sencilla y que está especialmente diseñada para usos de enseñanza y aprendizaje. Gracias a esta herramienta, los docentes pueden crear laboratorios virtuales y/o remotos (si utilizan, además, el software adicional necesario). Mientras que los laboratorios virtuales están basados en modelos matemáticos, los laboratorios remotos usan equipamiento real y, por lo tanto, los experimentos se desarrollan realmente.

Los laboratorios virtuales y remotos (creados con EJS, por ejemplo), así como los LMS (Moodle, por ejemplo) ofrecen recursos diferentes pero fundamentales tanto a los docentes como a los estudiantes. Puesto que estos recursos son complementarios (y no mutuamente excluyentes), los programas de aprendizaje a través de internet (o los que se apoyan en la web) debería ofrecer ambos tipos de recursos para ser considerados una experiencia completa para los estudiantes. Sin embargo, la integración entre estos dos recursos suponía aún un problema que no tenía fácil solución.

Esta tesis reúne los dos recursos anteriores y ofrece las herramientas necesarias

para desarrollar portales web de experimentación (como UNEDLabs, también presentado en este trabajo) que pueden ofrecer programas de enseñanza en internet basados en: 1) la experimentación (gracias al uso de los laboratorios virtuales y remotos) y 2) la provisión de documentación teórica, interactividad social y facilidad de gestión y manejo (gracias al uso de un LMS). Y lo que es más, las herramientas y metodología presentados permiten que personas no especializadas en estos temas particulares (tal y como son la inmensa mayoría de profesores) puedan crear muy fácilmente este tipo de portales de experimentación y poblarlos con laboratorios virtuales y/o remotos (VRLs). Cuatro nuevos plugins de Moodle facilitan esta tarea: EJSApp, EJSApp Booking System, EJSApp Files Browser y EJSApp Collab Sessions.

Actualmente, UNEDLabs aloja dos cursos; uno sobre Ingeniería de Control y otro sobre Física. Ambos ofrecen tres prácticas, las cuales están todas disponibles en sus dos versiones posibles: virtual y remoto. El curso de Física aún se encuentra en pleno crecimiento y cada año se añaden nuevos experimentos al mismo. En estos momentos, los tres laboratorios disponibles son: un láser motorizado capaz de girar en torno a una cubeta, usado para estudiar las leyes de la reflexión y la refracción de la luz en medios isótropos; un banco óptico motorizado, que permite determinar la distancia focal de una lente delgada; y un experimento con tres muelles que está relacionado con la ley de Hooke.

Gracias a las herramientas presentadas en este trabajo, no sólo se ha logrado que los laboratorios de UNEDLabs fuesen integrados en el portal web de Moodle de una manera muy sencilla y natural sino que, además, estos laboratorios adquieren algunas nuevas funcionalidades especiales de las que carecen cuando se utilizan fuera del entorno de este LMS. La primera de estas funcionalidades es la capacidad de grabar y/o cargar ficheros del repositorio privado de ficheros de Moodle. La segunda, es la posibilidad de crear sesiones colaborativas de trabajo experimental

con los propios laboratorios, invitando a las mismas a cualquier usuario de Moodle matriculado en el curso al que pertenezca el laboratorio seleccionado. La tercera es la integración automática con un sistema de reservas especialmente diseñado para organizar el acceso de los usuarios a los laboratorios remotos. Finalmente, los laboratorios EJS en Moodle se administran exactamente igual que cualquier otra actividad o recurso de Moodle, significando esto que pueden ser actualizados o borrados; que su acceso puede restringirse a un cierto grupo de usuarios o a usuarios que previamente han completado algunas fases o condiciones del curso; que se realizan copias de seguridad automáticamente durante los respaldos de sistema que Moodle efectúa, etc.

Acknowledgements

Foremost, I'd like to thank my supervisors, Profs. Sebastián Dormido and José Sánchez, who shared with me their knowledge, and a lot of their expertise and research insight. They both supported me in every possible way all the time and that is the biggest help anyone can receive. I'd also like to express my gratitude to Profs. Manuel Yuste, Carmen Carreras, and, specially, Juan Pedro Sánchez; the remote laboratories of this work would not have been completed so quickly without them. And I am deeply grateful to the Vicerrectorado de Calidad e Innovación Docente of the UNED and its corresponding Vicerrector, Prof. Miguel Santamaría for the trust and monetary support they gave me during my first two years of work in this project.

A very special thanks to Prof. Rubén Heradio. His help, good humor and loyal friendship has accompanied me since 2010 and has made this journey much easier and amusing.

What can I say about Profs. Marco Márquez and Andrés Mejías? We didn't have the opportunity to spend much time together, either working or joking (actually, both things always came together), and still, it was enough to give a great push to my work.

I also want to say thanks to Prof. Francisco Esquembre. He provided me with some very good ideas and he answered each of my questions about Easy Java Simulations.

It is difficult to overstate my appreciation to (ex) Prof. Bonifacio de Andrés Toro, who first brought me into the world of research and with whom I began to learn about control engineering. Not only a great mentor and colleague, he has also been a cornerstone in my professional development. I cannot think on him without also thinking on Prof. Jesús Manuel de la Cruz, who first put me in

contact with Prof. Sebastián Dormido, and Prof. Eva Besada, with whom I have had the pleasure to collaborate and work in different topics than the ones related to this thesis. My sincere thanks to all of them.

I would really like to thank all of my friends (those in Málaga and those in Madrid mostly but also some others who are in other places) which have accompanied me in discovering and enjoying life. However, the list would be too long so, even though you are all extremely important to me, I will simply say thank you very much to you all.

I cannot finish without saying how grateful I am with my family: my parents, my two brothers and my sister. Without them, I would not have come to Madrid and this thesis would have never existed. Also, they were the best support I could ever had in an extremely difficult moment of my life. Particular thanks to Diana (we finally never got married but you know you were family to me). It is due to her love and affection that I could stay strong even at the most difficult stages of my research. And of course, I could not forget about Diana's parents (who took me in as I was their son) and sister, who also became my family during those wonderful years.

All these people have always supported and encouraged me to do my best in all matters of life. To all of them I humbly dedicate this thesis. They all deserve much more.

Contents

Declaration of Authorship	III
Abstract	v
Acknowledgements	xv
List of Figures	xxi
Abbreviations	xxv
Symbols	xxvii
1. Introduction	1
1.1. Project Description	1
1.1.1. Background and General View	1
1.1.2. Some Considerations About Virtual Laboratories	4
1.1.3. Some Considerations About Remote Laboratories	5
1.1.4. Some Considerations About Learning Management Systems	8
1.1.5. UNEDLabs	9
1.2. State of the Art	11
1.2.1. Other Solutions for Providing Experimentation Facilities in Distance Education Paradigms	11
1.2.2. Virtual and Remote Laboratories in Physics	13
1.2.3. Virtual and Remote Laboratories in Engineering	14
1.2.4. Conclusions About the State of the Art in Virtual and Re- mote Laboratories	15

1.2.5.	Analysis of Existent Learning Management Systems	16
1.2.6.	UNEDLabs' Ancestors	21
1.2.7.	Other Networks of Web-based Laboratories	24
1.3.	Goals	26
1.4.	Structure of the Document	29
2.	Framework for Web Experimentation Portals. UNEDLabs	31
2.1.	Architecture	31
2.1.1.	General View	31
2.1.2.	Global Architecture	35
2.1.3.	RLs Interface - Control and Communication	38
2.1.3.1.	Client-Side Implementation	38
2.1.3.2.	Server-Side Implementation	39
2.2.	UNEDLabs: A Framework Application Example	40
2.2.1.	Network of VRLs	41
2.2.2.	Web Portal	42
2.2.3.	Decentralization	45
2.2.4.	Courses	46
2.2.5.	A VRL Example in UNEDLabs	49
2.2.6.	The Booking System	49
3.	New Web-Labs Developed	55
3.1.	Light in Isotropic Media	55
3.1.1.	Introduction	55
3.1.2.	Virtual Laboratory	58
3.1.3.	Remote Laboratory	61
3.2.	Hooke's Law and Non-Linear Springs	64
3.2.1.	Introduction	64
3.2.2.	Virtual Laboratory	66
3.2.3.	Remote Laboratory	68
3.3.	Focal Length of Thin Lenses	72
3.3.1.	Introduction	72
3.3.2.	Virtual Laboratory	74
3.3.3.	Remote Laboratory	76
4.	New Features Developed	79
4.1.	Introduction	79
4.2.	Moodle Plugins	80

4.2.1.	Activity/Resource Plugins	80
4.2.1.1.	EJSApp	81
4.2.1.2.	EJSApp Booking System	87
4.2.2.	Block Plugins	90
4.2.2.1.	EJSApp File Browser	90
4.2.2.2.	EJSApp Collaborative Sessions	92
	Supporting a Variety of Learning Paradigms	95
	Supporting Deep Collaboration	97
	Usability	98
	Scalability	98
	Moodle Support for Synchronous Collaborative VRLs	99
4.2.3.	EJSApp External Interface. Use With Third Party Applications	103
4.2.3.1.	Importing the EJSApp external interface	104
4.2.3.2.	Retrieving information from EJSApp	104
	EJSApp instances	104
	Getting the size of an EJSApp instance	108
	Getting EJSApp state files	109
4.2.3.3.	Drawing an EJS simulation	109
4.2.3.4.	Application Example: IPAL	111
4.3.	EJS Additions	111
4.3.1.	EJS Support for Synchronous Collaborative VRLs	113
4.3.2.	Saving Files to Moodle	117
4.3.3.	Loading Files from Moodle	117
4.4.	Information Exchanged Between Moodle and EJS	118
5.	Step-By-Step Process for Preparing and Deploying a VRL	121
5.1.	Preparing the VRL	121
5.1.1.	Using EJS to prepare the VL application	122
5.1.2.	Using LabView to create the RL controlling program	123
5.1.3.	Using EJS to prepare the RL application	124
5.2.	Deploying the VRL	126
5.2.1.	Installing the EJSApp plugin	127
5.2.2.	Sharing the VRL	129
6.	Conclusions and Future Directions	131
6.1.	Conclusions	131
6.2.	Future Work	133

A. Tools for Developing Web Experimentation Portals	139
A.1. Software Tools	139
A.1.1. Easy Java Simulations (EJS)	139
A.1.2. LabView	142
A.1.3. Matlab	146
A.1.4. TCP/IP Sockets and JIL and JIM libraries	147
A.1.5. Database Server (MySQL)	148
A.1.6. Web Server (Apache)	149
A.1.7. PHP	150
A.1.8. Moodle	151
A.2. Hardware Tools	155
A.2.1. National Instruments Motion	155
A.2.1.1. Controllers	157
A.2.1.2. UMI Connector Blocks	157
A.2.1.3. Drives	157
A.2.1.4. Motors	157
A.2.2. Arduino	158
A.2.3. Lego NXT	159
A.2.4. Other Components	161
B. Adapted Web-Labs: Heatflow System, Three Coupled Tanks and Servo Motor	163
B.1. Adapting Old VRLs Created With EJS	163
B.2. Virtual Laboratories	166
B.3. Remote Laboratories	168
C. Developer Documentation of the EJSApp Plugins	171
C.1. EJSApp	171
C.2. EJSApp Booking System	179
C.3. EJSApp File Browser	187
C.3.1. Methods of the block ej sapp file browser class	187
C.3.2. Methods of the block ej sapp file browser form class	188
C.3.3. Methods of the block ej sapp file browser renderer class	188
C.3.4. Methods of the ej sapp file browser tree class	189
C.4. EJSApp Collab Sessions	189

List of Figures

1.1. Typical Virtual Laboratories Deployment Architecture	6
1.2. Typical Remote Laboratories Architecture	8
1.3. Proposed Architecture for Deploying VRLs and Integrating them into a LMS with a Booking System	11
1.4. RLs State of the Art Study	14
1.5. Comparison of LMS Product Usage for the May 2007-2008 vs. May 2008-2009 Periods	20
1.6. LMS Market Share Trends	21
2.1. Communication Architecture for RLs	33
2.2. Communication Architecture for Collaborative Sessions	34
2.3. Global Architecture With SARLAB	36
2.4. Global Architecture With and Without SARLAB	37
2.5. UNEDLabs' Website Visits Statistics	42
2.6. UNEDLabs' Website Visiting Countries	43
2.7. UNEDLabs' Website Homepage (1)	44
2.8. UNEDLabs' Website Homepage (2)	46
2.9. A Course at UNEDLabs	47
2.10. Booking System Link in a Course at UNEDLabs	49
2.11. A Virtual Lab Example at UNEDLabs	50
2.12. Booking System	52
3.1. Experimental System of the Light in Isotropic Media Experiment	57
3.2. User Interface of the Light in Isotropic Media Experiment (Simula- tion Mode)	58
3.3. Linear Regression Applying Snell's Law	60
3.4. Linear Regression Applying the Paraxial Approximation	61
3.5. User Interface of the Light in Isotropic Media Experiment (Remote Mode)	62

3.6. Total Reflection in the Light in Isotropic Media Experiment (Remote Mode)	63
3.7. Experimental System of Hooke's law and Non-Linear Springs Experiment	65
3.8. User Interface of Hooke's Law and Non-Linear Springs Experiment (Simulation Mode)	67
3.9. Data Tab of VRLs Applications	68
3.10. User Interface of Hooke's Law and Non-Linear Springs Experiment (Remote Mode)	70
3.11. Linear Regression tool of Hooke's Law and Non-Linear Springs Experiment (Remote Mode)	71
3.12. Explanatory Scheme of Method 2	72
3.13. Experimental System of the Focal Length of Thin Lenses Experiment	74
3.14. User Interface of Focal Length of Thin Lenses (Simulation Mode)	75
3.15. Determining the Focal Length (Simulation Mode)	76
3.16. User Interface of Focal Length of Thin Lenses (Remote Mode)	77
3.17. Determining the Focal Length (Remote Mode)	77
4.1. Adding an EJSApp Activity to a Moodle Course	81
4.2. EJSApp Moodle Configuration Form for Adding a VRL	82
4.3. Descriptive Text in an EJSApp Activity	83
4.4. EJSApp Moodle Configuration Form for Controlling the Access Conditions to a VRL	84
4.5. EJSApp Moodle Configuration Form for Customizing the State of a VL	84
4.6. EJSApp Activity Using a Particular Initial State	85
4.7. EJSApp Moodle Configuration Form for Configuring the Access to a RL	85
4.8. Adding the EJSApp Booking System to a Moodle Course	88
4.9. Booking System View for a Teacher	89
4.10. Setting Users Permission Rights	90
4.11. Trying to Access a RL Without a Booking	91
4.12. <i>Online Users</i> block.	94
4.13. <i>Messages</i> block.	95
4.14. Example of two users participating in a collaborative experimental session.	96
4.15. Synchronous collaborative session in Moodle from the session director point of view	100

4.16. Synchronous collaborative session in Moodle from an invited user point of view	102
4.17. Importing the EJSApp External Interface	104
4.18. Retrieving EJSApp Instances in a Course	105
4.19. Result of Executing Figure 4.18.	106
4.20. Retrieving all EJSApp Instances in a Moodle Site	106
4.21. Retrieving the Size of an EJSApp Instance	108
4.22. Result of Executing Figure 4.21.	109
4.23. Retrieving Users' State Files Associated to an EJSApp Instance	110
4.24. Result of Executing Figure 4.23	110
4.25. Drawing EJS Simulations	111
4.26. Draw of an EJSApp	112
4.27. EJSApp and IPAL	113
4.28. Information Contained in MANIFEST.MF File	113
4.29. Control Panel of the Collaborative Session	114
4.30. Adding Collaborative Support in EJS	116
4.31. Information Exchanged Between Moodle and EJS	118
5.1. Creating a VL With EJS	122
5.2. Checking the Compatibility of an EJS applet With EJSApp	123
5.3. Creating a LabView Program for a RL	123
5.4. Creating a RL's UI With EJS	125
5.5. Configuring the SARLAB Element in EJS	126
5.6. Notifications Link in Moodle	127
5.7. Notifications Window in Moodle	128
6.1. Geiger Counter for the Radiation Experiment	134
6.2. Experimental System of the Electrostatic Experiment	135
A.1. EJS User Interface	141
A.2. LabView as a Graphical Programming Language	144
A.3. Front Panel and Blocks Diagram in a LabView Program	145
A.4. NI Plug-in Motion Control Solution	156
A.5. Arduino Board	159
A.6. Lego Mindstorms NXT	161
B.1. Heatflow Virtual Laboratory	165
B.2. Three Tanks Virtual Laboratory	166
B.3. DC Motor Virtual Laboratory	167

B.4. Heatflow Remote Laboratory	168
B.5. Three Tanks Remote Laboratory	169
B.6. DC Motor Remote Laboratory	170

Abbreviations

BS	B ooking S ystem
DAQ	D ata A cquisition
DC	D irect C urrent
DL	D igital L ibrary
EJS	E asy J ava S imulations
GPL	G raphical P rogramming L anguage
GUI	G raphical U ser I nterface
IP	I nternet P rotocol
ISE	I nteractive S creen E xperiment
JIL	J ava I nternet L abview
JIM	J ava I nternet M atlab
LMS	L earning M anagement S ystem
MOODLE	M odule O bject- O riented D ynamic L earning E nvironment
OS	O perating S ystem

P2P	Peer-to-Peer
RL	Remote Laboratory
SARLAB	Sistema de Acceso a Recursos de Laboratorios (System for Accessing Laboratories Resources)
TCP	Transmission Control Protocol
UNED	Universidad Nacional de Educación a Distancia (Spanish Open University)
VI	Virtual Instrument
VL	Virtual Laboratory
VRL	Virtual and/or Remote Laboratory

Symbols

A	section of the spring	m^2
E	modulus of elasticity	N/m^2
F	force	N
f	focal length of a lens	m
k	elastic constant of the springs	N/m
L	distance between the object and image planes	m
l	natural length of the springs	m
n_i	refractive index of the incidence media	
n_t	refractive index of the refractive media	
s	distance from the object plane to the lens	m
s'	distance from the image plane to the lens	m
x	stretch of the spring	m
θ_i	angle of incidence	rad

θ_l	limit angle	rad
θ_r	angle of reflection	rad
θ_t	angle of transmission (or refraction)	rad

*Dedicated to my family and friends; the present and the
past ones. . .*

Chapter 1

Introduction

This chapter presents the work developed in this thesis. It offers a brief description of either the components that play a main role in the project (virtual and remote laboratories and learning management systems) and the Web experimentation portal created in this work. A study and analysis of the state of the art concerning the previous elements is then presented. Finally, there is a discussion of the goals set for this work and an explanation of the structure this thesis follows.

1.1. Project Description

1.1.1. Background and General View

It is well known that education in science and engineering disciplines requires practical experimentation. While this has been carried out in laboratories or in field for ages, the use of computers has recently introduced new approaches. Simulated and remote laboratories are one of the new possibilities technology offers to cover the experimentation need in scientific and technical education programs.

A simulated or *virtual laboratory* (VL) corresponds to a computer application providing a graphical representation not only of the objects under experimentation but also of the instruments which should be used to perform the experiment in the real world. On a *remote laboratory* (RL), the control and observation of the real physical instruments and objects under experimentation is mediated through a computer while an adequate remote access to that computer is provided through a specific communication network [1].

Web-based laboratories is a term that is commonly used to refer to virtual and/or remote laboratories (VRLs) without distinction. VRLs are used mainly in two different scenarios. The first one involves the traditional face-to-face university. In these cases, Web-based laboratories are used as a complement for their hands-on laboratories practices, although this is still not as popular as it should be [2]. Thanks to their use, while students still have to perform experimental activities in a real lab, they can also spend more time experimenting from their own homes. The second scenario is offered by the distance university. In these cases, VRLs are used as a substitute to traditional labs, since hands-on experimentation in a real laboratory is not always an easy possibility in these situations.

While some education courses only offer one of its parts (either virtual or remote), it is commonly accepted that VRLs work better when both of them (the simulated experiment and its real -remotely controlled- counterpart) are offered. In fact, each of these parts can serve for different purposes. Anyhow, Web-based laboratories, in any of their forms, are nowadays quite well established in several scientific and technical disciplines since they help to illustrate scientific phenomena that require costly or difficult-to-assemble equipment [3]. Multiple examples of this can be found in [4, 5].

Although simulations can be very useful, real laboratories can not be replaced just by these tools. This is especially true in some scientific fields such as physics or

control engineering, where the actual behavior and response of the real elements used in the experiments are crucial [6]. Since they use real equipment, remote laboratories (which provide real laboratory facilities to students such as the simulation software does [7]) may constitute a better substitute or complement to real hands-on experiments. However, virtual laboratories (VLs) may still be used to serve as initial experimentation facilities for: 1) providing a first contact with the phenomena under study, 2) familiarize the student with the use of virtual and remote applications and 3) offering theory-based results of the experiments. Also, since remote laboratories provide real experimental results and virtual laboratories offer theoretical ones, students can compare reality with theory, which is one of the fundamental steps of the scientific method. Therefore, a Web-based laboratory consisting of both a simulation and the remote experiment, offers a more complete learning experience to the students.

Even though VRLs have proven very useful, they can still be complemented by other tools or resources that can enhance their distribution, use, functionality, pedagogical value and/or evaluation. This is where *Learning Management Systems* (LMS) play their role. A LMS is a software application or Web-based technology used to plan, implement, and assess a specific learning process. Using a LMS allows the administration, documentation, tracking, and reporting of e-learning programs. When these tools are combined with Web-based laboratories, instructors can offer a more complete e-learning program since it can be based on: 1) experimentation (thanks to the use of the VRLs) and 2) theory documentation provision, social interactivity and easy management (thanks to the use of the LMS).

This thesis proposes a structured framework for creating Web experimentation portals. The presented framework uses a set of tools that allows creating VRLs, creating Web portals based on a LMS, and embedding the first ones into the second ones. The framework also details several network and communication solutions and

strategies as well as suggestions for documentation provision and other pedagogical issues.

As an example of application of the proposed framework, UNEDLabs is presented. The UNEDLabs project constitutes a distributed network of virtual and remote laboratories, or Web-based labs, for engineering and science higher education via the Internet. It uses Moodle (a modern, free, and open source LMS) to, among other features, provide student/professor communication links and to make references, theory and exercises accessible to students. This network is already distributed between different Spanish faculties and universities. UNEDLabs offers students the possibility of performing hands-on experiences in different fields of physics and in control engineering in two ways: a real but remote one and a simulated one. The access to the VRLs (which are created using *Easy Java Simulations*, EJS) is granted to students by a Java enabled browser and an Internet connection. These two common tools let students to make practical experiences anytime, anywhere.

1.1.2. Some Considerations About Virtual Laboratories

The methodologies and/or styles of teaching are changing almost at the same rate than technological advances. As a consequence, the teaching community is each time more concerned about looking for new tools to support their educational tasks. In this context, VLs have been widely used to cope with this challenge, [8].

Digital media (such as simulations, videos, interactive screen experiments or remote laboratories) can positively impact students knowledge, skills and attitudes, [9]. In particular, simulations have evolved into interactive *Graphical User Interfaces* (GUIs) for student exploration, [10]. By means of these tools, students are not only able to observe the behavior and evolution of the simulated systems

but also to manipulate their parameters. In terms of pedagogical advantages, this interactivity is translated into new possibilities for students to understand the underlying concepts through inferring and predicting possible outcomes.

Simulations used in physics teaching -as well as in other disciplines- are computer programs (which can be made in Java, Flash, Bullet, PhysX, etc.) with an implicit model of the behavior of a physical system that allow students to explore and to visualize graphic representations, [11]. The possibility to carry out the experiments without the equipment constraints and the ease to perform the experimental activities are two of the main benefits from using simulations. Thanks to them, students can:

- Interact with the system by changing parameters.
- Observe the results of their manipulation.
- Make a great number of simulations without any restriction.
- Study phenomena which would not be possible to investigate in a traditional hands-on laboratory.

VLs are a particular approach of simulations to hands-on laboratories. They are computer based simulations which focus on presenting both similar views and similar ways of work to their traditional, hands-on, counterparts.

Figure 1.1 illustrates the architecture used for deploying VLs through the Web. In these cases, a unique computer with web services capabilities is used to host the VLs applications and serve them to the clients.

1.1.3. Some Considerations About Remote Laboratories

While VLs are computer simulations of a particular experiment, RLs imply the remote control of appliances that make up real experiments which can be

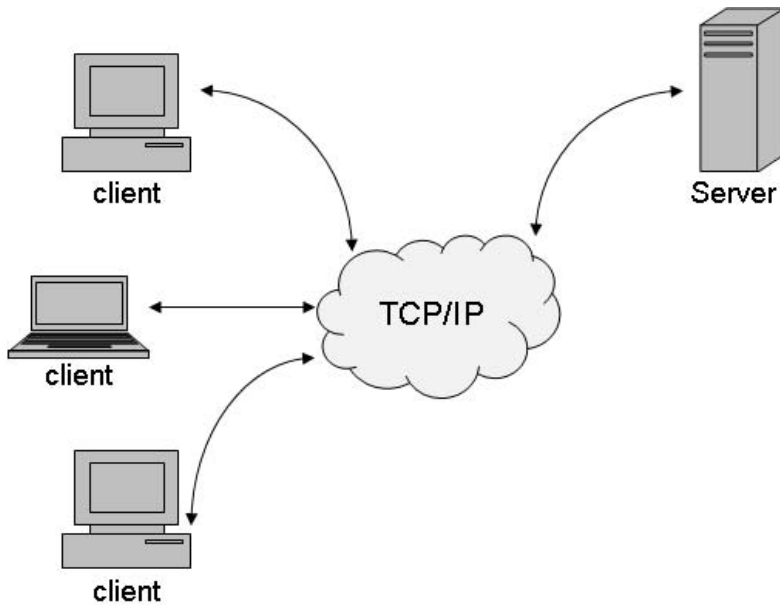


FIGURE 1.1: Typical virtual laboratories deploying architecture.

performed live remotely. The use of this kind of laboratories provides students with real experimental data to work with and to analyze for their laboratory sessions, which is an advantage when compared to VLs.

Now, when compared to traditional laboratories, RLs also present some other advantages, in spite of also presenting some disadvantages. The most obvious advantage is their improved accessibility with respect to their traditional counterparts. Since RLs can be used from students' homes, anybody can access to them, even handicapped people. Another of their advantages is their increased availability thanks to their capability to operate 24 hours a day without constant supervision. Finally, the safety of the laboratory devices is also guaranteed due to the software-controlled remote use of these resources.

RLs basically consist of two main parts: the experimental setup (real system + controller), and the software that controls the instruments of that setup (in the laboratory computer). The first one needs a design that fits the experiment

requirements and the appropriate physical devices: relays, DC or stepper motors, lasers, illumination conditions, force or contact sensors, etc. For the second part, several different programs can also be chosen; for example Matlab or LabView.

Even though the use of RLs for education purposes is increasing, they are not so widely used for research activities. However, since RLs can share complex equipment among different researchers, they could be used with this purpose more intensively [12].

Since a remote experiment can only be used by one person at a time, there must be a *booking system* (BS) which manages its use and avoids the problem of two students trying to access to the same RL at the same time. The problem of the BS is more complex than it might look at first since different students may be accessing from different parts of the world, with different time zones. Also, this system should prevent massive bookings from one only users, which could result in avoiding other users to get a booking for working with the RL. Finally, instructors usually want to be able to temporally grant or remove booking permissions to some or all their students in order to have more control over the real equipment.

Figure 1.2 illustrates the proposed architecture for the RLs with an example of just one RL. In these cases, two possible configurations are usually found. In the first one, the computer with the BS is used only for this purpose while the PC controlling the RL equipment is also in charge of serving the RL client software application. This is a complex solution since it requires each of these computers to have web services capabilities. In the second one, more simple and effective, the server with the BS also hosts and serves the RL client application while the PC controlling the RL hardware does not have to be in charge of any other task.

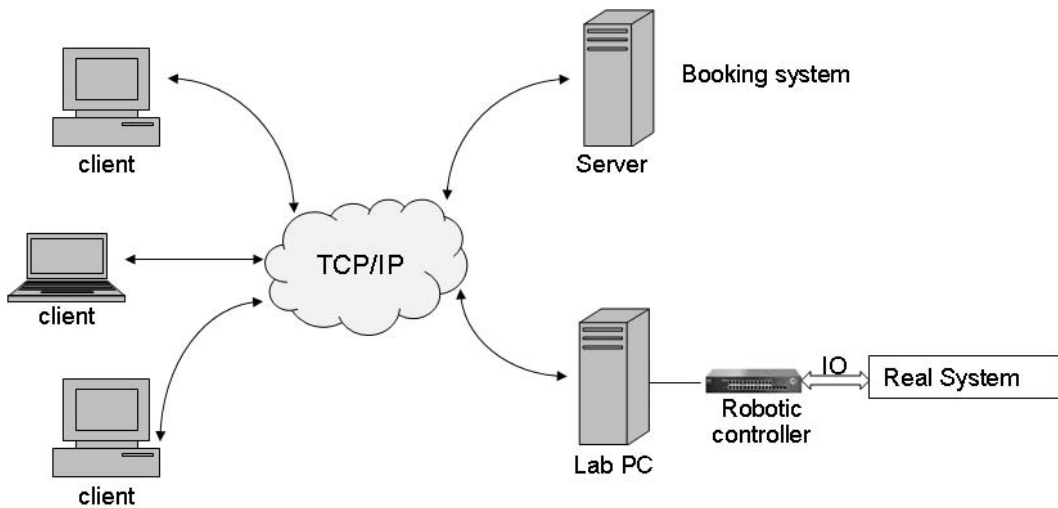


FIGURE 1.2: Typical remote laboratory architecture.

1.1.4. Some Considerations About Learning Management Systems

A LMS is a software package that enables the management and delivery of learning content and resources to students. Most LMS systems are Web-based to facilitate "anytime, anywhere" access to learning content and administration.

Typically, a LMS provides an instructor with a way to create and deliver content, monitor student participation, and assess student performance. It may also provide students with the ability to use interactive features such as threaded discussions, video conferencing, and discussion forums. Other important features of LMS are allowing for student registration, the delivery and tracking of e-learning courses and content, and testing.

In the most comprehensive of LMSs, there are also available tools such as competency management, skills-gap analysis, succession planning, certifications, virtual live classes, and resource allocation. Most systems allow for learner self-service, facilitating self-enrollment, and access to courses.

1.1.5. UNEDLabs

UNEDLabs is a network of VRLs which are integrated into a LMS and that uses a booking system for managing the access to the RLs. It uses and applies all the ideas, software and architecture analyzed, proposed and/or developed in this work, which form a complete and robust framework for developing these kind of Web experimentation networks.

Because digital media are only as effective as the pedagogical approach in their use, pedagogy and digital media have to be inextricably linked, [13]. With this purpose, the VLs at the UNEDLabs portal of the Spanish Open University (UNED) are designed based on real experiments which are also intended to be available for remote experimentation.

In spite of the advances in computer based simulations, few works have been made to study the impact of different implementation methods of digital learning objects, which is a fundamental topic that teachers have to cope with, [14]. However, some studies state that an educational simulation should serve to motivate learners to engage in problem solving, hypothesis testing, experimental learning, schema construction and development of mental models, [15]. For this reason, all the VLs at UNEDLabs present a description of the phenomena under study and of the didactic setup for remote experimentation, as well as the simulation source code.

VRLs and LMSs offer different but fundamental educational tools to both teachers and students. However, although these resources are complementary (and not mutually exclusive), the integration between them is still an open issue that must be addressed. Therefore, an e-learning program should offer both kinds of tools to be considered a complete experience for students.

Not only all the VRLs in UNEDLabs are added and integrated into the Moodle Web portal in a very easy and natural way, but they also acquire special functionalities they lack when used outside this LMS. One of these functionalities is the capability of saving and/or loading files to/from a private files repository in Moodle. Another one is the possibility to create collaborative experimental sessions with other users who are enrolled to the course the VRL belongs to. The BS for the RLs is also integrated in the LMS thanks to the tools developed in this work for the proposed framework and UNEDLabs makes use of it. Finally, VRLs in Moodle can be administrated exactly in the same way as any other Moodle resource or activity, meaning they can be updated or deleted; their access can be restricted to a certain group of users or to users that have previously fulfilled some steps or conditions; security copies are automatically performed during system backups, etc.

On the one hand, figure 1.1 showed the typical architecture for deploying VRLs via the internet by means of a Web server. On the other hand, figure 1.2 illustrated the architecture for deploying one single RL and managing its access thanks to the use of a BS.

This work proposes a framework for making available both virtual and remote labs (with booking capabilities) embedded into a LMS. Therefore, a combination of the two previous solutions must be applied. Figure 1.3 shows the complete architecture of this solution, in which only one server is used for hosting the LMS, the VRLs software applications and the BS, and N computers control the N real systems by means of the appropriate hardware controllers.

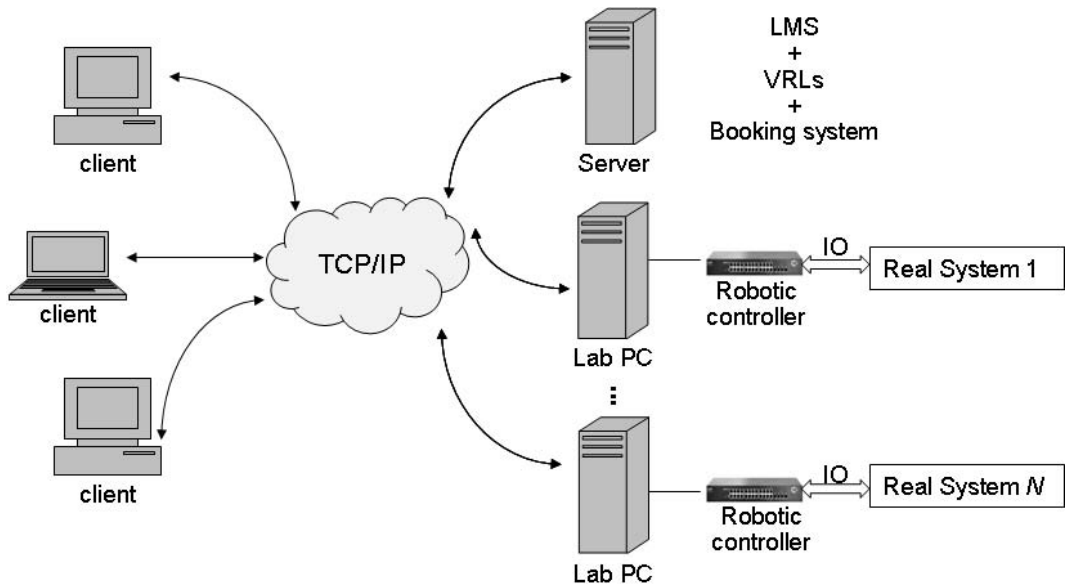


FIGURE 1.3: Proposed architecture for deploying VRLs and integrating them into a LMS with a booking system.

1.2. State of the Art

1.2.1. Other Solutions for Providing Experimentation Facilities in Distance Education Paradigms

Apart from VLs and RLs, there are also other approaches for the distance experimentation problem in science and engineering distance learning courses.

One of these solutions are the interactive screen experiments (ISEs), [16]. These tools are a highly interactive nonlinear movies that run under user control by clicking and dragging on the experimental equipment displayed in the movies. These movies can be played in an ordinary web browser with a free Shockwave¹

¹Adobe Shockwave is a multimedia player program that allows Adobe Director applications to be published on the Internet and viewed in a web browser on any computer which has the Shockwave plug-in installed.

or Flash² plug-in in a similar way as a Java applet does. An ISE consists of many hundreds of photographs of an actual laboratory experiment designed to capture all possible states of the apparatus. It is a kind of highly interactive movie where the student can control the development of the plot. ISEs are real experiments showing real phenomena, not simulations. When the student runs an ISE the effect is as similar as possible to running the real experiment by remote control. The difference for the laboratory provider is that there is no limit to the number of students who may simultaneously run the experiment and the real equipment itself is used only once. However, ISEs present a big drawback when compared to RLs: since the experimental data in ISEs are recorded (and the real equipment is never used again), students always obtain exactly the same results when a particular state of the experiment has been set. On the other hand, RLs may return different values when measures are taken even in the same conditions because the data are generated on live.

Another solution, applied by some open universities, is to send each student the required material to perform the experiments and then ask them to use it to complete some experimental activities. Although this solves the problem of removing traditional experimentation in distance education courses, this solution can only be applied for a few particular cases in which the required material is not very expensive, heavy, bulky or fragile, due to shipment constraints.

Other resources such as videos and interactive videos should not be considered as alternatives to ISEs, VLs and RLs but as a complement to them. While the former may serve as a tool used for providing theory lessons, the latter try to cover the experimental part of the subjects, and not the theoretical one.

²Adobe Flash is a multimedia platform used to add animation, video, and interactivity to Web pages.

1.2.2. Virtual and Remote Laboratories in Physics

Physics simulations have experienced a huge increase in number and quality thanks to Java applets mainly. In [17, 18], hundreds of simulated experiments and processes in different fields of physics are presented (all of them developed using EJS). Some of them, and many other simulations, are available for free at [19]. However, real experiments are not specifically considered in these works. Also, in spite of the huge number and the outstanding quality of the available simulations, the web page in [19] is conceived as a simple repository of physics simulations, and not as a web-enabled environment for virtual experimentation.

Other recent works focus on the introduction of computers into physics laboratories for data acquisition and analysis [20, 21] but not for distance learning purposes. Finally, most of the remote laboratories for physics-related experimentation are individual experiments [22, 23] or are limited to a particular field, such as optics [4, 24] or electronics, [25, 26]. Besides, none of the latter works offer the simulated counterparts of the remote experiments or a web-enabled environment supported by LMS facilities. Finally, just in [27, 28] each experiment is presented in a similar way to a traditional student lab: introduction, theory, exercises and problems, laboratory activities, analysis, discussion, and reference material.

It is also important to mention the lack of RLs dedicated to the topic of physics. In [29], sixty papers about hands-on labs, VLs and RLs are analyzed. While twenty of them were dedicated or related to RLs, only one of them was about physics but fifteen were about engineering (electrical engineering, mainly). Also, in [30], forty-two works about the same topic are studied and while eight covered the physics field, engineering was again far ahead with twenty-six papers dedicated to that discipline. The work in [30] also shows that Java is the most used program to create the GUI of the middleware through which information is

exchanged between the local and the remote computers. As UNEDLabs uses the same technology (Java), the framework presented in this thesis is as reusable as possible for many of those works. Figure 1.4 illustrates these numbers.

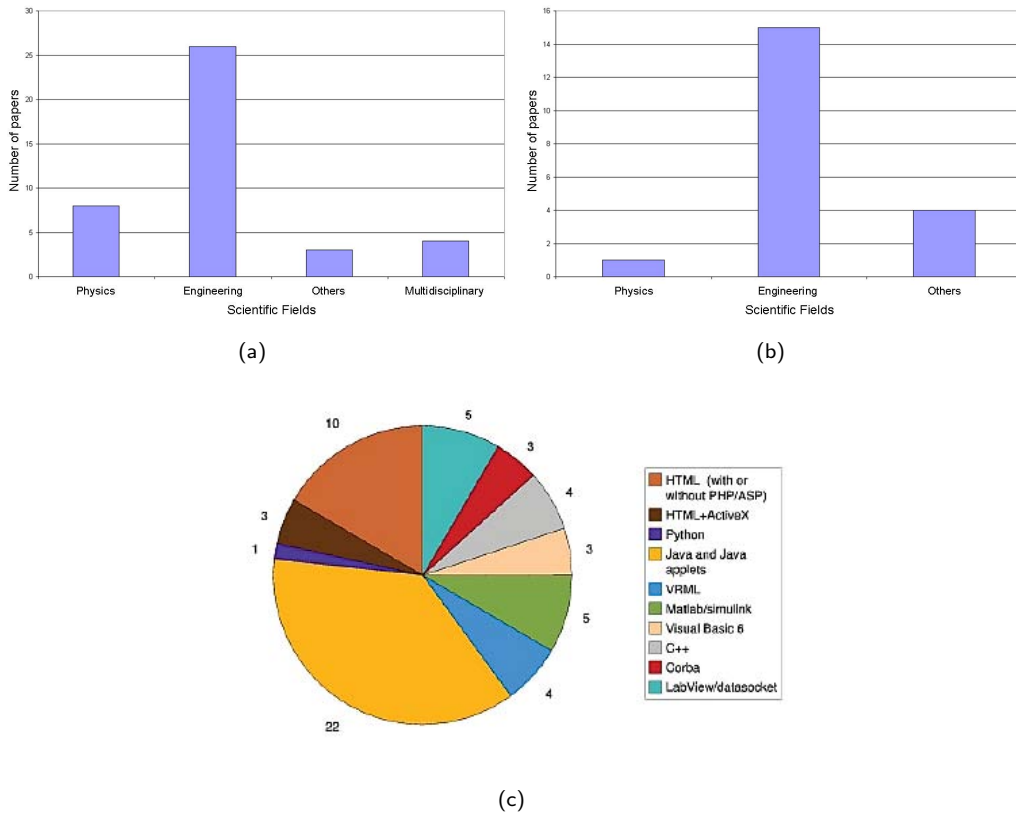


FIGURE 1.4: Number of publications related to RLS dedicated to engineering and physics according to [29] (a) and [30] (b). Figure (c) taken from [30]: a comparison between the technologies used for the RLS.

1.2.3. Virtual and Remote Laboratories in Engineering

As already said, [29] analyzed sixty papers and, regarding engineering disciplines, fifteen of them were dedicated or related to VLS while twenty were to RLS (see figure 1.4(a)). In [30], twenty-six papers dedicated to RLS in engineering were

found (figure 1.4(a)). As a result, we can conclude that both VLs and RLs are widely used in engineering courses.

Other works, not analyzed in [29] nor in [30], such as [31], present a solution for operating remote laboratories from a bootable device (a CD, for example) that allows selecting and controlling a wide variety of different experimental systems. While this avoids the problem of integrating the RLs into a web page, it also limits their deployment.

1.2.4. Conclusions About the State of the Art in Virtual and Remote Laboratories

The conclusions in [29] focus on stating that most of the laboratory articles were engineering-related and on discussing the effectiveness of VRLs. On the other hand, the work in [30] concludes, from their literature review, that there were still "four major issues for the leverage of remote laboratories. These are reusability, interoperability, collaborativeness and convergence with Learning Management Systems". Other works such as [32, 33] point out the same issues as current trends, not yet completely solved, in VRLs development.

While the last of these four issues (the one regarding integration with LMS) has been already solved somehow, when working under certain specific circumstances [34], the other three are still fully open. More specifically, [34] presents a solution for embedding VRLs developed using the iLab solution ([35, 36]) into SCORM packages, which are easily integrated into several LMSs but do not support interaction between the VRLs and the LMS itself. The present work not only offers easy integration of VRLs, based on a different solution and technology (EJS), into a LMS (in such a way they can "communicate" or interact) but also a solution to the reusability and collaborativeness issues. Another similar, but more limited,

work is [37], where an integration of LogicSim applets into Moodle is presented. It is more limited due to two main reasons: 1) LogicSim applets only serve for logic circuit VLS and 2) the additional features obtained by means of the integration of the applet with the LMS are very reduced.

Finally, [33], another literature review about VRLs, concludes that: 1) "There are lot of virtual and remote laboratories developed with LabVIEW, Java Applet and Flash" and 2) "To develop a remotely accessible laboratory, the developers have to master computer hardware and software, data digitization and collection, data transmission and visualization, and network. An engineering education laboratory developer usually has expertise in their research field, but not necessarily in remote laboratory development. The development of a unified user friendly remote laboratory publishing tool for laboratory developer is in great demand". The first conclusion supports the information provided in figure 1.4(a) (extracted from [30]), which means that this work is as reusable as possible since the proposed solution is applicable in VRLs developed with Easy Java Simulations. The second conclusion is a key point and this work is extremely focused in making life easy to instructors who want to develop their own VRLs (or reuse already existing ones) for their courses.

1.2.5. Analysis of Existent Learning Management Systems

Lots of (open source code or not) LMS are available for free. Each LMS offer their own advantages such as a more modern and user-friendly view, new communications options, better students assessments and survey tools, etc. Also, students might be organized in different ways depending on the LMS. Some of the LMS considered before choosing Moodle were:

- *Atutor*. This LMS is designed for adaptability to any of several teaching and learning scenarios. There are four main areas that reflect this design principle: themes, privileges, tool modules, and groups. The ATutor theme system allows administrators to easily customize the look and layout of the system to their particular needs. Themes are used to give ATutor a new look, to give categories of courses their own look, or to provide multiple versions of ATutor on a single system, from which users could choose one as a preference setting. The privilege system allows instructors to assign tool management privileges to particular members of a course. Instructors may create assistants or course tutors that had limited control over any of the authoring or management tools.
- *Claroline*. Each course space of this platform provides a list of tools enabling the teacher to: write a course description, publish documents in any format (text, pdf, html, video...), administer public and private forums, create groups of students, manage an agenda with tasks and deadlines, publish announcements, see the statistics of the users activity, or use the wiki to write collaborative documents.
- *Dokeos*. Some of its main tools are: sharable content object reference model courses authoring, rapid learning, templates-based document production, tests (with multiple choices, fill-in-the-blanks, matching, open questions, or hotspots types), forums, chats, groups, surveys, lightweight directory access protocol and OpenID authentication, gradebook, reservations, and users sessions.
- *Ilias*. This one offers standardized tools and templates for the learning and working process including integrated navigation and administration. Groups in ILIAS allow cooperative learning and working on the platform without

additional tools. Learning groups, working groups or groups for certain fields of interest could be created. Groups can use all ILIAS tools like wiki, forums or file sharing. It also offers multiple ways to deliver learning content. All types of document files can be uploaded and it supports standard ways of communication as chats, forums and mails. Beside RSS (Really Simple Syndication)¹ support, ILIAS offers the ability to manage podcasts. In ILIAS surveys can be used to easily collect information from a large number of users, for example to evaluate courses or other events. Finally, ILIAS test and assessment supports multiple choice, single choice, allocation questions, cloze questions (free text, select box), ordering, matching, hot spot and more question types.

- *Sakai*. This LMS includes many of the features common to course management systems, including document distribution, a gradebook, discussion, live chat, assignment uploads, and online testing. Sakai includes the ability to change the settings of all the tools based on roles, changing what the system permits different users to do with each tool. It also includes a wiki, mailing list distribution and archiving, and an RSS reader. The core tools can be augmented with tools designed for a particular application of Sakai. Examples might include sites for collaborative projects, teaching and portfolios.
- *Moodle*. Moodle's interoperability features include: authentication, enrollment, using IMS Enterprise² among other standard methods, or by direct interaction with an external database, and quizzes with several types of questions: calculated, description, essay, matching, embedded answers, multiple

¹Used to inform about web updates.

²An international standard for XML files format. It can be used to specify enrolments/unenrolments in courses, as well as course information and user information.

choice, short answer, numerical, random short-answer matching, or true/false. Also, syndication is possible using RSS or Atom newsfeeds.

The LMS Review Subgroup at Duke University reviewed LMS reports from eight institutions in order to identify trends and evaluation methods from different institutional efforts [38]. From these eight cases, six institutions considered Moodle as one of the possible LMS they should adopt. From these six reports in which Moodle was considered, four decided that Moodle was the LMS they should use, while one concluded Sakai 3.0 was the better choice and the last one opted for either Blackboard or Desire2Learn.

The Athabasca University, which is Canada's Open University, evaluated several LMS for use by the university [39]. WebCt, LotusNotes, and Moodle were considered. Moodle was chosen with 11 first place ratings and with only one third place rating. The first place preferences within individual criteria made by this university shown the following: WebCT 6; LotusNotes 7; and Moodle 58. Since the end of 2005, the Open University of the UK also uses Moodle.

In October 2010, the AUT University commissioned a literature review of LMS. The report [40] summed up the studies and considerations about LMS of seven other universities, all of them also from New Zealand or Australia. Five of these institutions decided to adopt Moodle while two of them opted for Blackboard.

In terms of usage and according to [41], in the higher education market as of fall 2011, Blackboard, Moodle and Desire2Learn are the two largest providers. It is important to highlight that while the first and the third are not free tools, Moodle is. An article published by eLearning Guild in 2009, [42], showed that Moodle was ranked as the #1 LMS product among eLearning Guild members with over 20.1% of respondents selecting it as their primary LMS. This survey measures use of over 100 professionally-developed LMS products and excludes in-house created systems.

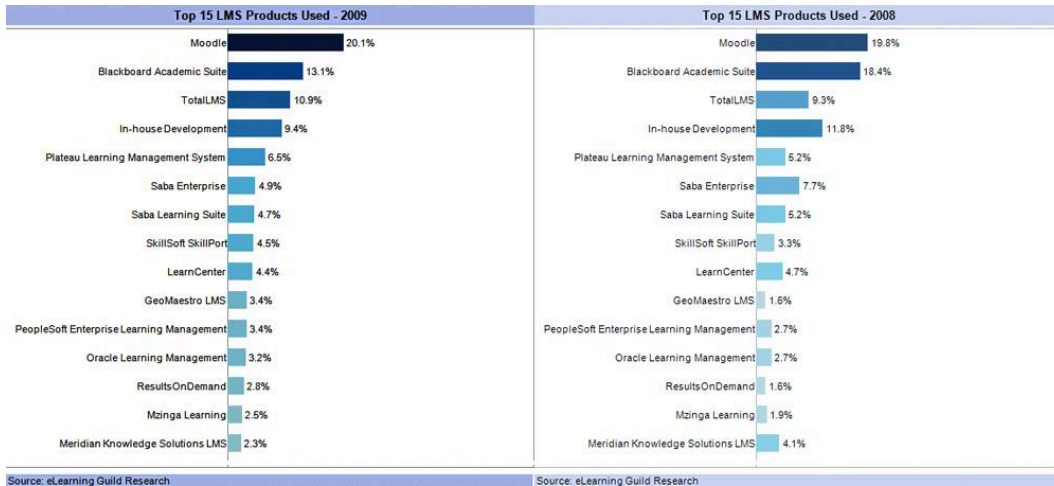


FIGURE 1.5: Comparison of LMS product usage for the May 2007-2008 vs. May 2008-2009 periods.

Figure 1.5 shows that in May 2008, Blackboard and Moodle were neck and neck for the dominant position among eLearning Guild members for their LMS product usage. However, market share among eLearning Guild members for Blackboard dropped from 18.4% in 2008 to 13.1% in 2009 while for Moodle it increased from 19.8% to 20.1%.

Figure 1.6 shows that Moodle is usually preferred for small or medium scale integrations while TotalLMS is the preferred for large scale ones.

In October 2012, over 500 learning professionals from around the world shared their top ten tools for learning recently in the 6th Annual Survey of Learning Tools, organized by the Centre for Learning & Performance Technologies (C4LPT). Moodle came 11th in the list of Top 100 Tools for Learning 2012 [43], down a little from 2011, when it was 8th. Moreover, Moodle came first in the Course Management System category.

Finally, as of fall 2012, Moodle has more than 63 million users according to its web page, which makes it the most used LMS around the world.

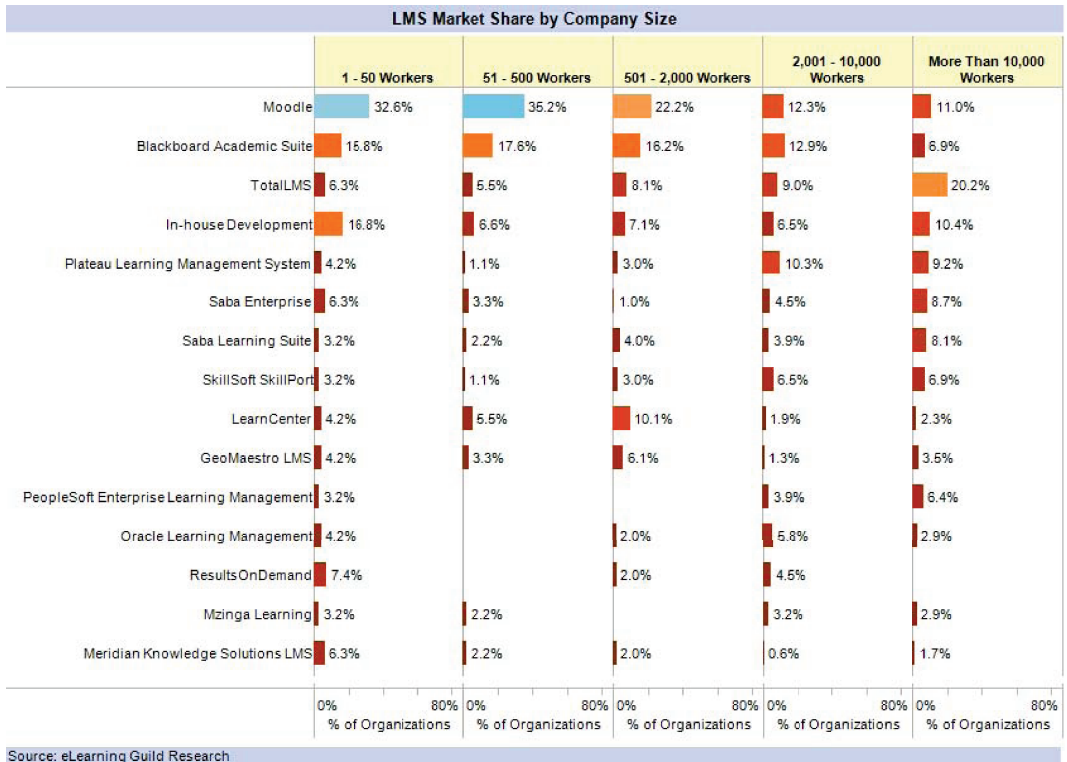


FIGURE 1.6: LMS market share trends.

Considering all the previous information, Moodle seems like the right choice when developing new tools for VRLs integration into LMS. This election not only means working with a free and open source tool which is up-to-date and that offers, at least, similar features and advantages as the other LMS, but also that a solution based in this system will be usable by a great number of people.

1.2.6. UNEDLabs' Ancestors

UNEDLabs uses the same structure as AutomatL@bs [44]. AutomatL@bs is a network of web-labs for learning/teaching of control engineering, which has been operative for five years now, offering experiments such as the three-tank system [45], the DC motor [46] or a heatflow system. As part of the UNEDLabs project,

several web-labs were developed and are already available (a motorized optical bench to determinate the focal length of a thin lens [47], a light reflection/refraction experiment [48], a Hooke's law experiment [49]) or are still in development (a rigid pendulum, a radiation experiment, a XYZ table to measure the distribution of potential over a resistive sheet of paper with different electrostatic field configurations, etc). While these experiments were initially planned to form part of the FisL@bs network [50], the present work finally gathered together AutomatL@bs and FisL@bs into one only network: UNEDLabs.

Each of the previous VRLs requires different materials and hardware tools to prepare the remote parts and allow them to be remotely controlled. In order to reduce costs, simplify the construction, and provide a user-friendly impression to students, some of the previous experiments (for instance, the Hooke's law experiment and the XYZ table) were built using LEGO Mindstorms pieces. Others setups are being made up using aluminum pieces, stepper motors and controllers, force sensors, and so on. Either way, all the hardware is always controlled using real-time virtual instruments previously programmed in Matlab or LabView, a graphical programming language specifically designed for developing instrumentation, diagnostics, and data acquisition systems [51]. LEGO robot kits have been used in science and engineering projects such as [52, 53], as well as in teaching/learning courses [54–57] when a limited budget is one of the restrictions to design the experimental setup. Another experiment currently available corresponds to Snell's law where other materials were used for its assembly.

In AutomatL@bs the GUI to remotely experiment with these laboratories was an applet created with Easy Java Simulations and the same solution was applied for FisL@bs. EJS is an authoring tool written in Java that helps to create interactive simulations in Java, mainly for teaching and learning purposes [58, 59]. The virtual experiments of each remote experiment at these portals were also programmed

using EJS. This tool was designed to work at a high conceptual level, letting programmers concentrate most of their time and efforts on the scientific aspects of the simulation. It is the computer that automatically performs all the other necessary but easily automated tasks. Basically, there are two main parts in EJS: the Model and the View. While the first one lets developers introduce the laws, equations, and programs that rule the simulated experiment, the second one offers the GUI to interact with the simulation, that is, the model. Once the development work is completed, EJS produces an applet as the final result. In this applet, the Model part is transparent to the user, who only needs the View to perform the activities related to the simulation.

AutomatL@bs and FisL@bs used eMersion [60], a Web-based learning environment which facilitates the deployment of pedagogical scenarios and learning resources for Web-based experimentation in education, for storing the EJS applets and publishing them on the Internet. Finally, in order to simplify the communication between the EJS-based applets and the LabView virtual instruments, a middleware layer that is based on a Java-Internet-LabView server was used. This middleware (composed of a stand-alone application called JIL server [61] and a Java library) links the LabView in the server side (the real experiments) with the EJS-based applets (located in the students' computer) by TCP/IP channels. The reason of such layer was twofold: a) reduce the Internet traffic load between applets and Matlab or LabView programs, and b) let developers connect Java variables with either LabView controllers and indicators or Matlab variables by simple point-and-click actions.

Finally, AutomatL@bs and FisL@bs used a custom booking system that was specially created to be used for the access to remote laboratories management problem. This tool worked as an independent system which had to be integrated with the rest of elements in these projects (eMersion, EJS, etc.). Although this

could be done, that integration never felt natural: the BS required independent databases, no interface was available to manage the students' booking rights, etc.

UNEDLabs inherits some of the tools Automatl@bs and FisL@bs used without touching one single thing about them. This is the case of LabView and/or Matlab. Other tools have been adapted or modified to obtain new features, make them more general or just easy their use, such as EJS and the BS. Finally, the old LMS (eMersion) has been completely replaced by a new one (Moodle). In the end, the framework proposed in this work, and applied to create UNEDLabs, offers an easier and more compact and complete way to deploy VRLs created with EJS than Automatl@bs and FisL@bs did. Moreover, UNEDLabs was also qualified to reuse, with almost no additional effort, the VRLs already developed for the previous experimentation portals (see Appendix B).

1.2.7. Other Networks of Web-based Laboratories

Moving on to the precise topic this work is about (creating web environments with all the required features for allowing virtual and remote experimentation in engineering and/or science courses), there are some other related projects which need to be mentioned. While there are not many works around there focusing on the methodology and process for creating these environments, there are some works that actually created networks of Web-based laboratories or, at least, a collection of VRLs shared thanks to the use of the Internet.

In [62], seventeen remote laboratories are offered. This is a real network of RLs (no VLs are offered) since their labs are scattered through different universities. Fourteen of these RLs are about physics, two about robots and one is a toll system. An introduction, a description of the experimental setup and some theory are given and the laboratory activities and analysis are detailed. The user interface for these

RLs is HTML and they include (when necessary) video feedback from a webcam. However, no communication channels between students and professors are given (except for the e-mail) and, what is more important, the web environment does not provide a tool for saving and storing the data obtained from the experimentation. No collaboration is considered and there is no LMS to enhance the pedagogical process. Finally, there is no BS to prevent students trying to access the same RL at the same time.

Nine RLs are offered in [63]: eight are related to physics topics and one to control engineering. As in [62], RLs are well documented with introduction, theory and so on. Moreover, the work in [64], related to this project, presents a set of hardware tools to standardize the creation of RLs and make them reusable. In this project, Java is used for the UI of the RL, instead of pure HTML and, by its use, these labs offer to their users the possibility to save the experimental data. However, these data files are saved into the users' computers and not into the server laboratory, where users could access again these files. On the other hand, the same limitations described for [62] about the integrated use of a LMS, and the need of a BS can be applied here.

Purdue University used to offer a RL with different experiments using pendulums but a couple of years ago it closed the access to them, unlike the two previous websites, which still offer free access [65].

Lila [66] is probably the most complete and ambitious project from all the above since it aims to build a portal which grants the access to virtual labs and remote experiments. It includes services like: BS, connection to library resources, tutoring system and 3D-environment for online collaboration. Moreover LiLa creates an organizational framework for the exchange of experiments between institutions and for the access to experimental setups, which is also one of the objectives of the present work.

iLab [35] is also a network of RLs that offers more than twenty of these experiments. Most of them are about electronics and telecommunications but there are also a few about physics or control engineering. Exactly the same limitations stated for [62] can be applied here. However, as well as Lila, this project takes into consideration the problem of facilitating the development and sharing of RLs and integrating them in an easy way into a web environment.

None of the latter web pages seem to offer a group of experiments which follow a university's organized course (for example, a course for students in the first year of their physics grade). Besides, none of them incorporates a popular LMS for their users which would allow to add important features such as a login system for tracking and reporting, data files management system or communication channels between students and/or professors. Only one of them seem to offer a booking systems and there are no options of synchronous collaborative work in any of the previous projects.

1.3. Goals

The first and most important goal of this work is to present a framework that offers the architecture, tools and procedure for creating Web experimentation portals with collaborative virtual and remote laboratories (such as UNEDLabs) in an easy way. Instructors face extra work when they transform an existing local system into a Web-based environment as, even though they might be used to manage hardware and software in a local control system, new problems arise when making it accessible via the Internet. In particular, developers must create interactive GUIs for the systems to be deployed via the Internet in the form of dedicated plugins such as Java applets, Flash, ActiveX, etc., to integrate them into a LMS. The software tools developed in this work, along with other already existing tools

and the described architecture and procedure allow any instructor to create and deploy VRLs into a LMS without the need of much time or technical skills and/or knowledge about remote control systems, networks or the Internet. Moreover, instructors that follow this framework to embed their VRLs into the LMS will benefit for free from special features and options that would not be available when using VRLs and/or LMS separately.

Focusing on the experimentation advantages, while traditional laboratories may sometimes present a low ratio between their use and their costs, a remote access to these laboratories can increase their frequency of use. The creation of networks of schools or universities interested in the same shared experiments can contribute to this objective. Although the economical aspect is important, there are other benefits tied to these laboratories. The most obvious one is their improved accessibility with respect to their traditional counterparts: since they can be used from students' homes, anybody can access to them, even handicapped people. Another advantage of remote laboratories is their increased availability thanks to their capability to operate twenty four hours a day without constant supervision. Finally, the safety of the laboratory devices is also guaranteed due to the software-controlled remote use of these resources.

Therefore, UNEDLabs not only reduces the necessity of students to travel to real laboratories in distance education courses but it also expands the experimentation possibilities of traditional laboratories thanks to the sharing of network's resources as well as to their full time operating capacity. Since the didactic setups at UNEDLabs can be used by students from any university, laboratories implementation and maintenance costs are drastically reduced. Both the virtual and the remote laboratories provide students with some experience and skills they can take advantage of when facing related laboratory experimentation in person for the first time. Finally, since students perform each experiment twice (in simulation and in

remote), they can check the differences between the theoretical model and the real system and search for a plausible explanation of any observed discrepancy.

UNED offers Spanish students all over the world the possibility of following graduated and post-graduated studies at distance. For this reason, this university has a great need of distance education resources, methods, and technologies. This need forces UNED to firmly focus its research on the development, improvement, and exploitation of these issues. UNEDLabs is an ongoing UNED project addressed to distance learning in the context of the European Space for Higher Education. UNEDLabs is qualified for stimulating the practical individual work (for instance, by virtual and remote laboratories) but without disregarding the collaborative work (granted by the use of a LMS, which is something many other web experimentation portals lack).

UNEDLabs is intended to serve as a tool for students to perform the laboratory experiments of their physics and engineering courses after the education European convergence. This work presents three well-known experiments in physics which were specifically developed for UNEDLabs: Snell's law of refraction, Hooke's law and a practice about the focal length of thin lenses. Also, three more experiments (already available at AutomatL@bs) were adapted to UNEDLabs. They serve as an example for showing how easy to integrate already developed VRLs into Moodle is.

As for every other future web-lab at UNEDLabs, each of the previous experiments consist of four components: the surrounding LMS (Moodle), a Java applet, a didactic setup for the considered experiment, and a LabView or a Matlab application. Moodle provides student/professor communication links, makes references, theory and exercises accessible to students, and offers a file management system for the saved data obtained during the experimentation sessions. The EJS-made applet carries out two functions: 1) the control, visualization, and evolution of the

experiment in simulation mode, and 2) act as user interface for the experiment in remote mode. The third component is the real experiment setup while the LabView or Matlab application is a program used to control the physical devices.

1.4. Structure of the Document

Chapter 2 presents: 1) the framework established in this work for creating Web experimentation portals and 2) the network of VRLs that was created following this framework (UNEDLabs). Three Web-labs, that were developed and integrated into UNEDLabs during this work, following the proposed framework, are presented in Chapter 3. Chapter 4 details the use and features of the four Moodle plugins developed in this work as well as the modifications made to the EJS program. Chapter 5 shows the step-by-step process for creating and deploying VRLs using the proposed methodology. Finally, the conclusions and the further research lines are given in Chapter 6.

Regarding the appendices, Appendix A presents the software and hardware tools used to create the virtual and remote laboratories and other elements of UNEDLabs (such as the web portal, the web-based experimentation framework and the booking system). Appendix B lists the already existing Web-labs that were adapted to be used within this framework (and integrated into UNEDLabs), and shows how simple is to complete this process. Appendix C offers the developer documentation of the EJSApp plugins that is distributed along with this software.

- Chapter 1: Introduction.
- Chapter 2: Framework for Web Experimentation Portals. UNEDLabs.
- Chapter 3: New Web-Labs Developed (light in isotropic media, Hooke's law and non-linear springs, and focal length of thins lenses).

- Chapter 4: New Features Developed (EJSApp plugins and EJS modifications).
- Chapter 5: Step-By-Step Process for Creating and Deploying a VRL.
- Chapter 6: Conclusions and Future Directions.
- Appendix A: Tools for Developing Web Experimentation Portals .
- Appendix B: Adapted Web-Labs: Heatflow System, Three Coupled Tanks and Servo Motor.
- Appendix C: Developer Documentation of the EJSApp Plugins.

Chapter 2

Framework for Web

Experimentation Portals.

UNEDLabs

This chapter first explains the architecture used to develop the framework proposed in this thesis [67, 68] and then shows the Web experimentation portal (UNEDLabs, [69, 70]) that was created following this framework.

2.1. Architecture

2.1.1. General View

The main architecture, used at two different levels in this framework and applied in UNEDLabs, is based on the well-known client-server model. This architecture has some important advantages such as:

- It enables the distribution of the roles and responsibilities of a computing system between several independent computers that are known to each other

only through a network. This way, it is possible to replace, repair, upgrade, or even relocate a server while its clients remain both unaware and unaffected by that change.

- All data is stored on the servers, which generally have greater security controls than most clients. Servers have better control access mechanisms to resources to guarantee that only those clients with the appropriate permissions may access and change data.
- It functions with multiple different clients of different capabilities.

Figure 1.3 in Chapter 1 already illustrated the idea of the client-server architecture for deploying VRLs applications in combination with a LMS for providing the web interface and a support for the booking system that manages the connections to the hardware of the remote laboratories. This is the first level in which the client-server architecture is used within this framework and it is represented by the *clients* and the *Server PC* icons of figure 1.3. However, this part alone does not take into account how the communication between clients and the hardware of the RLs is established.

The second level in which this architecture is applied covers this case since it is related with the control of the remote laboratories. In figure 1.3, it is represented by the *clients* and the *Lab PC* PC icons. Following the structure of the web-based laboratories applied in AutomatL@bs and FisL@bs projects, UNEDLabs uses the same client-server architecture (where TCP/IP is the communication protocol for exchanging data between the clients and the server) for this issue. Figure 2.1 shows this communication architecture, where arrows going from the client-side sender to the command parser and from the server-side sender to the receiver represent the communication links. Within this framework, each *Lab PC* represented in figure 1.3 implements this scheme. Specific examples of this communication between

the RL client application and the RL real hardware are described in Chapter 3, in Sections 3.1, 3.2 and 3.3, with the "Law in isotropic media", "Hooke's law and non-linear springs" and "Focal length of thin lenses" experiments.

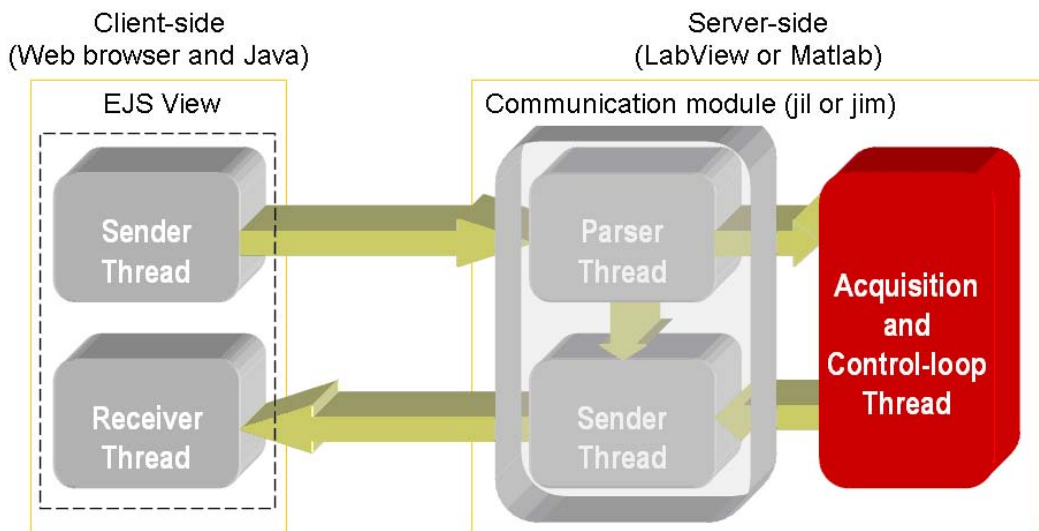


FIGURE 2.1: Client-server architecture of RLs.

Even though figure 2.1 considers a particular case in which LabView or Matlab is used in the server side to control the hardware, there are other possible options when using EJS for creating RLs. This program includes all the required tools for communicating online with Arduino boards (Phidgets support is also coming), which allows reading sensors as well as controlling actuators (such as motors) connected to these boards. Therefore, the proposed framework (based in the use of EJS) allows using a wide variety of hardware, for any element controllable with either LabView, Matlab or Arduino, so that it can be remotely accessed.

More information about the client-side and server-side implementations for RLs can be found in sections 2.1.3.1 and 2.1.3.2, respectively. Also, details about LabView, Matlab and Arduino are presented in sections A.1.2, A.1.3 and A.2.2, respectively.

On the other hand, a totally different architecture, that coexists in this framework with the previous one, is used for supporting one of the main features presented in this work: the collaborative experimental sessions. As it is detailed in section 4.2.2.2 (Chapter 4), when several users work in a synchronous collaborative session with the same VRL, a peer-to-peer (P2P) connection is established among them. This kind of architecture is a way to structure a distributed application so that it consists of many identical software modules, each module running on a different computer. The different software modules communicate with each other to complete the processing required for the completion of the distributed application. Therefore, it is ideal for this particular purpose of collaborative experimental work sessions. Figure 2.2 shows the P2P communication architecture in this framework when collaborative sessions are established.

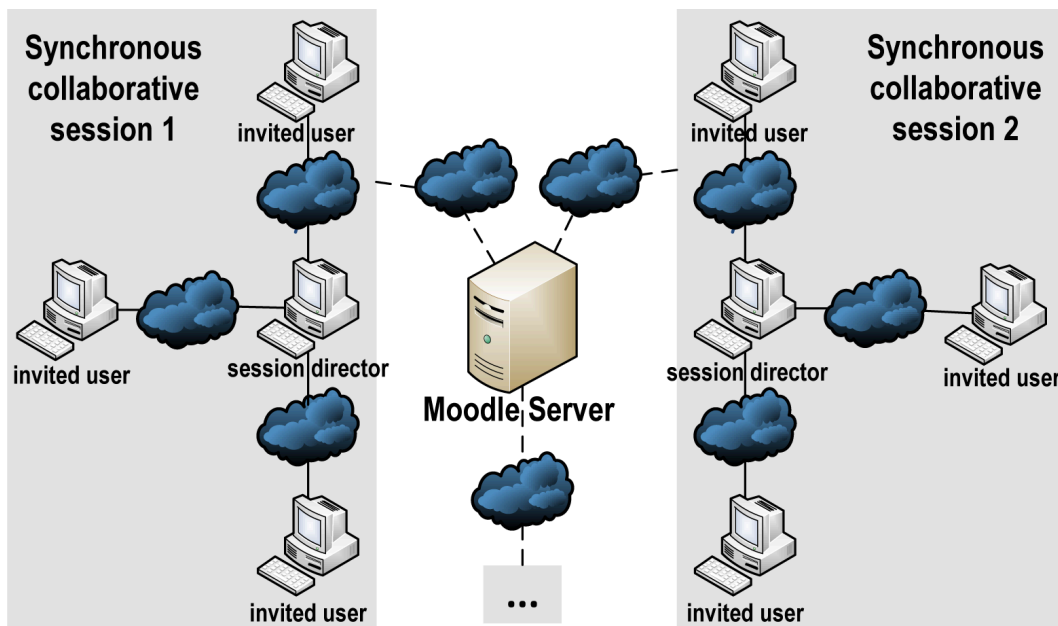


FIGURE 2.2: Peer-to-peer architecture of collaborative sessions.

For clarity reasons, the rest of this chapter only describes the global client-server architecture implementation of the framework as well as the communication links for controlling the RLs, which combination finally gives the main framework structure. More information about the special case in which P2P is used can be found in section 4.2.2.2.

2.1.2. Global Architecture

While figure 1.3 in Chapter 1 gave an insight into the global architecture of the proposed framework, the presented solution is more general than that and also offers another structure possibility.

One of the disadvantages of the structure represented in figure 1.3 is that each *Lab PC* needs to be visible in the Internet. This has two implications: 1) that each of those PCs need a public IP address and 2) that those PCs are exposed to everybody. While the first implication demands resources and money (one public IP address per *Lab PC*), the second one derives into security risks (each of those *Lab PC* can be attacked via Internet by anyone).

Fortunately, there are other structures that take care of these weaknesses and solve both problems. SARLAB (Sistema de Acceso a Recursos de LABORatorio) [71], is a free system that hides all *Lab PCs* behind a private network and manages the connection of external users (clients) to those computers. Figure 2.3 illustrates this idea.

Given this structure, the two problems mentioned above disappear since only the *SARLAB Server* uses a public IP and is exposed to the public Internet (which is one of the main concerns about RLs pointed out in [72]), instead of the N *Lab PC* used to control each of the N RLs, which only need private IPs under this structure. Moreover, SARLAB offers a software to control switching the power

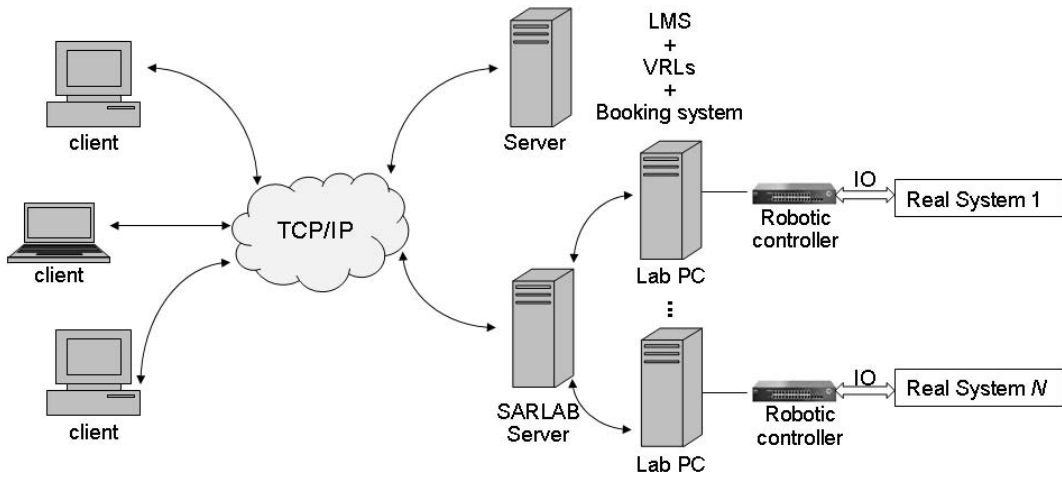


FIGURE 2.3: Global architecture using SARLAB.

supply on/off to the equipment used by the RLs, as well as a way to determine if a particular user actually has the rights to access the hardware controlled by one of the *Lab PCs* from the previous figure.

Despite all these advantages, not only an additional computer is needed in the framework when this structure is used, but it is also needed additional software (given by the SARLAB system) to manage connections from users in the Internet to the different *Lab PC*. Therefore, this increases the complexity of the whole system. While the advantages greatly exceed the disadvantages, the framework proposed in this work and the tools developed to implement it are general enough to allow the use of any of these two kind of structures or even a combination of both: using SARLAB to hide some *Lab PCs* and manage their access and leaving some other *Lab PCs* with the public IPs address solution. This way, this decision stays in the hands of the administrator of each *Lab PC*. This is specially important when creating a network of VRLs, formed by several groups, each contributing with a given number of VRLs and with their own structural preferences. This general structure, in which some *Lab PCs* may depend on a SARLAB system, some may

not, and in which several SARLAB systems are supported is illustrated in figure 2.4. It represents the most general structure, and it is the one used by the framework proposed in this thesis.

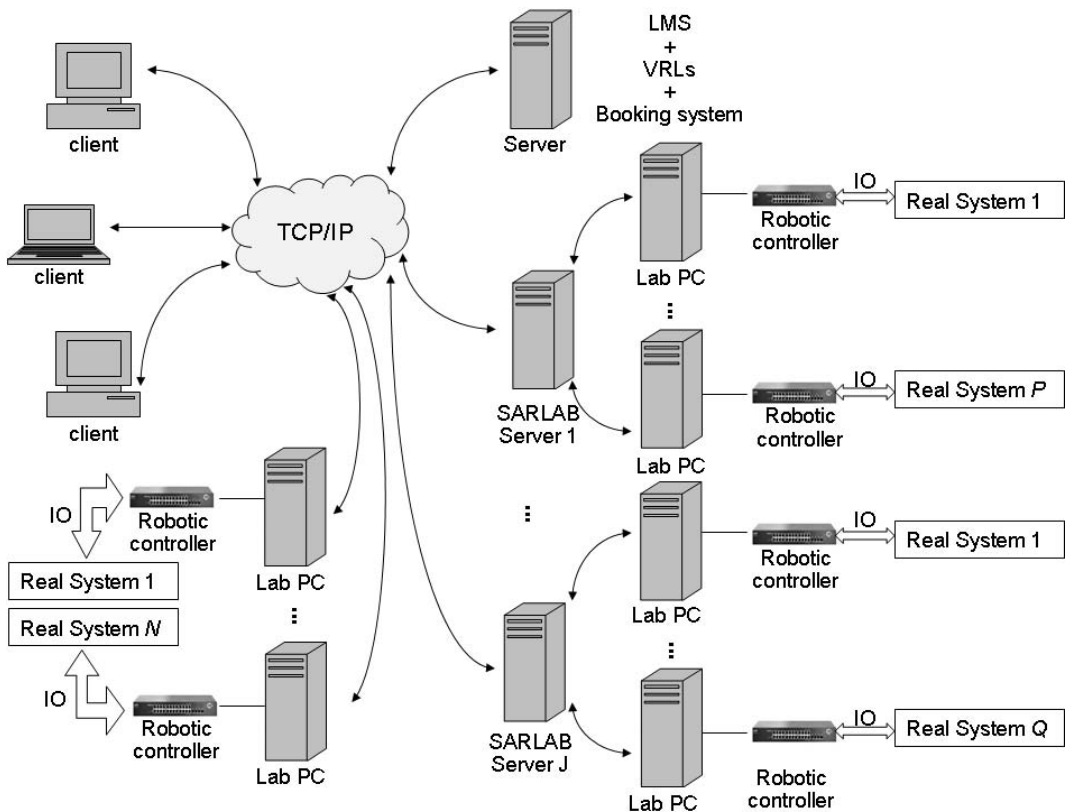


FIGURE 2.4: Global architecture with and without SARLAB.

This structure allows connecting directly (using public IP addresses) N different RLs to a network built following the proposed framework, as well as J SARLAB systems, each of them controlling the access to a different number of RLs.

For clarity reasons, none of the previous figures about possible architectures represented the use of webcams for providing visual feedback of the RLs. However, in this context, IP webcams can be considered as *Lab PCs* in terms of communication and accessibility. Webcams can use public IPs (when used without SARLAB)

or private IPs (when SARLAB is used). Therefore, IP webcams add no additional problems and their use is contemplated by the previous architectures.

2.1.3. RLS Interface - Control and Communication

While the first level in which the client-server architecture is used (deployment of the LMS, the VRLs applications, and the BS) does not require more explanation, some clarifications should be made in order to completely understand the second level (communication between clients, or users, and the hardware associated to the RLS).

2.1.3.1. Client-Side Implementation

The client side corresponds to the group of controllers, indicators, and visualization and graphical elements that form a GUI application users can utilize to carry out the experimentation activities. Since this application is the only part users actually see from the whole system, it must present a simple, user-friendly, and intuitive view.

Also, since this application may be used by a lot of different users, each with their own computers, OSs and web browsers, the software used for developing the client side must necessarily be multiplatform. For this reason, Java is a suitable choice thanks to its non-dependence on OSs. Furthermore, Java applets only need a Java-enabled web browser (and no additional software) to run. In this framework, the client side is a Java applet created with EJS and contained in the Moodle environment. This is valid for both web-lab cases: virtual and remote laboratories. This way, the internal architectural differences get unnoticed by users, allowing them to focus on the experiments themselves (in any of their two modes) instead of getting distracted by the way they have been implemented.

Finally, the communication protocol used between the user application (the client side) and the server is TCP/IP. This protocol allows the continuous exchange of data between both sides, a crucial matter for the remote laboratories implementation, in which it is necessary to modify some of the variables of the server and to receive some of their values in real time.

2.1.3.2. Server-Side Implementation

The server side corresponds to the computer which is placed close to the physical system and has the necessary hardware and software tools to communicate with the actuators and sensors of the real hardware setup (*Lab PC* icons in figure 1.3). In addition, the server must have at its disposal the necessary interfaces and/or protocols for communicating with other computers (i.e. the clients, when SARLAB is not used, or the SARLAB server when it is used).

There are a lot of nice tools that offer good solutions for this part. LabView and Matlab are two good examples, both supported in the proposed framework, used to built the UNEDLabs Webportal. These were also the solutions applied in AutomatL@bs, which allows an easy reuse of the resources.

Although the simulated experience is carried out completely at the client side, the use of the RLs works in a different way. The EJS-based applets that serve as GUI for a RL connect to real equipment. After loading the applet, the application connects to the *Lab PC* and then, the student can start interacting with the true laboratory. In the client side, interaction with the remote laboratory is done from the applet (the EJS View) using the special calls provided by the JIL or JIM classes (the sender and receiver commands of figure 2.1). These special calls are in charge of the communication dialog with LabView or Matlab in the server side by using TCP functions and routines that the JIL and JIM server hide. While the receiver command is continuously used for downloading the real data from the server side

and refreshing them in the EJS applet View (client side), the sender command is only executed to communicate to the LabView or Matlab application when a student changes a value in the EJS View.

As said before, controllers are developed in LabView for the JIL solution and in Matlab for the JIM solution and contain two information loops. The first one is an asynchronous loop that is in charge of communicating with the applet (receiving the user actions and sending the process state). The second one is a synchronous loop that monitors the true experiment in real time. The asynchronous loop may present some problems when trying to control certain processes due to their fast dynamics and the possible network delays. However, this is an issue that must be analyzed for each particular experiment (taking into account the time scale of the process dynamics) and, if problems appear, there is not a unique way to deal with them. Sections 3.1, 3.2 and 3.3 present three experiments whose control is easy enough to avoid this type of problems.

2.2. UNEDLabs: A Framework Application Example

UNEDLabs is a network of collaborative, virtual and remote laboratories developed with the framework proposed in this work. As March 2013, it has been working for more than one year, offering students from different faculties and universities the possibility to experiment with VRLs at distance. The url to UNEDLabs is <http://unedlabs.dia.uned.es>. A free test account can be used by anyone to access certain resources such as the VLs.

While this section gives an insight of UNEDLabs, a more detailed view of the features that distinguish this site from any other Moodle site is given in Chapter

4. All tools described in that chapter, were used for developing UNEDLabs and, therefore, all features mentioned there are present in this portal.

2.2.1. Network of VRLs

UNEDLabs uses the global architecture illustrated in figure 2.4 combined with the architectures shown in figures 2.1 and 2.2 for accessing the RLs and for working in collaborative sessions, respectively.

UNEDLabs is a network of virtual and remote laboratories because two reasons. The first one is that not all the *Lab PC* from figure 2.4 are placed at the same institution. Instead, they are distributed among different universities and faculties. The second reason is that different virtual laboratory applications were created by different people from the different universities and faculties that belong to the network.

In fact, the one and only requirement to form part of the UNEDLabs network is to provide at least one VRL and share it with the network. This structure allows sharing the available resources (VRLs) so other members of the network can use them in their own courses.

While sharing VRLs is not a problem, sharing the real hardware resources used by RLs (which can only be used by one person at the same time) can be more tricky. This is why a booking system is needed.

Thanks to the software reuse capabilities of the proposed framework, UNEDLabs allows the immediate integration of any VRL created with EJS into the network, such as [73–76], without any modification or additional effort.

2.2.2. Web Portal

The front end part of UNEDLabs is the web portal. The homepage of this website, which has been created in Moodle, lists the courses created by the members of the network. From this webpage, students can access the online resources for physics developed in this work. These resources consist of the VLs, the RLs, the documentation, some links to other related web pages and publications, and the LMS environment.

As of April 2013, UNEDLabs has more than 900 registered users and, during the last year, it was visited by more than 2000 people, and has received more than 6.500 total visits from almost 60 different countries. Figures 2.5 and 2.6 illustrate the previous data.

Figures 2.7 and 2.8 show the top and the bottom part of UNEDLabs' website homepage, respectively.

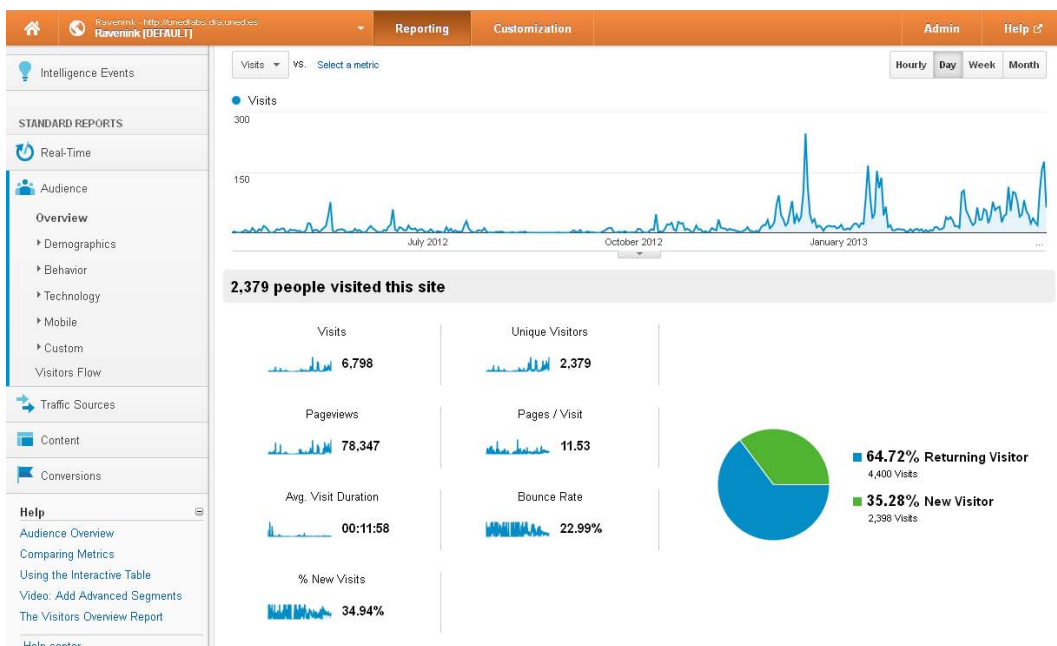


FIGURE 2.5: UNEDLabs' website visits statistics.

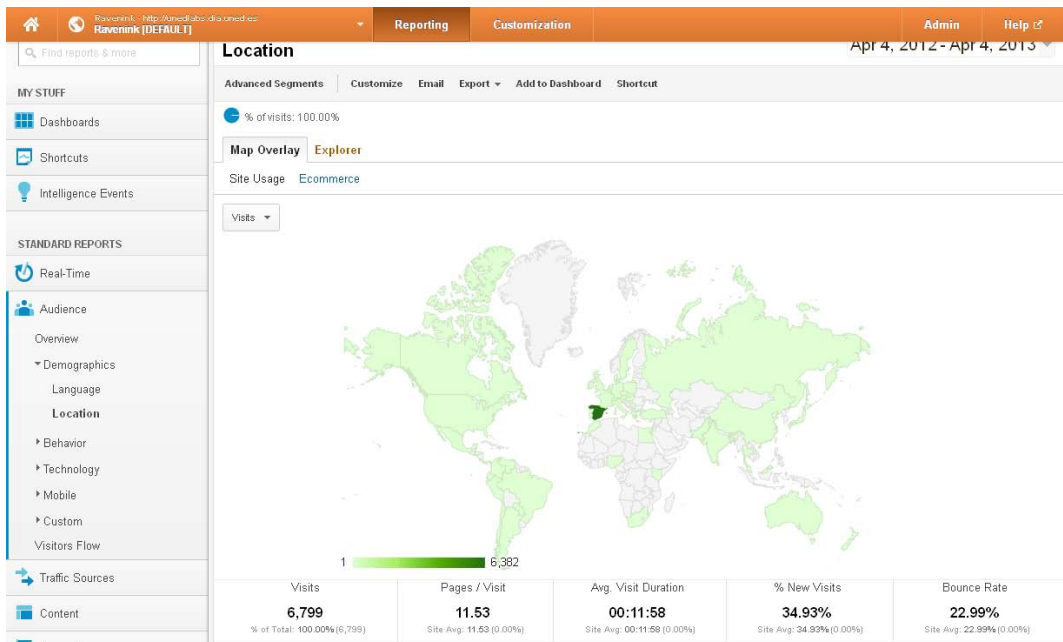


FIGURE 2.6: UNEDLabs' website visits per country.

Particularly, figure 2.7 shows some typical features of a Moodle site, such as the *CALENDAR* tool, the *NAVIGATION* menu or the *ONLINE USERS* block but it also shows some custom resources, created especially for UNEDLabs. This is the case of the video tutorials, embedded as Youtube videos, or the manuals and user guides available from the *MAIN MENU* block.

The squares of different colors in figure 2.7 highlight modules, tools, blocks or links described in the next lines:

- The green square, placed at the upper right side, shows the identification state of the user (not identified or logged in), as well as the *Login* button, which leads to the login page. Just below that, there is a dropdown menu for selecting the desired language in the environment's interface.

The screenshot displays the UNEDLabs website homepage. At the top, there is a green header with the UNEDLabs logo and the text "Network of Collaborative Virtual and Remote Labs". A user profile icon and a home icon are visible in the top left. On the right side of the header, there is a login status "You are not logged in. (Login)" and a language dropdown menu set to "English (en)".

The main content area is divided into several sections:

- MAIN MENU:** A vertical list of links including "Site News", "UNEDLabs User Guide", "How to Access the Remote Laboratories", "Manual for Accessing the Labs Applications", and "Old Automatlabs' project webpage".
- NAVIGATION:** A vertical list of links including "Home", "Site News", "UNEDLabs User Guide", "How to Access the Remote Laboratories", "Manual for Accessing the Labs Applications", "Old Automatlabs' project webpage", and "Courses".
- Tags:** A section for tagging content.
- Welcome Message:** A central area with a video player showing a welcome message and a large UNEDLabs logo. Below the video, there is a text box with the following content:

Welcome to the UNED Labs portal of collaborative, virtual and remote labs. Watch the video above to get a general view of the characteristics and features of the portal.

We also offer you the possibility to experiment with it. Free access to all our courses for unregistered users:

Username: **guestuser**

Password: **guestUser_1**

People using this account will be able to access all the virtual laboratories but not the remote ones.
- CALENDAR:** A calendar for December 2012.
- ONLINE USERS:** A section showing "Guest user" and "last 5 minutes".
- ONLINE USERS MAP:** A map showing the locations of online users across Europe, with labels for France, Austria, Italy, Portugal, and others.

FIGURE 2.7: UNEDLabs' website homepage (1).

- The blue square at the middle, highlights the logo of the Webportal, a set of Youtube videos related with the network and the VRLs and some instructions for guest users.
- The red square, placed at the upper left side, contains: 1) access to the *Site News* forum, where announcements and news related with UNEDLabs are placed and shared, 2) manuals and guides for users about accessing the VRLs and about how to use UNEDLabs, and 3) a link for accessing the old webpage of the Automatlabs project.
- The brown square, under the last one, shows the *Navigation* menu. As a user navigates around the different courses, practices, forums and other webpages of the portal, she can use this menu for easily jumping from one section to another.

- The pink square at the right highlights the calendar. Important events (such as deadlines for delivering a laboratory report, for example) and announcements appear marked in the calendar. These events are distinguished using different colors. In order to know what an event or announcement consists of, users only need to place their mouse over the event or announcement and then, a brief description about it will appear. When clicking on it, the Webportal will redirect the user to the description page of the appropriate event or announcement.
- The black square, placed under the last one, contains the blocks of online users. In these blocks, users can see what other users are online in that moment (either students, teachers and/or administrators). Moreover, by means of the online users block, it is possible to send instant messages to the other online users.

2.2.3. Decentralization

The framework proposed in this thesis along with the tools presented in Chapter 4 decentralizes the maintenance and management of the virtual courses. Therefore, in UNEDLabs teachers have as much control as possible over their online courses. Thanks to the use of Moodle teachers can, for example, add delete and/or modify the content of any of their courses. They can also review and or grade the students' progress, etc. The plugins developed in this work extend the autonomy of the teachers by giving them even more control options related to the VRLs, such as granting or removing students permissions for making bookings in RLs, or enabling/disabling the use of collaborative sessions for working with the VRLs.

2.2.4. Courses

Figure 2.8 shows the list of courses, available at UNEDLabs as December 2012, along with the names of the teachers of each of these courses.

As shown in this figure, there are several courses at UNEDLabs. Some of them belong to different universities and/or faculties rather than the School of Computer Sciences of UNED. While the topics may vary a lot from one course to another, some of them are very similar in their definition (such as the *Advanced Control* and the *Control of Advanced Processes* courses listed in figure 2.8) and so, their content can also be quite similar. In any case, even if the content is not, the structure of all courses is always similar. These similarities favor sharing the VRLs available at the network.

The screenshot shows the UNEDLabs website homepage. On the left is a sidebar with a 'TAGS' section containing various course-related terms like 'automatica', 'control', 'fisica', 'laboratorios', etc. The main content area is titled 'Available courses' and lists several courses with their respective teachers and descriptions. The courses listed are:

Course Name	Teacher(s)	Description
Autonomous Robots Practice	Teacher: José Antonio López Orozco Teacher: Dicitino Chaos García	
Advanced Control	Teacher: José Luis Díez Ruano	Virtual and remote laboratories for the Advanced Control subject of the Automatic and Industrial Informatics Master at Polytechnic Univ. of Valencia
Automatic I	Teacher: Fernando Morilla	Virtual and remote laboratories for the Automatica I subject of the Computer Science Engineering Degree at UNED
Control Engineering I and II	Teacher: Manuel Domínguez González	Virtual and remote laboratories for the Control Engineering I and II subjects of the Industrial Electronics and Automatic Engineering Degree at Univ. of León
Control of Advanced Processes	Teacher: José Luis Guzman	Virtual and remote laboratories for the Control of Advanced Processes subject of the Chemical Engineering and the Master on Industrial Computer Science at Univ. of Almería
Contaminación por Agentes Físicos	Teacher: Pablo Domínguez García Teacher: Amalia Willart Torres Teacher: Manuel Pancorbo	Virtual laboratories for the Pollution by Physics Agents subject of the Environmental Science Degree at UNED
Experimental Techniques in Physics	Teacher: Amalia Willart Torres Teacher: Juan Pedro Sánchez Teacher: Manuel Pancorbo Teacher: Pablo Domínguez García	Virtual and remote laboratories for the Experimental Techniques subject of the Physics Degree at UNED

FIGURE 2.8: UNEDLabs' website homepage (2).

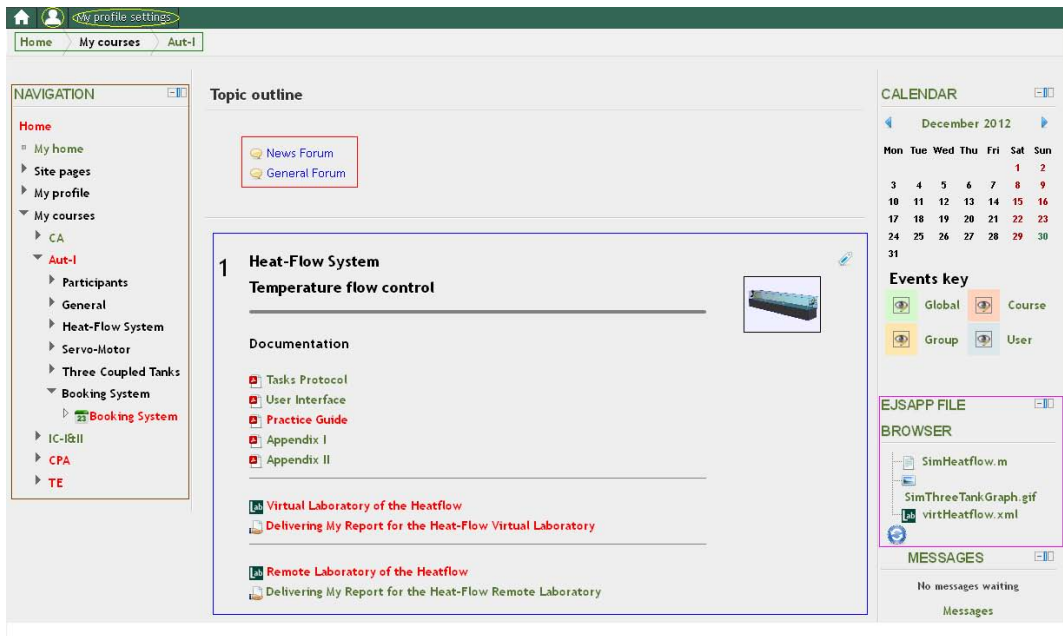


FIGURE 2.9: A course at UNEDLabs.

Figure 2.9 shows a screen-shot of one of the courses at UNEDLabs. The main key parts are:

- The yellow circles, placed at the upper left part, show the accesses to the personal menus of the user: blogs, profile settings (such as changing the password, the e-mail direction, the picture or avatar, the default language for the portal, etc.), messages, private files, etc.
- The brown square highlights, once more, the navigation menu but this time it is quite spread out.
- The green square, over the last one, indicates a navigation menu which can also be used to surf around the Webportal in an easy and quick way.
- The blue square delimits the Didactic Unit number 1 for the subject *Automatic I*. In this unit, users work, among other systems, with a heat flow

control system. In UNEDLabs, all didactic units have the same structure: a first part with documentation (where the tasks protocol, a manual explaining the applications' GUI, a theory guide for the practice and one or several possible annexes can be found), a second part with the experimental applications themselves (VL and RL) and, finally, a tool for delivering the practice reports prepared by the students in relation with the work done with the previous applications. While didactic units are enumerated in a way that indicates the order in which they are recommended to be done, students are free to perform the experimental activities in the order they prefer to.

- The red square shows the specific forums for the selected course. The first of them is always dedicated to publish announcements and news only related to the subject of that course and, therefore, only administrators and teachers of that course can publish messages in it. In the second forum, open to anyone enrolled in that course, offers a space of collaboration where doubts about the subject can be asked and other related helps can be given by both teachers and students.
- The pink square now highlights a private files repository. This file repository is a private space for each user that has logged in the system. It is to this repository where the files saved from the VRLs end up. By allowing users to use this cloud storage feature, their saved work is available to them anywhere/anytime. More details about this block can be found in section [4.2.2.1](#).

Finally, users can find the link for accessing the booking system at the bottom of each course. This is shown in figure [2.10](#). More information about the BS can be found at section [4.2.1.2](#).

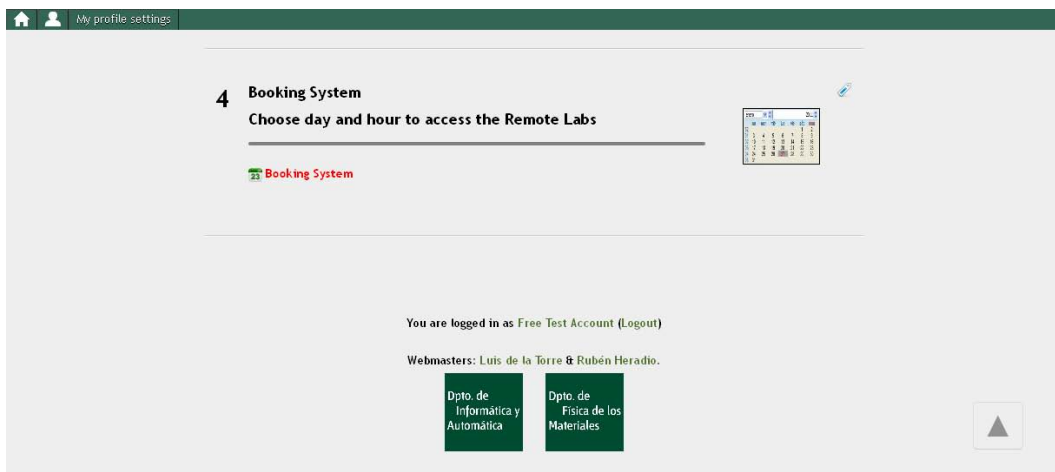


FIGURE 2.10: Booking system link in a course at UNEDLabs.

2.2.5. A VRL Example in UNEDLabs

The one and only purpose of this section is to show the look and feel of the VRLs embedded within the Moodle LMS. Detailed information about the VRLs can be found in Sections 3.1, 3.2, 3.3 and in Appendix B.

When a user clicks on one of the VRLs links, such as the ones shown in figure 2.9, the web portal will load the selected application. In the previous example, the heat flow (virtual or remote) system would be loaded. Figure 2.11 shows the java-applet application created with EJS for the VL of this system.

2.2.6. The Booking System

As described in Chapter 1, remote laboratories add interesting features that virtual laboratories do not present. However, they also present more complications when making them accessible to users. VLs can be simultaneously used by as many people as wanted (or at least, as many as the server computation capabilities and its Internet connection quality bear), but RLs can only be used by one user at

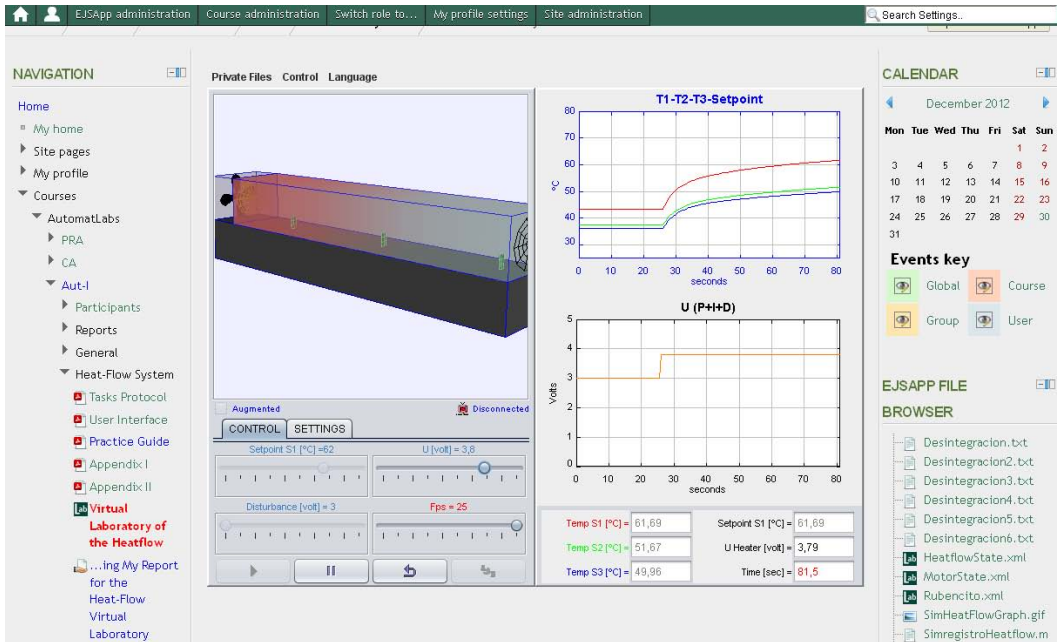


FIGURE 2.11: A virtual lab example (the heat flow system) at UNEDLabs.

a time per each experimental system. Furthermore, since remote devices require some heat power to work and they even get deteriorated by their use, they should only be used by the people that are supposed to. These two conditions imply the necessity of some kind of authentication system (to identify the user) and a booking system (to assure the availability of the experimental systems when the user needs them). Figure 1.2 reflects this requirement and the use of a BS that takes care of the previous tasks.

Even though each course in UNEDLabs has a link to the BS (as explained in section 2.2.4), this system is unique in the whole network (as figure 2.4 illustrates). This is a requirement, for the same RL can be used in different courses and, therefore, reservations must be managed in a global way for the whole network of RLs. From the client side point of view, it is necessary to introduce a module capable of sending the username and the encrypted password to the server. On the

other hand, the server needs to check whether the user actually has a reservation for the chosen experiment at the corresponding time slot.

The BS applications (both parts of it the one attached to the client and the one related to the server) used in UNEDLabs were developed in the Computer Sciences and Automatic Control Department of the UNED and presented in several works such as in [77]. This BS was originally designed to be used in AutomatL@bs, with the eMersion LMS. However, this work adapted and expanded these applications so they now work with Moodle and offer new features. The presented BS lets students choose a specific date and time to book the real plant for exclusive use. It is composed of two parts: 1) the booking system interface (at the client side) and, 2) the web-server that manages and maintains the students reservations (at the server side). Figure 2.12 shows the user interface whereby students perform their reservations of any experimentation laboratory.

These are the steps to make a booking:

- First, users need to be identified in the system with a registered username and password (Figure 2.12(a)). The developed Moodle plugin passes these data to the BS application, which recognizes the user and her language setting for the web portal.
- Once the user has accessed to the booking system, s/he must choose the date and the experimental system (Figure 2.12(b)) s/he wants to work with.
- Afterwards, the student must check if the remote plant selected is available for the proposed date (Figure 2.12(c)).
- The student must then select the timetable for the use of the remote lab (Figure 2.12(d)).

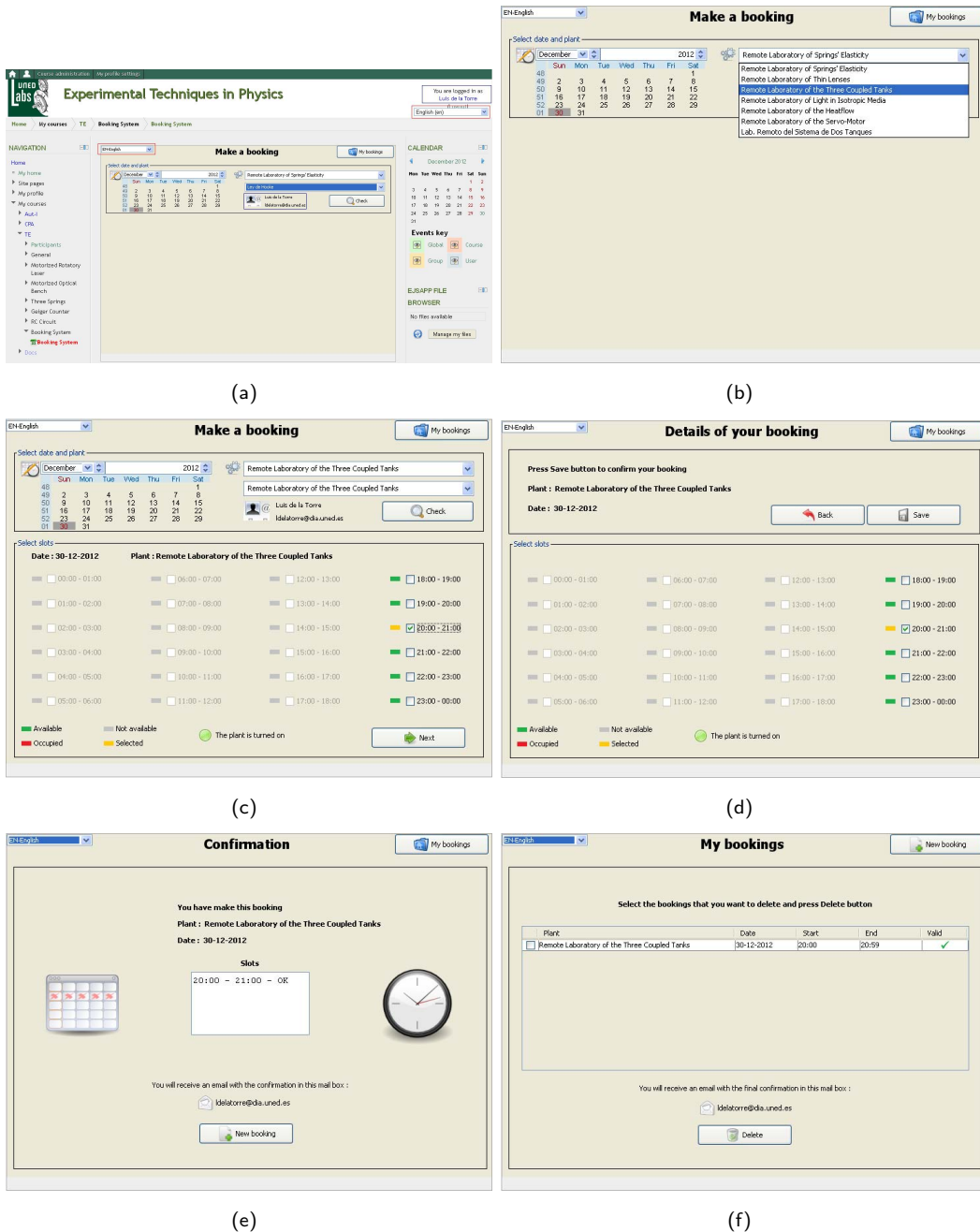


FIGURE 2.12: States of the client booking system interface during the reservation process.

- The booking system confirms the reservation performed by the user (Figure [2.12\(e\)](#)).
- Finally, the application shows the list of active bookings for that particular user (Figure [2.12\(f\)](#)).

Chapter 3

New Web-Labs Developed

This chapter presents all the virtual and remote laboratories that were specifically developed for and during this thesis. All of them are now part of the UN-EDLabs network, along with some already existing VRLs which were adapted (see Appendix B) with this purpose. While the old and adapted VRLs are all related to Control Engineering, these new web-labs are related to Physics topics.

3.1. Light in Isotropic Media

3.1.1. Introduction

Snell's law (or the law of refraction) along with the law of reflection constitute axioms already stated by Sir Isaac Newton in [78] and they are both part of the fundamental laws in optics. Particularly, Snell's law describes the relationship between the angles of incidence (θ_i) and transmission (or refraction, θ_t), when light (or other waves) passes through a boundary between two different isotropic media (air and water, for instance). This relationship is given by the indexes of

refraction of the two media (n_i and n_t) in such way that the ratio of the sines of these angles is a constant, $n = n_t/n_i$. Mathematically

$$n_i \sin \theta_i = n_t \sin \theta_t. \quad (3.1)$$

When $n_t > n_i$, $\theta_t < \theta_i$ so the light comes closer to the perpendicular line to the boundary. On the contrary, when the light passes from a more refracting media to a less refracting one ($n_t < n_i$), it gets away from the perpendicular line.

As later described, the experimental system allows the study of the relationship between the angle of refraction and the angle of incidence for both cases: $n_t > n_i$ and $n_t < n_i$. In this second case it is possible to check that there is a certain value of the angle of incidence for which the angle of refraction reaches 90° . This angle is known as limit angle (θ_l), since for greater values of the angle of incidence the refraction phenomenon disappears and the boundary between the two media behaves like a mirror. This is known as the total reflection effect.

The experimental system for Snell's law real experiment (see figure 3.1) was built using aluminum pieces, a transparent plastic container, a webcam, a green laser pointer, and a stepper motor. The aluminum pieces make up the basic structure of the experimental system which holds the plastic receptacle (containing water or oil, for example), the motor, and the laser pointer. An aluminum disc that rotates around its central axis supports the laser pointer. The stepper motor controls the rotation of the disc and so, the position of the laser pointer in relation with the plastic container and the angle in which its light incidences over the water (θ_i). Since the motor is controlled by steps of 0.9° , the error is $\delta\theta_i = \pm 0.9^\circ$. A relay is used in order to switch on/off the laser pointer when a user connects/disconnects to the remote laboratory, respectively.

¹This experimental system was designed by Prof. Juan Pedro Sánchez. This VRL was published in [50] and [48].

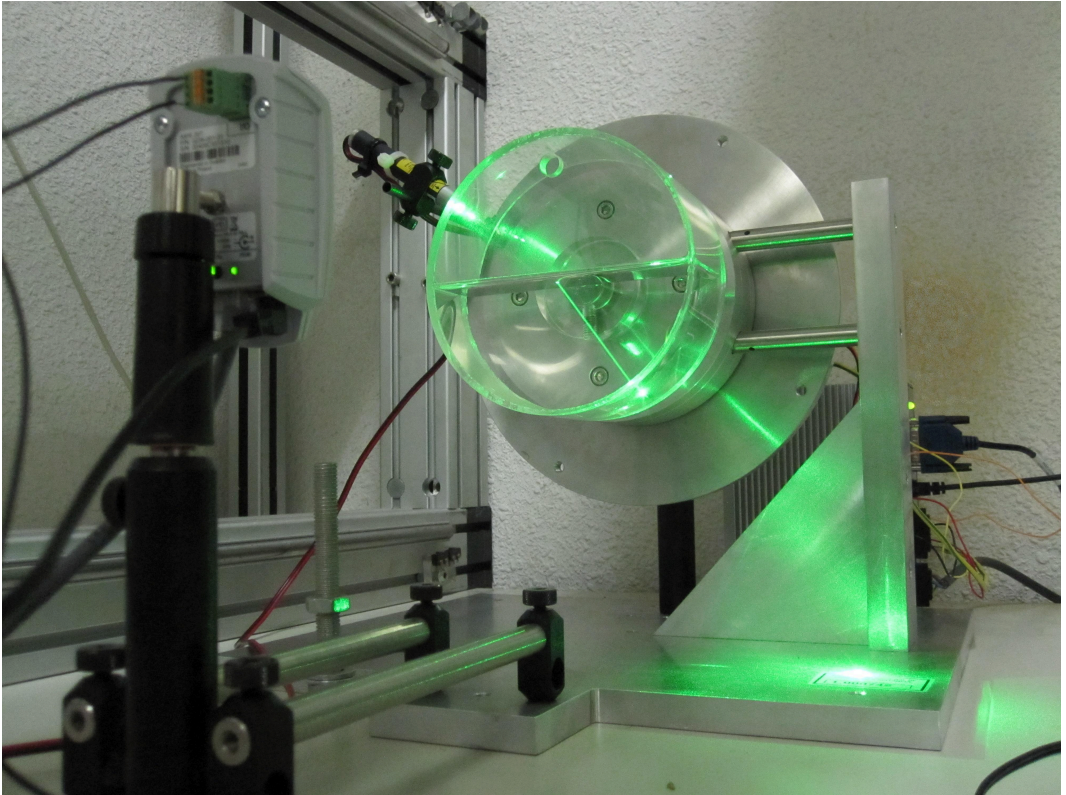


FIGURE 3.1: Experimental system of the light in isotropic media experiment¹.

To measure θ_t , the GUI of the remote laboratory (figure 3.5) lets the user move a virtual pointer over the image given by the webcam using a graduated slider. This way, once the user has positioned the pointer where the refracted ray is shown, θ_r is measured over the mentioned slider. The estimated error for these measurements is $\delta\theta_r = \pm 1^\circ$.

Since the only controlled variable in this experiment is the angle of incidence, there is only one control parameter (θ_i) to be sent from the client-side sender (figure 2.1) to the LabView hardware controller. Finally, since θ_t is measured visually during the remote experimentation (using the webcam and the virtual movable pointer), no information needs to be returned from the server to the

client.

3.1.2. Virtual Laboratory

The simulation for studying the laws of reflection and refraction is based on the experimental system used for remote experimentation. In this way, the simulated view also serves students for getting used to handling the experimental environment they will use later.

The central part of the window in figure 3.2 shows a 2D representation of the system (cylindrical container, laser pointer and graduated scale) using a real picture of the experimental system. The container has two different isotropic media in the same proportion. The user can select these media from a list clicking the *Media1* and *Media2* buttons, under the *Media selection* label. After reaching the

FIGURE 3.2: User interface of the light in isotropic media experiment working on the simulation mode².

boundary between the two media, the laser ray splits off into the reflected and the refracted rays. Students can change the position of the laser (and therefore, the incident angle) acting directly over the 2D representation of the system (clicking over the laser and dragging the mouse) or using the graduated slider. Either way, the incident angle changes and the behavior of the other two angles and rays (reflected and refracted) can be studied. As seen in figure 3.2, two displays, placed under the slider that controls the position of the laser, show the values of these two angles.

From the data obtained by means of this simulation, students can carry out the same activities they would be able to carry out during a traditional hands-on laboratory about light in isotropic media:

- Verification of the law of reflection: the angles of incidence and reflection match up ($\theta_i = \theta_r$).
- Verification of Snell's law: the graphic representation of $\sin \theta_i$ vs. $\sin \theta_t$ gives as result a straight line. By performing a lineal regression with the represented data it is also possible to determine the ratio of the refractive indexes of the two media, since n_t/n_i is the slope of the straight line.
- Range of validity of the Gauss approximation: the θ_i vs. θ_r graphic representation can be approximated with a straight line only for small values of these angles. Within this range, the slope of that straight line can still be considered as the ratio of the refractive indexes of the two media.
- Determination of the limit angle (total reflection phenomenon): if the incident media is more refracting than the second one, there is a particular value for the angle of incidence, θ_l , such as the ray does not cross the boundary

²This simulation is based in one originally made by Prof. Juan Pedro Sánchez.

between the two media for higher values of θ_i (i.e. when $\theta_i \geq \theta_l$). In these cases, the ray is completely reflected towards the interior of the incident media.

All the simulations in UNEDLabs include small random errors when the measured values are generated. For this particular case, the values for the reflected and refracted angles are generated using the theoretical ones and adding a small random error, which can be either positive or negative. This way, although there are uncertainties in the measurements, results are still theoretical. The *Facto* field under the *Precision* label (at the bottom part of the left column) lets users modify the maximum value of this random error, and generate measurements with variable precision. Using the default configuration, students can determine n_t/n_i with good enough precision in the simulated experiment: around $\pm 5 \cdot 10^{-3}$ when

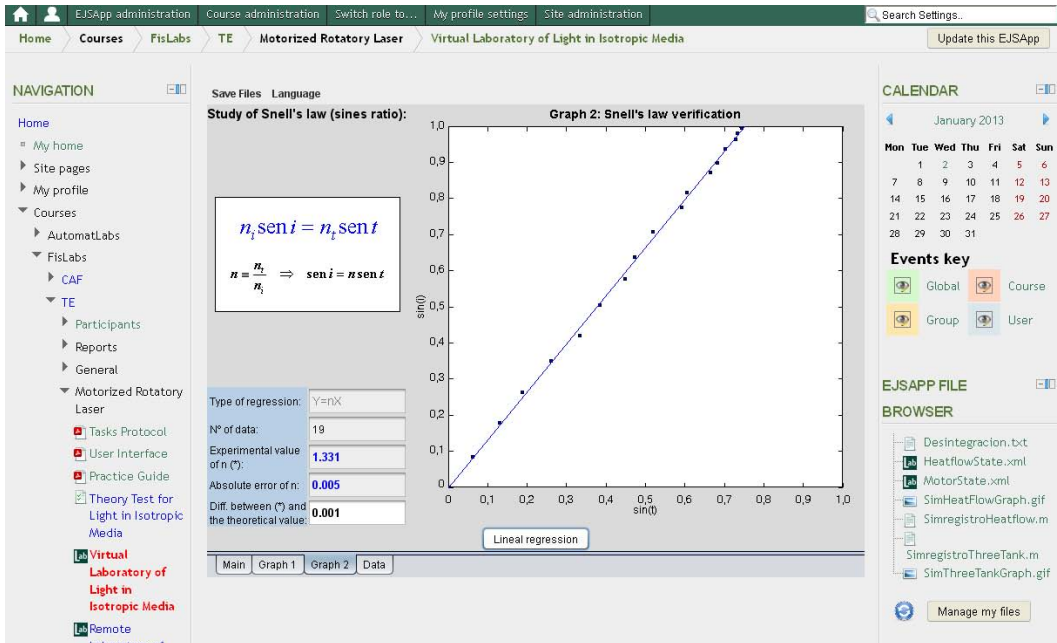


FIGURE 3.3: Linear regression using data obtained in simulation and applying Snell's law.

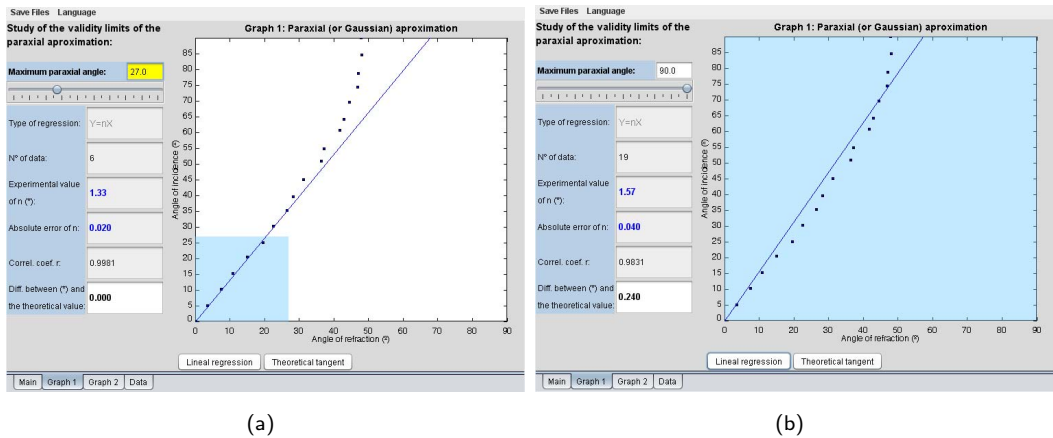


FIGURE 3.4: Linear regression using data obtained in simulation and applying the paraxial approximation to (a) small angles and (b) all measures.

eighteen points are used for the lineal regression, for example (see figure 3.3) or around $\pm 7 \cdot 10^{-3}$ when ten points are used.

Figures 3.3 and 3.4 show the linear regressions performed with the data obtained during a virtual experiment. As those figures show, the VL application (as the RL application also does) allows users to automatically perform linear regressions, determining the associated slope and errors. In this case, as it was explained before, the result obtained for the calculated slope (i.e. the media's index of refraction) using the paraxial approximation (figure 3.4) is only similar to the result obtained when using Snell's law (figure 3.3) when considering small angles (figure 3.4(a)) but not when big angles are taken into account (figure 3.4(b)).

3.1.3. Remote Laboratory

When a student is connected to the remote laboratory, s/he can check Snell's law using the experimental system of figure 3.1. Figure 3.5 shows the GUI of the applet in this mode. Two sliders are shown on the upper left part of the main window: one for controlling the angle of incidence and the other one for

measuring the angle of refraction. On the right side, a webcam shows an image of the experiment in real time while the on the left side, controls and indicators are provided to the user. The *Laser position* control allows changing the position of the laser. The *Angle of incidence* indicator translates the position of the laser into the angle of incidence of the light. Finally, the *Angle of refraction* slider is used to move the blue arrow, which helps measuring the angle of refraction. A checkbox labeled "Show virtual rays" allows users to draw over the real image the three free-error simulated rays (incident, refracted and reflected) to immediately compare and contrast the real and theoretical results. This can be considered like an augmented reality procedure.

All the four activities described for the simulated experiment in the previous

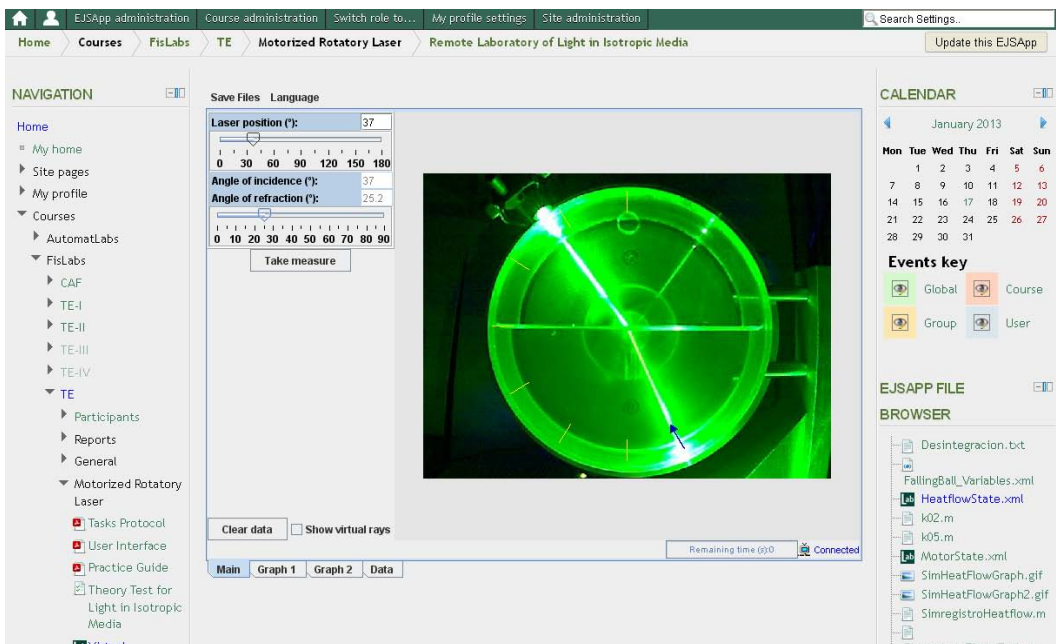


FIGURE 3.5: User interface of the light in isotropic media experiment working on remote mode. The virtual pointer (a blue arrow) is used to measure the real angle of refraction.

section can be performed remotely with this experimental system and the application offered in UNEDLabs. Figure 3.6, for example, shows how the total reflection phenomena can be visualized using the RL.

As shown in figures 3.3 and 3.4, the GUI for this experiment also offers students the possibility of automatically carrying out lineal regressions of the collected data. Taking into account the measurement errors in both θ_i and θ_t , students can determine n_t/n_i with good enough precision in the remote experiment (around $\pm 8 \cdot 10^{-3}$ when ten points are used for the lineal regression, for example). Therefore, the uncertainties in measurements and results for the real and simulated experiments are very similar ($\pm 8 \cdot 10^{-3}$ and $\pm 7 \cdot 10^{-3}$, respectively).

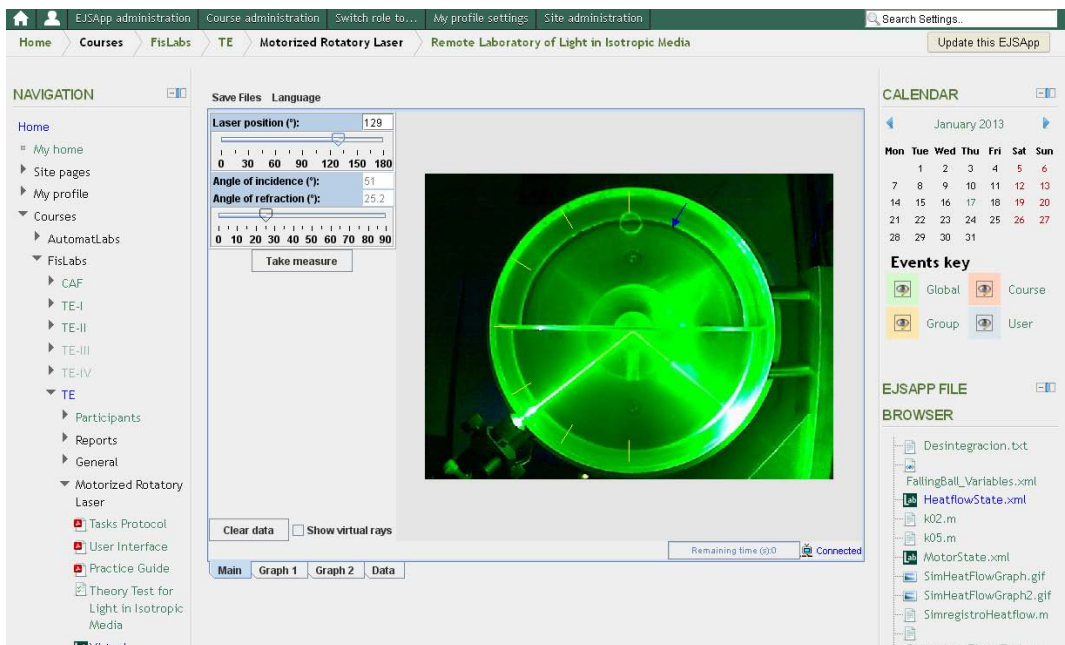


FIGURE 3.6: Visualization of the total reflection phenomena using the remote laboratory.

3.2. Hooke's Law and Non-Linear Springs

3.2.1. Introduction

In mechanics, Hooke's law of elasticity is an approximation that states that the extension of a spring is in direct proportion to the applied force. This statement is good enough as long as this force does not exceed the elastic limit. Mathematically

$$F(t) = -k \cdot (x(t) - l) \text{ (N)}. \quad (3.2)$$

where k is the spring constant, l is the natural length of the spring, $x(t)$ is the actual length of the spring at time t , and $F(t)$ is the restoring force exerted by the material at that moment.

Hooke's law real experiment uses the next elements: one LEGO NXT intelligent brick, four springs with different elastic constants, one NXT dc motor, a 25 cm long rack rail, one contact sensor, a Vernier force sensor, a webcam, and a ruler (if desired). The spring has one of its ends fixed to a wall, while the NXT dc motor (which is mounted over the rack rail) is attached to the free end of the spring; in this way the motor can stretch the spring when it moves along the rack rail. The contact sensor is used for resetting purposes when the motor reaches the end of the rack rail.

The Vernier force sensor has a wide enough measurement range ($[-10 \text{ N}, +10 \text{ N}]$) and a precision of $\delta F = \pm 0.01 \text{ N}$. Finally, the ruler has the usual markings with 1 mm graduations. In this experiment, students can control the position of the NXT motor over the rack rail and thus, the actual length of the spring. The NXT motor has a rotation encoder with $\pm 1^\circ$ of precision which translates into about $\pm 0.4 \text{ mm}$ of precision in the spring stretching (considering that the error of the known value, l , is zero). This means that students control the variable in

equation 3.2 with a precision of ± 0.4 mm. However, students may also use the ruler for measuring the spring stretching and so the obtained precision would be ± 1 mm. F is the variable measured by the force sensor and it is given to students with a precision of ± 0.01 N. The natural length of the spring, l , is known (and the associated error is not taken into account) and the spring constant, k , is the unknown, to be determined.

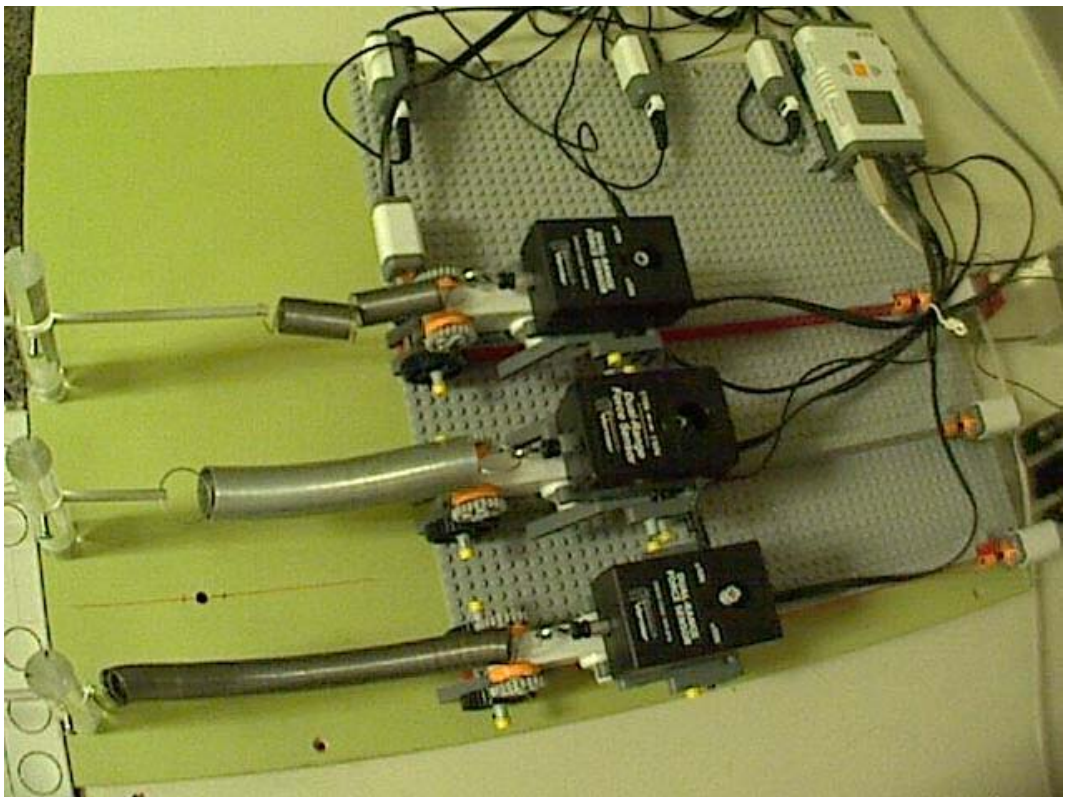


FIGURE 3.7: Experimental system of Hooke's law and non-linear springs experiment².

Since the cross-sectional area of the spring, A , is also known (and given to the students), and using equation 3.3, it is possible to obtain the modulus of elasticity

²This VRL was published in [50] and [49].

(Young's modulus actually, which is defined as the relation between the tensile stress and the tensile strain), E , once the value of k has been determined:

$$k = A \cdot (E/l) \text{ (N/m}^2\text{)}. \quad (3.3)$$

Once this value has been determined, students can compare it with the tabulated values of this parameter for different materials and try to find out which material is the spring made of.

Therefore, in this experiment there is just one control parameter to be sent from the client-side sender (figure 2.1) to the LabView hardware controller: the position of the NXT motor over the rack rail, x . The returned information from the server-side to the client-side receiver (figure 2.1) is the measured force and the reading of the encoder translated into the actual position of the NXT motor (F and x , respectively). The other way of measuring the position of the motor does not require sending any data because these measurements are directly done by the students using the ruler, which is visible thanks to the webcam.

3.2.2. Virtual Laboratory

The simulated experiment let students discover Hooke's law in linear springs, see how the potential energy (V) of a spring changes when it is stretched or compressed, and observe the behavior and differences between soft and hard non-linear springs. In figure 3.8, the upper graphic shows the F vs. x lineal relationship (that is, Hooke's law), while the lower one presents V against the stretching/compression of the spring. These graphics change dynamically when the student moves the ball attached to the spring (by dragging it with the mouse) in order to stretch or compress it or when the "Play" button is pressed.

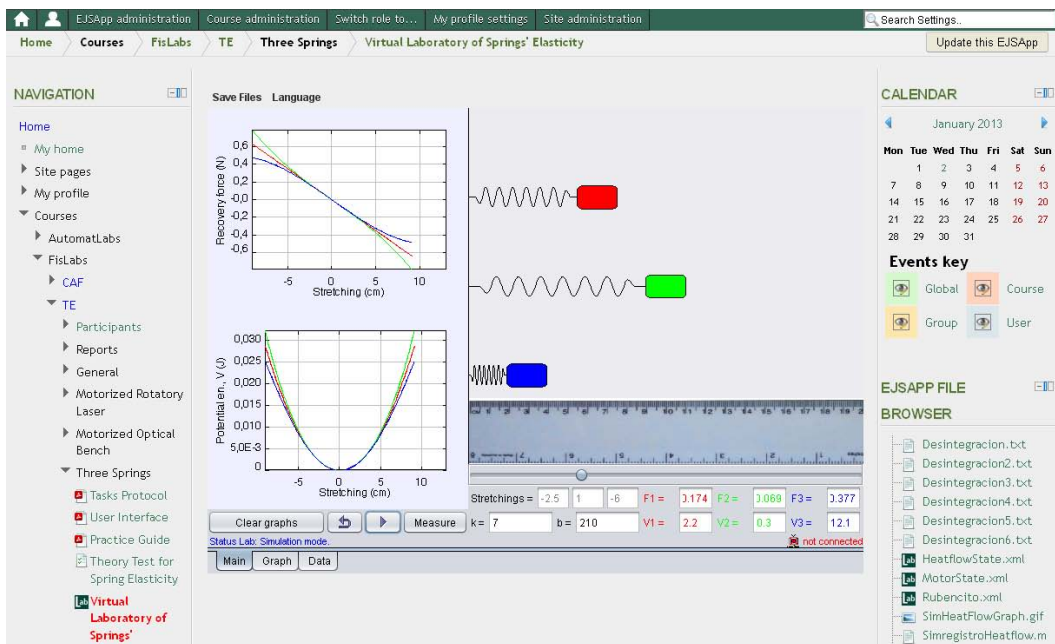


FIGURE 3.8: User interface of Hooke's law and non-linear springs experiment working on the simulation mode.

A default value for the spring constant (k) is used in the simulation but users can change it just writing the desired value in the numeric input field. The three springs in the simulation always have the same k . Non-linear springs are described by the next expression:

$$F(t) = -k \cdot (x(t) - l) + b \cdot (x(t) - l)^3 \text{ (N)}. \quad (3.4)$$

Equation 3.4 is a generalization of equation 3.2, where $b = 0$.

As in all simulations, the data collected during the simulation can be saved in text files so students can compare the results from the simulated experiment with the real ones (see section 3.2.3) and analyze their possible differences. Figure 3.9 shows the data tab every VRL application from Sections 3.1, 3.2 and 3.3 have. The data stored in here is the one users can save, as text files, into their Moodle

row #	Stretching (cm)	Force1 (N)	Force2 (N)	Force3 (N)	E	F	G
0	0,966	-0,042	0,220	-0,069	1,690	-0,036	0,680
1	1,977	-0,108	0,530	-0,164	0,760	-0,116	1,050
2	2,989	-0,205	2,870	-0,225	4,680	-0,189	2,770
3	3,954	-0,263	6,660	-0,295	5,350	-0,253	5,010
4	4,966	-0,318	9,160	-0,379	9,240	-0,344	8,270
5	5,977	-0,404	12,960	-0,488	12,870	-0,363	13,280
6	7,034	-0,521	17,680	-0,586	19,820	-0,386	16,560
7	8,046	-0,591	23,540	-0,674	26,560	-0,438	21,690
8	8,966	-0,598	28,280	-0,799	31,780	-0,466	26,420
9	9,977	-0,695	36,850	-0,908	39,550	-0,487	29,300
10	10,989	-0,744	42,150	-1,028	49,860	-0,502	34,220
11	11,954	-0,827	50,340	-1,216	59,260	-0,489	38,890

Main Graph Data

FIGURE 3.9: The data tab stores all collected data during the experimentation, either in virtual or in remote mode. These data can be saved to the *EJSApp File Browser* block using the *Save Files* menu.

private files repository (in the *EJSApp File Browser* block) using the *Save Files* menu.

3.2.3. Remote Laboratory

When connected to the remote laboratory, students are able to check Hooke's law using the experimental system. Two of the rails have linear springs with different k (call them k_1 and k_2 for example), while the last one actually has two linear springs in series. In this last case, the equivalent (total) elastic constant, k_{eq} , measured by the students would be such as $\frac{1}{k_{eq}} = \frac{1}{k_{3a}} + \frac{1}{k_{3b}}$, where k_{3a} and k_{3b}

are the elastic constants of each of the two springs. Actually, in the experimental system $k_{3a} = k_1$, which is told to students. Since they can determine k_1 and k_{3eq} , they can use the previous equation for springs in series to calculate the elastic constant of the other spring, k_{3b} . The spring in the middle (figure 3.7) has an elastic constant of $k_2 = 2$ N/m. The others are not calibrated but the values can be determined by means of this experiment. Using the data shown in 3.11, for example, we obtain $k_1 = k_{3a} = 9.7$ N/m, and $k_{3eq} = 5.45$ N/m, so $k_{3b} = 12.4$ N/m.

Figure 3.10 shows the GUI in this mode. Two numeric fields located on the upper left corner of the main window display in real time the force value (measured by the sensor) and the spring stretching. The slider bar located below them is used for changing the position of the motor over the rack rail and thus, the stretch of the spring. As in Snell's law remote experiment, a webcam image on the right side shows the experiment in real time while the graphic on the left side changes dynamically to represent the force vs. the stretching of the spring.

The physical phenomenon may be quite simple in this experiment but there are still many other important activities to carry out and lessons to learn related to metrology by means of this laboratory. For example, measurement uncertainties must be taken into account by students. As said before, the measurement uncertainties in this experiment are: $\delta x = \pm\{0.4, 1\}$ mm (depending on how the spring stretching is measured: using the motor encoder or the ruler), $\delta l = 0$ for the natural length of the spring, and $\delta F = 0.01$ N for the force measurements. Considering the last value, supposing a spring's elastic constant of 14 N/m, and taking into account the relation given by 3.2, students should realize that they should take measurements with at least 0.7 mm of difference between each stretching for such particular spring in order to be able to see force differences.

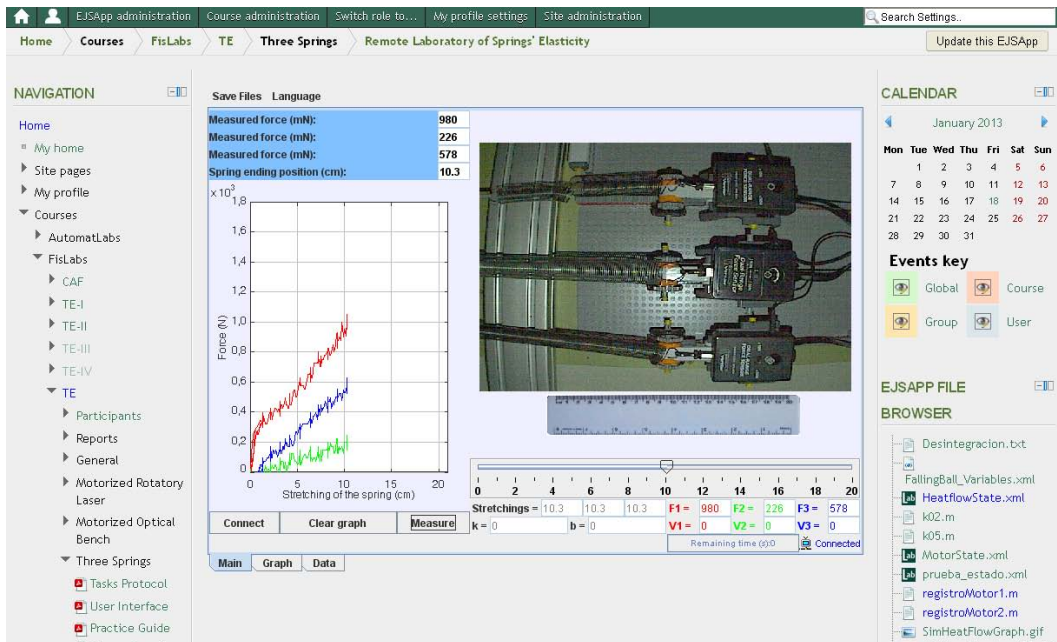


FIGURE 3.10: User interface of Hooke's law and non-linear springs experiment working on the remote mode

As seen in figure 3.10, the signal presents some noise due to the movement of the motor (not perfectly smooth), the limited precision of the force sensor, and possible network delays. However, students can take individual measurements and plot them on an independent graphic. This can be done by the *Measure* button located at the bottom part of the experimentation applet (figure 3.10). By pressing this button, the user grabs a measure of F and $x - l$ and plots the pair on the graphic presented in figure 3.11. The graphic is displayed by clicking on the tabbed panel *Graph* located below the button. Figure 3.11 shows an example of lineal regression using data corresponding to particular measurements from this experiment. This way, the result does not present the previous noise (although the uncertainty over the measurements is still there, and so the correlation of the regression is not exactly 1).

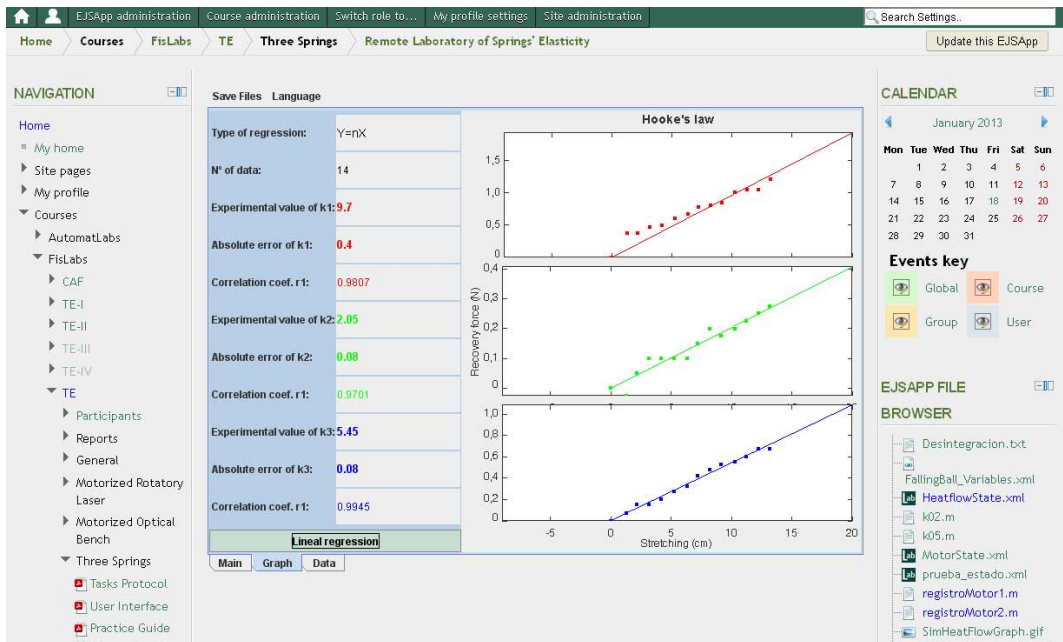


FIGURE 3.11: Linear regression tool of the RL application with a set of real data

Uncertainties calculus can also be practiced here since students should give their results (the spring constant and the modulus of elasticity values) with the appropriate uncertainties. Following these ideas, this experiment may also serve students to realize the importance of using well-balanced equipment. In this example, a great precision (small uncertainty) in the stretching measurements (using the NXT motor encoder) is considerably wasted when determining the spring constant value because of a quite poorer precision (big uncertainty) in the force measurements.

Finally, in order to obtain k , students are asked to make a linear regression with the collected data (F against $x - l$) for both the simulated and the remote experiment, and to compare the results with the graphs shown in figure 3.11. Therefore, least squares adjustments (manually and/or computerized) are also

practiced by students.

3.3. Focal Length of Thin Lenses

3.3.1. Introduction

The focal length, f , of an optical system is a measure of how it makes light converge (focus) or diverge (defocus). For a thin lens in air, the f is the distance from the center of the lens to the lens focal points. If the lens is converging, f is positive, and its value is the distance at which a collimated beam will be focused to a single spot. The f , the object distance from the object plane to the lens (s), and the image distance from the lens to the image plane (s') are related by:

$$\frac{1}{f} = \frac{1}{s} - \frac{1}{s'}. \quad (3.5)$$

By means of this VRL, students can do the following:

- *Determine the focal length of a thin lens* (method 1). Users collect several measurements of s and s' (listed in the Data tab of the applet) when the

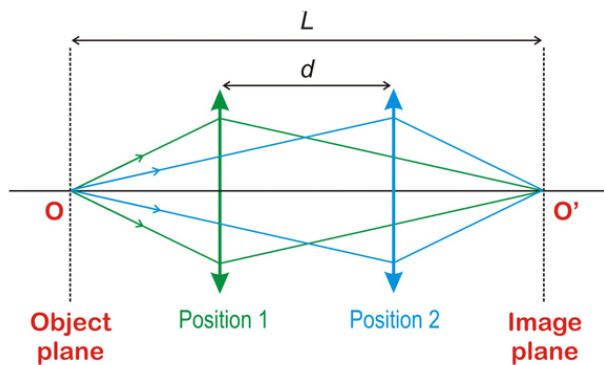


FIGURE 3.12: Explanatory scheme of method 2.

beam is focused over the screen. Then, using 3.5 and performing a linear regression (Graph tab of the applet), they obtain f .

- *Determine the focal length using Bessels method* (method 2, see figure 3.12). If the distance between the object and image planes, L , is greater than $4f$, we can find a second position for the lens in which the beam is also focused over the screen. These two positions are marked with blue arrows in the experimentation applet (figure 3.15(b)). Let d be the distance between those two positions of the lens, then:

$$f = \frac{L^2 - d^2}{4L}. \quad (3.6)$$

- *Visually check the focal length.* Students can move either the screen or the lens to a certain position, shift the other element (screen or lens, respectively) to a distance of f (determined in the previous steps), and check if the beam focuses over the screen. Students can compare the precision of measuring the focal length by both methods.

Figure 3.13 shows the experimental system for this experiment. Basically, a two axis motorized bench is used for setting the position of the lens and the screen independently. A relay is connected to the laser to switch it on or off when a user connects or disconnects from the RL. A diffraction grating is used to split the laser beam into several rays. Finally, two webcams are used: one for giving a general view of the experiment and another one, situated close to the screen, for watching whether the rays are focused over or not.



FIGURE 3.13: Experimental system of the focal length of thin lenses experiment³.

3.3.2. Virtual Laboratory

Figure 3.14 shows a snapshot of this web-lab working in simulation mode when using the first method for determining f . The virtual representation of the system shows a laser at the left part. Next to it, a diffraction grating splits the laser beam into several rays. The lens is the next element over the rule. Users can change its position moving its corresponding slider or clicking with the mouse over the visual element, dragging it to the new desired position and then dropping. Finally, there is the screen, which can be also moved in the same two ways as the lens. Two views of the screen are given: a side one and a front one (represented by the white rectangle). It is in this last view where students can check if there is only one dot visible (i.e. the rays focus over the screen) or there are more.

³This VRL was published in [69] and [47].

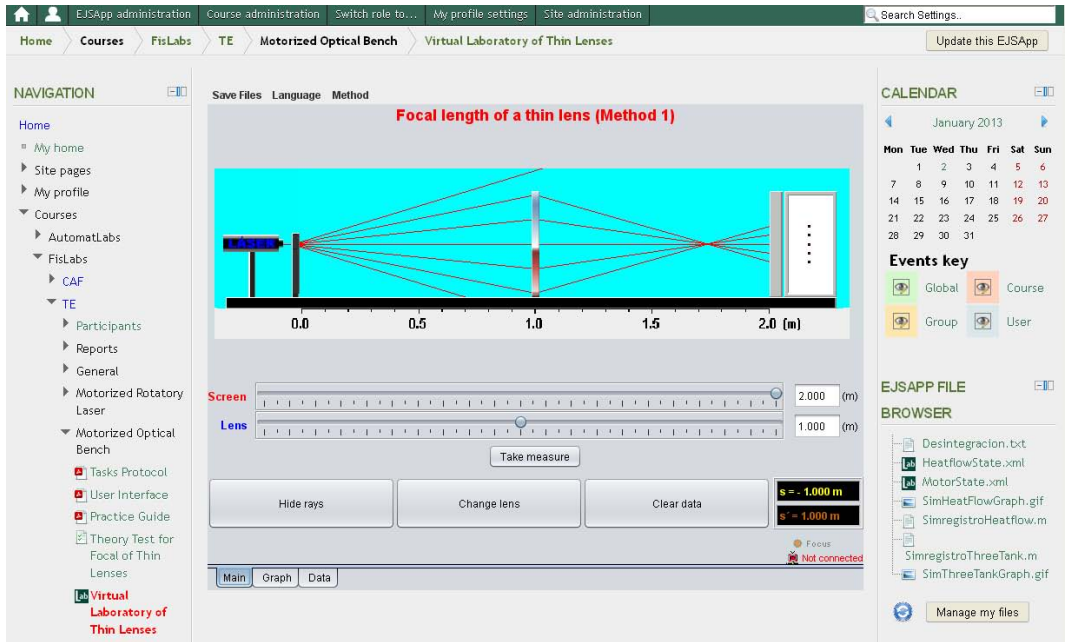


FIGURE 3.14: User interface of focal length of thin lenses working on the simulation mode.

Figure 3.15 shows: 1) the tool to perform linear regressions with the data collected in the first method and applying equation 3.5 as well as the result obtained for f and 2) the result obtained for f using the second method. It must be noted that the points plotted in the graph on 3.15(a) do not fit exactly to a line even though data were collected from a simulation. This is due to that finding the positions for the lens and the screen, in which the rays focus to one single point over the screen is, by itself, an approximate method that relies on how well you can distinguish the separation between the light points over the screen. Therefore, there are errors associated to the measurement of the focal length even in the simulation mode. Figure 3.15(b) shows the VL when the second method is being used and the two Bessel's positions are found. f is then determined by means of equation 3.6. As expected, the results given by both methods are very similar.

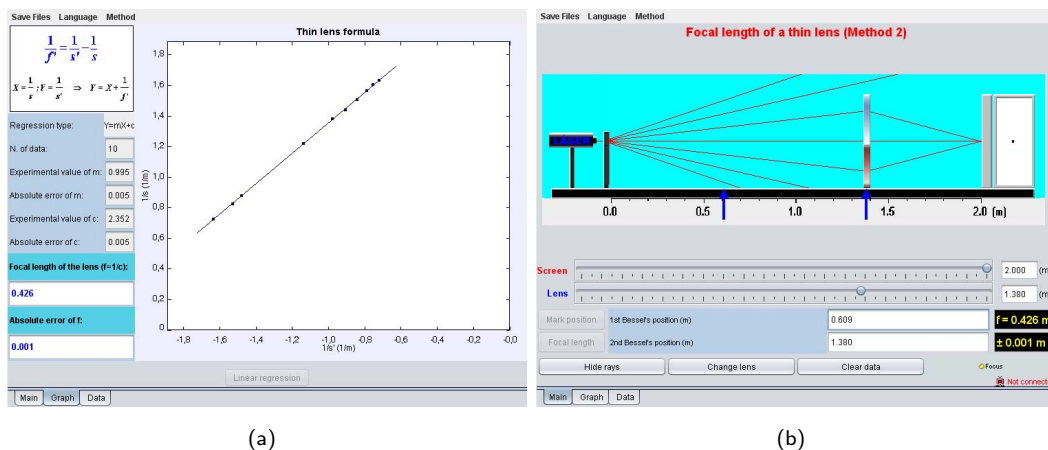


FIGURE 3.15: Determining the focal length of a thin lens when working on the simulation mode: (a) method 1 (b) method 2.

3.3.3. Remote Laboratory

Figure 3.16 shows a snapshot of this web-lab working in remote mode when using the first method for determining f . Note that the user interfaces are almost identical in simulation and remote modes, which facilitates the work of the students with the application. The webcam image on the right part of the EJS application shows several points when the lens and the screen are positioned as shown in figure 3.16. This way, students know they must keep on moving the lens before taking a new measure in this first method for determining the focal length of the lens. As in the VL mode, not only the positions of the screen and the lens are shown in two displays, but also the values of s and s' .

Figure 3.17 is the equivalent to figure 3.15 for the remote mode. The two Bessels positions are found and marked and therefore, the focal length is determined. When preparing the laboratory report, students are asked to estimate the errors in the measurements and compare the ones obtained for each method in remote mode with the ones obtained in simulation mode.

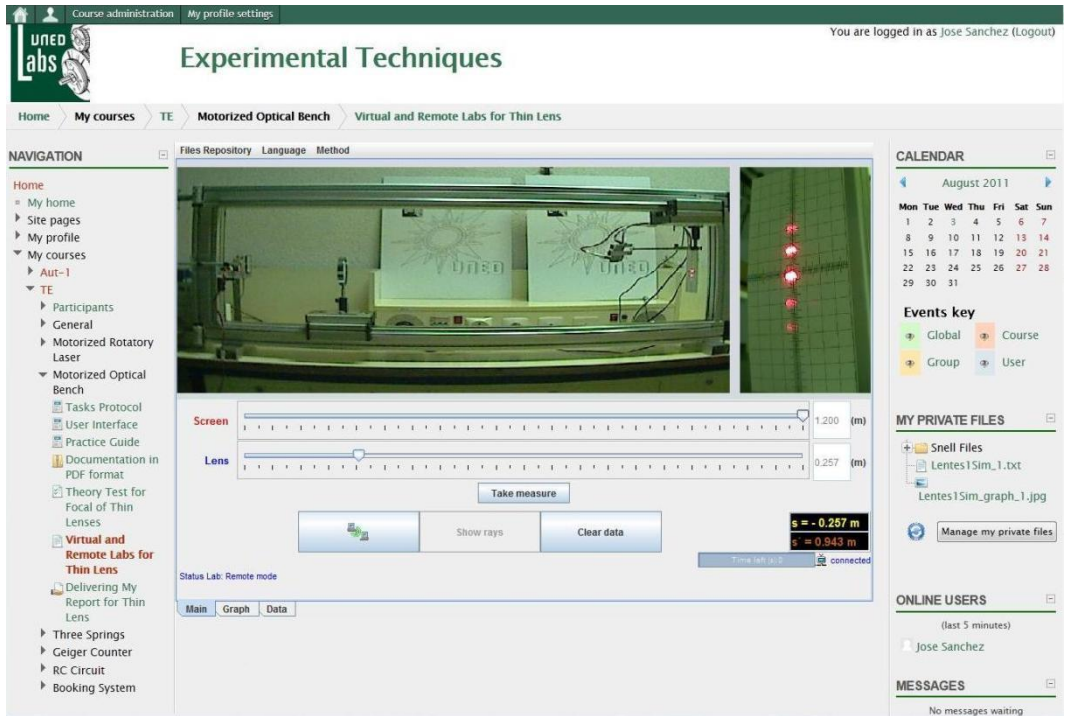


FIGURE 3.16: User interface of focal length of thin lenses working on the remote mode.

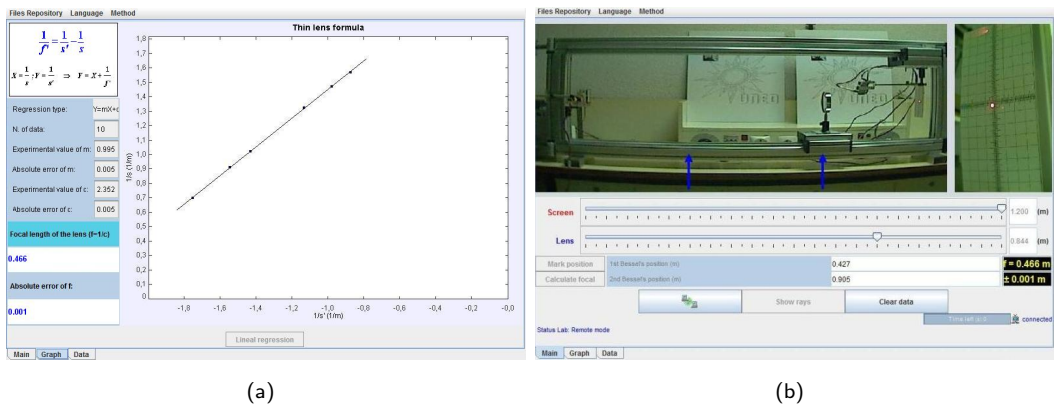


FIGURE 3.17: Determining the focal length of a thin lens when working on the remote mode: (a) method 1 (b) method 2.

Chapter 4

New Features Developed

This chapter explains what are the new features developed for VRLs created with EJS that get integrated into Moodle. First, an introduction pointing out the need for these features and their importance is presented. The two next sections describe the tools developed to achieve the inclusion of these features. Most of the work is related to the creation of Moodle plugins but there were also some required changes in EJS that had to be made.

Being modular as it is, Moodle allows the combination of any other existing plugin, such as the one presented in [79] or any other available at the Moodle plugins repository webpage, with all the tools presented in this chapter. This allows the combination of many different features to obtain a highly customized web portal in order to cover very specific needs.

4.1. Introduction

Gravier et al. [30] surveyed forty-two different remote laboratories finding that every project implements its own software architecture with no reuse. Examples

of this are [4, 22–26, 80–83]. Building a VRL from scratch is too expensive and time-consuming, so it should be avoided “reinventing the wheel”.

Thanks to the tools presented here, any existing VRL created with EJS can be automatically deployed into Moodle (with all the inherent benefits), and can also be automatically converted into a collaborative lab by just clicking a single button. Thus, VRLs developers can be focused on creating the experimentation activities, avoiding the routine and technical tasks of creating them from scratch and implementing a solution for deploying them.

These features also solve many of the issues pointed out in [32, 33, 84] when analyzing the current trends and state of the art of VRLs.

4.2. Moodle Plugins

There are different types of plugins in Moodle. Whereas EJSApp is an *activity module* and EJSApp Booking System is a *resource module*, EJSApp Collab Session and EJSApp Files Browser are *blocks*.

4.2.1. Activity/Resource Plugins

An activity is a general name for a group of features in a Moodle course. Usually, an activity is something that a student will do that interacts with other students and/or a teacher. A resource is an item that a teacher can use to support learning, such as a file or link.

Both type of plugins share the same code structure in Moodle and they are often confused since their use is very similar to the end users.

4.2.1.1. EJSApp

The EJSApp plugin was developed to support the one-click deployment of VRLs into Moodle. Figure 2.11 in Chapter 2 and figures in Chapter 3 show the visual result of a VRL integrated into Moodle using the EJSApp plugin.

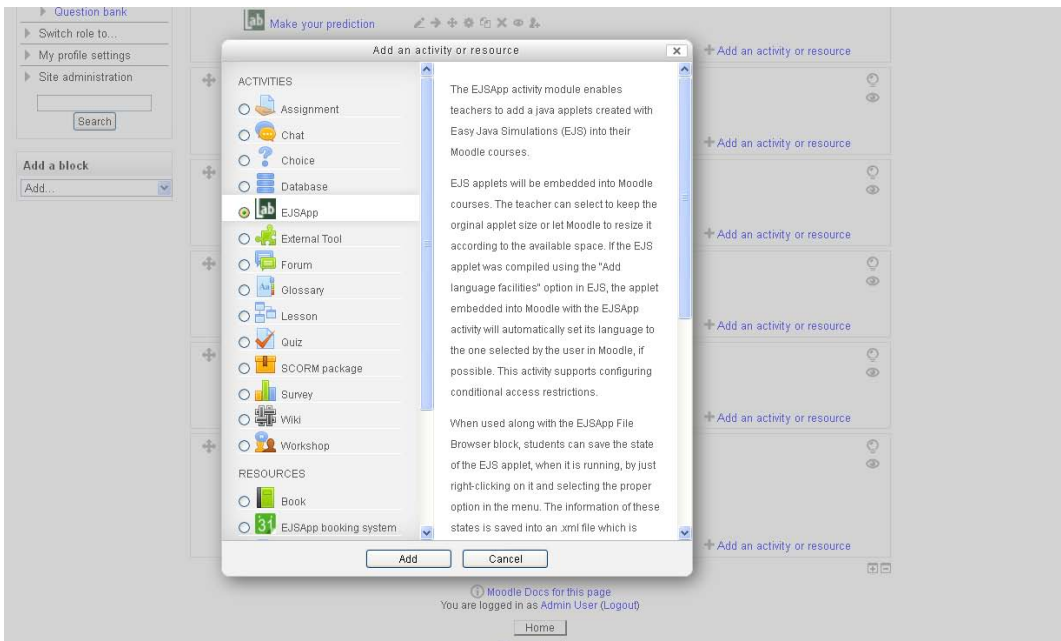


FIGURE 4.1: Adding an EJSApp activity to a Moodle course.

Figure 4.1 shows how to add an EJSApp activity into a Moodle course, which is done by simply selecting it from the list of Moodle activities and resources. Once this step is complete, the web browser loads a new window (see figure 4.2). This figure shows part of the Moodle configuration form of the EJSApp activity teachers must fulfill in order to deploy a VRL into this e-learning platform. Even though it may look complex, the process could not be more simple since there are only two required fields (those written in red). The two steps a teacher need to take in order to upload a VRL are 1) write a name for the activity (*Lab name* label in figure 4.2) and 2) upload the .jar file, created with EJS, using the File Picker

or by dragging the file and dropping it in the *You can drag and drop files here to add them.* box (*File* label in figure 4.2).

FIGURE 4.2: Configuring a VRL when adding it to a Moodle course.

However, figure 4.2 is just part of the Moodle configuration form and the EJSApp activity admit many more configuration parameters. To sum up, the EJSApp module supports:

1. **Deploying VRLs written in EJS¹**. Figure 4.2 shows part of the form EJSApp provides to add a VRL to Moodle. For doing so, only the fields in red are required. EJSApp uses the new Moodle 2 feature *File Picker*,

¹EJSApp supports the integration into Moodle of any kind of application developed with EJS (i.e., either collaborative or non–collaborative virtual or remote labs).

enabling VRLs to be uploaded not only from the user computer but also from a variety of repositories such as Dropbox, Alfresco, etc.

2. **Including descriptive text.** Figure 4.2 also shows a textbox with the label *Description* that allows teachers to add some descriptive text to appear before and/or after the VRL application. Figure 4.3 shows an example of the result.
3. **Resizing the EJS applet.** Even though the original EJS application was created with a particular size, the plugin allows multiple sizing options. While using the original size is still possible, teachers can also choose to let the EJSApp plugin to resize the applet so it occupies all the space in the mid column while it maintains the aspect ratio. The third option is to fix the applet size by hand, either maintaining the original aspect ratio or not.
4. **Controlling user access to the deployed labs.** Figure 4.4 shows the form EJSApp provides to set the VRL availability (start and end dates when the

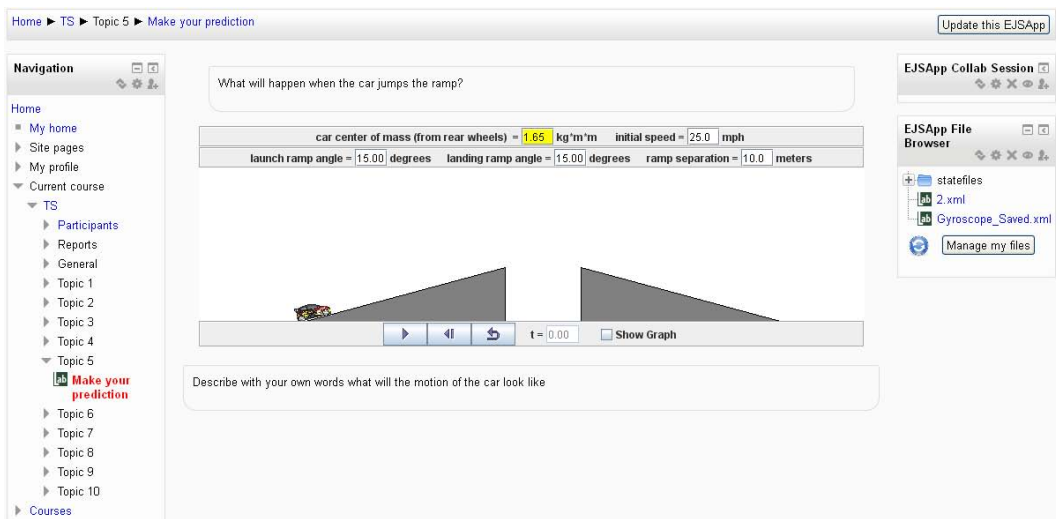


FIGURE 4.3: Descriptive text in an EJSApp activity.

Restrict access

Allow access from 22 January 2013 00:00 Enable

Allow access until 22 January 2013 00:00 Enable

Grade condition (none) must be at least % and less than %

Add 2 grade conditions to form

User field (none) contains

Add 2 user field conditions to form

Before activity can be accessed Show activity greyed-out, with restriction information

FIGURE 4.4: Controlling access conditions of a deployed lab.

.xml file with the state to be read when this EJS lab loads

File

Maximum size for new files: Unlimited, maximum attachments: 1 - drag and drop available

Add... Create folder

Files

You can drag and drop files here to add them.

FIGURE 4.5: Customizing the state of a deployed virtual lab.

VRL will be accessible to the students, minimum grade students need to get in other activities as a previous condition before having access to the VRL, etc.).

- 5. Customizing the state of VLS.** Figure 4.5 shows another File Picker box in the EJSApp Moodle form but while the one in figure 4.2 is used for uploading the .jar files that packages the VRL, this one is used for uploading .xml files that define a particular initial state of the VRL. Thanks to this option, teachers can customize the initial state and parameters of a VL and use it for many different experiments. Figure 4.6 shows the same VL used in figure 4.3 but with a different initial state. More information related to this feature can also be found in section 4.2.2.1.

FIGURE 4.6: EJSApp activity using an initial state previously customized by the teacher.

FIGURE 4.7: Configuring the access to a deployed remote lab.

6. **Configuring the access to RLS.** Figure 4.7 shows the options for configuring the access to a RL. By default, these options are hidden but a teacher can display them when needed, i.e. when a RL is uploaded to Moodle (see *Hide advanced* button, which labels changes depending whether the options are already hidden or not).

7. **Backup and restore.** EJSApp provides maintenance facilities for VRLs, packaging them into Moodle course backups and allowing to perform maintenance activities related to the deployed labs.
8. **Supervision and statistics.** Access from Moodle users to EJSApp activities are recorded and can be used for performing statistics and supervising the time students spend working in each lab.

Finally, the EJSApp plugin will automatically try to set the EJS VRL application language to the one configured in Moodle by each particular user, following their own preferences. If the language selected this way in the Moodle menu is available in the EJS application, then the language of its interface will match this selection.

In order to allow all the previous characteristics to be featured by the EJS VRLs in Moodle, the EJSApp plugin passes several parameters to the EJS applets with all the needed information. The basic structure of the code for embedding the applet into a webpage (in Moodle in our case) and passing it N parameters is given below:

```
<applet code="AppletClass.class" codebase="/path/" archive="AppletFile.jar"
  name="AppletName" id="AppletID" width="widthValue" height="heightValue">
<param name="param1Name" value="param1Value"/>
...
<param name="paramNName" value="paramNValue"/>
</applet>
```

In particular, the EJSApp plugin always needs to provide the next parameters:

```
<param name="codebase" value="codebaseValue"/>, for indicating the relative
  path in the Moodle server in which the applet file is stored.
<param name="archive" value="archiveValue"/>, for indicating the name of the
  applet file located in the previous path.
<param name="width" value="widthValue"/>, for setting the width of the applet.
```

```
<param name="height" value="heightValue"/>, for setting the height of the  
    applet.  
<param name="language" value="languageValue"/>, for setting the proper  
    language in the EJS application.
```

Thanks to two EJS native methods and the *language* parameter from above, teachers can create EJS simulations that automatically set their string labels to the language selected by each student in Moodle by just adding these two lines of code in their EJS application:

```
String lang = _getParameter("language");  
_view.setLocale(lang);
```

4.2.1.2. EJSApp Booking System

The EJSApp booking system plugin is in charge of managing the access of the students to the RLs. Figure 2.12 in Chapter 2 showed the booking system client application, and its use was explained in Section 2.2.6. This was a vision given from the student's point of view.

This section explains the use of the EJSApp booking system plugin from a teacher's point of view. Once a Moodle course has been configured with all the desired RLs, a teacher should add the EJSApp booking system Moodle resource. Figure 4.8 shows how to do it.

No further configuration is needed when adding the EJSApp booking system. The plugin automatically detects all the RLs (those configured with the *Remote experimental system?* parameter set to *Yes* in the EJSApp activity configuration page, see figure 4.7) that belong to the course in which the booking system was added and starts managing and controlling the access to these experimental activities. When a user with a role of teacher or administrator enters the booking

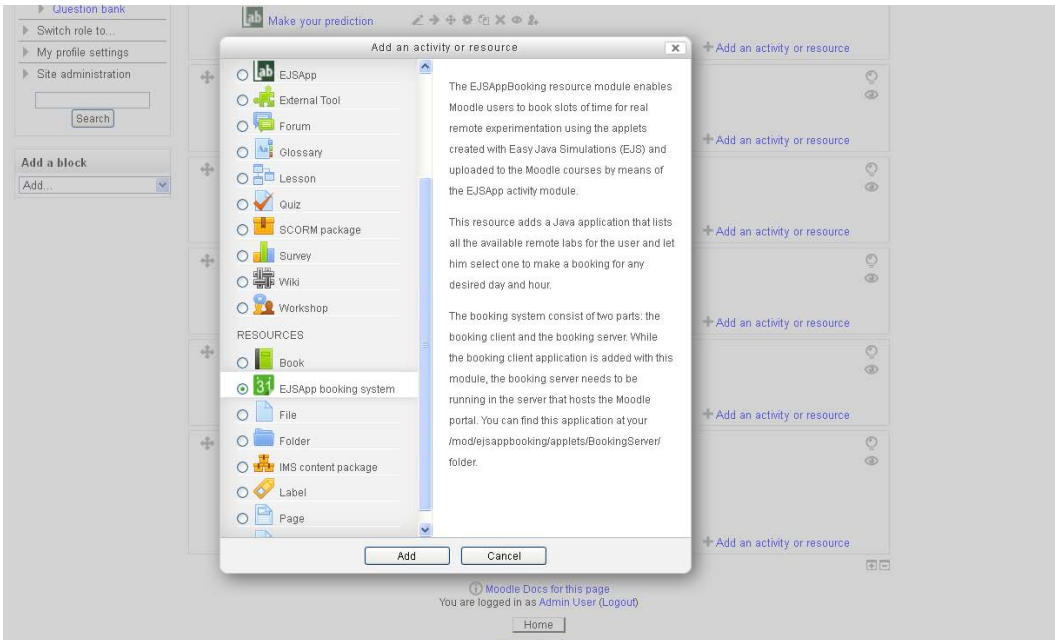


FIGURE 4.8: Adding the EJSApp Booking System to a Moodle course.

system resource, a button labeled *Manage users access* appears under the booking system client application (figure 4.9) described in 2.2.6.

By clicking on the previous button, teachers are redirected to a webpage in which they can set the each student's permission rights for making a booking in the different RLs (see figure 4.10). A teacher first selects the RL from the *Select a remote lab* drop-down menu at the top. Then, s/he ticks the checkboxes of the *Booking rights* columns for those students that should have booking rights over the selected RL and unticks (or leaves blank) those corresponding to the students that should not.

The booking system client application would only list those RLs for which a user has booking rights (configured by the teacher, see previous figure). This means a student can be allowed to make a booking for a certain RL but not for a different one, depending on his/her progress in the course.

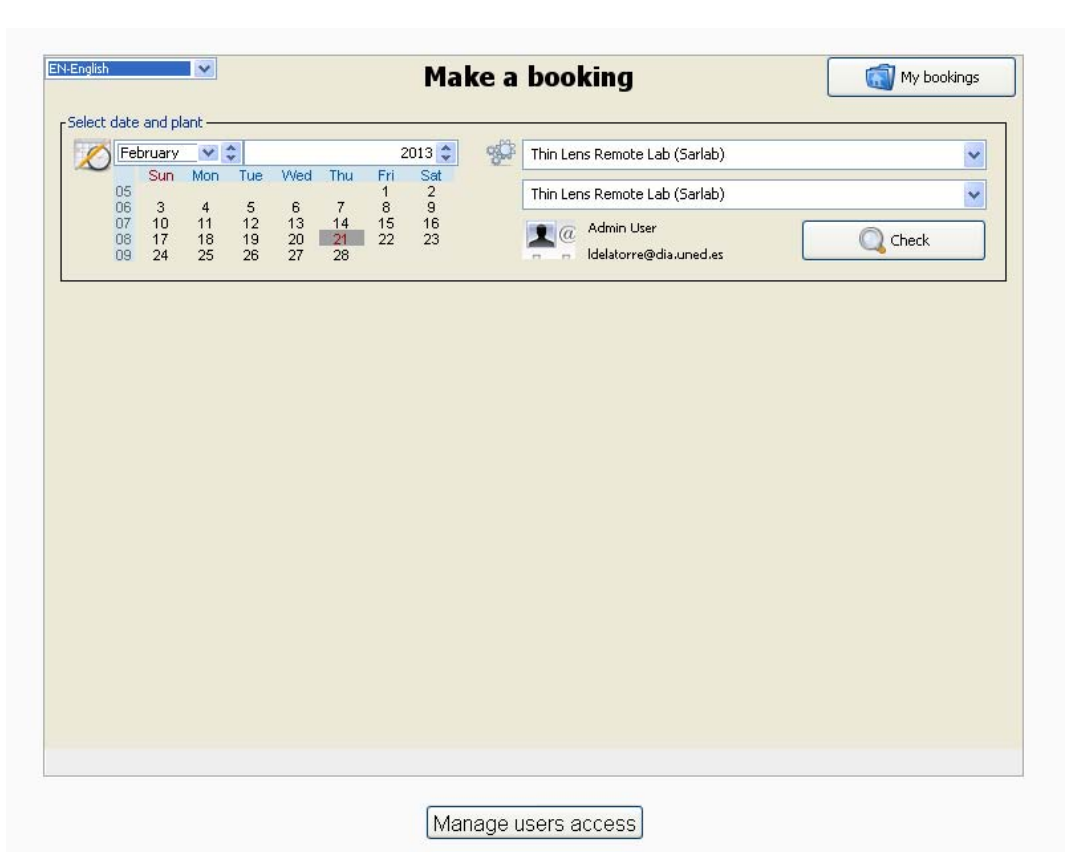


FIGURE 4.9: Booking system view for a teacher.

When a teacher gives booking rights for a particular RL to one or more students, the EJSApp Booking System plugin sends a notification to them in two different ways: 1) by means of a Moodle instant message and 2) sending an email. In both cases, the message informs the user that s/he was granted permissions to make bookings for the RL the teacher selected.

EJS applications of RLs, added to a Moodle course by means of the EJSApp activity, will always be accessible to every user if no booking system has been added to the course. Once it has, only users that have previously made a booking (using the booking system client application) for a particular RL will be able to load the EJS applet of that RL. Figure 4.11 shows what happens when a student tries to

FIGURE 4.10: Setting users permission rights.

access a RL without a booking.

4.2.2. Block Plugins

Blocks are items which may be added to the left or right column of any page in Moodle and may be visible throughout all the webpage. Due to this property, blocks are usually specific tools or gadgets (examples are calendars, instant messages, etc.) rather than activities or resources.

4.2.2.1. EJSApp File Browser

The tools presented in this work provide a framework that supports cloud storage. This is achieved thanks to this particular plugin. Data is stored online on the LMS server. Examples of stored data are intermediate running states of a VL, students' reports and data gathered during experimentation sessions. This enables a number of valuable features. For instance:

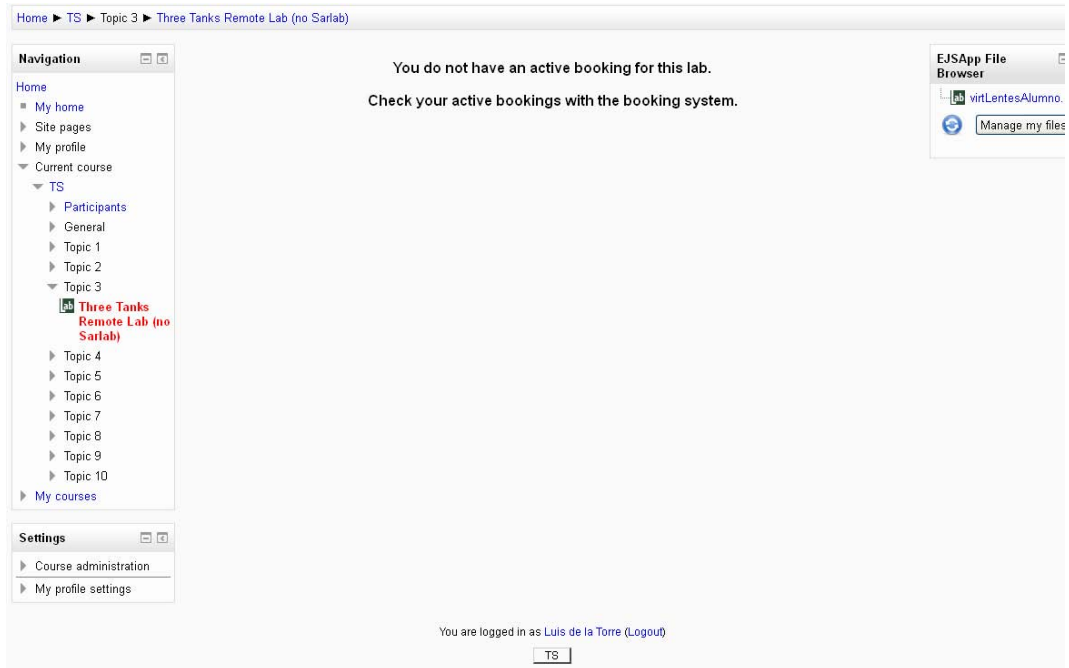



FIGURE 4.11: Trying to access a RL without a booking.

- Students' work is available anywhere/anytime.
- Students can start working with a particular VL, save the state of the simulation into the EJSApp file browser block and stop working with the application until another day. When they come back to the Web portal, they could resume the experimentation with the VL from the point they left it at, since the state was previously saved.
- A number of activities may be automatized, such as the backup of students' reports and data files, plagiarism detection and automated evaluation of students' reports, analysis of session logs to measure students' participation, etc.

The EJSApp File Browser block can be seen, for example, at figures 2.11, 3.2, 3.3, 3.5, 4.3, 4.6, 4.10 and 4.11, in the right column. Files represented in these

block with the  icon are state files (.xml), also mentioned in Section 4.2.1.1. Whenever a user clicks on one of the representations of these files, the system will redirect the web browser to the EJSApp activity (the webpage with the VRL application) that generated the file. Once the page and the EJS VRL application have been loaded, the state contained in the file clicked by the user will be read and loaded. This way, students can easily resume their work from a previous point or state. Clicking on any other file will produce the same result as doing it so with the original *My Private Files* Moodle block: i.e. it will open a dialog window for the user to choose whether to open or download the selected file.

In order to make the previous features available, the EJSApp plugin need to provide the next parameters to the EJS application:

```
<param name="context_id" value="context_idValue"/>, required info for storing  
the saved file to its proper location in Moodle.  
<param name="user_id" value="user_idValue"/>, required info for storing the  
saved file to its proper location in Moodle.  
<param name="ejsapp_id" value="ejsapp_idValue"/>, required info for storing  
the saved file to its proper location in Moodle.  
<param name="moodle_upload_file" value="moodle_upload_filePath"/>, for  
locating the EJSApp plugin php file in charge of doing the server  
operations needed to store the saved file.
```

4.2.2.2. EJSApp Collaborative Sessions

According to the constructivist learning theory [85], people generate knowledge and meaning (i) when they share their ideas and experiences and (ii) from the interaction between them. In this sense, web learning tools such as LMSs have been modified recently to include communication channels allowing user-to-user interaction in web courses. In addition, experiential learning is the process of making meaning from direct experience [86]. This approach is especially important

for scientific and technical courses, in which experimentation is a key issue for the learning process. Virtual and Remote Laboratories (VRLs) [70, 84, 87, 88] appeared to cover this necessity in distance education and also to serve as a didactical complement for traditional face-to-face courses. However, even though constructivist web learning environments and VRLs already exist, there is still a lack of: (i) convergence and interoperability between both tools [30] and (ii) real-time interaction between users when they work with VRLs [29, 30] and/or within a LMS. Several advantages could be achieved covering these two drawbacks, especially for distance education of practical experiences in technical or science subjects. This thesis presents a new approach that solves this scientific gap.

Currently, there are two different types of collaborative environments according to the moment when the student-teacher (or student-student) interaction takes place: asynchronous and synchronous [89]. The first ones allow data exchange in flexible timetables and remote access to web-based course materials to carry out activities in an asynchronous way. They use collaborative tools such as e-mail or forums for on-line communication. This is the typical approach and tools offered by most classic LMS. However, this type of communication can cause feelings of isolation in the student and hence reduces his/her motivation [80]. Furthermore, students do not receive instant feedback from their questions and cannot talk in real-time about results obtained in the learning activities. These limitations have been solved by applying synchronous technologies [90], as we have performed in the approach presented in the paper.

It is from the intersection of the three previous ideas (constructivism, experimental practice and real time interaction) that the concept of synchronous collaborative VRLs deployed into LMSs is born. The approach presented is based on this concept by means of the use of two valuable software applications for e-learning and VRL development are Moodle and EJS.

Regarding the proposal of synchronous collaborative labs presented here, Moodle provides two built-in blocks which are very helpful for user communication: *Online Users* and *Messages*. Figures 4.12 and 4.13 show a snapshot of each one. By means of the first block, users can see other connected users in order to know who can they invite for a collaborative experimentation session. Thanks to the second one, users can text each other while working together with a virtual or remote lab. An example of this is shown in Figure 4.13 where the “Admin user” is working with user “Luis de la Torre” in a collaborative session with a virtual experiment and has just received an instant message from him. In addition, Figure 4.13 shows the VRL corresponding to a heat-flow experiment [76] developed with EJS and deployed into Moodle with the EJSApp plugin.



FIGURE 4.12: *Online Users* block.

Moodle includes a good number of tools that provide asynchronous collaborative support (e.g., forums, the messaging system,...). The solution presented in this thesis takes advantage of such features by deploying VRLs into Moodle. Moreover, it enriches Moodle collaborative support by providing a new feature: the synchronous collaboration among the VRLs that are included into a Moodle course. Figure 4.14 shows an experimental session using all the provided features: virtual and/or remote experimentation, synchronous collaborative interaction and

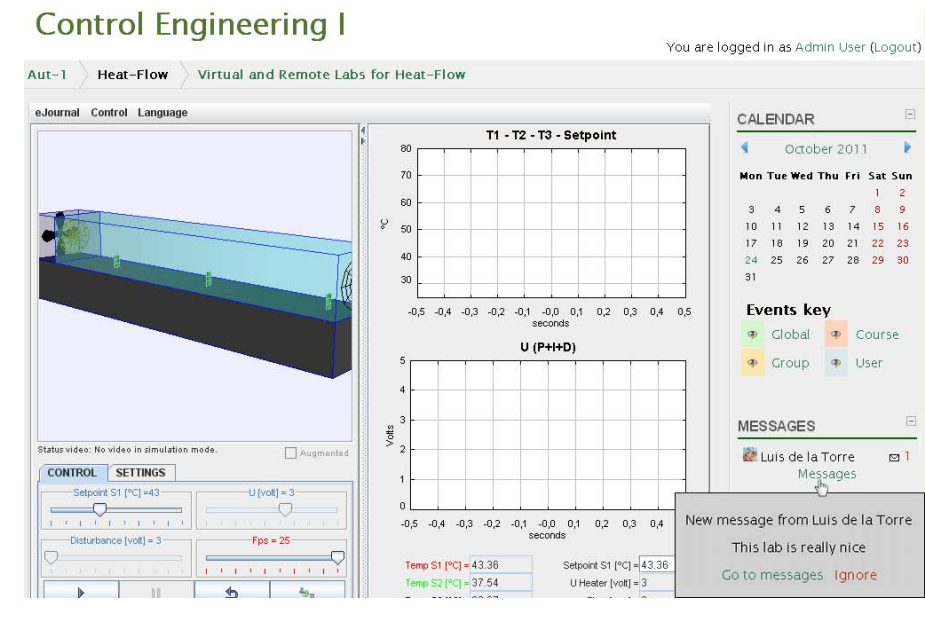


FIGURE 4.13: Messages block.

skype communication between users. Note that the laboratories of both users are synchronized, but only the session director can control the behavior of the VRL (i.e., whereas the lab control panel of the session director is activated, the one of the invited user is deactivated).

Supporting a Variety of Learning Paradigms

The EJSApp Collab Session tries to emulate a traditional classroom. Moreover, it is flexible enough to support a variety of learning paradigms, including:

1. *Reciprocal Teaching* [91]. The system supports that a user (either a teacher or a student) starts a collaborative session by inviting other users. At the beginning of the session, the user that initiated it plays the *session director* role, the remainder of the participants are *invited students*. While, initially, the session director has the control of the lab (either virtual or remote),

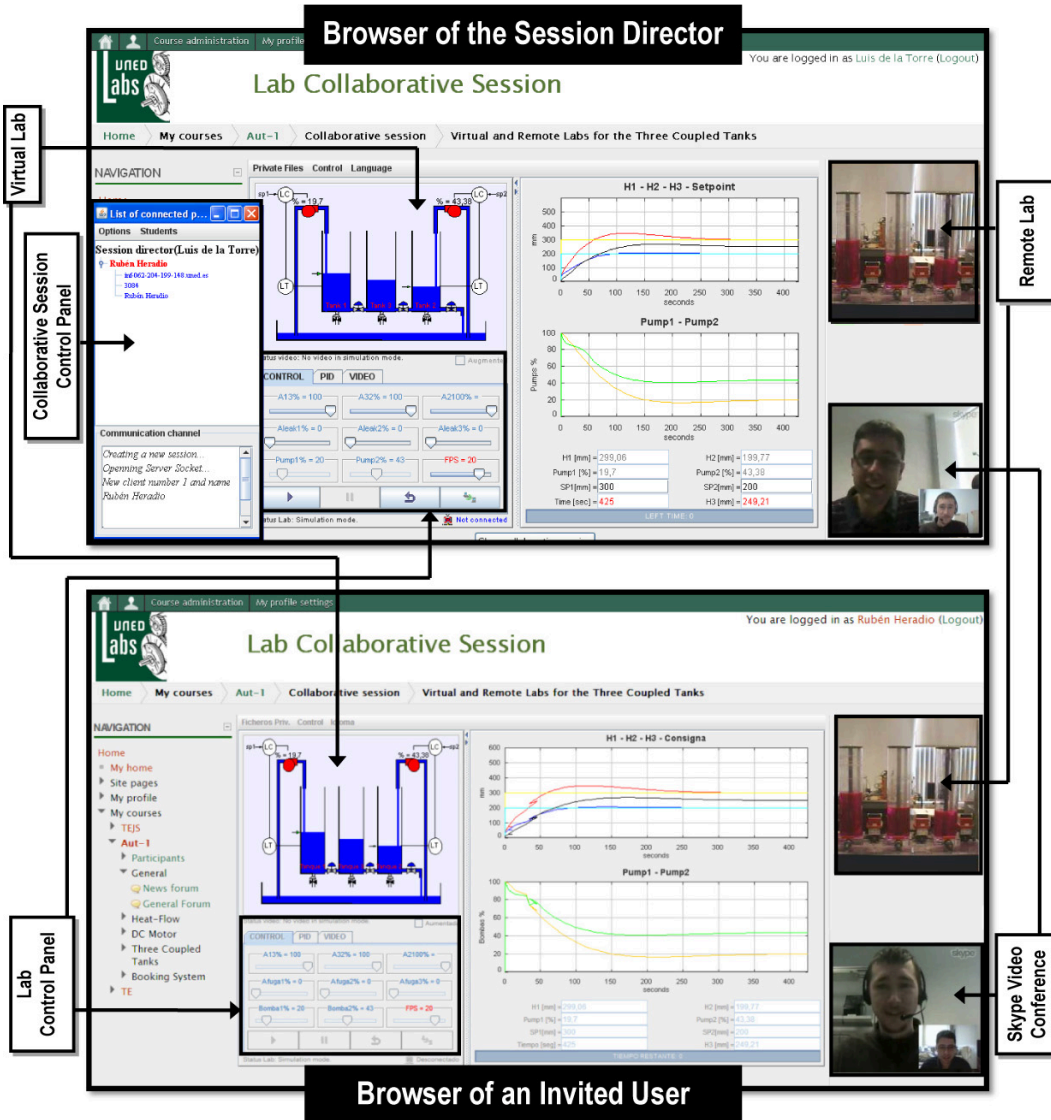


FIGURE 4.14: Example of two users participating in a collaborative experimental session.

s/he may exchange the role with any of the invited students for reciprocal teaching.

2. *Problem-Based Learning (PBL)* [92]. PBL is a student-centered approach where students work in small groups to identify what they already know, what they need to know, and how and where to access new information that may lead to the resolution of a problem. The teacher facilitates the learning process, building students confidence to take on the problem, encouraging students, while also stretching their understanding.
3. *Cooperative Work*. According to Roschelle et al. [93], “collaboration” is distinguished from “cooperation” in that cooperative work “is accomplished by the division of labor among participants, as an activity where each person is responsible for a portion of the problem solving”, whereas collaboration involves the “mutual engagement of participants in a coordinated effort to solve the problem together”.

Supporting Deep Collaboration

Many existing proposals on synchronous collaborative VRLs limit collaboration to multimedia streams coming from the equipment server and from the users [94, 95]. Thus, the only shared elements are audio, video and/or images.

Under the approach presented in this thesis, VRLs are deployed into Moodle, which has several plugins to provide synchronous sharing of audio, video and images (e.g., the Skype module²). Therefore, this proposal supports such type of synchronous collaboration. Other solutions that also provide this kind of collaboration, such as [96], do not integrate the VRL into a LMS since they need a specific and independent environment for the collaborative features to work.

²http://docs.moodle.org/22/en/Skype_module

In addition, this proposal provides a higher collaboration level. For each participant in a collaborative session, there is a running instance of the shared VRL. The state of all the instances is synchronized, i.e., whenever a participant acts over its VRL instance, the changes produced on the VRL state are propagated to the remainder of the participants' VRL instances.

Usability

This proposal provides a high level of usability³ for all the existing roles in the development, management and use of VRLs:

1. The *VRL developer* creates VRLs by using EJS. Thanks to the built EJS extension, any VRL can be automatically converted into a collaborative lab by just clicking a single button.
2. The *teacher* and the *students* participate in collaborative sessions by using an adaptive visual interface. That is, to simplify the user interface and prevent errors, the interface dynamically changes to only make available the correct options for a given state of the collaborative session. For instance, a student visualizes the “participate as an invited student” button (Figure 4.16.a) only when s/he has been previously invited to a collaborative session.

Scalability

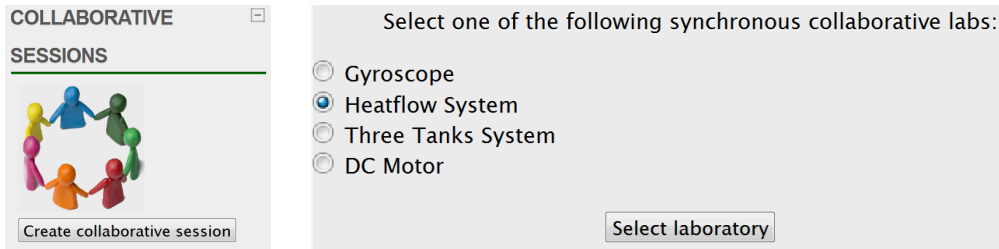
This proposal is highly scalable, i.e., many collaborative sessions may be running at the same time. A peer-to-peer (P2P) approach has been adopted in order to avoid that multiple collaborative sessions could overload the server that host the Moodle portal and the VRLs. When a collaborative session begins, users just interact with the server by downloading the applet that implements the VRL they

³Usability can be defined as “the ease of use and learnability of a human-made object”.

are going to use in the session. Then, an instance of the applet is locally run in each participant's computer. The instances communicate each other through a server-less collaboration over TCP and UDP protocols. Thus, the communication between the server and the participants' computers is limited to simple messages of session creation, session pause, session close, etc.

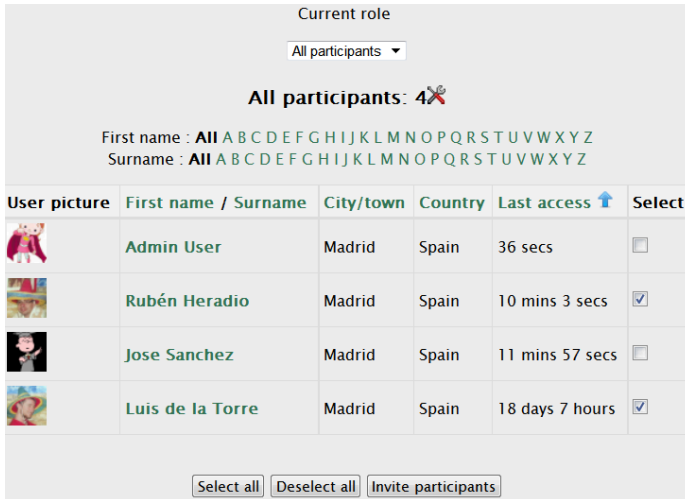
Moodle Support for Synchronous Collaborative VRLs

A fundamental issue in a synchronous collaborative system is the *floor control* [97]. This term points out how the system components share the computational resources. The main objective of our proposal is to offer a shared VRL that can be controlled in real-time by different members of a virtual class (e.g., students and teacher) and to be able to share the same experiments like in a traditional classroom. In our case, the shared VRL is composed of an applet generated with EJS. Hence, there are two main kinds of components to coordinate: one session director's applet and some invited user's applets. The session director is responsible for starting, monitoring and closing a collaborative session. Thanks to an EJS extension, the session director's applet manages in real-time the virtual class and synchronizes all the invited user's applets. She has a list of invited user's connected in the virtual session and can disconnect any invited user's at any moment. In order to have a suitable floor control, connected invited user's applets are locked and they cannot interact with the shared VRL in a first moment. They are only allowed to see in real-time what the session director is doing in the shared application. This way, the collaborative session avoids collisions of events which can cause unwanted and incoherent results. One example of this problem could be that the real equipment which controls the VRL becomes uncontrollable because of unsuitable user interactions.

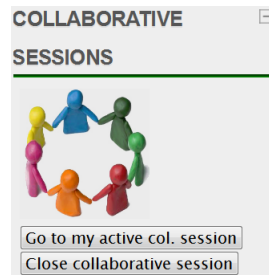


(a) Starting a synchronous collaborative session

(b) Selecting the collaborative VRL to be used within the session



(c) Selecting the session participants



(d) Closing a collaborative session

FIGURE 4.15: Synchronous collaborative session in Moodle from the session director point of view

In the following lines, the behavior of the EJSApp Collab Session block is illustrated:

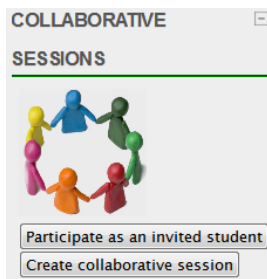
1. From the session director point of view, a collaborative session is composed of the following steps:

- a) A session is created by clicking the button “Create collaborative session” on the EJSApp Collab Session block (Figure 4.15.a). This block is always present and visible throughout the entire the web portal.

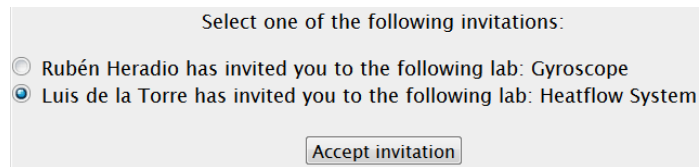
- b) When using the EJSApp Collab Session plugin, the session director selects, from all the collaborative VRLs that have been uploaded to Moodle, which one is going to be used in the session (Figure 4.15.b). It should be noted that one VRL may be shared by several collaborative sessions simultaneously.
 - c) The session director selects then the potential⁴ participants to the session he is creating (Figure 4.15.c). When the “Invite participants” button is clicked, they will be notified with (i) an e-mail and (ii) a Moodle message (i.e., using the *Messages* block introduced in Section 4.2.2.2).
 - d) The VRL is accessed in collaborative mode, i.e., the session director’s applet manages the virtual class and synchronizes all the invited user’s applets.
 - e) The collaborative session is finished by clicking the “Close collaborative session” on the EJSApp Collab Session block (Figure 4.15.d).
2. From an invited user point of view, a collaborative session is composed of the following steps:
- a) Once invited, the user clicks on the button “Participate as an invited student” on the EJSApp Collab Session block (Figure 4.16.a). To prevent misuses of EJSApp Collab Session, its graphical interface changes to show just the valid choices available to a given situation (see Figures 4.15.a, 4.15.d and 4.16.a). So, the “Participate as an invited student” button is only visible because the user has been invited to, at least, one collaborative session.

⁴i.e., the selected participants may or may not decide to participate into the session.

- b) From all the received invitations, the user selects in which session s/he wants to participate (Figure 4.16.b). Note that a course member can be invited to several collaborative sessions, but s/he can only participate in one of them at the same time.
- c) The VRL is accessed in collaborative mode.
- d) The user stops participating in the session when (i) s/he decides to leave it or (ii) when the session director closes it. In the former case, the user is free to enroll either to that session again or to any of the other current available invitations.



(a) Accepting an invitation to a synchronous collaborative session



(b) Selecting one of the available invitations

FIGURE 4.16: Synchronous collaborative session in Moodle from an invited user point of view

The next parameters are needed to make collaborative sessions work. Two different cases are considered, depending on the user's role in relation to the collaborative session.

When the applet is loaded by the director of the collaborative session:

```
<param name="is_collaborative" value="true"/>, to indicate a collaborative session is taking place.
```

`<param name="Port_Teacher" value="Port_TeacherValue"/>`, to let the applet know which port will be used for establishing the communication among the director and the invited users.

`<param name="directorname" value="directornameName"/>`, for writing the session director's name in the collaborative session control panel (see figure 4.9).

When the applet is loaded by a user invited to the collaborative session:

`<param name="is_collaborative" value="true"/>`, to indicate a collaborative session is taking place.

`<param name="Port_Teacher" value="Port_TeacherValue"/>`, to let the applet know which port will be used for establishing the communication among the director and the invited users.

`<param name="username" value="usernameName"/>`, for writing the invited user's name in the collaborative session control panel (see figure 4.9).

`<param name="IP_Teacher" value="IP_TeacherValue"/>`, to let the applet know the IP direction of the session director, needed for establishing the communication among the director and the invited users.

`<param name="MainFrame_Teacher" value="MainFrame_TeacherName"/>`, for synchronizing the state of the session director's applet with the state of the invited users' applets.

4.2.3. EJSApp External Interface. Use With Third Party Applications

This section summarizes the *EJSApp external interface*, which supports the inter-operation between EJS simulations managed into moodle by EJSApp and third-party Moodle plugins. One example is presented at the end of this section.

4.2.3.1. Importing the EJSApp external interface

The external interface is implemented by the file `ejsapp_external_interface.php`, placed inside the `external_interface`

folder of the EJSApp plugin. To import it from a third-party Moodle plugin, a developer just needs to add the code in Figure 4.17 to her own Moodle code.

```
require_once  
($CFG->dirroot.'/mod/ejsapp/external_interface/ejsapp_external_interface.php')  
;
```

FIGURE 4.17: Importing the EJSApp external interface.

Once this step is completed, a developer could use any of the functions that are presented next.

4.2.3.2. Retrieving information from EJSApp

The external interface allows third-party Moodle plugins developers to easily extract information regarding existing EJSApp instances. Such functionality helps third-party plugins to provide end users (either teachers or students, usually) with configuration menus to specify what EJS simulations should be visualized, and to customize their state and appearance.

EJSApp instances

In order to let other developers get all the available EJSApp instances in a Moodle site, the external interface offers the function `get_ejsapp_instances`. It has one optional parameter to constraint the search for a given Moodle course (instead for the entire site) and returns an array of objects with EJSApp instances information. For instance:

- Figure 4.18 gets all EJSApp instances in course with id 30. Figure 4.19 shows the corresponding output for this hypothetical course. As this figure shows, two `ejsapp` instances are available in that course.

```
$courseid = 30;
$ejsapp_instances = get_ejsapp_instances($courseid);
var_dump($ejsapp_instances);
```

FIGURE 4.18: Retrieving the EJSApp instances in the course with id 30.

- Figure 4.20 gets all EJSApp instances for all courses where the user has editing permissions.

Remark: the resulting array produced by `get_ejsapp_instances` just includes EJSApp instances of courses where the user has one of these roles: `editingteacher` or `manager`. That is, `get_ejsapp_instances` guarantees that a teacher cannot retrieve information about EJSApp instances that belong to courses where she does not have editing permissions.

Here is a brief description of the meaning of the fields given by `get_ejsapp_instances` for each EJSApp instance:

- *id* is a number that uniquely identifies the considered EJSApp activity instance.
- *course* is the id for the course the EJSApp activity instance belongs to.
- *name* is the name given to the EJSApp activity instance by the teacher who created it.
- *intro* contains any text written in the *Description* field of the EJSApp instance form shown in Figure 4.2.
- *introformat* is a Moodle representation of the format used for the text in the *intro* field.
- *appwording* contains any text written in the *Wording* field of the EJSApp instance form.

```

array
  0 =>
    object(stdClass) [350]
      public 'id' => string '83' (length=2)
      public 'course' => string '30' (length=2)
      public 'name' => string '3 tanks' (length=7)
      public 'intro' => string 'Three Tanks Virtual Lab.' (length=0)
      public 'introformat' => string '1' (length=1)
      public 'appwording' => string '' (length=0)
      public 'appwordingformat' => string '1' (length=1)
      public 'timecreated' => string '1354202230' (length=10)
      public 'timemodified' => string '0' (length=1)
      public 'applet_name' => string 'virtThreeTanks' (length=18)
      public 'class_file' => string
        'Threetank.virt3Tanks_pkg.virt3TanksApplet.class' (length=63)
      public 'codebase' => string '/ejsapp/jarfiles/30/83/' (length=27)
      public 'mainframe' => string 'MainFrame' (length=9)
      public 'is_collaborative' => string '0' (length=1)
      public 'applet_size_conf' => string '2' (length=1)
      public 'preserve_aspect_ratio' => string '1' (length=1)
      public 'custom_width' => 600
      public 'custom_height' => null
      public 'is_rem_lab' => string '0' (length=1)
      public 'height' => string '523' (length=3)
      public 'width' => string '667' (length=3)
  1 =>
    object(stdClass) [347]
      public 'id' => string '84' (length=2)
      public 'course' => string '30' (length=2)
      public 'name' => string 'rem servo' (length=9)
      public 'intro' => string 'Servo Motor Remote Lab.' (length=0)
      public 'introformat' => string '1' (length=1)
      public 'appwording' => string '' (length=0)
      public 'appwordingformat' => string '1' (length=1)
      public 'timecreated' => string '1354202283' (length=10)
      public 'timemodified' => string '0' (length=1)
      public 'applet_name' => string 'remServoUNED' (length=12)
      public 'class_file' => string
        'Servo.remServo_pkg.remServoApplet.class' (length=54)
      public 'codebase' => string '/ejsapp/jarfiles/30/84/' (length=27)
      public 'mainframe' => string 'MainFrame' (length=9)
      public 'is_collaborative' => string '0' (length=1)
      public 'applet_size_conf' => string '0' (length=1)
      public 'preserve_aspect_ratio' => string '0' (length=1)
      public 'custom_width' => null
      public 'custom_height' => null
      public 'is_rem_lab' => string '1' (length=1)
      public 'height' => string '558' (length=3)
      public 'width' => string '750' (length=3)

```

FIGURE 4.19: Result of executing Figure 4.18.

```

$ejsapp_instances = get_ejsapp_instances();

```

FIGURE 4.20: Retrieving all EJSApp instances for the whole Moodle site.

- *appwordingformat* is a Moodle representation of the format used for the text in the *appwording* field.
- *timecreated* informs about the date and hour in which the activity was created.
- *timemodified* informs about the date and hour in which the activity was last modified.
- *applet_name* is the applet file's name.
- *classfile* is the main .class file of the applet.
- *codebase* gives the relative path to the applet in the Moodle server.
- *mainframe* is used for the collaborative features.
- *is_collaborative* indicates if the EJSApp activity instance can work in a synchronous collaborative (1) way or not (0).
- *applet_size_conf* determines how the applet should be scaled by Moodle: 0 means it should be maximized while maintaining the original aspect ratio, 1 means the teacher can select the size of the applet by hand, and 2 means the applet should be displayed with its original size.
- *preserve_aspect_ratio* indicates if the aspect ratio should be preserved (1) or not (0) when the previous parameter (*applet_size_conf*) is set to 1.
- *custom_width* contains a value different of null when the teacher has selected a custom width for the applet.
- *custom_height* contains a value different of null when the teacher has selected a custom height for the applet.

- *is_rem_lab* indicates whether the EJSApp activity instance contains a RL (1) or not (0).
- *height* is the original height of the applet.
- *width* is the original width of the applet.

Getting the size of an EJSApp instance

For security reasons, `get_ejsapp_instances` is rather restrictive. Thus, students cannot use it to retrieve information about EJSApp instances. However, to draw correctly an EJSApp instance, it is sometimes useful to know its size. Such functionality is supported by `get_ejsapp_size`, which does not have the access limitations previously explained. This function receives the identifier of the EJSApp instance and returns an object with its width and height.

For instance, Figures 4.21 and 4.22 show an example of retrieving the size of the EJSApp instance with id 85.

```
$ejsapp_id = 85;
$size = get_ejsapp_size($ejsapp_id);
var_dump($size);
```

FIGURE 4.21: Retrieving the size of the EJSApp instance with id 85.

```
object(stdClass) [523]
  public 'width' => string '554' (length=3)
  public 'height' => string '540' (length=3)
```

FIGURE 4.22: Result of executing Figure 4.21.

Getting EJSApp state files

As seen in Section 4.2.1.1, teachers may optionally specify an initial state file for an EJS simulation through the EJSApp form. In addition, students can store additional state files of the same simulation by selecting the option `State input\output⇒Save state` on the contextual menu that emerges by clicking the mouse right button on an EJS applet (see Section 4.2.2.1). Such additional state files are stored in the users' private files area (the EJSApp File Browser block).

The external interface allows third-party Moodle plugins developer to get all the state files related to an EJSApp (both the ones specified by the teachers and those saved by the students) thanks to the function `get_ejsapp_states`. This function receives the identifier of an EJSApp instance and returns an array of objects with information about:

1. The initial state file of the EJSApp instance (if it exists).
2. All the state files associated to the EJSApp instance that are stored in the users' private files area.

For instance, Figures 4.23 and 4.24 show an example of retrieving the current user's state files associated to the EJSApp instance with id 85.

```
$ejsapp_id = 85;  
$state_files = get_ejsapp_states(85);  
var_dump($state_files);
```

FIGURE 4.23: Retrieving the current user's state files associated to the EJSApp instance with id 85.

```
array
  0 =>
    object(stdClass) [301]
      public 'state_name' => string 'Gyroscope_ruben.xml' (length=19)
      public 'state_id' => string
        '77/mod_ejsapp/private/0/Gyroscope_ruben.xml' (length=43)
  1 =>
    object(stdClass) [300]
      public 'state_name' => string 'Gyroscope_Variables.xml' (length=23)
      public 'state_id' => string
        '327/mod_ejsapp/xmlfiles/85/Gyroscope_Variables.xml' (length=50)
```

FIGURE 4.24: Result of executing Figure 4.23

4.2.3.3. Drawing an EJS simulation

Function `draw_ejsapp_instance` generates the HTML and javascript code to visualize an EJSApp simulation. That is, using the functions described in Section 4.2.3.2, third-party plugins can retrieve information from EJSApp, process it, and show it to their users as configuration menus. Then, `draw_ejsapp_instance` is used to print the EJS simulations according to the user preferences.

`draw_ejsapp_instance` has the following parameters:

1. `$ejsapp_id`: identifier of the EJSApp instance which is going to be visualized.
2. `$state_file`: optional string parameter that identifies an state file (see the `state_id` field in Figure 4.24, lines 5 and 10). If it is specified, the EJSApp instance is printed in the state described by the `$state_file` parameter, elsewhere, it values `null` by default and the EJSApp instance is printed in the initial state file configured in the EJSApp form.
3. `$width` and `$height`: optional int parameters that set the size of the printed EJS simulation. If they are not specified, the simulation will be printed as configured in the EJSApp form.

```
// Draw ejsapp with id=85 following its form configuration
$code_1 = draw_ejsapp_instance(85);
echo $code_1;

echo '<br/>'; // write an end of line

// Draw ejsapp with id=85, $state_files[0]->state_id, width=500 and height=300
$code_2 = draw_ejsapp_instance(85, $state_files[0]->state_id, 500, 300);
echo $code_2;
```

FIGURE 4.25: Drawing EJS simulations.

Remark: `draw_ejsapp_instance` does not require the EJSApp activities to be visible. In other words, it can print EJSApp instances hidden by the course administrators.

4.2.3.4. Application Example: IPAL

IPAL⁵ (In-class Polling for All Learners) is a Moodle plugin, developed by Dr. William Junkin at the Eckerd College, that enables in-class polling, allowing students to respond by using a mixture of devices such as clickers, iPads or laptops.

Working in collaboration with Dr. Junkin and thanks to the use of the EJSApp external interface, IPAL was adapted to be able to use EJS applications which are embedded into Moodle courses as EJSApp activities. This way, IPAL's in-class polling can now show EJS simulations and so, these Moodle embedded resources can be used for creating these questionnaires.

Figure 4.27 shows the result of combining the EJSApp and the IPAL plugins.

4.3. EJS Additions

Section 4.2 described the plugins developed for Moodle and the features they introduce into the deployed VRLs. This section describes the changes made to the

⁵<http://www.compadre.org/ipal>

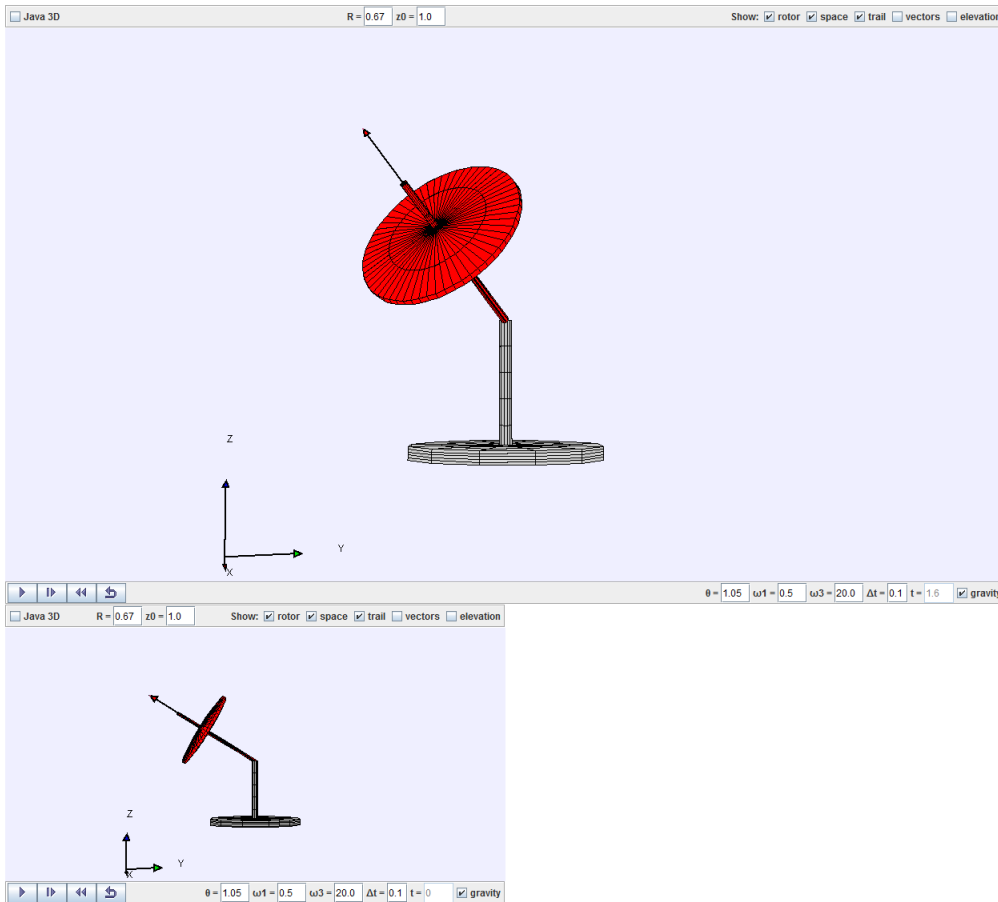
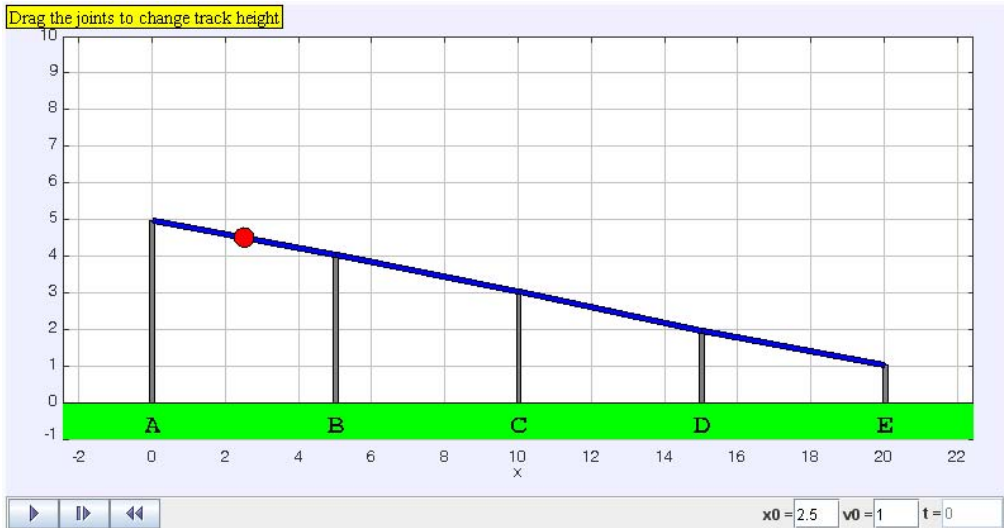


FIGURE 4.26: Result of executing Figure 4.25.

EJS program in order to let the created VRLs to take advantage of the previous plugins and the features they offer.

Every time a .jar file is created using EJS, the program adds five lines to its MANIFEST.MF file. These lines are similar to the ones shown in figure 4.28 and always share the same structure.

While the first and last lines are explained in the next section, the other two can be clarified already. *Applet-Height* and *Applet-Width* store the information of the applet's dimensions. These are the original dimensions given to the applet by



Which best describes the plot of velocity versus time for the ball rolling down the track?

- horizontal line
- line sloping up
- line sloping down
- parabola-- concave up (like a "U")
- parabola-- concave down (like an "n")

FIGURE 4.27: An EJSApp activity in Moodle (EJS simulation) combined with IPAL so it can be used for in-class polling.

```
Main-Frame: mainFrame
Applet-Height: 543
Applet-Width: 680
Is-Collaborative: false
```

FIGURE 4.28: Information contained in MANIFEST.MF file.

its author and they might be used by the EJSApp plugin depending on the resizing options selected by the teacher in Moodle (see Figure 4.2 and section 4.2.1.1).

4.3.1. EJS Support for Synchronous Collaborative VRLs

The EJS extension, originally developed by Prof. Carlos A. Jara and then adapted by himself, in collaboration with Luis de la Torre and Prof. Rubén Heradio,

to support the EJSApp Moodle plugins, provides the session director with the control panel shown in Figure 4.29. It includes a list of the invited users connected to the collaborative session. For example, Figure 4.29 shows that “Luis de la Torre” is the session director and, “Rubén Heradio” and “Jose Sánchez” are the invited users. These names are obtained by the EJS applet from the *username* and *directorname* parameters passed to the applet by the EJSApp plugins.

Using the list shown in Figure 4.29, the session director can perform the following tasks:

1. Supervising which users have already connected to the collaborative session in order to call the roll before starting the experimentation.
2. Disconnecting any invited user at any moment.
3. *Assigning the chalk* to an invited user. With this feature, the session director gives permission to control the shared VRL to a specific invited user, by selecting him from the list. The chalk only enables a student to manage the VRL, but not the collaborative session.

Figure 2.2 depicts the communication framework that underlays the collaborative sessions. When a session begins, users just interact with the Moodle server

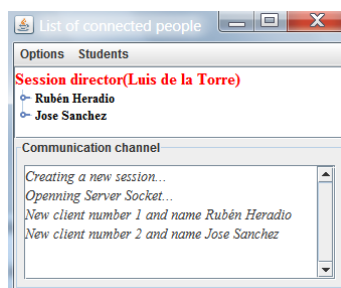


FIGURE 4.29: Control panel of the collaborative session.

by downloading the applet that implements the VRL they are going to use in that session (see dashed lines in Figure 2.2). On the other hand, users participating in a session interact each other through a server-less collaboration over TCP and UDP protocols (see solid lines in Figure 2.2). Thus, the communication framework proposed here supports multiple simultaneous sessions without overloading the Moodle server.

Invited users' applets are connected directly to the session director's applet in a P2P centralized overlay network, which is done using the information provided by the *Port_Teacher* and *IP_Teacher* parameters, given by the EJSApp plugins. In contrast with server-based approaches, our e-learning system is focused in a server-less architecture. This communication method avoids delays caused by the server processing in the data flow because the communication engine is embedded in the Java applets downloaded by the users. In addition, the number of network connections can be substantially decreased because the session director's applet can manage the session, the floor control, and the data exchange having higher control over the invited user applets. As stated, the P2P network is centralized around the session director's applet. This last application is the central node of the collaborative class and contains a multithread communication module which manages the synchronization of all the applets that compose the shared VRL. Invited users' applets are connected to the central node over TCP and UDP sockets performing a centralized network.

To synchronize in real-time all the applets connected to the virtual class a method based in Java object tokens [97] is used. Java object tokens are small update messages which contain a String object that defines the action to be performed by other applets of the same session. The small amount of sent information optimizes network utilization and reduces the connection delay.



FIGURE 4.30: Adding collaborative support in EJS.

Since all the applets must be in the same state at any time, it is necessary to synchronize them. This is what the information contained in the *MainFrame_Teacher* parameter (which is passed to the applet and read by the EJS program contained in the applet) is used for. The developed communication framework provides a transport service suitable for updating all the data: a TCP-based channel for reliable messages and a UDP-based channel for fast messages. The TCP channel is used to update all the variables of the application because the transmission of the values needs the reliability of an ACK-based protocol. The UDP channel is used to transmit the small changes in the user-interface and this requires to be quickly updated in the rest of the applets.

While those two lines with *Main-Frame* and *Is-Collaborative* are always written in the MANIFEST.MF file of the EJS applet, the first one is only used by the EJSApp Moodle plugin when the second one shows a true value instead of a false value. This happens when the author of the EJS application marks the option *Add support for collaborative applets* in EJS (from the EJS *Information* panel, see figure 4.30) before creating the applet .jar file.

4.3.2. Saving Files to Moodle

There are four ways enabled for sending data from an EJS application to the EJSApp File Browser block in Moodle. All of them use EJS predefined (ready-to-use) methods that were modified to take into account this possibility.

1. `_saveText(String filename, String filecontent)`, used for saving a text (the content, defined in the *filecontent* parameter, must be defined by the user) into a file (its name is defined in the *filename* parameter).
2. `_saveImage(String filename, String elementName)`, used for saving an image contained in the *elementName* element of the EJS view to a file.
3. `_saveState(String filename)`, used for saving the complete list of variables that define the current state of the simulation into an `.xml` file.
4. `_saveVariables(String filename, java.util.List<String> variablesList)`, used for saving only some selected variables of the current state of the simulation (also into an `.xml` file).

4.3.3. Loading Files from Moodle

Two predefined EJS methods were also modified to offer a couple of ways for reading data (`.xml` files previously stored in the EJSApp File Browser) from an EJS applet embedded into Moodle:

1. `_readState(null)` is used to read the complete list of variables of the simulation from an `.xml` file.
2. `_readVariables(null, java.util.List<String> vars)` is used to read a selected list of variables of the simulation from an `.xml` file.

4.4. Information Exchanged Between Moodle and EJS

The purpose of this section is to summarize all the information exchanged between the EJSApp Moodle plugins and the EJS applications. Figure 4.31 illustrates all these communication links. Solid lines going from Moodle to EJS represent information provided by Moodle to the EJS applets in form of parameters. Solid lines going from EJS to Moodle represent information read by Moodle from the applet when the file is loaded, which happens when a new EJSApp activity is created or edited. Dashed lines going from Moodle to EJS represent information read by EJS from Moodle which only happens when the user of the applet orders it (reading data files). Finally, dashed lines going from EJS to Moodle represent information

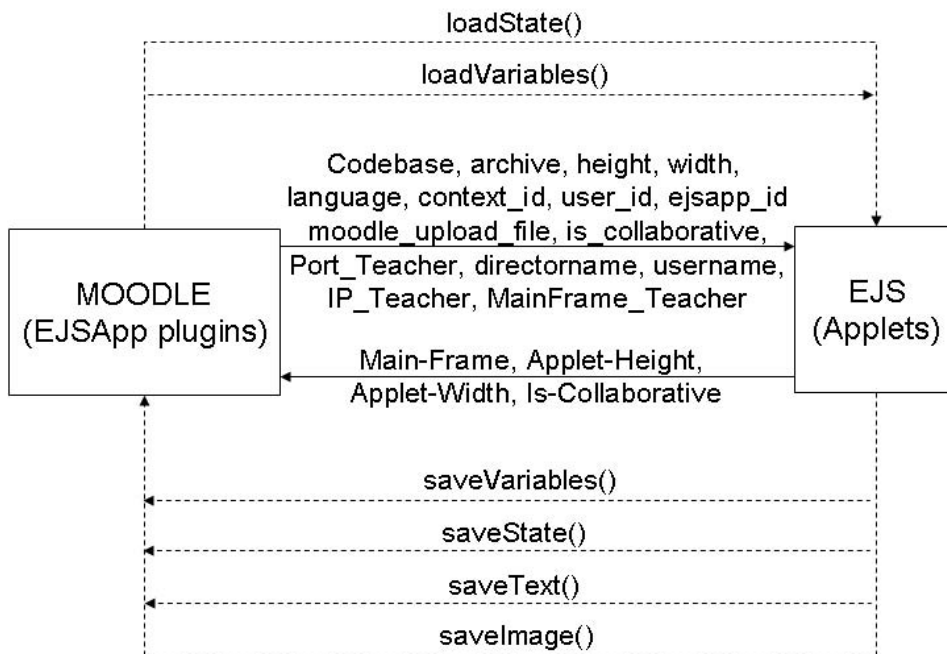


FIGURE 4.31: Information exchanged between Moodle and EJS applications.

which is stored in Moodle from EJS by command of the user of the applet (saving files).

Chapter 5

Step-By-Step Process for Preparing and Deploying a VRL

This chapter is meant to give an insight of how easy is to create a new VRL and deploy it with Moodle when using the proposed framework. With this purpose, the chapter is divided into two main sections, each of one explains the details of one of the two parts of the process (creation and deployment).

The VRL from Section [3.2](#) is used as an example to illustrate the whole process step by step.

5.1. Preparing the VRL

When preparing a VRL there are three very different parts the teacher must face. The first one is the creation of the VL and the second one is the creation of the RL. For the VL, the single use of EJS is sufficient. However, for the RL, an experimental system along with the LabView or Matlab program that controls its hardware, is also needed.

5.1.1. Using EJS to prepare the VL application

The first step is to create a new simulation or VL, or download an existent one. Places such as [19, 98] offer a lot of EJS simulations any user can download for free. Since they can download the EJS source code (and not only the applet application), users can even customize the simulations to their own taste.

Figure 5.1 shows the *Variables* and *Evolution* tabs of the EJS program used for the Hooke's law VL, which usually are the very basic of any EJS application. Of course, the *View* has also to be built and many other things (such as *Custom*) methods may be needed depending on the simulation.

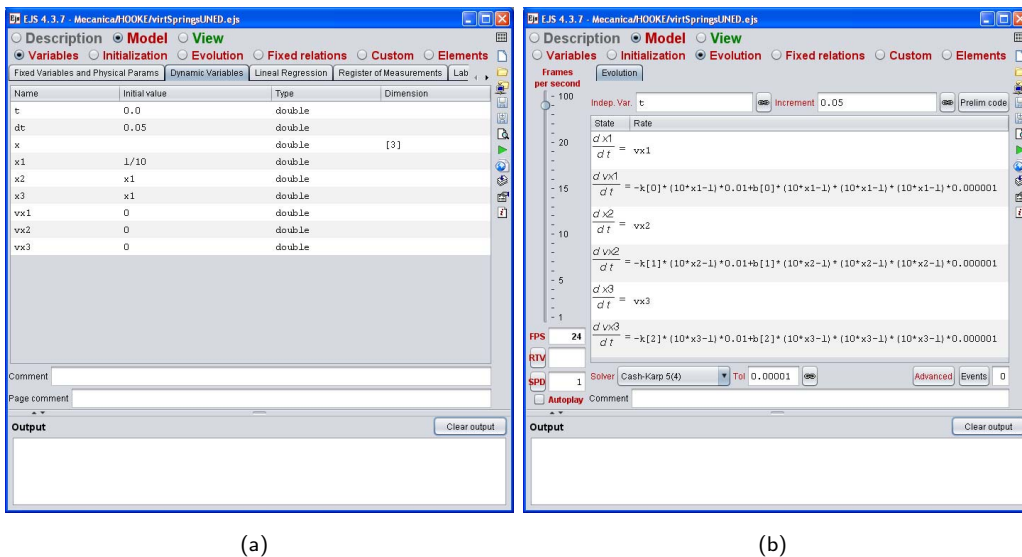


FIGURE 5.1: Creating a VL with EJS.

In any case, in order to be sure that the newly created EJS application could be correctly embedded into Moodle, a user only needs to compile it with version 4.37 (build 121201 or later) of EJS.

If the EJS applet was downloaded from the Internet, the application may be already prepared to work with EJSApp and Moodle. There is a simple way to check

this, which consists in opening the .jar file downloaded and read the /META-INF/MANIFEST.MF file. If the lines from figure 5.2 can be found, then the application can be used with EJSApp. If not, a recompilation with an updated version of EJS is required.

```
Applet-Height: <int>
Applet-Width: <int>
```

FIGURE 5.2: Checking whether an EJS applet is compatible with EJSApp or not. If these lines are present in the MANIFEST.MF file, it is.

5.1.2. Using LabView to create the RL controlling program

In this particular example, the RL is controlled with LabView. Next figure shows the complete LabView program used for controlling the Hooke's law experimental system. This program is placed in the *Lab PC* where the experimental system is located and it is invisible to the end users, who only face the EJS application.

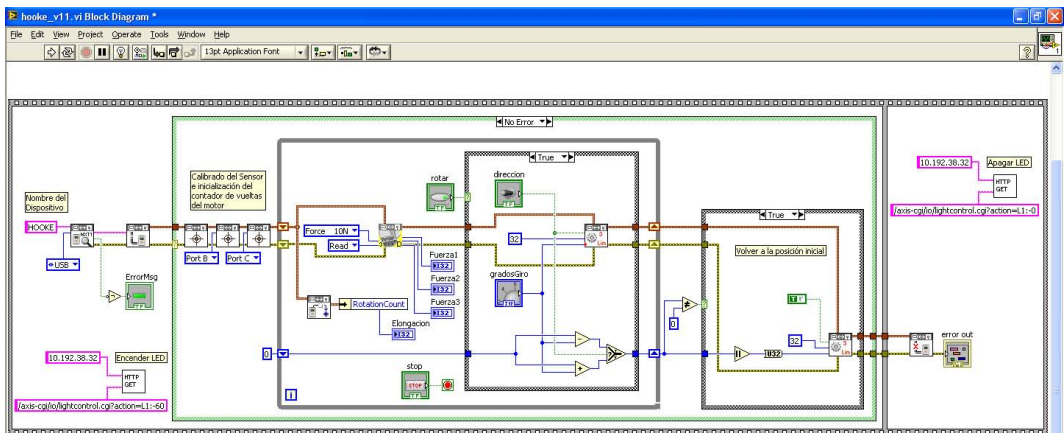


FIGURE 5.3: Creating a LabView program for a RL.

To sum up, the most important part of a LabView program for a RL are the *controls* and *indicators*. Controls and indicators are variables that serve to modify

or to visualize the value of a certain parameter, respectively. These LabView variables must be exchanged with the EJS variables in order to offer the end user a way to act over the RL controlling program as well as a way to read the results of these actions.

In the previous example these are the controls:

- *gradosGiro*, an integer value that orders the motors that pull the springs to rotate a certain number of degrees.
- *rotar*, a boolean value that tells the motors whether to rotate *gradosGiro* or not.
- *direccion*, a boolean value that configures the sense of rotation of the motors (clockwise or counterclockwise).

And the indicators:

- *Fuerza1*, an integer value that gives the force measured by the first force sensor (spring 1).
- *Fuerza2*, an integer value that gives the force measured by the second force sensor (spring 2).
- *Fuerza3*, an integer value that gives the force measured by the third force sensor (spring 3).
- *Elongacion*, an integer value that measures the stretching of the springs.

5.1.3. Using EJS to prepare the RL application

Once the previous step is completed, the EJS RL application needs to be prepared to exchange the information (controls and indicators) with the LabView

program. This is done by importing the JiL library and including some lines of code. Following the previous example, next figure shows how this is done for Hooke's law experimental system and RL.

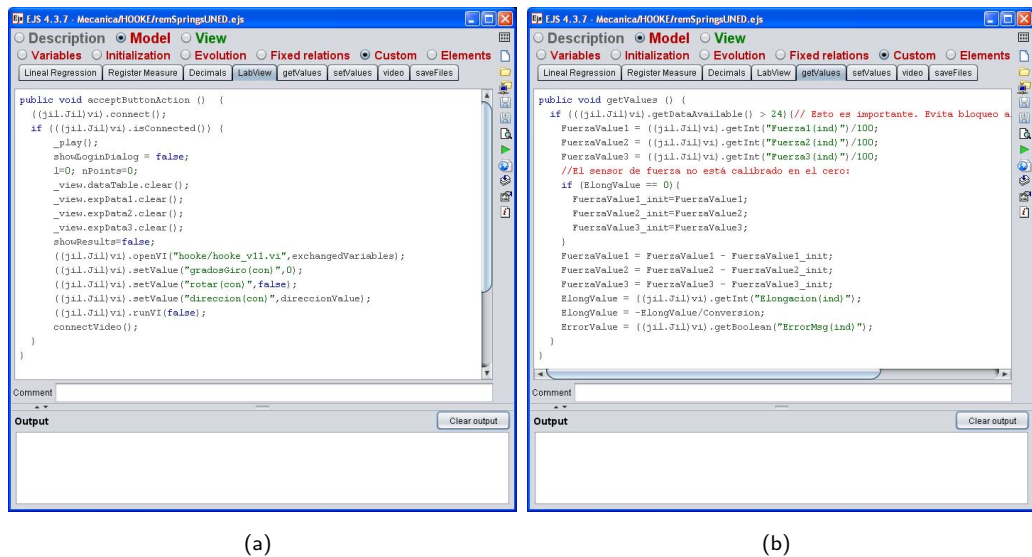


FIGURE 5.4: Creating a client user interface for a RL with EJS.

In 5.4(a), the JiL *openVI()* method is used to open the appropriate LabView program (see previous section). Also, the *setValue()* method is used for setting the initial values of the controls.

In 5.4(b), the JiL *getInt()* method is used for reading the values of the indicators.

Finally, if SARLAB is going to be used with the RL (see section 2.1), the EJS application needs to use the SARLAB EJS element, developed by Marco A. Marquez. Adding and configuring this element in EJS is extremely easy. The only parameters that must be filled in are the IP direction of the SARLAB server, the port for establishing communication with it, and the practice identifier in SARLAB. Next figure illustrates this step. If SARLAB is not used, then this whole last step can be skipped.

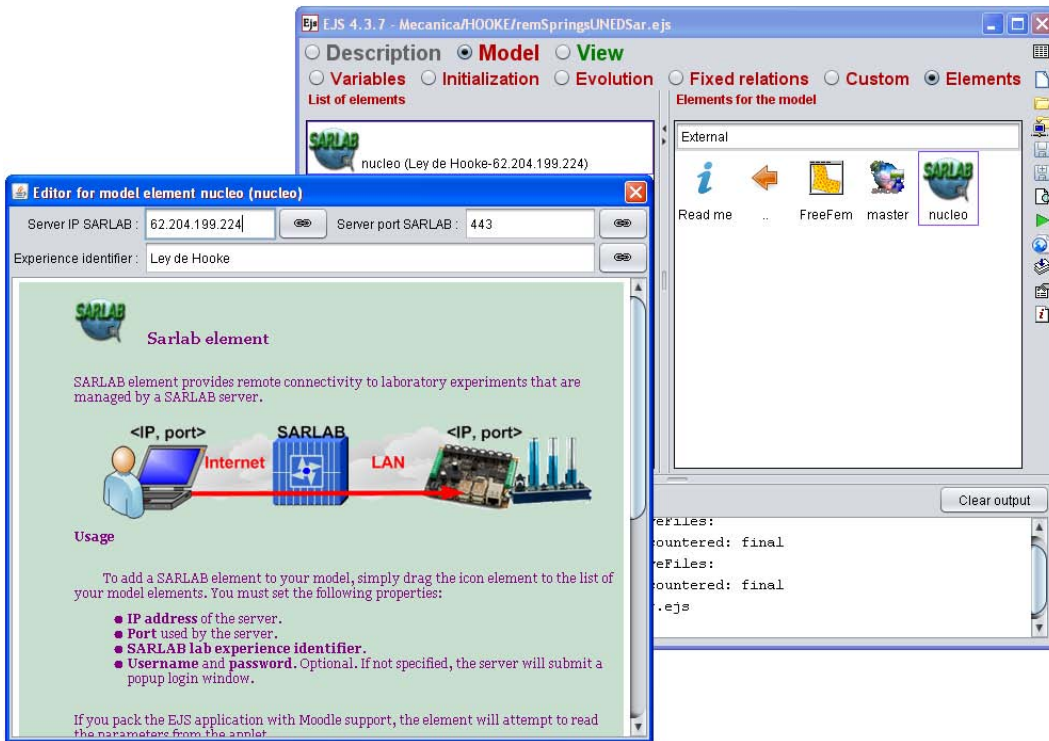


FIGURE 5.5: Configuring the SARLAB element in EJS.

Once the SARLAB element has been added and configured, the only required additional line of code to make things work is: `nucleo.connect()`. This command must be added in the *Initialization* tab of EJS and makes the SARLAB element to establish communication with the SARLAB server, allowing the access of this application to the RL hardware resources.

5.2. Deploying the VRL

Having the VRL prepared, it is time to deploy it and make it accessible to the students. This chapter is only presenting a step-by-step process of how to use the VRL with Moodle in the most basic and fundamental form. Therefore, no

further explanation is given about how to use the additional features such as the saving/loading of files, the collaborative sessions or the booking system.

5.2.1. Installing the EJSApp plugin

The first step requires the installation of the EJSApp plugin, which is the Moodle add-on that allows deploying EJS applications into the LMS. However, the installation of the EJSApp plugin is really simple as it is done the same standard way any other Moodle add-on is installed. These are the instructions:

First, download the EJSApp plugin either from https://moodle.org/plugins/view.php?plugin=mod_ejsapp (for the latest stable version) or from https://github.com/UNEDLabs/moodle-mod_ejsapp

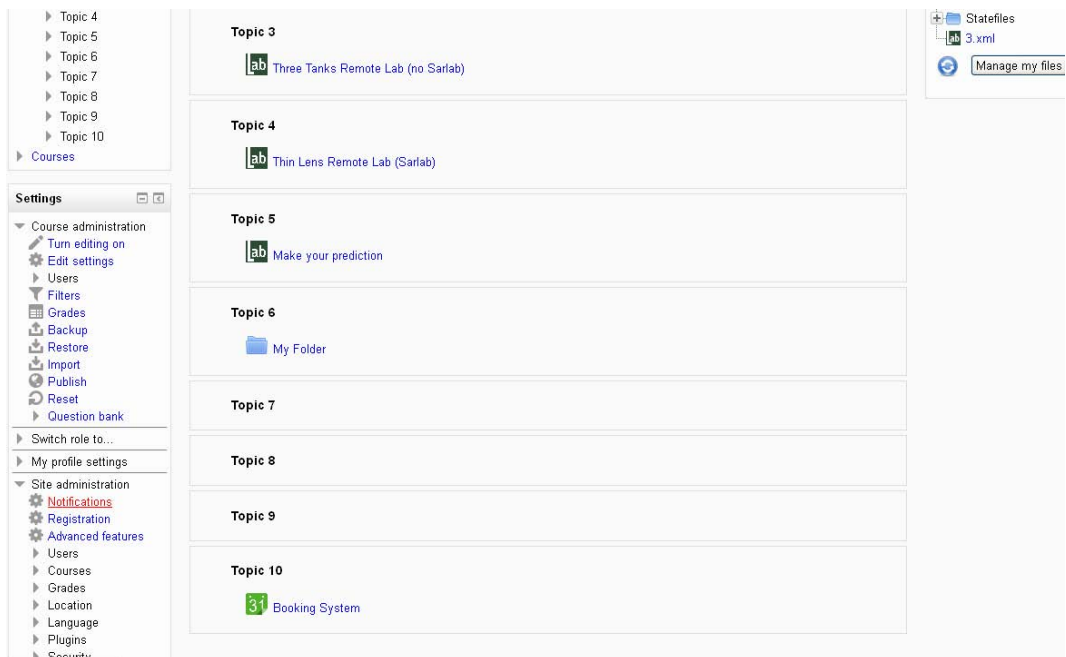


FIGURE 5.6: Notifications link in Moodle.

(for the latest version in development). Then, unzip the content into the `/mod` folder, placed inside the folder in which Moodle is installed.

Then open a web browser window and visit the Moodle website. Login as administrator and click on the *Notifications* link within the *Site administration* menu. The placement of this link may vary depending on the visual theme or the Moodle version being used. Figure 5.6 shows an example of its location when Moodle's default visual theme is being used.

The *Notifications* window in Moodle will show a list of plugins which are either outdated or completely new. In order to install or update an already installed plugin, users just need to follow the instructions shown in their screens. With just a couple of clicks, the automatic process is completed. Figure 5.7 shows an example.

Plugins check You are logged in as Admin User (Logout)

This page displays plugins that may require your attention during the upgrade. Highlighted items include new plugins that are about to be installed, updated plugins that are about to be upgraded and any missing plugins. Contributed plugins are also highlighted. It is recommended that you check whether there are more recent versions of contributed plugins available and update their source code before continuing with this Moodle upgrade.

[Check for available updates](#)

Number of plugins requiring your attention: 4

[Display the full list of installed plugins](#)

Plugin name	Directory	Source	Current version	New version	Requires	Status
Activity modules						
EJSApp	/mod/ejsapp	Contributed	2013031800	2013031801	Moodle 2010.112.400	To be upgraded
EJSAppBooking	/mod/ejsappbooking	Contributed	2013030100	2013030100	Moodle 2010.112.400 mod_ejsapp (2012.12.1800)	Installed
Blocks						
EJSApp collab session	/blocks/ejsapp_collab_session	Contributed		2013030100	Moodle 2010.112.400 mod_upgrade (2012.12.1800)	To be installed
EJSApp "private files" browser	/blocks/ejsapp_file_browser	Contributed	2013030100	2013030100	Moodle 2010.112.400 mod_upgrade (2012.12.1800)	Installed

[Reload](#)

[Upgrade Moodle database now](#)

FIGURE 5.7: Notifications window in Moodle.

5.2.2. Sharing the VRL

The last step is to use the EJSApp plugin to share the EJS VRL, adding it to a Moodle course. However, this process is explained in detail in section [4.2.1.1](#).

Chapter 6

Conclusions and Future Directions

This last chapter is divided into two sections. The first one describes the main conclusions of this thesis while the second one gives some clues about the future directions this work will follow.

6.1. Conclusions

UNEDLabs is an ongoing work which was started after the development and implementation of the AutomatL@bs project. UNEDLabs inherits the well-proven structure of its successful ancestor but changes everything else: the LMS, the way VRLs are published, the features of these VRLs, etc. The reuse of the laboratory resources of AutomatL@bs is an easy task thanks to the fact that UNEDLabs uses the same architecture, tools and technologies.

UNEDLabs is a network of VRLs created following the framework established in this thesis. Nowadays, it counts with seven completely operative VRLs (six from UNED and one from the University of Almería) and three different active courses that use these VRLs.

The simulated experiments not only serve students as a first contact with the phenomena under study but also as a way to obtain theoretical data which can be later compared with the real one, acquired from the remote experimentation. By means of these simulations, students can discover Snell's and Hooke's laws, determine the focal length of a thin lens by two different methods, confirm the law of reflection, observe the total reflection phenomena, study the potential energy variation in a spring, and observe the differences between linear and non-linear springs. Continuing with the same examples, the remote experiences let students empirically check some of these laws and phenomena with a real experiment and compare these results with the theoretical ones. Although these are quite simple experiments, by their use students can not only learn about the previous topics but also about measurement of uncertainties, uncertainties calculus, and linear regressions. Finally, besides these new VRLs and the ones inherited from AutomatL@bs, more complex and challenging VRLs are planned.

Four Moodle plugins were developed to: 1) enhance the use and experience of the VRLs and 2) allow teachers to easily deploy their own VRLs into self-made Moodle courses and to easily manage the VRLs and the created virtual courses.

The *EJSApp* was developed to support the oneclick deployment of VRLs into Moodle. This plugin has the following features: 1) deployment of labs written in EJS, 2) control user access to the deployed labs and distinguish between VRLs (or simulations) and RLs, 3) allow several resizing options for displaying the embedded applets, 4) backup and restore (*EJSApp* provides maintenance facilities for labs, packaging them into Moodle course backups), and 5) integration with SARLAB.

The *EJSApp Booking System* plugin enriches the *EJSApp* module by providing an automatic booking system to schedule and control the access to the physical resources used by the RLs and which can only be used by one person at the same

time. Also, it is able to send Moodle instant messages and e-mails in order to alert about new bookings, assignment of booking rights and so on.

The data collected during the simulated and real (remote) experimentation can be saved thanks to the *EJSApp Files Browser* plugin and so, students are able to compare and contrast the theoretical results with the real ones, which constitutes one of the fundamental issues of the scientific method. These data can be saved basically in two different ways: as plain text (in order to let the students use that data however she needs it) and as xml files (which can be read/loaded by the EJS applications). Finally, since the remote control of elements for all experiments is done with high enough precision, the laboratories presented in this thesis allow students to get good measures to work with.

The *EJSApp Collab Sessions* plugin provides synchronous collaborative support to any VRL developed with EJS, i.e., thanks to this extension, any existing lab written in EJS can be automatically converted into a collaborative lab with no cost. The approach presented in this work not only supports the teacher's presentation or explanation of course material by emulating a traditional classroom on the Internet. More interestingly, it also supports collaborative learning activities centered on students' exploration or application of the course material through VRLs. That is, students working in groups of two or more, mutually searching for understanding, solutions, or meanings.

6.2. Future Work

UNEDLabs has been working as a pilot experience during two courses now, with a repertory of experiments from different fields: control engineering, mechanics and optics. The long-term plans are to maintain this portal, and to develop and add

new web-based virtual and remote experiments for higher courses and/or belonging to new universities into this network.

As already said, other experiments are in development, such as a radiation experiment, a rigid pendulum, an experiment about diffraction, etc. All of these experiments (with both remote and virtual experiences) will be integrated into the network of laboratories that constitute UNEDLabs. This is possible because they all use the same structure and the same software tools, although the hardware used for each experimental systems may be different.

The radiation experiment uses a Geiger counter which can be connected to a PC. The LabView program that enables its control (setting a measuring time, changing the input voltage, etc.) and reading was partially shown in figure A.3. The radioactive samples need to be inside the tube assembly of figure 6.1, which is connected to the PC through the Geiger counter. A LEGO tray exchanger is planned to alternate between different radioactive samples inside the tube assembly.

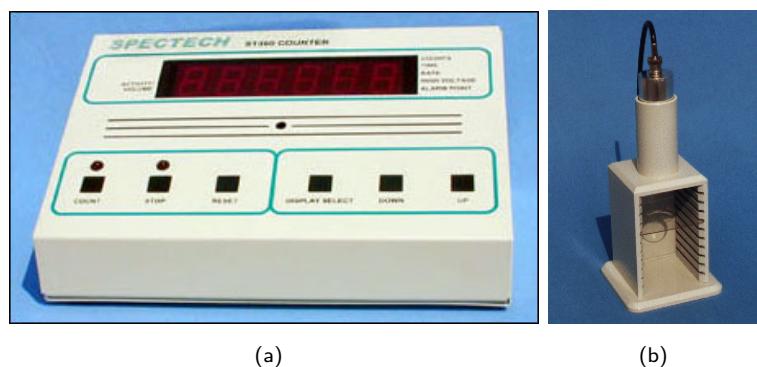


FIGURE 6.1: (a) Geiger counter and (b) tube detector assembly for the radiation experiment.

A plotter or XYZ table for a distribution of potential experiment was built with LEGO pieces. It uses three NXT motors, one contact sensor, two pneumatic pumps, one NXT intelligent brick, one potential sensor, a power supply, and many LEGO pieces. Two of the NXT motors control the 'x' and 'y' coordinates of the

potential sensor respectively while the third one is in charge of the pneumatic pumps, which get up and down the sensor, putting it in contact (or not) with the resistive sheet of paper and so, taking measures (or not). Figure 6.2 shows the experimental system corresponding to this electrostatic experiment. Although the apparatus is finished, the virtual experiment and some details of the remote control are still being polished up.

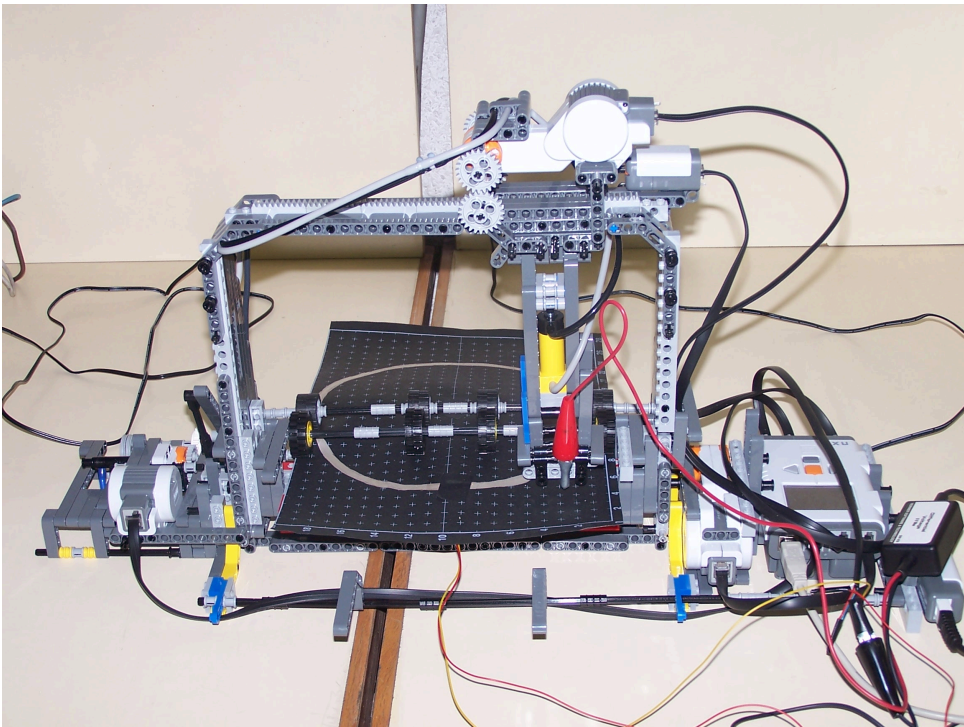


FIGURE 6.2: Experimental system of the electrostatic experiment.

Working in collaboration with Prof. Juan Pedro Sánchez, there are two more experimental systems in preparation. The first one is for a new RL which will allow students to watch and study the diffraction phenomenon. The second one will be used as a RL to experiment with the photoelectric effect.

Collaborating with Prof. Jaime Arturo de la Torre, there are also two more experimental systems being prepared. One with vibrating chords for a RL about

waves and harmonics and another one with mobile carts for a RL for studying the energy and momentum conservation in elastic and inelastic collisions.

Leaving the creation of new VRLs to be added to the UNEDLabs network aside, there are also plans for future software and structure development that will enhance the system usability and robustness.

The ComPADRE Digital Library is a network of free online resource collections supporting faculty, students, and teachers in Physics and Astronomy Education. The Open Source Physics (OSP) collection¹ within ComPADRE provides curriculum resources that engage students in physics, computation, and computer modeling. There, hundreds of EJS simulations created by the community can be easily found and downloaded for free. The National Taiwan Normal University (NTNU) JAVA Virtual Physics Laboratory² is an online forum where physics teachers share physics simulations resources. A big section is dedicated to EJS and, as in the OSP collection, hundreds of EJS programs and simulations can be downloaded for free. Moodle allows the creation and use of the so called repository plugins. These plugins are used to pick digital resources, available in external repositories, directly from Moodle. Examples of these are Youtube, Dropbox, etc. Future plans are to create a new repository plugin for EJS applications which could connect to the OSP collection in ComPADRE and to the NTNU JAVA forum in a way that Moodle users could search and pick EJS resources, stored in any of the two previous repositories, directly from Moodle, using the EJSApp add instance form (Figure 4.2).

EJSApp Collab Session (4.2.2.2) allows creating synchronous collaborative sessions among different users to share a lab and work with it at the same time. However, integration into this solution of verbal communication between users is

¹<http://www.compadre.org/osp/>

²<http://www.phy.ntnu.edu.tw/ntnujava/>

not provided yet. There is a Moodle plugin for skype that would suppose a great addition to this collaborative work. Future plans are to extend the EJSApp Collab Session to automatically send skype calls (whenever this Moodle plugin is installed) to all the Moodle users invited to a collaborative session.

Finally, a great improvement of the EJSApp Booking System plugin is in preparation. The idea is to make this Moodle plugin to periodically check whether the *Lab PCs* are on or not (establishing communication with SARLAB, if this system is in used). If the plugin detects that a *Lab PC* has passed from on to off since the last check, it would send an e-mail alerting those teachers in charge of the RL controlled by that particular *Lab PC*.

All these future lines of work in the software will enhance even more the usability and robustness of the presented tools and framework.

Appendix A

Tools for Developing Web Experimentation Portals

This appendix looks over all the software and hardware tools that were used for creating both the general framework proposed in this work and the VRLs that could be developed within this framework or that were already developed (see Sections 3.1, 3.2 and 3.3 in Chapter 3 and Appendix B).

A.1. Software Tools

A.1.1. Easy Java Simulations (EJS)

Easy Java Simulations (EJS)¹ is an authoring tool written in Java that helps to create interactive simulations in Java, mainly for teaching and learning purposes in physics, [58, 59, 99]. This tool was designed to work at a high conceptual level, letting programmers concentrate most of their time and efforts on the scientific aspects of the simulation. The program hides to the user all the other necessary

¹Easy Java Simulations can be downloaded for free at <http://www.um.es/fem/Ejs>

but easily automated tasks (numerical computation, graphical drawing, etc.) while the computer automatically performs them.

EJS is inspired by the model-view-controller (MVC) paradigm, which states that simulations are formed by three different parts:

- The *model*, which describes the studied phenomena in terms of variables and relations between them.
- The *control*, which defines the actions users can make over the simulation.
- The *view*, which offers a representation of the different states of the simulated system.

EJS makes things even simpler eliminating the *control* element of the MVC paradigm and embedding one part in the *view* and the other one in the *model*. This way, there are just two main parts in EJS: the model and the view. While the first one lets developers introduce the laws, equations, and programs that rule the simulated experiment, the second one offers the GUI to interact with the simulation, that is, the model.

Thus, applications in EJS are created in two steps: 1) the building of the *model* to simulate by using the built-in simulation mechanism of EJS, and 2) the construction of the *view* to show the model state and its reactions to the changes made by users.

The *view* can be seen as an interface that links the model and the user. It is built in order to offer a visual representation of the main properties of both the model itself and its dynamic behavior as well as for adding interactivity in the simulation. EJS includes a complete library of visual elements which can be used to build very complex *views*.

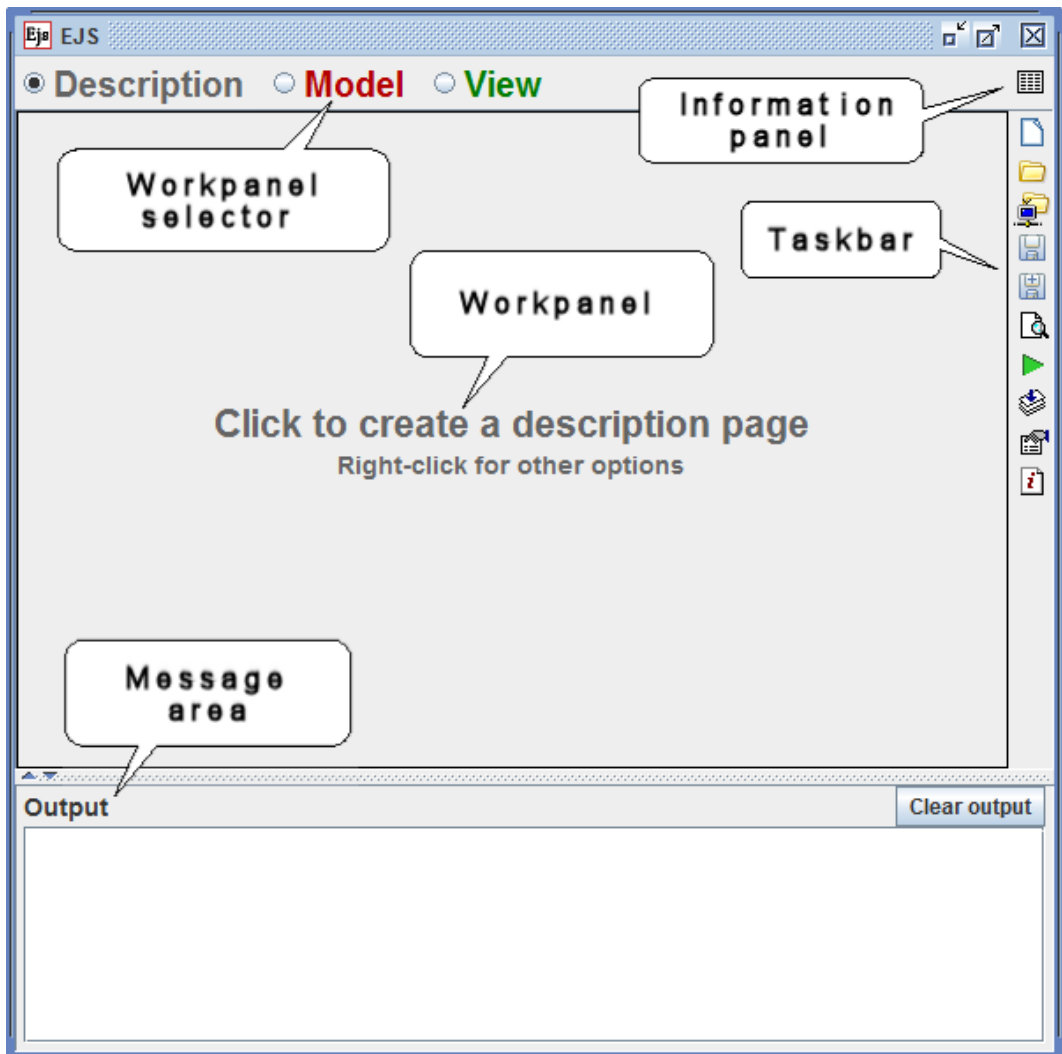


FIGURE A.1: The EJS user interface.

Elements' properties of the *view* can be linked to the variables of the *model*, which establishes a bidirectional information flow between both parts. Any change in a variable of the *model* is then immediately updated by the *view*. Also, any change caused by the user in the *view*, automatically modifies the value of the associated variable of the *model*.

To sum up, while the *model* is the scientific part of the simulation, the *view* (which offers the GUI for the user) requires greater programming skills.

Once the development work is completed, EJS produces the Java code, it compiles the program into an applet and it generates several html files as the final result. In this applet, the *model* part is transparent to the user, who only needs the *view* to perform the activities related with the simulation. However, as EJS is a Java code generator, developers have the possibility to write their own Java code and add it by means of external libraries (*.jar files). This particular characteristic allows using EJS to develop the remote laboratories since the *view* can be connected somehow to a remote server (placed at a laboratory, near the experimental system) through the Internet. This is done thanks to the works presented in [61, 100], in which the JIL and the JIM applications are presented.

During the last few years, EJS has grown for helping to create web-accessible laboratories in control engineering education. With this objective in mind, recent releases of EJS support connections with external applications, such as LabView and Matlab/Simulink. Hence, EJS not only is useful to create virtual labs, but also the GUIs of their remote counterparts [77, 101]. Also, in [102], EJS presents a new feature to communicate via the Internet with Arduino Ethernet boards, expanding even further its possibilities of controlling hardware in a remote way.

A.1.2. LabView

LabView was developed by National Instruments in 1976. Programs made with LabView are called virtual instruments (VIs). This name is related to their main function, which is control instrumentation.

In order to make a laboratory accessible from the Internet, instruments that can be remotely controlled are needed. Data acquisition devices (DAQ) are commonly

used to remotely control any type of external hardware. A computer connected to the device that needs to be controlled and equipped with a DAQ allows to control all the instrument's functions. There are some programs and software developed specifically for controlling the most common devices. LabView has become one of the most important of these tools.

LabView is a graphical programming language (GPL), which make it easy to use. Thanks to it, people with low programming skills can create programs, which would be impossible to achieve for them with other traditional programming languages.

There are two big differences between LabView and most other GPLs. First, GPL code developed with LabVIEW executes following the rules of data flow instead of the more traditional procedural approaches (a sequential series of commands to be carried out) found in most text-based programming languages (such as C, for example). GPLs are dataflow languages that promote data as the main concept behind any program. The flow of data between nodes in the program (and not sequential lines of text) is what determines the execution order. The second main difference is that LabView contains the same programming concepts found in most traditional languages. For example, it includes all the standard constructs, such as data types, loops, event handling, variables, recursion, and object-oriented programming.

Like most people, scientists and engineers think best using pictures. In addition, since dataflow languages base the structure of the program around the flow of data and these two last type of professionals are used to visualize and/or model diagrammatically processes in terms of blocks and flowcharts (which also follow the rules of data flow), scientist and engineers typically understand even quicker GPL code since. Most general-purpose programming languages require users to

²Image obtained from <http://www.ni.com/labview/whatis/graphical-programming>

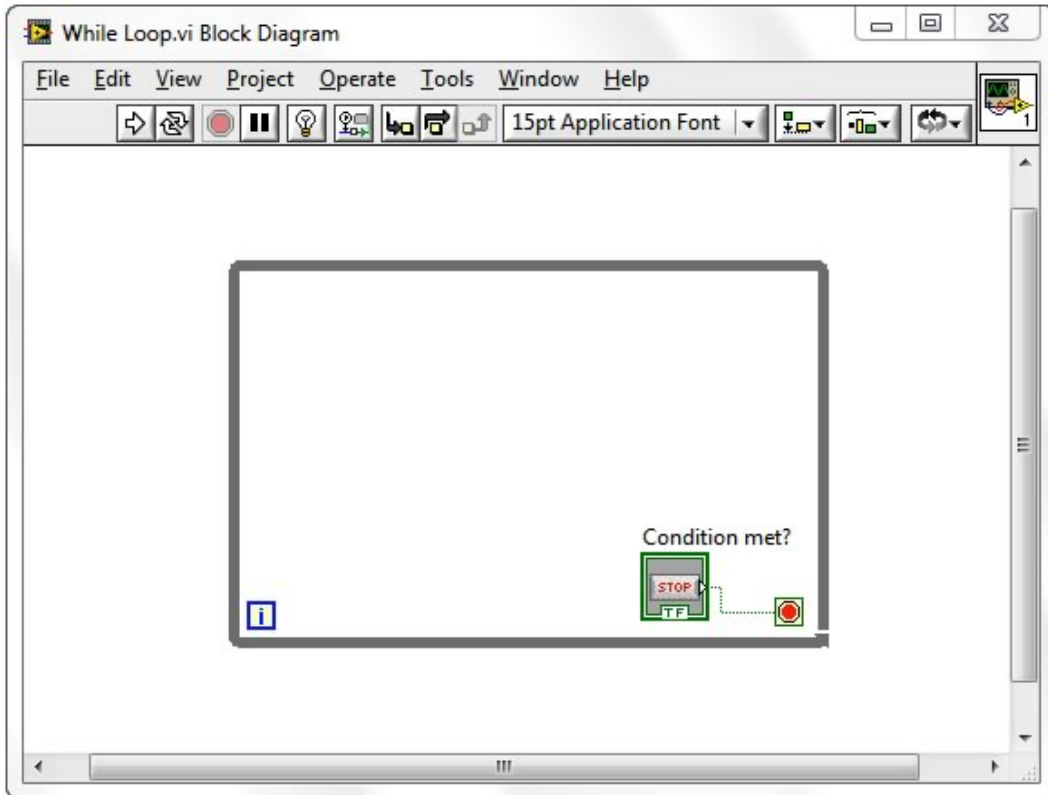


FIGURE A.2: A While Loop in this GPL is intuitively represented by a graphical loop, which executes until a stop condition is met².

spend a lot of time in learning the syntax associated with that language and then to imaging how to use and adapt the structure of the language to apply it to the problem they want to solve. GPLs, such as LabView, however, provide a more intuitive experience. Finally, LabView offers a great number of libraries that allow to easily handle:

- Different communication interfaces: Serial port, parallel port, GPIB, PXI, VXI, TCP/IP, UDP, DataSocket, Irda, Bluetooth, USB, and OPC.
- Capacity to interact with external applications: DLL, ActiveX, data bases or even invoke Matlab scripts.

- Tools for digital processing of images.
- Visualization and handling of graphics with dynamic data.
- Images acquisition and processing (NI Vision).
- Movement control.
- Real time features (NI Real Time).
- FPGAs programming.
- Synchronization.

Figure A.3 shows part of a program in LabView that controls a Geiger counter (used for an experiment about radiation which is still unfinished but will be published in UNEDLabs in the future).

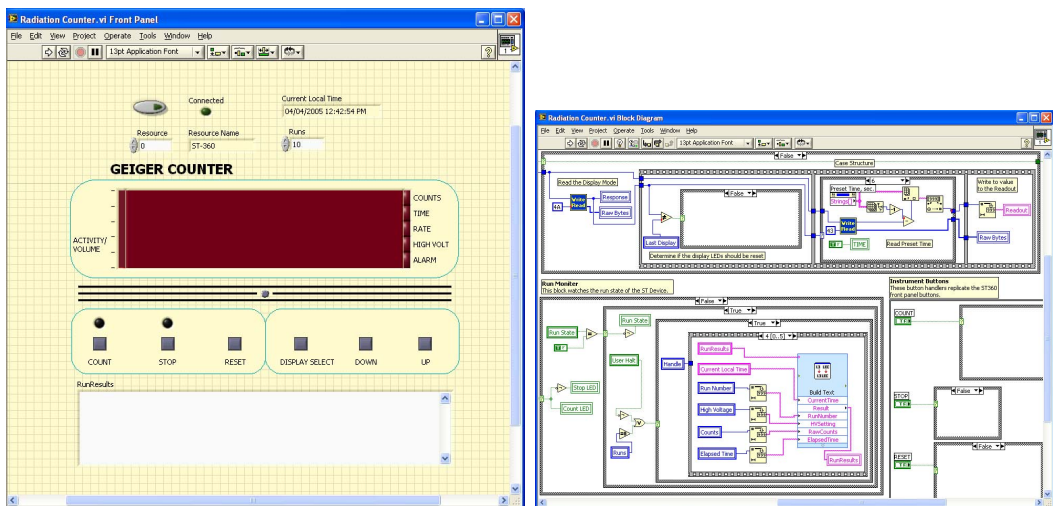


FIGURE A.3: Front panel and blocks diagram in a LabView program.

The same figure shows that a program made in LabView has two different parts: a front panel and a blocks diagram. The frontal panel is the interface with

the user, and it is here where the controls and indicators must be defined. The blocks diagram is the program itself (where its functionality is defined), and it is in this part where the different icons with their associated functions are placed and connected to each others.

In relation to this work, LabView allows carrying out the main actions of the communication between the client interface (TCP/IP) and the experimental system hardware, as well as the verification in the data bases for filtering the access of users to the laboratory resources.

LabView has been extensively used in RLs for controlling its associated hardware part. A few examples are [46, 103–107]. This tool was the solution applied to the three RLs presented in Chapter 3. While the light in isotropic media and the focal length of thin lens experiments use the hardware described in Section A.2.1, the third one (Hooke's law) uses the hardware presented in Section A.2.3. The framework presented in this work allows easily reusing any RL which hardware is controlled by LabView since the only needed change would be to build a new UI for the RL application using EJS.

A.1.3. Matlab

MATLAB (MATrix LABoratory) is a high-level language for numerical computation, programming, and visualization. Matlab allows to analyze data, develop algorithms, and create models and applications but it can also be used for many other applications, including control systems, test and measurement, signal processing and communications, image and video processing and computational biology.

Although Matlab is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain

simulation and Model-Based Design for dynamic and embedded systems.

In terms of controlling hardware, Matlab supports serial devices including RS-232 using the Instrument Control Toolbox. This toolbox supports communication with instruments and devices directly from Matlab through commonly used communication protocols, instrument standards, and drivers. The toolbox provides a consistent interface to all devices independent of interface, hardware manufacturer, or instrument type. Also, Matlab's Data Acquisition Toolbox supports PC-compatible data acquisition hardware from multiple vendors, including National Instruments, Advantech, and Measurement Computing. The HDL Verifier provides FPGA-based hardware-in-the-loop capabilities. This allows execution of an algorithm to be done at hardware speeds, allowing a much more exhaustive set of stimulus and test data to be applied. Finally, interfacing off-the-shelf DSP/analog hardware to Matlab is also possible using DirectDSP software.

Although it is not so popular for controlling hardware as LabView is, Matlab has also been used in some works as a solution for RLs. Examples are [[100](#), [108](#), [109](#)].

A.1.4. TCP/IP Sockets and JIL and JIM libraries

TCP/IP protocol represents the core of the Internet and serves for connecting and allowing the communication between computers running with different operating systems (OS), including PCs and central (server) computers over both, local area networks (LAN) and wide area networks (WAN) as the Internet. Thanks to this, it is possible to connect any instrument and device to the Internet through a TCP/IP interface, allowing the control of that instrument or device from any part of the world (as long as an Internet connection is available). Therefore, TCP/IP protocol is used for connecting the client application with the server.

The incorporation of this programming module to the EJS program is done thanks to the works developed in [61, 100], which offer Java libraries (jil.jar and jim.jar) that define the necessary methods for data sending and reception by means of objects that belong to the class Socket of Java. These methods are in charge of connecting the variables defined in EJS (section A.1.1) with the ones defined in LabView (section A.1.2) or Matlab (section A.1.3).

A.1.5. Database Server (MySQL)

A database is a set or collection of information organized in such a manner that a software program can select and access desired pieces of data. Usually, the data are organized to model important aspects of reality in a way that supports processes requiring this information. Regarding their structure, databases are organized by fields, records, and files. A field is a single piece of information; a record is one complete set of fields; and a file is a collection of records.

The term database is correctly applied to the data and their supporting data structures, and not to the database management system (DBMS). The database data collection with DBMS is called a database system.

Examples of DBMSs are Oracle, MySQL, PostgreSQL, SQLite or Microsoft SQL Server.

The framework presented in this work, based in the use of Moodle, is compatible with any of the previous DBMSs. UNEDLabs, for example, uses MySQL. In relation to UNEDLabs, MySQL is in charge, among many other things, of managing and storing the users' information. The booking system (section 4.2.1.2) is tightly related to the database. The database is also used for storing data about the VRLs and the courses they belong to (see section 4.2.1.1, keeping a relation between the

files saved from the VRLs and the users that own those files (section 4.2.2.1) and recording the active collaborative sessions (sections 4.2.2.2).

MySQL runs as a server providing multi-user access to a number of databases. Some of its main characteristics are:

- A broad subset of ANSI SQL 99, as well as extensions.
- Cross-platform support.
- Different stored procedures.
- Secure socket layer (SSL) support, which grants secure conexions.
- Full-text indexing and searching using MyISAM³ engine.

A.1.6. Web Server (Apache)

A Web server is a software (computer program) that acts like a virtual machine or a computer that hosts the web pages over the Internet. The main function of a web server is to deliver the html documents (which may involve text, images, videos, audio etc) to the clients.

Examples of Web servers are Apache server, Microsoft IIS server, IBM Lotus, Lighttpd. The common features of all these web servers (and any other) are virtual hosting, bandwidth throttling, large file support, server side scripting.

The framework presented in this work, based in the use of Moodle, is compatible with any of the previous Web servers. UNEDLabs, for example, uses an Apache server. In relation to UNEDLabs, the Apache server is in charge of serving the Moodle webpages (some of them embed the VRLs applications) to the clients.

³MyISAM is the default storage engine for the MySQL relational database management system versions prior to 5.5. It is based on the older ISAM code but has many useful extensions. Each MyISAM table is stored on disk in three files. The files have names that begin with the table name and have an extension to indicate the file type.

An Apache server has a modular structure and hence it can be customized very easily. It is also an open software, so it is possible to add new modules to the server when required and make modifications that suit specific needs. It is more stable than the other web servers and is easier to solve administration related issues and make changes to settings or some features. Moreover, it can be installed successfully in multiple platforms.

A.1.7. PHP

PHP (Hypertext Preprocessor) was designed for Web development to produce dynamic Web pages. It is an open source general-purpose server-side scripting language and it is one of the first of its kind to be embedded into an HTML source document rather than calling an external file to process data. The code is interpreted by a Web server with a PHP processor module which generates the resulting Web page. PHP is an HTML-embedded scripting language. Much of its syntax is borrowed from C, Java and Perl with a couple of unique PHP-specific features thrown in. The goal of the language is to allow web developers to write dynamically generated pages quickly.

PHP has evolved to include a command-line interface capability and can be used in standalone client-side GUI applications. It can be deployed on most Web servers, on almost every operating systems and platforms and can be used with many relational database management systems (RDBMS), free of charge. PHP is installed on more than 20 million Web sites and 1 million Web servers. Software that uses PHP includes Moodle, Drupal, Joomla, MediaWiki, and WordPress.

A.1.8. Moodle

A Learning Management System is a software application for the administration, documentation, tracking, and reporting of training programs, classroom and online events, e-learning programs, and training contents.

LMSs range from systems for managing training and educational records, to software for distributing courses over the Internet with features for online collaboration. There is a wide range of dimensions where LMSs can be applied to: Student self-service (e.g., self-registration on instructor-led training), training workflow (user notification, manager approval, wait-list management), provision of on-line learning (Computer-Based Training, read & understand), on-line assessment, management of continuous professional education, collaborative learning (application sharing, discussion threads), or training resource management (instructors, facilities, equipment).

Some LMSs are Web-based to facilitate access to learning contents and administration services. These ones are commonly used by educational institutions to enhance and support classroom teaching and to offer courses to a larger population of learners across the globe.

AutomatL@bs and FisL@bs were based on a simple LMS called e-Mersion [60], a web-based environment of management, administration and collaborative work developed at the École Polytechnique Fédérale de Lausanne. UNEDLabs is based on a free, open source, and up-to-date LMS called Moodle [110].

Thanks to the use of Moodle, teachers have the possibility to manage the work modules associated to an specific group of students (those with a particular common subject, for example). Its use makes also possible to continuously plan the activities and to offer learning scenarios where the individual work of the students can be combined with collaborative work among them.

Moodle has several features considered typical of a LMS, plus some original innovations (like its filtering system). Moodle can be used in many types of environments such as in education, training and development, and business settings. Some typical features of Moodle are:

- Centralize and automate administration. Teachers and/or other administrators can follow and control the online modules and the students progression from only one application.
- Use self-service and self-guided services and assemble and deliver learning contents rapidly. Students can access the online resources from their own homes and use them whenever and however they wish to do it.
- Consolidate training initiatives on a scalable web-based platform.
- Support portability and standards. Any user can use the LMS and access its contents, no matter the computer, OS, or browser he/she uses.
- Personalize content and enable knowledge reuse.
- Assignment submission. Students can use Moodle to deliver their laboratory reports, for example.
- Discussion forums. Asynchronous collaboration is achieved thanks to the use of open forums.
- Files download and files repositories.
- Grading. Teacher can use Moodle to grade the activities created; for example the delivering of a laboratory report.
- Instant messages. Synchronous collaboration is achieved thanks to the use of instant messages.

- Online calendar.
- Online news and announcements.
- Online quizzes.
- Wikis.

Developers can extend Moodle's modular construction by creating plugins for specific new functionality. Moodle's infrastructure supports many types of plug-ins:

- Activities.
- Resource types.
- Blocks.
- Question types: multiple choice, true and false, fill in the blank, etc.
- Visual/graphical themes.
- Authentication and/or enrollment methods.
- Content filters.

UNEDLabs adds VRLs to Moodle. This way, the web portal tries to assemble the benefits of face-to-face laboratory sessions, specially for technical and scientific studies, in which the presence of experimentation is so important. In this context, some important things must always be added to Moodle by the developers of the online resources and courses:

- Theory documentation. It should include the theory background needed by the student to successfully perform the experiments. Objectives of the experimentation should also be presented as well as a brief description of

the devices and hardware to be used. A list and description of the tasks students must carry out is also given here. Finally, it is important to offer a description of the client interface (the Java applet) in order to make clear how to use it. As seen in figure 2.9, UNEDLabs provides all this information in several .pdf files which can be accessed from Moodle's web environment.

- Client interface. As already seen, UNEDLabs uses a Java applet (created with EJS for this purpose) which serves for both the virtual and the remote experimentation. So, by means of this applet users can carry out experimentation activities that are: 1) simulated, based on a mathematical model of the phenomena or 2) real, performed on the experimental systems located at the universities' laboratories.

The framework presented in this work adds experimentation capabilities to Moodle thanks to the use of four plug-ins specifically designed with this purpose. These plugins are:

- EJSApp. This is an activity plug-in. Used for adding the VRLs into a Moodle course.
- EJSApp Booking System. This is a resource type plug-in. Used for managing the access of users to the RLs.
- EJSApp File Browser. This is a block plug-in. Used for storing in Moodle the files saved from the VRLs.
- EJSApp Collab Sessions. This is also a block plug-in. Used for creating synchronous collaborative experimental sessions with the VRLs.

More information about these plugins can be found in Chapter 4.

A.2. Hardware Tools

At first glance, this could be the ideal place in the document to present a description of the hardware tools used to build the experimental systems of the different experiments. However, since each of these experiments have different characteristics, the associated phenomena are not the same, and the variables or parameters that need to be measured and controlled change, the hardware tools involved in their experimental assemblies may vary a lot (and it does in fact) from one case to another.

Therefore, the materials and components used for the three presented experiments (light in isotropic media, non-linear springs and Hooke's law, and focal length of thin lenses) are detailed in Sections 3.1, 3.2 and 3.3, respectively.

Still, this section enumerates and describes the categories and type of materials supported by the framework for assembling the experimental systems, giving and insight of the wide variety of products developers could choose for creating their own RLs if they were working with this framework. Also, the main characteristics (and both advantages and disadvantages) for each hardware solution are explained.

A.2.1. National Instruments Motion

National Instruments (NI) motion control is a line of software (as LabView toolkits, for example) and hardware (controllers, drives and motors) products for measurement and control platforms by National Instruments company.

NI has two separate architectures for deterministic execution of these components: (1) plug-in motion controllers and (2) LabView real-time targets as motion controllers through NI LabView.

Focusing on the plugin motion controllers (which is the solution applied for two of the three RLs presented in Chapter 3), NI offers PCI and PXI boards. These

controllers contain onboard DSP and FPGA chips that allow them handle most of the tasks in a motion control system. The Universal Motion Interface (UMI) can be used to connect to 3rd party stepper and servo drives. Plug-in motion boards are great for adding motion control to an existing desktop or PXI system. Even though it is an expensive solution for motion control when compared to other third party products, it is the most affordable one between those offered by NI. On the other hand, this is a very robust and ready-to-work solution.

As described in section 2.1, EJS can be used to communicate with LabView via the Internet by using the JIL applications. Since this hardware is controlled using LabView, the presented framework is compatible with the use of these components.

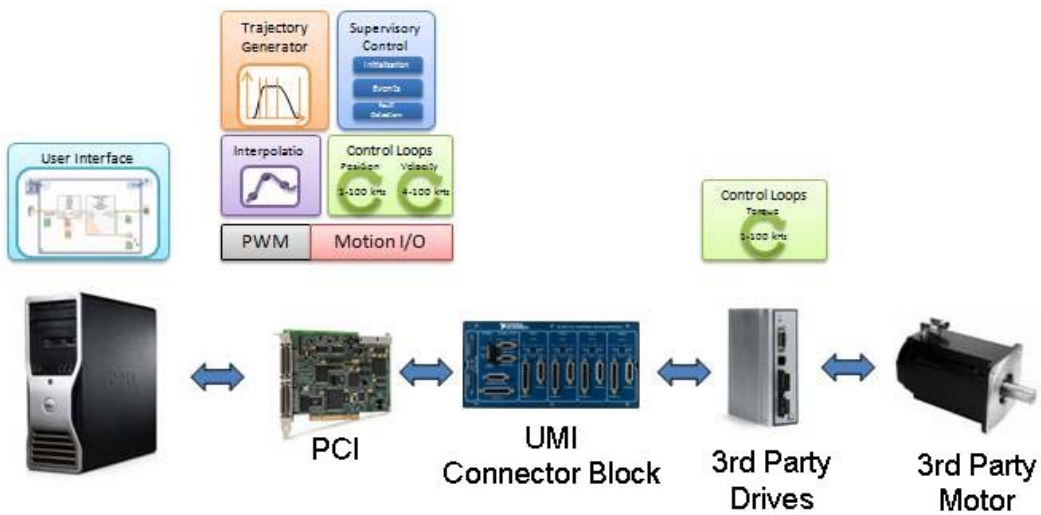


FIGURE A.4: NI PCI plug-in motion control solution.

Figure A.4 depicts this solution by NI. Although third party drives and motors can be used (which makes this solution adjustable to personal needs), NI also offers a wide variety of these two components.

A.2.1.1. Controllers

Applications using NI plug-in motion controllers are programmed with the NI-Motion driver and the NI Motion Assistant.

A.2.1.2. UMI Connector Blocks

The NI UMI are special motion control interfaces for connecting NI motion controllers to third-party drives or amplifiers. These interfaces offer a variety of features ideal for industrial environments such as D Sub connectivity and signal isolation.

A.2.1.3. Drives

National Instruments motor power drives provide reliable, easy-to-connect drive solutions for all NI motion controllers in a variety of power ranges and form factors. NI offers two types of drives: one for servo motors and another for stepper motors.

The brushless servo drives feature cutting-edge technology for a single motion axis. They provide plug-and-play configuration with brushless servo motors.

The stepper motor drives provide reliable, easy-to-connect cabling solutions. NI offers one-axis power drives with high-torque output, microstepping, and anti-resonance control.

A.2.1.4. Motors

NI offers servo and stepper motors.

NI brushless servo motors provide superior dynamic performance, offer plug-and-play configuration through smart feedback technology, and are perfectly matched with NI servo drives. NI brushless servo motors incorporate low-inertia rotors and stand out due to their low-cog, low-harmonic distortion magnetic design.

NI stepper motors provide precise, high-torque performance and easy connectivity to NI stepper motor drives. Because of their brushless design, rugged bearings, and innovative cooling technology, these motors offer long-term durability. The two-phase design ensures small and precise movement in 1.8 degree increments (200 steps/revolution) and is simple to control.

A.2.2. Arduino

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. Arduino can receive input from a variety of sensors and it can also affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring³) and the Arduino development environment (based on Processing⁴).

The boards can be built by hand (the hardware reference designs of the Arduino boards are available under an open-source license) or purchased preassembled. The software can be downloaded for free.

This is an extremely cheap solution with another great advantage when it is used along with the framework presented in this work for creating and deploying RLs: Arduino Ethernet boards can be controlled via the Internet directly by the most recent versions of EJS. For older versions of EJS, the JIL applications can still be used for communicating with Arduino since these boards can be controlled from LabView while JIM allows the same when using Matlab/Simulink to control these boards.

³Wiring is an open-source programming framework for microcontrollers (<http://wiring.org.co/>)

⁴Processing is an open source programming language and environment for people who want to create images, animations, and interactions (<http://processing.org/>)

⁵Photo by the Arduino Team, obtained from <http://www.arduino.cc/>

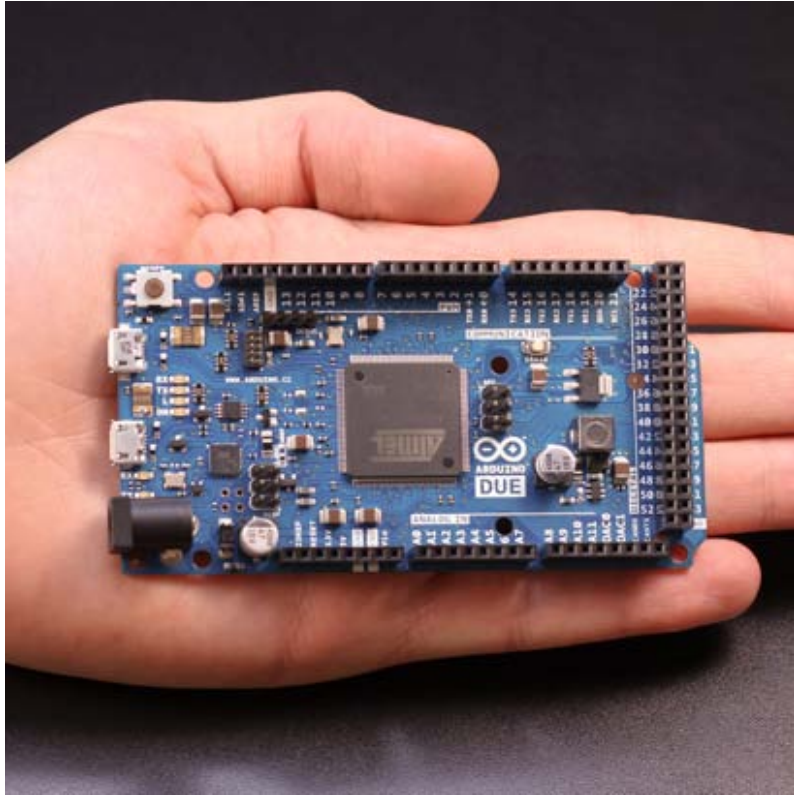


FIGURE A.5: Arduino board⁵.

A.2.3. Lego NXT

Lego Mindstorms NXT 2.0 combines an intelligent microcomputer brick and the versatility of the LEGO building system with all-new technologies. Moreover, the microcomputer brick can be easily programmed with LabView. The 2.0 toolkit features everything you need to create many robotics inventions.

The NXT Intelligent Brick features a 32-bit microprocessor and Flash memory, plus support for Bluetooth and USB. The NXT intelligent Brick includes:

- 32-bit ARM7 microprocessor.
- Support for Bluetooth wireless communication.

- 1 USB 2.0 port.
- 4 input ports.
- 3 output ports.

The output ports of the NXT Intelligent Brick can be used to connect up to three servo motors. The servo motor offered by Lego has a built-in rotation sensor that measures speed and distance, and reports back to the NXT Intelligent Brick. This allows for precise steps and complete motor control within one degree of accuracy. Several motors can be aligned to drive at the same speed.

The input ports allow connecting up to four sensors. These are the sensors commercialized by Lego:

- Touch sensor.
- Light sensor.
- Sound sensor.
- Ultrasonic sensor.
- Compass sensor.
- Accelerometer sensor.
- Color sensor.

The Lego Mindstorms NXT solution was applied for the RL described in Section 3.2. These components have been used in many projects, come of them with learning objectives [52–57]. Again, LabView can be used to control the Lego NXT Intelligent Brick and so, the combination of EJS plus JIL enables the use of Lego Mindstorms components for creating RLs within the proposed framework.



FIGURE A.6: Lego Mindstorms NXT.

A.2.4. Other Components

There are some other components, apart from motors and sensors, that are commonly required for creating experimental systems used as RLs. Some examples are webcams, relays and lasers.

A webcam is always necessary in a RL for providing visual feedback to the students and approach the remote experiment to the remote experimenter.

Relays are useful elements for switching on or off certain hardware components when a user connects or disconnects to the RL, respectively.

Lasers are needed for optics experiments, such as the ones presented in Sections [3.1](#) and [3.3](#) from Chapter [3](#).

Appendix B

Adapted Web-Labs: Heatflow System, Three Coupled Tanks and Servo Motor

While Chapter 3 presented new VRLs created and deployed using the framework proposed in this thesis, this appendix looks over the already existing VRLs (used in AutomatLabs and developed for Dr. Hector Vargas' thesis [111]) that were adapted to be integrated into the Moodle web platform used by UNEDLabs, so they can benefit from the tools presented in Chapter 4.

B.1. Adapting Old VRLs Created With EJS

An important issue when technology and software solutions move forward is to warranty backwards compatibility so that old things can be reused. In the particular case of this work (which proposes a whole new framework for integrating VRLs into web learning environments), this means offering the possibility to reuse already

existing VRLs applications and allowing them to be embedded into the LMS so they can take advantage of the new features offered by this integration.

The three VRLs presented in this appendix were created as one only EJS application that serve for both purposes: the virtual and the remote experimentation. However, a better way to present them is to deploy VRLs as two different applications: one for the VL and another one for the RL (such as it is done with the VRLs presented in Chapter 3). Therefore, the first step for adapting these three already existing VRLs was to separate each of them into two different EJS applications. Using VLs and RLS applications independently (such as UNEDLabs does) has several advantages when compared to Automatlabs (in which both parts were packed into one single application):

1. In the vast majority of cases, students only work either with the virtual or the remote part of an experiment. This is due to the fact that each experience may take a considerable amount of time; up to two hours, depending on the complexity of the VRL. Therefore, when one only application is shared for both the virtual and the remote parts, around half of the downloaded application is never used. Dividing the VRL into two different .jar files, each application's weight is reduced to almost the half of the original one, allowing students to load the virtual or the remote laboratory much quickly.
2. In Automatlabs, each time a student wanted to access a RL, she had to load the VRL application, which was always first initialized in the simulation mode. Once loaded, users had to click on a connect button to switch from the VL to the RL. Using independent applications for RL removes this steps and, therefore, enhances the usability of these tools.
3. When VRLs are divided in two different applications, they can be uploaded to Moodle as two different EJSApp activities. This allows using Moodle

built-in tools for grading them separately, which would be impossible if one only application was used.

4. Since RLs are distributed in their own independent application in UNED-Labs, those EJSApp activities can be defined (and distinguished from others EJSApp activities) as remote experiments. This allows using external tools and processes (such as the EJSApp Booking System) to identify users when trying to access the remote laboratories hardware resources. In Automatlabs, students had to provide a username and a password within the VRL application when connecting to the RL. In UNEDLabs, this necessity no longer exists.

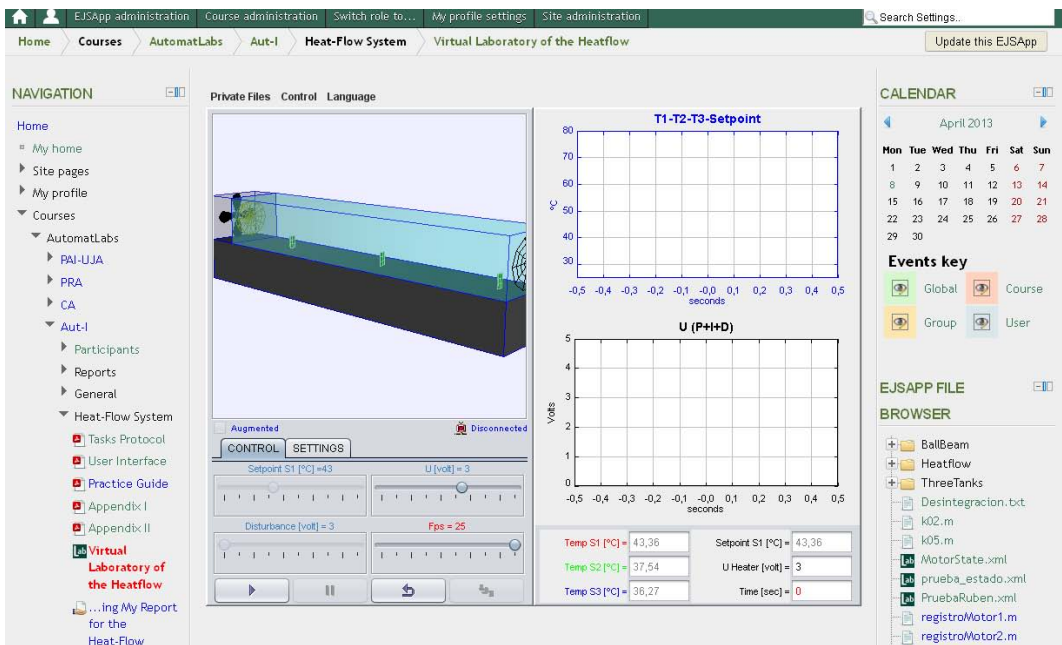


FIGURE B.1: The heatflow virtual laboratory integrated into a Moodle course.

B.2. Virtual Laboratories

The original three EJS programs (one per each VRL from AutomatLabs) were slightly modified in order to let them take advantage from all the features presented in this thesis. These modifications were:

1. Add the automatic language translation support. This is achieved by using the EJS native method `_setLocale()` and makes the EJS applet to use string labels in the language configured by the user in Moodle.
2. Use the appropriate EJS methods for saving/reading data. This is achieved by using the methods presented in section 4.3.2 and allows the EJS applet to save files to the EJS File Browser Moodle block and, for xml state files, also to read them and load their data.

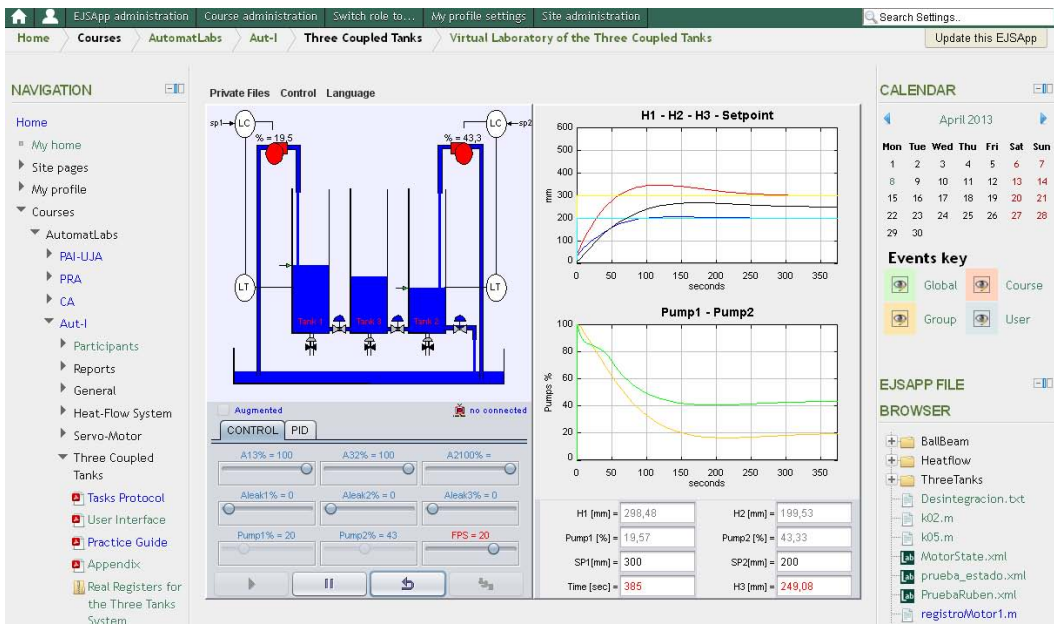


FIGURE B.2: The three tanks virtual laboratory integrated into a Moodle course.

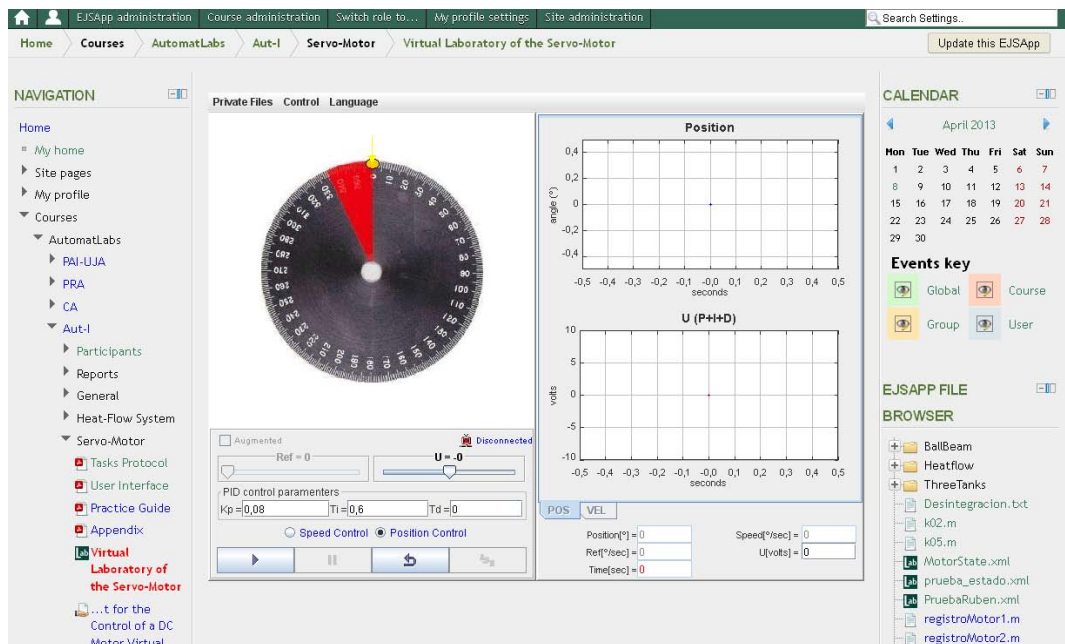


FIGURE B.3: The DC motor virtual laboratory integrated into a Moodle course.

3. Mark the *Add support for collaborative applets* option. This allows using the EJSApp Collab Session Moodle block so users can work together with the same EJS applet.
4. Recompile the EJS application source code with a new version of EJS. This assures that the resulting EJS applet will be compatible with the EJSApp plugins for Moodle.

To give an idea of how much work this implies, it is important to highlight that no more than fifteen minutes were needed to complete the previous steps with any of the three adapted VRLs. Figures B.1, B.2 and B.3 show these EJS VL applications embedded into the UNEDLabs Moodle website.

B.3. Remote Laboratories

Old VRLs in AutomatLabs always required to use a public IP for the *Lab PC* controlling the hardware of the RL. However, as shown in figure 2.4 from Chapter 2, UNEDLabs can work either with public or private IPs. In the latter case, SARLAB must be used. As explained in section 5.1.3, this requires the use of the SARLAB element.

In order to demonstrate that the tools and the framework presented in this work support the easy integration of both solutions (RLs that either use SARLAB or not), these three applications were added to UNEDLabs using public IPs and, therefore, they do not need the use of SARLAB.

This means that the only required changes to make the older VRLs applications used in AutomatLabs work as RLs in UNEDLabs were: 1) separate the VL from

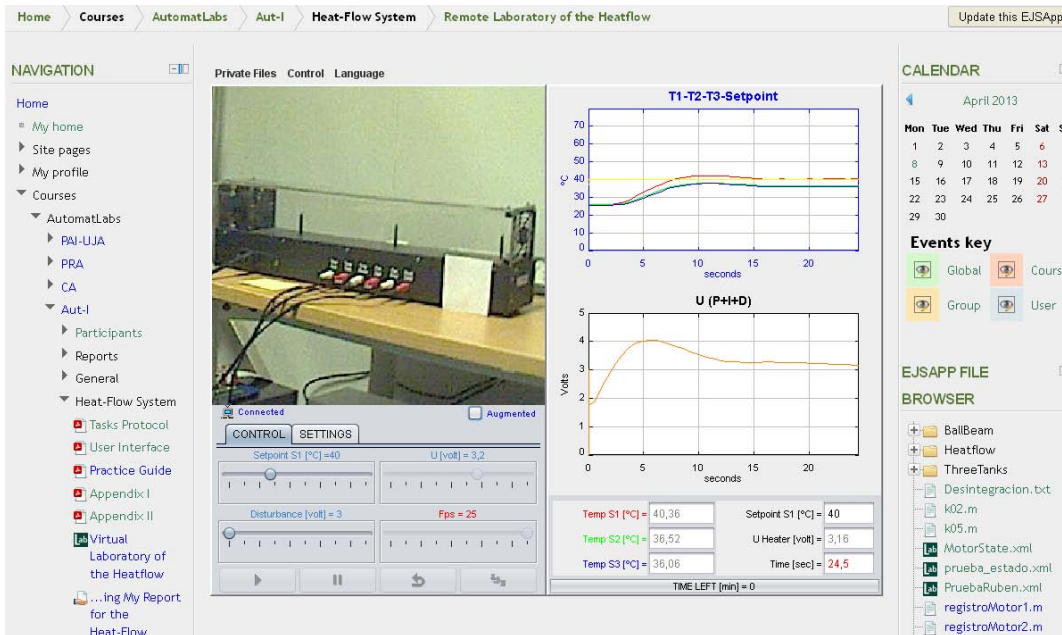


FIGURE B.4: The heatflow remote laboratory integrated into a Moodle course.

The screenshot displays a Moodle course page titled "Lab Collaborative Session". The main content area shows a remote laboratory interface for three coupled tanks. The interface is divided into several sections:

- NAVIGATION:** Includes "Home", "My courses", "Aut-1", "Collaborative session", and "Virtual and Remote Labs for the Three Coupled Tanks".
- Private Files Control Language:** A header for the lab interface.
- Video Feed:** A live video stream of the physical laboratory setup, showing three red tanks and associated piping.
- CONTROL PID VIDEO:** A control panel with various sliders and buttons. Parameters include:
 - A13% = 100, A32% = 100, A2100% =
 - A1eak1% = 0, A1eak2% = 0, A1eak3% = 0
 - Pump1% = 31, Pump2% = 52, FPS = 25
- Graphs:**
 - H1 - H2 - H3 - Setpoint:** A line graph showing pressure (mm) vs. time (seconds) for three tanks. The y-axis ranges from 0 to 500 mm, and the x-axis ranges from 0 to 100 seconds. Three lines (red, yellow, blue) show increasing pressure over time.
 - Pump1 - Pump2:** A line graph showing pump percentages vs. time (seconds). The y-axis ranges from 0 to 100%, and the x-axis ranges from 0 to 100 seconds. Two lines (green, yellow) show decreasing pump percentages over time.
- Status and Data:**
 - Status: "Remote mode with labserver. Connected"
 - Parameters: H1 (mm) = 340,63; H2 (mm) = 200,23; Pump1 (%) = 31,38; Pump2 (%) = 52,14; SP1 (mm) = 300; SP2 (mm) = 200; Time (sec) = 103; LEFT TIME: 27005627
- CALENDAR:** Shows a calendar for February 2012.
- Events key:** Includes "Global", "Course", "Group", and "User".
- VRLAB FILE BROWSER:** Lists files like "HookeSim_graph1.jpg", "HookeSim_graph2.jpg", and "HookeSim.txt".

FIGURE B.5: The three tanks remote laboratory integrated into a Moodle course.

the RL, 2) add the automatic language translation support, 3) use the appropriate EJS methods for saving/reading data, 4) tick the *Add support for collaborative applets* option if desired and 5) recompile the EJS application source code with a new version of EJS. To sum up, preparing the RLs did not required any additional work than preparing the VLs.

Figures B.4, B.5 and B.6 show the three RLs embedded into Moodle.

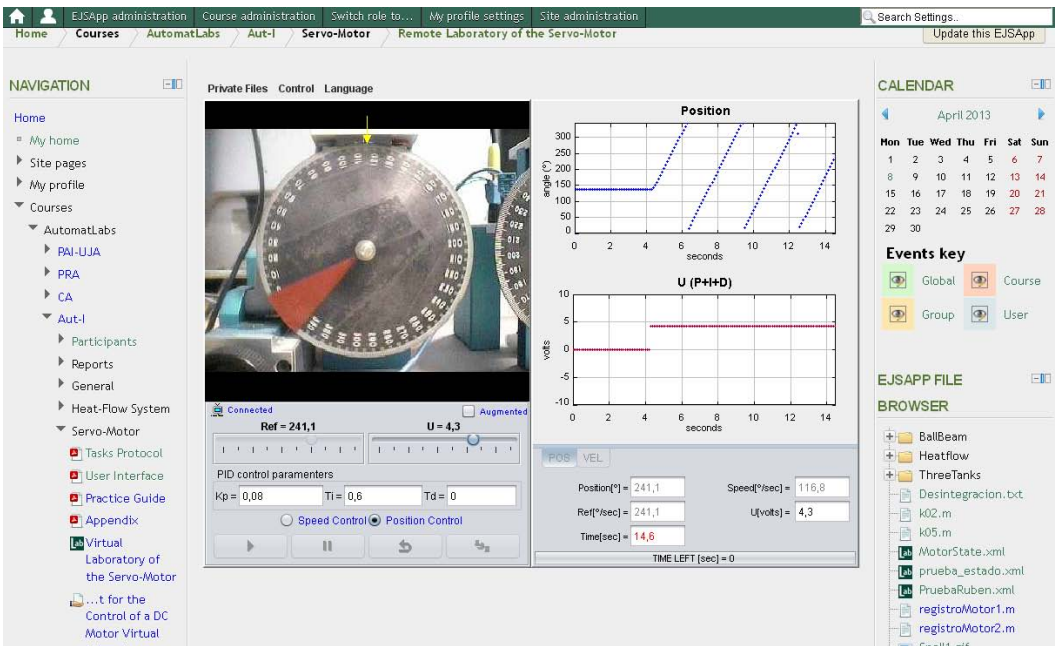


FIGURE B.6: The DC motor remote laboratory integrated into a Moodle course.

Appendix C

Developer Documentation of the EJSApp Plugins

This appendix presents the developer documentation deployed along with the EJSApp plugins code.

C.1. EJSApp

Removes non alphanumeric symbols from a string.

```
delete_non_alphanumeric_symbols($str) : string
```

Parameters

`$str`

string Text with alphanumeric symbols

Returns

string Text without the alphanumeric symbols

Deletes a directory from the server.

```
delete_recurisvely(\file $dir) : bool
```

Parameters

`$dir`

```
\file Directory to delete
```

Returns

```
bool True on success or false on failure
```

Returns the html code to draw an ejsapp instance.

```
draw_ejsapp_instance($ejsappid, $state_file, $width, $height)
```

Parameters

```
$ejsappid
    int ID of an instance of this module
$state_file
    string url to the state file
$width
    int width for drawing the applet
$height
    int height for drawing the applet
```

Returns

```
string Html code that embeds the EJS applet
```

Given an object containing all the necessary data, (defined by the plugin mform) this function will create a new instance and return the id number of the new instance.

```
ejsapp_add_instance(object $ejsapp, $mform) : int
```

Parameters

```
$ejsapp
    object An object from the form in mod_form.php
$mform
```

Returns

```
int The id of the newly inserted ejsapp record
```

Function to be run periodically according to the moodle cron This function searches for things that need to be done, such as sending out mail, toggling flags etc.

```
ejsapp_cron() : boolean
```

```
..
```

Returns

```
boolean
```

Given an ID of an instance of this module, this function will permanently delete the instance and any data that depends on it.

```
ejsapp_delete_instance(int $id) : boolean
```

Parameters

```
    $id
        int Id of the module instance
```

Returns

```
    boolean Success/Failure
```

Returns the lists of all browsable file areas within the given module context.

```
ejsapp_get_file_areas(\stdClass $course, \stdClass $cm, \stdClass $context) :
```

```
    array
```

The file area 'intro' for the activity introduction field is added

```
    automatically by file_browser::get_file_info_context_module()
```

Parameters

```
    $course
        \stdClass
    $cm
        \stdClass
    $context
        \stdClass
```

Returns

```
    arrayof [(string)filearea] => (string)description
```

File browsing support for ejsapp file areas.

```
ejsapp_get_file_info(\file_browser $browser, array $areas, \stdClass $course,
    \stdClass $cm, \stdClass $context, string $filearea, int $itemid, string
    $filepath, string $filename) : \file_info
```

Parameters

```
    $browser
        \file_browser
    $areas
        array
    $course
        \stdClass
    $cm
        \stdClass
    $context
        \stdClass
```

```

$filearea
    string
$itemid
    int
$filepath
    string
$filename
    string

```

Returns

```
\file_infoinstance or null if not found
```

Must return an array of users who are participants for a given instance of ejsapp.

```
ejsapp_get_participants(int $ejsappid) : boolean | array
```

Must include every user involved in the instance, independent of his role (student, teacher, admin...).

Parameters

```

$ejsappid
    int ID of an instance of this module

```

Returns

```
boolean array False if no participants, array of objects otherwise
```

Prepares the recent activity data.

```
ejsapp_get_recent_mod_activity(array $activities, int $index, int $timestart,
    int $courseid, int $cmid, int $userid, int $groupid) : void
```

This callback function is supposed to populate the passed array with custom activity records. These records are then rendered into HTML via `newmodule_print_recent_mod_activity()`.

Parameters

```

$activities
    array sequentially indexed array of objects with the 'cmid'
property
$index
    int the index in the $activities to use for the next record
$timestart
    int append activity since this time
$courseid
    int the id of the course we produce the report for
$cmid

```

```

        int course module id
    $userid
        int check for a particular user's activity only, defaults to 0
        (all users)
    $groupid
        int check for a particular group's activity only, defaults to
    0 (all groups)

```

Serves the files from the ejsapp file areas.

```

ejsapp_pluginfile(\stdClass $course, \stdClass $cm, \stdClass $context, string
    $filearea, array $args, bool $forcedownload, array $options) : \file

```

Parameters

```

    $course
        \stdClass the course object
    $cm
        \stdClass the course module object
    $context
        \stdClass the ejsapp's context
    $filearea
        string the name of the file area
    $args
        array extra arguments (itemid, path)
    $forcedownload
        bool whether or not force download
    $options
        array additional options affecting the file serving

```

Returns

```

    \fileserved file

```

Given a course and a time, this module should find recent activity that has occurred in ejsapp activities and print it out.

```

ejsapp_print_recent_activity($course, $viewfullnames, $timestart) : boolean

```

Parameters

```

    $course
        \stdClass the course object
    $viewfullnames
    $timestart

```

Returns

```

    boolean true if there was output, or false if there was none.

```

Prints single activity item

```
ejsapp_print_recent_mod_activity($activity, $courseid, $detail, $modnames,
$viewfullnames) : void
```

Parameters

```

    $activity
    $courseid
        int the id of the course we produce the report for
    $detail
    $modnames
    $viewfullnames
```

Moodle features supported by EJSApp.

```
ejsapp_supports(\constant $feature) : boolean
```

Parameters

```

    $feature
        \constant feature to be supported
```

Returns

```
boolean true if EJSApp supports the feature, false elsewhere
```

Given an object containing all the necessary data, (defined by the form in the plugin mform) this function will update an existing instance with new data.

```
ejsapp_update_instance(object $ejsapp, $mform) : boolean
```

Parameters

```

    $ejsapp
        object An object from the form in mod_form.php
    $mform
```

Returns

```
boolean Success/Fail
```

Print a detailed representation of what a user has done with a given particular instance of this module, for user activity reports.

```
ejsapp_user_complete($course, $user, $mod, $ejsapp) : boolean
```

Parameters

```

    $course
        \stdClass the course object
    $user
    $mod
```

```

    $ejsapp
        \stdClass record from table ejsapp

```

Returns

```

    boolean

```

Return a small object with summary information about what a user has done with a given particular instance of this module. Used for user activity reports.

```

ejsapp_user_outline($course, $user, $mod, $ejsapp) : \$return->time

```

Parameters

```

    $course
        \stdClass the course object
    $user
    $mod
    $ejsapp
        \stdClass record from table ejsapp

```

Returns

```

    \$return->time= the time they did it
    \return->info = a short text description

```

Returns the code that embeds an EJS applet into Moodle.

```

generate_applet_embedding_code(\stdClass $ejsapp, \stdClass | null $sarlabinfo
, string | null $state_file, \stdClass | null $collabinfo, \stdClass |
null $external_size) : string

```

This function returns the HTML and JavaScript code that embeds an EJS applet into Moodle. It is used for four different cases: 1) when only the EJSApp activity is being used 2) when the EJSApp File Browser is used to load a state file 3) when the EJSApp Collab Session is used 4) when third party plugins want to display EJS applets in their own activities by means of the EJSApp external interface

Parameters

```

    $ejsapp
        \stdClass record from table ejsapp
    $sarlabinfo
        \stdClassnull
        $sarlabinfo->instance: int sarlab id,
        $sarlabinfo->practice: int practice id, Null if sarlab is not
used
    $state_file

```

stringnull if generate_applet_embedding_code is called from block ejsapp_file_browser it is the name of the xml file that stores the state of an EJS applet, elsewhere it is null

```

$collabinfo
    \stdClassnull
    $collabinfo->session: int collaborative session id,
    $collabinfo->ip: string collaborative session ip,
    $collabinfo->port: int collaborative session port,
    $collabinfo->director: int id of the collaborative session
master user, ' Null if generate_applet_embedding_code is not called from
block ejsapp_collab_session
$external_size
    \stdClassnull
    $external_size->width: int value (in pixels) for the width of
the applet to be drawn
    $external_size->height: int value (in pixels) for the height
of the applet to be drawn Null if generate_applet_embedding_code is not
called from the external interface (draw_ejsapp_instance() function)

```

Returns

```
string code that embeds an EJS applet into Moodle
```

Gives a list of the existing ejsapp instances.

```
get_ejsapp_instances($course_id)
```

Parameters

```

$course_id
    int the id of the course we produce the report for

```

Returns the size of a particular ejsapp instance.

```
get_ejsapp_size($ejsapp_id)
```

Parameters

```

$ejsapp_id
    int id record from table ejsapp

```

Returns a list of state files saved by a particular ejsapp instance.

```
get_ejsapp_states($ejsapp_id)
```

Parameters

```

$ejsapp_id
    int id record from table ejsapp

```

Returns all other caps used in the module.

```
newmodule_get_extra_capabilities() : array
```

Returns

```
array
```

Examples

Updates the EJSApp tables according to the .jar information.

```
update_db(\stdClass $ejsapp, int $contextid)
```

Parameters

```
$ejsapp
```

```
\stdClass record from table ej sapp
```

```
$contextid
```

```
int
```

Install function for the ej sapp module.

```
xml db_ej sapp_install()
```

Custom uninstallation procedure.

```
xml db_ej sapp_uninstall()
```

Upgrade function for the ej sapp module.

```
xml db_ej sapp_upgrade(string $oldversion)
```

Parameters

```
$oldversion
```

```
string
```

C.2. EJSApp Booking System

Saves a new instance of the ej sappbooking into the database.

```
ej sappbooking_add_instance(object $ej sappbooking) : int
```

Given an object containing all the necessary data, (defined by the form in mod_form.php) this function will create a new instance and return the id number of the new instance.

Parameters

```
$ej sappbooking
```

```
object an object from the form in mod_form.php
```

Returns

```
int The id of the newly inserted ejsappbooking record
```

Function to be run periodically according to the moodle cron. This function checks whether all users and all ejsapps are in the ejsappbookingusersaccess database. If not, it adds them..

```
ejsappbooking_cron() : boolean
```

It is necessary when a ejsappbooking instance has already been added to a course and then new ejsapp are added or new users are enrolled to that course. Also, the function checks if there are users and ejsapps in the ejsappbooking_usersaccess database that no longer exist. In that case, it deletes them.

Returns

```
boolean
```

Removes an instance of the ejsappbooking from the database.

```
ejsappbooking_delete_instance(int $id) : boolean
```

Given an ID of an instance of this module, this function will permanently delete the instance and any data that depends on it.

Parameters

```
$id
    int id of the module instance
```

Returns

```
boolean Success/Failure
```

Extends the global navigation tree by adding ejsappbooking nodes if there is a relevant content.

```
ejsappbooking_extend_navigation(\navigation_node $navref, \stdClass $course, \
    stdClass $module, \cm_info $cm)
```

This can be called by an AJAX request so do not rely on \$PAGE as it might not be set up properly.

Parameters

```
$navref
    \navigation_node an object representing the navigation tree
node of the ejsappbooking module instance
$course
```



```

        \stdClass
    $module
        \stdClass
    $cm
        \cm_info

```

Extends the settings navigation with the ejsappbooking settings.

```
ejsappbooking_extend_settings_navigation(\settings_navigation $settingsnav, \
    navigation_node $ejsappbookingnode)
```

This function is called when the context for the page is a ejsappbooking module. This is not called by AJAX so it is safe to rely on the \$PAGE.

Parameters

```

    $settingsnav
        \settings_navigation {@link settings_navigation}
    $ejsappbookingnode
        \navigation_node {@link navigation_node}

```

Returns all other caps used in the module.

```
ejsappbooking_get_extra_capabilities() : array
```

Returns

```
array
```

Returns the lists of all browsable file areas within the given module context.

```
ejsappbooking_get_file_areas(\stdClass $course, \stdClass $cm, \stdClass
    $context) : array
```

The file area 'intro' for the activity introduction field is added automatically by file_browser::get_file_info_context_module()

Parameters

```

    $course
        \stdClass
    $cm
        \stdClass
    $context
        \stdClass

```

Returns

```
arrayof [(string)filearea] => (string)description
```

File browsing support for ejsappbooking file areas.

```
ejsappbooking_get_file_info(\file_browser $browser, array $areas, \stdClass
    $course, \stdClass $cm, \stdClass $context, string $filearea, int $itemid,
    string $filepath, string $filename) : \file_info
```

Parameters

```
    $browser
        \file_browser
    $areas
        array
    $course
        \stdClass
    $cm
        \stdClass
    $context
        \stdClass
    $filearea
        string
    $itemid
        int
    $filepath
        string
    $filename
        string
```

Returns

```
\file_infoinstance or null if not found
```

Prepares the recent activity data.

```
ejsappbooking_get_recent_mod_activity(array $activities, int $index, int
    $timestart, int $courseid, int $cmid, int $userid, int $groupid) : void
```

This callback function is supposed to populate the passed array with custom activity records. These records are then rendered into HTML via `ejsappbooking_print_recent_mod_activity()`.

Parameters

```
    $activities
        array sequentially indexed array of objects with the 'cmid'
property
    $index
        int the index in the $activities to use for the next record
    $timestart
        int append activity since this time
```

```

$courseid
    int the id of the course we produce the report for
$cmid
    int course module id
$userid
    int check for a particular user's activity only, defaults to 0
    (all users)
$groupid
    int check for a particular group's activity only, defaults to
0
    (all groups)

```

Creates or updates grade item for the give ejsappbooking instance.

```
ejsappbooking_grade_item_update(\stdClass $ejsappbooking) : void
```

Needed by grade_update_mod_grades() in lib/gradelib.php

Parameters

```

$ejsappbooking
    \stdClass instance object with extra cmidnumber and modname
    property

```

Serves the files from the ejsappbooking file areas.

```
ejsappbooking_pluginfile(\stdClass $course, \stdClass $cm, \stdClass $context,
    string $filearea, array $args, bool $forcedownload, array $options)
```

Parameters

```

$course
    \stdClass the course object
$cm
    \stdClass the course module object
$context
    \stdClass the ejsappbooking's context
$filearea
    string the name of the file area
$args
    array extra arguments (itemid, path)
$forcedownload
    bool whether or not force download
$options
    array additional options affecting the file serving

```

Given a course and a time, this module should find recent activity that has occurred in ejsappbooking activities and print it out.

```
ejsappbooking_print_recent_activity($course, $viewfullnames, $timestart) :
```

```
    boolean
```

```
Return true if there was output, or false is there was none.
```

```
Parameters
```

```
    $course
```

```
    $viewfullnames
```

```
    $timestart
```

```
Returns
```

```
    boolean
```

Prints single activity item

```
ejsappbooking_print_recent_mod_activity($activity, $courseid, $detail,  
    $modnames, $viewfullnames) : void
```

```
Parameters
```

```
    $activity
```

```
    $courseid
```

```
    $detail
```

```
    $modnames
```

```
    $viewfullnames
```

Is a given scale used by the instance of ejsappbooking?

```
ejsappbooking_scale_used(int $ejsappbookingid, bool $scaleid) : bool
```

```
This function returns if a scale is being used by one ejsappbooking if it has  
support for grading and scales. Commented code should be modified if  
necessary. See forum, glossary or journal modules as reference.
```

```
Parameters
```

```
    $ejsappbookingid
```

```
        int ID of an instance of this module
```

```
    $scaleid
```

```
        bool
```

```
Returns
```

```
    bool True if the scale is used by the given ejsappbooking instance
```

Checks if scale is being used by any instance of ejsappbooking.

```
ejsappbooking_scale_used_anywhere($scaleid) : boolean
```

```
This is used to find out if scale used anywhere.
```

Parameters

```
$scaleid
    int
```

Returns

```
boolean True if the scale is used by any ejsappbooking instance
```

Returns the information on whether the module supports a feature.

```
ejsappbooking_supports(string $feature) : mixed
```

```
see in lib/moodlelib.php
```

Parameters

```
$feature
    string FEATURE_xx constant for requested feature
```

Returns

```
mixed True if the feature is supported, null if unknown
```

Update ejsappbooking grades in the gradebook.

```
ejsappbooking_update_grades(\stdClass $ejsappbooking, int $userid) : void
```

```
Needed by grade_update_mod_grades() in lib/gradelib.php
```

Parameters

```
$ejsappbooking
    \stdClass instance object with extra cmidnumber and modname
    property
```

```
$userid
    int update grade of specific user only, 0 means all
    participants
```

Updates an instance of the ejsappbooking in the database.

```
ejsappbooking_update_instance(object $ejsappbooking) : boolean
```

```
Given an object containing all the necessary data, (defined by the form in
    mod_form.php) this function will update an existing instance with new data
```

Parameters

```
$ejsappbooking
    object an object from the form in mod_form.php
```

Returns

```
boolean Success/Fail
```

Prints a detailed representation of what a user has done with a given particular instance of this module, for user activity reports.

```
ejsappbooking_user_complete(\stdClass $course, \stdClass $user, \cm_info $mod,
    \stdClass $ejsappbooking) : \void
```

Parameters

```
    $course
        \stdClass the current course record
    $user
        \stdClass the record of the user we are generating report for
    $mod
        \cm_info course module info
    $ejsappbooking
        \stdClass the module instance record
```

Returns

```
\void is supposed to echo directly
```

Returns a small object with summary information about what a user has done with a given particular instance of this module Used for user activity reports.

```
ejsappbooking_user_outline(int $course, $user, $mod, $ejsappbooking) : \
    stdClass | null
$return->time = the time they did it $return->info = a short text description
```

Parameters

```
    $course
        int
    $user
    $mod
    $ejsappbooking
```

Returns

```
\stdClass null
```

Returns the course last access.

```
get_course_lastaccess_sql(int $accesssince)
```

Parameters

```
    $accesssince
        int
```

Returns the user last access.

```
get_user_lastaccess_sql(int $accesssince)
```

Parameters

```
    $accesssince
```

int

Post installation procedure.

```
xmldb_ejsappbooking_install()  
see      \global\upgrade_plugins_modules()
```

Post installation recovery procedure.

```
xmldb_ejsappbooking_install_recovery()  
see      \global\upgrade_plugins_modules()
```

Custom uninstallation procedure.

```
xmldb_ejsappbooking_uninstall()
```

Custom uninstallation procedure. Execute ejsappbooking upgrade from the given old version.

```
xmldb_ejsappbooking_upgrade(int $oldversion) : bool
```

Parameters

```
    $oldversion  
        int
```

Returns

```
    bool Constants
```

C.3. EJSApp File Browser

C.3.1. Methods of the block ejsapp file browser class

Applicable formats function for the EJSApp File Browser block.

```
applicable_formats()  
Defines the content of the block.  
get_content() : array  
no      params  
Returns  
    array The content of the block
```

Defines the javascript code for refreshing the page in which the block is embedded.

```
get_javascript_code() : string
no      params
Returns
      string Javascript code for refreshing the page
```

Init function for the EJSApp File Browser block.

```
init()
```

Allow multiple instances function for the EJSApp File Browser block.

```
instance_allow_multiple()
```

Specialization function for the EJSApp File Browser block.

```
specialization()
```

C.3.2. Methods of the block ejsapp file browser form class

Minimalistic edit form definition.

```
definition()
```

C.3.3. Methods of the block ejsapp file browser renderer class

Prints ejsapp file browser tree view.

```
ejsapp_file_browser_tree() : string
Returns
      string Html code that prints the tree view
```

Prints ejsapp file browser tree view.

```
render_ejsapp_file_browser_tree(\ejsapp_file_browser_tree $tree) : string
```

Parameters

```
    $tree
        \ejsapp_file_browser_tree
```

Returns

```
    string Html code that prints the tree view
```

C.3.4. Methods of the ej sapp file browser tree class

Constructor

```
__construct()
```

C.4. EJSApp Collab Sessions

Returns true if the caller is a the master user of a collaborative session.

```
am_i_master_user()
```

Initialization function that creates all tables required by the ej sappcollab session block.

```
create_non_existing_tables()
```

Finishes a collaborative session.

```
delete_collaborative_session(int $master_user)
```

Parameters

```
    $master_user
        int id of the master user
```

Drops out a user from the collaborative session.

```
delete_me_as_collaborative_user()
```

Drops out a non master user from the collaborative session.

```
delete_non_master_user_from_collaborative_users()
```

Returns true if the input table exists in the moodle database.

```
does_the_table_exists(string $table_name)
```

Parameters

```
    $table_name  
        string
```

Returns the names of all collaborative EJS simulations in a given course.

```
get_all_collaborative_lab_names(int $course)
```

Parameters

```
    $course  
        int id of the course
```

Returns the id of the collaborative session directed by a given master user.

```
get_collaborative_session_id(int $master_user)
```

Parameters

```
    $master_user  
        int id of the master user
```

Returns the course id that where the collaborative session is being executed.

```
get_course(int $session_id)
```

Parameters

```
    $session_id  
        int id of the collaborative session
```

Returns the course last access.

```
get_course_lastaccess_sql(int $accesssince)
```

Parameters

```
    $accesssince  
        int
```

Returns the name of a course.

```
get_course_name(int $course_id)
```

Parameters

```
    $course_id  
        int id of the course
```

Returns the id of the EJS simulation shared in a collaborative session.

```
get_ejsapp(int $master_user)
```

Parameters

```
    $master_user
        int id of the master user
```

Returns the name of the EJS simulation shared in a collaborative session.

```
get_ejsapp_name(int $master_user)
```

Parameters

```
    $master_user
        int id of the master user
```

Returns an object that fully describes the simulation shared in a collaborative session.

```
get_ejsapp_object(int $session)
```

Parameters

```
    $session
        int id of the collaborative session
```

Returns an object that fully describes the master user that leads a collaborative session.

```
get_master_user_object(int $session)
```

Parameters

```
    $session
        int id of the collaborative session
```

Returns the connection port that the session director is using in the collaborative session.

```
get_port(int $session_id)
```

Parameters

```
    $session_id
        int id of the collaborative session
```

Returns an list of all collaborative sessions where I have been invited.

```
get_sessions_where_i_am_invited()
```

Returns the user last access.

```
get_user_lastaccess_sql(int $accesssince)
```

Parameters

```
    $accesssince
        int
```

Returns the name of a user.

```
get_user_name(int $user)
```

Parameters

```
    $user
        int id of user
```

Returns true if the user has been invited in some collaborative session.

```
has_the_user_been_invited_to_any_session()
```

Creates a collaborative invitation.

```
insert_collaborative_invitation(int $invited_user, int $collaborative_session)
```

Parameters

```
    $invited_user
        int id of the invited user
    $collaborative_session
        int id of the collaborative session
```

Creates a new collaborative session.

```
insert_collaborative_session(int $port, int $ejsapp, int $master_user, int $ip
, int $course)
```

Parameters

```
    $port
        int connection port of the master user
    $ejsapp
        int id of the EJS simulation to be shared
    $master_user
        int id of the master user
    $ip
        int connection ip of the session director
    $course
        int id of the course that includes the collaborative EJS
simulation
```

Includes a user into a new collaborative session.

```
insert_collaborative_user(int $id, int $ip, int $collaborative_session)
```

Parameters

```
    $id
        int user id
    $ip
        int connection ip of the session director
    $collaborative_session
        int id of the collaborative session
```

Returns true if the user is participating in some collaborative session.

```
is_the_user_participating_in_any_session()
```

Bibliography

- [1] Keith Harry. *Higher education through open and distance learning*. Routledge, 1999.
- [2] I. Gustavsson, K. Nilsson, J. Zackrisson, J. Garcia-Zubia, U. Hernandez-Jayo, A. Nafalski, Z. Nedic, O. Gol, J. Machotka, M. I. Pettersson, T. Lago, and L. Hkansson. On objectives of instructional laboratories, individual assessment, and use of collaborative remote laboratories. *IEEE Transactions on Learning Technologies*, 2(4):263–274, 2009. ISSN 1939-1382.
- [3] G. C. Goodwin, A. M. Medioli, W. Sher, L. B. Vlacic, and J. S. Welsh. Emulation-based virtual laboratories: A low-cost alternative to physical experiments in control engineering education. *IEEE Transactions on Education*, 54:48–55, 2011.
- [4] Gao-Wei Chang, Zong-Mu Yeh, Hsiu-Ming Chang, and Shih-Yao Pan. Teaching photonics laboratory using remote-control web technologies. *IEEE Transactions on Education*, 48(4):642–651, 2005.
- [5] S.C. Sivakumar, W. Robertson, M. Artimy, and N. Aslam. A web-based remote interactive laboratory for internetworking education. *IEEE Transactions on Education*, 48(4):586–598, 2005.

- [6] B. Alhalabi, D. Marcovitz, K. Hamza, and M. Petrie. Remote labs: An innovative leap in the world of distance education. In *Proc. of the 4th Multi Conf. on Systemic, Cybern. and Informatics*, pages 303–307, 2000.
- [7] J. Gorrel. Outcomes of using computer simulations. *Journal of Research on Computer Education*, 24:359–366, 1992.
- [8] D. Grimaldi and S. Rapuano. Hardware and software to design virtual laboratory for education in instrumentation and measurement. *Measurement*, 24(4):485–493, 2009.
- [9] R. Kozma. Will media influence learning? reframing the debate. *Education Technology Research and Development*, 42:7–19, 1994.
- [10] J. Gratch, J. Kelly, and C. Bradley. Science simulations: What do they contribute to student learning? In *Society for Information Technology and Teacher Education International Conference*, pages 3422–3425 Chesapeake, USA, Chesapeake, USA, 2007.
- [11] C. Concari, S. Giorgi, C. Camara, and N. Giacosa. *Current Developments in Technology-Assisted Education*, chapter Didactic strategies using simulations for Physics teaching, pages 2041–4026. Formatex, 2006.
- [12] I. Santana, M. Ferre, E. Izaguirre, R. Aracil, and L. Hernandez. Remote laboratories for education and research purposes in automatic control systems. *IEEE Transactions on Industrial Informatics*, 9:547–556, 2013.
- [13] J. Osborne and S. Hennessy. Literature review in science education and the role of ict: Promise, problems and future directions. Technical Report 6, Futurelab, 2003.

- [14] H. Urban-Woldron. Innovative teaching and learning scenarios using interactive simulations. In *Multimedia Physics Teaching and Learning*, Udine, Italy, 2009.
- [15] L. Lunce. Simulations: Bringing the benefits of situated learning to traditional classroom. *Journal of Applied Educational Technology*, 3(1):37–45, 2006.
- [16] P.A. Hatherly, S.E. Jordan, and A. Cayless. Interactive screen experiments innovative virtual laboratories for distance learners. *European Journal of Physics*, 30:751–762, 2009.
- [17] F. K. Hwang. Ntnu virtual physics laboratory: Java simulations in physics. In *Multimedia in Physics Teaching and Learning Conference*, 2009.
- [18] W. Christian and M. Belloni. *Physlet Physics*. USA: Prentice Hall, 2004.
- [19] Open source physics collection. URL <http://www.compadre.org/osp/>.
- [20] S. Kocijancic. Online experiments in physics and technology teaching. *IEEE Transactions on Education*, 45:26–32, 2002.
- [21] F. Schauer, I. Kuritka, and F. Lustig. *iNEER Special Volume: Innovations 2006*, chapter Creative Laboratory Experiments for Basic Physics using Computer Data Collection and Evaluation Exemplified on the Intelligent School Experimental System (ISES), pages 305–312. Potomac, 2006.
- [22] Sang Tae Park, Heebok Lee, Keun Cheol Yuk, and Heeman Lee. Web-based nuclear physics laboratory. In *Recent Research Developments in Learning Technologies (2005)*, pages 1165–1169, 2005.

- [23] F. Schauer, F. Lustig, and M. Ozvoldova. Remote scientific experiments across internet: Photovoltaic cell characterization. In *International Conference on Interactive Collaborative Learning*, 2006.
- [24] D. Gurkan, A. Mickelson, and D. Benhaddou. Remote laboratories for optical circuits. *IEEE Transactions on Education*, 51:53–60, 2008.
- [25] M. E. Macias and I. Mendez. elab - remote electronics lab in real time. In *Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007. FIE '07. 37th Annual*, pages S3G–12 –S3G–17, 2007. doi: 10.1109/FIE.2007.4418154.
- [26] I. Moon, S. Han, K. Choi, D. Kim, C. Jeon, S. Lee, and S. Woo. A remote laboratory for electric circuit using passive devices controlled. In *Proceedings of the International Conference on Engineering Education*, 2008.
- [27] S. Grober, M. Vetter, B. Eckert, and H. J. Jodl. Remotely controlled laboratories: Aims, examples and exercises. *American Journal of Physics*, 76:374–378, 2008.
- [28] S. Grober, M. Vetter, B. Eckert, and H. J. Jodl. Experimenting from a distance - remotely controlled laboratory (rcl). *European Journal of Physics*, 28:127–141, 2007.
- [29] Jing Ma and Jeffrey V. Nickerson. Hands-on, simulated, and remote laboratories: A comparative literature review. *ACM Computing Surveys*, 38, 2006.
- [30] Christophe Gravier, Jacques Fayolle, Bernard Bayard, Mikael Ates, and Jeremy Lardon. State of the art about remote laboratories paradigms -

- foundations of ongoing mutations. *International Journal of Online Engineering*, 4(1), 2008.
- [31] Marco Casini, Domenico Prattichizzo, and Antonio Vicino. Operating remote laboratories through a bootable device. *IEEE Transactions on Industrial Electronics*, 54:3134–3140, 2007.
- [32] Luis Gomes. Current trends in remote laboratories. *IEEE Transactions on Industrial Electronics*, 56:4744–4756, 2009.
- [33] Xuemin Chen, Gangbing Song, and Yongpeng Zhang. Virtual and remote laboratory development: A review. In *Earth and Space 2010: Engineering, Science, Construction, and Operations in Challenging Environments*, 2010.
- [34] E. San-Cristobal-Ruiz, P. Baley, M.A. Castro-Gil, K. DeLong, J. Hardison, and J. Harward. Integration view of web labs and learning management systems. In *IEEE EDUCON*, 2010.
- [35] Massachusetts institute of technology - ilab. URL <http://ilab.mit.edu/wiki>.
- [36] J. L. Hardison, K. DeLong, P. H. Bailey, and V. J. Harward. Deploying interactive remote labs using the ilab shared architecture. In *Frontiers in Education Conference*, 2008.
- [37] Martin Magdin, Martin Cpay, and Martin Halme. Implementation of logicsim in lms moodle. In *IEEE International Conference on Emerging eLearning Technologies and Applications*, 2012.
- [38] LMS Review Subgroup. Learning management system (lms) review summary of findings. Technical report, Duke University, 2009.

- [39] Brian Stewart, Derek Briton, Mike Gismondi, Bob Heller, Dietmar Kenepohl, Rory McGreal, and Christine Nelson. Choosing moodle: An evaluation of learning management systems at athabasca university. *International Journal of Distance Education Technologies*, 5, 2007.
- [40] Sara Bennett. Learning management systems: A review. Technical report, AUT University, 2011.
- [41] CampusComputing. A profile of the lms market. Technical report, The Campus Computing Project, 2011.
- [42] E-learning guild. URL <http://www.elearningguild.com/content/>.
- [43] Centre for learning & performance technologies. URL <http://c4lpt.co.uk/top-100-tools/best-of-breed-tools-2012/>.
- [44] H. Vargas, J. Sanchez, and S. Dormido. The spanish university network of web-based laboratories for control engineering education: The automatl@bs project. In *European Control Conference*, 2009.
- [45] R. Dormido, H. Vargas, and et al. Development of a web-based control laboratory for automation technicians: The three-tank system. *IEEE Transactions on Education*, 51:35–43, 2008.
- [46] 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*, 14:505–524, 2008.
- [47] L. de la Torre, R. Heradio, J. Snchez, S. Dormido, J.P. Sanchez, C. Carreras, and M. Yuste. A thin lens virtual and remote laboratory at the new

- fislabs portal. In *Multimedia in Physics Teaching and Learning - International Conference on Hands on Science (MPTL-HSCI)*, 2011.
- [48] L. de la Torre, J.P. Snchez, J. Snchez, S. Dormido, M. Yuste, and C. Carreras. The virtual and remote laboratory for snells law at the fislabs portal. In *Groupe International de Recherche sur l'Enseignement de la Physique - International Commission on Physics Education - Multimedia in Physics Teaching and Learning (GIREP-ICPE-MPTL)*, 2010.
- [49] L. de la Torre, J. Snchez, and S. Dormido. The fisl@bs portal: A network of virtual and remote laboratories for physics education. In *Multimedia in Physics Teaching and Learning (MPTL)*, 2009.
- [50] L. de la Torre, J. Sanchez, J. P. Sanchez, M. Yuste, and C. Carreras. Two web-based laboratories of the fisl@bs network: Hooke's and snell's laws. *European Journal of Physics*, 32:571–584, 2011.
- [51] J. Travis. *Using Java with Labview*, chapter Internet Applications in LabView. USA: Prentice-Hall, 2000.
- [52] T.K. Iversen, K.J. Kristoffersen, K.G. Larsen, M. Laursen, R.G. Madsen, S.K. Mortensen, P. Pettersson, and C.B. Thomasen. Model-checking real-time control programs: verifying lego mindstormstm systems using uppaal. In *12th Euromicro Conference on Real-Time Systems*, 2000.
- [53] P. Trung. Development of vision service in robotics studio for road signs recognition and control of lego mindstorms robot. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, 2009.
- [54] S. H. Kim and J. W. Jeon. Introduction for freshmen to embedded systems using lego mindstorms. *IEEE Transactions on Education*, 2009:99–108, 52.

- [55] Y. Kim. Control systems lab using a lego mindstorms nxt motor system. *IEEE Transactions on Education*, 54:452–461, 2011.
- [56] J. M. Gomez de Gabriel, A. Mandow, J. Fernandez-Lozano, , and A. J. Garcia-Cerezo. Using lego nxt mobile robots with labview for undergraduate courses on mechatronics. *IEEE Transactions on Education*, 54:41–47, 2011.
- [57] M. Pinto, A. P. Moreira, and A. Matos. Localization of mobile robots using an extended kalman filter in a lego nxt. *IEEE Transactions on Education*, 55:135–144, 2012.
- [58] W. Christian and F. Esquembre. Modeling physics with easy java simulations. *The Physics Teacher*, 45:475–480, 2007.
- [59] F. Esquembre. Easy java simulations: A software tool to create scientific simulations in java. *Computer Physics Communications*, 156(6):199–204, January 2004.
- [60] D. Gillet, Anh Vu Nguyen Ngoc, and Y. Rekik. Collaborative web-based experimentation in flexible engineering education. *IEEE Transactions on Education*, 48(4):696–704, nov. 2005. ISSN 0018-9359.
- [61] Héctor Vargas, José Sánchez-Moreno, Sebastián Dormido, Christophe Salzmann, Denis Gillet, and Francisco Esquembre. Web-enabled remote scientific environments. *Computing in Science and Engineering*, 11:36–46, May 2009. ISSN 1521-9615.
- [62] Remotely controlled laboratories - rcls. URL <http://rcl.physik.uni-kl.de/>.
- [63] Internet school experimental system ises. URL <http://www.ises.info/index.php/en/laboratory>.

- [64] F. Schauer, F. Lustig, J. Dvorak, and M. Ozvoldova. An easy-to-build remote laboratory with data transfer using the internet school experimental system. *European Journal of Physics*, 29:751–762, 2008.
- [65] Pendulums remote laboratory of the dpt. of physics. URL <http://www.physics.purdue.edu/class/>.
- [66] Lila project. URL <http://www.lila-project.org/>.
- [67] R. Heradio, L. de la Torre, H. Vargas, J. Snchez, and S. Dormido. An architecture for virtual and remote laboratories to support distance learning. In *Research in Engineering Education Symposium - Book of Proceedings*, 2011.
- [68] L. de la Torre, R. Heradio, H. Vargas, J. Sanchez, and S. Dormido. A framework for implementing virtual and remote laboratories in scientific courses. In *WORLDCOMP 2010 - FECS'11 The Int. Conf. Frontiers in Education: CS and CE*, 2011.
- [69] Luis de la Torre, Juan P. Sanchez, Ruben Heradio, Carmen Carreras, Manuel Yuste, Jose Sanchez, and Sebastian Dormido. Unedlabs - an example of ejs labs integration into moodle. In *World Conference on Physics Education*, 2012.
- [70] S. Dormido, J. Sanchez, L. de la Torre, R. Heradio, C. Carreras, J.P. Sanchez, and M. Yuste. Physics experiments at the unedlabs portal. *International Journal of Online Engineering*, 8:26–27, 2012.
- [71] Andres Mejias, Marco A. Marquez, Jose Manuel Andujar, and Maria Reyes Sanchez. A complete solution for developing remote labs. In *Advances in Control Education 2013*, 2013.

- [72] Ch. Salzmänn and D. Gillet. Challenges in remote laboratory sustainability. In *International Conference on Engineering Education*, Coimbra, Portugal, 2007.
- [73] E. Besada-Portas, J. A. López-Orozco, L. de la Torre, and J.M. de la Cruz-García. Remote control laboratory using ejs applets and twincat programmable logic controllers. *IEEE Transactions on Education*, 56:156–164, 2012.
- [74] E. Fabregas, G. Farias, S. Dormido-Canto, S. Dormido, and F. Esquembre. Developing a remote laboratory for engineering education. *Computers & Education*, 57(2):1686–1697, 2011. ISSN 03601315.
- [75] M. Guinaldo, J. Sánchez, H. Vargas, and S. Dormido. An advanced web-based control laboratory for the ball and beam system. In *9th Portuguese Conference on Automatic Control (CONTROLO2010)*, 2010.
- [76] Hector Vargas, Gonzalo Farias, Sebastian Dormido, Jose Sanchez, Raquel Dormido-Canto, Natividad Duro, Sebastian Dormido-Canto, and Francisco Esquembre. Web-based learning resources for automation technicians vocational training: Illustrated with a heat-flow and a liquid level. In *Advances in Control Education*, Madrid, Spain, 2006.
- [77] H. Vargas, J. Sánchez Moreno, C. A. Jara, F. A. Candelas, F. Torres, and S. Dormido. A network of automatic control web-based laboratories. *IEEE Transactions on Learning Technologies*, 4(3):197–208, 2011. ISSN 1939-1382.
- [78] Isaac Newton. *Opticks or A treatise of the reflections, refractions, inflections and colours of light*. 2nd edition, 1718.

- [79] M. A. Trenas, J. Ramos, E. D. Gutierrez, S. Romero, and F. Corbera. Use of a new moodle module for improving the teaching of a basic course on computer architecture. *IEEE Transactions on Education*, 54:222–228, 2011.
- [80] M. N. K. Boulos, A. D. Taylor, and A. Breton. A synchronous communication experiment within an online distance learning program: A case study. *Telemedicine journal and e-health*, 11(5):583–93, 2005.
- [81] Carlos A. Jara, Francisco A. Candelas, and Fernando Torres. Virtual and remote laboratory for robotics e-learning. In *18th European Symposium on Computer Aided Process Engineering*, 2008.
- [82] C. Lazar and S. Carari. A remote-control engineering laboratory. *IEEE Transactions on Industrial Electronics*, 55:2368–2375, 2008.
- [83] Dimitra Tsovaltzi, Nikol Rummel, Bruce M. McLaren, Niels Pinkwart, Oliver Scheuer, Andreas Harrer, and Isabel Braun. Extending a virtual chemistry laboratory with a collaboration script to promote conceptual learning. *International Journal of Technology Enhanced Learning*, 2:91–110, 2010.
- [84] Eliane G. Guimaraes, Eleri Cardozo, Daniel H. Moraes, and Paulo R. Coelho. Design and implementation issues for modern remote laboratories. *IEEE Transactions on Learning Technologies*, 4:149–161, 2011.
- [85] D. Satterly. *Piaget and Education*. The Oxford Companion to the Mind - Oxford University Press, 1987.
- [86] C. M. Itin. Reasserting the philosophy of experiential education as a vehicle for change in the 21st century. *The Journal of Experimental Education*, 22: 91–98, 1999.

- [87] B. Hanson, P. Culmer, J. Gallagher, K. Page, E. Read, A. Weightman, and M. Levesley. Reload: Real laboratories operated at a distance. *IEEE Transactions on Learning Technologies*, 2(4):331–341, 2009. ISSN 1939-1382.
- [88] M. Wannous and H. Nakano. Nvlab, a networking virtual web-based laboratory that implements virtualization and virtual network computing tech. *IEEE Transactions on Learning Technologies*, 3(2):129–138, 2010. ISSN 1939-1382.
- [89] G. Bafoustou and G. Mentzas. Review and functional classification of collaborative systems. *International Journal on Information Management*, 22:281–305, 2002.
- [90] O. Marjanovic. Learning and teaching in a synchronous collaborative environment. *Journal of Computer Assisted Learning*, 15:129–138, 1999.
- [91] R. E. Riggio, J. W. Fantuzzo, S. C. Connelly, and L. A. Dimeff. Reciprocal peer tutoring: A classroom strategy for promoting academic and social integration in undergraduate students. *Journal of Social Behaviour and Personality*, 6(2):387–396, 1991.
- [92] Howard S. Barrows. Problem-based learning in medicine and beyond: A brief overview. *New Directions for Teaching and Learning*, (68):3–12, 1996.
- [93] Jeremy Roschelle and Stephanie D. Teasley. The construction of shared knowledge in collaborative problem solving. In Claire O’Malley, editor, *Computer-Supported Collaborative Learning*, pages 69–97. Springer, Berlin, 1995.

- [94] M. A. Bochicchio and A. Longo. Hands-on remote labs: Collaborative web laboratories as a case study for it engineering classes. *IEEE Transactions on Learning Technologies*, 2(4):320–330, 2009. ISSN 1939-1382.
- [95] J. Machotka, Z. Nedic, A. Nafalski, and O. Gol. Collaboration in the remote laboratory netlab. In *1st WIETE Annual Conference on Engineering and Technology Education*, 2010.
- [96] Wouter R. van Joolingen, Ton de Jong, Ard W. Lazonder, Elwin R. Savelsbergh, and Sarah Manlove. Co-lab: research and development of an online learning environment for collaborative scientific discovery learning. *Computers in Human Behavior*, 21:671–688, 2005.
- [97] Hans Peter Dommel and J. J. Garcia-Luna. Floor control for multimedia conferencing and collaboration. *Multimedia Systems*, 5:23–38, January 1997. ISSN 0942-4962.
- [98] Ntnu java virtual physics laboratory. URL <http://www.phy.ntnu.edu.tw/ntnujava/index.php>.
- [99] Wolfgang Christian, Francisco Esquembre, and Lyle Barbato. Open source physics. *Science*, 334(6059):1077–1078, 2011.
- [100] G. Farias, R. De Keyser, S. Dormido, and F. Esquembre. Developing networked control labs: A matlab and easy java simulations approach. *IEEE Transactions on Industrial Electronics*, 57(10):3266–3275, 2010. ISSN 0278-0046.
- [101] Ruben Heradio, Luis de la Torre, Jose Sanchez, Sebastian Dormido, and Hector Vargas. An architecture for virtual and remote laboratories to support

- distance learning. In *Research in Engineering Education Symposium*, Madrid, Spain, October 2011.
- [102] F. Esquembre, L. de la Torre, R. Heradio, A. Mejias, and M. A. Marquez. Easy java simulations meets moodle, arduino and (walks towards) the ipad. 2013.
- [103] Ch. Salzmann, D. Gillet, and P. Huguenin. Introduction to real-time control using labview with an application to distance learning. *International Journal of Engineering Education*, 16:255–272, 2000.
- [104] S. Dormido, H. Vargas, J. Sanchez, R. Dormido, N. Duro, S. Dormido-Canto, and F. Morilla. Developing and implementing virtual and remote labs for control education: The uned pilot experience. In *17th IFAC World Congress*, 2008.
- [105] Ayse Yayla and Aynur Akar. Web based real time remote laboratory with labview access for analog and digital communication courses. *Journal of Electrical and Electronics Engineering*, 8:671–681, 2008.
- [106] A. Gontean. Labview powered remote lab. In *(SIITME) 2009 15th International Symposium for Design and Technology of Electronics Packages*, 2009.
- [107] M. Stefanovic, V. Cvijetkovic, M. Matijevic, and V. Simic. A labview-based remote laboratory experiments for control engineering education. *Computer Applications in Engineering Education*, 19:538–549, 2011.
- [108] Marco Casini, Andrea Garulli, Antonio Giannitrapani, and Antonio Vicino. A matlab-based remote lab for multi-robot experiments. In *2009 Advances in Control Education*, 2009.

-
- [109] R. Puerto, L.M. Jimenez, and O. Reinoso. Remote control laboratory via internet using matlab and simulink. *Computer Applications in Engineering Education*, 18:694–702, 2009.
- [110] William Rice. *Moodle 2.0 E-Learning Course Development*. Packt Publishing, 2011.
- [111] Hector Vargas. *An Integral Web-Based Environment for Control Engineering Education*. PhD thesis, ETSI Ingenieria Informatica (UNED), 2010.