

Releases of the Lab Owner and Cloud Services -Initial

Irene Lequerica Zorrozua

▶ **To cite this version:**

Irene Lequerica Zorrozua. Releases of the Lab Owner and Cloud Services -Initial. [Research Report] Go-Lab Project. 2014. hal-01201969

HAL Id: hal-01201969
https://telearn.hal.science/hal-01201969

Submitted on 18 Sep 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Go-Lab

Global Online Science Labs for Inquiry Learning at School

Collaborative Project in European Union's Seventh Framework Programme

Grant Agreement no. 317601



Deliverable D4.3

Releases of the Lab Owner and Cloud Services - Initial

Editor	Irene Lequerica Zorrozua (UNED)
Date	30th July, 2014
Dissemination Level	Public
Status	Final



© 2014, Go-Lab consortium

The Go-Lab Consortium

Beneficiary Number	Beneficiary name	Beneficiary short name	Country
1	University Twente	UT	The Netherlands
2	Ellinogermaniki Agogi Scholi Panagea Savva AE	EA	Greece
3	École Polytechnique Fédérale de Lausanne	EPFL	Switzerland
4	EUN Partnership AISBL	EUN	Belgium
5	IMC AG	IMC	Germany
6	Reseau Menon E.E.I.G.	MENON	Belgium
7	Universidad Nacional de Educación a Distancia	UNED	Spain
8	University of Leicester	ULEIC	United Kingdom
9	University of Cyprus	UCY	Cyprus
10	Universität Duisburg-Essen	UDE	Germany
11	Centre for Research and Technology Hellas	CERTH	Greece
12	Universidad de la Iglesia de Deusto	UDEUSTO	Spain
13	Fachhochschule Kärnten – Gemeinnützige Privatstiftung	CUAS	Austria
14	Tartu Ulikool	UTE	Estonia
15	European Organization for Nuclear Research	CERN	Switzerland
16	European Space Agency	ESA	France
17	University of Glamorgan	UoG	United Kingdom
18	Institute of Accelerating Systems and Applications	IASA	Greece
19	Núcleo Interactivo de Astronomia	NUCLIO	Portugal

Contributors

Name	Institution
Sten Govaerts, Christophe Salzmann, Wissam Halimi, Denis Gillet	EPFL
Irene Lequerica Zorrozua, Elio San Cristóbal, Germán Carro	UNED
Danilo Garbi Zutin	CUAS
Pablo Orduña	UDEUSTO
Lars Bollen (internal reviewer)	UT
Alexandros Trichos (internal reviewer)	CERTH

Legal Notices

The information in this document is subject to change without notice.

The Members of the Go-Lab Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the Go-Lab Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

The information and views set out in this deliverable are those of the author(s) and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

Executive Summary

This deliverable is a companion document to the initial releases of the Lab Owner and Cloud services (Smart Device and Smart Gateway) that have been delivered at M21. These first releases implement the revised version of the Smart Device and Smart Gateway specifications as defined in the revised version of D4.1 (M21).The deliverable describes the software components available on GitHub together with their documentation and provides all related links. Next to that, this deliverable provides some additional background information and explains to choices made.

In this initial release of the lab-owner services dedicated to new remote labs, we consider 4 alternative Lab Server implementations. A dedicated Lab Server is considered as a plug solution combining hardware and software layers linking a physical lab to the Internet. These four solutions target implementation on either desktop or embedded computers. The former corresponds to typically lab-owner implementation in academic institutions and scientific organizations. The latter corresponds to an emerging scheme enabling large scale deployment of remote labs at low cost. Two software alternatives are proposed to expose the lab server to various clients following the Smart Device specifications (D4.1). The first alternative is based on LabVIEW and corresponds to the mainstream solution. The second one is based on JavaScript and enables implementation on compact embedded computers. Examples of physical labs plugged using the proposed solutions are provided as templates to enable new remote lab implementations by lab owners.

As defined in D4.1, the Smart Device specification is meant for implementing new remote labs, while the Smart Gateway on the other hand as a cloud service is meant for integrating legacy labs shared through existing third parties remote lab management systems (RLMS) within the Go-Lab infrastructure. This deliverable discusses the implementation details of the Smart Gateway and its plug-in architecture. The initial release of the Smart Gateway offers five plug-ins to exploit legacy labs from the following RLMS: WebLab-Deusto, iLab, UNR-FCEIA, PhET, and ViSH.

The final specifications and releases of the lab-owner and cloud services will be documented in the deliverables D4.5 and D4.7.

Table of Contents

The Go-Lab Consortium **2**

Executive Summary **4**

Table of Contents **5**

List of Figures **7**

List of Tables **9**

1 Introduction **10**

1.1 Lab-owner services released as smart device plug solutions **11**

1.2 Cloud Services released as a smart gateway with dedicated plug-ins **12**

1.3 Software repository **13**

2 Lab owner (plug) services (smart devices) **14**

2.1 Desktop computer with LabVIEW: Desktop plug **14**

2.1.1 Short description 14

2.1.2 Services and functionalities implemented 14

2.1.3 Guideline for developers 15

2.2 myRIO embedded computer with Javascript: myRIO Plug **15**

2.2.1 Short description 15

2.2.2 Services and functionalities implemented 16

2.2.3 Guideline for developers 16

2.3 BeagleBone Black embedded computer with Javascript: BB-B Plug **16**

2.3.1 Short description 17

2.3.2 Services and functionalities implemented 17

2.3.3 Guideline for developers 18

2.4 Raspberry Pi embedded computer with Javascript: R-Pi Plug **18**

2.4.1 Short description 18

2.4.2 Services and functionalities implemented 20

2.4.3 Guideline for developers 20

3 Prototypes of Cloud Services **22**

3.1 Introduction **22**

3.2 Smart Gateway Architecture **22**

3.3 The Smart Gateway software 24
3.3.1 Support for standards 25
3.3.2 Support for remote laboratories. The plug-in system. 26
3.3.3 Demo and Software Repository 30
3.3.4 Summary of the benefits for integrated remote laboratories 31
3.4 Prototypes 32
3.4.1 WebLab-Deusto 32
3.4.2 iLab Shared Architecture (ISA) 34
3.4.3 PhET 37
3.4.4 ViSH 38
3.4.5 UNR-FCEIA 39
4 Conclusion and future work 41
5 Apendix 42
5.1 Appendix A: Lab owner survey results 42
5.2 Appendix B: Brief history of gateway4labs 44
5.3 Appendix C: Smart Gateway plug-ins size 46
5.4 Appendix D: Smart Gateway Support for advanced features on top of OpenSocial . 48
5.5 Appendix F: Lab Owner's Survey 52
5.6 Appendix G: Ongoing Smart Gateway Work 52
5.7 Appendix H: Management features of the Smart Gateway 53
6 References 57

List of Figures

Figure 1. The remote lab architecture (taken from D4.1) 10
Figure 2. Plug Solutions for Lab Owners 11
Figure 3. The three simple client apps (current position, video, set position) 14
Figure 4. The simple client apps (current speed, set speed) 16
Figure 5. Client application is succesfully connected to the websocket application 17
Figure 6. Turning on the PWM control (servo motor control) and changing the position of the motor's shaft 17
Figure 7. Turning off the PWM: disabling the control 17
Figure 8. Robotic Arm deployment architecture 19
Figure 9. The simple client app integrated in Graasp (video and actuators controls) 20
Figure 10: Cloud Services Architecture 23
Figure 11: Gateway4Labs overall architecture 24
Figure 12: Configuration process in Python plug-ins 28
Figure 13: HTTP plug-in configuration stored in the plug-in side 30
Figure 14: Federation algorithm in WebLab-Deusto 33
Figure 15: Two of the four balls in the Archimedes laboratory 34
Figure 16: Gateway4labs plug-in for ISA workflow 35
Figure 17: Radioactivity Lab embedded in an ILS 37
Figure 18: A PhET simulation included in the ILS platform 38
Figure 19: A ViSH resource included in the ILS platform 39
Figure 20: A ViSH resource included in the ILS platform 40
Figure 21: Development of the labmanager 45
Figure 22: Development of the WebLab-Deusto plug-in 46
Figure 23: Development of the iLab Shared Architecture plug-in 46
Figure 24: Development of the PhET plug-in 46
Figure 25: Percent of each of the plug-ins and the Labmanager 47
Figure 26: Comparing a single laboratory with the rest of the Labmanager code 48
Figure 27: Registering the school in the Labmanager 49
Figure 28: Log in as a school user 49
Figure 29: School users 50
Figure 30: Laboratories available for being requested 50
Figure 31: Laboratories registered for the school 50
Figure 32: Register parent spaces 51
Figure 33: List of registered spaces 51
Figure 34: List of permissions on registered spaces 51
Figure 35: List of plug-in instances 53
Figure 36: Example of registration of a RLMS 54
Figure 37: List of laboratories provided by one of the plug-ins 54

Figure 38: Making laboratories public so they can be accessed by anyone 55
Figure 39: List of publicly available laboratories 55
Figure 40: List of widgets of a particular laboratory, including the necessary link and a preview of the widget 56

List of Tables

Table 1. Description of the launchLabClient service 36
Table 2: Results of Lab Owner's Survey 44
Table 3: Lines of code of the plug-ins and shared components (Labmanager) 47
Table 4: Overview of survey's answers 52

1 Introduction

Enabling the access to and the interaction with a Physical Lab at distance to turn it into a remote lab relies on a client-server architecture as described in Figure 1. This deliverable discusses alternative ways of implementing the Lab Server to easily plug a Physical Lab online and exposing it to various clients as a Smart Device through standardized Internet Services (as defined in D4.1).

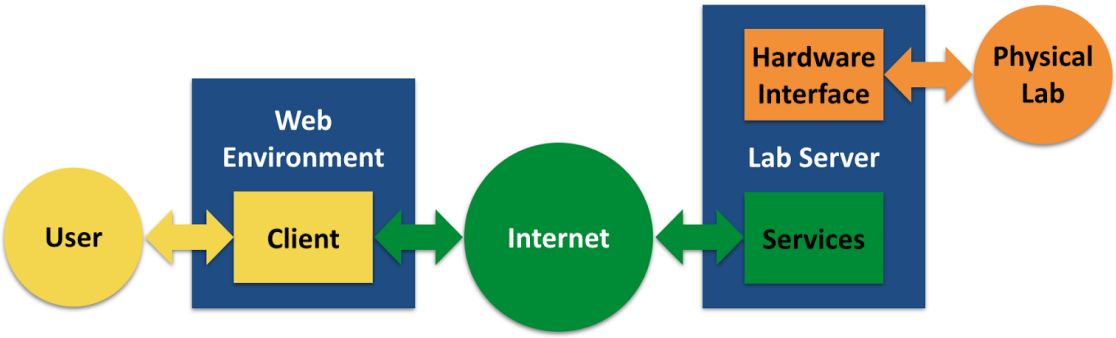


Figure 1. The remote lab architecture (taken from D4.1)

Lab servers always combine hardware and software components. To ease the selection of such interdependent components for lab owners, we propose in this deliverable four Smart Device plug solutions for new remote labs and one Smart Gateway with five plug-ins for legacy remote lab management systems (RLMS) in line with the current lab owner's practices and new best practices promoted by Go-Lab.

Because of the large variety of Physical Labs, the implementation of the Smart Device plug solutions and the Smart Gateway plug-ins always combines standardized (green part in Figure 1) and ad hoc (orange part in Figure 1) components. So, no API can easily be provided. At this stage, we provide implementation examples on simple physical labs that can be used as templates for implementation with other labs by lab owners.

The Lab Server provides the following functionalities:

- The Hardware Interface (orange box): This interface provides access to the real world via the Hardware interface. The hardware interface ensures that the measurements and actuations are available at the Lab Server level. Depending on the hardware configuration, these accesses can be done through a DAQ board, a built-in general purpose input-output (GPIO) interface or through a connection bus (USB, thunderbolt, serial, etc.). The hardware access is really ad-hoc and depends on the connected Physical Lab.
- The Smart Devices Services (green box): These services are defined in D4.1 and are mainly accessible through WebSockets. A WebSocket server needs to be in place to answer the client applications requests. The same server or an extra web server can also serve static information such as metadata.
- The Internal Management (blue box): The internal management is the Lab Server orchestrator. Its main task is to manage the connection between the client request and measurements/actuations made on the Physical Lab. It also decodes and validates client requests, accesses the appropriate sensors/actuators, formats the replies, manages concurrent access, etc. A lot of the Lab Owner know-how will be translated into the Lab Server's internal management.

Selected software packages

We focused on two main software packages to implement the software component of the Lab Server: LabVIEW and Javascript.

LabVIEW is a language dedicated to data acquisition and control[1] but not only, it supports a wide range of high-level libraries including network related ones. Over the years, LabVIEW became the de-facto language/environment to access Physical Labs through hardware interfaces. LabVIEW comes with an extended set of drivers to interface hardware. This means that, for example, a sensor measurement can be displayed in oscilloscope window in a matter of minutes. LabVIEW is cross platform, which means the same source code can be run on OSX, Windows and Linux (although some features are platform specific). Once completed, the LabVIEW code can be saved as an executable and deployed so that the complete LabVIEW environment does not need to be installed.

JavaScript is one of the main Web programming languages. Since the advent of Node.js[2], developers can develop the complete client-server stack only in JavaScript. This can enable faster development and less investment in learning new technologies. The Node.js community has grown very fast in the recent years and the available software packages for Node.js are surpassing Java and PHP[3]. For our purpose, the interface to the Internet via WebSockets is natively supported by JavaScript. Node.js runs on both desktop computers and different embedded computers due to its efficiency and low memory footprint. Numerous libraries exist to support communication via GPIO, USB and other IO ports on desktop and embedded computers.

1.1 Lab-owner services released as Smart Device plug solutions

In this initial release of the lab-owner services dedicated to new remote labs, we consider 4 alternative Lab Server implementations. As such, the Lab Server can be considered as a plug solution combining hardware and software layers linking a physical lab to the Internet. These four solutions are described below (Figure 2) and detailed in Section 2.

<table>
<tr><th></th><th>Software</th><th>Hardware</th><th>Physical Lab Interfaces</th><th>Examples as Templates</th></tr>
<tr><td>Desktop Computer</td><td rowspan="2">LabVIEW</td><td>Mac or PC</td><td>DAQ Boards</td><td>RED Servo</td></tr>
<tr><td rowspan="3">Embedded Computer</td><td>myRIO</td><td rowspan="2">Built-in GPIO</td><td>BLACK Servo</td></tr>
<tr><td rowspan="2">Javascript</td><td>BeagleBone</td><td>MINI Servo</td></tr>
<tr><td>Raspberry PI</td><td>Arduino</td><td>Robot Arm</td></tr>
</table>

Figure 2. Plug Solutions for Lab Owners

[1] http://www.ni.com/labview/
[2] http://nodejs.org/
[3] http://www.modulecounts.com/

The analysis of the contributions to the REV Conference in 2014[4] related to remote lab implementation (19 papers) shows that 85% of Lab owners use desktop computers as Lab Server hardware, while 15% are implementing it on embedded computers. 53% of the lab owners rely on LabVIEW as Lab Server software, while the rest rely on various alternative software packages. Considering that embedded computers are gaining popularity due to their increasing performance and decreasing price, the four proposed plug solutions are:

- **The Desktop Plug**, i.e. a Mac or a PC with software implemented in LabVIEW. The rationale here is to support the mainstream lab owner solution.
- **The myRIO Plug**, i.e. a myRIO embedded computer with software implemented in LabVIEW[5]. The rationale here is to enable the port of a mainstream lab owner solution (Desktop Plug) on a modern all-in-one embedded computer without any software adaptation.
- **The BB-B Plug**, i.e. a BeagleBone Black embedded computer with software implemented in Javascript[6]. The rationale here is to promote one of the leading embedded computers in a compact form factor the equivalent of a desktop computer combined with a set of input and output interfaces to connect with the sensors and actuators of a physical lab with a low cost.
- **The R-Pi Plug**, i.e. a Raspberry Pi embedded computer with software implemented in Javascript[7]. The rationale here is to offer a solution compatible with a low-cost and well-spread embedded computer supported by a large developer community. It should however be often combined with additional hardware such as Arduino[8] to enable complex data acquisition and control, due to the limited number of GPIO pins that the Raspberry Pi board provides.

1.2 Cloud Services released as a Smart Gateway with dedicated plug-ins

In this initial release of the cloud services dedicated to existing remote labs, we propose one Smart Gateway with five plug-ins for legacy remote lab management systems (RLMS) enabling integration in the Go-lab infrastructure. The Smart Gateway and the plug-ins are detailed in section **Error! Reference source not found.**. The RLMS currently supported through plug-ins are described below.

- **WebLab-Deusto**: The RLMS of the University of Deusto used by academic engineering institutions and secondary schools from different countries worldwide, as well as in multiple research projects[9].
- **iLab**: A shared architecture created by MIT and developed in collaboration with partners from the Global Online Laboratory Consortium (GOLC) with implementations in developed and developing countries[10].
- **UNR-FCEIA**. LabRem-FCEIA is a remote lab management system of the University of Rosario in Argentina[11].

[4] http://www.rev-conference.org/REV2014/
[5] http://www.ni.com/myrio/
[6] http://www.ti.com/tool/beaglebk
[7] http://www.raspberrypi.org/product/model-b/
[8] http://store.arduino.cc/product/A000066
[9] http://www.weblab.deusto.es
[10] http://ilab.mit.edu/wiki
[11] http://labremf4a.fceia.unr.edu.ar

- **PhET**[12]. A project at the University of Colorado to deliver fun, interactive, research-based simulations of physical phenomena. The corresponding plug-in generating links to the PhET simulation.
- **ViSH**: Virtual Science Hub[13]. ViSH indexes open resources such as simulations and freely available remote laboratories embedded in Excursions similar to Go-Lab Inquiry Learning Spaces. The corresponding plug-in generating links to the simulation.

1.3 Software repository

The templates for the Smart Device plug solutions and the associated examples are provided on GitHub[14]. These solutions are updated continuously to follow the evolution of the Smart Device specifications and to ease the exploitation of the examples as templates for further deployment of remote labs by lab owners. For each solution, a description wiki, examples, source code, instructions and guidelines are provided.

Smart Device plug solutions:

- Desktop plug: https://github.com/go-lab/smart-device/tree/master/Desktop
- myRIO plug: https://github.com/go-lab/smart-device/tree/master/myRIO
- BB-B plug: https://github.com/go-lab/smart-device/tree/master/BeagleBoneBlack
- R-Pi plug: https://github.com/go-lab/smart-device/tree/master/RaspberryPi

The Smart Gateway component and its plug-ins are also available on GitHub.

Component

- LabManager: https://github.com/gateway4labs/labmanager

Plug-ins

- WebLab-Deusto: https://github.com/gateway4labs/rlms_weblabdeusto
- iLab Shared Architecture: https://github.com/gateway4labs/rlms_ilabs/
- UNR-FCEIA: https://github.com/gateway4labs/rlms_unr
- PhET: https://github.com/gateway4labs/rlms_phet
- ViSH: https://github.com/gateway4labs/rlms_vish

The software provided by Go-Lab and listed above is shared under the Open Source MIT license and the external libraries are shared with their respective licenses.

https://github.com/go-lab/smart-device/blob/master/LICENSE.md

[12] http://phet.colorado.edu
[13] http://vishub.org
[14] https://github.com/go-lab/smart-device

2 Lab owner (plug) services (Smart Devices)

This chapter details the plug solutions listed in section 1.1. Each plug solution is implemented as an example to remotely operate a simple physical lab. The idea is to illustrate how the plug solutions can be deployed. The associated code is intended be used as a template with parts to be adapted by lab owners to interface their own labs. The example labs have been chosen to enable a simple implementation and to illustrate in a simple way how to implement the Smart Device specifications. They are not all dedicated to be exploited as such for educational purposes.

2.1 Desktop computer with LabVIEW: Desktop plug

Source code: https://github.com/go-lab/smart-device/tree/master/Desktop

Documentation Wiki: https://github.com/go-lab/smart-device/wiki/Desktop-wiki

Hardware: desktop computer (Mac, PC) with DAQ card (NI PCIe 6259)

Software: LabVIEW (2013)

2.1.1 Short description

The proposed template illustrates the angular position control of an electrical drive. It reads the motor axle position and control the motor voltage to reach a reference position. In addition, it shows how to interface an USB Video Class (UVC) webcam.

Hardware: Standard Mac or PC with dedicated PCIe slot (or via thunderbolt extension). The DAQ hardware is a PCIe 6259 DAQ card. This card is available for both Mac and PC[15].

Software: The software is the Full LabVIEW Development package (2013). The provided hardware drivers are obviously platform specific.

IO: One quadrature encoder to read the encoder mounted on the motor axle. This encoder position decoding is handled in hardware on the PCIe 6259. One output (DA) to control the motor voltage. The DAQ card carries many more I/O.

Demo: https://github.com/go-lab/smart-device/wiki/Desktop-wiki

The provided client apps are depicted in Figure 3.

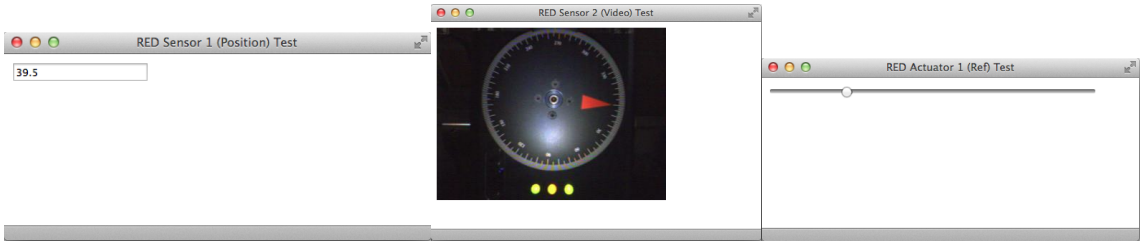


Figure 3. The three simple client apps (current position, video, set position)

2.1.2 Services and functionalities implemented

The proposed LabVIEW template provides:

- an example of sensor and actuators access (including video access)
- a combined web and WebSocket server that handles the Smart Device services
- a mechanism to link the sensor/actuator to their respective services
- an example of internal management (request validation, controller, etc)

[15] http://sine.ni.com/nips/cds/view/p/lang/en/nid/201814

- concurrent users access with race policy for the controller mode (refer to D 4.1 to get more information about concurrency)

The proposed LabVIEW template interfaces an electrical drive (RED Servo) where the user can set and read the angular position of the drive axle, the client application can:

- access the axle position (sensor 1)
- access a video image of the disk connected to the drive axle (sensor 2)
- set the axle position (actuator 1)

In addition the LabVIEW template provides:

- simple web apps (HTML client applications) for the above sensors/actuators
- internal controller to track the desired axle position

2.1.3 Guideline for developers

To adapt the template to other physical lab one needs to:

- provide a virtual instruments (VI) that read new sensors and place the values into LabVIEW queue
- provide a virtual instruments (VI) that write actuator value taken from a LabVIEW queue
- Extend the existing VIs that validates user request
- Enable/disable the internal controller
- Specify the linkage for the various services <-> sensors/actuators queues
- Update/extend the existing metadata

2.2 myRIO embedded computer with Javascript: myRIO Plug

Source code: https://github.com/go-lab/smart-device/tree/master/myRIO

Documentation Wiki: https://github.com/go-lab/smart-device/wiki/myRIO-wiki

Hardware: NI myRIO 1900

Software: LabVIEW (2013)

2.2.1 Short description

The proposed template illustrates the angular speed control of an electrical drive (BLACK Servo). It read the motor axle speed and control the motor voltage to reach a reference speed. The myRIO and the desktop templates are very similar. At this stage it remains as 2 differents templates due to the myRIO specific hardware access. Eventually the two templates will be merged.

Hardware: The embedded computer myRIO 1900

http://sine.ni.com/nips/cds/view/p/lang/en/nid/211694

Contains: Arm processor + FPGA + Customizable I/O

Software: The software is the Full LabVIEW Development package (2013) + myRIO extention (come with the myRIO). Note: The LabVIEW code is cross-compiled to the myRIO, an external PC/Mac is required during the development stage.

IO: One AD to read the motor speed. One output (DA) to control the motor voltage. The DAQ card carries many more I/O.

Demo: https://github.com/go-lab/smart-device/wiki/myRIO-wiki

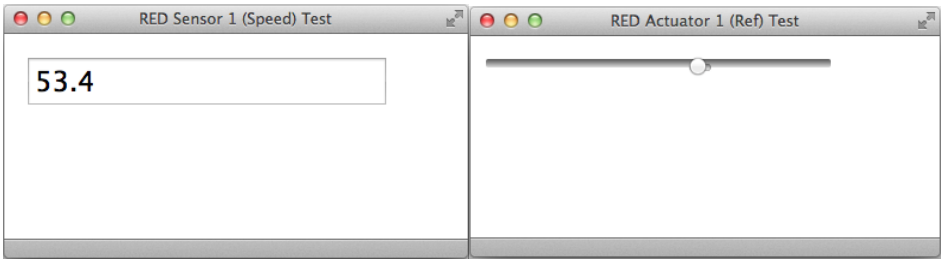


Figure 4. The simple client apps (current speed, set speed)

2.2.2 Services and functionalities implemented

The proposed LabVIEW template provides:

- an example of sensor and actuators access
- a combined web and WebSocket server that handles the Smart Device services
- a mechanism to link the sensor/actuator to their respective services
- an example of internal management (request validation, controller, etc)
- concurrent users access with race policy for the controller mode (refer to D 4.1 to get more information about concurrency)

The proposed LabVIEW template interfaces an electrical drive where the user can set and read the angular position of the drive axle, the client application can:

- access the axle speed (sensor 1)
- set the axle speed (actuator 1)

In addition the LabVIEW template provides:

- simple web apps (HTML client applications) for the above sensors/actuators
- internal controller to track the desired axle position

2.2.3 Guideline for developers

To adapt the template to other physical lab one needs to:

- provide a virtual instruments (VI) that read new sensors and place the values into LabVIEW queue
- provide a virtual instruments (VI) that write actuator value taken from a LabVIEW queue
- Extend the existing VIs that validates user request
- Enable/disable the internal controller
- Specify the linkage for the various services <-> sensors/actuators queues
- Update/extend the existing metadata

2.3 BeagleBone Black embedded computer with Javascript: BB-B Plug

Source code:

https://github.com/go-lab/smart-device/tree/master/BeagleBoneBlack/servo-beaglebone-black

Documentation Wiki:

- Main documentation: https://github.com/go-lab/smart-device/tree/master/BeagleBoneBlack/servo-beaglebone-black#servo-beaglebone-black
- Installation documentation: https://github.com/go-lab/smart-device/wiki/BeagleBone-Black-for-Servo-Motor

Hardware: BeagleBone Black embedded computer

Software: WebSockets with Node.js

2.3.1 Short description

The proposed template illustrates the control of a servo motor (MINI Servo). Thanks to the web client, the user knows the status of the lab and the position of the motor's shaft. The user can change the position of the motor's arm by moving a slider to the left or right. The motor of the lab provides the controlling endpoint.

Hardware: The BeagleBone Black embedded computer.

Software: The complete software application is built according to the Node.Js framework and is using the WebSocket library to implementing the WebSocket technology. Needed drivers for installing the BeagleBone Black (BBB) on different operating systems can be found here: https://learn.adafruit.com/ssh-to-beaglebone-black-over-usb/overview

IO: Header pins of the BBB configured for PWM control

Demo: https://github.com/go-lab/smart-device/wiki/BeagleBone-wiki

URL: http://graasp.epfl.ch/#item=widget_8388

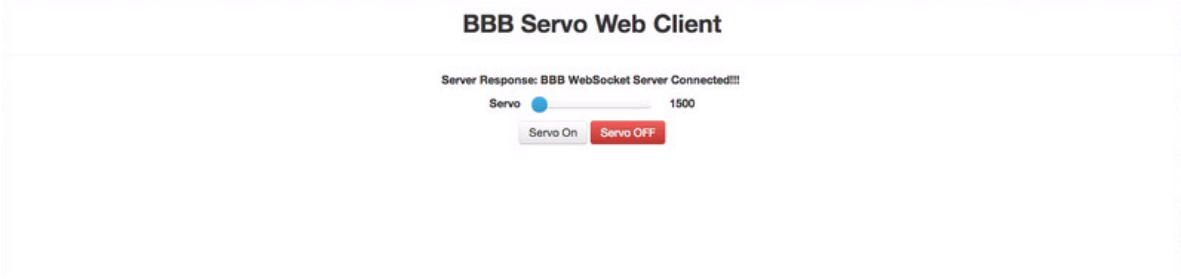


Figure 5. Client application is successfully connected to the WebSocket application



Figure 6. Turning on the PWM control (servo motor control) and changing the position of the motor's shaft



Figure 7. Turning off the PWM: disabling the control

2.3.2 Services and functionalities implemented

- Metadata Service
- Sensor Service
- Actuator Service

2.3.3 Guideline for developers

- How to use the example as a template: https://github.com/go-lab/smart-device/wiki/How-to-use-Servo-with-BBB-example-as-template
- Installation guide on GitHub: https://github.com/go-lab/smart-device/wiki/BeagleBone-Black-for-Servo-Motor

2.4 Raspberry Pi embedded computer with Javascript: R-Pi Plug

Source code:

https://github.com/go-lab/smart-device/tree/master/RaspberryPi/robotic_arm-raspberry-pi

Documentation Wiki:

- Main documentation: https://github.com/go-lab/smart-device/wiki/Robotic-Arm-Laboratory
- Installation documentation: https://github.com/go-lab/smart-device/wiki/How-to-use-Node.js-and-Socket.io-labratories-with-Raspberry-Pi-and-Arduino

Hardware: Raspberry Pi B[16] embedded computer, Arduino UNO[17] microcontroller board

Software: Raspbian O.S., node.js, socket.io, serialport, Apache, bootstrap

Demo URL: http://graasp.epfl.ch/#item=space_15624

2.4.1 Short description

The proposed template illustrates the use of the R-Pi plug to control a robotic arm in order to simulate usual human arm movements like: move right/left, move backwards/onwards, grabbing an object, etc.

Hardware: Raspberry Pi B embedded computer, Arduino UNO microcontroller board.

Software:

- Raspbian Operating System[18]
- Apache web server[19]
- Bootstrap[20]: it is a very popular HTML, CSS, and JS framework for developing responsive projects on the web. It has been selected in order to build the user interface
- Node.js[21]: it is a JavaScript framework that works as a web server, serving the html files to the user through http. Its event-driven, non-blocking I/O model makes it fit for real-time applications suitable for communicating with the remote laboratory It makes use of Socket.io.
- Socket.io[22]: it enables real-time bidirectional event-based communication between the client applicacion and the web server thought WebSockets. It is used by Node.js in order to make use of WebSockets technology.
- Serialport library: it allows the connection between the Raspberry Pi and the Arduino boards through an USB port.

[16] http://www.raspberrypi.org/
[17] http://www.arduino.cc/
[18] http://www.raspbian.org/
[19] http://www.apache.org
[20] http://getbootstrap.com/
[21] http://nodejs.org/
[22] http://socket.io/

The architecture must allow a final user, having access to Internet through a navigator, to control the Robotic Arm. The following figure represents the final implemented hardware and software architecture:

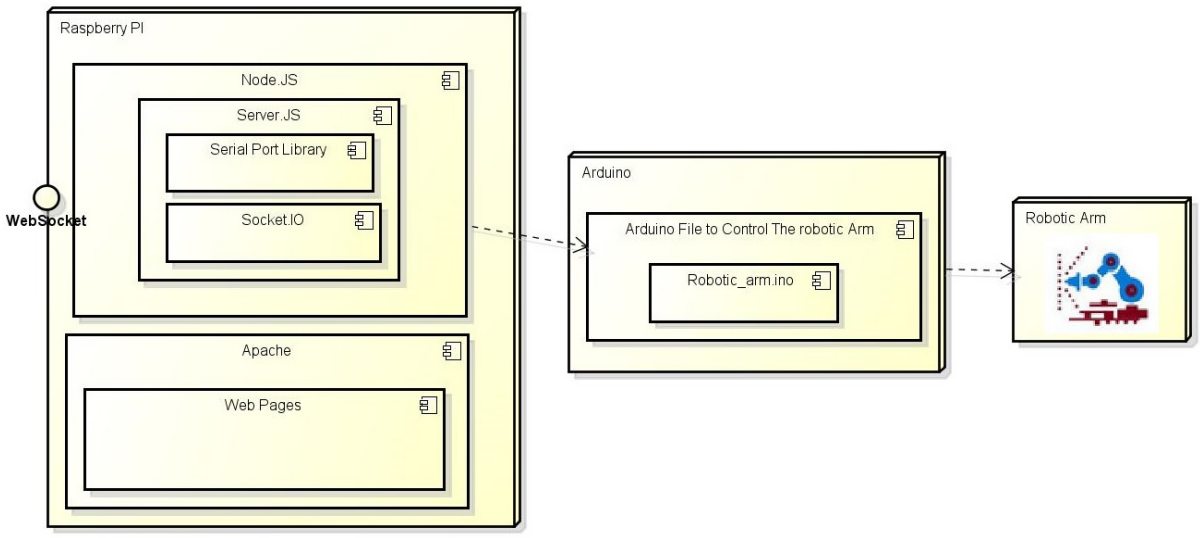


Figure 8. Robotic Arm deployment architecture

IO: An Arduino UNO board is used to connect the Raspberry PI to the Physical Lab. It is connected to the Robotic Arm actuators as follow:

- wrist: ports I1 and I2
- elbow: ports I3 and I4
- shoulder: ports I5 and I6
- base: ports I7 and I8
- clamp: ports I9 and I10
- led: port I12

Demo: http://graasp.epfl.ch/#item=space_15624

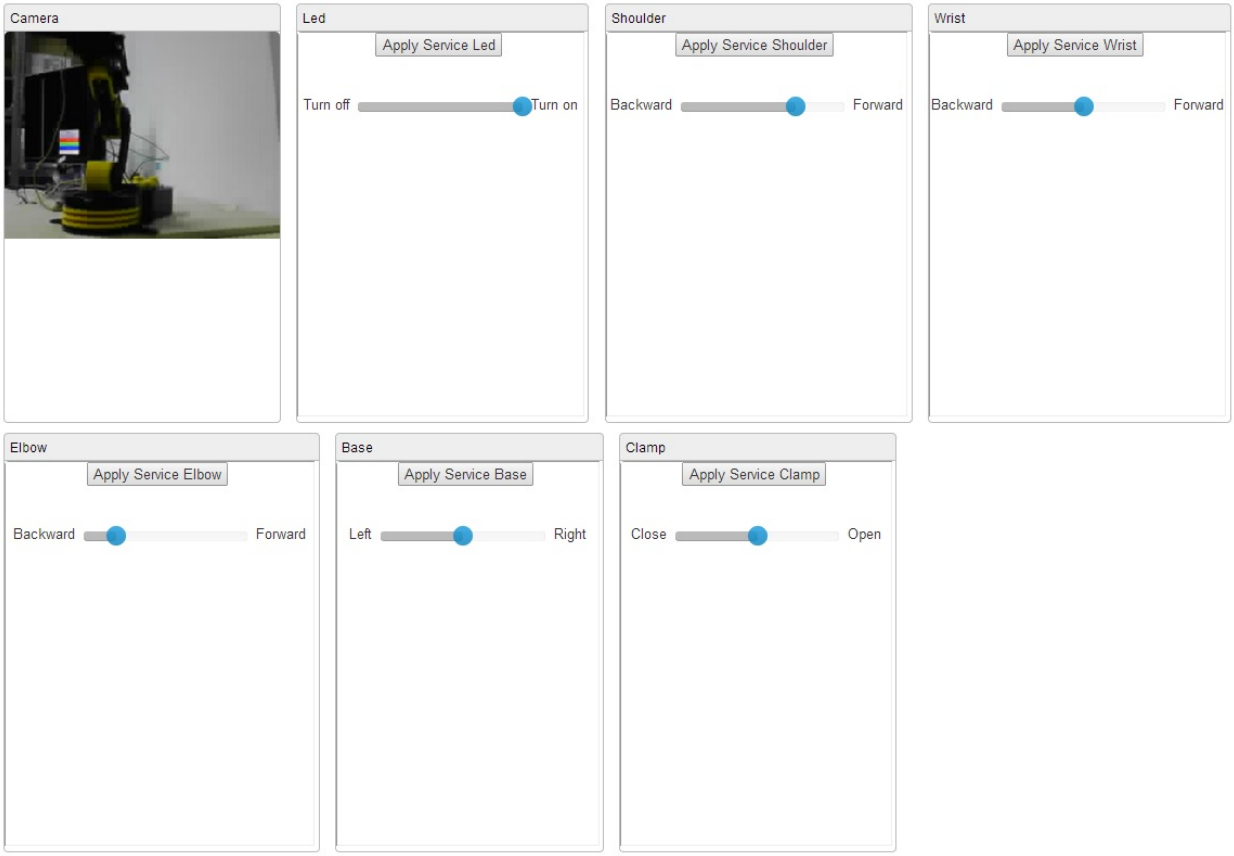


Figure 9. The simple client app integrated in Graasp (video and actuators controls)

2.4.2 Services and functionalities implemented

Sensors:

- A webcam service.

Actuators:

- A service for each of the motors involved in the Robotic Arm motion: clamp, wrist, elbow, shoulder and base.
- A service to switch on and off a simple blue LED.

Each of the actuators is independently controlled. The movements allowed are: (1) from left to right and vice versa for the base, (2) from backwards onwards and vice versa for the wrist, elbow and shoulder and (3) open and close the clamp. DC motors have an endless rotation either in one direction or the opposite, meaning that this movement must be controlled in order not to break the physical instrumentation. For the base, shoulder, elbow, wrist and clamp motors 5 decimal values can be selected: 20, 40, 60, 80 and 100 both, in one direction and in the opposite. A 150 milliseconds time slot interval is also defined. Any other values, instead of 20, 40, 60, 80 and 100, could have been chosen since they just represent for how long the DC motor is going to rotate. This is implemented as 1, 2 3, 4 or 5 times the time slot interval, which is translated into a value from 1 to 12 before sending it to the Arduino board. Each or the numbers represents a different movement.

2.4.3 Guideline for developers

The elements that could be reused are:

- The programming framework: apache web server, node.js, server.js, socket.io, bootstrap and serialport libraries. The serialport library allows connecting the Raspberry Pi with the

Arduino board through USB, but it should be possible to use a wireless communication protocol or even the Ethernet protocol by making use of the respective libraries.

- The UI html files. Though customized UIs, using JavaScript technologies or building an Opensocial application to be fully integrated in Go-Lab platform, could be implemented as far as they comply with the specifications.
- The WebSocket used for receiving the data from the client and send them to the Arduino board through the serial port.

Reusing the software could imply some programming modifications, which could be the case of deciding not to use the Arduino board. In this case, the Raspberry Pi would be the responsible of interacting with a limited version of the robotic arm, due to the insufficient number of GPIO a Raspberry Pi board has. In the case of the “Robotic_arm.io” file installed in the Arduino board is very likely to be modified since it is responsible of controlling the hardware through a very low-level programming. The program must comply with the end user requirements and therefore must be able to send and receive the data from and to the physical instrumentation as requested.

3 Prototypes of Cloud Services

As defined in D4.1, by Cloud Services we understand a set of services designed to extend the functionalities of the Go-Lab infrastructure to lab owners of legacy lab systems. In the scope of this document, legacy lab systems are all online lab platforms not designed according to the Smart Device specifications. This section will detail the implementation of the Smart Gateway - the core interoperability component of the Cloud Services- in different contexts and online labs.

3.1 Introduction

In D4.1, the requirements and the design of the Smart Gateway are presented. The Smart Gateway is presented as an approach to integrate legacy laboratories composed by two main components:

1. gateway4labs, as a management tool for integrating the reservation process of the legacy laboratories, which might require wrapping authentication, authorization, scheduling and providing additional services (such as metadata or Go-Lab level booking).
2. protocol translator, as an optional tool for translating the existing communications to Smart Device compliant services.

As already discussed in D4.1, there is a tradeoff between the development effort for both components and the features provided.

This chapter is organized as follows. Section 3.2 provides an overview of the Smart Gateway architecture, its components and the different options to integrate a legacy lab into Go-Lab (details have been provided in D4.1). Section 3.3 focuses on the Smart Gateway software. It shortly describes its functionalities and how the support for heterogeneous online laboratories was implemented. Section 3.4 focuses on the developed prototypes for some RLMSs and other legacy lab systems. Furthermore, appendix 5.5 and 5.6 present the results of a survey carried out with lab owners that aimed to collect more detailed information about the community and their lab systems and an overview about the future work regarding the integration of new online lab systems.

3.2 Smart Gateway Architecture

The main purpose of the Smart Gateway is to allow for the integration of legacy lab systems with the Go-Lab ILS Platform by making them fully or partially compliant with the Smart Device specifications. The level of compatibility depends on the implementation strategy adopted (see D4.1 for a description of the different integration levels). Figure 10 below shows the Smart Gateway architecture in detail.

The gateway4labs is the core of the architecture. It provides a core component (a web application called LabManager), and different approaches for including external resources. It provides metadata services and exports the legacy lab clients as OpenSocial gadgets so that these gadgets can be embedded into inquiry learning spaces. The Smart Gateway supports different legacy lab systems via different plug-ins. In this architecture the plug-ins are responsible to bridge the communication between the LabManager and the legacy lab system. The functionalities that a plug-in should implement depend on the level of integration that is desired to achieve (see D4.1 for details). For online labs managed by a Remote Laboratory Management System (RLMS) (an RLMS is a system that provides a management layer for multiple remote laboratories, optionally including authentication, authorization, scheduling or user tracking), a single plug-in is necessary to integrate all labs managed by the same RLMS.

Plug-ins for some well-known RLMSs like WebLab-Deusto and iLabs Shared Architecture are provided. The detailed implementation of each one will be discussed in the following sections.

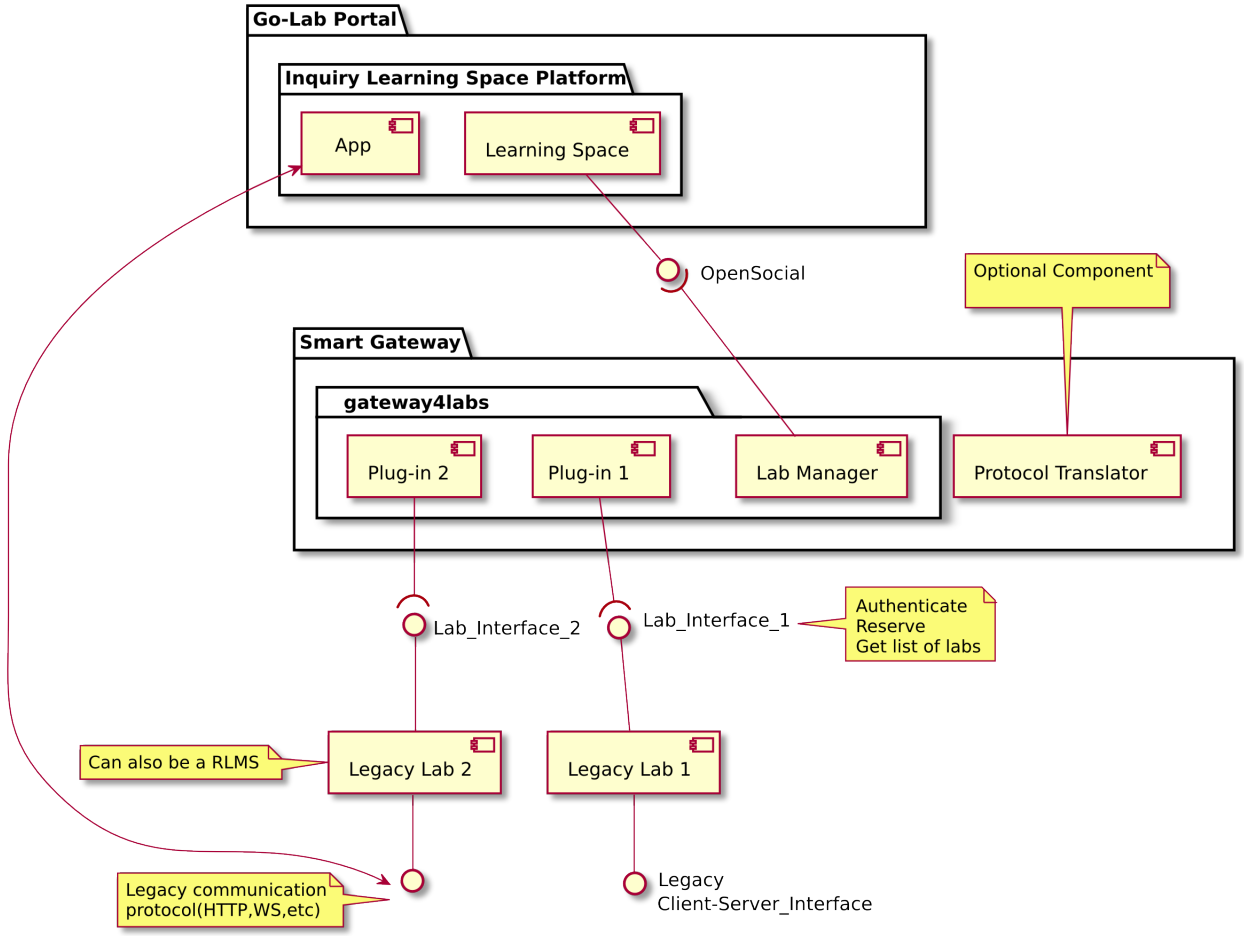


Figure 10: Cloud Services Architecture

Lab owners can choose from four different options to plug their system into the Smart Gateway that will affect and the development of the plug-in and the features supported. Therefore there is a tradeoff between the supported features and the implementation efforts. These options were described in detail in D4.1, but we will briefly recapitulate it here:

1. **iFrame in Smart Gateway:** This can be implemented if authentication is not required by the legacy system. This option consists in providing the legacy client as an OpenSocial application, so technically it is an enhanced iFrame. Additionally, by including a lab, the Smart Gateway administrator has the possibility to author some metadata content for the lab that the Smart Gateway will use to provide the metadata services (part of the Smart Device specification).
2. **A simple version of the plug-in:** If the legacy lab requires authentication the plug-in can be implemented in such a way that it will log into the legacy system with a fixed user account. In this approach the legacy lab administrator will not be able to uniquely identify the user, so some features like the possibility of tracking used actions in the context of an experiment would be unavailable.
3. **A full version of the plug-in:** In this case all reservation features provided by the legacy system are bridged by the plug-in. Users can be uniquely identified. Parts of these features also depend on implementations at the legacy lab side. It should return a URL that will be loaded by the client.

4. **A full version of the plug-in plus a protocol translator:** Additionally to implementing a full version of the plug-in this option requires all messages exchanged between client and server to be translated according to the Smart Device specification. It requires potentially a large implementation effort and an individual solution for each legacy system.

3.3 The Smart Gateway software

The Smart Gateway is consists of gateway4labs (which manages the authentication, scheduling mechanisms) and the protocol translator. The following diagram describes the overview of the gateway4labs software components:

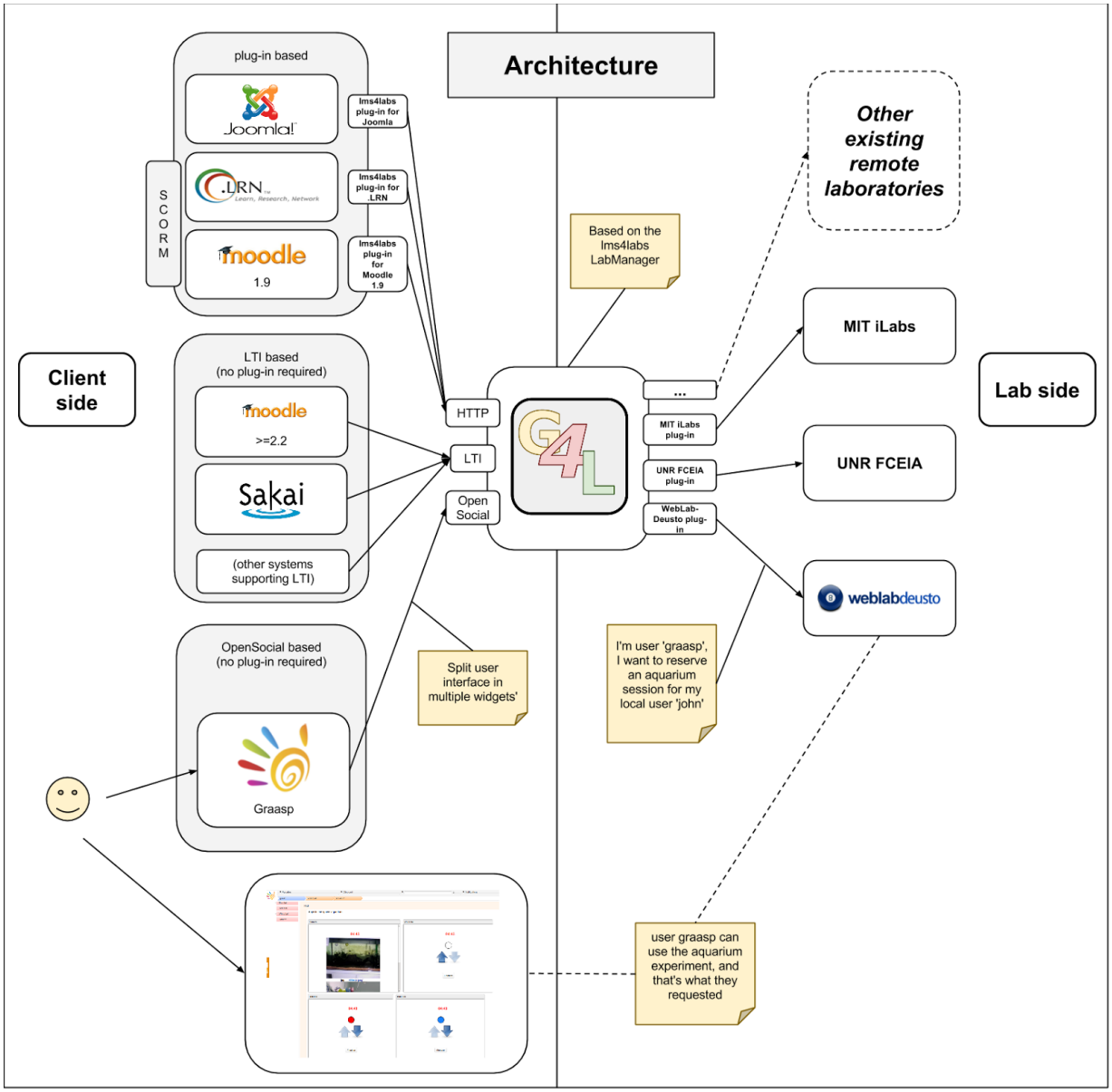


Figure 11: Gateway4Labs overall architecture

One can identify two main layers on Figure 11: the left side is client side and the right side is lab side. On the left side, different learning tools can be integrated through the support of existing standards by the core component of gateway4labs, which is called Labmanager. On the right side, different remote laboratories can be integrated through a set of plug-ins. In the middle, the

Labmanager performs all the management operations and conversions between the different actors of the left side and the right side.

In this way, we can split the functionality of the gateway4labs in three parts:

1. The support for standards, and OpenSocial in particular since it is the one being supported by the ILS platform. This is detailed in section 3.3.1.
2. The support for remote laboratories, through the plug-in mechanism, detailed in Section 3.3.2.
3. The management side of the core component (Labmanager). This is explained in an appendix in section 5.7.

The remainder of the section focuses on each of the parts above. Additionally, the benefits of integrating the remote laboratories in the Smart Gateway are explained in section 3.3.4. Additionally, section 5.2 covers the brief history of gateway4labs project since its origins before Go-Lab.

3.3.1 Support for standards

In gateway4labs, the core component (labmanager) natively supports three different systems:

- **OpenSocial:** described in this section.
- **IMS LTI:** Learning Tools Interoperability, it is a standard supported by the main LMS environments, which supports integrating external tools natively. Since the labmanager implements this standard, every laboratory supported can be automatically integrated in any LMS, which supports it.
- **HTTP-based plug-ins:** A custom HTTP interface is provided, so external tools not supporting IMS LTI or OpenSocial can consume this interface to connect to gateway4labs. This includes plug-ins in systems such as Joomla[23] or an LMS not supporting LTI, such as dotLRN[24].

Since the interface between gateway4labs and the ILS platform is based on OpenSocial, this section is focused on this implementation.

3.3.1.1 Context information management

When a user (teacher or student) is using the ILS platform, there is certain context information which could be useful for gateway4labs, such as: who is the user (if identified), where is the user accessing from (ILS) or what language is using in the ILS platform (e.g., French, German, Spanish, etc.). This contextual information is useful to provide feedback to the laboratory about who is using the laboratory.

3.3.1.2 Public laboratories

Once the administrator registers a laboratory, by default gateway4labs does not make it publicly available for unregistered users. This is the default behaviour in remote laboratory management systems. However, the administrator can configure that a particular laboratory is openly available for everyone.

This way, in the OpenSocial version it is possible to provide the laboratory to certain ILSs, or it can be available for every space, as it will be the common case.

3.3.1.3 Other features

Additionally, gateway4labs supports other optional features, such as enabling the laboratory to restart the widget whenever it is required. In WebLab-Deusto, for example, whenever the user

[23] https://github.com/gateway4labs/cms_joomla
[24] https://github.com/gateway4labs/lms4labs/tree/master/lms/dotLRN

performs a reservation, the user will be able to use a single reservation. Whenever the reservation is over (and the student is not using the laboratory anymore), a message is submitted to the labmanager to restart the widget and be able to perform a new reservation if desired.

So as to support this optional mechanism, gateway4labs provides a URL that will force a reload of all the widgets of the same laboratory. This URL is passed to the laboratory plug-in, so it can optionally load this URL when finished. Certain laboratories (like the iLab radioactivity laboratory, where the user can perform more operations once using it) do not require these features.

3.3.2 Support for remote laboratories. The plug-in system.

The core functionality of gateway4labs is to provide an API to support a wide range of laboratories. This API aims to be simple so as to encourage laboratory owners to develop their plug-ins, by avoiding strict requirements other than a link to the final laboratory crossing the reservation processes of the final laboratory. This API is described in detail in D4.1.

3.3.2.1 Implementations of the plug-in system

The base of the plug-in system is that the laboratory must support some mechanism to provide a link that identifies the current reservation. There are different ways to approach this:

- Using federation protocols: in the case of WebLab-Deusto, a federation mechanism is used to access a WebLab-Deusto server as if it was another WebLab-Deusto server requesting a reservation. WebLab-Deusto returns a URL which includes a reservation identifier, and the labmanager can use this link to redirect the user to that location. From that point, all the communications do not cross the gateway4labs infrastructure.
- Creating users dynamically: in the case of the iLab Shared Architecture, the plug-in creates a new user (if it was not previously made) and grants this user permission to use that laboratory (if he did not have that permission), and finally the user is redirected to the final system. There, the user can start using the laboratory he has access to.
- Using encryption to sign messages: in the case of the UNR (explained in Section 3.4.5), a secret is stored in the plug-in, and a message that includes the username, the current timestamp, and the laboratory is signed. The message and the signature are returned to the labmanager in the form of a URL. When the user is redirected to that URL, the laboratory at UNR verifies whether this message is valid and if it is, it enables the user to access the laboratory until the timestamp expires.
- Simple redirection: in the case of PhET (explained in Section 3.4.3), the plug-in just generates the public link to the PhET simulation. This way, the user is redirected to the public link directly.
- Federated search: in the case of ViSH (explained in Section 3.4.4), the plug-in must forward the query provided by the user to the ViSH repository, and then generate a link.

Other mechanisms could be employed, since this depends completely on how the final laboratory enables users to access.

Additionally, the plug-in must provide a way to define the initial configuration. In the case of WebLab-Deusto or iLab, this includes the particular server and credentials of this server. In the case of UNR, the secret key must be configured. In most cases, a default height is provided to establish a default height of the widget unless the iframe resizer feature is implemented by the particular internal laboratory.

3.3.2.2 Python version

Since gateway4labs is developed in Python, the basic plug-in structure is based on Python. The developer only needs to develop a module or a package called g4l_rlms_{{ name }}, and then in the configuration file of the Labmanager, the developer must add the name of the plug-in to the RLMS variable (which is a list, as shown below). For example, if we create a plug-in called “foo”, we just need to create a file called g4l_rlms_foo.py (or a package called g4l_rlms_foo, with different files inside), and in the config.py add:

RLMS = [‘weblabdeusto’,’ilabs’,**’foo’**]

Then, it can provide a function called get_module(version), so it could support different versions through having different modules. For example, iLab could have two independent modules that fulfill the Python API in the same plug-in, namely “g4l_rlms_ilab.py”, “ilab_3_1.py” and “ilab_3_5.py”, and in the first one it could provide the get_module(version) function. Whenever version was 3.1, it could rely on one module, and whenever the version was 3.5, it could rely on the other one.

The module used must provide an instance FORM_CREATOR, which creates the required forms. In the case of the Python version, the configuration is managed by subclassing a set of classes based on WTForms[25]. In these classes, the developer can select how to convert these values into a single JSON document. This JSON document will be stored in the database, and it will be used with all the instances of the plug-in (e.g., different instances of WebLab-Deusto). This process is detailed in Figure 12.

[25] http://wtforms.readthedocs.org/

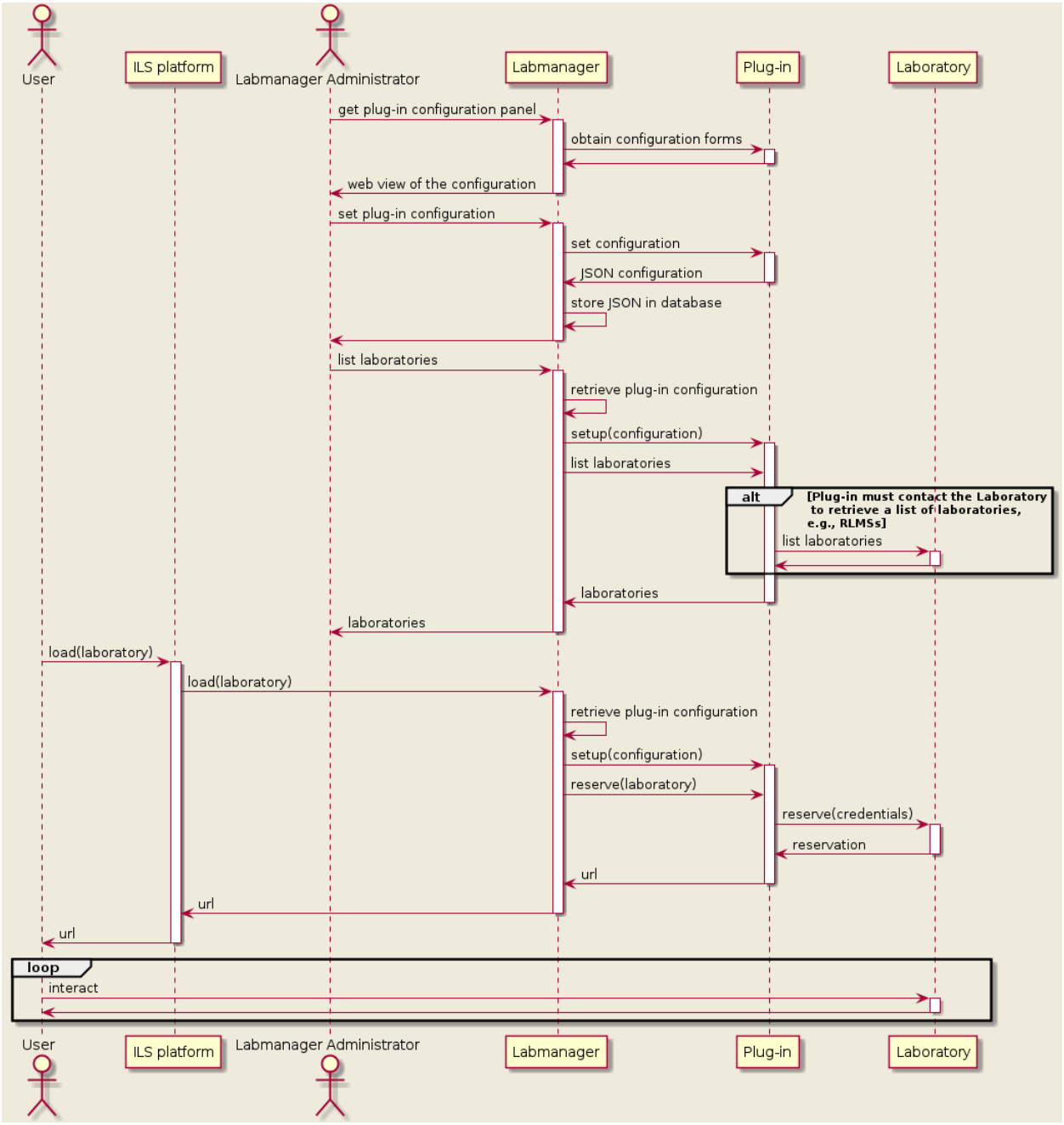


Figure 12: Configuration process in Python plug-ins

Finally, the module must create a class called RLMS. This class will be instantiated with the JSON configuration, and it is expected to use that configuration to perform each of the tasks mentioned in the previous subsection.

The following are examples of Python plug-ins:

- WebLab-Deusto plug-in.
  - Repository: https://github.com/gateway4labs/rlms_weblabdeusto
  - Notes: three files are used in a single package g4l_rlms_weblabdeusto. Two of them (weblabdeusto_client.py and weblabdeusto_data.py) are taken from WebLab-Deusto directly, while the third one (__init__.py) specifies the rest.

- iLab Shared Architecture plug-in.
  - Repository: https://github.com/gateway4labs/rlms_ilabs/
  - Notes: there is a single g4l_rlms_ilabs.py file which matches the specification explained above. There is also a ASPX file to be optionally added to the iLab server.
- UNR-FCEIA plug-in.
  - Repository: https://github.com/gateway4labs/rlms_unr
  - Notes: there is a single g4l_rlms_unr.py file. In this case, there is no communication between the plug-in and the final server, since it's based on a cryptographic solution where the plug-in generates and signs tokens that the user will forward to the final server.
- PhET plug-in.
  - Repository: https://github.com/gateway4labs/rlms_phet
  - Notes: this represents a set of simulations. There is no reservation process, so the reserve method is focused on generating links to the simulation.
- ViSH plug-in.
  - Repository: https://github.com/gateway4labs/rlms_vish
  - Notes: this represents a set of simulations. There is no reservation process, so the reserve method is focused on generating links to the simulation.

3.3.2.3 HTTP version

The HTTP version relies on a RESTful interface that can be implemented in any language to support the development of plug-ins in other technologies. It also makes it possible to decouple the Labmanager maintenance and the plug-in maintenance, since the plug-in could be deployed in another institution, as discussed in D4.1. Furthermore, a Python server supporting this RESTful interface will be implemented to make it possible to distribute the existing plug-ins in other institutions if desired.

In the approach selected (and explained in D4.1), the plug-in stores information of the final system such as credentials or URLs. To configure the plug-in and add this information, the Smart Gateway administrator will be redirected to a website provided by the plug-in. This workflow is described in Figure 13.

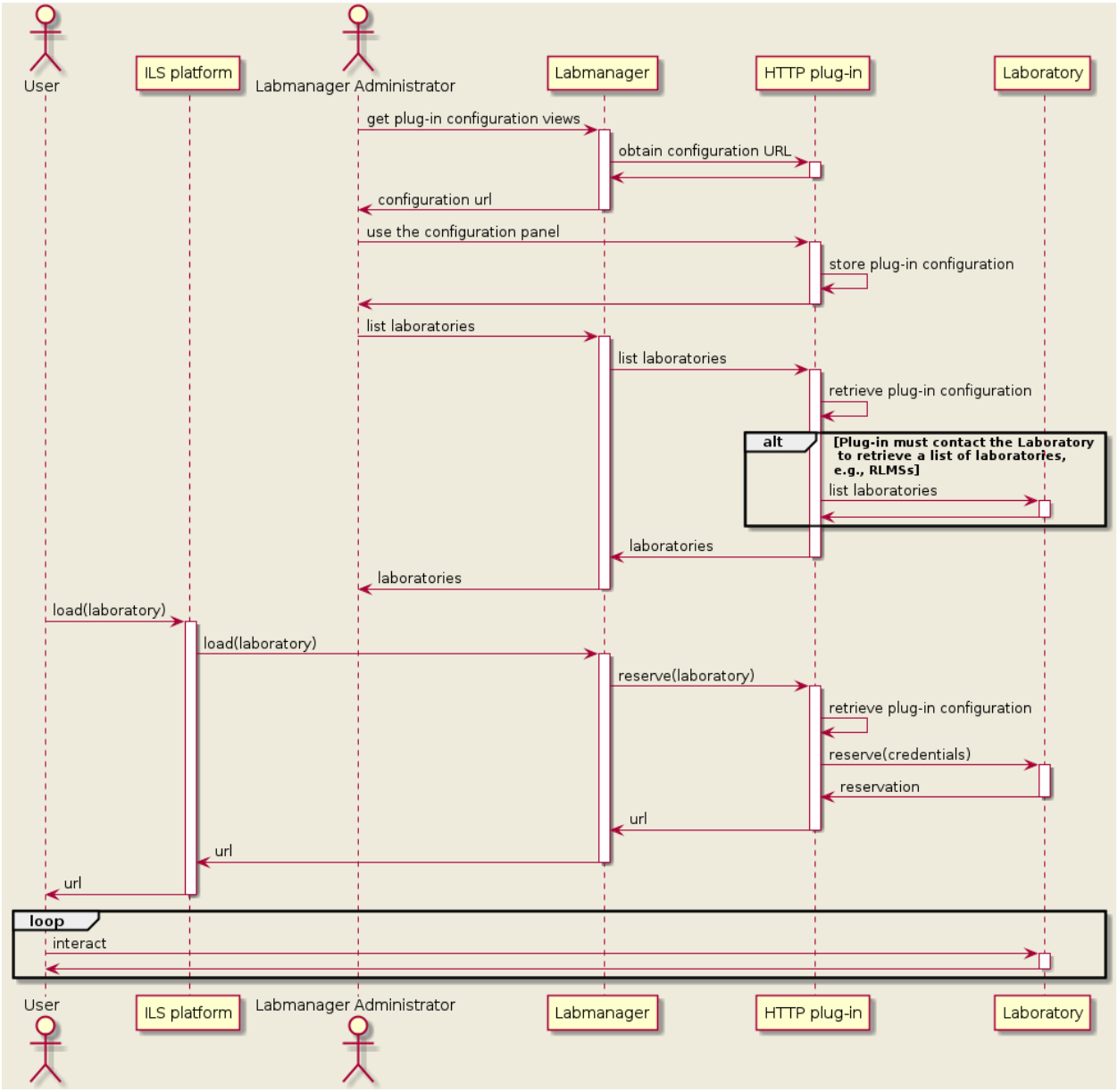


Figure 13: HTTP plug-in configuration stored in the plug-in side

This approach provides a considerably wider flexibility to the HTTP plug-in developer, since he can provide and upgrade in the future the configuration variables without interacting with the Labmanager. The panel can be very simple or very sophisticated, depending on the complexity of the remote laboratory. The model becomes much simpler, since it does not need to agree the available constraints. Additionally, the configuration secrets are kept on the HTTP plug-in side. On the other hand, this requires the HTTP plug-in to be publicly available to the Internet, while previously it could be just available in the Labmanager. It also requires the HTTP plug-in to be responsible of more roles, such as storing the configuration information.

A list of examples is provided in the following repository:

https://github.com/gateway4labs/labmanager/tree/master/examples/http_plugins/

3.3.3 Demo and Software Repository

A demo is provided with the following details. Take into account that for the sake of security, the whole database might be restored frequently, so changes might be discarded:

- URL: https://www.weblab.deusto.es/golab/labmanager_review/

- Username: admin
- Password: reviewerspassword

The details of the administration tools used for managing the Labmanager are available in Section 5.7. The remainder operations are detailed in Section 5.4.

Additionally, all the source code of the labmanager is in the following GitHub repository and documented in the following readthedocs site:

- http://github.com/gateway4labs/labmanager/
- http://gateway4labs.readthedocs.org/

3.3.4 Summary of the benefits for integrated remote laboratories

The focus of gateway4labs is attracting laboratory owners to the Go-Lab ecosystem, so they can share their labs in Go-Lab. To achieve this goal, the following incentives are provided:

1. **Easy integration:** A flexible, pragmatic approach to integrate existing remote laboratories into gateway4labs through the plug-in mechanism and the management panels. The amount of code required is not relevant, since it only acts as an initial bridge, and two interfaces (native Python API and HTTP API) are provided. Multiple deployment schemas are supported, as specified in the Architecture section of D4.1.
2. **Additional Go-Lab incentives:** Go-Lab will provide the laboratory owners with several benefits. The most important one is the visibility of the laboratories. Thousands of students and teachers will be able to easily find the federated laboratories. Other benefits include the support of Go-Lab Add-on services, such as the booking system. Certain remote laboratories might have a queue for managing students, which does not scale well. However, if the laboratory is integrated in gateway4labs and gateway4labs supports the booking mechanism of the Go-Lab portal, then the laboratory will be only available to students of groups which have booked the laboratory, reducing the amount of concurrent students.
3. **Keep control of the laboratories:** The Go-Lab project requires that the laboratories are open and no registration is needed, and the Smart Gateway will encourage developers to keep their laboratories open. However, gateway4labs provides mechanisms to enable that a remote laboratory is only available to a particular ILS (see Section 5.4). This would require those interested schools to register in the gateway4labs server used by that laboratory owner, and then the laboratory owner could provide the laboratory only to that school. The visibility of the laboratory would be consequently decreased. However, guaranteeing this type of control to the laboratory owner, it is possible that certain laboratory owners could have fewer rejections to participate in the integration, and once integrated; they can consider if they finally open their laboratories.
4. **Use a federated approach:** gateway4labs plug-ins support federation mechanisms if these are provided by the integrated systems. For example, WebLab-Deusto provides a federation protocol so one WebLab-Deusto system with 4 laboratories can share a subset of them to other WebLab-Deusto system. The plug-in of this WebLab-Deusto benefits of this feature so it translates requests from gateway4labs (which come from the ILS platform) as if it was an external WebLab-Deusto system requesting a laboratory for a local user. This federated approach enables remote laboratories to be also provided through their original portals or be integrated in other tools, while increasing their visibility by sharing them with Go-Lab.
5. **Incentives outside Go-Lab:** gateway4labs does not only support OpenSocial, but also other specifications, such as IMS LTI, and a HTTP interface to be deployed in custom systems (such as CMS as Joomla). If a laboratory owner aims to integrate a remote

laboratory in a Moodle LMS or a Joomla CMS, then gateway4labs is a tool that makes this process easy, so the laboratory owner only needs to develop the plug-in for the remote laboratory. Once this plug-in is developed, intended for supporting an LMS or a CMS, this remote laboratory is, from a technical perspective, available and compatible for the Go-Lab context.

3.4 Prototypes

In this section we will introduce the Smart Gateway plug-in prototypes developed for well-known remote lab management systems and other legacy online lab systems. For any legacy remote lab managed by one of these RLMS the Smart Gateway will offer out of the box support for its integration in the Go-Lab infrastructure. Depending on the RLMS and the integration level desired (see D4.1 for details on the different integration levels) the support for the plug-in might require some implementation also at the RLMS or lab owner's side. In cases like this, not all versions of the RLMS will be supported.

3.4.1 WebLab-Deusto

WebLab-Deusto is an Open Source remote laboratory management system. It supports the development and administration of remote laboratories. In WebLab-Deusto, remote laboratories are usually managed through a priority queue.

A key feature of WebLab-Deusto in this context is its federation model. A simple scenario where two WebLab-Deusto instances are using it is presented in Figure 14. On it, a user reserves a laboratory in the University A, which forwards the request to University B without requiring the user to be registered in University B. Essentially, the plug-in consists of a client of WebLab-Deusto (extracted from WebLab-Deusto, since it is also developed in Python). The client only requires a URL, a username and a password. This username and password represent a federated node, and each federated node can use it to proxy all their requests.

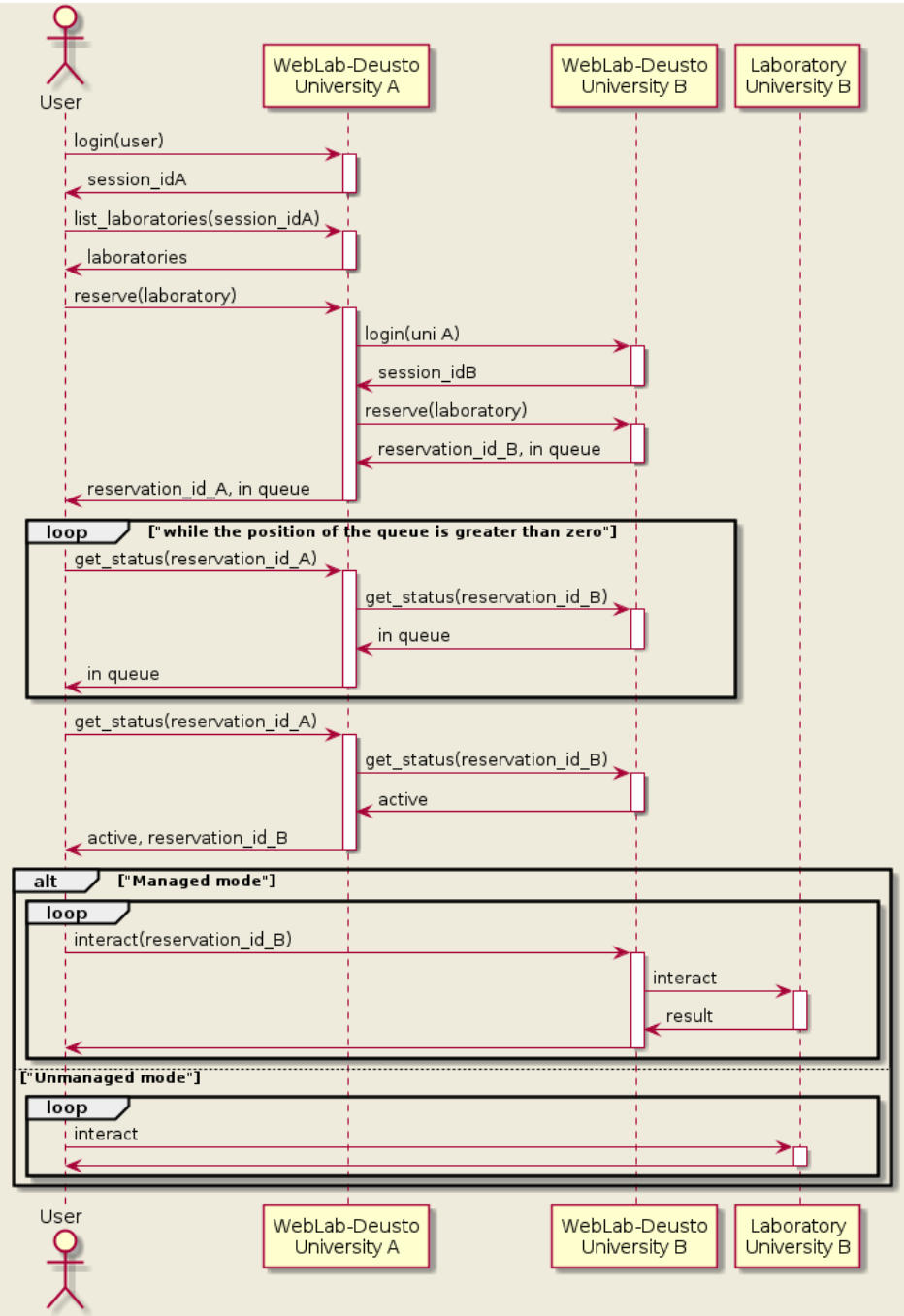


Figure 14: Federation algorithm in WebLab-Deusto

While WebLab-Deusto through the different institutions that compound it provides different remote laboratories for engineering studies, some of them are suitable for secondary schools. In particular in Go-Lab the Archimedes laboratory will be used:

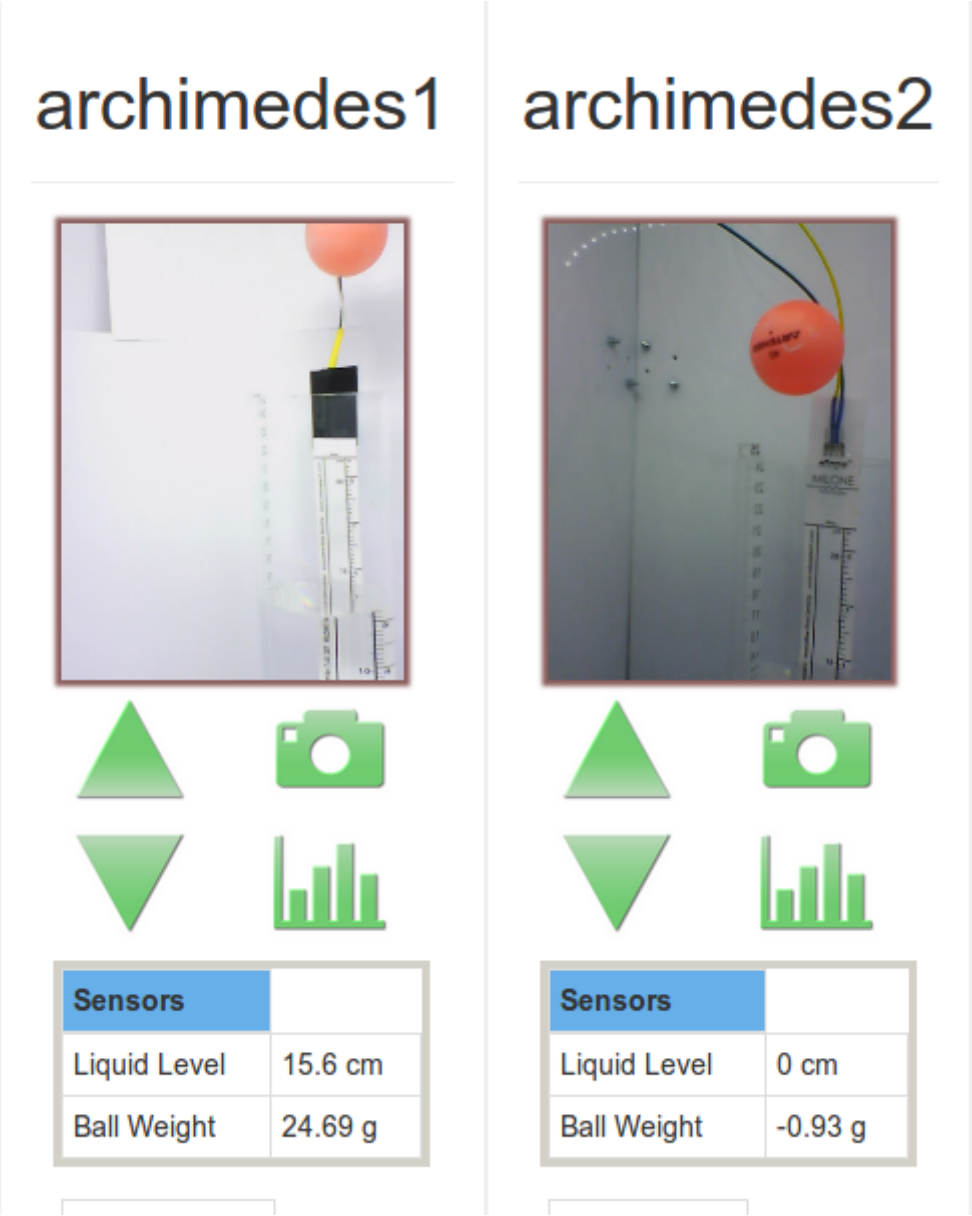


Figure 15: Two of the four balls in the Archimedes laboratory

3.4.2 iLab Shared Architecture (ISA)

The iLab Shared Architecture is a Web services-based software architecture developed at the Massachusetts Institute of Technology (MIT) that offers online lab developers and users a common framework to use and share Online Labs. It separates the experiment logic from the management part like managing users' accounts, user authentication and other tasks that follow a lab session. It is middleware architecture with a service broker providing common shared services for lab servers and lab clients.

3.4.2.1 Smart Gateway Plug-in for ISA

According to the specifications defined in deliverable D4.1 a Full version of the plug-in for the iLab shared architecture was developed (see levels of integration in D4.1). Since ISA requires a user to be uniquely identified within a service broker to run an experiment, the developed plug-in bridges the legacy system authentication. The interaction of the plug-in with the legacy lab system (iLab service broker) is described in Figure 16. In this diagram it was skipped the interactions that take place before between user, ILS Platform and Smart Gateway since it was described in detail in deliverable D4.1.

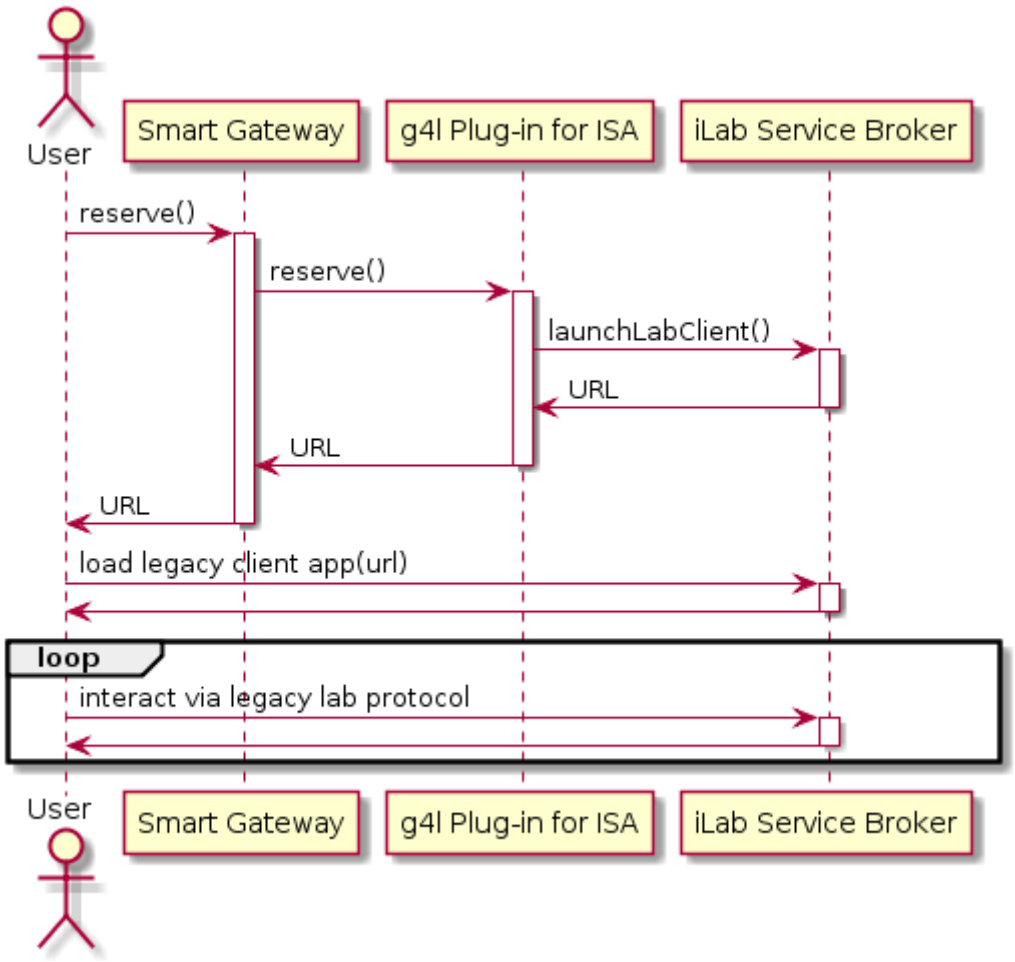


Figure 16: Gateway4labs plug-in for ISA workflow

This diagram assumes that the lab gadget was successfully loaded and that Smart Gateway and iLab Service Broker administrators exchanged the necessary credentials and that Smart Gateway and iLab RMLS are registered respectively.

ISA exposes its functionalities to external clients and lab servers via a SOAP Web services API. When a request to reserve a lab is started by the user this causes the plug-in to contact the iLab service broker by calling its Web method *launchLabClient().* The parameters provided are described below.

Name	Type	Description
clientGuid	string	Unique ID of the client to be launched
groupName	string	Name of the group it belongs to
userName	string	User requesting access to it
authorityKey	string	Unique ID of the Smart Gateway (assigned by service broker). This credential should be exchanged beforehand with system administrators
start	dateTime	Reservation starting time

duration	long	Duration of reservation in seconds

Table 1. Description of the launchLabClient service

This service will return a URL that will launch the particular lab client requested. At the user's side the lab client will be launched in the OpenSocial container. The interaction that follows between lab client and server is carried out using the legacy lab's own protocol.

3.4.2.2 The Radioactivity Laboratory

The radioactivity laboratory was developed and deployed at the University of Queensland in Australia and it has been used by secondary schools in the USA through the Northwestern University. This lab allows students to explore how radioactive radiation changes as a function of distance. The intensity of radiation emitted by Strontium -90 sources is measured by a Geiger counter at different distances set by the students. This lab was initially designed in compliance with ISA and runs in batched mode.

In the context of Go-Lab, the Carinthia University of Applied Sciences (CUAS) deploys a MIT iLab Service Broker, and the Radioactivity laboratory is registered on it. Therefore, using the g4labs plug-in for ISA, it is possible to have this lab available in the Go-Lab infrastructure. The figure below shows the radioactivity lab client available to all Go-Lab users via an ILS

ILS URL: http://graasp.epfl.ch/metawidget/1/014683d8d8664a5b95090e668eeb7fbe44aaf59a

The radioactivity lab was chosen as the first ISA compliant lab available via the Smart Gateway because it is very well suited for secondary schools. It has been successfully deployed in large scale for secondary school students in the Unites States. However, since the g4l plug-in for ISA allows any ISA compliant lab to be included in the Go-Lab platform, a large pool of labs could be easily made via Go-Lab.

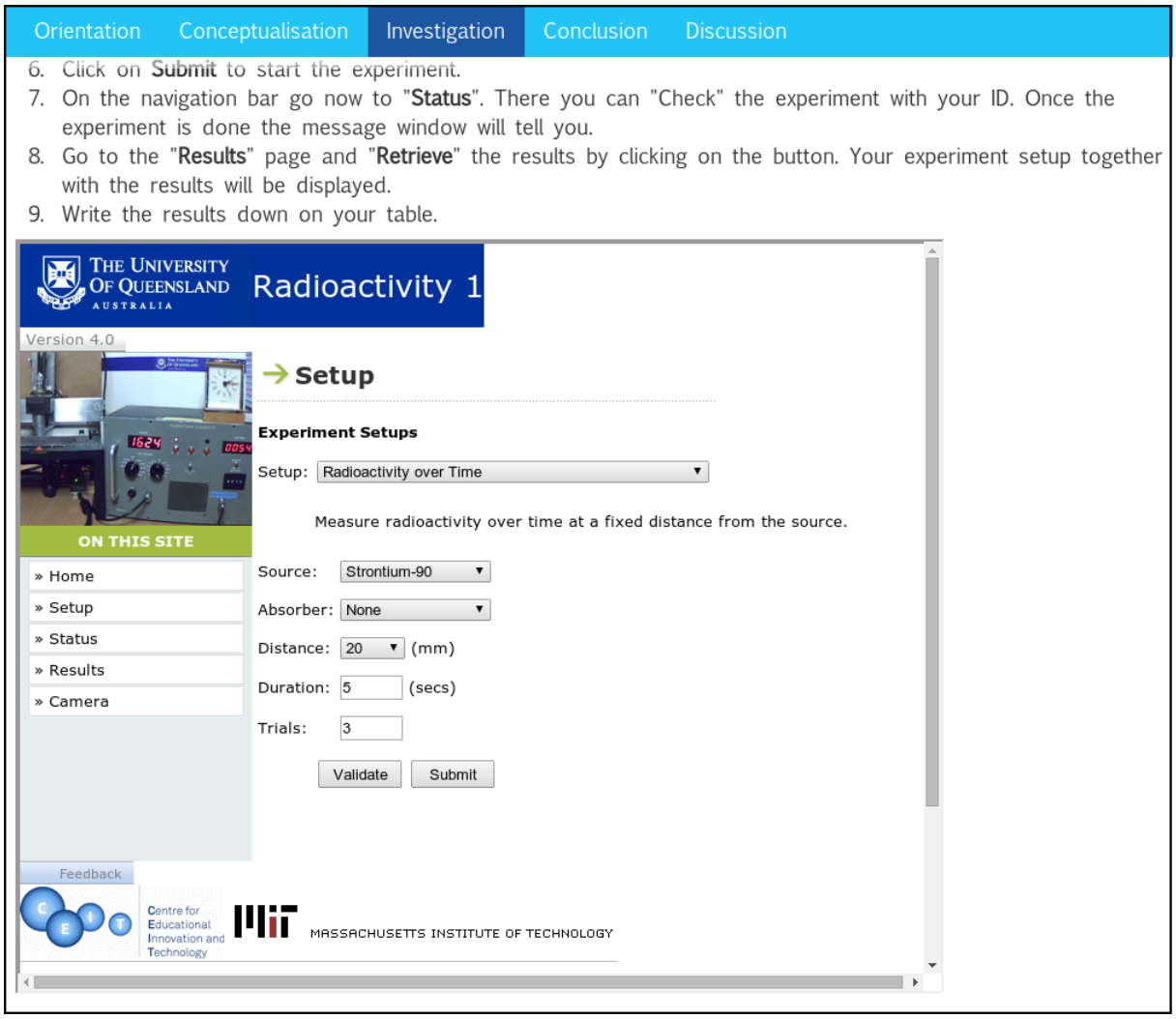


Figure 17: Radioactivity Lab embedded in an ILS

3.4.3 PhET

The PhET Project (Physics Education Technology) created useful simulations for teaching and learning physics and makes them freely available from the PhET website (http://phet.colorado.edu). The simulations (sims) are animated, interactive, and game-like environments in which students learn through exploration [1].

Many of the simulations cover introductory high school and college physics, while others introduce more advanced topics, e.g., lasers, semiconductors, greenhouse effect, radioactivity, nuclear weapons, and Fourier analysis. Users, however, have included students from grade school through graduate school. On the website, the sims are organized under nine loose categories: Motion; Work, Energy & Power; Sound & Waves; Heat & Thermo; Electricity & Circuits; Light & Radiation; Quantum Phenomena; Chemistry; Math Tools; and Cutting Edge Research [1].

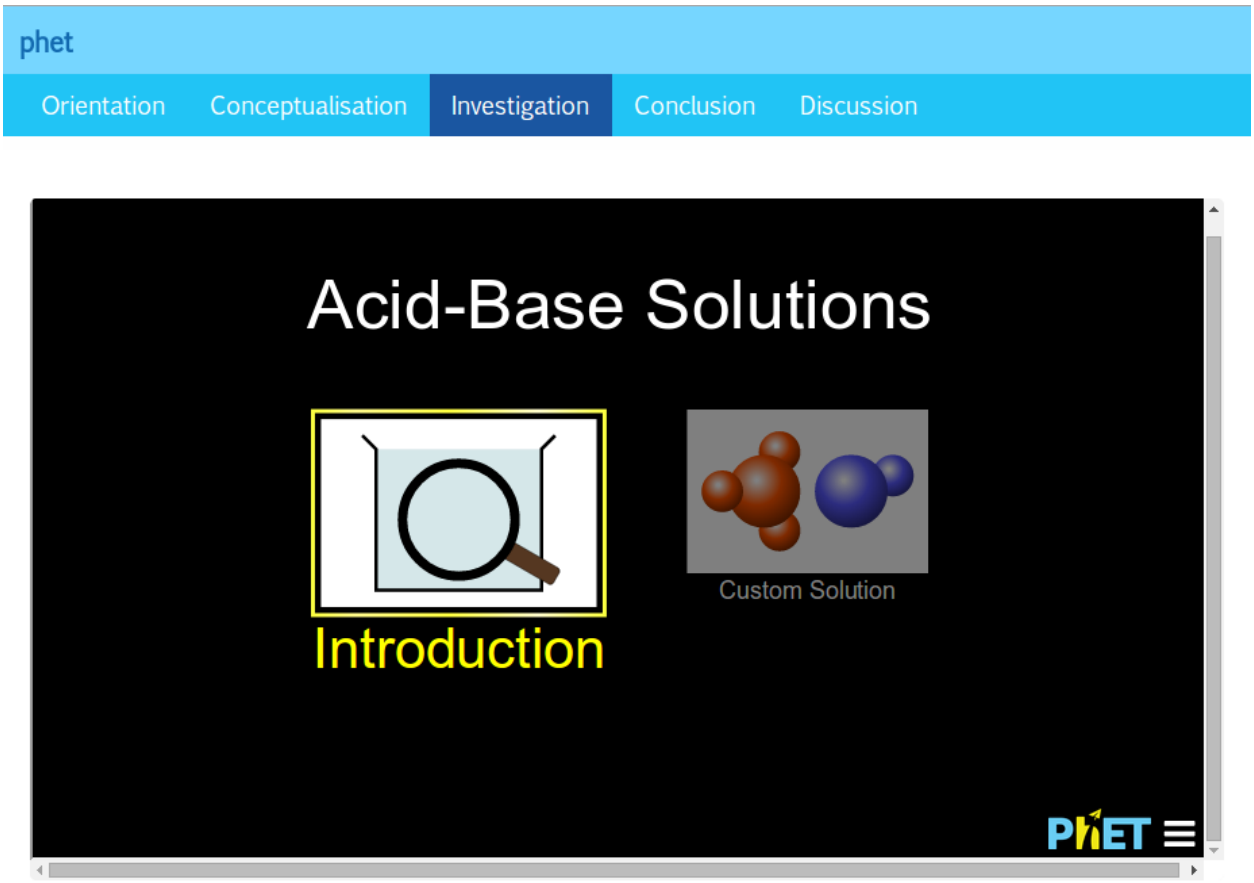


Figure 18: A PhET simulation included in the ILS platform

3.4.3.1 The gateway4labs PhET Plug-in

The plug-in developed to support PhET laboratories follows option 1 (iFrame in Smart Gateway) described in section 3.2. Since PhET laboratories do not require any type of authentication the plug-in development was very simple. It provides the PhET lab via an iFrame in an OpenSocial container. Furthermore the plug-in harvests the PhET Website to fetch the labs available.

PhET laboratories are simulations that run locally in the client application. Some of these applications are Java-based that can be downloaded and launched via the Java Web Start framework.

3.4.4 ViSH

The ViSH project[26] is part of the FP7 Global Excursion Project[27], which provides a portal that lists, organizes and displays a wide set of open educational resources including simulations and remote laboratories among others. All the contents must be open and they are publicly available, and HTTP APIs are provided for searching resources.

The resources can be listed alone, but they are commonly organized in "Excursions". An Excursion is similar to a slideshow, but where each slide is a resource. Resources include embedded pages (e.g., wikipedia), tests, Adobe Flash objects, laboratories and other rich contents.

[26] http://vishub.org/
[27] http://www.globalexcursion-project.eu/

3.4.4.1 The gateway4labs ViSH Plug-in

The plug-in developed to support ViSH laboratories acts as a federation proxy. It uses the search API (which is a JSON HTTP interface) to enable the Smart Gateway administrator searching in the ViSH repository.

This way, the gateway4labs plug-in enables the Smart Gateway administrator to search for resources, and if it is an excursion, it splits the excursion into multiple apps that can be exported through gateway4labs to the ILS. The following figure shows a slide of an existing excursion[28] displayed in a ILS.

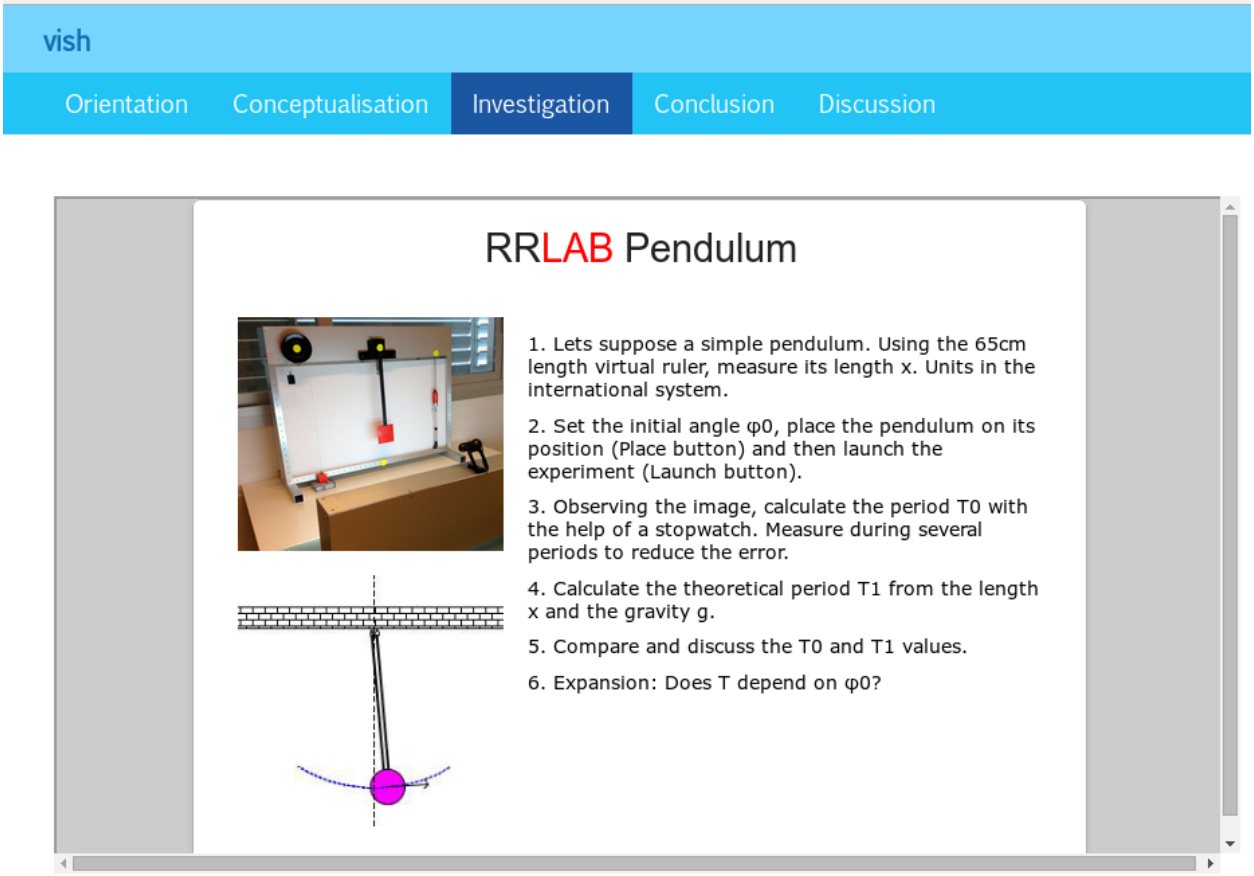


Figure 19: A ViSH resource included in the ILS platform

3.4.5 UNR-FCEIA

The National University of Rosario (Argentina) have developed a physics remote laboratory[29]. On it, students can create electronic circuits and test currents. The remote laboratory is a pure HTML5 application that does not require any external plug-in and internally it has an internal queue so if multiple students attempt to use the laboratory at the same time, they will be multiplexed in time. A mechanism for its inclusion in external systems is developed in the system itself, relying on a cryptography mechanism.

[28] http://vishub.org/excursions/162.full
[29] http://labremf4a.fceia.unr.edu.ar/

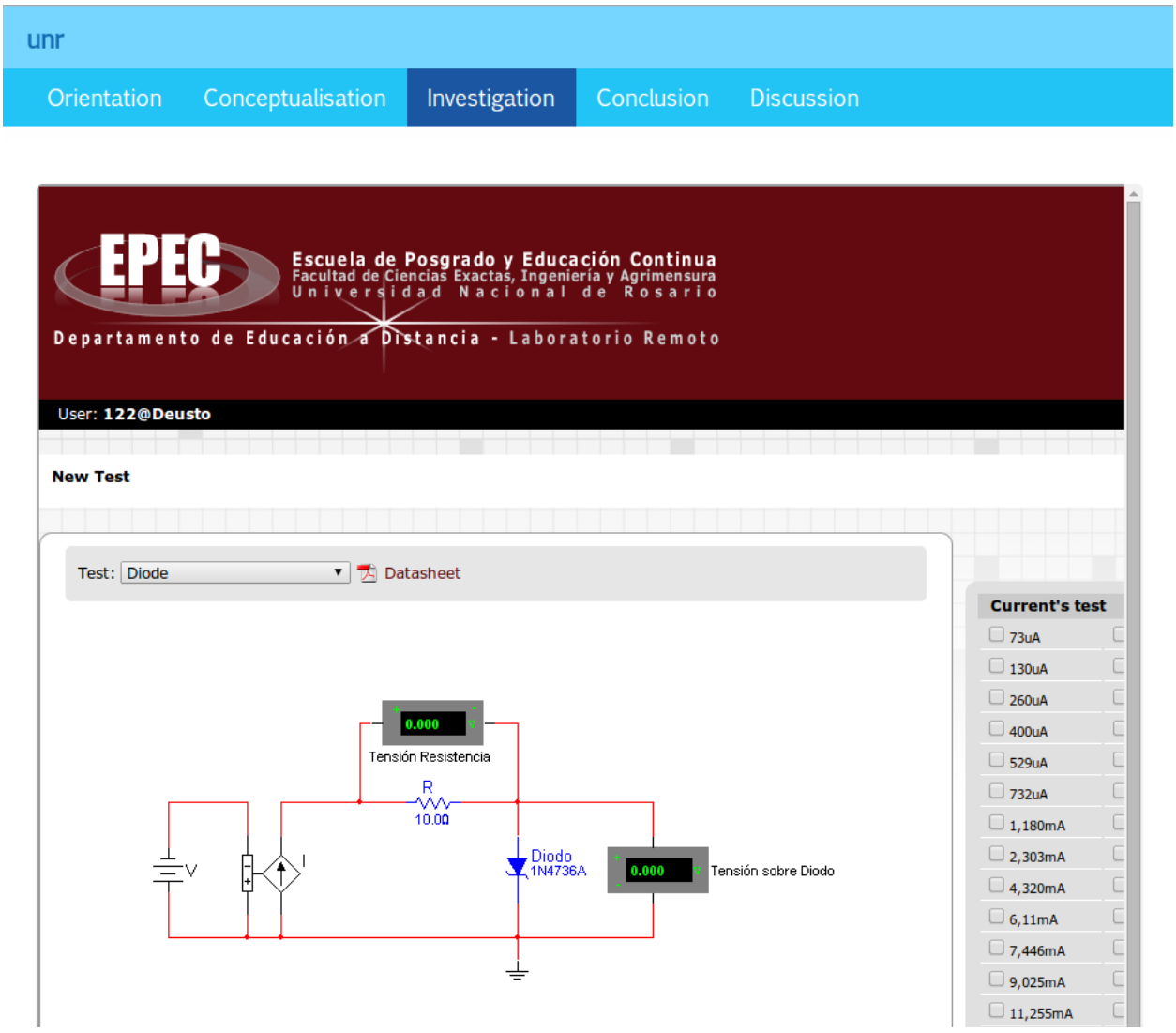


Figure 20: A ViSH resource included in the ILS platform

3.4.5.1 The gateway4labs UNR-FCEIA Plug-in

The UNR-FCEIA plug-in takes advantage of this cryptography mechanism. Basically, the plug-in uses a shared secret stored in the gateway4labs database to sign a message that provides details such as a timestamp or who is accessing. Then, the user will be redirected to the remote laboratory with that message. The remote laboratory will take this message and verify that it has been signed correctly by a valid actor. This way, the plug-in never contacts the remote laboratory never directly.

4 Conclusion and future work

The GitHub repositories contain the current software releases of the Smart Device and Smart Gateway components (lab-owner and cloud services). It also contains a wiki and a set of readme files that allow lab owners to use the examples provided as templates to add their own labs online or federate their own remote lab management systems with the Go-Lab infrastructure.

This repository will be continuously updated to reflect the evolution of the Smart Device and Smart Gateway specifications and to provide a richer set of templates.

The releases delivered at this initial stage rely on a pragmatic approach to trigger adoption by lab owners while also promoting innovative practices. Adoption is triggered by offering a set of alternative but consistent plug solutions for new remote labs following the Smart Device specifications, as well as a set of Smart Gateway plug-ins enabling various levels of integration of legacy labs already available in existing remote lab management systems. The usage of examples as templates is also suitable for lab owners being typically not Web developers. The innovation is triggered by the implementation of specifications relying on state-of-the-art open web standards like WebSockets and the recommendation of all-in-one embedded solutions aiming at simplifying deployment efforts and reducing costs.

One should also highlight that the current version of the Smart Gateway not only enables the integration of remote lab management systems, but also the federation of selected repositories with seamless packaging of online labs as open social apps for easy integration in Go-lab inquiry learning spaces. To be in line with lab-owners requirements and Go-Lab quality control policy, no automatic harvesting is implemented. The proposed approach enables to select only the labs relevant for STEM education at school.

The objectives for the final releases of the lab-owner and cloud services due M27 (D4.7) is to enrich the set of examples with more advanced labs serving not mainly as templates for lab-owners but also fully as resources for students. Additional remote lab management systems will be integrated in the Go-Lab infrastructure following a combined top-down and bottom up approach; i.e. by responding to requests from RLMS owners to get visibility though Go-Lab and from teachers to access selected labs available in existing RLMS. The implementation of these examples will contribute to the refining of the Smart Device and Smart Gateway specifications towards the delivery of their final version (D4.5) at M30.

5 Apendix

5.1 Appendix A: Lab owner survey results

In this section is is presented the results of the Lab Owner's survey.

Questions:

1. What laboratories do you have that could fit in the curriculum of primary and secondary schools?

2. Would you share your labs with the Go-Lab community?

3. Would you share your labs to increase their visibility?

4. Would you share your labs to get third parties involved to create pedagogical content using the labs?

5. Would you share your labs without requiring direct economic remuneration?

6. Under which Remote Laboratory Management System do you have these labs, if any?

7. Which scheduling mechanism does your system use?

8. Does your system support concurrent access by multiple students to the remote laboratory?

For questions 2, 3, 4, 5 a rating scale was used, being 1 = not likely, 6 = very likely.

1	2	3	4	5	6	7	8	Institution
Currently all our labs are for Universities	4	1	5	2	Labicom	Calendar	No	BMSTU/Labicom
Mobile Remote Experiment rexlab.ufsc.br	6	6	6	6	None	None	No	Universidade Federal de Santa Catarina
Digital electronics Analog electronics (under development)	4	5	4	4	Open University of Catalonia propietrary access	Calendar	No	Open University of Catalonia
At this moment none. Our lab is more for university (diodes, BJT, FET, LED)	6	6	6	6	LABREM-FCEIA linked to WebLab-Deusto	Queue	Yes	FCEIA-UNR (Argentina)
NetLab - a remote laboratory for electrical circuit analysis	6	6	6	6	NetLab has it's own management system	NetLab has it's own booking system - users are able to book any time	Yes	University of South Australia
Laboratory with LabVIEW.	4	4	4	5	CompactRIO in LabVIEW	LabVIEW	No	"Vasile Alecsandri" University of Bacau
Knowledge Lab	6	6	5	4	None	Don't know	No	Innovindia

The Genius Maker Virtual Physical Laboratory (VPL) Virtual Physics	5	5	5	4	Own L/WAN setup	Queue	Yes	University of Ulster
We used to have an on-line telerobot and operated this from 1994-2005. However, the maintenance depended on volunteers and when research interests moved on, we had to close it and restrict access to our own students which we have done since then.	1	1	1	1	Out own LabVIEW system	Calendar, Queue	Yes	University of Western Australia
We currently have curriculum based upon radioactivity (see link below) and we are working on curriculum based on remote access to an X-Ray Diffractometer and a Nitrogen Analyzer. We are also working on developing a new user interface for our labs which will be ready for wide use by Fall 2014.	6	6	6	6	iLabs Shared Architecture	Queue	No	Northwestern University
eLab3D (a remote laboratory for learning electronics based on 3D virtual worlds)	5	6	6	4	Technical University of Madrid Virtual Labs	Calendar	Yes	Technical University of Madrid
Engineering experiments. Showing cause and effect, stimulus and response, input and output. Temperature, flow rate, pressure, water level, concentration, etc.	6	6	6	6	LabVIEW. iLabs future	None	Yes	Univ Tennessee at Chattanooga
	6	5	6	6	elabs FEUP	a booking system not compatible with present moodle versions...	Yes	FEUP
At the moment, None	6	4	5	6	None	None	No	
None. Our remote laboratory (R-DSP Lab) is focusing in undergraduate and postgraduate students of universities	3	4	4	3	The R-DSP Lab has its own management system	Queue, It will be added a calendar based booking system	Yes	University of Patras, Greece
We have two radioactivity rigs that are being used by high school students in the US.	6	6	6	6	iLabs Shared Architecture	Queue	No	The University of Queensland
We are able to characterize	5	5	4	3	iLabs Shared	Calendar,	No	IUL / TU

material with highend testing equipment. This could be used for studying physics for example.					Architecture	Queue		Dortmund University
Laboratory of Optic and Quantum Physics	4	3	3	3	RIDS - BMSTU, Moscow, Russia	Calendar	Yes	Bauman Moscow State Technical University
nQuire (notably the nQuire Moon Rock Demonstrator) www.nquire.org.uk	6	6	5	6	None	None	Yes	The Open University
Laboratorio Remoto de FCEIA-UNR	6	6	6	6	WebLab-Deusto	Queue	Yes	Universidad Nacional de Rosario, Argentina
We plan to re-setup our VISIR Lab.	5	6	6	5	None	Don't know	Yes	FH Campus Wien
Black Body radiation, Long Jump , cucumber growth	6	3	4	6	iLabs Shared Architecture, None	Queue, None	Yes	CUAS
Control reservoir and tank system	5	6	4	6	None	Queue	No	
elabs in electronic field elab in industrial computer field	6	6	5	4	iLabs Shared Architecture	Calendar	Yes	ENET'COM
At this moment we do not have any online laboratory that would fit for primary and secondary schools.	6	6	5	5	Custom built	Queue	Yes	
Tesla coil Converting heat to electricity (Thermoelectric converter) Demonstration Solar Cell String resonance	6	6	4	6	iLabs Shared Architecture	None	No	Ilia State University

Table 2: Results of Lab Owner's Survey

5.2 Appendix B: Brief history of gateway4labs

In 2012 [2], an integration of the WebLab-Deusto Remote Laboratory Management System (RLMS) in a Learning Management System (Moodle) and a Content Management System (Joomla) is described by University of Deusto and UNED. The key concept was that a federation protocol was used to perform this integration: WebLab-Deusto did not manage the authentication or authorization of the individual students (managed by the Moodle and Joomla administrators), but only the connection with the two tools. Both plug-ins for Moodle and Joomla were developed, so they could connect to WebLab-Deusto using its federation protocol, and they provided management layers (e.g., which course could access what laboratory).

This concept, developed in September 2011, could not scale to other Remote Laboratory Management Systems: while the concept could be applied, supporting 3 LMS in 3 RLMS would require 9 plug-ins (3x3), since the plug-in for Moodle for WebLab-Deusto would not work for the iLab Share Architecture. Furthermore, each of these plug-ins dealt with their own management panels and database tables.

For this reason, in May 2012 a pet project called lms4labs started being developed between University of Deusto and UNED, with no associated funded project. Its focus was to avoid the problems presented in the previous solution, by putting in the middle a core component, called LabManager. This component supported an HTTP interface designed to be particularly small and for being consumed by LMS/CMS/PLE. It also supported a plug-in mechanism for supporting more than one RLMS. On June 2012, a first demo was available of a single LMS (Moodle) using a single RLMS (WebLab-Deusto, and all its remote laboratories). Now supporting 3 LMS and 3 RLMS would require only 6 plug-ins (3+3). Additionally the size of each plug-in is considerably smaller, since all the management was already provided by this core component. And if RLMS A developed a plug-in for LMS A, then RLMS B would benefit from that plug-in, too.

MIT (developers of the iLab Remote Laboratory Management System) was interested in this approach, so they started supporting the project by providing two developers between October 2012 and February 2013, adding support for IMS LTI. This standard is supported by many LMS systems, including Moodle (since version 2.2), so with this contribution, every RLMS would automatically support all those LMS, in addition to those supported through plug-ins. A plug-in was developed for Joomla. The results of this pet project were presented in [3].

Within the context of Go-Lab, lms4labs was chosen to act as an initial codebase for the Smart Gateway, since it provided a plug-in mechanism for integrating remote laboratories. As such, it was renamed to gateway4labs[30]. This way, the project grew, with new developers from other Go-Lab partners, and major changes in the architecture for supporting OpenSocial (all the plug-ins had to be changed), with more management layers, supporting public (with no authentication) laboratories for constraints of the project and becoming more robust and scalable. Features not previously provided, such as internationalization, which covers the whole codebase, had to be implemented within Go-Lab. Additionally, new remote laboratories were supported, including iLab (which required changes in the iLab Shared Architecture) and PhET.

The following screenshots show the commits done in the development of the core component (labmanager), as well as in the WebLab-Deusto, iLab Shared Architecture and PhET plug-ins. In the first one, it has been selected (shadowed) the time since the Go-Lab project started.

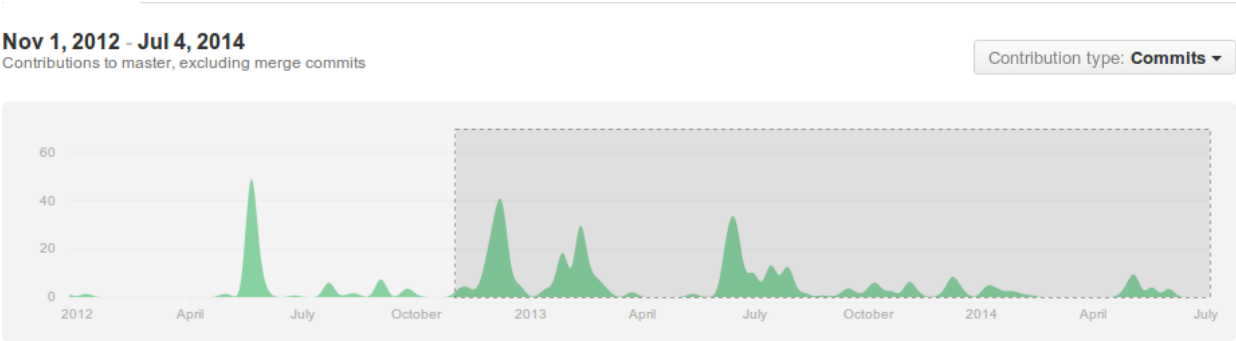


Figure 21: Development of the labmanager[31]

[30] http://gateway4labs.readthedocs.org/
[31] https://github.com/gateway4labs/labmanager/graphs/contributors



Figure 22: Development of the WebLab-Deusto plug-in[32]



Figure 23: Development of the iLab Shared Architecture plug-in[33]

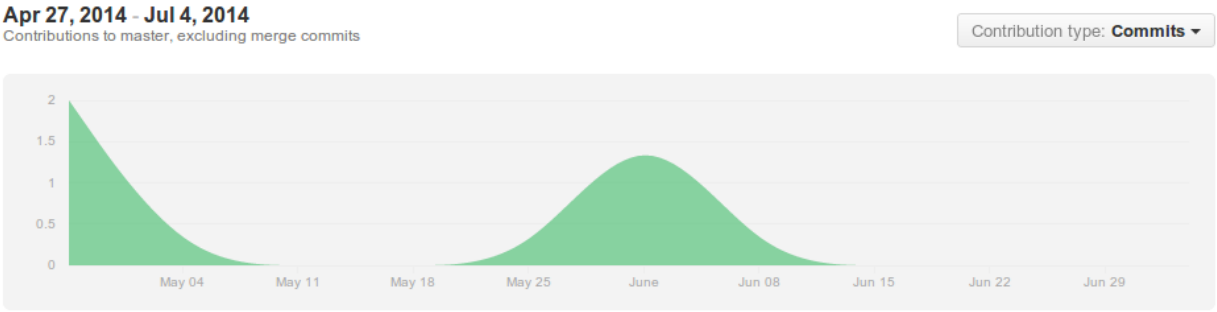


Figure 24: Development of the PhET plug-in[34]

5.3 Appendix C: Smart Gateway plug-ins size

As it has been previously explained, the plug-in is only a translator of the reservation process among different systems (i.e., plug-in) and the Labmanager. The development of each process is not too complex. Indeed, the percent of code that each of the plug-ins represent is typically small. In the following table, the lines of code of the Labmanager and each of the plug-ins (in the case of Weblab-Deusto is splitted in two since the client was taken from the source code of Weblab-Deusto):

Component	LoC

[32] https://github.com/gateway4labs/rlms_weblabdeusto/graphs/contributors
[33] https://github.com/gateway4labs/rlms_ilabs/graphs/contributors
[34] https://github.com/gateway4labs/rlms_phet/graphs/contributors

Labmanager (code)	5209
Labmanager (templates)	1834
WebLab-Deusto (client)	693
WebLab-Deusto (integration)	**253**
iLab Shared Architecture	**223**
UNR-FCEIA	**151**
Virtual	**93**
PhET	**153**

Table 3: Lines of code of the plug-ins and shared components (Labmanager)

As it can be seen, the amount of code of each plug-in is considerably low, while the Labmanager (which provides most features) is shared among all of them. Taking this into account, the following diagram shows graphically this proportion:

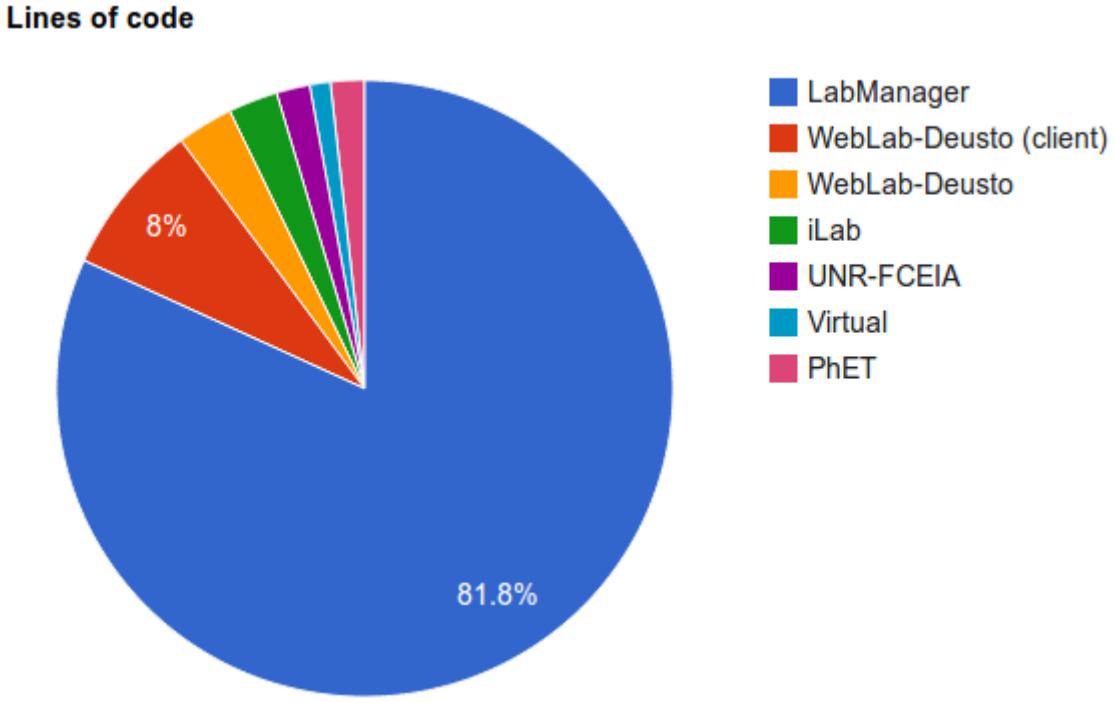


Figure 25: Percent of each of the plug-ins and the Labmanager

However, when a lab owner develops a plug-in for a particular laboratory, it does not matter the code of the rest of the plug-ins, since they are not going to be used for the target laboratory. In that case, if we take the iLab plug-in as an example, the percent of lines of code obtained by the Labmanager is considerably big compared to the lines required for integrating the laboratory:

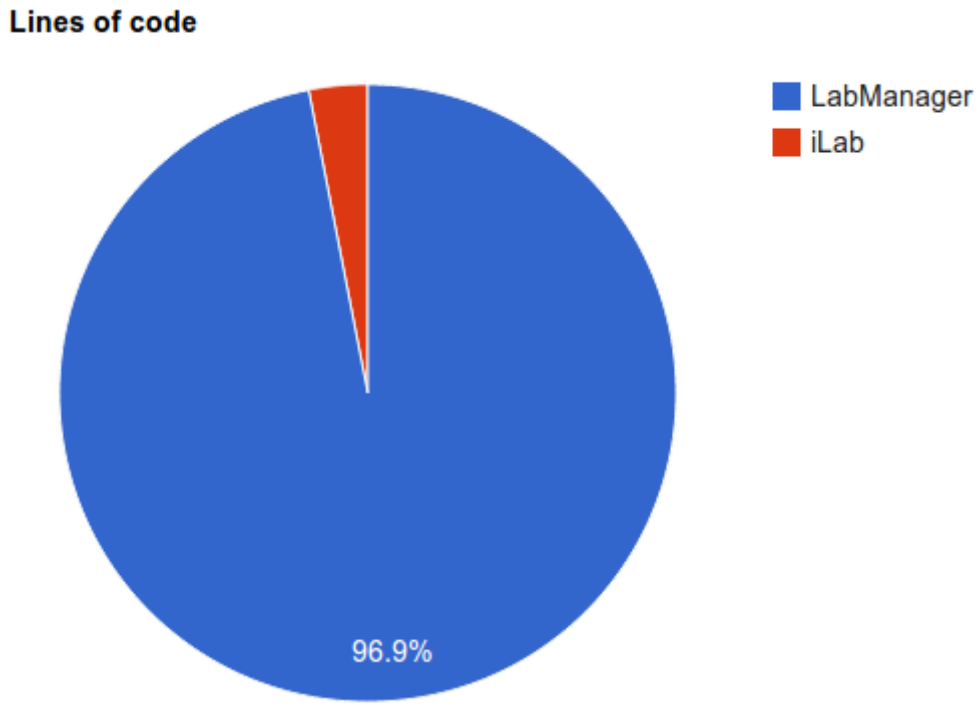


Figure 26: Comparing a single laboratory with the rest of the Labmanager code

While measuring a software component in lines of code is not accurate, the values presented do give an impression of the magnitude of obtained code for the required code.

5.4 Appendix D: Smart Gateway Support for advanced features on top of OpenSocial

As previously stated, every laboratory published in the Go-Lab portal must be publicly available without requiring teachers to register the school in the laboratory or in the Smart Gateway. However, mechanisms to grant laboratory owners control the access were implemented. The target of these mechanisms is to encourage laboratory owners to develop the plug-ins safely (being in control of the full process), and once they are developed and tested and they can see the whole process working, so from a technical perspective no further effort is required to integrate those laboratories in Go-Lab, encourage them to share them in Go-Lab by making them public. Therefore this process is the establishment of a set of optional steps targeting laboratory owners who are less receptive to the idea of sharing their resources publicly.

So as to support this, first the school must be registered. This can be done by a teacher:

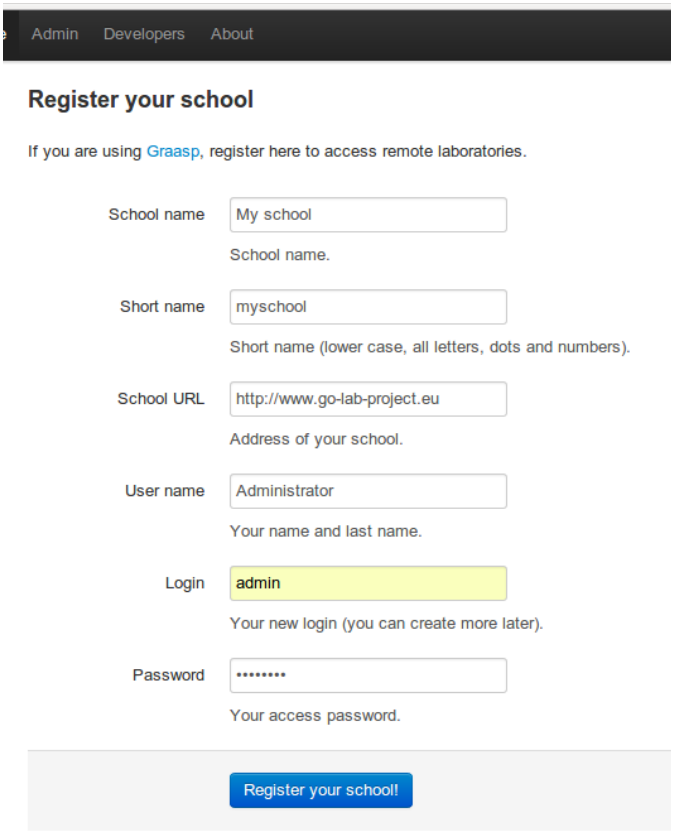


Figure 27: Registering the school in the Labmanager

Once the school is created, the teacher must log in the Labmanager with the credentials just created:

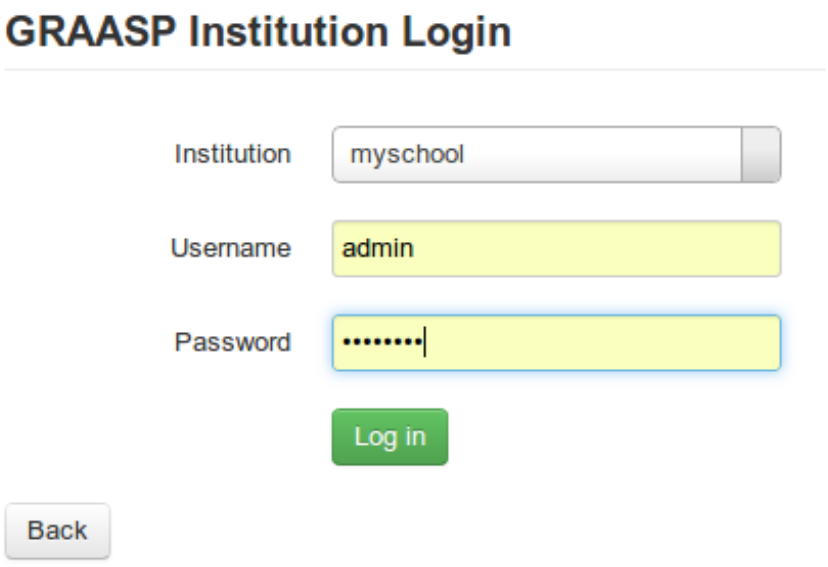


Figure 28: Log in as a school user

Once logged in, the teacher can create more users, such as other teachers. Some of them can be added as “teacher”, so they can add spaces, while others as administrators, who can create more users, request laboratories to the Labmanager administrator, etc.

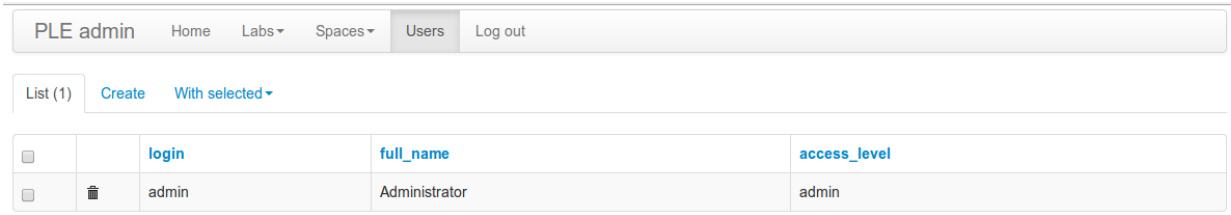


Figure 29: School users

From this moment, the Labmanager administrator can select which laboratories are available for which schools. This requires the teacher to wait for the Labmanager administrator to log in and establish those permissions. Other option is that the Labmanager administrator selects some laboratories and makes them visible for all the registered schools. This means that the school can then request what laboratories are desired, and if the Labmanager administrator has selected that the laboratory is publicly available for all the registered schools, then the permission is granted automatically. Otherwise, the school must wait until the Labmanager administrator grants access on this laboratory.

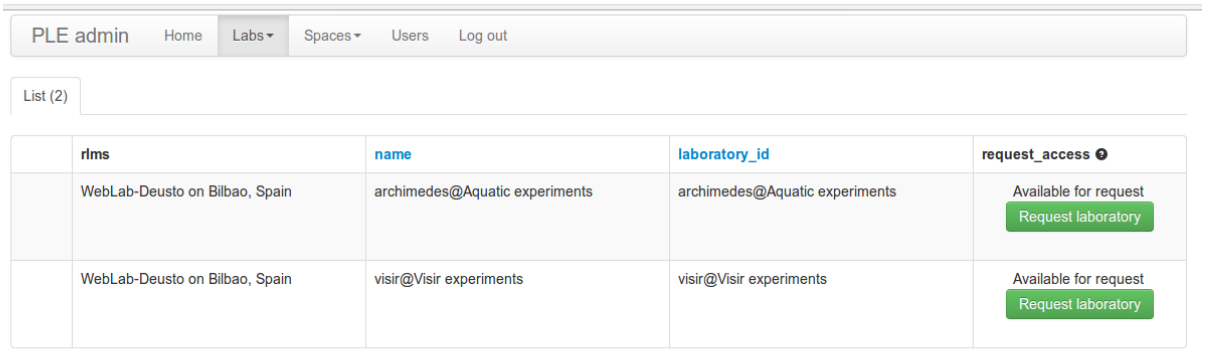


Figure 30: Laboratories available for being requested

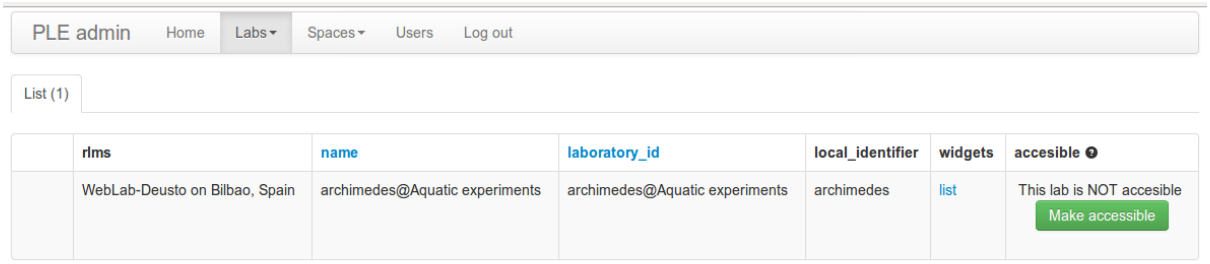


Figure 31: Laboratories registered for the school

Once the laboratories are registered, the teacher can register spaces from the ILS platform. So as to do this, the teacher only needs to copy the URL of a ILS, or of a parent space that contains all the ILS of the school, and paste it in the Labmanager:

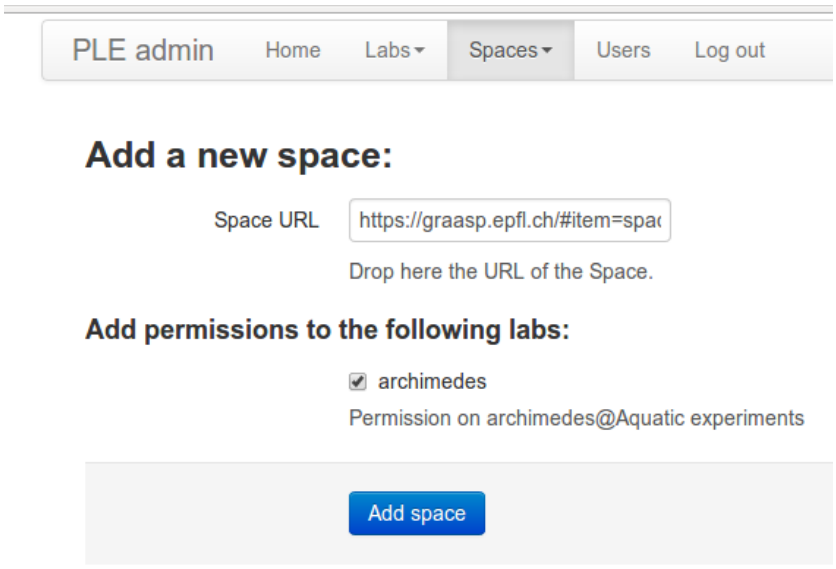


Figure 32: Register parent spaces

A list with the spaces is provided, as well as with the existing permissions, which can be changed manually during time. The most convenient form to manage these permissions is by targeting not each individual ILS, but a parent space where all the ILS are. The Labmanager will explore the space and parents whenever a request arrives to check which school the request is coming from.

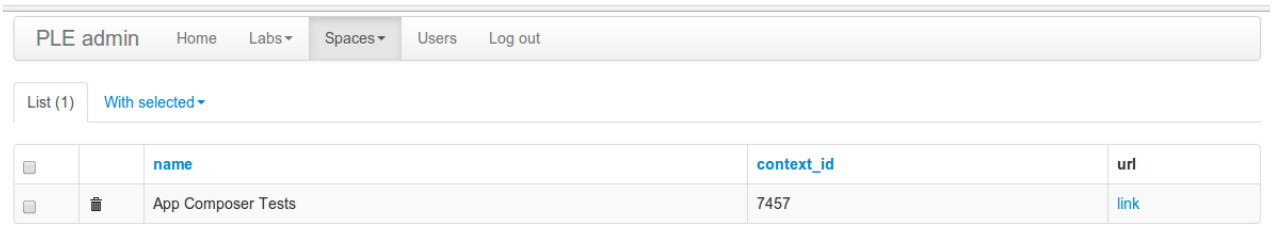


Figure 33: List of registered spaces

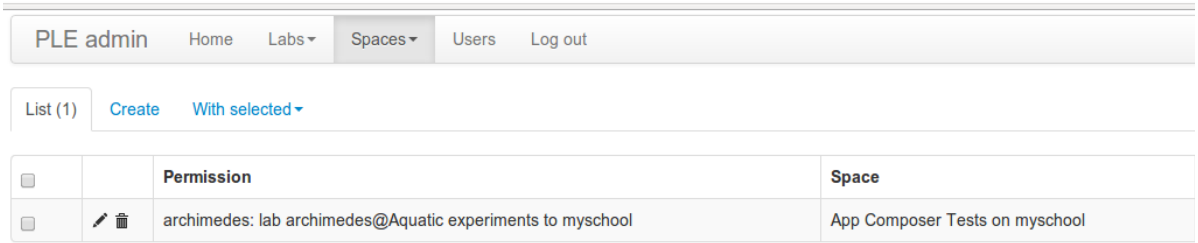


Figure 34: List of permissions on registered spaces

5.5 Appendix F: Lab Owner's Survey

In the scope of this deliverable and to improve our knowledge about the external community of lab owners and their legacy lab systems we have conducted a survey. The main focus of this survey was to learn about the community's willingness and constraints regarding lab sharing schemes envisioned in Go-Lab. Lab Owners were asked general questions about their labs, if they would be willing to share labs with the Go-Lab community and what incentives they would consider of relevance. The answers were in a rating scale from 1 (not likely) up to 6 (very likely). In total 26 responses were received. Below we show the calculated arithmetic mean and standard deviation of the received answers.

Question	**Mean**	**Std. Dev.**
Would you share your labs with the Go-Lab community?	5,19	1,23
Would you share your labs to increase their visibility?	5,00	1,52
Would you share your labs to get third parties involved to create pedagogical content using the labs?	4,88	1,21
Would you share your labs without requiring direct economic remuneration?	4.81	1,47

Table 4: Overview of survey's answers

In general, lab owners show willingness to share their labs with Go-Lab. They also agree that increasing visibility of their lab is valid incentive. The individual responses are included in the appendix section.

Furthermore other more specific technical questions were asked regarding the RLMS they use to deploy the lab (if any), scheduling mechanism and concurrency lab use. The answers show that several labs are managed by Smart Gateway supported RLMS (WebLab-Deusto and ISA), what facilitates their integration into Go-Lab. The detailed list with all responses can be found in the appendix section.

Link to the survey:

https://docs.google.com/forms/d/1cKP1hguKNdbKTeydTAo9teOf_H-dZIq6Yi9YIuFC3Fw/viewform

5.6 Appendix G: Ongoing Smart Gateway Work

In order to gather feedback from external experts on the draft specifications for Smart Device and Smart Gateway a workshop for lab owner's was organized in Madrid from June 4 - 6, 2014. During this meeting the Smart Device and Smart Gateway concepts were introduced to the external experts who provided valuable comments and suggestions. The experts who attended the meeting were representatives from the following projects/initiatives:

Labicom (https://labicom.net)- A Web-based Remote Lab Management System to host and share remote laboratories. It separates the laboratory server from the management part and focus in providing solutions implemented in LabVIEW to interface complex equipment. [4]

RECLab /e-Lab (http://www.reclab.pt/bin/view/REC/)**-** REC Lab is a software framework that enables the control & execution of real scientific experiments over the internet. [5]. It is focused on a Java Framework to deliver remote experimentation.

RexLab (http://rexlab.net/) **-** A project that aims at delivering simple remote experiments to secondary school students in Brazil [6]. At the client-side their experiments are HTML5 based. The lab systems uses not scheduling or booking mechanism whatsoever.

RemLabNet - A remote lab management system under development [7]. Its client applications are mostly Java-based but work is being carried out to migrate the implementations to HTML5/JavaScript.

During this workshop the external experts were given the opportunity to present their systems and discussed on what would be the most suitable approach to integrate their labs into Go-Lab. The general consensus was that the Smart Gateway would be the better option since rewriting their servers and clients to be compliant with the Smart Device specification would require a lot of effort.

5.7 Appendix H: Management features of the Smart Gateway

In the following screenshots, it will be shown how the Labmanager administrator can add laboratories through the plug-in system. In the administration panel, there is an option called “ReLMS Management”, where all the instances of the supported laboratories (through plug-ins) are listed

It is important to note that one plug-in can be instantiated multiple times. For example, both iLab and WebLab-Deusto are widely used in several countries by different institutions. With a single plug-in (e.g., the WebLab-Deusto plug-in), it is possible to create multiple instances representing different WebLab-Deusto servers (e.g., one in Spain and other in Slovakia).

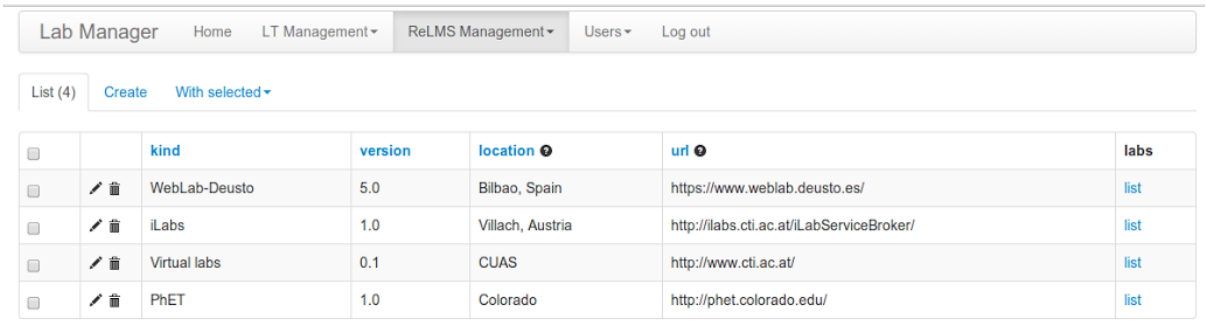


Figure 35: List of plug-in instances

When clicking on the Create button, it is possible to register a new instance of a plug-in. In the example below, WebLab-Deusto is used. The first argument (“kind”) lists all the available plug-ins, and the “location” and “url” arguments are provided by default by gateway4labs. The rest of the arguments (Login, Password, Base URL and Mappings) are retrieved from the plug-in, and will differ from plug-in to plug-in.

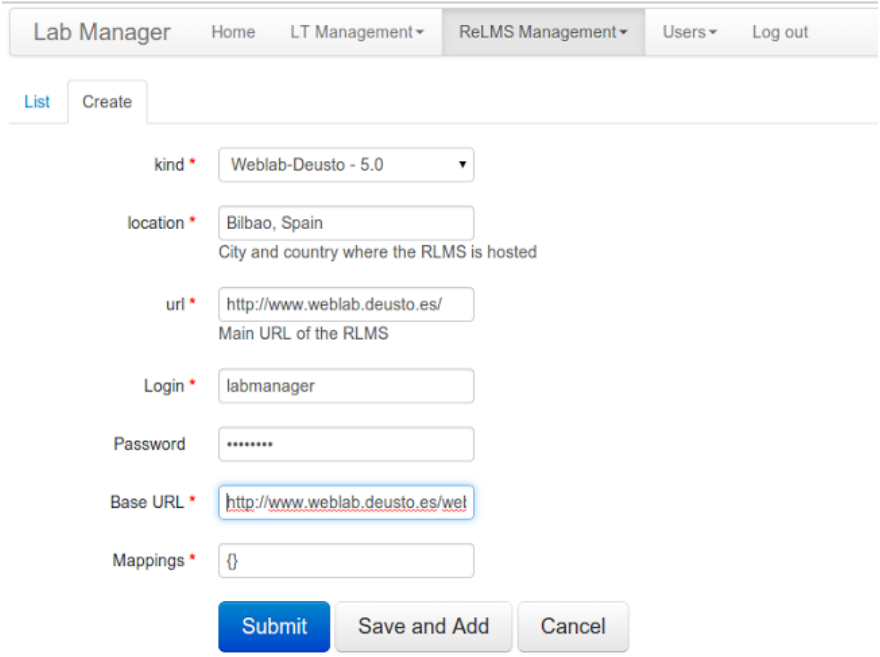


Figure 36: Example of registration of a RLMS

Once the plug-in is instantiated, it is possible to list what laboratories are provided for that plug-in with that configuration. The iLab or WebLab-Deusto plug-ins, for example, will provide different lists depending on which servers are being contacted and with what permissions. This feature calls the method “list laboratories” of the plug-in. The list is displayed and the administrator must select which laboratories wants to register in the Labmanager database. Only those registered laboratories will be later available to the users.

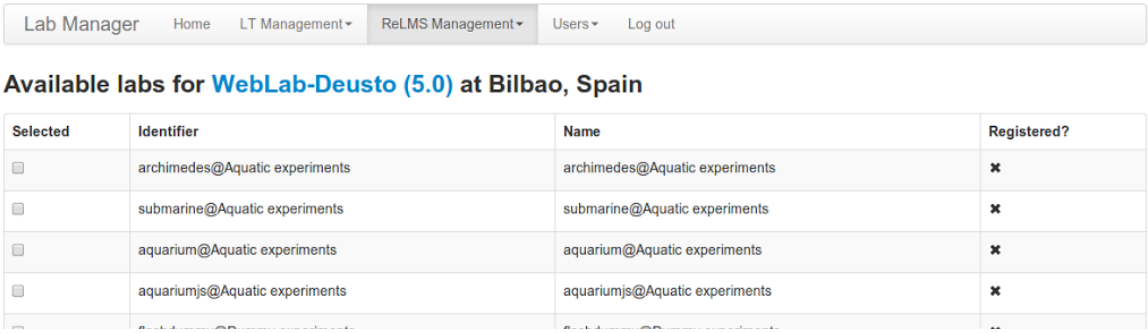


Figure 37: List of laboratories provided by one of the plug-ins

Once they are registered, it is possible to define them as public laboratories. This means that anyone, without registering in the Labmanager, will be able to use the laboratory, and therefore it will be listed as public laboratory. This is the way used by every external laboratory integrated in Go-Lab. A name which will act as an identifier must be provided.



Figure 38: Making laboratories public so they can be accessed by anyone

Once done this, it is possible to log out and in the initial screen, click on the “Graasp public labs” link. Anyone, without registration, can do this and the list of registered public laboratories is provided. For each of them, the user can list the widgets available for that particular laboratory.

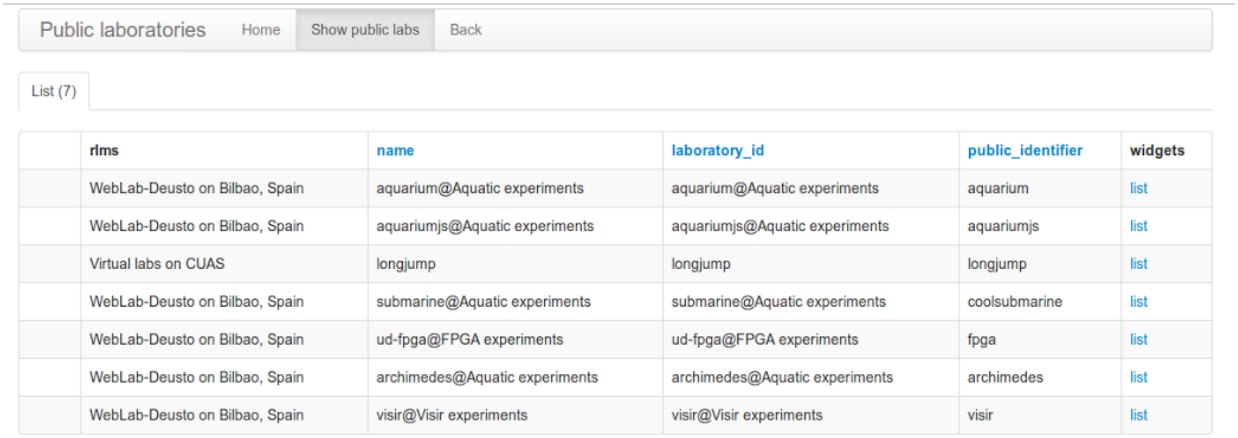


Figure 39: List of publicly available laboratories

When clicking one of these laboratories ("list" button), the list of widgets will be available, as well as a preview of the laboratory. A permanent link to the OpenSocial XML code is displayed, which can be copied and pasted in the Go-Lab portal. If the plug-in developer supported multiple widgets, these would be listed here.

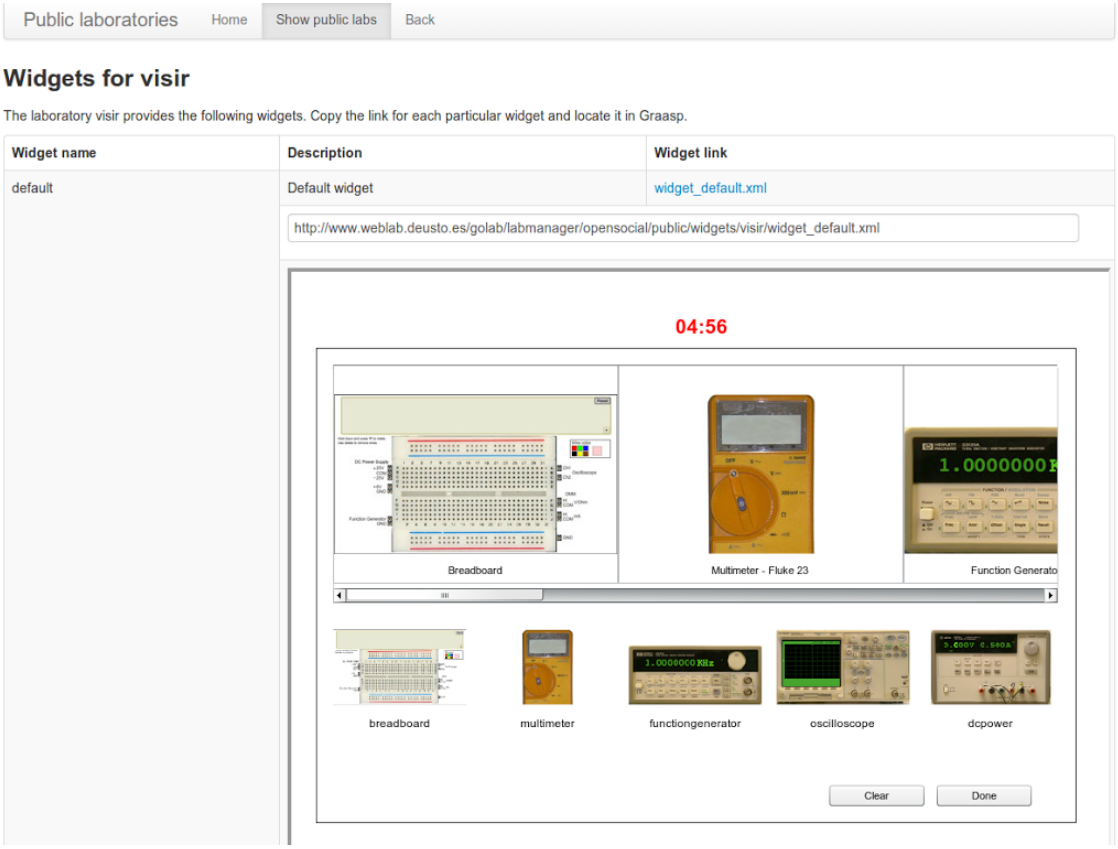


Figure 40: List of widgets of a particular laboratory, including the necessary link and a preview of the widget

6 References

[1] K. Perkins, W. Adams, M. Dubson, N. Finkelstein, S. Reid y C. &. L. R. Wieman, "Interactive simulations for teaching and learning physiscs", in *The Physics Teacher,* vol. 44, nº 1, pp. 18-23, 2006.

[2] P. Orduña, E. Sancristóbal, M. Emaldi, M. Castro y G.-Z. J. López-de-Ipiña, D. &, "Modelling Remote Laboratories integrations in e-Learning tools through Remote Laboratories federation protocols", in *Frontiers in Education Conference*, Seattle, 2012.

[3] P. Orduña, S. Botero Uribe, N. Hock Isaza, E. Sancristóbal, M. Emaldi, A. Pesquera Martin, K. De Long, P. Bailey, D. López-de-Ipiña, M. Castro y J. García-Zubia, "Generic integration of remote laboratories in learning and content management systems through federation protocols", in *Frontiers in Education*, Oklahoma, 2013.

[4] I. Titov, "Labicom. net-The on-line laboratories platform", in *Global Engineering Education Conference (EDUCON)*, Berlin, 2013.

[5] REC Lab, [Online]. Available: http://www.reclab.pt/bin/view/REC/. [Last accessed: 04 07 2014].

[6] RexLab, [Online]. Available: http://rexlab.ufsc.br/?q=en. [Last accessed: 04 07 2014].

[7] F. Schauer, M. Krbecek, P. Beno, M. Gerza, L. Palka y P. Spilakova, "REMLABNET - open remote laboratory management system for e-experiments", in *IEEE REV*, Porto, 2014.