weblablib: new approach for creating remote laboratories

Pablo Orduña[1,2], Luis Rodriguez-Gil[1], Ignacio Angulo[2], Unai Hernandez[2], Aitor Villar[1], and Javier Garcia-Zubia[2]

[1] LabsLand, Bilbao, Spain
{pablo,luis,aitor}@labsland.com,
[2] DeustoTech, Fundacion Deusto, Bilbao, Spain
{pablo.orduna,ignacio.angulo,unai.hernandez}@deusto.es

Abstract. Remote laboratories are hardware and software tools that enable students to access real equipment through the Internet. Remote Laboratory Management Systems (RLMS) are software tools developed for creating remote laboratories in an easier way, providing some of the transversal features common in most remote labs (such as authentication, authorization, scheduling platforms or administration tools), and some protocols or APIs (Application Programming Interfaces) for creating the laboratories. WebLab-Deusto is a popular open source RLMS used in different universities to create or administer their remote laboratories; and it offers two approaches for developing remote laboratories: managed (where all the communications go through WebLab-Deusto) and unmanaged (where the communications are managed by the remote lab developer). While originally the managed approach had a number of advantages over the unmanaged, nowadays, with web development technologies fastly changing and increasing productivity, it became important to provide a proper support for the unmanaged by creating a completely new framework called weblablib, developed by LabsLand and also Open Source. This article describes this framework, and the different tradeoffs that remote lab developers have to deal with when implementing a remote laboratory.

1 Introduction

An Educational Remote Laboratory is a software and hardware solution that enables students to access real equipment located in their institution, as if they were in a hands-on-lab session, using an standard web-browser. The laboratories are typically hosted in universities or research centers.

A key factor of remote laboratories is that once they are available through the Internet their usage can be scaled up and used by students of other institutions. Thus, two or more institutions can share different equipment to reduce costs by requiring less duplicated equipment: it is typically only used in certain hours of the day and in certain days of the year. Furthermore, this empowers a sharing economy where multiple providers provide access to their laboratories to each other, freely or not.

In the literature there is a wide range of remote laboratories in many fields (e.g., robotics, electronics, physics, chemistry). Software frameworks have been developed to make the development of remote laboratories more affordable (Remote Laboratory Management Systems such as WebLab-Deusto[3] [19], iLab Shared Architecture[4], RemLabNet[5] [23] or Labshare Sahara[6] [15]) and tools (e.g., gateway4labs[7] [20]) to provide integrations with educational tools (such as Moodle, Sakai or other LMS, both through ad hoc solutions and through standards such as IMS LTI) or repositories linking remote and virtual laboratories (such as Go-Lab [14, 8], LiLa[21] or iLabCentral).

Most technologies are motivated by the idea of leveraging remote laboratories for shared growth: if two universities have 3 remote laboratories each, technically, it would become possible (through sharing them) to have 6 laboratories together. However, there are a set of organizational challenges associated to this that traditionally has made it more difficult, such as reliability or consumer trust. For example, how can the lecturers of one of the two universities know that the laboratories in the other university are going to be maintained for several years? In small environments (e.g., within a funded project or among universities that have extensive collaborations), this is possible. However, at a larger scale, these issues become a problem.

With the goal of scaling the field of remote laboratories, the WebLab-Deusto team created LabsLand[8] as a spin-off company of the University of Deusto, focused on providing the necessary reliability and business background that guarantees trust for the involved partners. There are multiple institutions in several countries who are already either providing resources to build new laboratories or consuming laboratories well supported from the LabsLand network (using different RLMS and not only WebLab-Deusto).

However, for LabsLand to be successful, many remote laboratories must be available in the network covering a wide range of subjects. With this in mind, LabsLand is also developing technologies to create remote laboratories in a faster and more reliable way. While some of these tools are proprietary, weblablib is an Open Source new framework for creating remote laboratories, that internally relies on WebLab-Deusto.

The focus of this article is to describe weblablib. To this end, this article covers the basics of RLMS, and of WebLab-Deusto in particular, and explains what are the trade-offs of the design of WebLab-Deusto before the development of weblablib. Then, the article focuses on weblablib itself, describing its features and what it is good for and what it is not.

[3] http://weblab.deusto.es
[4] http://ilab.mit.edu
[5] http://www.remlabnet.eu
[6] https://remotelabs.eng.uts.edu.au
[7] http://gateway4labs.readthedocs.org
[8] https://labsland.com

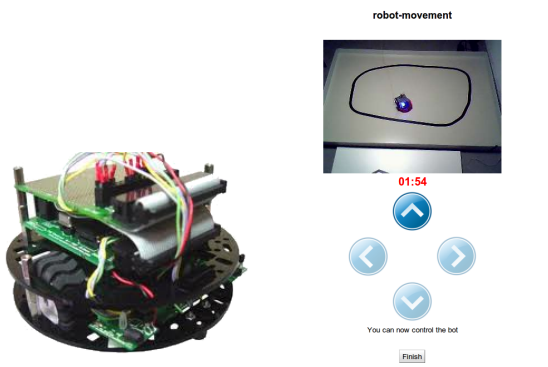

Fig. 1. Robot laboratory [6]. At the left, the mobile robot itself. At the right, the user interface once the program has been submitted.

2 Current solutions for sharing remote laboratories

This section introduces the concepts of remote laboratories, Remote Laboratory Management Systems (RLMS), remote laboratory federations and portals for sharing remote laboratories.

2.1 Remote Laboratories

A remote laboratory is a software and hardware tool that allows students to remotely access real equipment located in the university. Users access this equipment as if they were in a traditional hands-on-lab session, but through the Internet. To show a clear example, Figure 1 shows a mobile low cost robot laboratory described in [6]. Students learn to program a Microchip PIC microcontroller, and they write the code at home, compile it with the proper tools, and then submit the binary file to a real robot through the Internet. Then, students can see how the robot performs with their program through the Internet (e.g., if it follows the black line according to the submitted program, etc.) in a real environment.

In this line, there are many examples and classifications in the literature [9, 10]. Indeed, remote laboratories were born nearly two decades ago [2, 1, 13], and since then they have been adopted in multiple fields: chemistry [4, 3], physics [5, 7], electronics [11, 16], robotics, [22, 24], acustics[25], and even nuclear reactor [12].

2.2 Remote Laboratory Management Systems

Every remote laboratory manages at least a subset of the following features: authentication, authorization, scheduling users to ensure exclusive accesses - typically through a queue or calendar-based booking-, user tracking and administration tools. These features are common to most remote laboratories, and are

actually independent of the particular remote laboratory settings. For example, an authentication and queuing system is valid both for an electronics laboratory and for a chemistry laboratory.

For this reason, Remote Laboratory Management Systems (RLMSs) arose. These systems (e.g., MIT iLabs[9], WebLab-Deusto[10] or Labshare Sahara[11]) provide development toolkits for developing new remote laboratories, as well as management tools and common services (authentication, authorization, scheduling mechanisms). The key idea is that by adding a feature to a RLMS (e.g., supporting LDAP, a Learning Analytics panels [18] or similar cross-laboratory features), all the laboratories which are managed with that RLMS will support this feature automatically.

2.3 Federating Remote Laboratories

As previously stated in the introduction, a key factor of remote laboratories is that once the laboratory is available on the Internet, it can also be shared with other institutions.

To do this, there are three general approaches:

- Leave the laboratories completely open, so whoever wants to use them can use them. This may reduce the chances of providing proper Learning Analytics or supporting proper accountability mechanisms, in addition to avoiding priorities among students coming from different institutions, leading to a tradeoff between accessibility and advanced features [20].
- Share accounts between the different RLMS: if *University A* want to use laboratories of *University B*, then someone in *University A* will provide a list of usernames to *University B* and students will go to this institution using credentials in *University B*. Ideally, some federated authentication could be used to avoid providing credentials in different domains (such as Shibboleth, OAuth or similar), but it is not typically the case.
- Federate laboratories: if a RLMS supports federation, then if installed in two different institutions (e.g., *University A* and *University B*), students of *University A* will go to the RLMS of *University A* and they will transparently use laboratories in *University B*, working in a institution-to-institution basis (so *University B* does not need to know the list of students of *University A* and simply rely on an existing agreement with that university).

From the items in this list, the most advanced mechanism is the federation of remote laboratories through proper protocols oriented to market-like situations. These federation protocols have been used for fostering interoperability between RLMS [19]. These interoperable bridges between different systems can be enhanced if properties such as transitivity or federated load balance are provided [17].

[9] http://ilab.mit.edu
[10] http://weblab.deusto.es
[11] http://github.com/saharalabs

3 WebLab-Deusto software

WebLab-Deusto is an Open Source Remote Laboratory Management System that allows remote laboratory developers to develop new laboratories without dealing with most of the pains. In particular, it provides:

- An advanced queueing mechanism that supports both local and federated load balancing (e.g., if there are two copies of the same laboratory, the queue will split among the different copies automatically)
- Administration tools and dashboards for learning analytics. Administrators and teachers can see who used what and what they did in the lab.
- Sharing laboratories with other WebLab-Deusto systems in other institutions without sharing credentials or duplicating users.
- Multiple authentication mechanisms. Also, through gateway4labs, support for LTI so it can be used from Learning Management Systems.
- Native support in the LabsLand network and in the Smart Gateway of the Go-Lab project [20].

WebLab-Deusto has been designed with the following notions:

- Remote laboratory developers have very different backgrounds and skills. Some are more comfortable with one technology while others with other. For this reason, WebLab-Deusto must support multiple technologies for its adoption (instead of a single technology).
- Some remote laboratory developers might not have heavy IT skills and should not need to deal with them. It is better if WebLab-Deusto provides some abstraction level for them. However, other remote laboratory developers might have these skills and might be willing to work with the latest technologies without limitations.

For these reasons, WebLab-Deusto was designed with two development approaches in the same system. The first approach, the *managed approach*, enables remote laboratory developers to avoid dealing with networks or security systems in detail. It provides a basic model where some code on the client side (e.g., in JavaScript) send commands or messages through an API (without dealing with any network), and WebLab guarantees that those commands or messages are sent all the way to the final experiment, which will be running other code with a simple API. The developer does not need to know how many instances of this remote laboratory there are, where are they, how is https configured, etc.: they just send messages and receive messages. This protocol was implemented in many programming languages, including Python, C, C++, LabVIEW (code), .NET, Java and Node.js. So remote laboratory developers familiar with any of these systems could start implementing a remote laboratory.

The second approach, the *unmanaged approach*, provided an HTTP interface for creating remote laboratories. WebLab-Deusto would call this interface, and the remote laboratory developer would need to manage all the deployment, addressing, security, etc. In general, for a simple laboratory, it would require

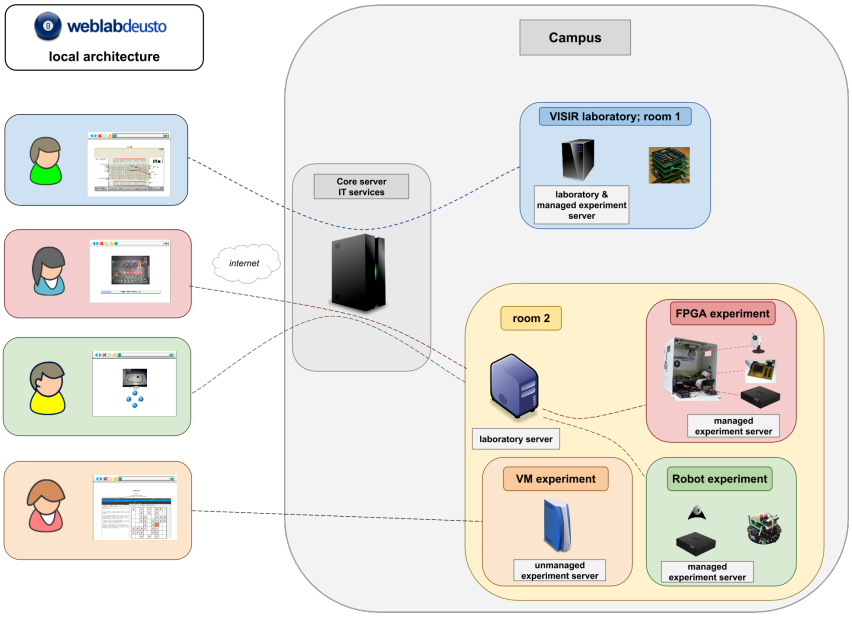

Fig. 2. WebLab-Deusto architecture

more work to use the *unmanaged approach*. However, the *unmanaged approach* provided advanced developers a customizable interface that they could extend in any web framework and use any web technology; and therefore it was intended to be used by more advanced developers.

In Figure 2, the basic local architecture of WebLab-Deusto is displayed (local of a single university; as compared with the federated architecture that explains how laboratories can be shared). Different users will go through the same WebLab-Deusto interface to access different laboratories, that can be deployed in different locations of the university. This allows remote lab developers to have one or multiple WebLab-Deusto "core servers" on the campus (audited by IT services) and different remote laboratories that can be using unexpensive technologies (such as Raspberry Pi) in different locations of the university. In the managed laboratories, all the commands/messages are sent through these "core servers" and "laboratory servers", while in the case of the unmanaged laboratories, the reservation process is done through WebLab-Deusto, but then it forwards the user to the final laboratory directly.

4 weblablib

As explained in the previous section, WebLab-Deusto supports the *managed approach* and the *unmanaged approach*. Libraries for different languages were

provided for the *managed approach*; however, this cannot be done in the *unmanaged approach* since the variety of potential web frameworks that a developer can choose are too many and are going to change too often. For this reason, the HTTP interface is public (so developers of other frameworks can implement it; and few small examples are provided for PHP and Python), but a full implementation with support for many libraries have been implemented and called weblablib. This full implementation -weblablib- relies on a concrete popular Python framework called Flask, and its ecosystem (database, networking, authentication mechanism, etc.). This way, weblablib is used for most new WebLab-Deusto laboratories, and it is oriented to make the development process faster.

4.1 Features

weblablib provides the following features:

- It is a pure Flask plug-in. Flask is a popular Open Source Python web microframework, highly extensible and easy to learn. weblablib integrates very easily and interacts with the rest of the components in an easy way. Developers can rely on the Flask ecosystem documentation to see details on how to deploy the system in production.
- Simplified model: the remote lab developer does not need to deal with credentials: the user is already authenticated in WebLab-Deusto (or Moodle or other Learning Management System through LabsLand or gateway4labs), and WebLab-Deusto tells weblablib the relevant user data. The same applies to scheduling or authorization: in production, WebLab-Deusto manages the groups, users or who has access to what copy of what laboratory. weblablib is only called whenever the user is valid.
- User information: weblablib integrates in Flask-Login so that whenever a user uses the laboratory, the application has easy access to the name of the user and a global unique identifier. It also integrates with Flask-SQLAlchemy to be able to store data in a local SQL database.
- WebSockets: weblablib integrates natively Flask-SocketIO, therefore enabling the remote lab developer to implement WebSockets in a secure way (relying on the authentication mechanism, etc.). WebSockets are a HTML5 modern protocol that enables the server to asynchronously push information to the client, which is very relevant in the remote laboratory context.
- Concurrency and task management: the system enables an easy interface for launching background tasks (such as programming a device) that have access to the rest of the features (user identification, etc.) but can be processed in batch.
- Internationalization (i18n): the system already understand what language (English, Spanish...) WebLab-Deusto requests and supports on Flask-Babel to manage the internationalization of the remote lab.

The framework also supports its own debugging system, and developers do not need to start or configure a WebLab-Deusto system for development. This way, the development process is lightweight and fast, and when the laboratory is ready, it can be configured in a WebLab-Deusto in production.

4.2 Examples

```python
from flask import Flask, url_for
from weblablib import WebLab, weblab_user, requires_active

app = Flask(__name__)

weblab = WebLab(app)

@weblab.on_start
def on_start(client_data, server_data):
    # ...
    print("Starting user")

@weblab.on_dispose
def on_dispose():
    # ...
    print("Ending user")

@weblab.initial_url
def initial_url():
    return url_for('index')

@app.route('/')
@requires_active
def index():
    return "Hello, {}".format(weblab_user.username)
```

Fig. 3. weblablib sample lab

The full documentation[12] provides examples on how to use weblablib. Figure 3 contains a very simple code using weblablib. In this example code, the developer establishes what functions should be called when the user session starts (where tasks for preparing the session should be run) and when the user session ends (e.g., time finishes for the user and it needs to clean resources -such as stopping motors if running- or make automated checks so the next user will be in the initial state again). It also shows how it is possible to use standard Flask web methods (*@app.route('/')*), but supporting authentication mechanisms (*@requires_active*) that makes that if someone enteres in that website it will fail unless the student has been redirected from WebLab-Deusto and WebLab-Deusto using the unmanaged HTTP protocol has sent information about the student. Also, there are global variables such as *@weblab_user.username* that provides data about the current user each time it is called in a simple way.

In Figure 4, it is shown how tasks can be defined. By adding *@weblab.task* it becomes possible to define that a function can be called either as usual or in

[12] https://docs.labsland.com/weblablib

```python
@weblab.task()
def program_device(contents):
    """ Programs a device. Typically takes 5-10 seconds """
    if weblab_user.time_left < 10:
        raise Exception("Error: user does not have "
                        "enough time to run it!")

    arduino.program("my_file.bin") # In this case
    return len(contents)
```

```python
# This other code runs it in a different
# process
task = program_device.delay(code)

# The following is a string that you can store in
# Flask session or in weblab_user.data
task.task_id

# a string 'submitted', 'running' or 'failed'/'done' if finished.
task.status

task.submitted  # bool: not yet started by a worker
task.running    # bool: started by a worker, not yet finished
task.done       # bool: finished successfully
task.failed     # bool: finished with error
task.finished   # task.failed or task.done

# These two attributes are None while 'submitted' or 'running'
task.result # the result of the function
task.error # the exception data, if an exception happened

# Join operations
task.join(timeout=5, error_on_timeout=False) # wait 5 seconds
```

Fig. 4. weblablib tasks usage

background, returning an object that defines if the task is still running, what is its identifier, status, or be able to wait for it if necessary. By default, if a task is long, weblablib will wait for it to finish during the clean resources period. The example is very simple so it does not show the details about how it also supports communication with the task, so it is possible to request the task to stop, or exchange data about its status, and how inside the task, the developer has access to all the information about the current user (e.g., username, full name, language, etc.).

In Figure 5, it is displayed how WebSockets can be used with weblablib. It shows that there are certain methods such as *@socket_requires_active* which have been adapted to guarantee that the WebSocket can only be opened by valid users with a valid scheduling slot; both on connection and when emitting any data. This is important so once the user session is over, weblablib already provides a security mechanism by default to avoid the client to push information to the hardware.

The code in Figure 6 shows how the system also supports a native integration with databases using the Flask-SQLAlchemy library. The global *weblab_user* object supports the concept of calling its *user* object to obtain internally access

```python
from weblablib import socket_requires_active

@socketio.on('connect', namespace='/mylab')
@socket_requires_active
def connect_handler():
    emit('board-status', hardware_status(), namespace='/mylab')

@socketio.on('lights', namespace='/mylab')
@socket_requires_active
def lights_event(data):
    switch_light(data['number'] - 1, data['state'])
    emit('board-status', hardware_status(), namespace='/mylab')
```

Fig. 5. weblablib WebSocket usage

```python
# Using Flask-SQLAlchemy ( http://flask-sqlalchemy.pocoo.org/ )
from .models import LabUser

@weblab.user_loader
def load_user(username_unique):
    return LabUser.query.filter_by(username_unique=username_unique)

 @app.route('/files')
 @requires_active
 def files():
     user_folder = weblab_user.user.folder
     return jsonify(files=os.listdir(user_folder))
```

Fig. 6. weblablib database usage

to an object retrieved from the database, that could include information such as some folder with certain files or so.

The complete structure of this API is available in the weblablib documentation.

5 Conclusions and future work

Developing a remote laboratory is a complicated task, and it requires an important effort on the programming side. WebLab-Deusto is focused on delivering different approaches for different profiles of remote laboratory developers. One of them is the managed approach, with multiple APIs for different programming languages. The other is the unmanaged approach, that is more complicated but better for experienced developers. This one supports an HTTP interface, but so as to provide an easier to adopt toolkit, it provides weblablib as an open source framework for a particular technology (Flask for Python and its ecosystem), making the effort in that framework considerably simpler.

References

1. Aktan, B., Bohus, C., Crowl, L., Shor, M.: Distance learning applied to control engineering laboratories. Education, IEEE Transactions on 39(3), 320–326 (1996)

2. Carisa, B., Burain, A., Molly H, S., Lawrence, C.: Running control engineering experiments over the internet (1995)
3. Cedazo, R., Sanchez, F., Sebastian, J., Martínez, A., Pinazo, A., Barros, B., Read, T.: Ciclope chemical: a remote laboratory to control a spectrograph. Advances in Control Education–ACE 6 (2006)
4. Coble, A., Smallbone, A., Bhave, A., Watson, R., Braumann, A., Kraft, M.: Delivering authentic experiences for engineering students and professionals through e-labs. In: Education Engineering (EDUCON), 2010 IEEE. pp. 1085–1090. IEEE (2010)
5. Del Alamo, J., Brooks, L., McLean, C., Hardison, J., Mishuris, G., Chang, V., Hui, L.: The mit microelectronics weblab: A web-enabled remote laboratory for microelectronic device characterization. In: World Congress on Networked Learning in a Global Environment, Berlin (Germany) (2002)
6. Dziabenko, O., García-Zubia, J., Angulo, I.: Time to play with a microcontroller managed mobile bot. In: Global Engineering Education Conference (EDUCON), 2012 IEEE. pp. 1–5. IEEE (2012)
7. Gillet, D., Latchman, H., Salzmann, C., Crisalle, O.: Hands-on laboratory experiments in flexible and distance learning. Journal of Engineering Education 90(2), 187–191 (2001)
8. Gillet, D., de Jong, T., Sotirou, S., Salzmann, C.: Personalised learning spaces and federated online labs for stem education at school. In: Global Engineering Education Conference (EDUCON), 2013 IEEE. pp. 769–773. IEEE (2013)
9. Gomes, L., Bogosyan, S.: Current trends in remote laboratories. Industrial Electronics, IEEE Transactions on 56(12), 4744–4756 (2009)
10. Gravier, C., Fayolle, J., Bayard, B., Ates, M., Lardon, J.: State of the art about remote laboratories paradigms-foundations of ongoing mutations. iJOE 4(1) (2008)
11. Gustavsson, I., Zackrisson, J., Håkansson, L., Claesson, I., Lagö, T.: The visir project–an open source software initiative for distributed online laboratories. In: Proceedings of the REV 2007 Conference, Porto, Portugal (2007)
12. Hardison, J., DeLong, K., Bailey, P., Harward, V.: Deploying interactive remote labs using the ilab shared architecture. In: Frontiers in Education Conference, 2008. FIE 2008. 38th Annual. pp. S2A–1. IEEE (2008)
13. Henry, J.: Running laboratory experiments via the world wide web. In: ASEE Annual Conference (1996)
14. de Jong, T., Linn, M.C., Zacharia, Z.C.: Physical and virtual laboratories in science and engineering education. Science 340(6130), 305–308 (2013)
15. Lowe, D., Machet, T., Kostulski, T.: Uts remote labs, labshare, and the sahara architecture. Using Remote Labs in Education: Two Little Ducks in Remote Experimentation p. 403 (2012)
16. Nedic, Z., Machotka, J., Nafalski, A.: Remote laboratory netlab for effective interaction with real equipment over the internet. In: Human System Interactions, 2008 Conference on. pp. 846–851. IEEE (2008)
17. Orduña, P.: Transitive and Scalable Federation Model for Remote Laboratories. Ph.D. thesis, Universidad de Deusto, Bilbao, Spain (May 2013), `https://morelab.deusto.es/people/members/pablo-orduna/phd_dissertation/`
18. Orduña, P., Almeida, A., Ros, S., Lpez-de Ipiña, D., García-Zubia, J.: Leveraging Non-explicit Social Communities for Learning Analytics in Mobile Remote Laboratories. Journal of Universal Computer Science 20(15), 2043–2053 (Dec 2014), 00000

19. Orduña, P., Bailey, P., DeLong, K., López-de Ipiña, D., García-Zubia, J.: Towards federated interoperable bridges for sharing educational remote laboratories. Computers in Human Behavior 30, 389–395 (Jan 2014), http://www.sciencedirect.com/science/article/pii/S0747563213001416
20. Orduña, P., Garbi Zutin, D., Govaerts, S., Lequerica Zorrozua, I., Bailey, P.H., Sancristobal, E., Salzmann, C., Rodriguez-Gil, L., DeLong, K., Gillet, D., et al.: An extensible architecture for the integration of remote and virtual laboratories in public learning tools. Tecnologias del Aprendizaje, IEEE Revista Iberoamericana de 10(4), 223–233 (2015)
21. Richter, T., Boehringer, D., Jeschke, S.: Lila: A european project on networked experiments. Automation, Communication and Cybernetics in Science and Engineering 2009/2010 pp. 307–317 (2011)
22. Safaric, R., Truntič, M., Hercog, D., Pačnik, G.: Control and robotics remote laboratory for engineering education. International Journal of Online Engineering (iJOE) 1(1) (2005)
23. Schauer, F., Krbecek, M., Beno, P., Gerza, M., Palka, L., Spilakov, P., Tkac, L.: Remlabnet iii – federated remote laboratory management system for university and secondary schools. In: 2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV). pp. 238–241. IEEE (2016)
24. Torres, F., Candelas, F., Puente, S., Pomares, J., Gil, P., Ortiz, F.: Experiences with virtual environment and remote laboratory for teaching and learning robotics at the university of alicante. International Journal of Engineering Education 22(4), 766–776 (2006)
25. Zappatore, M., Longo, A., Bochicchio, M.A.: Enabling mool in acoustics by mobile crowd-sensing paradigm. In: 2016 IEEE Global Engineering Education Conference (EDUCON). pp. 733–740. IEEE (2016)