Reconfigurable Web-Interface Remote Lab for Instrumentation and Electronic Learning

https://doi.org/10.3991/ijoe.v16i14.16951

Jose María Sierra-Fernández (✉), Olivia Florencias-Oliveros,
Manuel Jesús Espinosa-Gavira, José Carlos Palomares-Salas,
Agustín Agüera-Pérez, Juan José González-de-la-Rosa
Universidad de Cádiz, Algeciras, Spain
josemaria.sierra@uca.es

Abstract—Lab sessions in Engineering education are designed to reinforce theoretical concepts. However, there is usually not enough time to reinforce all of them. Remote and virtual labs give students more time to reinforce those concepts. In particular, with remote labs, this can be done interacting with real lab instruments and specific configurations. This work proposes a flexible configuration for Remote Lab Sessions, based on some of 2019 most popular programming languages (Python and JavaScript). This configuration needs minimal network privileges, it is easy to scale and reconfigure. Its structure is based on a unique Reception-Server (which hosts User database, and Time Shift Manager, it is accessible from The Internet, and connects Users with Instruments-Servers) and some Instrument-Servers (which manage hardware connection and host experiences). Users always connect to the Reception-Server, and book a shift for an experience. During the time range associate to that shift, User is internally forwarded to Instrument-Server associated with the selected experience, so User is still connected to the Reception-Serer. In this way, Reception-Server acts as a firewall, protecting Instrument-Servers, which never are open to The Internet. A triple evaluation system is implemented, User session logging with auto-evaluation (objectives accomplished), a knowledge test and an interaction survey. An example experience is implemented, controlling a DC source using Standard Commands for Programmable Instruments.

Keywords—Remote lab, Web Interface, Flex design, minimal network privileges, Scalable

1 Introduction

In Engineering learning, Lab sessions act as a reinforcement of theoretical concepts, taking a really important role in student education. However, there is usually not enough time to put in practice all theoretical concepts, due to time limitations to access to physical lab. In this line, more experiences can be offered, expanding lab access time to 24/7, this is Remote Labs.

Sometimes, Remote Lab concept is put together with Virtual Lab (a simulated lab). When these two options want to be compared, few concepts should be considered, as stated in [1]. There, cost and effect over learning are studied. Conclusions of that work pustule that the increase of the cost of Remote Lab is compensated by the improvement in the learning process, due to students are more implicated when they feel that they are interacting with real equipment.

Remote labs allow Students to access expensive lab Instruments, and specific configuration, any time, in order to put in practice concepts studied in theoretical lessons. Remote access to labs instruments is gaining attention in order to grant access to students which cannot be there (medical problems, case of force majeure, or other excused absence). Nowadays, at 2020 globally outbreak of COVID-19 is one of those case of force majeure, which has turn into remote as more as possible activities.

First option for Remote Labs is to use a commercial equipment, as VISIR, as stated in [2][3], which is a matrix of connections and components, where student can change connections. In other situations, Remote labs are mixed with virtual labs, an example is Easy Java Simulations (EJS) as stated in [4]. This system, designed for simulated experiences (virtual labs), can be connected with hardware (oscilloscopes, engines, sensors, etc.) to perform an interactive experience.

There are too non-commercial systems in Remote Labs, e.g., as stated in [5] [6], where Labicom, a completely new system was designed. Server, client, solutions, all ad-hoc solutions are designed.

Even a compact solution has been developed, all integrated in a Raspberry Pi, as stated in [7], hosting a webserver, with an Arduino as sensor interface.

Other implementations, as the stated in [8], proposed a complete system for flexible Remote Lab testing, based in Lab Server and Web Server. It is a good approach for deallocated systems, due to all communications are done throw The Internet. However, it requires for each Lab Server, external access, if it is in a corporative network, it is a complex, or impossible configuration.

Even there are different approaches for Remote Labs, centred in collaborative work, as stated in [9]. This work is centred in a system where few students can work together in the same experience.

With a general study of the work, as stated in [10], it can be observed that these solutions are only a part of the problem, and it is demonstrated that a system structure stable, flexible and scalable is needed.

However, all reviewed Remote Labs systems are not enough flexible, due to hardware compatibility are limited, or web interface are not enough intuitive. The aim of this work is to design a structure and function of a Remote Lab system for engineering learning, flexible, in order to be able to connect any Instrument (with computer interface), scalable (in order to add nodes) and with enough security level.

With that objective, servers are designed using the programming language Python, one of the most popular programming languages with several libraries, which allows interacting with almost any device. In addition, it can be set up as a web server, providing sufficient level of security and simplifying and ensuring data storage. The user interface is designed in HTML and JavaScript, introducing a fluid and asynchronous experience.

This work is structured as follows: In section II general lines and objectives of this project are given; then in section III the system structure is explained, followed in IV, where the relation with learning and examples are explained, the evaluation methods for learning outcomes are explained in section V, finally conclusions are given in section VI.

2 System Objective

The aim of this work is to present a complete Remote Lab solution which can implement experiences related with engineering learning. This system must manage user login, and user data in a safe way, and must have a time shift manager, in order to create time slots for remote experiences and allow users to book them. In addition, networks requirements needed for the implementation in a complex network must be minimal, due to University networks are usually really complex.

This proposal allows students the interaction with Lab Equipment from anywhere in assigned time slots. In this way, Lab Instruments can be used beyond the Lab sessions hours.

Theoretical concepts to be tested in Remote Lab changes during the semester, for that reason different physical configurations could be needed. Few experiences can be offered together, but some physical configurations limit the experiences to be offered in that Instrument-Server. So, once the Instrument-Server is configured for the experiences, time slots are defined in Time Shift Manager, which would be booked by users.

During a time, slot booked, Reception-Server forward User to Instrument-Server associated to the selected experience, using a SSH tunnel. Experience is takes place in the Instrument-Server.

When time slot is almost over, only if following time slot is available, it is offered to the user which is doing the experience, for be booked. When time slot booked expires, user is returned to Reception-Server.

User cannot book another time slot until has completed a booked time slot (not multiple booked are allowed. Even if user does not enter in the Remote Lab experience in a booked time slot, a penalty is applied, blocking the time slots booking for 1 day (by default). This is done for a fair use of the time slots.

Time slots are calculated have enough time to do all steps of experiences (Reading instructions), so take more time is only for free use of instruments, under the experience configuration.

Once time slot is over, in Reception-Server, evaluation activities are available for a day (by default) to be done.

Each Instrument-Server could have a different set of Instruments, hardware and configurations, and each one could implement a different set of experiences. In any moment, hardware is grouped by useful for sets of experiences (in electronics, DC response requires different hardware than AC response).

These experiences, for example, include the use and limits of experimental instruments, the interface for instrument controls, the use of instruments for measuring electronic experiences, etc.

3 System Structure

The proposed system structure is explained as follows. Each Instrument-Sever has two parts, a Web-Server and an Instrument-Connector prepared for all connected hardware. Back-end of the Web-Server and connector are implanted in Python, and work together.

In most situations, Instrument-Connector would be a set of functions to link instrument's communication systems (GPIB, USB, Ethernet, etc.) with Web-Server. However, in some situations, it would need to process information, or establish a more complex communication with the devices, with a simple interface with the Web-Server (e.g., use an Arduino to measure temperature and humidity, take information from the serial port, and send temperature and humidity instead serial-data stream). In other words, Instrument-Connector implements all functions needed for interact with physical devices, and group interactions in simple functions, which will be called from Web-Server Back-end.

For the Web-Server, a specific Python Framework, Django with REST framework, is used. Django simplifies most of the server operations, including security issues, user login and logout, database connection and data management, etc. It implements protections against common vulnerabilities and it is open source project under active develop. The REST framework increases Django's capabilities, creating an application programming interface (API), making data server available (with a security layer) from the front-end (user interface), in a really easy way. With all this, Instrument-Connector can be easily linked to the user interface with most common web-API functions as get or post.

In order to provide beautiful, flexible and configurable front-end, a HTML design, structured with CSS is done for the basic structure. In order to allow in-frame video streaming, and variable exchange with the back-end (with instrument-connector and data storage) without the needed of refresh the site, "Asynchronous JavaScript and XML" (AJAX) is used.

JavaScript and, in particular AJAX, allow to implement a complete program in the user we browser. When this is joining with REST in the back-end, experience is half front-end half back-end, in order non injections or alterations can be done over data server or over Instrument-Connector. User interaction and functions related are implemented in the front-end (in JavaScript) and steps (values, validation, sequences, steps done, etc.) are implemented in back-end. In this way, user cannot modify variables related with the progress or grade of the experience.

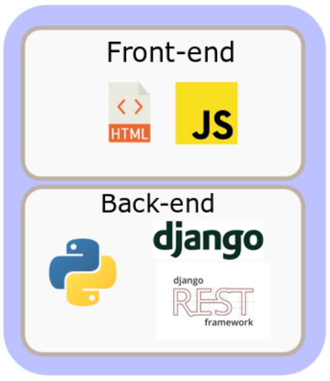


Fig. 1. Basic structure of technologies in the Remote Lab system

Instrument-Connector controls all hardware connected to the Instrument-Server, but with that, different experiences can be implemented. This implies different user interfaces, in the same Instrument-Server, and the steps for the experience.

For some experiences, same interface can be used, changing steps (e.g. DC controlled source and voltmeter, for voltage divider, amplification, diode polarization, other DC bias point, etc.).

Actually, most experiences would have a similar structure, a camera streaming, few controls and indicators, question selector and answer bow. Depending the instruments connected it would be different set of controls/indicators. In addition, help a documentation will be added.

With that, a well-designed (even partially reconfigurable) interface can be used for different experiences, changing the back-end programming of the experience.

Some scenarios or experiences may require test-boards, which work with the equipment connected to the Instrument-Server. In that situation, those experiences would only be available when those boards are connected.

In addition, if a scalable system can be designed, hardware required for create the Instrument-Server must be flexible. Actually, almost any computer nowadays can be used. It is only needed support python and have the power for the Web-Server and the Instrument-Connector. If there are not any requirement in the Instrument-Connector (privative driver which require Windows, or a complex process of data), function of Instrument-Server can be done by a Raspberry Pi.

Now, there are few different equipment, acting as Instrument-Servers, with different Instrument connected and experiences implemented, in the same local network. However, user is not in the local network, connection will be done from The Internet. It would be needed a link element among user, connecting from The Internet, and Instrument-Server, in local Network, and it is the Reception-Server.

Reception-Server is a single server, which can be reached from The Internet. It redirects all traffic to each Instrument-server, according to the time shifts management. This server hosts the User's database, the time shifts management system, the main user interface and the information related to all Instrument-Servers. However, it does not host any instrument. In other words, it makes possible the connection with the Instrument-Servers (which are in a private network), from The Internet, with a security layer. And with not special needed of network configuration in Instrument-Servers.

From the User point of view, he connects with the Reception-Server web site, and still there all the time. User create credentials in Reception-Server, (referenced to the Learning Management Systems (LMS), as Moodle, if used). Once logging, they can book an experience, for an available time slot (only one book at time, for a fair use of the system). Reception-Server will forward user to the related Instrument-Server of selected experience during the selected time slot, and during that time interval, experience can be done, in a frame inside the main web-interface. Only if next time slot is available, it is offered to be booked to User during the experience, when time slot it almost over, in order it can continue practicing with instruments. Only after User ends the experience, he can book another time slot for another experience.

The Reception-Server register the communication parameters of Instrument-Servers (IP, ports, API keys) and available items (Instruments and experiences). Over that, working parameters can be set in the Reception-Server or in Instrument-Server (Available-Time, time slots for experiences). All this information is periodically sync among Instrument Servers and the Reception-Server (each 1 min) in addition to Status. In this way, any change can be detected, even if an Instrument-Server disconnects, it can be detected and a warning can be raised.

There is implemented a fast connection method among Instrument-Server and Reception-Server, where the Instrument-Server point to the IP of the Reception-Server and make a call of an API function, giving all its information, creating the registration in the Instrument-Servers list.

With all this, there are a lot of communication among Instrument-Servers and Reception-Server. For store some of that information and other data related to the experiences, a flexible storage system is used, in this situation the database is PostgreSQL; due to its capability to store more types of data as usual databases, in particular JSON (very useful for log sessions and save experience structures). In addition to the Instrument-Servers parameters, each User interaction implies a data transfer between each Instrument-Server and the Reception-Server (login, evolution in experience, time slot book, etc.)

In addition, once the Instrument-Server has been registered in the Reception-Server, a SSH tunnel is prepared and tested. With it, the connection between the user, accessing from The Internet, and the main web interface of the Instrument-Server related to the selected experience, in the private network, is possible. In addition, the main Instrument-Server web interface is embedded in an Iframe in the Reception-Server site, so interaction with the Reception-Server is also possible. The Apache server can redirect the tunnel objectives to port 80, so the entire User experience will be at port 80 (or 443 for SSL).

The Reception-Server, as mentioned before, host systems related with system data (databases, time shift manager) and main user interface, and stablish the connection among user and Instrument-Servers.

The Instrument-Servers host experiences and have direct interaction with hardware. Time slot reservations are done throw the Reception-Server and direct user interaction with Instrument-Servers is only done during the experience. Even a very powerful programming languages tandem have been selected (CSS, HTML, JavaScript and python), they must be known for program and reconfigure Instrument-Servers. In particular Python and JavaScript are the most popular languages in 2019 as stated in [11], [12], this implies active development, support, many packages available, lower probability of end of use soon, etc.

With this structure, the Reception-Server, a single server (which can be protected with a redundant copy) controls all data flow, and any number of Instrument-Servers can be connected or disconnected (even Instrument-Servers are monitored for detect sudden disconnections, and mark them as not available). All these make a system stable, scalable, and easy to reconfigure, due to main configuration is only in the Reception-Server, and even few parameters of Instrument-Servers can be configured from the Reception-Server. User connect throw usual web-traffic port 80/443, in a single endpoint, that makes simple the network configuration requirements, and connections to Instrument-Servers are done throw a SSH tunnel by Reception-Server. System structure is represented in Figure 2.

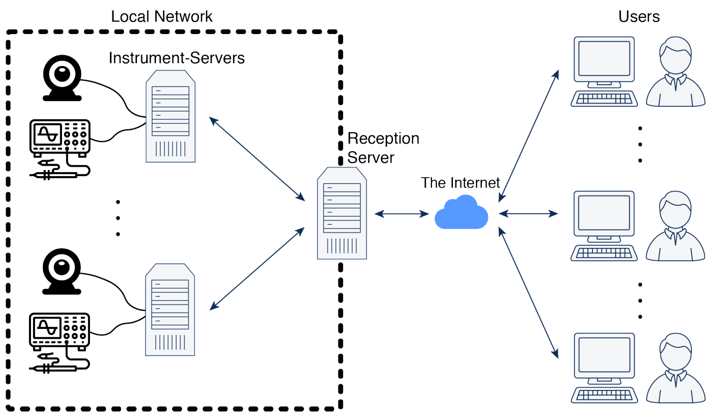


Fig. 2. System global structure and data flow.

The system is not defined as open source, non- proprietary, or other license free tags, because Instrument-Connector could require proprietary libraries, not for commercial use, but most of JavaScript libraries (those used), are GNU, GPL or MIT, and DJANGO has a Creative Commons license and REST is authorized to use it with or without modifications by the copyright owners. Therefore, the core system has no blocking or license cost.

4 Integration in Engineering Sessions

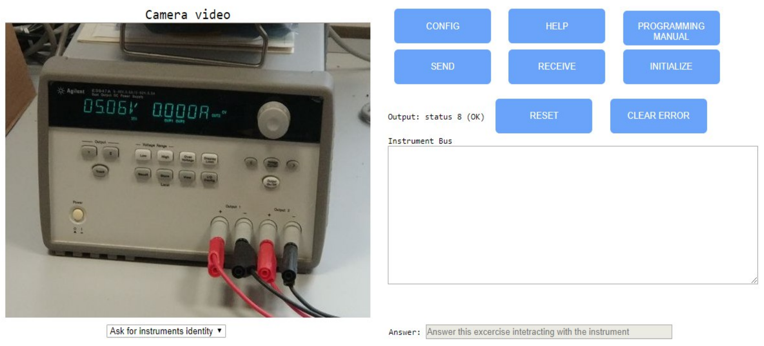
In engineering, as in other technical university learnings, there are many concepts explained in lessons, that cannot be experienced in lab, due to lack of time.

Experiences, in the same subject, are grouped by thematic, and in most of situations, those groups use the same Instruments, or share most of them. E.g. for DC bias point study, a dc controllable source and a voltmeter are used (same instruments can be used for experience voltage divider, beta of bipolar junction transistor, zero frequency amplification of an amplifier, limits of an Operational Amplifier, diode response, etc.) even the same can be used for study 1st order response system (charge and discharge of capacitor or inductance). Including a function generator and an Oscilloscope (or DAQ-Card), system can be studied in an AC bias point, in a frequency different to zero (or experiences as amplification, rectification, upper or lower cut-off frequency in a filter, bandwidth, etc).

Instrument-Connector is configured for control all instruments connected to the Instrument-Server, in an easy way from the Python back-end. It implements any needed action for set to or read from the Instrument (or the hardware) data or configuration. All actions over Hardware are done throw Instrument-Connector, e.g. for de DC source, set range, set voltage, activate/deactivate outputs, measure current, measure voltage (to check if there are not problems).

User experiences are designed using Instrument-Connector, so in this point, we can focus in the experience, and interaction with hardware are a single line (and all needed steps are in Instrument-Connector). Steps for the experience (its questions and actions needed), are designed, and all data which will be saved are marked (by default interactions, number of attempts for each step, the time needed to solve each step, steps passed, and number of steps done when experience ends). This information is done for an auto-evaluation (by default, this evaluation is related with the number of steps done when experience finished, but it accepts evaluation by number of attempts, even by time taken in the experience, but this does not encourage to try), and it is saved for track student evolution.

As example, it has been implemented a DC power supply control using Standard Commands for Programmable Instruments (SCPI) (the base of the VISA protocol), connected with the General-Purpose Instrumentation Bus (GPIB). Communication with many lab instruments ends in a VISA commands exchange, throw different busses. This experience gives to students the chance to experience and understand this communication. In any moment, a camera is streaming the front panel of the Instrument, in order to students can see the consequences of the instructions introduced. In Figure 3 it can be seen the main interaction page of this experience.



Output: status 8 (OK)

Answer: Answer this excercise intetracting with the instrument

Fig. 3. Main web interface GPIB SCPI experience

The centre of this experience is the camera, streaming the front panel of the Instrument, in this situation, Agilent E3646A DC power supply. Under camera streaming box, experience step can be selected, which must be answered in the "answer" box, in the lower right position, or interacting in the "Instrument Bus", with the proper command. In addition, there are few buttons, in the upper right area. "HELP" opens in a new window a manual about the experience interaction, and experience guide. "PROGRAMING MANUAL" open in a new window Instrument documentation, with all needed SCPI commands and its complete explanation. "Initialize" realize all steps for start communication of GPIB with the Instrument, according to "CONFIG", where connection parameters are set. During the experience, some instructions generate errors, which can be cleared with "CLEAR ERRORS" and for start from a referenced point in the DC source, with the base configuration, we have the "RESET" button. "SEND" transmit to the Instrument the instructions written in the "Instrument Bus" text box, and "RECEIVE" read the information from the Instrument, and write it in the "Instrument Bus" text box. On each communication, status byte is tested and shown, for detect communication errors (e.g. if Instrument cannot be reached)

When a proper answer has been done for a step (with the proper interaction with the Instrument or with the proper answer in the answer box), the background for it turn into green in the combo-box. When all steps are green, experience finished with a pop-up, but Instrument-Server still connected during reserved time-slot.

As part of the experience, evaluations are implemented in Instrument-Server, but User performs them in the Reception-Server, in order to not to require the forward to the Instrument-Server. In this way, they can be ended, out of the booked time slot. So, in the moment that evaluation activities are needed (time slot ends or exercise are complete, the one which occurs earlier), they are sent to the Reception-Server. There, User can take them during a specified time interval, by default 1 day.

If user have been registered link to a LMS system, when experience and evaluation activities have been done, data is prepared to be uploaded to the LMS, and used as evaluation activities.

5 Learning Results

A Remote Lab experience is not easy to evaluate for a continuous improvement. In this work, the evaluation focuses on three aspects of the User: interaction, experience and learning.

A first evaluation step is done during the User interaction with the Remote Lab. Goals are set (steps) and parameters are selected (number of interactions in each step, number of tries, steps done at time end, etc.). With this an auto evaluation can be done over the user interaction, and a grade can be obtained over the interaction. Moreover, these logs can be used to evaluate the pedagogical level of the experience (if non user ends the experience, or if all of them end experience in half proposed time, it is not well time calibrated, and must be redesigned).

It is important to know if there are issues in user interaction, and those problems create difficulties in experience in the Remote Lab. In order to obtain this information, a quick survey is done asking user for camera quality, ease of use of the experience, the amount of time for slot assignment, and their overall evaluation of the experience from 1 (very bad) to 5 (perfect).

Finally, a short knowledge test is done, related with the knowledge experienced. If experience has been useful, this test would be really easy.

The evaluation of student learning is carried out in two phases. First with its evolution in experience in Instrument-Server and finally with the survey and the test in Reception-Server.

6 Conclusion

Proposed Remote Lab system is flexible, due to Instrument-Connectors can be designed to communicate with almost any device (Instruments, boards, controllers, interfaces, etc). It is easy to scale, due to Instrument-Servers can be added or removed, only registering them in the Reception-Server. Even easy to reconfigure, due to many of the configuration is done in the Reception-Server.

This system, can implement any kind of experience, including any kind of hardware, due to there are not practical limitations in the core of the system. Depending on the interaction required, it would need a more complex programming, but there is not a limitation.

Only one equipment of the system requires special network privileges, allowing connection from The Internet to Instrument-Servers, which are regular equipment in the private network, which make easier the implementation of those equipment. Even the single computer which require special network configuration, the Reception-Server, only requires external access, not more special configuration, and only common web ports 80/443.

Additionally, a multiple evaluation system is designed for the student learning. Its actions during the experience are evaluated, and used for revise the experience itself, and after the experience, a knowledge test is done, in order to conform that theoretical concepts experienced have been acquired. In addition, a quick survey is done over the remote experience itself, in order to fix problems in camera, interaction, etc.

7 Acknowledgement

The Spanish Ministry of Economy, Industry and Competitiveness [Grant No. supported this work TEC2016-77632-C3-3-R]. The authors would like to thank the Andalusian Government for funding the Research Unit PAIDI-TIC-168 in Computational Instrumentation and Industrial Electronics (ICEI) and the University of Cádiz.

8 References

[1] K. Jona, R. Roque, J. Skolnik, D. Uttal, and D. Rapp, "Are remote labs worth the cost? Insights from a study of student perceptions of remote labs", International Journal of Online and Biomedical Engineering (iJOE), Vol 7, No 2 (2011). https://doi.org/10.3991/ijoe.v7i2.1394
[2] I. Evangelista et al., "Science education at high school: A VISIR remote lab implementation," in Proceedings of 2017 4th Experiment at International Conference: Online Experimentation, exp.at 2017, 2017, pp. 13–17.
[3] J. Garcia-Zubia et al., "Dashboard for the VISIR remote lab," in Proceedings of the 2019 5th Experiment at International Conference, exp.at 2019, 2019, pp. 42–46. https://doi.org/10.1109/expat.2019.8876527
[4] L. De La Torre, M. Guinaldo, R. Heradio, and S. Dormido, "The ball and beam system: A case study of virtual and remote lab enhancement with Moodle," IEEE Trans. Ind. Informatics, vol. 11, no. 4, pp. 934–945, Aug. 2015. https://doi.org/10.1109/tii.2015.2443721
[5] I. Titov, "Labicom.net - The on-line laboratories platform," in IEEE Global Engineering Education Conference, EDUCON, 2013, pp. 1137–1140.
[6] I. Titov, A. Glotov, Y. Andrey, and V. Petrov, "Labicom labs: Remote and virtual solid-state laser lab, RF & microwave amplifier remote and virtual lab: Interactive demonstration of Labicom labs in winter 2016," in Proceedings of 2016 13th International Conference on Remote Engineering and Virtual Instrumentation, REV 2016, 2016, pp. 336–338. https://doi.org/10.1109/rev.2016.7444496
[7] M. D. Da, "Rasberry Pi Based Remote Lab Implementation," Int. J. Sci. Eng. Res., vol. 7, no. 8, 2016.
[8] W. Farag, "An Innovative Remote-Lab Framework for Educational Experimentation", International Journal of Online and Biomedical Engineering (iJOE), Vol 13, No 02 (2017). https://doi.org/10.3991/ijoe.v13i02.6609
[9] S. Odeh, E. Ketaneh, "A Remote Engineering Lab for Collaborative Experimentation", International Journal of Online and Biomedical Engineering (iJOE), Vol 9, No 3 (2013) https://doi.org/10.3991/ijoe.v9i3.2500
[10] I. Angulo, L. Rodriguez-Gil, and J. Garcia-Zubia, "Scaling up the Lab: An Adaptable and Scalable Architecture for Embedded Systems Remote Labs," IEEE Access, vol. 6, pp. 16887–16900, Mar. 2018. https://doi.org/10.1109/access.2018.2812925
[11] "10 top Programming Languages in 2019 for Businesses." [Online]. Available: https://codeburst.io/10-top-programming-languages-in-2019-for-developers-a2921798d652.[Accessed: 11-Dec-2019].

[12] "Top 10 Programming Languages of the World – 2019 to begin with… - Geeks for Geeks." [Online]. Available: https://www.geeksforgeeks.org/top-10-programming-languages-of-the-world-2019-to-begin-with/[Accessed: 11-Dec-2019].

9 Authors

Jose María Sierra-Fernández is member of the Research Group ICEI PAIDI-TIC-168, in statistical signal processing, power quality and sensor networks. He received his Ph.D. degree in Engineering from Universidad de Cádiz in Algeciras (Spain) in 2017, and currently works as Associate Professor for the Spanish university "Universidad de Cádiz" in Electronics, in Escuela Politécnica Superior de Algeciras, 11202, Avda. Ramón Puyol s.n. Email: josemaria.sierra@uca.es

Olivia Florencias-Oliveros is member of the Research Group ICEI PAIDI-TIC-168, in statistical signal processing, power quality and sensor networks. She received his Ph.D. degree in Engineering from Universidad de Cádiz in Algeciras (Spain) in 2019, and currently works as Research associated for the Spanish university "Universidad de Cádiz" in Electronics, in Escuela Politécnica Superior de Algeciras, 11202, Avda. Ramón Puyol s.n.

Manuel Jesús Espinosa-Gavira is member of the Research Group ICEI PAIDI-TIC-168, in sensor networks, short-time prediction and environmental data. He received his Master. degree in Engineering from Universidad de Cádiz in Algeciras (Spain) in 2018, and currently he is writing his Ph.D. thesis. He actually works as Predoctoral Fellow for the Spanish university "Universidad de Cádiz" in Electronics, in Escuela Politécnica Superior de Algeciras, 11202, Avda. Ramón Puyol s.n.

José Carlos Palomares-Salas is member of the Research Group ICEI PAIDI-TIC-168, in machine learning, power quality and renewables energies. He received his Ph.D. degree in Engineering from Universidad de Cádiz in Algeciras (Spain) in 2013, and currently works as Associate Professor for the Spanish university "Universidad de Cádiz" in Electronics, in Escuela Politécnica Superior de Algeciras, 11202, Avda. Ramón Puyol s.n.

Agustín Agüera-Pérez is member of the Research Group ICEI PAIDI-TIC-168, in renewables energies, short-time prediction and environmental data. He received his Ph.D. degree in Engineering from Universidad de Cádiz in Algeciras (Spain) in 2013, and currently works as Associate Professor for the Spanish university "Universidad de Cádiz" in Electronics, in Escuela Politécnica Superior de Algeciras, 11202, Avda. Ramón Puyol s.n.

Juan José González-de-la-Rosa is member of the Research Group ICEI PAIDI-TIC-168, in statistical signal processing, power quality and instrumentation. He received his Ph.D. degree in Physics-Electronics from Universidad de Cádiz in Algeciras (Spain) in 1999, and his Full Professor status in 2018, and currently works as Full Professor for the Spanish university "Universidad de Cádiz" in Electronics, in Escuela Politécnica Superior de Algeciras, 11202, Avda. Ramón Puyol s.n.

Article submitted 2020-07-10. Resubmitted 2020-09-08. Final acceptance 2020-09-10. Final version published as submitted by the authors.