Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

ELab: A Lightweight SCADA System for Control Engineering Research and Education

Martin Kalúz * **Ľuboš Čirka** * **Miroslav Fikar** *

* *Institute of Information Engineering, Automation and Mathematics STU in Bratislava, Radlinského 9, 812 37 Bratislava, Slovakia email: {martin.kaluz, lubos.cirka, miroslav.fikar}@stuba.sk*

Abstract:
This paper presents a lightweight SCADA system *eLab* that is based on open-source and affordable hardware/software technologies. The primary purpose of *eLab* is to provide an easy means for researchers and students to perform laboratory experiments, without requirements for extensive configuration on the side of the user. The architecture of the system consists of several functional parts. I/O nodes are hardware devices that directly connect and electrically control sensors and actuators of laboratory equipment. For this purpose, an implementer can either use a dedicated MCU, such as Arduino board, a single-board computer like Raspberry Pi, or any device with UART communication capabilities. The central part of *eLab* is a SCADA master, i.e., the computer that serves all the functionalities required by a SCADA system. The SCADA software is implemented in server-side JavaScript (Node.js). The communication between I/O nodes and SCADA master is served via XBee radio modules. The master computer acts as a communication gateway between I/O nodes and other parts of the system. The gateway provides a dedicated RESTful API that is used for the front-end connection to control software or HMI. Additionally, the system uses an internal database for configuration of experiments, tags, and data sessions. The *eLab* also provides a novel integration with DCore blockchain technology so that the users can store the data either in private or public blockchain network. The use of blockchain ensures the preservation, immutability, and verifiability of measured data.

Keywords: Process Control, Education, Laboratories, SCADA, RESTful API, XBee, Microcontroller

1. INTRODUCTION

Monitoring and control of laboratory processes and devices is a critical topic in both control education and IT-driven practical research. Many different technologies are utilized to achieve interaction between user and laboratory equipment. Remote laboratories (RLs) are a typical application of network-based control systems with spatially distributed functional parts. Many RLs are based on ad-hoc hardware and software solutions that are designed for specific use cases of particular implementations. If RL or other remotely operated instrument requires a presence of industry-standard technologies, the implementers usually choose PLCs and RTUs as a direct control equipment (Golob and Bratina, 2013; Cuayo et al., 2018). In some cases, these types of laboratories use supervisory control and data acquisition (SCADA) systems for monitoring, data management, and HMI integration (Domínguez et al., 2019), usually along with an OPC technology for data storage and access (González and Calderón, 2019). RLs that utilize industrial hardware are usually more expensive than other ad-hoc solutions. In recent years, the most successful approach to RL systems development is the use of low-cost programmable devices, such as microcontroller units (MCUs). Some of the researchers also focus on the combination of industrial systems (SCADA, OPC) and low-cost devices (MCUs) (González and Calderón, 2019). In many cases, the back-end industrial hardware, whose role is to connect instruments to RL or control system, is substituted by embedded devices (Daros et al., 2015; Rojas and Barbieri, 2019). The majority of available architectures for RLs rely on the use of coordinator computers, running a set of middleware services. The coordinators mediate user-to-instrument communication, data storage, and overall system management.

The use of low-cost IoT approaches in the development of education-oriented SCADA systems is also quite popular in recent years (Rêgo Segundo et al., 2015; Aghenta and Iqbal, 2019). The main advantage of SCADA solutions is that they are in the first place control-oriented systems with a focus on communication reliability, scalability, and modularity.

This paper describes *eLab*, a lightweight implementation of a SCADA system for process monitoring and control. The system provides a modular architecture and can be used for a variety of different processes or devices of scientific and educational nature. The original *eLab* (Kalúz et al.,

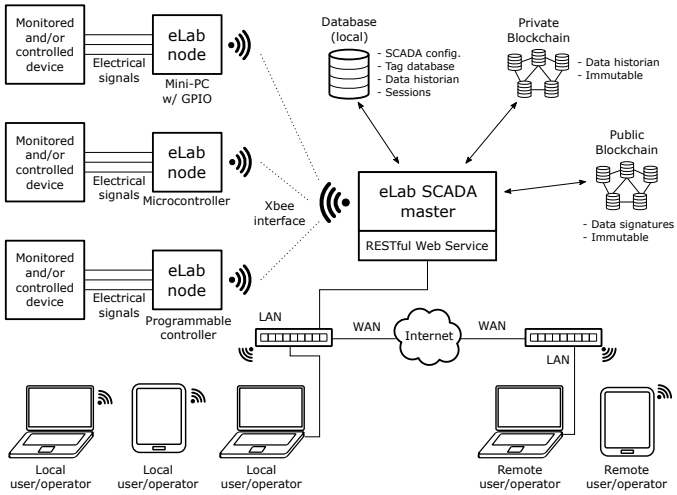


Fig. 1. The architecture of the *eLab* SCADA system, showing all of the functional components and communication interfaces.

2017) is well established at our institute and has been used in educational and scientific process for several years. The main idea behind the *eLab* is to unify the use of laboratory instruments and processes and to provide them to users in an IoT fashion. Students can come to laboratory with own computer, connect to available network, and use any process/device that is connected to SCADA system. Before the first implementation of *eLab*, every laboratory device in our laboratories was controlled via a standalone computer equipped with a dedicated I/O interface (usually a DAQ card). This solution had multiple points of failure, required frequent maintenance, and was expensive.

In the current version of *eLab*, the whole architecture of the system was modified to implement all the necessary features that define the SCADA system. The original HTTP-based communication between the user's computer and end-point device was replaced by coordinator SCADA master computer and two communication layers. First is wireless radio interface between end-point device and SCADA master, and second is HTTP-based RESTful API for communication between user and SCADA master. The current system also has a dedicated local database for data logging, tags, configuration, and a public blockchain database for persistent records. One of the main benefits of new *eLab* is that it does not require specific hardware to be implemented. The SCADA master software can run on either a big server machine or a low-cost single-board computer. The end-points devices (also called nodes) can be implemented on a computer, MCU, or even programmable logic controller (PLC). This variability in hardware can result in the very low-cost implementation of the education-oriented SCADA system.

2. TECHNICAL DESCRIPTION OF *ELAB*

The architecture of *eLab* is shown in Fig. 1. The system contains instruments that are electrically connected to *eLab* nodes. Each node maintains an XBee-based wireless connection with master computer and continuously streams measured process data and listens to issued commands. Master computer mediates access to the databases and provides the main interface for SCADA usage, which is in the form of RESTful Web Services.

2.1 ELab *SCADA master*

SCADA master is a server-side application deployed on a dedicated computer (Fig. 2) that serves as an intermediary processing and communication layer between *eLab* nodes, databases, and client devices. The application is written in Node.js, a server-side implementation of JavaScript. SCADA master holds in memory a state representation of every laboratory process/device that is connected to the system or actively used. Each process/device in the system is identified by a unique name and has associated a set of process tags. Every tag represents a real point of either measurement (usually a sensor) or control (an actuator). In program implementation, a tag is an object that carries the information about the measured or controlled signal, such as the type (analog, digital, modulated), location of the signal on the physical interface of the node, signal resolution, minimum/maximum/default values, engineering unit, and a short description. Based on tag information, the SCADA master is able to translate measured data from binary representation to engineering representation and forth. The configuration of each process/device is stored in a local relational database in JavaScript Object Notation (JSON) format.

The master application operates two communication interfaces for data flow. First is a wireless interface based on XBee 802.15.4 radio modules that interconnects the master computer with nodes. The second interface is a RESTful Web service that provides a set of API routes to monitor and control individual processes/devices.

Fig. 2. A computer running SCADA master and containing local database.

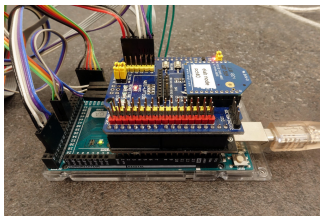
2.2 ELab *node*

In the *eLab* SCADA system, a node is a device that directly connects to the laboratory process via electrical signals, monitors the state of sensors, and controls the actuators. Every laboratory process is governed by at least one node device. In the case of large-scale processes with many signals exceeding the IO capabilities of one node, several nodes can be used in fusion.

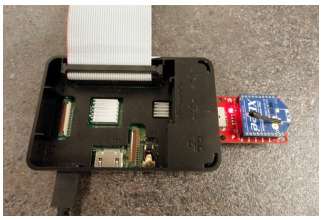
The implementer of a new process connection is not limited by the use of specific hardware as a node. The main requirements on a node hardware platform are the presence

of an appropriate electrical signal interface, programming capabilities, and ability to operate an XBee module (either by UART or USB-to-serial interface). The majority of IoT capable devices support all of these features by default.

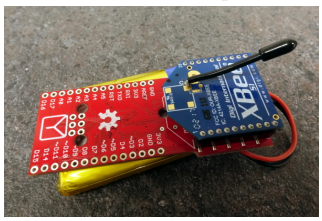
All the nodes used for process/device control in our laboratories are based on MCU boards such as Arduino Mega 2560 (Fig. 3a), Sparkfun Fio v3 (Fig. 3c) or single-board computers like Raspberry Pi (Fig. 3b). The firmware software of *eLab* node is written in C language for MCUs and in Python for Raspberry Pi nodes. The firmware for all 8-bit MCUs used in our laboratories is unified, so that the implementer will make only minor changes in the code, mostly related to the configuration of the signal interface, which is different for every process.



(a) Node based on Arduino Mega 2560



(b) Node based on Raspberry Pi computer



(c) Node based on Fio v.3 portable MCU

Fig. 3. Different types of *eLab* nodes.

2.3 Communication interfaces

In the legacy implementation of *eLab*, the communication of process data was carried out directly between nodes and user computers using an HTTP protocol. The first disadvantage of this approach was that nodes had to contain a server program that brought additional processing requirements, which was undesirable, especially for low-performance 8-bit MCUs. Secondly, the data was transferred in a quite inefficient way, using an ASCII encoded HEX string that represented the current state of MCU's signal interface. The data string contained all the signals, even though not all of them were used to serve sensors or actuators. Moreover, every message was accompanied by data overhead in the form of HTTP headers. These inefficiencies of communication led to limitations of process sampling and update rate, especially for low-performance MCUs.

Master-to-Node XBee Interface In the current version of *eLab*, nodes do not communicate data directly to the users, but through the SCADA master computer. The data link uses XBee radio modules and transmits data in binary form. The communication contains two protocol layers. First is a packet-based XBee API mode that ensures addressing, transmission control, error check, and optional encryption. On the second, lower layer, the *eLab* protocol is implemented. XBee API mode message contains a dedicated frame for communicating process data (Tab. 1) and commands (Tab. 2). The *eLab* protocol is designed to take a minimum amount of transmission capacity to accomplish full-duplex communication. For example, if the process provides 16 digital (binary) measurements, the associated data occupies only two bytes of the binary message. Every analog signal is transferred in a numerical representation of ADC reading or DAC command (two bytes for every ADC/DAC signal up to 16-bit resolution). As a result, this new implementation of data links can achieve higher sampling rates than the previous approach. A sampling rate of 50Hz was successfully tested on four concurrently running nodes, without signs of communication lags or errors.

Table 1. Communication frame with measurements streamed by *eLab* node (in byte order)

Byte length	Data	Description
1	start byte (0x7B)	Indicates start of message
1	frame ID (0–255)	Identifies the message
1	DI length N_D (0–255)	Number of digital inputs in the following bytes
$\lceil N_D/8 \rceil$	Digital inputs (0–255)	One byte can store up to 8 signals in binary form
1	ADC length N_A (0–255)	Number of analog inputs in the following words
$2N_A$	Analog inputs (0x0000–0xFFFF)	One analog reading is stored in 2 bytes
1	stop byte (0xDF)	Indicates end of message

Table 2. Communication frame with commands received by *eLab* node (in byte order)

Byte length	Data	Description
1	start byte (0x67)	Indicates start of message
1	frame ID (0–255)	Identifies the message
1	mode (0–255)	A type of command
1	DO length N_D (0–255)	Number of digital outputs in the following bytes
$\lceil N_D/8 \rceil$	Digital Outputs (0–255)	One byte can store up to 8 signals in binary form
1	PWM length N_P (0–255)	Number of PWM outputs in the following bytes
N_P	PWM Outputs (0–255)	8-bit duty cycle for each PWM output
1	DAC length N_A (0–255)	Number of analog outputs in the following words
$2N_A$	Analog outputs (0x0000–0xFFFF)	One analog command is stored in 2 bytes
1	SPWM length N_P (0–255)	Num. of slow PWM outputs in the following bytes
N_P	Slow PWM Outputs (0–255)	8-bit duty cycle for each SPWM output
1	stop byte (0xCB)	Indicates end of message

RESTful Web Service API The communication between SCADA master and front-end applications/HMIs is ensured by a RESTful Web Service API. The SCADA application provides a Web server that serves users' requests via a set of API routes and sub-routes. *ELab* contains four APIs for a user to interact with: standard API, session API, DCore API, and crypto API. In the standard API,

two main route types are implemented: *set* routes are used to control SCADA features and issue commands to nodes, and *get* routes generally serve to access node measurements and configurations. Session API allows the user to set up sessions, which automatically perform live data logging to either local database or blockchain. This API also provides routes for accessing the lists of sessions and to acquire historical data. DCore API allows the user to utilize messaging features of the implemented blockchain network, i.e., send, receive, and archive custom messages. Crypto API is an experimental version of standard API for communication with SCADA master in homomorphically encrypted way. This API uses extended Pailler's cryptosystem that allows performing a simple arithmetic operation on the encrypted ciphertexts (commands and measurements). Homomorphic encryption has been implemented in *eLab* due to the research purposes for studying fully encrypted control loops.

2.4 Scalability

During the development of *eLab*, it was decided to design the system in such a way that it would be modular and fully scalable on node level as well as a SCADA master level. The configuration of an end-point can be set in such a way that multiple *eLab* nodes control one laboratory process or device. In such a case, the master software performs a logical fusion of the nodes into one combined node. This mechanism is particularly useful for processes with a large number of signals that cannot be electrically covered by one physical node, not even by those with large-scale I/O interface such as Arduino Mega 2560. In our laboratories, the distillation column Armfield UOP1 (Fig. 4a) is such a process, and it is controlled by two nodes (Arduino Mega 2560).

The system can be scaled up also by adding more master computers. In such a scenario, each SCADA master governs a set of nodes, usually located in its direct vicinity (a reach of XBee radio signals). The communication among master computers is carried out over a local network using secured WebSocket protocol. User can use a Web Service of any master computer to establish connection with node that resides in different laboratory.

3. ADVANCED SECURITY FEATURES

3.1 Communication security

Every information system, regardless of whether it is industrial or educational, should provide adequate security of transmitted data. The architecture diagram of *eLab* (Fig. 1) shows several different communication domains. Some can be considered private, such as local database connection, but all others are public. An XBee interface is secured by 128-bit AES encryption implemented directly on a top of the protocol layer. This encryption ensures the security of data transferred between *eLab* nodes and SCADA master. The Ethernet connections utilized by master software are also secured on a level of the application protocol. RESTful Web Service is secured by TLS since it uses HTTPS, and the same applies for blockchain connection that uses WebSocket.

3.2 Process data storage, immutability, and verifiability

ELab employs a combination of data-storage techniques to ensure the immutability and verifiability of process data. This is especially valuable for SCADA systems, where every historical state of the process is stored in the database. The data should be tamper-proof, and a mechanism of tamper detection should be implemented. The motivation for such measures can be illustrated for both industrial and educational points of view. In an industrial plant, the operator with access to the SCADA database can easily change the historical tag values to hide a mistake (e.g., a command that led to a faulty operation). In education, students, even without access to the database, can present altered data results in a hope that there is no reference to compare them with.

To ensure data immutability in the SCADA system, *eLab* uses an open-source blockchain technology DCore developed by DECENT Foundation. Data is stored in a distributed database of consecutive blocks, and every data sample is tied to a verified blockchain transaction. Unlike the proof-of-work consensus, used in many blockchain technologies such as the Bitcoin network, the DCore uses a much more computationally and time-efficient proof-of-stake mechanism. After transaction verification, which takes up to several seconds, the data is persistently stored in blocks governed by the whole blockchain network and cannot be changed in the future.

A data verifiability mechanism is implemented on a local database level in the form of cryptographic signatures. Every process measurement, as a set of data samples with corresponding timestamps and information of origin, is stored as one JSON entry in the PostgreSQL database. Before the data object is stored, an ElGamal signature is computed on a JSON string that also contains the signature of the previous data sample. This mechanism provides an ability to recursively verify the integrity of data series between any two measured samples, and therefore, to detect tampered data.

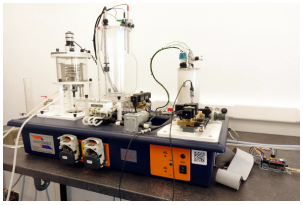
4. LABORATORY DEVICES AND PROCESSES

Currently, *eLab* is used to control several different laboratory processes and devices at our university. Most of the processes represent plants from the chemical and food industry. The biggest plant in the laboratory is a distillation column Armfield UOP1 (Fig. 4a). This process contains a 9-stage column, reboiler, condenser, two heat exchangers, pump, and multiple sensors and actuators. Students also work with two reconfigurable process training stations. First is an Armfield PCT23 (Fig. 4b), a laboratory-scale pasteurization plant, with a multiphase heat exchanger, two storage tanks, a heating vessel, two peristaltic pumps, a piping system with multiple electrically operated values, and a set of mostly temperature sensors. The second training plant is an Armfield PCT40 (Fig. 4c) that represents a chemical plant, with a small-scale water-heated/cooled chemical reactor, storage tank, heating vessel, two peristaltic pumps, one hot water pump, valve-operated piping system, and various sensors. Additionally, several small scale devices are available to students for practical experimentation. These are thermal-optical device (Huba et al.,

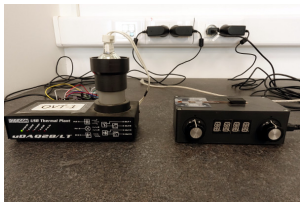
2019), Flexy^2 (Kalúz et al., 2019) (Fig. 4d), and small portable nodes with directly attached sensors (temperature, humidity, etc.) (Fig. 4e).

(a) Distillation column

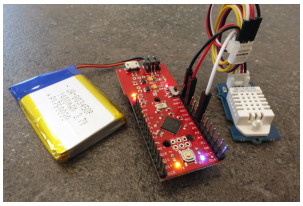
(b) Pasteurization plant



(c) Chemical plant



(d) Thermal device and Flexy2 device



(e) Portable temperature and humidity sensor

Fig. 4. Overview of the processes and devices available in *eLab* SCADA system

5. HOW STUDENTS USE *ELAB*

The current set of laboratory instruments and processes can be divided into two classes: those that require an active presence of student or supervisor in the laboratory; and those that are safe to be operated remotely. The first class contains bigger-scale processes such as pasteurization process, chemical plant, and distillation column. These processes are used in control engineering education mostly for students' semestral and diploma projects as well as for research purposes. The second class of instruments is generally suitable for both local and remote operations. These experiments include the thermal-optical device, Flexy2 device, and portable sensors/actuators. Students use these devices in various courses. For example, the thermal-optical and Flexy^2 devices are used remotely by students of the course *Integrated Control in Process Industries*, where they carry out the tasks of data acquisition, process identification, control design and implementation.

The RESTful Web Services of *eLab* are used in IT-oriented courses and semestral/diploma projects, where students design and develop information systems for data acquisition and visualization.

Since our study program is mainly focused on control system engineering, automation, and process control, the students use MATLAB as a primary tool to handle their study tasks. Due to this reason, the first client-side control interface of the new *eLab* was developed for MATLAB and Simulink. Students connect to *eLab* mostly using their computers equipped with MATLAB that is available for all students at the university. MATLAB implementation of *eLab* is provided in a form of toolbox that can be installed via third-party tbxManager [1]. The manager installs two main interface classes: `ELab()` for management and control of instruments and `ELabData()` for historical data access. At the first usage, the student executes *eLab* in manager mode and install files for a particular laboratory process. This can be done using the following commands.

```
% Creates an instance of eLab in manager mode
manager = ELab()
% Displays a list of available processes
manager.list()
% Installs examples and libraries for
% therma-optical device UDAQ28
manager.install('udaq28')
% Opens command line and Simulink examples
% for installed process
manager.open('udaq28')
```

At this point, the student can choose between two usage scenarios: command-line control and Simulink control. In the first scenario, the process is controlled directly from the command line or MATLAB scripts written by the student. The example commands are shown in the following code snippet.

```
% Creates a control interface for the device
my_device = ELab('udaq28', 'control')
% Get all measured data at once
tags = my_device.getAllTags()
% Get a specific tag
temp = my_device.getTag('temperature')
% Get a value of specific tag
light_value = my_device.getTagValue('light')
% Set a value of specific tag
my_device.setTag('bulb',85)
% Set values of multiple tags at the same time
my_device.setTags({'bulb', 50.8, 'led', 100})
```

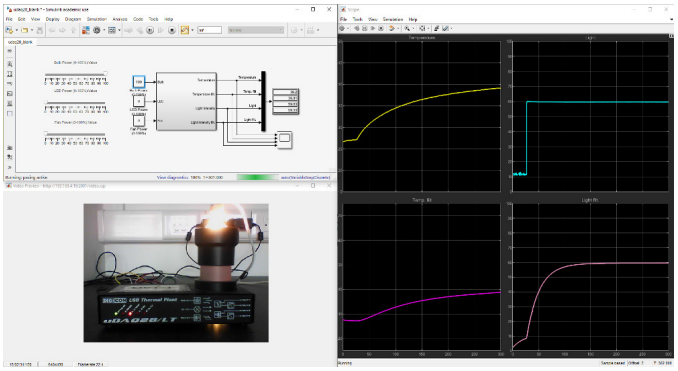


Fig. 5. User interface in MATLAB/Simulink for remote operation of thermal-optical device

Alternatively, the second scenario allows students to use Simulink to design and execute custom control schemes in a visual form. Simulink can also be used as a front-end for the implementation of remote control laboratories. All the small-scale devices mentioned in Section 4 are online 24/7 and accessible via the Internet through a VPN of the

[1] `https://www.tbxmanager.com/`

university. Fig. 5 shows an example of MATLAB-based remote laboratory using the *eLab* SCADA system.

6. RESEARCH AND APPLICATION OPPORTUNITIES

ELab can be used as a framework for development of remote laboratories. Most of the RL solutions require hardware architecture capable of interconnecting physical devices with user's computer, ideally by using Web-based technologies. *ELab* provides a ready-made solution to both physical and communication architecture. The endpoint for accessing the whole system is a Web Service API allowing the implementers to build Web-based user interfaces. In such a case, the RL developer does not need to cope with specific hardware requirements of an experiment, neither the middleware layers, but can focus purely on a client-side implementation of the laboratory and its features.

The foremost research opportunity that *eLab* provides lies in its cryptographic middleware and blockchain database. Researchers can study numerous scenarios of sensitive process data handling and transfer in a secure manner. Currently, ongoing research is focused on the encryption of the whole control loops. In these scenarios, the measurements are homomorphically encrypted and sent via the network to the controller that directly computes decisions on cyphertexts without the need for decryption. The generated control action is also in the form of cyphertext and is decripted on site, just before its application to the process.

7. CONCLUSIONS AND FUTURE WORK

ELab, as a lightweight SCADA system, provides a unified architecture that allows implementers to build integrated laboratories with various experimental devices. The system provides all the features necessary to monitor and control instruments, perform data logging, and manage existing or new instruments to laboratories.

Currently, the students and researchers operate the system using MATLAB and Simulink, but since *eLab* provides RESTful Web Services, any software technology/language capable of HTTP communication can be utilized for this purpose. Web Services also provide a useful framework for the development of Web-based remote laboratories.

In future work, the development will be focused on software tools that will allow implementers to add and configure new laboratory devices with minimal effort. This tool will automatically generate a firmware for particular *eLab* node, set up the communication interface, and register the device in the SCADA master database. Moreover, we plan to develop a unified client-side interface for *eLab*-based remote laboratories in the future.

ACKNOWLEDGEMENTS

The authors acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grant 1/0004/17 and the Slovak Research and Development Agency under the project APVV-15-0007.

REFERENCES

Aghenta, L.O. and Iqbal, M.T. (2019). Development of an iot based open source scada system for pv system monitoring. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, 1–4. doi:10.1109/CCECE.2019.8861827.

Cuayo, L.D., Kerbee Culla, J., Gualvez, J., Padua, S.E., and John Gallano, R. (2018). Development of a wireless microcontroller-based scada rtu. In *TENCON 2018 - 2018 IEEE Region 10 Conference*, 2566–2570. doi:10.1109/TENCON.2018.8650114.

Daros, M.R., de Lima, J.P.C., Rochadel, W., Bento Silva, J., and Simão, J.S. (2015). Remote experimentation in basic education using an architecture with raspberry pi. In *2015 3rd Experiment International Conference (exp.at'15)*, 75–78. doi:10.1109/EXPAT.2015.7463218.

Domínguez, M., Morán, A., Alonso, S., Prada, M.A., Pérez, D., and Fuertes, J.J. (2019). Experimentation environment for industrial control systems cybersecurity: On-site and remote training. *IFAC-PapersOnLine*, 52(9), 248 – 253. doi:https://doi.org/10.1016/j.ifacol.2019.08.209. URL `http://www.sciencedirect.com/science/article/pii/S2405896319305464`. 12th IFAC Symposium on Advances in Control Education ACE 2019.

Golob, M. and Bratina, B. (2013). Web-based monitoring and control of industrial processes used for control education. *IFAC Proceedings Volumes*, 46(17), 162 – 167. doi:https://doi.org/10.3182/20130828-3-UK-2039.00041. URL `http://www.sciencedirect.com/science/article/pii/S1474667015340945`. 10th IFAC Symposium Advances in Control Education.

González, I. and Calderón, A.J. (2019). Integration of open source hardware arduino platform in automation systems applied to smart grids/micro-grids. *Sustainable Energy Technologies and Assessments*, 36, 100557. doi:https://doi.org/10.1016/j.seta.2019.100557.

Huba, M., Kurčík, P., and Kamenský, M. (2019). Thermo-Optical Laboratory Plant uDAQ28/LT. `https://www.eas.sk/mod/product/show.php?ID=59`. Online, Accessed: 2019-11-9.

Kalúz, M., Klaučo, M., Čirka, Ľ., and Fikar, M. (2019). Flexy2: A portable laboratory device for control engineering education. In *12th IFAC Symposium Advances in Control Education*, 159–164.

Kalúz, M., Čirka Ľ., Valo, R., and Fikar, M. (2017). Lab of Things: A Network-Based I/O Services for Laboratory Experimentation. *IFAC-PapersOnLine*, 50(1), 13486 – 13491. doi:https://doi.org/10.1016/j.ifacol.2017.08.2330. URL `http://www.sciencedirect.com/science/article/pii/S2405896317331452`. 20th IFAC World Congress.

Rêgo Segundo, A.K., Cocota, J.A.N., Hilário, R.Q., d. O. Gomide, V., and Ferreira, D.V.M. (2015). Low cost scada system for education. In *2015 IEEE Global Engineering Education Conference (EDUCON)*, 536–542. doi:10.1109/EDUCON.2015.7096022.

Rojas, A.M. and Barbieri, G. (2019). A low-cost and scaled automation system for education in industrial automation. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 439–444. doi:10.1109/ETFA.2019.8869535.