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

Development of a Remote Industrial Laboratory for Automatic Control based on Node-RED

Manuel Domínguez * **Raúl González-Herbón** **
José R. Rodríguez-Ossorio ** **Juan J. Fuertes** *
Miguel A. Prada * **Antonio Morán** *

* *Grupo de investigación en Supervisión, Control y Automatización de Procesos Industriales (SUPPRESS), Esc. de Ing. Industrial e Informática, Universidad de León, Campus de Vegazana s/n, 24007, León, Spain. (e-mail: manuel.dominguez@ unileon.es).*
** *Universidad de León, Campus de Vegazana s/n, 24007, León, Spain.*

Abstract: In this paper, we propose a remote laboratory for automatic control that enables an easier interconnection and integration of its elements. It is based on Node-RED and the MQTT protocol, which enable the easy development of laboratories and an intuitive but flexible user interface for the students. Node-RED is an open-source programming platform oriented to easily connect hardware devices, APIs and online services. The remote laboratory uses three different elements: an instructor's client, one student's client for each student and a central broker. The instructor's client is the only one with direct access to the plant, through a Modbus TCP connection to the PLC, so it decides which student can manage the system and monitor their actions, hiding the complexity of automation to the students and providing an additional layer for safety and security. The students' clients are installed in the laboratory or students' own PCs. To perform the tasks, the students use a predefined dashboard or the Node-RED editor for easy graphical programming. In any case, the process is as follows: instructor's client reads/writes variables of the plant, publishes their values using a multi-level hierarchical structure, subscribes to the control actions published by the students' clients and manages how and when those actions are sent to the corresponding pilot plant. This approach has been assessed through the implementation of a hands-on task where students need to set the parameters of a PID to appropriately control the level of a tank in a real pilot plant.

Keywords: Control education, remote laboratory, interconnection technology, digitalization.

1. INTRODUCTION

The technological architecture that supports the educational remote laboratories needs to consider the current advances in digitalization (Grodotzki et al., 2018), especially when they rely on industrial-level physical systems. Remote laboratories have been successfully used in the field of automatic control in the last two decades (Heradio et al., 2016) and their development in this field evolves according to the technologies available for industrial control systems. Therefore, it is necessary to take into consideration the principles of Industry 4.0 (Lasi et al., 2014) and the Industrial Internet of Things (IIoT), leading to a greater stress on interconnectivity, cybersecurity, cloud and edge computing, etc.

For that reason, in this paper, we propose a remote laboratory prototype for automatic control that enables an easier interconnection and integration of its elements. It is essentially based on a single technology, Node-RED, which provides advantages, such as the easy development of laboratories and an intuitive but flexible user interface for the students. Node-RED is an open-source programming platform oriented to easily connect hardware devices, APIs and online services, which was created by IBM. The interface editor is developed in the web browser, where programming is performed graphically using flows, an approach that enables rapid development of applications. Some of the major manufacturers of industrial equipment, such as Siemens and Schneider Electric, support this technology in their latest edge computing devices.

The prototype of remote laboratory presented in this paper allows students to remotely control an industrial pilot plant. This functionality is provided by three different elements: the instructor's client, the student's client and the central broker. Both clients are implemented in Node-RED and communicate with each other through the central broker using the MQTT (Message Queuing Telemetry Transport) protocol. The instructor's client is the one with direct access to the plant, and the one in charge of management of the students and systems. The students' clients, which can be installed in the laboratory or students' own PCs, are the point where students develop their practical tasks.

The proposed approach has been implemented to develop a hands-on task for an introductory control course. In this

task, the students need to adequately configure a PID controller to fulfill some requirements of the response in a control loop of a tank level in a real pilot plant available in the University of León. The parameterization of the PID is performed by means of a dashboard, where students can monitor the effect of their changes in the system behavior. The paper is structured as follows. First, we discuss the background related to the application of new digitalization approaches to remote laboratories. Next, the proposed method is explained in detail. In section 4, the experimental assessment of the proposed approach is described. Finally, conclusions are drawn.

2. BACKGROUND

The educational approaches proposed to address the knowledge gap that has appeared due to digitalization focus on learning strategies that are active and based on practice and the use of enabling technologies (Baena et al., 2017; Benešová et al., 2018). However, key ideas behind this trend such as interoperability, cyber-physical systems or IIoT have been aligned with the concepts behind remote laboratories since their earliest application to control education. Indeed, latest advances in this area are generally oriented to enhance interoperability (Sáenz et al., 2015).

Higher integration with information technologies is leading to a convergence on platforms and protocols that are available both for low-cost hardware and industrial-level systems. As commented before, Node-RED is an open platform for the easy connection of devices that is supported by some modern equipment of major manufacturers, but can also run in a resource-constrained compact computer. This is possible because it is built on the Node.js JavaScript run-time environment and designed to be light and robust. Node-RED provides an unparalleled connectivity to other platforms and APIs, through the different nodes available in the palette (library), including from implementations of industrial protocols to social network connectivity. The usefulness of this platform for interconnection in industrial environments has already been highlighted (Tabaa et al., 2018; Toc and Korodi, 2018), especially when it is used along with standard communication protocols that are suited for communication among devices but also to transfer data to the cloud or data analytics platforms. Node-RED also enables the connection with cloud services for storage and analytics. Although there are contributed nodes for all the major platforms, a wider support is provided for the IBM Cloud services.

Nonetheless, there are other alternatives to be considered to develop IoT architectures (Derhamy et al., 2015; Ray, 2017). For instance, Eclipse IoT, also open source, supports cloud connectivity and communication standards. Its IoT edge framework, Eclipse Kura, has some characteristics in common with Node-RED, e.g., Kura Wires. And the IoTivity open source framework provides APIs for different programming languages oriented to facilitate the secure connection of devices. Although these technologies provide a wider set of functionalities for IoT-related development, the emphasis of Node-RED on easy development makes it more suitable for educational purposes.

On the other hand, MQTT (Message Queuing Telemetry Transport), a standard open protocol created by IBM (Banks and Gupta, 2014) for lightweight machine-to-machine (M2M) communication, has lately become one of the most common communication technologies in the field of Internet of Things (IoT) and cloud interconnection. Its lightweight nature also makes it available for resource-constrained devices. It uses a publish-subscribe messaging model and a star topology, where each client can publish its own topics or subscribe to topics created by other clients. However, communication among clients is not direct, since a central node or broker manages and transmits messages among clients, ensuring flexibility and scalability. Furthermore, MQTT messages can be encrypted, unlike in traditional industrial control protocols. This functionality is important, because cybersecurity is an important challenge of digitalization. An alternative standard lightweight protocol is CoAP (Constrained Application Protocol), which is also extensively used in IoT. However, its one-to-one communication is not as well suited for the aims of this work as the many-to-many communication that MQTT provides through a central broker.

In the field of education, the application of technologies such as Node-RED has been scarce (Chaczko and Braun, 2017). The usefulness of MQTT for the communication with resource-constrained devices in the area of remote laboratories has already been shown in Prada et al. (2016).

3. PROPOSED METHOD

The application of open standards, as well as modular and multi-tier architectures, is useful to achieve aims such as seamless integration of diverse physical systems, hiding their underlying complexity and providing enough flexibility for the development of different practical tasks (Prada et al., 2015). Flexibility is also increased when it is possible to define both programming tasks and assignments where the student is offered an interactive dashboard. It is also important to let faculty focus on the development of educational content, avoiding that management and maintenance of the remote laboratory becomes a burden. Obviously, reducing the number of different tools is interesting to guarantee easy development. Finally, it is necessary to use communication protocols that enable connection to cloud services, data exploitation and cybersecurity measures.

For that purpose, in this paper, we propose an architecture for a remote industrial laboratory for automatic control that is composed of three different elements: an instructor's Node-RED client, one student's Node-RED client deployed for each student, and a central broker (see Figure 1). Information exchange between the students' client and the instructor's one uses MQTT. Both types of clients are implemented as Node-RED flows, whereas any standard implementation of the broker can be used for the central element.

The Node-RED client run by the instructor communicates with two servers. On one hand, it is linked to the process controller, using the specific communication technology needed for that purpose. On the other hand, it communicates with the students' clients. Additionally, a dashboard is developed so that the instructor manages and monitors the hands-on task, deciding which student can control the

system and supervising their actions. The presence of this element is useful for two different reasons: it hides the complexity of the automation to the students, letting them focus in the task at hand, and provides an additional layer of control which can be used to guarantee the safe and secure operation of the plant.

The students' clients are installed in the laboratory workstations or anywhere else, if the hands-on task is performed remotely. The students will run this instance of Node-RED and use either the editor (to modify/create a flow) and/or a predefined dashboard to perform the task following the instructions. Graphical programming in the web editor serves the purpose of being readily usable for students without strong programming knowledge, in a similar way to tools such as Blockly (Fraser et al., 2013; Galan et al., 2017) or Scratch (Resnick et al., 2009). In the context of a remote laboratory of automatic control, this functionality could be used to propose a practical task where students need to program the controller. In this case, students would use the MQTT communication nodes and the other ones they need to solve the task. Furthermore, it is also possible to build dashboards that run in the web environment, which is useful to obtain a visual feedback of what is happening in the real system, but also for control education tasks where students just need to set some predefined parameters, e.g., for PID design or tuning tasks, with a controller implemented either in Node-RED or in the corresponding PLC. One of the main advantages of these alternatives is the flexibility to design different hands-on control tasks with real systems.

The MQTT broker manages the exchange of information between the instructor's client and the students' clients. The instructor's client reads/writes input/output variables of the plant, publishes the status of those variables in the corresponding topics, subscribes to the control action topics published by the students' clients and manages how and when the control actions are sent to the corresponding plant. The students' Node-RED clients subscribe to the topics corresponding to the pilot plant variables published by the instructor's client. These clients publish the control actions computed by the controller programmed/configured by the student in the hands-on task. Since the instructor's client is subscribed to these control actions, it will receive them and, whenever the conditions are met, it will sent them to the corresponding actuators through the communication link the instructor's client has with the real system.

Since all the communications are performed through the broker, it is possible to extend this approach for truly remote experimentation by simply placing it in an internet-facing device. In this case, it is especially important to set up the security properties available for MQTT. The most useful ones are the authentication with client certificates (X.509) and encrypted communication over Transport Layer Security (TLS). Furthermore, the MQTT broker can be located in a demilitarized zone to filter any other undesired connections.

Variables in MQTT are organized as topics in a multi-level hierarchical structure, where each level is separated with '/'. There are wildcard symbols that allow us to select all the topics, such as # (multi-level downstream) or + (single-level). Therefore, the instructor's client can structure the input/output variables it publishes in different levels: the highest level of the hierarchy can correspond to the plant, thus enabling the addition of new systems to the set of available plants; the second level can correspond to the variable type (i.e., digital/analog input/output); and the third level to the variable tags. If a student's client needs to subscribe to all the analog inputs of the second pilot plant, it will subscribe to "Plant2/AnalogInput/#". On the other hand, the students' clients will publish control actions as outputs. The topic structure is similar to the one built by the instructor's client but, in this case, the first level can correspond to the student's ID (e.g., the university e-mail account). Using the management and monitoring dashboard available in the client, the instructor decides which student can interact with a certain plant during a certain time slot and, as a result, it subscribes to the corresponding topic set "StudentID/#" and sends the values published in the topic to the corresponding output variables in the PLC. Figure 2 summarizes the proposed hierarchical structure.

In short, both student's programming/configuration and instructor's management and monitoring is made through Node-RED. The approach provides enough flexibility and simplicity to enable an easy definition of hands-on tasks on different physical systems. Furthermore, the technical complexity of communication with the PLC to access the pilot plants is completely hidden to the student, who only needs to know the broker IP and the variables to develop the controller. That serves the purpose of letting students focus on the concepts rather than on the technical details.

4. EDUCATIONAL EXPERIENCE

4.1 Practical task

The hands-on task we designed to assess the proposed approach is of application to a first course on control theory for engineers. It consists in the parameterization of a PID controller that is used in a networked SISO control loop of a tank level available in an industrial process scale model (Domínguez et al., 2010). This pilot plant, implemented by the research group both for research and educational purposes, allows to control four physical variables (pressure, flow, level and temperature), through all the necessary instrumentation. There are three copies of the plant available in the laboratories of the University of León. Each plant is composed of a main process circuit and two utility circuits associated to temperature (heating and cooling circuits). All those circuits can be used independently or interact among themselves. Recirculation to its cascade tanks is obtained by means of a pumping circuit fed by a centrifugal pump. The pumped fluid passes through a valve to vary the flow rate and two heat exchangers to regulate the temperature. The fluid is pumped into the top cascade tank by its base. It was designed to ensure safe operation in educational conditions. The plant and its manageable variables are shown in Figure 3.

In the proposed task, the student's client subscribes to the variable that needs to be controlled (the tank level LT21), establishes a setpoint, computes the control action and publishes that action to the output variable SZ21 (pump

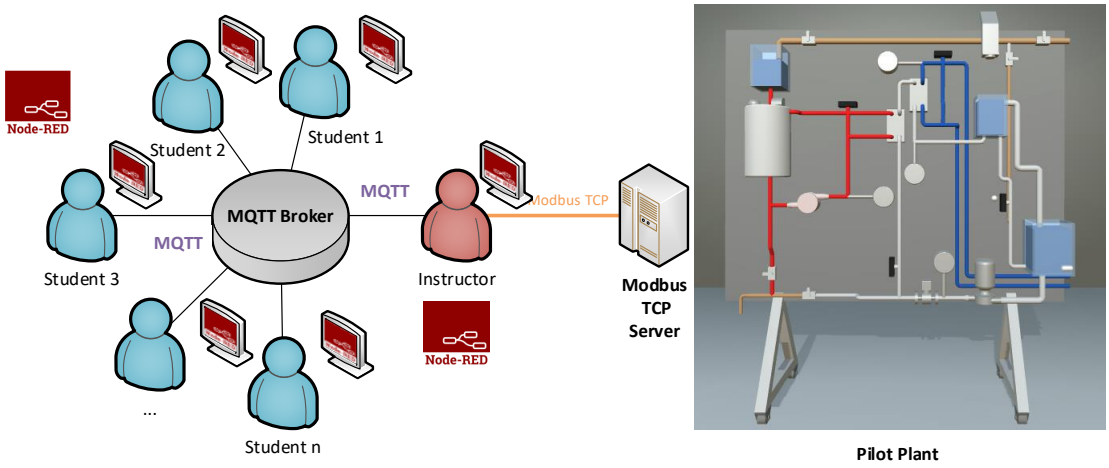

Fig. 1. Schema of the system architecture

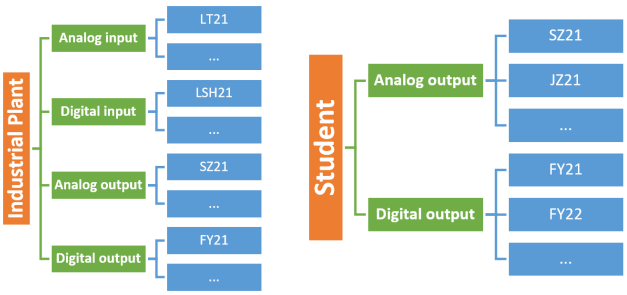

Fig. 2. Topic hierarchies for the prototype

speed). The control action is computed by a predefined PID controller that the student needs to parameterize. The students perform this parameterization and monitor the results using a dashboard that was also developed in Node-RED.

Apart from monitoring the response of the system in real time through the webcam feed and the charts, the student can store the values of the variables involved in the control loop throughout the whole experiment. These data can be downloaded as a .csv. This way, students have a record of the task that can be used for further analysis or evaluation.

4.2 Implementation and results

Although there are different commercial or open source MQTT brokers that can be used, the open source Eclipse Mosquitto (Light et al., 2017) has been used in this work. The MQTT broker has been configured in a Schneider Electric iPC Box IIoT, which is a compact industrial PC designed for edge computing. The instructor's client has been installed in another PC with local network access to the physical system (i.e., to the PLC controlling the process).

The programmable logic controller used for this process is a Schneider Electric Modicon M340, and the communication between the instructor's client and the PLC uses the Modbus TCP protocol. It is necessary that the link between the variables and the Modbus registers is properly configured in the control strategy of the PLC. The package used in Node-RED for the Modbus TCP communication with the PLC is node-red-contrib-modbustcp (Swales et al., 1999).

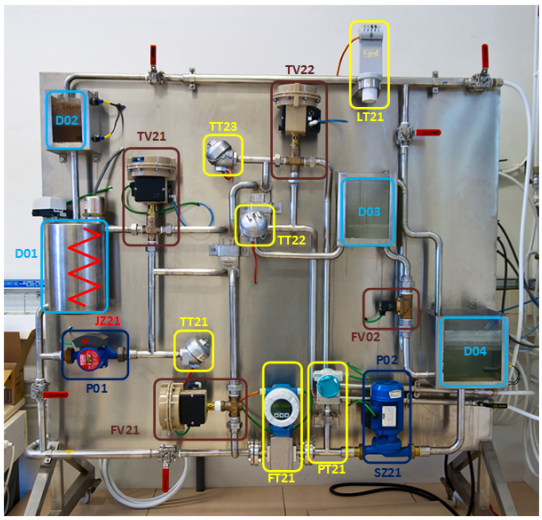

Fig. 3. Plant

With regard to MQTT communication, it is necessary in all the clients (flows) to set 3 parameters of the MQTT input node (from the Node-RED core set): the route to the MQTT broker (as a string with the structure client_ID@Broker_IP:Broker_Port); the topic(s) it subscribes to (generally using wildcards, such as +/AI/# to let a student's client receive all the analog inputs of the available plants); and the required quality of service (QoS), which specifies the guarantee of delivery of a message to at most once (0), at least once (1) or exactly once (2).

The Node-RED implementation of the instructor's client provides a dashboard that the instructor can use for a simple management and monitoring of the hands-on task. In different tabs, the instructor can manage the students or monitor the plant(s) involved in the practical task (see Figure 4). The first tab is oriented to manage students for the hands-on task. The second tab displays the entire list of students. Finally, the dashboard provides as many tables as pilot plants connected to the system, with the aim that the instructor can interact with these plants and monitor the students' performance in them. To make management

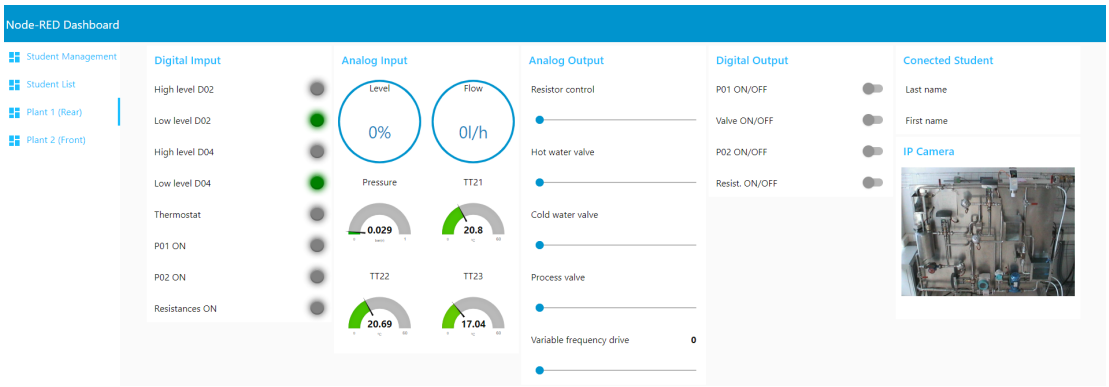

Fig. 4. Instructor's dashboard

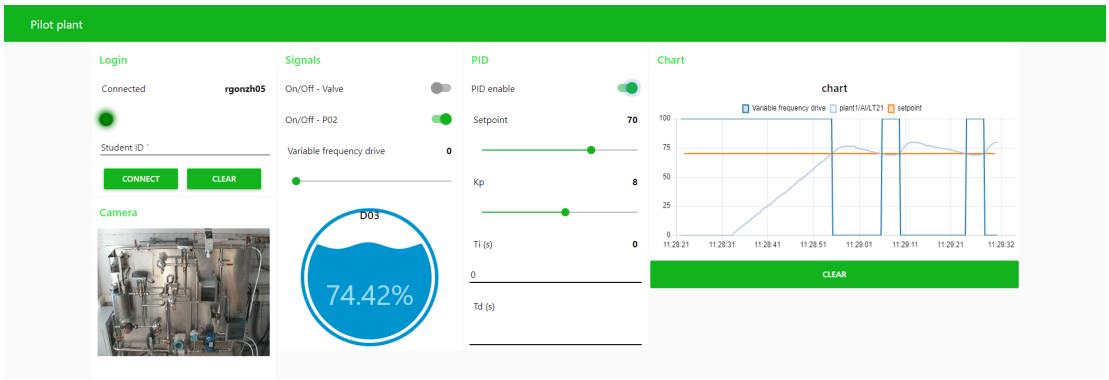

Fig. 5. Student's dashboard

of students easier, a list of students can be imported (in CSV format). Afterwards, the instructor can select the active student from a drop-down list, i. e., decide which one has write access to the output variables of a certain plant.

The instructor's client subscribes to the control actions provided by the students. After selecting a certain student, the instructor reads values from the updates of the topics in the hierarchy. This Node-RED flow reads values from those topics, converts them to a suitable format and sends them to the PLC through Modbus TCP. In more detail, Figure 6, shows how the MQTT node, subscribed to all the digital outputs sent by the students (+/DO/#), reads the updated values, which are later set to a selector that filters messages according to the StudentID with access rights to each plant. Afterwards, these messages are distributed according to its variable tag to redirect them, after a type conversion, to their corresponding Modbus coils and dashboard elements. Analogously, the flow also implements the acquisition of input data from the plant through Modbus TCP, their conversion to the appropriate variable types, their consequent publication as MQTT topics and their visualization in the dashboard.

With respect to the student's client, it is necessary that students set their ID in the dashboard (see Figure 5). The accessible plant will be determined according to what the instructor has configured. A task input panel in the dashboard allows the student to enter the setpoint and parameters of a PID, but also to deactivate the controller and send manual actions to the actuators. The Node-RED flow links the controlled process variable and the setpoint with the PID node (node-red-contrib-pid), whose parameters are configured with the values set by the student in the corresponding dashboard fields. The resulting control action is redirected to the Modbus register of the pump speed, after a type conversion.

The student's dashboard also includes other features that are commonly present in user interfaces of remote laboratories, such as a line chart of the relevant variables and a camera stream. Monitoring the status of plant variables using the chart or the video feed is useful for the students to achieve a better understanding of the system and of the effect that changes in the PID parameters have in its behavior.

5. CONCLUSION

In this paper, we presented a remote laboratory for automatic control that uses Node-RED and the MQTT protocol to achieve certain aims such as easy development or greater flexibility for interconnection and integration of new systems. The prototype of remote laboratory uses three different elements: an instructor's client, one student's client for each student (both of them created as Node-RED flows) and a standard MQTT central broker. The instructor's client is the only one with direct access to the plant, through a Modbus TCP connection to the PLC. This architecture allows the instructor to decide how and when a student can manage the system and to monitor their actions and provides an additional layer for safety and security. The automation and communication details are hidden to the students, who will use a predefined dashboard for the hands-on task. The proposed approach

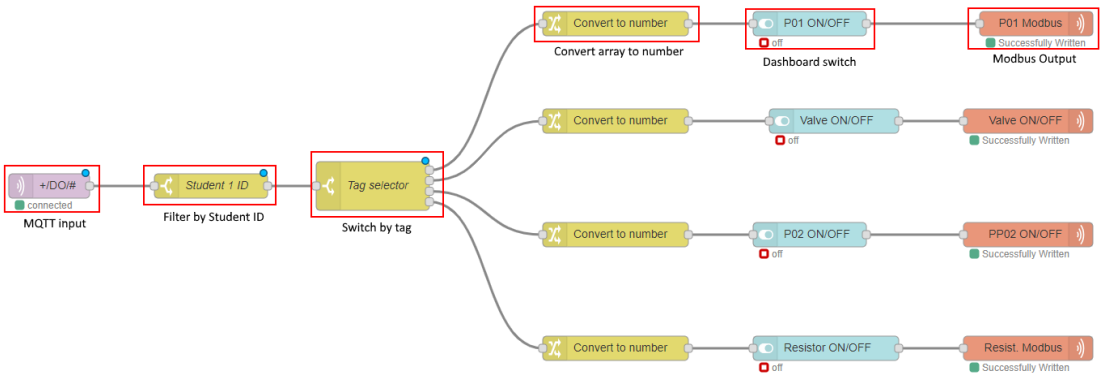

Fig. 6. Flow to read outputs provided by the student and apply them to the plant.

creates a topic structure in MQTT, so that the instructor's client publishes the input variables of the physical system and subscribes to the control actions provided by the students. Equivalently, the student's client subscribes to the input variables, which are used to compute the corresponding control action that is later published.

The network architecture, the topic structure, the communication procedure and its implementation in Node-RED are explained in detail. Finally, we show the application of this approach to an educational task, where students need to configure a PID controller on a level control loop of one of three real industrial plants available in the University of León.

REFERENCES

Baena, F., Guarin, A., Mora, J., Sauza, J., and Retat, S. (2017). Learning factory: The path to industry 4.0. *Procedia Manufacturing*, 9, 73–80.

Banks, A. and Gupta, R. (2014). MQTT version 3.1.1. Technical report, OASIS standard.

Benešová, A., Hirman, M., Steiner, F., and Tupa, J. (2018). Analysis of education requirements for electronics manufacturing within concept industry 4.0. In *2018 41st International Spring Seminar on Electronics Technology (ISSE)*, 1–5. IEEE.

Chaczko, Z. and Braun, R. (2017). Learning data engineering: Creating iot apps using the node-RED and the RPI technologies. In *16th International Conference on Information Technology Based Higher Education and Training (ITHET 2017)*, 1–8. doi: 10.1109/ITHET.2017.8067827.

Derhamy, H., Eliasson, J., Delsing, J., and Priller, P. (2015). A survey of commercial frameworks for the internet of things. In *20th IEEE Conference on Emerging Technologies & Factory Automation (ETFA)*, 1–8. IEEE.

Domínguez, M., Fuertes, J.J., Reguera, P., González, J.J., and Ramón, J.M. (2010). Maqueta industrial para docencia e investigación. *Revista Iberoamericana de automática e informática industrial*, 1(2), 58–63.

Fraser, N. et al. (2013). Blockly: A visual programming editor. Technical report, Google. URL `https://code.google.com/p/blockly`.

Galan, D., Heradio, R., de la Torre, L., Dormido, S., and Esquembre, F. (2017). Conducting online lab experiments with Blockly. *IFAC-PapersOnLine*, 50(1), 13474–13479.

Grodotzki, J., Ortelt, T.R., and Tekkaya, A.E. (2018). Remote and virtual labs for engineering education 4.0: Achievements of the elli project at the TU Dortmund University. *Procedia Manufacturing*, 26, 1349–1360.

Heradio, R., de la Torre, L., and Dormido, S. (2016). Virtual and remote labs in control education: A survey. *Annual Reviews in Control*, 42, 1–10.

Lasi, H., Fettke, P., Kemper, H.G., Feld, T., and Hoffmann, M. (2014). Industry 4.0. *Business & information systems engineering*, 6(4), 239–242.

Light, R.A. et al. (2017). Mosquitto: server and client implementation of the MQTT protocol. *The Journal of Open Source Software*, 2(13), 265.

Prada, M.A., Fuertes, J.J., Alonso, S., García, S., and Domínguez, M. (2015). Challenges and solutions in remote laboratories. application to a remote laboratory of an electro-pneumatic classification cell. *Computers & Education*, 85, 180–190.

Prada, M.A., Reguera, P., Alonso, S., Morán, A., Fuertes, J.J., and Domínguez, M. (2016). Communication with resource-constrained devices through MQTT for control education. *IFAC-PapersOnLine*, 49(6), 150–155.

Ray, P.P. (2017). A survey on visual programming languages in internet of things. *Scientific Programming*, 2017.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J.S., Silverman, B., et al. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60–67.

Sáenz, J., Chacón, J., De La Torre, L., Visioli, A., and Dormido, S. (2015). Open and low-cost virtual and remote labs on control engineering. *IEEE Access*, 3, 805–814.

Swales, A. et al. (1999). Open Modbus/TCP specification. Technical report, Schneider Electric.

Tabaa, M., Chouri, B., Saadaoui, S., and Alami, K. (2018). Industrial communication based on Modbus and Node-RED. *Procedia computer science*, 130, 583–588.

Toc, S. and Korodi, A. (2018). Modbus-OPC UA wrapper using node-RED and IoT-2040 with application in the water industry. In *IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY2018)*, 99–104. doi:10.1109/SISY.2018.8524749.