Interdisciplinary Description of Complex Systems 17(3-A), 430-437, 2019

LEARNING PLATFORM FOR SMART CITY APPLICATION DEVELOPMENT

Dalibor Dobrilović[1,] *, Milan Malić[2] and Dušan Malić[3]

[1]University of Novi Sad, Technical Faculty "Mihajlo Pupin"
Zrenjanin, Serbia

[2]Panonit
Novi Sad, Serbia

[3]Technical College of Applied Sciences in Zrenjanin
Zrenjanin, Serbia

DOI: 10.7906/indecs.17.3.1
Regular article

Received: 30 November 2018.
Accepted: 31 August 2019.

ABSTRACT

With the emerging trend of the Internet of Things and Smart City environments, new trends have arisen in building applications, too. The current approach to application development, known as monolithic architecture, is not suitable for the newest appliances. Recently, the microservice-based application architecture has emerged as a potential solution that could meet the imposed challenges. In order to educate new generations of software and IT engineers to be able to develop such microservice-based applications, educators at universities should establish an efficient platform to ensure this process. This article presents the portable and lightweight platform for teaching application development based on microservices. The platform is focused on supporting lightweight publish/subscribe messaging protocols, such as MQTT, and built-on open-source hardware development boards such as Arduino/Genuino. This Smart City learning platform is designed to enable the collaborative development of systems, applications and services for the cities of the future, and it should be efficient enough to support the learning of all the necessary skills that can be used for the development of complex and large-scale systems. The architecture of the proposed platform, as well as its elements are presented in this article.

KEY WORDS

smart city services, engineering education, publish-subscribe protocols, microservices, IoT

CLASSIFICATION

ACM: 10011007.10010940.10010971.10010972.10010975
JEL: L86
PACS: 07.07.Df; 07.05.Bx

*Corresponding author, η: dalibor.dobrilovic@uns.ac.rs; ++381 62 8019760;
Technical Faculty "Mihajlo Pupin", Djure Djakovica bb, 23 000 Zrenjanin, Serbia

INTRODUCTION

The variety of the software and hardware technologies that have appeared in the past decade has shaped new trends for the ICT market. With novel trends, such as the Internet of Things and Smart City environments, new solutions for building applications have emerged as well. The current approach of monolithic architecture development is not suitable for the appliances in these new environments. In order to meet the imposed challenges, the architecture of microservice-based systems has arisen as a potential solution. The capability of higher education institutions to educate new software and IT engineers for the development of microservice-based applications has become an important issue. In order to address this challenge, universities should establish an effective learning platform for such process.

This article presents the portable and lightweight platform for teaching microservice application development. The platform is based on open-source hardware and software and it is focused on supporting lightweight publish/subscribe messaging protocols such as MQTT. Open-source hardware development boards such as Arduino/Genuino and clones are used as devices for the platform. The Smart city learning platform is planned to be used in the learning process within university curricula and it is designed to enable the collaborative development of systems, applications and services for the smart cities of the future.

RELATED WORK

The importance of integrating microservice-based application development in engineering education, especially in IoT and Smart City environments, has been underlined by the following references. According to the first paper, the microservice architecture has emerged to meet the industry's needs for scalability, evolvability and maintainability of large-scale distributed systems. Also, microservices have received a wide adoption in the industry among companies building large-scale applications, such as Amazon and Netflix, as well as Platform-as-a-Service (PaaS) providers, such as Pivotal [1]. At the same time, the authors shared their early experience in applying the microservice architectural style to build a Smart City IoT platform for a variety of applications.

In the second paper [2] the authors present a microservice-based architecture equipped with a real-time environmental sensor system that has highly scalable applications in a cloud environment. The purpose of the proposed system is to monitor and control the transportation of hazardous materials. The authors integrated the global positioning system (GPS) technology, wireless sensor networks (WSN), geographic information system (GIS), in addition to IoT technologies, to achieve real-time monitoring and tracking of hazardous materials.

The use of a microservice architecture designed to address the key practical challenges in smart city platforms called InterSCity is presented in another paper [3]. This is a microservice-based, open-source, smart city platform that enables the collaborative development of large-scale systems, applications and services for the cities of the future, contributing to turn them into truly smart cyber–physical environments [4]. The platform is presented together with the set of experiments that evaluate its scalability [5].

This article presents the platform for teaching microservice application development based on open-source hardware and software. The main motivation for developing the presented platform was to enable the teaching of application development based on a group of protocols. This group of protocols can be called application layer protocols, although different names such as messaging protocols, publishing-subscribe protocols or machine-to-machine protocols can be used as well. This group includes protocols such as MQTT

(Message Queuing Telemetry Transport), AMQP (Advanced Message Queuing Protocol), XMPP (Extensible Messaging and Presence Protocol), DDS (Data Distribution Service), HTTP (Hypertext Transfer Protocol) and CoAP (Constrained Application Protocol) [6]. They are widely used in Smart City systems, and, as an example, the comparison of the response time of communication protocols such as CoAP, MQTT, XMPP and WebSocket, used in smart parking system is given in [7].

The platform has inspired similar studies [8-11] and the previous works of the authors of this article [13], with a focus on supporting lightweight publish/subscribe messaging protocols such as MQTT [11, 12]. The platform is built on open-source hardware development boards such as Arduino/Genuino. This Smart City learning platform is designed to enable the collaborative development of systems, applications and services for the cities of the future, and it should be efficient enough to support the learning of all the necessary skills required for the development of complex and large-scale systems. The architecture of the proposed platform (hardware and software components of the learning system) as well the experience with its usage, are presented in this article.

HARDWARE COMPONENTS OF THE LEARNING PLATFORM

Figure 1 shows the scheme of the learning system called LearnMQ. The system has six hardware components. These components include a laptop, an access point and three sensor nodes. All devices in the system support the IEEE 802.11 technology for networking. The laptop has a built-in wireless interface, and the other devices use ESP8266 communication modules. The function of the hardware components is described in this section.

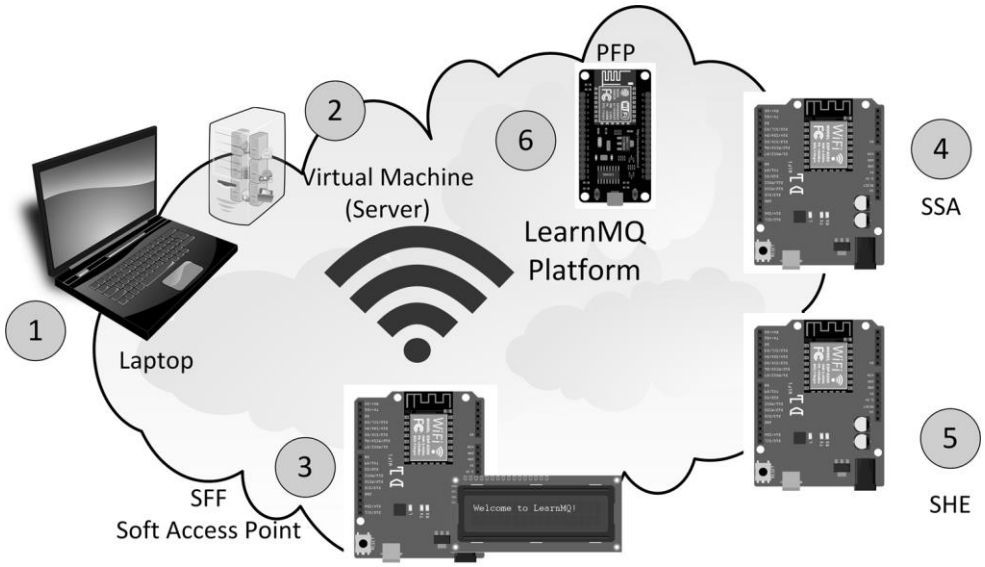

Figure 1. The topology of the learning system.

The first system component is a laptop (1). The laptop has Windows 10 operating system and the following software installed: Oracle VM virtual box as virtualization software, Python 3.6 for application development and Wireshark as the network traffic analysing software. The role of this device is to be one of the wireless clients in the network, and to host Python publishing and subscribe applications. The laptop also hosts an SQLite database for sensor data storage. The second component installed on the same device is a virtual machine (2). This is a CentOS 7.4 server with Mosquitto broker installed. It has the role of a publish/subscribe broker server. The virtual machine uses virtual Wi-Fi interface in bridged mode with fixed IP address 192.168.100.100 needed for the Mosquitto MQTT broker. All the other devices use dynamically allocated IP addresses assigned by the access point (3). The

utilization of the Wi-Fi interface is essential for this platform since that interface is used for transferring the data from all sensor nodes to the MQTT broker and subscriber application. The same interface is used for analysing the network traffic with Wireshark.

The third component (3) is designed as an access point. It is an Arduino/Genuino UNO clone with an integrated ESP8266 communication module called Wemos D1 R2. This module enables communication by using the IEEE 802.11 technology. This device is designed to have the role of the access point. In order to provide the feedback information for users, the I2C LCD 20x4 is attached to it. Its role is to act as the access point in the wireless network and to allow connection for other wireless clients. This device also has the role of a DHCP server.

The devices (1), (4), (5) and (6) use the access point to connect to the network. The LCD is used to show the number of devices connected to the network, and the information that can be used for other client devices to access the network, such as the network name (SSID) and passphrase. The IP address of the access point (192.168.100.2) is displayed as well. The problem with the usage of Wemos D1 R2 or any similar device based on ESP8266 as the access point is that there is a limitation to connecting up to four wireless clients. The default setting of the device is limited to four clients, although, according to the documentation, there is a possibility to extend this limitation up to eight devices. For the present research, the authors did not make a change to increase the limit of connected devices. As an alternative, an Android smartphone as a Wi-Fi hotspot can be used. The Android smartphone also has a limitation of allowing up to eight connected devices per network. The second alternative is to use the real access point as a dedicated device or a device that is part of the infrastructure. In order to make this learning platform easy to set up and portable, the Wemos D1 R2 or Android smartphone is considered for usage. For this particular research, only the Wemos D1 R2 is used. The laptop with a Python publishing script simulates the fourth sensor node with stationID TIS. The laptop does not have a sensor attached to it, and the sensor values are generated randomly.

Three devices (4), (5) and (6) represent wireless sensor nodes. They are built on Arduino/UNO clone devices. Each device is built upon a different clone type. The node with stationID SSA is based on Wemos D1 R2 (4), the node with stationID SHE (5) is based on Espduino and the node with stationID PFP (6) is based on NodeMCU. Each node has an attached gas sensor MQ-135 for air quality monitoring. The sensors are not calibrated, and sensor accuracy is not important in this case. These sensors are used to generate data to be transferred (published) to the message broker. The devices use MQTT protocol for publishing sensor data. The Arduino IDE is used for device programming with the usage of additional MQTT library. All three devices can be programmed using the same IDE code (with minor changes) and libraries.

SOFTWARE COMPONENTS OF THE LEARNING PLATFORM

The software architecture of the system is based on the microservices model, and it is presented in Figure 2. The figure describes six main software components of the learning platform. The first presented component of the system (1) is the MQTT broker. The Mosquitto software is used as the MQTT broker. It is installed on the previously described CentOS server, deployed on the virtual machine. The Mosquitto server listens to port 1883 for accepting publishing messages. All messages containing sensor data and additional information are sent using the MQTT protocol and are directed to my_queue.

The sensor nodes (2), (4), (5) and (6) send sensor data via the MQTT protocol to the Mosquitto broker. First node (2) uses Python script to simulate sensor data. The other three nodes (Arduino/Genuino UNO clones) are prototyped wireless sensor stations that send gas

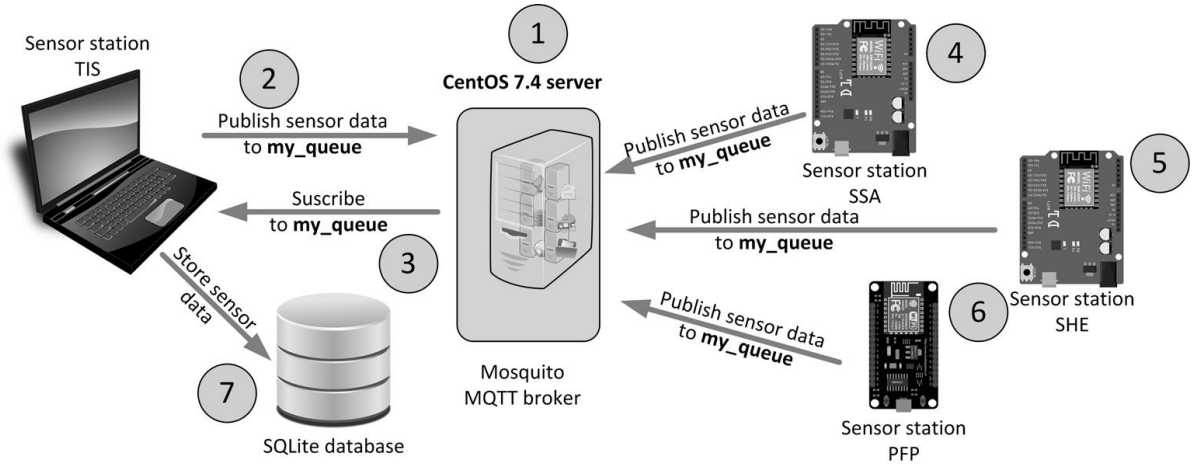

Figure 2. The architecture of the learning system.

sensor data. Data acquisition is enabled by a non-calibrated MQ-135 gas sensor designed for air quality monitoring. Since the purpose of the presented platform is not environment monitoring in real systems, the sensor data are sent in raw format. The value of the data ranges from 0 to 1023 and represents an analogue value read with the Arduino function analogRead(). All nodes have a stationID consisting of three letters. The three nodes have following IDs respectively: SSA, SHE and PFP. The laptop has stationID TIS. All four nodes send the data in CSV format. The message begins with the stationID, followed by a comma and the sensor value, e.g., DHE,748. In this research no particular attention is paid to define the format of the message. So the message length is 8 bytes. Three bytes are used for the stationID, one byte for comma, and four bytes for sensor values. The MQTT message has a total length of 16 bytes, 8 bytes for sensor node data with the addition of 8 bytes for the name of the queue (my_queue).

The software component (3) is a Python script that enables subscription to the MQTT broker and the data sent to my_queue. Besides the subscription to the sensor data published in my_queue, the script displays the received messages on a computer screen and stores the received data in an SQLite database (7). The SQLite database is used in this architecture because of its simplicity, but for the future usage, document-oriented database systems such as MongoDB or time series database (TSDB) such as InfluxDB will be considered.

EXPERIENCES WITH THE PLATFORM

The platform has not yet been validated in the teaching process. However, platform validation was carried out in the testing phase with experimental runs. During these experimental runs, the platform proved to be stable and worked without problems. The platform is easy to set up, and the devices connect in the ad-hoc wireless network without a problem. After powering on all devices, the platform starts to work automatically, since the Mosquitto broker runs automatically on server start, and sensor nodes automatically attempt to assign to the Mosquitto broker. After successful assignment to the broker, each sensor node immediately starts to send sensor data in five-second intervals. The example of messages received with the subscription script is displayed on the laptop screen in the corresponding form (Figure 3).

Traffic analyses, which can be used to teach students the main principles of publish/subscribe protocols, and network data analyses are performed with Wireshark – widely used network analyzing software. Wireshark has supported the MQTT protocol since version 1.12.0. The platform in operating mode during the testing period is presented in Figure 4.

```
Received message: TIS, 392
Received message: SHE, 214
Received message: TIS, 990
Received message: SSA, 350
Received message: PFP, 528
Received message: TIS, 990
Received message: TIS, 987
Received message: SHE, 222
Received message: SSA, 272
Received message: PFP, 515
 Received message: TIS, 85
```

Figure 3. Received messages displayed by the subscriber.

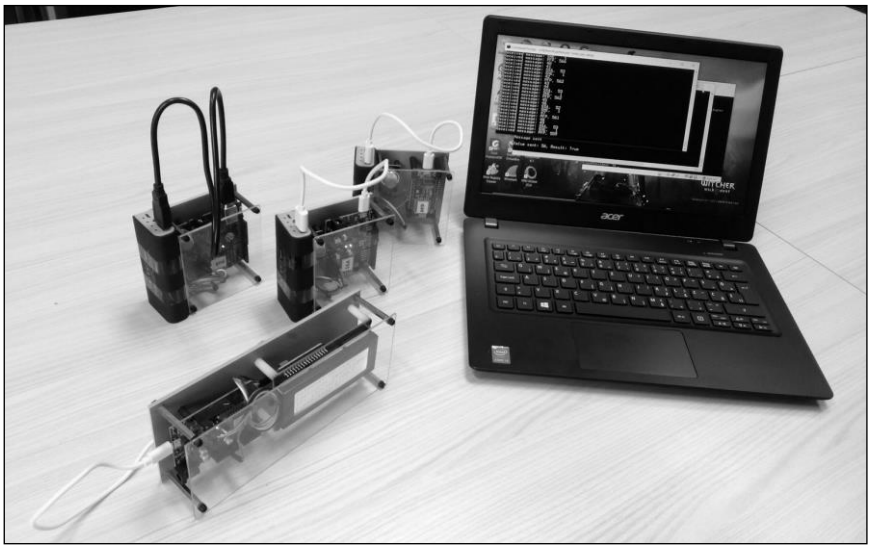

Figure 4. The learning system in testing period.

The platform was in test usage four times for about five hours. During this period, the platform worked without any problems. Based on the operation of the platform during the testing phase, it can be concluded that the platform is stable and it works correctly. Regarding the easiness of using the platform, it was concluded that the platform is suitable for usage in the classroom, for demonstrating the functioning of publishing/subscribe protocols, as well as for the students' collaborative work on the development of microservice-based applications.

CONCLUSION

This article presents a portable and lightweight learning platform for teaching IoT or publishing/subscribe protocols. This platform is focused on teaching and explaining the main principles of the the MQTT protocol, but it can be expanded with other similar protocols as well (AMQP, XMPP, etc.). The platform is designed to be used in the engineering education process within university curricula in order to enable the students to learn IoT technologies. This platform can be used for teaching microservice-based application development for the IoT. The presented platform is entirely based on open-source hardware and software, which makes this platform low-cost and easily applicable.

Further research will be conducted in three principal directions. The first direction is the expansion of this platform in order to support a more extensive range of protocols, primarily the AMQP (with server such as RabitMQ) and XMPP (with server such as Prosody). The usage of other databases such as MongoDB and InfluxDB will also be also considered. In order to use the platform in the learning process, educational materials and tutorials should be provided. The lack of tutorials is one of the main reasons why the platform has not yet been

evaluated in the learning process. After the completion of the tutorials and the design of laboratory exercises, the next step will be the experimental usage of the platform in the learning process. Finally, an effort will be made to adopt this platform to be used as a testing model for real-world IoT application development.

ACKNOWLEDGMENT

This research is supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia under the project number TR32044 "The development of software tools for business process analysis and improvement" 2011-2019.

REFERENCES

[1] Krylovskiy, A.; Jahn, M. and Patti, E.: *Designing a Smart City Internet of Things Platform with Microservice Architecture*.
Proceedings of 3rd International Conference on Future Internet of Things and Cloud. IEEE, Rome, 2015,
http://dx.doi.org/10.1109/FiCloud.2015.55,

[2] Cherradi, G.; EL Bouziri, A. and Boulmakoul, A.; Zeitouni, K.: *Real-Time Microservices Based Environmental Sensors System for Hazmat Transportation Networks Monitoring*.
Transportation Research Procedia **27**, 873-880, 2017,
http://dx.doi.org/10.1016/j.trpro.2017.12.087,

[3] Del Esposte, A.M.; Kon, F.; Costa, F.M. and Lago, N.: *InterSCity: A Scalable Microservice-based Open Source Platform for Smart Cities*.
Proceedings of the 6th International Conference on Smart Cities and Green ICT Systems. Science and Technology Publications, Lda, Porto, 2017,
http://dx.doi.org/10.5220/0006306200350046,

[4] Flammini, F.: *Resilience of Cyber-Physical Systems: From Risk Modelling to Threat Counteraction.*
Springer, Cham, 2019,
http://dx.doi.org/10.1007/978-3-319-95597-1,

[5] Del Espostea, A.M., et al.: *Design and evaluation of a scalable smart city software platform with large-scale simulations.*
Future Generation Computer Systems **93,** 427-441, 2018,
http://dx.doi.org/10.1016/j.future.2018.10.026,

[6] Dizdarević, J.; Carpio, F.; Jukan, A. and Masip-Bruin, X.: *A Survey of Communication Protocols for Internet-of-Things and Related Challenges of Fog and Cloud Computing Integration.*
ACM Computing Surveys **51**(1), No. 116, 2019,
http://dx.doi.org/10.1145/3292674,

[7] Kayal, P. and Perros, H.: *A comparison of IoT application layer protocols through a smart parking implementation*.
20th Conference on Innovations in Clouds, Internet and Networks (ICIN). IEEE, Paris, 2017,
http://dx.doi.org/10.1109/ICIN.2017.7899436,

[8] Kashyap, M.; Sharma, V. and Gupta, N.: *Taking MQTT and NodeMcu to IOT: Communication in Internet of Things.*
Procedia Computer Science **132**, 1611-1618, 2018,
http://dx.doi.org/10.1016/j.procs.2018.05.126,

[9] Barbon, G.; Margolis, M.; Palumbo, F.; Raimondi, F. and Weldin, N.: *Taking Arduino to the Internet of Things: The ASIP programming model*.
Computer Communications **89-90**, 128-140, 2016,
http://dx.doi.org/10.1016/j.comcom.2016.03.016,

[10] Prada, M.A., et al.: *Communication with resource-constrained devices through MQTT for control education*.
IFAC-PapersOnLine **49(**6), 150-155, 2016,
http://dx.doi.org/10.1016/j.ifacol.2016.07.169,

[11] Komkrit Chooruang, K. and Mangkalakeeree, P.: *Wireless Heart Rate Monitoring System Using MQTT*.
Procedia Computer Science **86**, 160-163, 2016,
http://dx.doi.org/10.1016/j.procs.2016.05.045,

[12] Hillar, G.C.: *MQTT Essentials – A Lightweight IoT Protocol*.
Packt Publishing Ltd., Birmingham, 2017,

[13] Malić, M.; Dobrilović, D. and Malić, D.: *Predlog arhitekture sistema za podršku bežičnim senzorskim mrežama zasnovanog na mikroservisima*. In Serbian.
XXXVI Simpozijum o novim tehnologijama u poštanskom i telekomunikacionom saobraćaju. Beograd, 2018.