Research Article

Khoa Dang* and Igor Trotskii

Architecture For Automation System Metrics Collection, Visualization and Data Engineering – HAMK Sheet Metal Center Building Automation Case Study

<https://doi.org/10.1515/eng-2019-0065>

Received Aug 28, 2019; accepted Sep 29, 2019

Abstract: Ever growing building energy consumption requires advanced automation and monitoring solutions in order to improve building energy efficiency. Furthermore, aggregation of building automation data, similarly to industrial scenarios allows for condition monitoring and fault diagnostics of the Heating, Ventilations and Air Conditioning (HVAC) system. For existing buildings, the commissioned SCADA solutions provide historical trends, alarms management and setpoint curve adjustments, which are essential features for facility management personnel. The development in Internet of Things (IoT) and Industry 4.0, as well as software microservices enables higher system integration, data analytics and rich visualization to be integrated into the existing infrastructure. This paper presents the implementation of a technology stack, which can be used as a framework for improving existing and new building automation systems by increasing interconnection and integrating data analytics solutions. The implementation solution is realized and evaluated for a nearly zero energy building, as a case study.

Keywords: Building automation, IoT, anomaly detection, analytics

1 Introduction

The research originated from the needs to open up building automation system (BAS) data from a near-zero energy building (nZEB) SCADA for a wide audience, comprised of different technical disciplines. The first solution,

implemented in 2017 was a monolith server application with time series database, REST API and visualization web application [3]. The solution had several drawbacks and caused inconvenience in development and management due to the monolithic nature. Furthermore, the first implementation limited the deployment opportunities, as the only supported orchestration was a virtual machine (VM) on either public or local cloud. Finally, it was only possible to scale the solution vertically, as in upgrading the hardware to achieve better performance. Nevertheless, it provided a solid starting point for later development and resolved the original legacy SCADA connectivity issue.

The proposed architecture in this paper was achieved by the formulation of the following research questions:

1. *Based on the same technology stack, how could the solution be scaled horizontally to meet with modern operation environment e.g. public and private cloud?*
2. *How to break apart the monolith, hence enabling the continuous integration of newly developed features with least effort?*
3. *Given the HVAC process data, which analytics techniques should be used to generate insights?*

To address questions one and two, containerization of the system and redesigning the system to microservices architecture was carried out. Practically, the first step is to replace the binary components with their containers equivalent and specify the connection scheme. This step was achieved fairly easily as the whole software industry largely adopted Docker and container orchestration in 2018, as can be seen with the support from public cloud e.g. Azure and Google Cloud Platform (GCP), resulting in the release of a containerized release of every component in the existing system. In addition, message queues, specifically MQTT, are embraced to replace the REST API as the field data ingress interface as well as serving the internal communication of the whole stack. The publisher – subscriber model of MQTT, as well as the WebSocket transport layer, allowed for the field data to be used also in custom-

*Corresponding Author: Khoa Dang: Häme University of Applied Sciences; Email: khoa.dang@hamk.fi

Igor Trotskii: Häme University of Applied Sciences; Email: igor.trotskii@hamk.fi

made Progressive Web Applications, therefore allowed the different possibilities for visualizing the process data for the general public audience.

First attempts to tackle the third research question started with the experimentation of different neural network models using the BAS data to forecast the energy consumption of the nZEB in a thesis project [14]. Further experiments with principal component analysis (PCA) and k-means clustering revealed possibilities for remote observation of the geothermal heat pump operation. Further literature reviews in process data analytics [8] and machine learning applications for buildings [13] pointed towards two-step PCA and k-Shape clustering for working with process automation time series. Hence, the two analytics methods were chosen to be implemented as analytics example of this research.

The proposed technological stack in the architecture consists of OPC-UA as industrial communication protocol for efficient machine-to-machine data transmission on the field level, combined with Node-RED with OPC-UA package for simple interconnection between different software interfaces. For transmission from multiple fields and reusability of data, Node-RED also performs packaging and sending data through MQTT to a private broker. Following this on the server side, time series storage and analytics software, represented by InfluxData's time series platform, are used for data ingress, preprocess and warehousing. Grafana is used for generating dashboards to perform preliminary inspection and production of visualization elements e.g. charts, gauges and metrics overview tables. Grafana also supports exporting CSV files from built elements for further analytics with Python such as feature extraction and anomaly detection, which supports the condition monitoring and condition-based maintenance processes. Finally, Docker is used for delivery and management of all components at their respective level.

Reasons for selection of aforementioned technologies include their open-sourced nature, reproducibility and adaptability. OPC-UA is widely adopted by industrial manufacturers nowadays and could be implemented in existing programs with minimal efforts, allowing for operation data extraction from field devices. As all the used software solutions are containerized, the connection from the field can easily be realized by deployment of gateway containers, i.e. Node-RED, on capable PLCs or SCADA computers. Similarly, the server-side stack is easily reproduced by deployment of server components on the IT infrastructure. Aside from the long list of connectable data sources, Grafana supports integration with different identity and access management solutions such as OAuth and LDAP, allowing for information isolation and customized access

for different personnel levels in enterprise environment. Analytics microservices built from Python allows for extensive feature extraction, classifications and clustering on collected building automation data to classify operation modes and identify anomalies where the system is not operating in designed regimes. The analytics results can then be illustrated on Grafana to present the information to process operators and maintenance staffs to perform cause analysis in a timely manner. Finally, the framework is implemented in a modular manner, allowing for adoption of better technologies when available and flexible deployment environments, as containerized applications are widely supported by cloud service providers nowadays.

2 SCADA, historians and the analogy to current software offerings

This section deals specifically with SCADA historian databases and trend graph, as well as how current time series database and dashboard platforms could be used to achieve similar functionalities, richer visualisations and more flexibility.

Bangemann *et al.* [2] described the software architecture of SCADA with the following components and features:

1. RTU devices – communication between control room and field/plant
2. Databases – historians for process data and general relational database for system operation
3. HMI and management interface – information gateway for human operators

Typically, the user interface presents the process data to human operators in terms of live subprocess data on HMI screens, short-term trends and alarms in hazardous conditions. The database usually is SQL relational database, due to their proven resilience and long history. The main drawback is SQL relational database often needs to be carefully optimized for large and fast data ingress, often leading to limitation in data storage frequency and duration. Trend displays often employ proprietary native client software or occasionally a web client, with basic plotting capabilities from the raw data and limited analytics of such data.

On the other hand, current time series database (TSDB) offerings found their basis in software system performance metrics and financial analysis applications.

Thanks to advances in computer hardware and new programming paradigm, these offerings often are capable of high volume data ingress, more data processing built into the query language, much lower disk usage footprint and more diverse interfaces for data access e.g. REST API and clients for integrating with different programming languages. Compared to a relational SQL database where data downsampling and discarding are achieved with custom triggers, these features are built-in and quite often automatically enforced by default in TSDBs. Another difference worth mentioning is horizontal scaling for better performance and redundancy is offered out-of-the-box by almost every TSDB vendor, whereas with relational SQL databases the solution will vary between each software integrators that built the information system. Finally, functions such as minimum, maximum, average, etc. are built in to the query language of TSDBs, allowing for more efficient data querying and less custom programming [1]. Thanks to the rich interfaces provided by TSDBs, visualizations tools offerings also diversified in both commercial (Tableau, PowerBI) and open-source non-commercial (Grafana, Kibana) solutions. Often build as progressive web applications, these solutions allows multiple users to build their own visualization according to their own needs, exporting the data as the user sees them and ultimately engaging users to inspect time series in an explorative manner. These offerings also provides organizational identity management integrations, reducing the workload of enterprise system administrators and support easier day-to-day usage of the users.

It can clearly be seen that by connecting process data to TSDBs, the data could be stored at longer durations and higher frequency, while providing easier access to external application such as visualization, analytics and reporting.

3 Microservices and containerization

Modularity provides numerous benefits:

1. Modularity increases reliability of the system *i.e.* when some fault occurs, only small part of the system is involved and so other services are still operational.
2. Modularity provides some level of abstraction. It is not necessary to know how other parts of the system work. It is enough for developer to know what the inputs and outputs are of the certain service to utilize it in efficient way.

3. Containerization significantly simplifies deployment of the whole system. With proper orchestration and development practice, it is possible to have continuous integration and delivery, which helps to keep the system in up to date state.

Microservice architecture separates a monolith application into multiple components called microservices, each running as their own and communicate with each other through a request-response model or an event broker. Each software developer team is responsible for one microservice from the technical aspects and business requirement, to the operation and continuous operation of said services [6]. Containerization takes advantage of the virtualization technologies, enabling the packaging and deployment of applications on different platforms with minimal customization. In short, a containerized application comprises of a base operational system layer and other application packages, combined with description for mapping of the container's storage space to the host's storage space. Containers cluster orchestration, such as Docker Swarm and Kubernetes, also allows the developer to specify the network connectivity and physical resource allowance for each containers [10].

Applying microservices architecture and containerization, one approach is to divide the previous solution into components based on their functionality. The results are as following:

1. Database: InfluxDB
2. Data receiver: NodeRED and Telegraf
3. Data output and REST API provider: NodeRED
4. Analytics and data preprocessing: Kapacitor
5. Message broker: Mosquitto
6. Custom API integration: NodeRED flows and custom NodeJS apps
7. Visualization: custom web apps and Grafana

Each of the components already has containerized release from their developers, allowing for the migration process to be less problematic. Docker provides round-robin load balancing for containers of the same type with same description on top of a server load balancing by NGINX webserver.

As the platform deals with different physical plants and organizations, at the migration time a question was raised whether different clusters should be deployed for different organizations. The option would provide perfect data isolation, yet increase the complexity in managing different clusters and add extra overheads in running all clusters. For the time being, as all the data in the platform serves public research, only one cluster is used with the

Figure 1: Architecture description – from field to cloud

organization isolation provided at the database and visualization levels. In details, each organization and research project is given own database in the InfluxDB container, and own Grafana organization.

4 From field to cloud – OPC, fieldbuses and gateways

In this research, gateways are implemented on three main targets:

1. Industrial PC PLCs running Windows or Linux
2. Raspberry Pi for communicating low-power sensors using local LoRaWAN or EnOcean
3. Linux VMs inside enterprise network for external API integration and offloading for industrial PCs

Different levels of the architecture, as well as the corresponding communication protocol at each levels is described in Figure 1. The language of choice for developing gateway application is NodeJS, due to the strong support for asynchronous programming and community support for different application. In addition, NodeRED is a simple

visual programming tool which allows for intuitional programming, smoothing the learning curve for new developers to get started with system integration.

In general, the architecture was designed collect data from different sources *e.g.* OPC and REST, then wrapping those to MQTT to lower the computation overhead and enable flexible information routing. The architecture employs a private MQTT broker, in this case Mosquitto, yet cloud alternative such as Azure IoT Hub could be used also. In addition, MQTT also support WebSocket transport, allowing web developers to leverage process data for experimenting with different visualization applications. Finally, both the private broker and public cloud offerings supports encryption and device identification, allowing secure communication and device management. Lower field level device and fieldbus can be integrated through the PLC program, or directly in the case of Modbus TCP. JSON is chosen as the markup language, with minimal message object containing measurement ID, value and source timestamp, quite often other information such as geographic location and device status is included.

Currently the usage of OPC UA in the architecture is limited to data access and device diagnostics status. As

per the industrial R&D practice reflected in [4, 5, 7, 12], three conclusions can be drawn. One, the development of OPC UA applications will be significantly eased in the near future with companion specifications, allowing the gateway development process to be generalized and less case-specific. Second, direct field device integration with OPC UA server will extend the gateway with less effort down the field level, possibly eliminating the PLC for pure monitoring application. Finally, OPC UA PubSub could possibly replace the gateway application in the future. Although this remains ambiguous due to OPC being popular in the process industry yet fairly unexplored in the embedded devices and low power sensor network. In addition, the generic programmers and web developers might not be familiar with OPC UA, which negatively affects their learning curve and prototyping speed.

5 Data analytics

Collected process data does not provide any significant value by itself, as it is usual for a real system to have more than a hundred of measurements. In order to get valuable insights from such amounts of data it is necessary to have automatic monitoring tools, which can interpret and analyze streamed measurements in real-time.

This paper describes two powerful tools for time series data analysis: two-step principal component analysis and k-Shape clustering, and provides results of their comparison to more usual techniques, *e.g.* traditional principal component analysis (PCA) and k-means clustering.

5.1 Two-step principal component analysis

Traditional PCA is based on transforming original highly-dimensional process data consisting of correlated variables into a smaller set of uncorrelated variables. The uncorrelated variables or principal components can be considered as linear combination of original variables. This method tries to maximize retained information from original dataset, however it does not handle autocorrelation between variables, meaning it is only gives best performance for processes in stationary state [16].

In order to overcome aforementioned problem, new method called two-step principal component analysis (TS-PCA) was used for anomaly and fault detection [8]. TS-PCA first evaluates dynamic properties of the process by constructing a linear model for the process:

$$(t) = X(t-1)A_1 + X(t-2)A_2 + \dots + \quad (1)$$

$$+ X(t-q)A_q + U(t) = \tilde{X}(t)\tilde{A} + U(t) \quad (1)$$

where $X(t)$ is row of measurements at timestamp t , $A(i)$ parameters of linear model, used to approximate dynamic part, $U(t)$ is a disturbance at timestamp t and q is the time lag. With such approximation the model can be regarded as q -order auto regression model, which can be evaluated by least squares algorithm. Disturbance part then can be calculated as follows:

$$\hat{U}(t) = X(t) - \tilde{X}(t)\tilde{A} \quad (2)$$

$U(t)$ is not time depended and so can be used to evaluate performance of the studied process by utilizing standard PCA metrics, *i.e.* Hotelling's T^2 and SPE. Hotelling's T^2 provides the variations of each sample in the PCA model *i.e.* variations in process variable that do not change the nature of the process operation. On the other hand, SPE measures the variation of process variable in the residual subspace, which can be understood as the change in system characteristics. T^2 and SPE time series could then be monitored and evaluated using hypothesis testing to indicate anomalies. Detailed calculations and explanation of the metrics are presented in [8, 9].

5.2 k-Shape clustering

Clustering allows the separation of continuous stream of data into different groups and can be used for automatics discovery of various building operations modes or separating normal operation from anomalous. However, most common techniques, *e.g.* k-means and similar clustering algorithms have several issues related to time series nature of data in building automation and maintenance domains:

1. k-means does not consider auto-correlation part, meaning the assigned class of each measurement based only on that measurement, although, measurements with same values can have completely different implication based on previous state. It is possible to use different distance metrics for k-means in order to solve that problem, but it leads to another issue.
2. Most of distance measures suitable for time series data are domain specific and require careful tuning, what partially removes the benefits unsupervised learning usually provides.

k-Shape is a novel technique for time series clustering. Unlike k-means, it allows to process time series data almost without prior knowledge about the process and it considers not only the values of the measurements but also their arrangements or shapes during label assignment [11].

Table 1: Variable description for analytics algorithms

Variable name	Description	Type
JN01LM01_FE	Flow measurement from heat well	Continuous
JN01LM01_T1	Temperature measurement from heat well	Continuous
JN01LM01_T2	Temperature measurement from heat well	Continuous
JN01LM01_Tdiff	Temperature difference from heat well	Continuous
JN01LM02_FE	Flow measurement from energy piles	Continuous
JN01LM02_T1	Temperature measurement from energy piles	Continuous
JN01LM02_T2	Temperature measurement from energy piles	Continuous
JN01LM02_Tdiff	Temperature difference from energy piles	Continuous
LP01LM01_FE	Flow measurement from heatpump output	Continuous
LP01LM01_T1	Temperature measurement from heatpump output	Continuous
LP01LM01_T2	Temperature measurement from heatpump output	Continuous
LP01LM01_Tdiff	Temperature difference from heatpump output	Continuous
JN01LM01_QE	Heating power from heat wells	Composition
JN01LM02_QE	Heating power from energy piles	Composition
LP01LM01_QE	Heatpump heat powet output	Composition

k-Shape clustering can be used for energy usage patterns [15], process operation mode discovery and anomaly detection. TS-PCA is able to accurately evaluate probability of anomalous behavior by checking measurements streams in real-time, whereas k-Shape can compare different time intervals, *e.g.* hours, days or weeks, and find out unexpected variabilities in data, which can indicate unwanted deviation.

5.3 Integration with Microsoft Azure and local cloud deployment

TS-PCA and k-Shape clustering are used as separate services and to further support the idea of complete modularity of the solution, containerized. Designed services can be deployed in any environment with docker container support.

There are two main deployment targets for the services: local machines and Microsoft Azure. Integration with Microsoft Azure provides such services as time series insights, databases and event hubs. While the designed system is self-sustainable and does not require any Azure services, adoption of at least event hub and further integration with Office 365 suite drastically simplifies notification of the personnel, generated by data analysis tools.

On the other hand, the local implementation is easily achievable with Python sklearn, tslearn and paho-mqtt packages. Currently, these Python microservices are pre-trained with historical data from processes of interests.

Then, the real-time process measurements and analytics results are emitted from the microservices using MQTT.

6 Analytics results

The training dataset consisted of 15 variables, listed in Table 1. The dataset was collected from October 2018 to February 2019 with 10 seconds sampling period and resampled at 1 minute for training TS-PCA. k-Shape was trained on data collected from March 2018 to the end of March 2019.

6.1 Using TS-PCA for improving heat pump operation

TS-PCA was able to detect anomalies in both real-time streaming data and historical data. Hotelling's T^2 statistic was able to detect a major deviation shown on Figure 2. in energy flows, caused by appearance of solar energy production. Solar power production is represented by JN02LM01_QE and heat flow from the heat well is represented by JN01LM02_QE. At the time the anomaly was recorded, the heat pump was operating in a winter mode and so it was completely unexpected to get negative heat flow from the heat well.

The deviation can be explained by the fact that training data did not include any warm sunny days. Therefore, the trained model did not expect solar power production and considered it as an anomaly. The interesting part is that anomaly was captured by T^2 statistic, meaning that

Figure 2: Deviations in energy flows, caused by solar energy

Figure 3: Anomalies captured by SPE metric

the anomaly was captured in terms of retained principal components, thus TS-PCA is able to interpret energy flows inside the system even with new disturbances added.

The second anomaly was captured by SPE metric, when the detection algorithm was tested against historical data. The main idea was to check if it is possible to detect periods of poor operation of the heat pump, which occurred prior to maintenance period between

05.11.2018 and 20.11.2018, and undocumented setpoint change, which was performed on 22.11.2018. The results are shown on Figure 3. The setpoint change is represented by the biggest spike in SPE and poor performance is shown in both high T^2 and SPE values during October.

Figure 4: Distribution of clusters based on month. March contains double amount of labels as it was presented in two years: 2018 and 2019

6.2 k-Shape based anomaly detection

The dataset used for training k-Shape clustering algorithm was separated into one day long sequences of equal lengths of 1440 data points, *i.e.* 1 point per minute. The algorithm was able to separate sequences based on their operation mode: summer or winter. Distribution of clusters based on months is shown on Figure 4.

As illustrated in the clustering results, k-Shape can be used for anomaly detection. If a winter operation sequence was detected during summer or vice versa, then there is a high chance of anomaly. This idea is proven by distribution of clusters in November, when heat pump was offline for 15 days due to maintenance and so the heat flows were rather similar to the heat flows during the summer, what explains abnormally high number of summer labels for this month.

7 Implemented architecture at HAMK

The structure of implemented solution is a complex system of different services presented on Figure 1. Such structure is necessary to accommodate vast diversity in used data sources. So far, the data coming into the platform has been coming from PLC, low power sensors and third-party APIs.

Figure 5 presented the data flow in the implemented architecture at HAMK and Fig. 6 further presented the communication scheme at the field level. In order to be able to use as many data sources as possible without the need to modify existing applications and services, three major communication protocols are used on two different levels: OPC-UA for site level communication between various ma-

Figure 5: Architecture of the designed system

Figure 6: Main communication protocols and their usage

chines located relatively close to each other and MQTT as a main communication protocol between devices and services located on different sites. HTTP protocol is used as auxiliary protocol as to have a possibility to get access to third-party devices and services.

As OPC-UA is only used for local communications, it is necessary to transform it to MQTT by passing signals through gateways, *e.g.* Raspberry Pi or PLC itself. Gateway runs a simple node.js or Node-RED program, which listens to OPC-UA messages and sends corresponding MQTT messages. Using gateways helps to format, select and transport the data the user can be interested in and in a way most suitable for further processing. Currently, gateway implementation are done with NodeJS or NodeRED, on Linux VMs, Pi and Windows runtime of PLCs. Conversion of lower fieldbuses are done in the IEC 61131-3 runtime and data is brought over as PLC variables, or in the gateway implementation in cases where the data is not critical for process operation and is available through IP-based fieldbuses *e.g.* ModbusTCP.

On the plant, one Beckhoff PLC is used for collecting building automation data from legacy Pyramid SCADA and provide lighting control of the nZEB hall as well as electricity quality monitoring. The gateway application is deployed in a VM on HAMK production network for collecting data from the building automation PLC and other machinery in the industrial hall.

Figure 7: Example of Grafana dashboards

MQTT messages are used in two different ways: they are consumed by Telegraf module and then passed to InfluxDB or byservices, which require real-time access to the data streams, e.g. online anomaly detection, Azure Event Hubs for further user notification through integration with Office 365. Telegraf is a server agent used by InfluxDB to simplify data consumption. It helps to gather information from such data sources as MQTT, third-party APIs and servers' metrics. History data from InfluxDB can be easily visualized by means of Grafana. Grafana is integrated with Active Directory for simplified user management through LDAP integration. An example dashboard for presenting the heating consumption of the building is presented in Figure 7.

8 Conclusions

The paper presented the architecture for collecting, visualizing and applying data analytics for automation systems. The collected data is able to be visualized for different audiences, and analytics applied have resulted in change of heat pump operation, leading to better operation efficiency of the whole energy system.

In the near future, the continuing development of OPC UA specification as well as other software will be taken into account as the platform develops. Companion specifications is expected to standardize and simplify the data ingress part of the architecture. On the other hand, more

machine learning and analytics will be tested and optimized to further benefit the building automation system.

Acknowledgement: This research was supported by the research project "Terveellinen Digitalo", funded by European Regional Development Fund under grant number A72991.

References

- [1] Bader, A., Kopp, O., & Falkenthal, M. (PDF) Survey and Comparison of Open Source Time Series Databases. Retrieved February 4, 2019, from https://www.researchgate.net/publication/315838456_Survey_and_Comparison_of_Open_Source_Time_Series_Databases.
- [2] Bangemann, T., Karnouskos, S., Camp, R., Carlsson, O., Riedl, M., McLeod, S., ... Stluka, P., State of the Art in Industrial Automation. In *Industrial Cloud-Based Cyber-Physical Systems* (pp. 23–47), 2014. https://doi.org/10.1007/978-3-319-05624-1_2.
- [3] Dang, K., *Industry 4.0-ready Building Automation System - System design and commissioning for HAMK Sheet Metal Center building*, B.Eng. thesis, Häme University of Applied Sciences, 2017. Retrieved from <http://www.theseus.fi/handle/10024/139881>.
- [4] Faath, A. (n.d.-d). *VDMA supports developing OPC UA CS*. Retrieved from https://www.automaatioseura.fi/site/assets/files/1824/03_opc_finland_vdma_andreas_faath.pdf.
- [5] Karaila, M. (n.d.-e). *Valmet DNA OPC UA Server & Client*. Retrieved from https://www.automaatioseura.fi/site/assets/files/1824/13_dna_opc_ua_server_ua_client_for_customers_-_mika_karaila.pdf.

- [6] Koschel, A., Astrova, I., & Dotterl, J., Making the move to microservice architecture. *2017 International Conference on Information Society (i-Society)*, 74–79, 2017. <https://doi.org/10.23919/i-Society.2017.8354675>.
- [7] Laubenstein, A. (n.d.-g). *OPC UA and Field Device Integration (FDI) for Industrie 4.0*. Retrieved from https://www.automaatioseura.fi/site/assets/files/1824/04_laubenstein_opc_day_tam pere_2018_v01_opcf.pdf.
- [8] Lou, Z., Shen, D., & Wang, Y., Two-step principal component analysis for dynamic processes monitoring. *The Canadian Journal of Chemical Engineering*, 96(1), 160–170, 2018. <https://doi.org/10.1002/cjce.22855>.
- [9] Mujica, L., Rodellar, J., Fernández, A., & Güemes, A., Q-statistic and T2-statistic PCA-based measures for damage assessment in structures. *Structural Health Monitoring: An International Journal*, 10(5), 539–553, 2011. <https://doi.org/10.1177/1475921710388972>.
- [10] Pahl, C., Brogi, A., Soldani, J., & Jamshidi, P., Cloud Container Technologies: a State-of-the-Art Review. *IEEE Transactions on Cloud Computing*, 1–1, 2017. <https://doi.org/10.1109/TCC.2017.2702586>.
- [11] Paparrizos, J., & Gravano, L., k-Shape: Efficient and Accurate Clustering of Time Series. *ACM SIGMOD Record*, 45(1), 69–76, 2016. <https://doi.org/10.1145/2949741.2949758>.
- [12] Peltola, J. (n.d.-l). *OPC UA in the Real World Introduction*. Retrieved from https://www.automaatioseura.fi/site/assets/files/1824/12_opc_ua_in_the_real_world_jukka_peltola_v2.pdf.
- [13] Seyedzadeh, S., Rahimian, F. P., Glesk, I., & Roper, M., Machine learning for estimation of building energy consumption and performance: a review. *Visualization in Engineering*, 6(1), 5, 2018. <https://doi.org/10.1186/s40327-018-0064-7>.
- [14] Trotskii, I., *Heating Energy Consumption Forecasting Based on Machine Learning*, B.Eng. thesis, Häme University of Applied Sciences, 2018. Retrieved from <http://www.theseus.fi/handle/10024/150183>.
- [15] Yang, J., Ning, C., Deb, C., Zhang, F., Cheong, D., Lee, S. E., ... Tham, K. W., k-Shape clustering algorithm for building energy usage patterns analysis and forecasting model accuracy improvement. *Energy and Buildings*, 146, 27–37, 2017. <https://doi.org/10.1016/j.enbuild.2017.03.071>.
- [16] Yin, S., Ding, S. X., Xie, X., & Luo, H., A Review on Basic Data-Driven Approaches for Industrial Process Monitoring. *IEEE Transactions on Industrial Electronics*, 61(11), 6418–6428, 2014. <https://doi.org/10.1109/TIE.2014.2301773>.