Next generation control units simplifying industrial machine learning

Stefano De Blasi
Bosch Rexroth AG
Lohr am Main, Germany
Stefano.DeBlasi@boschrexroth.de

Elmar Engels
UAS Fulda
Fulda, Germany
Elmar.Engels@et.hs-fulda.de

***Abstract*—The increasing amount of sensor data creates new possibilities through data-driven projects in industry. The demands on the final solution architecture are different and typically include at least one controller (e.g. PLC). In this paper, we focus on these industrial controllers and identify their possible roles in smart factories: Collection, distribution and pre-processing of field data, execution of intelligence and functionalities to ease rapid prototyping approaches. Furthermore, we specify the capabilities leading to the fulfillment of these roles and present an application example of drive anomaly detection to demonstrate how to setup an industrial controller for machine learning projects.**

***Index Terms*—controllers for machine learning, industrial automation, real-time system**

I. Introduction

In a modern factory, large amounts of data are generated and collected, which requires optimized data processing to control production processes in a more efficient way [1]. Some complex systems even generate gigabytes of data per second [2]. The increasing scope of sensor data opens up new opportunities to use previously unknown information for benefits such as improved process quality or avoided downtime. To find this information or generate logic based on data, methods of data mining or machine learning are increasingly used in industry [3].

The processing of industry data for machine learning includes data formatting, feature extraction, dimensionality reduction, and performance evaluation before applying a model, e.g., for a prediction task [4]. These tasks vary by project and have different requirements for the final application. One major component for industrial automation is the programmable logic controller (PLC), which controls the actuators of a process based on the sensor signals in real-time. However, the architecture for data-driven real-time reactions is a very controversial issue [5].

In this paper, we identify the possible roles of control units in smart factories and specify the capabilities that should be additional included for the next generation of PLCs so that data scientists can develop and include their industrial solutions best possible and efficient. First, we cover the general current trend in PLCs and then discuss capabilities in following groups: Data collection and distribution, pre-processing of data, execution of intelligence and functionalities to ease rapid prototyping approaches. Finally, we present an application example of drive anomaly detection to demonstrate how to setup a control unit for machine learning projects.

II. Evolution of PLCs

PLCs provide an industrial Real-Time Capable Operating System (RTOS) to control an automated process in guaranteed reaction time. The increasing number of integrated sensors in the production line and regular innovations in data analysis through data-driven algorithms intensify the need for a collector and distributor of field data. While some older PLCs can be connected to a hardware module called an Internet of Things (IoT) gateway that acts as a distributor for the cloud, modern PLCs already provide an optional feature for this purpose without any additional hardware.

For real-time sensitive applications, latency via cloud computing is unacceptable in many cases, so there is a trend towards decentralized edge computing to optimize the data processing [5]. The once clearly defined role of an industrial controller is now increasingly blurring with that of an industrial computer. As a result of this trend, PLCs are increasingly integrated by manufacturers on open platforms.

A. Software flexibility

The usage of Docker enables the Container as a Service (CaaS) principle, which lead to isolated environments for optional software functionality [2]. Linux Containers (LXC) are also mentioned here as an alternative to Docker to provide lightweight containers. Such software containers for industrial control units increase the flexibility of future automation systems [6]. For example, the PLC functionality could be installed as a container on the open platform. This type of PLC is known as soft PLC in contrast to the inflexible hardware based PLCs.

Furthermore, real-time control applications based on containers are also valuable for the reusability and portability of solutions [7]. For example, by installing software modules on different devices it could be possible to provide functionality centrally on fully autonomous control units but also as distributed functionality with an edge server [2]. The illustration in Fig. 1 indicates the variety of such possible software modules. An analogy to extending capabilities by installing modules is, for example, smartphone applications. Some of

978-1-7281-5635-4/20/$31.00 ©2020 IEEE

these functionalities also depend on specific hardware. To enable real flexibility, the control unit manufacturer is therefore required to provide adequate hardware modules. As a result of this architecture it is possible to modify the control unit so that it is well suited for multiple tasks.

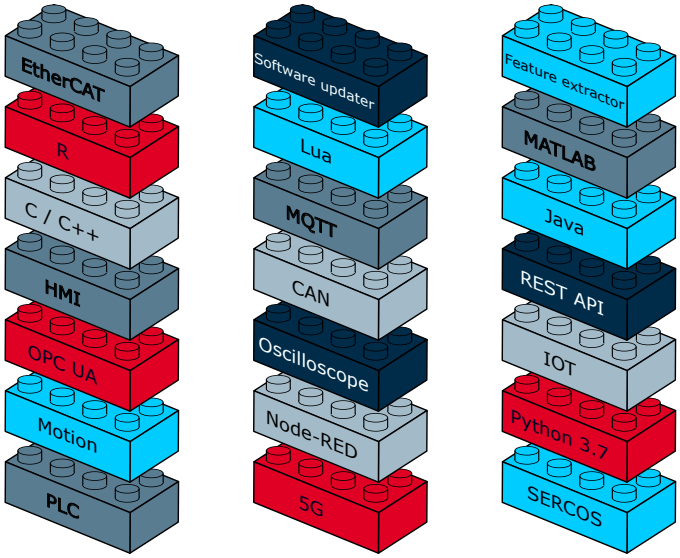

Fig. 1. Several exemplary software modules of a control unit. Some modules may have specific software or even hardware dependencies. Brick colors illustrate diversity.

B. Programming language flexibility

Data scientists are commonly proficient in at least one high-level programming language such as Python, R, C++, Java or MATLAB. Conventional PLCs on the other hand use special programming languages such as function block diagram defined according to IEC 61131-3. The compliance with this IEC standard guarantees a real-time behavior that cannot be ensured with native Python, for example. Furthermore, they are designed in such a way that they are easy to understand even without any programming knowledge. However, some modern control systems can also be programmed in adapted C++ or Java to promote an open platform. This can be realized, for example, via a Motion-Logic Programming Interface (MLPI) [8] for different programming languages.

An open platform enables a wide range of functions to be used for the decentralization of different machine learning tasks directly on the control unit. Depending on the application, it should be possible to extend the functions with user-defined libraries from the common toolbox repositories. Based on the wide open-source community, Python is one of the most popular programming languages, especially for data scientists. By building its own toolboxes, MATLAB has also become an useful programming language for machine learning, which is particularly popular among engineers.

III. Next Generation Capabilities

The computing power of hardware components has been increasing rapidly for decades. To benefit from this evolution, the focus should be on developing scalable software architectures that minimize the effort and time to market for future generations of devices. In this work we focus on simplifying the daily work of data scientists in an industrial context. Thereby we use the principle of cross-industry standard process for data mining (CRISP-DM) [9] to derive the most important capabilities for working on machine learning projects. These capabilities will ultimately lead to smaller software modules like described in Sec. II-A to enable custom project work. We do not focus on the intelligence for image processing, as this increasingly takes place in programmable cameras with specialized hardware and only their results are passed on as sensor data. The usual field data (e.g. sensor data) in automation results from one-dimensional sensor signals, which lead to time series data by recording. In general, next generation control units should be able to fulfill one or more of the following roles during a data driven project:

- collector and distributor of field data,
- pre-processor of field data,
- executor of intelligence,
- auxiliary device for rapid prototyping.

The abilities listed below are only to be understood as a supplement to the abilities of standard PLCs to fulfill these roles. For example, the provision of an RTOS or the use of safety protocols is a basic requirement.

A. Collector and distributor of field data

Currently the most common role of a PLC in machine learning projects is to access the field data and to create the training data set. This requires, on the one hand, general capabilities for the acquisition of field data and their intermediate storage on the controller and, on the other hand, communication capabilities regarding sensors and other network components.

1) Recording data: In the following, the software functionality to temporary store field data is called oscilloscope. To enable qualitative analysis of single-channel time series data, a good recording is defined by maintaining a constant and sufficiently high sampling rate in addition to the measuring accuracy. If triggered immediately, the oscilloscope should not lose any value between the end of the first recording and the beginning of the following recording. The oscilloscope functionality is getting more complex for multi-channel recordings. Here, a shared time stamp is essential to combine signals for more dimensional information. By using the same time stamp, several control units are able to collect data for a cross-machine purpose. Furthermore, enabling asynchronous recording is desirable for multiple data acquisition with different uses in parallel.

In some applications, the sensor signals are previously recorded continuously and the recording is then divided into process sequences in order to compare them with historical data. This segmentation of time series signals usually requires engineering effort or problem specific algorithms [10], [11]. By labeling the sequences automatically with the executed PLC commands, the signal processing complexity and the calculation effort can be reduced. Referencing to the mentioned

cross-machine purpose, this triggering of the oscilloscope functionality should also be able by external sources like a second PLC.

2) Communication: In reference to the intended CaaS principle for control units, the installed modules must be enabled for data exchange. The core functionality of the controller should include an access point for reading and writing data in different programming languages to support a diversity of software modules. Furthermore, by supporting security protocols this access point could be used for the communication between different control units. By providing a message broker, the core functionality enables event-based module interactions.

Open Platform Communications Unified Architecture (OPC-UA) is an interoperability protocol standard for secure and reliable data exchange regarding machine-to-machine communication in industrial automation. There are also other protocol standards such as MTConnect, which are designed to provide easy access to process data. In order to enable projects involving the use of components from different manufacturers, it is essential that the control unit supports these protocols. Another important standard is the IEEE Time-Sensitive Networking (TSN), which is one of the enabler technologies for many applications within a smart factory. Here, the end-to-end latency between two components can be guaranteed, which ensures the industrial real-time requirements. The combination of OPC-UA and TSN is therefore of great interest and a challenging task [12]–[14], leading to the fulfillment of real-time, flexibility, safety and security features within the smart factory [15].

In combination with the fifth generation of cellular networks, commonly known as 5G, TSN will be a key technology [16]. Industrial controllers will mainly collect and distribute data to a centralized server, processing the data and returning decisions to the controllers with a guaranteed low latency. This evolution will also increase the importance of direct data streaming abilities of control units.

Another emerging trend are sensors with embedded hardware like microprocessors. These so-called smart sensors are capable of pre-processing data locally and streaming it event-based via message queue broker. For instance, Message Queue Telemetry Transport (MQTT) is a popular protocol especially for communication between devices. For safety reasons, at least version 5.0 of MQTT must be used to transfer data outside the factory. Alternatively, Advanced Message Queuing Protocol (AMQP) can be used [17].

For the sake of completeness, fieldbus communication is still the backbone of industrial automation, especially for complex actuator and sensor signals. The support of several common fieldbus systems such as Ethernet for Control Automation Technology (EtherCAT), Serial Realtime Communication System (Sercos) or Profinet additionally increases flexibility also with regard to component vendors.

B. Pre-processor of field data

Before field data can be used as input for models, it often has to be prepared. This preparation step can include the filtering of raw data and the extraction of features. The control unit should be able to fulfill this role for two cases. First, decentralized calculations of field data are used to reduce network congestion, as it is not recommended to send the whole raw sensor data to the cloud [17]. Second, whenever the control unit executes a model (see Sec. III-C), the correct input is required directly on the device.

1) Filtering: The provision of parameterizable digital filter options for time series signals such as low pass filtering for noise reduction is essential for further processing. Especially for large industrial environments interference signals can lead to significant manipulation of data. While basic digital filters are easy to design and implement, MATLAB and the Python library SciPy provide powerful support for nonlinear filter design [18]. Thus, the control unit can easily be equipped with complex filters by providing a runtime for such third-party functionalities.

2) Extracting features: Time series signals of sensor data acquired by the PLC mostly require more complex pre-processing in addition to filtering for machine learning approaches. For example, a bearing damage could only be determined via vibration sensor signals in the time and frequency domain together [19]. Therefore, the time series signal from the vibration sensor are processed to obtain the complex features within the signal, e.g. mean value, variance, coefficient of skewness or C factor. This task is called feature extraction and there are widely used toolboxes in different programming languages like Python [20], [21], R [22] or MATLAB [23].

C. Executor of intelligence

In general, knowledge about the maximum processing time of the model execution is necessary to guarantee real-time behavior. Furthermore, this execution is callable from the soft PLC to include the intelligent decisions into the automated process. Running models on the edge device requires model input to be provided by the device itself or by other network components. For instance, fulfilling the capabilities in Sec. III-B2 allows the use of locally extracted features as model input.

We distinguish between three use cases for the inclusion of intelligence in automation technology. First, using a server for the machine learning model. Here, continuous data must be transferred to the server and when the model response is needed, it is sent backwards. The control unit only performs as a collector, distributor and possibly also as pre-processor of field data. Popular examples are planning tasks or component lifetime prediction (predictive maintenance). Secondly, expensive trained models are executed on the control unit on the basis of the collected data and can be directly integrated into the control loop. The additional capabilities for this are discussed in Sec. III-C1. Third, the control unit performs cheap, iterative machine learning. See Sec. III-C2 for the capabilities in this case.

1) Deploying models: To easily import models trained by users with different preferences for machine learning environments such as scikit-learn, MATLAB, PyTorch, the import has to be consistent. This motivation led to the development of the Open Neural Network Exchange (ONNX) format [24] for trained machine learning models. Since the hardware components of next generation control units should be application-dependent, the importance of performance portability across different hardware back-ends continues to grow. Here, an automated optimization compiler like TVM [25] could help to deploy trained models to different hardware platforms, also with accelerators like Field Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs). The machines within a factory change over time, which can negatively affect the accuracy of a trained model and is a known problem for machine learning [26], [27]. Dealing with these changes remains a challenge to enable robust, continuous lifelong learning [28]. An automated update strategy for retrained models can therefore ensure the best possible performance based on the available data [29]. This recurring training is performed on a central server (edge or cloud depending on the use case).

In summary, an uniform import of models with measured model execution time for the applied hardware configuration and an automated update feedback loop over newly acquired data is desirable.

2) Training models online: Another approach of adapting a model to changes over time is iterative online learning. The Smart Factory vision for the future includes intelligent machining processes that will ultimately lead to self-optimization and adaptation to uncontrollable variables [30], based on real-time sensor data to ensure the best possible operating performance [31]. This self-learning of system behavior during operation requires an online learning environment with low decision latency for intelligent parameterization. Most often the machine learning projects are based on toolboxes without modifying the basic algorithms. A control device should therefore support the widely used toolboxes for machine learning, usually based on Python, MATLAB, C/C++ or Java. Since secure and robust functionality must be guaranteed in industry, the inclusion of novel, self-developed algorithms should be well promoted, as this is an ongoing topic in research. Such algorithms should include a conservative, safe exploration and a good exploitation without fatal errors [32]. While the iterative learning step must be performed between two decisions, the intelligent decision time is required on demand. Both latencies have to meet the requirements of the use case to ensure real-time behavior.

D. Auxiliary device for rapid prototyping

According to the principle of CRISP-DM, solutions are based on iterative work in which steps are repeated until the result is satisfactory. Online visualization or event-based plotting during the execution increases the understanding of data and the process. To enable iterative jumps between the implementation steps in the simplest possible way, one also wants to provide interactive adjustment feedback via the dashboard, which can be used for hyperparameter tuning, for example. By hosting a custom dashboard, the next generation of control units should be able to support this iterative workflow. An obvious variant would be Node-RED, as it is widely used for dashboard prototyping in combination with the REST API. With its visual, flow-based development environment, it gained popularity through numerous private Internet of Things projects. Users can also develop their own function modules, so-called nodes, and make them freely available. Due to its flexibility, Node-RED is also becoming increasingly attractive in the industry. For instance, Node-RED can be used as a basis for a complete industrial communication strategy with different protocols like Modbus [33]. Alternatively, plotly [34] can be used for dashboard hosting with advanced plotting, including options for using MATLAB graphics.

IV. Exemplary ML project

Based on the proposed recommendations, the work on an exemplary ML project is conceptualized using a next generation control unit. Here, a fully self-sufficient solution (without cloud usage) for a drive anomaly detection of a Cartesian robot is realized. For this use case, we select the illustrated software complexity in Fig. 2 for the used control unit.

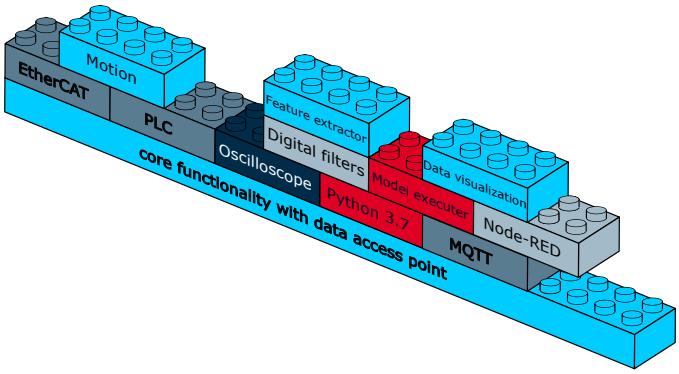

Fig. 2. Exemplary software complexity of a control unit. Based on the core functionality with shared data access point, the user can add optional modules and exclude not needed modules to obtain the best performance with minimal complexity. Bricks represent modules and stacks indicate dependencies of functionality for the overall solution. Brick colors illustrate diversity.

Since container dependencies are generally not CaaS-compliant, the specified dependencies refer to the functionality for the overall solution. For example, EtherCAT, soft PLC and the motion kernel functionality are required in interaction to communicate with and control the drives. Of course a different protocol can be used instead of EtherCAT depending on the drives. According to the strict CaaS principle, each of the four Python-dependent modules (digital filters, feature extractor, model executer and data visualization) should provide its own Python interpreter. However, it is controversial in this context because of the multiplication of the memory used. A Python 3.7 module is shown for illustration purposes, but strict CaaS conformance is recommended whenever possible.

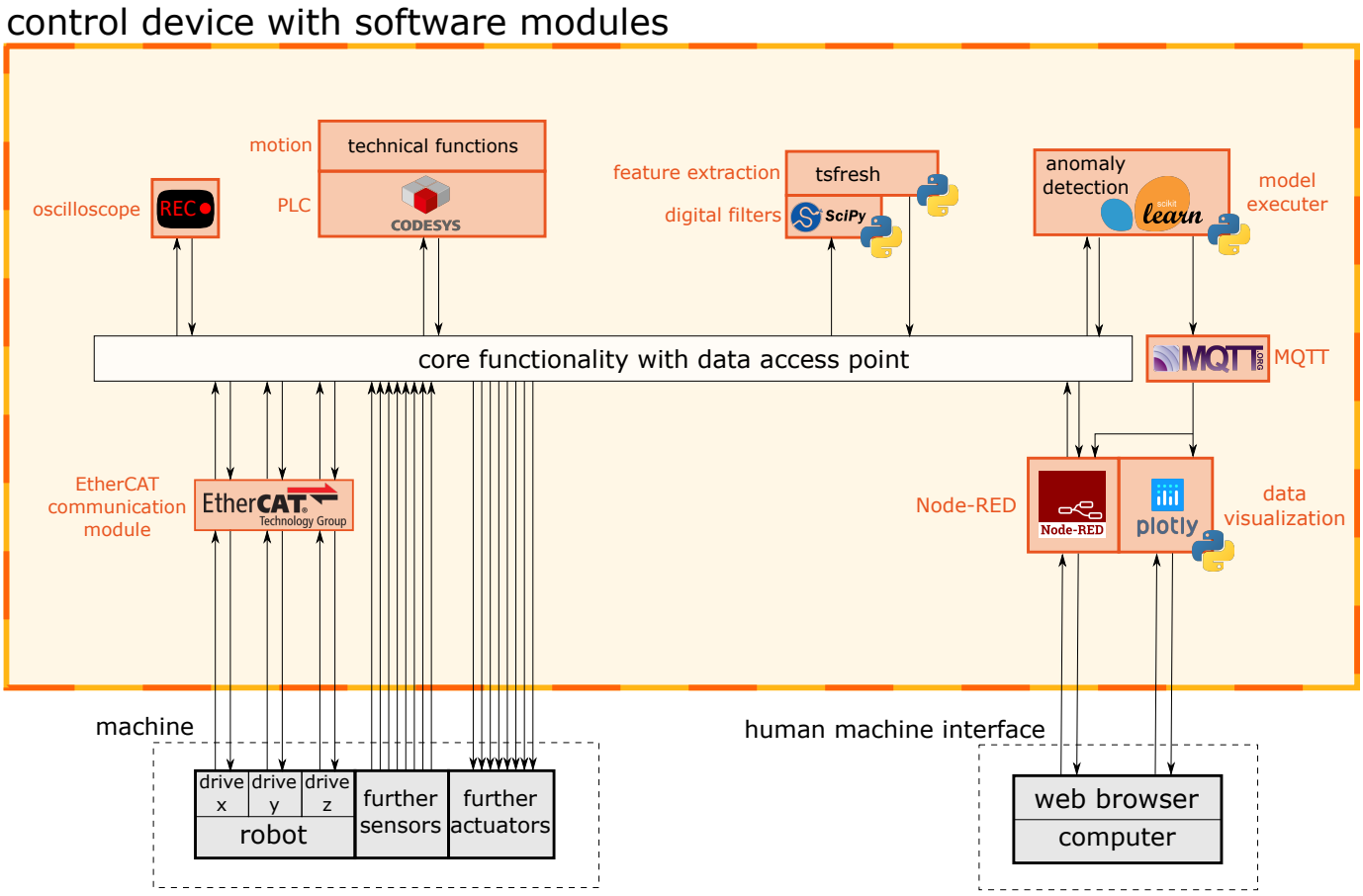

Fig. 3. Exemplary software architecture for a drive anomaly detection project. The information flow is represented by arrows and software containers as orange boxes.

The solution architecture shown in Fig. 3 illustrates the underlying information flow within the control device. Commands and data between the control unit and the drive are exchanged by EtherCAT communication. Additional sensors (e.g. buttons and security system) and actuators (e.g. indicator lights) are connected to a standard I/O system. The drive is controlled by the motion kernel functionality in the soft PLC, for example based on Codesys (development environment for IEC 61131-3 programming languages). During movement, the drive signals are recorded by the oscilloscope functionality and then stored to the central data access point. The data is pre-processed with Python library (e.g. SciPy) based digital filters before the feature extraction, which is based on another Python library (e.g. tsfresh [20]). These calculated features are used as input for the trained model or an online training. This could be done with one of the many freely available Python-based machine learning libraries (e.g. scikit-learn). Via a message broker (e.g. MQTT) the newly occurring results are transferred and cause the interaction between result values and the interface. Normally the internal message broker of the core functionality can be used instead of MQTT, but it was not fully implemented and ready for use at the time of the project. The models output is visualized with some additional feedback functions on a Node-RED dashboard with plotly [34] functionality (hosted from Python). These additional features are used by data scientists to further improve model accuracy by testing the detection online following the principle of CRISP-DM.

We implemented the exemplary ML project on a ctrlX CORE of Bosch Rexroth AG as a proof of concept. Digital filters were not required, but are presented to emphasize the scalability of complexity with future-oriented control units that meet the capabilities of Sec. III.

V. Conclusion

The layered data analysis architecture presented in [35] includes a data acquisition layer in which pre-processing tasks such as filtering or feature extraction can be performed. Instead of the entire time series, only the relevant extracted features could be transferred to the upper data queue layer. Finally, the data processing layer uses the data and includes the execution of logic or trained models. As presented in this paper, the next generation of control units should be able to perform not only one but, depending on the use case, several layers in one device. Furthermore, the identified roles (collector, distributor and pre-processor of field data as well as for the final intelligence executions) can be divided among

several control units for cross-machine solutions. We also recommend using the next generation of industrial controllers for rapid prototyping with live plotting and feedback strategies to accelerate process understanding for data scientists.

Since the requirements within a smart factory are constantly changing due to advancing research results, adaptability of the system architecture is crucial. In particular, the successful combination of 5G and TSN will open up new possibilities, underlining the need why software flexibility is the key feature of next generation control units.

REFERENCES

[1] H. S. Kang, J. Y. Lee, S. Choi, H. Kim, J. H. Park, J. Y. Son, B. H. Kim, and S. Do Noh, "Smart manufacturing: Past research, present findings, and future directions," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 3, no. 1, pp. 111–128, 2016.

[2] V. Gezer, J. Um, and M. Ruskowski, "An extensible edge computing architecture: Definition, requirements and enablers," in *UBICOMM 2017: The Eleventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2017.

[3] Z. Ge, Z. Song, S. X. Ding, and B. Huang, "Data mining and analytics in the process industry: The role of machine learning," *IEEE Access*, vol. 5, pp. 20 590–20 616, 2017.

[4] J. Yan, Y. Meng, L. Lu, and L. Li, "Industrial big data in an industry 4.0 environment: Challenges, schemes, and applications for predictive maintenance," *IEEE Access*, vol. 5, pp. 23 484–23 491, 2017.

[5] S. Trinks and C. Felden, "Edge computing architecture to support real time analytic applications: A state-of-the-art within the application area of smart factory and industry 4.0," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 2930–2939.

[6] T. Goldschmidt and S. Hauck-Stattelmann, "Software containers for industrial control," in *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Aug 2016, pp. 258–265.

[7] T. Tasci, J. Melcher, and A. Verl, "A container-based architecture for real-time control applications," in *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, June 2018, pp. 1–9.

[8] E. Engels and S. Krauskopf, "Innovation in motion-logic programming - a versatile interface," in *Proceedings to the 12th International Workshop on Research and Education in Mechatronics*, 2011.

[9] R. Wirth and J. Hipp, "Crisp-dm: Towards a standard process model for data mining," in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*. Citeseer, 2000, pp. 29–39.

[10] T. Borgi, A. Hidri, B. Neef, and M. S. Naceur, "Data analytics for predictive maintenance of industrial robots," in *2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*. IEEE, 2017, pp. 412–417.

[11] D. Pantazis, P. Goodall, P. P. Conway, and A. A. West, "An automated feature extraction method with application to empirical model development from machining power data," *Mechanical Systems and Signal Processing*, vol. 124, pp. 21–35, 2019.

[12] A. Eckhardt and S. Müller, "Analysis of the round trip time of opc ua and tsn based peer-to-peer communication," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2019, pp. 161–167.

[13] A. Gogolev, F. Mendoza, and R. Braun, "Tsn-enabled opc ua in field devices," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2018, pp. 297–303.

[14] D. Bruckner, M. Stănică, R. Blair, S. Schriegel, S. Kehrer, M. Seewald, and T. Sauter, "An introduction to opc ua tsn for industrial communication systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1121–1131, June 2019.

[15] F. Zezulka, P. Marcon, Z. Bradac, J. Arm, T. Benesl, and I. Vesely, "Communication systems for industry 4.0 and the iiot," *IFAC-PapersOnLine*, vol. 51, no. 6, pp. 150–155, 2018.

[16] S. Vitturi, C. Zunino, and T. Sauter, "Industrial communication systems and their future challenges: Next-generation ethernet, iiot, and 5g," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 944–961, 2019.

[17] V. Gezer, J. Um, and M. Ruskowski, "An introduction to edge computing and a real-time capable server architecture," *Int. J. Adv. Intell. Syst.(IARIA)*, vol. 11, no. 7, pp. 105–114, 2018.

[18] R. K. Pearson and M. Gabbouj, *Nonlinear Digital Filtering with Python: An Introduction*. Boca Raton: CRC Press, 2016.

[19] L. S. Dhamande and M. B. Chaudhari, "Compound gear-bearing fault feature extraction using statistical features based on time-frequency method," *Measurement*, vol. 125, pp. 63–77, 2018.

[20] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.

[21] D. M. Burns and C. M. Whyne, "Seglearn: a python package for learning sequences and time series," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 3238–3244, 2018.

[22] N. A. Roque and N. Ram, "tsfeaturex: An r package for automating time series feature extraction," *Journal of Open Source Software*, vol. 10, 2019.

[23] B. D. Fulcher and N. S. Jones, "hctsa: A computational framework for automated time-series phenotyping using massive feature extraction," *Cell systems*, vol. 5, no. 5, pp. 527–531, 2017.

[24] J. Bai, F. Lu, K. Zhang *et al.*, "Onnx: Open neural network exchange," *GitHub repository*, 2019.

[25] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze *et al.*, "{TVM}: An automated end-to-end optimizing compiler for deep learning," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 578–594.

[26] V. Agrawal, B. K. Panigrahi, and P. Subbarao, "Increasing reliability of fault detection systems for industrial applications," *IEEE Intelligent Systems*, vol. 33, no. 3, pp. 28–39, 2018.

[27] L. Baier, N. Kühl, and G. Satzger, "How to cope with change?-preserving validity of predictive services over time," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[28] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, 2019.

[29] H. Wang, Y. Yang, and Y. Song, "A general data renewal model for prediction algorithms in industrial data analytics," *arXiv preprint arXiv:1908.08368*, 2019.

[30] D.-H. Kim, T. J. Kim, X. Wang, M. Kim, Y.-J. Quan, J. W. Oh, S.-H. Min, H. Kim, B. Bhandari, I. Yang *et al.*, "Smart machining process using machine learning: A review and perspective on machining industry," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 5, no. 4, pp. 555–568, 2018.

[31] Y. Qu, X. Ming, Z. Liu, X. Zhang, and Z. Hou, "Smart manufacturing systems: state of the art and future trends," *The International Journal of Advanced Manufacturing Technology*, pp. 1–18, 2019.

[32] S. De Blasi, "Active learning approach for safe process parameter tuning," in *Machine Learning, Optimization, and Data Science*. Cham: Springer International Publishing, 2019, pp. 689–699.

[33] M. Tabaa, B. Chouri, S. Saadaoui, and K. Alami, "Industrial communication based on modbus and node-red," *Procedia computer science*, vol. 130, pp. 583–588, 2018.

[34] P. T. Inc. (2015) Collaborative data science. Montreal, QC. [Online]. Available: https://plot.ly

[35] R. S. Peres, A. D. Rocha, A. Coelho, and J. B. Oliveira, "A highly flexible, distributed data analysis framework for industry 4.0 manufacturing systems," in *International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing*. Springer, 2016, pp. 373–381.