Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 126 (2024) 111–116

www.elsevier.com/locate/procedia

17th CIRP Conference on Intelligent Computation in Manufacturing Engineering (CIRP ICME '23)

CNC Machine Tool Focused Edge Computing in Manufacturing

L. Tonejca (née Plessing)*, C. Mayer, T. Trautner, G. Mauthner, F. Bleicher

TU Wien, Institute of Production Engineering and Photonic Technologies, Karlsplatz 13, 1040 Vienna, Austria

* Corresponding author. Tel.: +43676844881108; *E-mail address:* tonejca@ift.at

Abstract

This paper presents a method for standardized integration of external sensors and actuators into CNC machine tool systems utilizing edge computing technology. Different communication protocols and various means for triggering external devices at a chosen machining operation were studied and evaluated using a sensory tool holder. The benefits of the presented architecture are the independence of the host operating system, easy customization, lightweight architecture, and standard protocols. Focusing on the demands for modern edge devices led to containerized modules with customizable interfaces. During manufacturing processes, the system was evaluated in terms of performance and reliability.

Keywords: Smart manufacturing; Digital twin; Edge computing; Dockerisation

1. Introduction

Trends such as Industry 4.0 and Cyber Physical Production Systems (CPPS) have led to the development of various sensor and actuator systems and respective communication protocols utilized in Computer Numerical Control (CNC) machining processes such as milling. These new systems provide large amounts of high-frequent process data points regarding tool vibration, temperature, or cutting forces, and can be used for process optimization purposes.

However, the integration and synchronization of sensor and actuator systems into commercial CNC systems is still a challenge for the manufacturing industry [1], [2] For data-driven process control, the sensors and actuators must interact at an application level (of the ISO-OSI model[3]) with the CNC. Retrofitting existing machine tools remains a difficult task requiring manual engineering effort due to the proprietary interfaces and the lack of standardization of individual control systems.

Problems may also occur when processing the vast amount of data produced by multiple sensors utilizing the computing resources of the CNC, especially if cameras or other sensors with high data rates are involved. Additionally, the fusion of raw sensor data and relevant CNC data, like axis positions, spindle speed, etc. may be a difficult task requiring additional manual engineering effort, thus, limiting the opportunities for data analysis and machine learning approaches [4]

Due to this increasing demand for a flexible integration of conventional CNC systems with new sensor and actuator technology as well as new needs regarding high-frequency data processing, edge computing increasingly becomes the focus of researchers and developers [5], [6].

With its capabilities to provide additional computation power and open microservice architecture, edge devices enable new opportunities for manufacturers to retrofit existing machines and integrate new sensor/actuator technologies [2]

10.1016/j.procir.2024.08.309

2. Related Work

Due to trends such as Industry 4.0 and the Industrial Internet of Things (IIoT), research has focused on industrial communication protocols as well as edge/cloud architectures in recent years. Historically, industrial applications utilize a vast variety of fieldbus systems such as CAN, Modbus, Profibus, Profinet or EtherCAT for integrating CNC controllers and its respective periphery such as programmable logic controllers (PLC), sensors and actuators [7]. Additionally, recent developments led to new TPC/IP based protocols acting on the application level such as CoAP, HTTP, AMQP, OPC UA and MQTT [8]. While some attempt to standardize industrial communication between different machine tools and its periphery, the integration of sensors remains a challenge [9].

With the convergence of IT (Information Technology) and OT (Operational Technology) edge devices come increasingly into focus. They provide CPPS with additional computing power and flexibility in terms of applications and functionality. Especially the concept of application containerization is a trend for enhancing easy upscaling and flexible adjustment of software code [10]. Current industrial edge device vendors, such as Siemens or TTTech, offer functionality regarding data connectivity, vertical integration of additional hardware as well as Docker environments for containerized applications.

The Siemens Industrial Edge for Machine Tools focuses on the need of domain specific manufacturers and aims a straight-forward integration with popular CNC systems. Providing a variety of connectivity protocols such as OPC UA, Modbus TCP or Profinet, it allows an easy integration of different heterogeneous field devices into the existing system. With docker application being implemented on the hardware operating system, it is possible to run applications written in high-level coding languages like C++ or Python [11]. In contrast, Nerve is an edge computing platform by the company TTTech. Additionally, own hardware with multiple ports to connect different sensors and actuators, as well as an I/O port to ensure Ethernet-based fieldbus connectivity is offered. Applications can be deployed as docker containers or utilize the integrated CoDeSys Runtime for Soft PLC applications [12].

Demonstrating a manufacturing use-case utilizing edge devices, Garcia et al. [2] proposed a micro-service architecture based on Docker container. A job manager service handles service requests, data acquisition and provision of results to the Programmable Logic Controller (PLC). However, the communication with the production system is reduced to a message from a placeholder and does not address the special case of CNC machines.

In [13], the authors demonstrate the integration of an external vibration sensor into the CNC control, enabling bi-directional communication using machine tool individual M-Commands and NC-Code comments, to trigger communication between the two separate systems. While communication between the sensor and the CNC has been implemented successfully, machine tool specific integration work had to be performed to provide the respective functionality.

Past research in the field of industrial communication protocols, edge computing, and containerization of micro-services highlights new opportunities for vertical integration in the industrial networks. While several relevant technologies have been discussed individually, current research indicates a lack of approaches for respective system integration especially in the machine tool and CNC machining domain.

3. Concept for flexible and portable sensor and actuator integration for CNC machine tools

This paper presents an innovative concept for flexible and portable integration of retrofitted sensor and actuator technology in modern machine tools. In this work, edge devices are used as standardized middleware for bi-directional communication between the sensors/actuators and the CNC system. First, proper design guidelines have been selected to specific requirements for the development. Second, the envisioned architecture is described. Third, an overview about the prototype development is given.

3.1. Design Requirements

To investigate the optimal edge architecture for retrofitting existing hardware, several requirements about the design of the envisioned system architecture need to be defined. The following requirements have been defined in order to design a applicable prototype for a wide group of manufacturers:

- No additional hardware other than a standard industrial edge device is needed to enhance retrofitting.
- Interaction between machine tools and external devices shall take place only at the application level.
- Adaptability of the application by the customer is necessary to react to changing needs such as new sensors or actuators.
- In general, high usability, manageability, and scalability is a must to enable a wide range of applications.

Considering the software and hardware requirements in manufacturing companies, several boundary conditions for the prototype development were defined and evaluated. *Table 1* illustrates the requirements for the architecture design and prototype development.

Table 1. Hardware and software requirements for prototype development.

	Edge device hardware requirements
Operating System (OS)	Likely OSs of modern edge devices are Linux-based variants or Windows distributions.
Control interface	Most controls are accessible through IP on top of Ethernet as a network access protocol.
Control protocol	Either a proprietary API or a standard communication interface, like OPC UA, MQTT, or Modbus, are available at the control for data exchange.
Control vendors	Popular control manufacturers for CNC machines are Heidenhain, Siemens and Fanuc.
Edge hardware	An optimal edge architecture allows a straightforward implementation with existing edge hardware.

	Functional requirements for the edge application
External connectivity	Connectivity at application level to the CNC machine and to retrofitted devices.
Internal connectivity	Connectivity from the machine tool to the sensors or actuators over the edge application.
Data streaming	Continuous data collection from the machine tool and retrofitted sensors.
Data processing	Onward streaming to an information distributor or storage of the collected data.
Flexibility	Flexibility in common machine tools, edge devices, controls, and OSs of edge hardware.
Programming language	Python is one of the most popular high-level programming languages amongst engineers and comes with a large community and open-source libraries [14].
Functionality distribution	As continuous integration and deployment (CI/CD) processes are essential for time and resource-efficient implementation, maintenance, and horizontal scaling, containerization is becoming popular in software development communities as an alternative to virtualization [15]. Docker is a containerization tool with a large community and a free repository with already compiled images of various applications.

3.2. System Architecture

The proposed architecture (*Fig. 1*) has been setup similarly to other existing middleware solutions e.g., the *fledge-iot architecture* [16], using southbound and northbound communication services, notification services for events and a local buffer. An industrial edge device is connected to the machine tool control as well as to the retrofitted sensors and actuators. The edge device provides additional computation capacity thus, the machine tool control is not overloaded with various data extraction requests.

A deployment platform is running on the edge device that hosts individual applications using an open micro-service architecture based on docker containers. This modularity ensures easy deployment and flexibility regardless of the machine tool control or sensor/actuator provider. Hence, the proposed architecture remains independent and flexible for various adaptions during the lifecycle.

For providing bi-directional connectivity between the machine tool control and the sensor/actuator system, two separate *communication modules* are developed, providing easy reconfigurability for different open or proprietary interfaces used. Each application module is written in Python and containerized via Docker.

The main task of the *Control Communicator Module (CC)* is to provide pre-defined trigger points integrated in the NC-Code to synchronize the bi-directional communication between the sensor/actuator and the numerical control. In the proposed architecture, these trigger points can be identified by the edge application during execution, triggering relevant actions such as start of a data gathering service or the activation of a feedrate optimization service on the machine control. A suitable trigger point is e.g., using the current tool number from the numerical control databus. In that case, the CC is frequently requesting the last used T-Command from the PLC memory providing the currently used tool identification number. In case of a tool change, the associated sensor- or actuator-ID is looked up in a configuration file, and respective actions such as sensor activation for data streaming can be executed.

To pass on the activation commands, the CC communicates with the *Sensor Communicator Module (SC)* via HTTP protocol over the internal Docker network. The SC is a HTTP-Server providing a REST-API based on the Python library *aiohttp* and performs three tasks: (I) Connection/disconnection of sensor devices, (II) start/stop the data stream of sensor devices and (III) providing subsequent data processing services with sensor data. Those three services run as asynchronous background tasks, triggered by HTTP requests. Once the communication is established, the data stream is initiated by sending another HTTP request to the SC. The proposed implementation uses standard ethernet based protocols which run on any Linux-based hardware for internal communication.

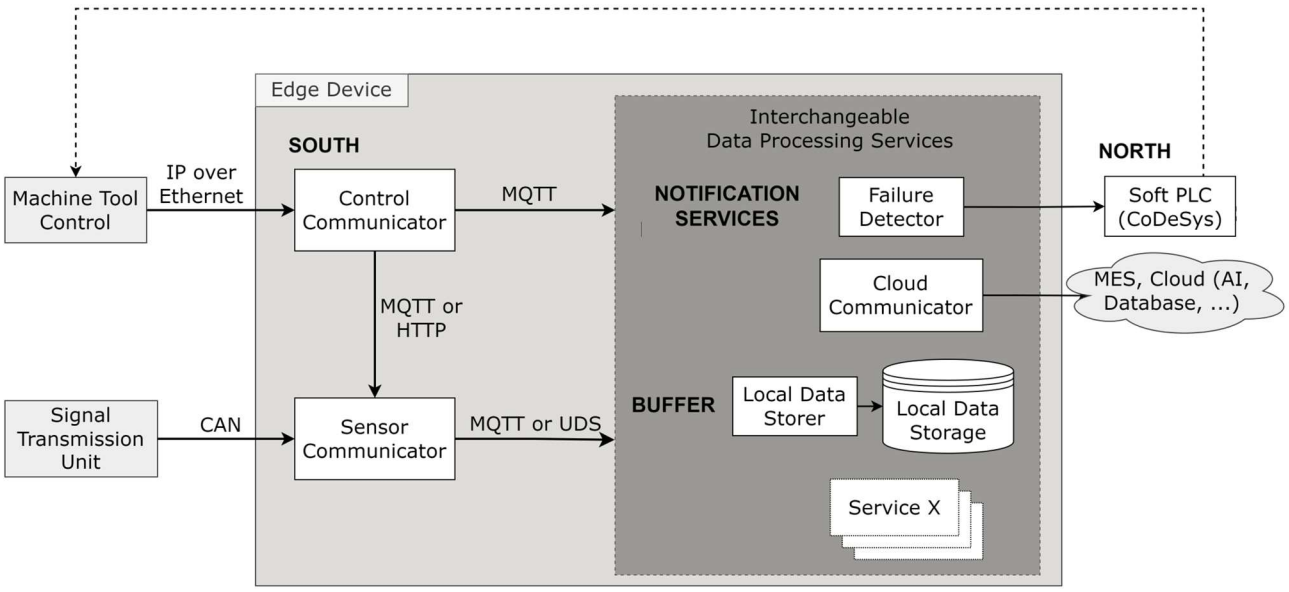

Fig. 1: Modular edge architecture for retrofitting an existing CNC machine with sensors and actuators.

Before collecting data, the connection status of the sensor is verified to avoid connection errors. By sending the respective HTTP request, the data collection terminates. Temporarily, the data is stored in a queue and provided to other containers or services via a UNIX Domain Socket (UDS). UDSs are standard components of UNIX systems and are used for Inter-Process Communication on the host [8].

The *Local Data Storer* reads data from the UDS to store it locally, while the *Cloud Communicator* prepares the data for cloud processing. The architecture allows the deployment of individual modules for further data processing tasks on the edge device e.g., a *Failure Detector Module*, which analyses the incoming acceleration data and triggers a warning, if a threshold is exceeded.

In this study, the interchangeable structure was organized by using a Docker-compose files which controls the installation of the application modules as Docker images.

4. Prototype Evaluation

4.1. Experimental Setup

The concept has been implemented as a prototype at the manufacturing laboratory of *TU Wien Institute for Production Engineering and Photonic Technologies* (IFT) in Vienna. To analyze and evaluate key design requirements such as flexibility and portability, different prototype versions have been implemented utilizing different machine tool manufacturers, CNC systems, communication protocols as well as edge device hardware. An overview about the different experimental setups used are shown in *Fig. 2*. Different variants of the prototypes are listed, beginning with the Operating System (OS), the industrial communication protocol, the NC manufacturer, and the method of triggering external devices from the machine tool. The vertical arrows indicate the different combinations used for the respective prototype.

The first four versions represent the evolving prototypes of the edge architecture. To evaluate its general usability, the following two versions are test runs of the 4th prototype on a different edge device (4*) and at a different machine tool (4**). All prototypes were tested on the DMU 75 Monoblock with a Fanuc control. The last prototype was also verified on the WFL M35 Millturn with a Sinumerik control.

Based on the Focas-API, Fanuc controls can be used with a proprietary OPC UA server which uses embedded Ethernet on the network access layer. The server is not preinstalled on the control but must be deployed retrospectively and at the moment only runs on Windows 10 or lower Windows Versions. Alternatively, the Focas-API can be implemented on any OS using a C-library for direct access of Fanuc NC variables. While Focas refers to the direct use of the C-library, a C-Wrapper provides the same API as Python functions. In contrast, the Sinumerik control has an OPC UA server implemented to begin with.

The sensor triggering and operation information was implemented since the 4th prototype as was the final containerized architecture. While the hardware in the first four prototypes is a laptop with 3,2GHz and 16GB RAM, the one for 4* was the Nerve MFN100 from TTTech with Debian as OS, and the one in 4** an Intel Nuc11.

For retrofitting demonstration purposes, an exemplary sensor system has been selected. The *Sensory Tool Holder* (STH) is a development of the IFT and utilizes an acceleration sensor integrated in the tool holder for high-frequent vibration measurements during the machining process. Via Bluetooth Low Energy, the sensor passes the data to a Signal Transmission Unit (STU), which communicates over a CAN-USB-Adapter with the edge device. The STU requires the tool-ID of a specific sensory tool holder as input to start the data stream, especially if more are in close proximity of the reading device. Thus, the correct input at the right time of the machining process is needed to ensure successful monitoring [17].

4.2. Results

The different prototypes have been implemented in the laboratory and were evaluated in experiments on CNC machines including the sensory tool holder at the application level. Evaluation questions proposed by Cabaj [18] considering aspects such as effectiveness, scalability or stakeholder support, have been utilized for systematic and comparable evaluation of the implementations.

The first prototype consists of two threads: one for communicating to the machine tool via OPC UA and the other to the sensor by activating the command line tool from within the program – starting the data collection of both systems through manual activation. The data stream over the OPC UA server yielded an update frequency of 9Hz for the transmission of the absolute position of the 5 axes: X, Y, Z, A, and C (20 Bytes in total), while the STH streamed data at 5 kHz.

Due to the C-bound API, the second prototype did not communicate to the machine tool and the sensor in a parallel Python program. This and the third prototype required manual STH activation. In the fourth and last prototype, a C-wrapper for Python provided the functions of the raw C-library as Python library but shows similar computational efficiency to pure C code. The fourth prototype consists of Docker containers and is managed by a docker-compose file that defines dependencies and starts the services accordingly.

As shown in *Fig. 3*, the data of the CNC and the STH are collected in parallel by asynchronous services of the prototype.

For the first variant of the 4th prototype (4* in *Fig. 2*), the edge hardware was switched to the Nerve MFN 100 of TTTech

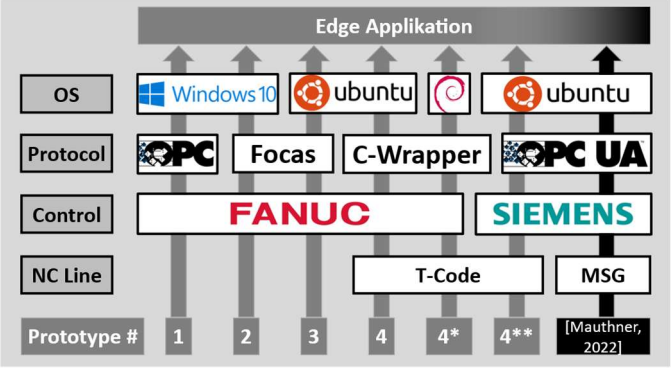

Fig. 2: Overview of developed test prototypes.

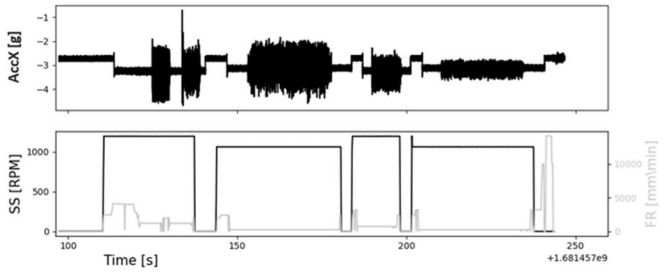

Fig. 3: Acceleration data (AccX), spindle speed (SS) and feed rate (FR) from different data sources using a common timestamp.

- an industrial edge device that runs on Debian (Linux-based OS) and comes with a preinstalled Docker environment.

In the second variant (4**), a machine tool from a different vendor and its present edge device were retrofitted with STH through the proposed edge application. To do so, the CC was switched to an application that reads the data points of the CNC machine using OPC UA and distributes them using MQTT.

The functionality of the sensor activation through the current tool was verified on a different machine and different edge device. While the data processing container of the earlier variant saves the data locally, the other calculates an indicator for instability, calculated from the acceleration data, and passes the information on to an MQTT broker (*Fig. 4*).

Table 2 summarizes the main evaluation for the described prototypes. The evaluation questions about the effectiveness of the edge application, its implementation and maintenance efforts (time and cost wise), its scalability and likeliness to be supported by stakeholders, are answered positively with the 4th prototype.

In the order of the prototype evolution, the responses of the evaluation questions are justified in *Table 3*.

Table 2. Prototype Evaluation in comparison.

Evaluation question \ Prototype #	1	2	3	4
Effectiveness	~	~	~	~ +
Feasibility (time)	~	~	~	+
Viability (cost)	-	~	+	+
Scalability	-	+	+	+
Stakeholder support	~	-	~	+

Table 3. Answers of evaluation questions per prototype evolution.

Question	Rate	Justification
1st prototype (DMU + OPC UA + Windows)		
Effectiveness	~	(+) Connectivity between CNC and STH tested (-) NC data rate: 1,5 kbps (20 Byte à 9Hz)
Feasibility (time)	~	(-) extra OPC UA server installation needed (+) fast OPC UA server configuration
Viability (cost)	-	(-) extra Focas OPC UA license necessary (-) extra Windows Edge needed, if not present
Scalability	-	(-) no containerization due to Windows dependency
Stakeholder support	~	(+) OPC UA popularity (-) Windows supporting hardware required
2nd prototype (DMU + C library + Windows)		
Effectiveness	~	(~) only CNC connection tested (+) NC data rate: 20kbps (140 Byte à 19Hz)
Feasibility (time)	~	(+) C-library installation by script possible (-) configuration requires C-Know how
Viability (cost)	~	(+) no extra Focas OPC UA license necessary (-) extra Windows Edge needed, if not present
Scalability	+	(+) containerization of C-Library possible
Stakeholder support	-	(-) C is not popular amongst engineers (-) Windows supporting hardware required
3rd prototype (DMU + C library + Ubuntu)		
Effectiveness	~	(~) only CNC connection tested (+) NC data rate: 9 kbps (80Byte à 14Hz)
Feasibility (time)	~	(+) C-library installation by script possible (-) configuration requires C-Know how
Viability (cost)	+	(+) no extra Focas OPC UA license necessary (+) no extra Windows Edge needed
Scalability	+	(+) containerization of C-Library possible
Stakeholder support	~	(-) C is not popular amongst engineers (+) Linux hardware is popular in IoT
4th prototype (DMU + C-Wrapper + Ubuntu)		
Effectiveness	~ +	(+) Connectivity between CNC and STH tested (~) NC data rate: 5 kbps (80Byte à 6Hz)
Feasibility (time)	+	(+) C-library installation by script possible (+) easy configuration with Python
Viability (cost)	+	(+) no extra Focas OPC UA license necessary (+) no extra Windows Edge needed
Scalability	+	(+) containerization of C-Library possible
Stakeholder support	+	(+) Python is popular amongst engineers (+) Linux hardware is popular in IoT

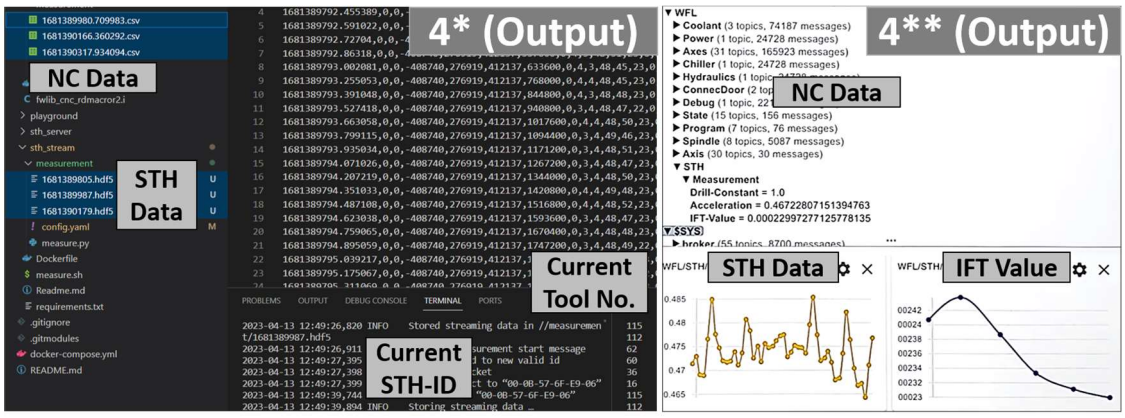

Fig. 4: Prototype 4* saves the files locally (left), while 4** distributes them over an MQTT broker (right).

5. Discussion

The need for adding external sensors or actuators to existing setups is met by the use of standard edge hardware, standard industrial protocols, and formats, using a popular high-level programming language with open-source libraries. Additionally, three ways of informing retrofitted devices about the current state of the machining process through the edge application. We demonstrated the flexibility of the software modules by switching the NC for the subscription of MQTT topics. Additionally, we switched the data processing task from one that stores the data locally to an external visualization service as shown in *Fig. 4*. Basic Python and Docker skills are sufficient for the modification. We studied the usability of the application on different hardware. The switching between two Linux-based OS is straightforward. A transition to a Windows-based system is possible but requires extra attention in terms of interfaces. Different communication protocols, MQTT, HTTP, and UDS, were used to exchange information with the SC.

Our prototype validations confirm the previous results of Garcia et al. [4]. They implemented a modular middle-ware consisting of asynchronous jobs to achieve a reduced complexity of the software and with it, higher feasibility for SMEs. In contrast to their model, we extended the border of the edge application to the machine tool itself. That way, only one application must be maintained saving time and necessary human resources. The work of Mauthner et al. [13], who validated the approach of passing on the moment of a new CAM operation to the edge application through NC-Code messages, presents a valuable alternative for a more specific sensor triggering to our approach of writing operation numbers to macro variables. As the concept aims to broaden the applicable setups, it would add a possible solution for Sinumerik controls.

6. Conclusion and Future Work

Four different prototypes of an edge application for retrofitting CNC tools with external hardware have been implemented and tested - not only concerning the internal architecture but also communicating the current line of a running NC-Code to the edge application. As a result, the architecture of an edge application is proposed that allows general usability by making assumptions about standard hardware and software conditions of a manufacturing system and by testing the result on different machine controls.

The final edge application is deployed on a modern edge device and consists of containerized application modules written in Python using standard protocols and interfaces for inside and outside communication. To verify its flexibility in the system environment, we tested it in a different setup and exchanged the modules with other applications.

Future research should study the integration of different external devices and additional modules, be it in terms of varying functionality, hardware setups, or interfaces.

References

[1] J. Lee, J. Singh, M. Azamfar, and V. Pandhare, “Industrial AI and predictive analytics for smart manufacturing systems,” *Smart Manufacturing: Concepts and Methods*, pp. 213–244, Jan. 2020, doi: 10.1016/B978-0-12-820027-8.00008-3.

[2] A. Garcia, J. Franco, F. Saez, J. R. Sanchez, and J. L. Bruse, “Containerized edge architecture for manufacturing data analysis in Cyber-Physical Production Systems,” *Procedia Comput Sci*, vol. 204, pp. 378–384, Jan. 2022, doi: 10.1016/J.PROCS.2022.08.046.

[3] GeeksforGeeks, “Layers of OSI Model.” https://www.geeksforgeeks.org/layers-of-osi-model/ (accessed May 18, 2023).

[4] L. Kong, X. Peng, Y. Chen, P. Wang, and M. Xu, “Multi-sensor measurement and data fusion technology for manufacturing process monitoring: a literature review,” *International Journal of Extreme Manufacturing*, vol. 2, no. 2, p. 022001, Mar. 2020, doi: 10.1088/2631-7990/AB7AE6.

[5] C. H. Lee, Z. L. Wu, Y. T. Chiu, and V. S. Chen, “Heterogeneous Industrial IoT Integration for Manufacturing Production,” *Proceedings - 2019 International Symposium on Intelligent Signal Processing and Communication Systems, ISPACS 2019*, Dec. 2019, doi: 10.1109/ISPACS48206.2019.8986308.

[6] M. H. ur Rehman, I. Yaqoob, K. Salah, M. Imran, P. P. Jayaraman, and C. Perera, “The role of big data analytics in industrial Internet of Things,” *Future Generation Computer Systems*, vol. 99, pp. 247–259, Oct. 2019, doi: 10.1016/J.FUTURE.2019.04.020.

[7] I. Ungurean and N. C. Gaitan, “A software architecture for the industrial internet of things—a conceptual model,” *Sensors (Switzerland)*, vol. 20, no. 19, pp. 1–19, Oct. 2020, doi: 10.3390/s20195603.

[8] G. Aceto, V. Persico, and A. Pescapé, “A Survey on Information and Communication Technologies for Industry 4.0: State-of-the-Art, Taxonomies, Perspectives, and Challenges,” *IEEE Communications Surveys and Tutorials*, vol. 21, no. 4, pp. 3467–3501, Oct. 2019, doi: 10.1109/COMST.2019.2938259.

[9] W. Z. Khan, M. H. Rehman, H. M. Zangoti, M. K. Afzal, N. Armi, and K. Salah, “Industrial internet of things: Recent advances, enabling technologies and open challenges,” *Computers & Electrical Engineering*, vol. 81, p. 106522, Jan. 2020, doi: 10.1016/J.COMPELECENG.2019.106522.

[10] B. I. Ismail *et al.*, “Evaluation of Docker as Edge computing platform,” in *ICOS 2015 - 2015 IEEE Conference on Open Systems*, Institute of Electrical and Electronics Engineers Inc., Jan. 2016, pp. 130–135. doi: 10.1109/ICOS.2015.7377291.

[11] “Edge for IT specialists - Industrial Edge - Global.” https://www.siemens.com/global/en/products/automation/topic-areas/industrial-edge/it-specialists.html#Edgedevices (accessed May 08, 2023).

[12] “MFN 100 – Nerve 2.6.1 Documentation.” https://docs.nerve.cloud/device_guide/mfn100/#additional-device-specific-information (accessed May 08, 2023).

[13] G. Mauthner, W. Votruba, C. Ramsauer, L. Plessing, T. Trautner, and F. Bleicher, “Development of a CAM-in-the-Loop System for Cutting Parameter Optimization using an Instrumented Tool Holder,” *Procedia CIRP*, vol. 107, pp. 326–331, 2022, doi: 10.1016/J.PROCIR.2022.04.053.

[14] A. Dwivedi and A. Jaiswal, “Python: The Versatile Language-Recent Trends in Programming Languages,” vol. 8, no. 1, p. 2021, 2021, doi: 10.37591/RTPL.

[15] A. Eftimie and E. Borcoci, *Containerization Using Docker Technology*.

[16] fledge-iot, “Fledge Architecture.” https://fledge-iot.readthedocs.io/en/v1.9.2/fledge_architecture.html (accessed May 19, 2023).

[17] P. Schörghofer, F. Pauker, N. Leder, J. Mangler, C. Ramsauer, and F. Bleicher, “Using sensory tool holder data for optimizing production processes,” *Journal of Machine Engineering*, vol. 19, no. Vol. 19, No. 3, pp. 43--55, 2019, doi: 10.5604/01.3001.0013.4079.

[18] M. Cabaj, “Evaluating Prototypes,” 2016. [Online]. Available: www.betterblock.org