Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2019) 000–000

www.elsevier.com/locate/procedia

CIRP Manufacturing Systems Conference 2019

Modular smart controller for Industry 4.0 functions in machine tools

David Barton[a]*, Philipp Gönnheimer[a], Florian Schade[b], Christopher Ehrmann[a,c], Jürgen Becker[b], Jürgen Fleischer[a]

[a] *wbk Institute of Production Science, Karlsruhe Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe, Germany*
[b] *ITIV Institute for Information Processing Technologies, Karlsruhe Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe, Germany*
[c] *Advanced Manufacturing Technology Center (AMTC),Tongji University, Shanghai, 201804, China*

* Corresponding author. Tel.: +49-1523-9502565. *E-mail address:* david.barton@kit.edu

Abstract

In machine tools, Industry 4.0 functions can increase availability through predictive maintenance, while other functions improve productivity and workpiece quality through process supervision and optimisation. Many of these functions rely on data communication between systems from different suppliers. Requirements regarding latency and computing vary widely depending on the application. Based on an analysis of these requirements, a smart controller for the implementation of Industry 4.0 is designed, using a hypervisor to allow for the integration of soft real-time and best-effort applications.

Keywords: Monitoring; Machine tool; Distributed Control

1. Introduction

Due to increasing demands for productivity and workpiece quality, the importance of process monitoring and predictive maintenance is increasing in production, also promising economic benefits [1]. However, a diverse hardware and software landscape hampers implementation of these Industry 4.0 functions in new (greenfield) and existing (brownfield) production systems [2]. This contribution offers a concept for a modular *smart controller* to enable flexible and manufacturer-independent solutions for process monitoring and predictive maintenance functions, focussing on machine tools. Relevant work on Industry 4.0 functions is reviewed in order to derive requirements. An architecture for data communication and computing is then proposed and implemented within an exemplary machine.

1.1. Connected smart components in machine tools

A widespread key performance indicator (KPI) in the operation of machine tools is the overall equipment effectiveness (OEE), which depends on the availability and productivity of the machine as well as the quality of the produced workpieces. High availability can be achieved by planning repairs based on condition monitoring. Productivity and workpiece quality can be improved through process supervision and optimisation, especially in the production of small batches. [2]

In the following, metal cutting machine tools such as milling machines will be used as an example in order to discuss the relevant requirements. The main spindle and the feed axes are especially relevant subsystems with regard to workpiece quality and system availability. In the main spindle, bearings in particular are subject to wear and influence the quality of the

produced workpiece. In the feed axis, mechanical transmissions, such as ball screws or rack-and-pinion drives, and guiding systems (e.g. roll guides) have a decisive influence.

These components are responsible for transmitting the process force, which means they constitute potential locations for measuring quantities required for monitoring the machining process. They are also susceptible to be replaced several times during the lifetime of a machine tool due to wear. It is therefore of interest to monitor the wear of these components and plan maintenance measures accordingly. In order to do that, intelligence in the form of additional sensors (typically strain gauges, structure-borne sound and temperature) and local component-specific preprocessing is required. Thus the mechanical component becomes a cyber-physical *smart component*. This smart component is a mechatronic system that is capable of recording data from the environment via sensors, processing the data to derive information on the status of the component, and passing this on to other systems. [3, 4]

Existing solutions based on smart components rely on proprietary architectures within isolated applications. Another suitable and more flexible solution would be a modular controller, connecting to one or more such smart components within the machine. This *smart controller* would receive sensor data and other information from the components as well as additional data from the machine control unit. The controller would evaluate the data to determine if one of the following interventions is necessary:

- Emergency stop
- Adjustment of process settings
- Planned or immediate maintenance measures

If data from several machines is to be aggregated, or the data processing is too computationally expensive to be performed locally, it is expedient to exchange data with a server outside the machine. This may for example be a cloud-based prognosis service [5] or a manufacturing execution system (MES). Depending on the required computing power and reaction time, the data should be evaluated by the smart controller and/or transmitted to other computers. The resulting communication architecture is shown in Fig. 1. The smart controller should be implemented as an additional system parallel to the machine control system, thus reducing the need to adapt the machine control and limiting adverse effects of the additional functions on the basic functionality of the machine.

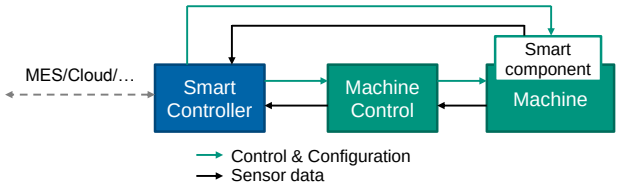

Fig. 1. Interaction in between smart controller and machine

1.2. Approaches for predictive maintenance

Predictive maintenance aims to maximise the useful lifetime of machine components while avoiding unplanned production downtime. Many approaches based on the acquisition of data within such components can be found in existing publications.

The lifetime of ball screws can be extended by adaptive lubrication based on data from strain gauges [6] and the current wear status can be monitored based on measured vibration data [7,8] or strain gauges [9, 10]. In ball bearings, vibration and acoustic emission data can be used for fault diagnosis and classification [11, 12] as well as remaining useful lifetime prognosis [13, 14, 15]. Prognostics and health monitoring can be enhanced by combining measured data with a digital twin of the monitored equipment [16].

Earlier approaches focus on time-domain, frequency-domain, and time-frequency analysis of vibration data [17]. More recently, support vector machines [11] and deep learning approaches (e.g. Auto-encoders, Restricted Boltzmann Machines, Convolutional Neural Networks, Recurrent Neural Networks) have been shown to be suitable for machine health monitoring problems [18].

1.3. Approaches for monitoring machining processes

An overview of possible scopes for monitoring machining operations is given by Teti et al. [19]. The wear of cutting tools can be monitored based on the cutting force [20, 21], motor current, acceleration or a combination of these data sources [22]. Another type of process monitoring consists in avoiding or detecting collisions and overload. This is achieved by adding sensors to measure process forces [23], evaluating existing data such as motor currents [24] or predicting collisions using geometrical models [25]. In all three cases, a short reaction time is necessary in order to avoid or reduce damage to tool, workpiece, and machine. In metal cutting operations, vibration analysis can be used to detect undesirable cutting conditions such as chatter [26, 27] and determine process boundaries represented by stability lobe diagrams [28].

Data processing methods for monitoring the machining process include simple thresholds, conventional time-frequency analysis [19] and neural networks [29]. Integrating on-line monitoring with virtual machining simulations can enable more robust performance in detection tasks [30]. In some machining applications measured process data has been used to introduce an additional dynamic control loop for the machining process. In milling for example this may consist in a feed rate control based on the cutting force [31].

2. Requirements analysis

The approaches shown above differ greatly with regard to the permissible response time. In the case of predictive maintenance approaches, the state of the monitored machine elements changes slowly over a period of several months or years. The reaction to the acquired data consists in scheduling maintenance measures. If predictive maintenance has been implemented effectively, failures due to wear are detected well in advance and no real time reaction is necessary.

On the other hand, approaches for monitoring machining processes may require a fast reaction, requiring a soft real-time system. As described above, an important scope for process monitoring is detecting collisions. In this case, damage occurs

when components are subject to plastic deformation. Therefore, damage can only be avoided if the overload is detected and the machine is stopped within the initial period of elastic deformation. In typical machining applications this requires an emergency stop to be initiated within 5 ms [32].

The examples described above also vary widely in terms of the complexity of the required computing, ranging from simple thresholds to complex machine learning architectures. The latter require an environment offering high flexibility with regard to the implementation of data processing algorithms. In some cases, the implementation of a model or digital twin is also required. In approaches aiming to learn from data from several machines or requiring large computational resources, it is necessary to exchange data with other systems located outside of the machine.

Table 1. Requirements for data collection and processing.

Function	Acceptable latency (order of magnitude)	Sensor data sampling rate (order of magnitude)	Data processing complexity
Collision detection	5 ms	1 kHz	Low
Remaining lifetime prognosis	4 hours	50 kHz	High

For a prototypical implementation and validation, this contribution focuses on two applications representing different segments of the requirements spectrum: a remaining useful lifetime prognosis for bearings and a function for collision detection. The monitored component is the main spindle of a machining centre, equipped with sensors measuring vibrations and displacement. These sensors provide integrated data processing and use a CAN communication interface for data transmission. An industry 4.0 testbed at KIT consisting of a 4-axis machining centre is used for a practical validation (Fig. 2). The relevant requirements for the data processing unit in these applications are summarised in Table 1. Additionally, an ideal system should enable widespread use by not depending on any particular control system vendor.

Fig. 2. Industry 4.0 machine tool test bed

3. Existing solutions for data collection and processing

3.1. Existing devices for collection and processing of industrial data

Conventional qualified industrial solutions for real-time data processing are PLCs (programmable logic controllers). Especially PC-based controllers are suitable for implementing analytic applications locally in the machine and interfacing with cloud computing services. Restrictions here are the limited system independence and the need to implement the application within a complex control system. Given that our approach is focused on providing a modular, manufacturer-independent solution, it is advantageous to perform data analysis on a separate system. [33]

Connecting the physical process to a higher-level architecture for analysis is one of the central topics of Industry 4.0. Connectivity systems for Industry 4.0 or IIoT (Industrial Internet of Things) are available from many different suppliers. However, none of the existing devices have achieved widespread industrial use for monitoring functions in machine tools yet. A comprehensive system independent of the machine function, combining real time capability and flexible programming of additional functions, has yet to be established. [34]

3.2. Hypervisor-based software architectures for machine control and data processing

Using hypervisor-based architectures, partitions can be isolated to fulfil differentiated requirements with regard to latency and flexibility.

In the industrial field, the pICASSO project has investigated the realisation of a platform for virtualised real-time operating systems with deterministic timing behaviour. Its focus is on cloud-based machine control. The combination of KVM hypervisor and QEMU emulator was identified as not suitable for hosting real-time applications within virtual machines. Instead, the statically-partitioning hypervisor Jailhouse was enhanced by adding support for ARM processors. Its small code base and time-deterministic behaviour seems more promising for hosting real-time applications. [35]

Biondi et al. discuss the use of virtualisation in facing the complexity of heterogeneous embedded computing platforms. It names exemplary interference channels on custom-off-the-shelf (COTS) multi-core processors and thereby identifies limitations in hypervisors when it comes to hosting time critical applications. An automotive use case is presented, where the Jailhouse hypervisor is used to isolate the automotive-oriented ErikaOS real-time operating system (RTOS) for timing-critical tasks and a common Linux OS for less critical tasks. This use case is investigated as part of the RETINA project. [36]

Also focusing the automotive domain, the HERCULES project aims at providing design methodologies as well as a software stack for the development of safety-critical applications on high-performance COTS computation platforms for next-generation automotive control units. For highly safety-critical tasks, a subsystem comprising an automotive-grade microcontroller is envisioned, while for

lower levels of safety, ARM cores are planned. Within the less critical part, the parallel execution of real-time and best-effort applications shall be realised using the Jailhouse hypervisor, separating a real-time ErikaOS RTOS partition from a Linux OS partition. To reduce interference by parallel accesses to the main memory, the use of a "Predictable Execution Model" is proposed. [37, 38]

Considering hypervisor performance evaluations, Toumassian et al. compare the performance impact of Xen and Jailhouse hypervisors on the application execution time. Measurements were done on ARM processors, specifically the Banana Pi single-board computer, using the "cpuburn-a8" program, indicating that Jailhouse hypervisor leads to a relative overhead of 0.04% in execution time, compared to 21.6% and 7.4% for Xen, depending on the scheduler. [39]

Thus Jailhouse is a promising solution for realising hypervisor-based architectures in data processing applications. It remains to be shown how such an architecture can be used in the context of Industry 4.0 applications in manufacturing.

4. Concept for a smart controller

Existing Industry 4.0 approaches promise to deliver improvements in the OEE of machine tools, especially through the above mentioned application for predictive maintenance and process monitoring. However, their widespread implementation would be facilitated by adding a modular device to the machine, capable of running both soft real-time data processing and flexibly programmable best-effort applications. The proposed *smart controller* aims to fulfil this role. It interfaces one more *smart components* and other intelligent sensors as well as the machine control unit and an MES or cloud. The smart controller processes data received from these data sources to improve the machine availability or the production process. The concept is developed based on two representative applications, as described above: collision detection and remaining useful lifetime prognosis.

To deliver the first application, the smart controller needs to be real-time capable. Soft real-time capability is considered sufficient since all safety-critical features can be assumed to be implemented in the machine control unit itself. Concerning the second application high flexibility and simplified software development is desired, as provided by a general-purpose operating system (GPOS). To combine both real-time capabilities and the flexibility and ease of development and use of a GPOS, it is proposed to use a hypervisor to run a real-time partition and a best-effort partition in parallel. The resulting software structure is shown in Fig. 3.

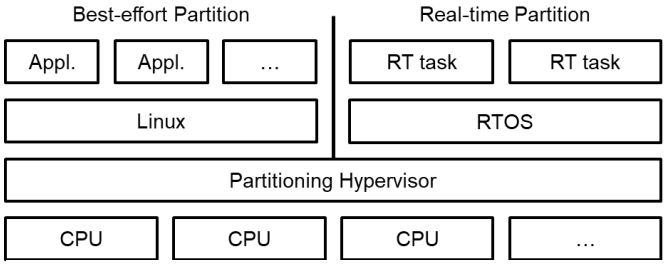

Fig. 3. Smart Controller software architecture

In the real-time partition, a real-time operating system can be used to run a fixed set of real-time tasks. In the best-effort partition, Linux can be run, allowing for the dynamic starting and stopping of multiple applications, and providing various drivers, easing development. The hypervisor isolates the partitions and defines partition resource access. Concerning isolation, resources, such as memory and peripherals, are statically assigned to partitions. A communication channel between the partitions is established to transfer sensor data and status information from the real-time partition to the best-effort partition. The smart controller's external communication interfaces as well as their assignment to its partitions are shown in Fig. 4. The real-time partition controls the interfaces to the smart components and the machine control unit while the best-effort partition controls the interface towards management units, such as cluster controllers, the MES or the cloud.

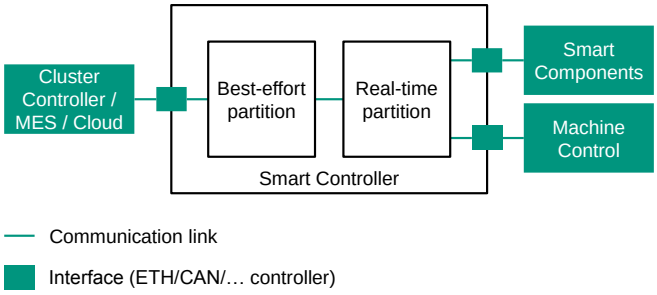

Fig. 4. Communication interfaces of the Smart Controller

Using a static partitioning approach as proposed in this publication allows for the parallel use of a real-time OS and a general-purpose OS such as Linux. It effectively isolates the two OS concerning access to specific memory regions (and thereby memory-mapped peripherals). However, accesses to shared resources such as caches, memory busses, and memory banks can still cause interference between the partitions. This may lead to unexpected delays in the execution of software in one partition, based on the other partition's behaviour. The use of this platform is therefore envisioned for soft real-time applications only, where single deadline misses are acceptable.

5. Implementation and preliminary assessment

A prototype for the smart controller concept is realised on the Lemaker BananaPi M1 single-board computer. It is based on an Allwinner A20 System-on-Chip (SoC) featuring an ARM Cortex-A7 dual-core CPU and 1 GB DDR3 SDRAM. For connectivity, the SoC supports Gigabit USB, Ethernet and an on-chip CAN controller, among others.

For partitioning, Jailhouse was chosen. The hypervisor implements static partitioning of a system, i.e. isolates bare-metal applications or operating systems from each other. It does not implement VM scheduling and does not allow for overcommitting of resources. CPU cores are statically mapped to one partition ("cell") each. Due to its minimal design it avoids VM exits and leads to low overhead in comparison to a non-virtualized system, while efficiently isolating the partitions concerning accessible memory and interrupts [39]. Its small

codebase simplifies attempts for certification for the use in safety-critical applications.

On the A20 SoC, the hypervisor Jailhouse was set up to realise two partitions: the best-effort partition hosts a Bananian Linux, a Debian installation optimized for the use on the Banana Pi [40]. FreeRTOS, a small-footprint open-source real-time operating system, is used in the real-time partition. It provides basic kernel services, such as scheduling, inter-task communication, and synchronization [41]. Since CAN is envisioned as an interface for connecting smart components, a first proof-of-concept was set up, where CAN is used to control a SCHUNK LWA 4D lightweight robotic arm. In this setup, the control application was running on Linux, which was accessing the CAN controller. Current work focuses on the realisation of CAN control from the FreeRTOS partition.

6. Summary and outlook

Based on a review of approaches for predictive maintenance and the monitoring of machining processes, requirements regarding computing and communication were derived for the implementation of these Industry 4.0 functions in machine tools. Within the wide range of requirements, two representative examples were chosen, both related to the same subsystem within the machine (main spindle). The first task consists in triggering a machine stop in the event of a collision, requiring a simple threshold-based data processing with a low latency. The second task is to monitor the condition of bearings in the spindle and predict their remaining lifetime. For this task latency is not a critical issue, the focus is on flexible implementation of complex models.

The proposed *smart controller* concept represents a comprehensive system combining soft real-time capability in one partition and flexible programming in a general-purpose operating system in the best-effort partition, thus enabling the fulfilment of the requirements mentioned above. Both partitions can communicate with embedded sensors and smart components, using CAN in the real-time partition and Ethernet in the best-effort partition. Additionally, the best-effort partition can communicate with further smart controllers or with a cloud service via OPC UA.

An important strength of this concept is its flexibility: the architecture is independent of any particular supplier, does not rely on a specific chip, and enables the implementation of state-of-the-art Industry 4.0 functions with few restrictions. In cases where it is preferable to process data on a central server or a cloud service, the smart controller enables data to be aggregated locally before being sent to other devices.

Building on this, the next step will be to carry out tests in a real production environment on the Industry 4.0 test bed shown. In this production environment, further intelligent components are to be integrated and tested in order to assess the functionality of the concept. Further research will also be carried out in the area of connecting and embedding the Smart Controller in higher-level architectures.

Acknowledgements

This research and development project is funded by the German Federal Ministry of Education and Research (BMBF) within the Program “Innovations for Tomorrow’s Production, Services, and Work” (02P17X000 ff.) and managed by the Project Management Agency Karlsruhe (PTKA). The author is responsible for the contents of this publication.

References

[1] S. Heng. Industry 4.0 – Upgrading of Germany‘s industrial capabilities on the horizon, Deutsche Bank AG (2014)

[2] G. Schuh, R. Anderl, J. Gausemeier, M. ten Hompel, W. Wahlster. Industrie 4.0 Maturity Index – Managing the Digital Transformation of Companies, acatech STUDY, Munich (2017)

[3] T. Bauernhansl, J. Krüger, G. Reinhart, G. Schuh. WGP-Standpunkt Industrie 4.0, eds. Wissenschaftliche Gesellschaft für Produktionstechnik WGP e. V.

[4] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, K. Ueda. Cyber-physical systems in manufacturing, CIRP Annals. 65 (2016) 621-41.

[5] R. Gao, L. Wang, R. Teti, D. Dornfeld, S. Kumara, M. Mori, M. Helu. Cloud-enabled prognosis for manufacturing. CIRP Annals 64 (2015) 749–772.

[6] A. Spohrer, L. Leitold, F. Straub, J. Hillenbrand, J. Fleischer. Ressourceneffizienter Kugelgewindetrieb durch adaptive Schmierung, Reibung, Schmierung und Verschleiß - Forschung und praktische Anwendungen, eds. Gesellschaft für Tribologie e.V., GfT, Aachen, (2017) 39- 49.

[7] M. Schopp, Sensorbasierte Zustandsdiagnose und -prognose von Kugelgewindetrieben. Dissertation, University of Karlsruhe; 2009.

[8] L. Zhang, H. Gao, J. Wen, S. Li, Q. Liu. A deep learning-based recognition method for degradation monitoring of ball screw with multi-sensor data fusion. Microelectronics Reliability 75 (2017) 215–222.

[9] H.-C. Möhring, O. Bertram. Integrated autonomous monitoring of ball screw drives. CIRP Annals 61 (2012) 355-358.

[10] J. Hillenbrand, A. Spohrer, J. & Fleischer, Zustandsüberwachung bei Kugelgewindetrieben. wt Werkstatttechnik online, no. 8 (2018) 493-498

[11] R. Ziani, A. Felkaoui, R. Zegadi, Bearing fault diagnosis using multiclass support vector machines with binary particle swarm optimization and regularized Fisher’s criterion. Journal of Intelligent Manufacturing 28 (2017) 405-417

[12] H.O.A. Ahmed, M.L.D. Wong, A.K. Nandi. Intelligent condition monitoring method for bearing faults from highly compressed measurements using sparse over-complete features. Mechanical Systems and Signal Processing 99 (2018) 459-477.

[13] Y. Qian, R. Yan. Remaining Useful Life Prediction of RollingBearings Using an Enhanced Particle Filter. IEEE Transactions on Instrumentation and measurement 64 (2015) 2696-2707.

[14] L. Guo, N. Li, F. Jia, Y. Lei, J. Lin. A recurrent neural network based health indicator for remaining useful life prediction of bearings. Neurocomputing 240 (2017) 98-109.

[15] M. Elforjani, S. Shanbr. Prognosis of Bearing Acoustic Emission Signals Using Machine Learning. IEEE Transactions on industrial electronics 65 (2018) 5864-5871.

[16] F. Tao, M. Zhang, Y. Liu,A.Y.C. Nee. Digital twin driven prognostics and health management for complex equipment. CIRP Annals 67 (2018) 169-172.

[17] A.K.S. Jardine, D. Lin, D. Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. Mechanical Systems and Signal Processing 20 (2006) 1483-1510.

[18] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, Robert X. Gao. Deep learning and its applications to machine health monitoring. In: Mechanical Systems and Signal Processing, Volume 115, 2019, Pages 213-237, ISSN 0888-3270, https://doi.org/10.1016/j.ymssp.2018.05.050.

[19] R.Teti, K. Jemielniak, G. O’Donnell, D. Dornfeld. Advanced monitoring of machining operations. CIRP Annals 59 (2010) 717-739.

[20] M. Nouri, B.K. Fussell, B. L. Ziniti, E. Linder. Real-time tool wear monitoring in milling using a cutting condition independent method. International Journal of Machine Tools and Manufacture 89 (2015) 1-13.
[21] K. Zhu, Y. Zhang. A generic tool wear model and its application to force modeling and wear monitoring in high speed milling. Mechanical Systems and Signal Processing 115 (2019) 147-161.
[22] X.Y. Zhang, X. Lu, S. Wang, W. Wang, W.D. Li. A multi-sensor based online tool condition monitoring system for milling process. Procedia CIRP 72 (2018) 1136–1141.
[23] G. Byrne, G.E. O'Donnell. An Integrated Force Sensor for Process Monitoring of Drilling Operations. CIRP Annals 56 (2007) 89-92.
[24] T. Shigematsu, R. Koike, Y. Kakinuma, T. Aoyama, K. Ohnishi. Sensorless tool collision detection for multi-axis machine tools by integration of disturbance information. Procedia CIRP 57 (2016) 658 – 663.
[25] M. Schumann, M. Witt, P. Klimant. A Real-Time Collision Prevention System for Machine Tools. Procedia CIRP 7 (2013) 329 – 334.
[26] E. Kuljanic, M. Sortino, G. Totis. Multisensor approaches for chatter detection in milling. Journal of Sound and Vibration Volume 312 (2008) 672–693.
[27] C. Liu, L. Zhu, C. Ni. Chatter detection in milling process based on VMD and energy entropy. Mechanical Systems and Signal Processing 105 (2018) 169-182.
[28] J. Friedrich, J. Torzewski, A. Verl. Online learning of stability lobe diagrams in milling. Procedia CIRP 67 (2018) 278-283
[29] U. Zuperl, F. Cus, M. Reibenschuh. Neural control strategy of constant cutting force system in end milling. Robotics and Computer-Integrated Manufacturing 27 (2011) 485–493.
[30] Y. Altintas, D. Aslan. Integration of virtual and on-line machining process control and monitoring. CIRP Annals 66 (2017) 349–352.
[31] S. Stemmler, D.Abel, O.Adams, F. Klocke. Model Predictive Feed Rate Control for a Milling Machine. IFAC-PapersOnLine 49 (2016) 11-16.
[32] T. Rudolf, C. Brecher, F. Possel-Dölken, Contact-based collision detection—a new approach to avoid hard collisions in machine tools. InInternational Conference on Smart Machining Systems 2007 Mar 13 (pp. 1-4).
[33] Beckhoff. TwinCAT Analytics https://download.beckhoff.com/download/press/2016/english/pcc_0116_twincat_analytics_e.pdf
[34] M. Schleipen, R. Henßen, T. Bischoff, J. Pfrommer, O. Sauer, D. Schneider, F. Jungbluth, H. Flatt, D. Barton, J. Fleischer, E. Bollhöfer, C. Moll, J. Lindauer, R. Davis, H. Baron, T. Danner, T. Hillerich, C. Schmuck, M. Blume, S. Finster, A. Fechner, R. Tschepat, D. Kazakov, R. Kühbauch, W. Klöblen, D. Sproll, B. Fellhauer, D. Osswald, SecurePLUGandWORK – Abschlussbericht, Karlsruhe, Fraunhofer IOSB; (2017), http://publica.fraunhofer.de/dokumente/N-439001.html.
[35] F. Kretschmer, J. Altenberg, Steuerungstechnik aus der Cloud. Echtzeit virtualisiert (2016) 35-37
[36] A. Biondi, M. Marinoni, G. Buttazzo, C. Scordino, P. Gai, Challenges in virtualizing safety-critical cyberphysical systems. Proceedings of Embedded World (2018)
[37] P. Burgio, M. Bertogna, N. Capodieci, R. Cavicchioli, M. Sojka, P. Houdek, B. Morelli, A software stack for next-generation automotive systems on many-core heterogeneous platforms. Microprocessors and Microsystems 52 (2017) 299-311
[38] P. Burgio, M. Bertogna, I.S. Olmedo, P. Gai, A. Marongiu, M. Sojka, A Software Stack for Next-Generation Automotive Systems on Many-Core Heterogeneous Platforms. Euromicro Conference on Digital System Design (DSD), Limassol, Cyprus (2016)
[39] S. Toumassian, R. Werner, A. Sikora, Performance measurements for hypervisors on embedded ARM precessors. International Conference on Advances in Computing, Communications and Informatics (ICACCI) (2018)
[40] N. Isenbeck, Bananian Linux https://www.bananian.org/
[41] FreeRTOS https://www.freertos.org/

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2019) 000–000

www.elsevier.com/locate/procedia

Peer-review under responsibility of the scientific committee of the 52nd CIRP Conference on Manufacturing Systems.