PeerJ Computer Science

Towards a general application programming interface (API) for injection molding machines

Olga Ogorodnyk[1], Mats Larsen[2], Ole Vidar Lyngstad[2] and Kristian Martinsen[1]

[1] Department of Manufacturing and Civil Engineering, Norwegian University of Science and Technology (NTNU), Gjøvik, Norway
[2] SINTEF Manufacturing, Raufoss, Norway

ABSTRACT

Injection molding is a complicated process, and the final part quality depends on many machine and process parameters settings. To increase controllability of the injection molding process, acquisition of the process data is necessary. This paper describes the architecture and development of a prototype of an open application programming interface (API) for injection molding machines (IMMs), which has the potential to be used with different IMMs to log and set the necessary process parameter values. At the moment, the API includes an implementation of EMI data exchange protocol and can be used with ENGEL IMMs with CC300 and RC300 controllers. Data collection of up to 97 machine and process parameters (the number might vary depending on the type of machine at hand), obtained from sensors installed in the machine by the manufacturer is allowed. The API also includes a module for the acquisition of data from additional 3d party sensors. Industrial Raspberry Pi (RevPi) was used to perform analog-to-digital signal conversion and make sensors data accessible via the API prototype. The logging of parameters from the machine and from sensors is synchronized and the sampling frequency can be adjusted if necessary. The system can provide soft real-time communication.

Submitted 30 March 2020
Accepted 21 September 2020
Published 2 November 2020

Corresponding author
Olga Ogorodnyk,
olga.ogorodnyk@ntnu.no

Academic editor
James Procter

Additional Information and Declarations can be found on page 17

DOI **10.7717/peerj-cs.302**

Subjects Real-Time and Embedded Systems, Scientific Computing and Simulation
Keywords Application programming interface (API), Data acquisition system, Injection molding, Open source, Industry 4.0, Cyber-physical systems

INTRODUCTION

Manufacturing systems gradually become more sophisticated due to an increasing demand on quality of manufactured products, production system performance, as well as economic profitability (*Yin et al., 2014*). Therefore, data acquisition, process monitoring and control are becoming more and more important in the manufacturing industry (*Negri, Fumagalli & Macchi, 2017*). A significant number of studies stress the importance of collection and analysis of data from different manufacturing processes and injection molding is no exception (*Vrabič, Kozjek & Butala, 2017*). The concepts of Industry 4.0 and cyber-physical systems (CPS) are important drivers for this development.

It is becoming increasingly common to demand machine tools equipped with sensors from the machine tool manufacturers, and later to add sensors from the 3d party sensor suppliers. Appropriate sensors in machine tools, dies and molds can provide

useful information such as temperature and pressure, while communication protocols, e.g., MTConnect and OPC/UA allow the recording of the necessary controller signals (*Lee, Kao & Yang, 2014*). Using data acquired from these sensors can assist quality control routines and could establish models for suggestion of optimal process parameters (*Zhao et al., 2014*), decrease scrap rates and energy usage through the elimination of production cycles that produce defect parts and minimize unwanted "*manual adjustments and variations on product quality*" (*Saldivar et al., 2016*).

Before the start of the production of an injection molded part, parameter values such as holding pressure, backpressure, cooling time, injection speed, screw speed and others need to be set. These parameter settings influence the part quality, and erroneous parameter values might cause various defects (*Zhao et al., 2014*; *Ogorodnyk & Martinsen, 2018*). To aid the operation of the IMMs, many suppliers offer manufacturing execution systems (MES), which include data logging functions. These systems can be more or less complicated, although they will usually provide such functions as monitoring of machine status, remote access to machine set-up, data logging of machine and process parameters and displaying the data in a feasible way. Examples of such systems are TIG's authentig (*TIG. TIG authentig, 2020*), ARBURG's Host computer system (ALS) (*ARBURG. Host computer system , ALS*), KraussMafei's MaXecution (*KraussMaffei. MaXecution, 2020*), MES HYDRA (*MDPV. Manufacturing Execution System HYDRA, 2020*), SAP Manuacturing Execution (*SAP. SAP Manufacturing Execution, 2020*) and Polmak Plastik QMaster (*Polmak_Plastik. Polmak Plastik Products, 2020*).

There are, however, challenges for the usage of these systems for analysis and research purposes.

1. They are rather costly and often compatible only with newer generations of IMMs and IMM controllers.
2. It might be hard to synchronize data logging from machine built-in sensors with logging of data from 3d party sensors installed in the mold.
3. The MES systems often log data only once per production cycle and this might lead to missing information about the process dynamics and peak values of process parameters.
4. Commercially available MES software has a closed architecture that does not always allow to access all the machine and process data of interest, this might be an obstacle in terms of research and advanced process data analysis.

Therefore, in this article a question of *how to design a universal API to control and monitor injection molding for the next generation industrial systems* is discussed. A prototype solution of an open API that allows interaction with IMM's control unit is presented and made available through the following repository: https://github.com/SintefManufacturing/IMM_API. The prototype allows collection of production data through EMI data exchange protocol and can be built on to add support for other IMMs or new data collection functionality.

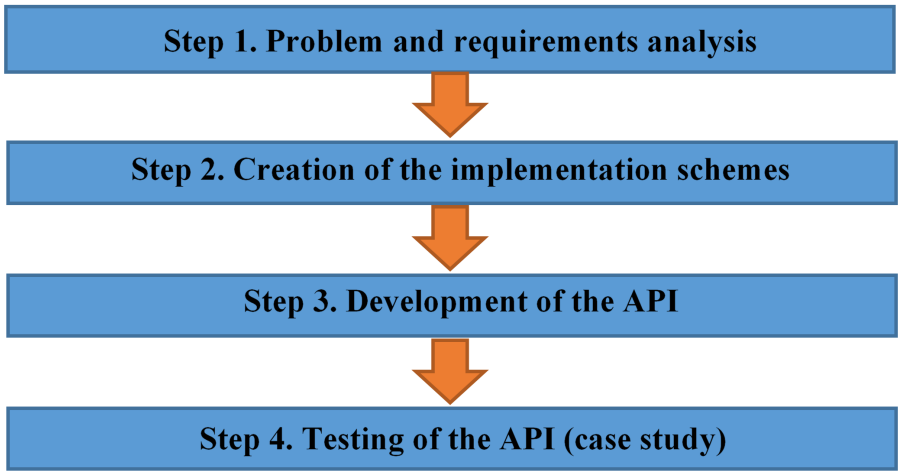


Figure 1 Research methodology steps.
Full-size DOI: 10.7717/peerjcs.302/fig-1

BACKGROUND AND REQUIREMENTS TO THE PROPOSED API

The described study is based on four main steps shown in Fig. 1. At first, a problem and requirements analysis was conducted. This included review of several available manufacturing execution systems, communication protocols used for connection of IMMs to other devices, as well as studies related to the development of similar systems by academia. In addition, the stakeholders' needs and requirements for an IMM API were identified. The primary stakeholders in this case are the companies and research institutions participating in the project, while potential stakeholders include the IMM manufacturers, other injection molding companies and researchers. The second step was dedicated to summarizing which models and frameworks are to be used to develop the proposed API prototype and creation of the implementation schemes of the system (UML diagrams). The third step included development of the prototype. In the final step, the system was tested with different sampling frequencies to assess its performance.

Injection molding process

Injection molding is one of the most widely used processes for production of polymeric products, as about 30% of them are manufactured using this process (*Osswald & Hernandez-Ortiz, 2006*). It is a cyclic process for production of identical parts through injection of molten material into a mold cavity of a desired shape (*Zheng, Tanner & Fan, 2011*). Its main advantage is an ability to fabricate high volumes of parts of various sizes and geometries, from the smallest components (micro-injection molding) to entire car body panels. The reciprocating screw IMMs are mostly used for production of the injection molded parts. Their main components are a hopper, a rotating screw, a heated barrel and a clamping unit consisting of a mold typically made of two halves.

An injection molding cycle usually includes the following phases: *plasticization*, *injection*, *packing*, *cooling* and *ejection* (*Zheng, Tanner & Fan, 2011*). During the *plasticization* stage

plastic pellets are fed into the heated barrel through the hopper. Here they are melted, mixed and homogenized with help of the reciprocating screw. When enough plastic melt is accumulated, the screw moves forward and forces the molten material into the mold cavity at high velocity and pressure (*injection* phase). When the mold cavity is 95–98% full, the process switches from the constant velocity to the constant pressure control to avoid pressure spikes. At the point, when the cavity is completely filled, the *packing* stage starts: the screw stays in the forward position or keeps moving with a small displacement to maintain the necessary holding pressure. The material cools down and shrinks allowing another small portion of the material to enter the cavity. This stage continues until the material in the cavity entrance solidifies. Next, the holding pressure is reduced to a value close to zero and the part continues to cool down and solidify (*cooling*) until the mold is opened and the part is ejected (*ejection*).

To achieve a repeatable process that allows production of high quality parts with specific material properties, a significant number of machine and process parameters need to be considered. Many of them, however, are dynamic, and adopt a characteristic curve over the production cycle. Examples of such parameters are screw speed, rotation speed, material cushion, mold temperature and pressure, etc. Most of these parameters can be acquired from the IMM sensors installed by its manufacturer, however, some require additional sensors in the molds, as for example, the mold temperature and pressure data. This data is suggested as one of the essential indicators of production of the high quality parts, as it reflects the evolution of the polymer conditions inside the mold, and is used and investigated by both industry and academia (*Zhao et al., 2014*; *Kurt et al., 2009*; *Kurt et al., 2010*; *Ageyeva, Horváth & Kovács, 2019*). There are a number of sensors available on the market for installation in molds (piezoelectric/piezoresistive, strain gage, surface mounted thermocouples, ultrasonic sensors, optical sensors, etc.). Some of them have operating temperatures that do not allow insertion into the mold cavity and touching the molten material (indirect, contact-free measurement), while others are able to operate in these extreme conditions (direct measurement) (*Ageyeva, Horváth & Kovács, 2019*).

The frequencies for logging the IMM and mold data may differ depending on the further use of this data, as well as on the size of the produced part. For example, if a part is manufactured using micro-injection molding and its cycle time is equal to a couple of seconds, a higher logging frequency might be of interest, however, if a bigger part is produced and the cycle time is longer the frequency of interest might be lower.

Commercially available MES and solutions proposed by academia

MES are computerized systems used in manufacturing to obtain the necessary data from the production floor in order to optimize the production output (*Mahmoud et al., 2015*). A suitable data exchange protocol needs to be utilized to establish communication between an IMM and an MES or a PC. Currently, the most widely used protocols are EUROMAP63 and EUROMAP77. EUROMAP63 was developed in 2000 and uses file-based data transfer (*EUROMAP, 2020*). EUROMAP77 was released in 2018 and is based on OPC/UA to be applied in Industry 4.0 cases. EUROMAP77's downside is that it is not directly connected to the IMM's control loop and thus it might not be able to guarantee

real-time or soft real-time communication. Another example of a data exchange interface is ENGEL Machine Interface (EMI). EMI is directly connected to the machine's control loop via Ethernet (TCP/IP protocol) and communication is based on XML byte streams. EMI is much faster in comparison to EUROMAP63, as EMI exchanges data approximately every 0.05 s, while EUROMAP63 exchange period is 1-2 s, based on the authors' experience.

There exist a significant number of manufacturing execution systems applicable to injection molding. Table 1 shows functionality in terms of the process data collection and openness of some of the MES systems used by the injection molding companies. There are, other systems available on the market, however, most of them are similar to each other in a sense that their functionality misses either (a) openness for changes and addition of custom modules developed by the system's users and not only developers, (b) synchronized acquisition of data from both machine and 3d party sensors or (c) ability to adjust the sampling rate. The functionality of the available systems could be adjusted to enable realization of (a)–(c), thereby enabling a new use-case of the commercially available systems making them more open and flexible for research and development purposes.

When it comes to the solutions proposed by academia, the authors have found only a few examples of the development of open APIs capable of solving the above-described challenges. The literature, however, does include examples of systems for the acquisition of data from sensors additionally installed on the IMMs, and descriptions of rapid testing of novel algorithm prototypes. In (*Tellaeche & Arana, 2013*) a "*flexible data acquisition system that allows rapid implementation of complex algorithms to assess their correct performance and can be integrated in the quality control process*" is proposed. The system is based on one of National Instruments (NI) data acquisition systems (DAQs) and is used to log data from two force sensors installed in a mold. No data from in-built machine sensors are being used in this study, due to the inability to access it. A similar case is described in (*Zhou et al., 2017*), where mold temperature, screw displacement and velocity data are recorded through the installation of additional (not provided by machine manufacturer) sensors in the IMM, as well as the mold. In *Zhao et al. (2014)*, different types of sensors and probes (thermocouples, displacement and pressure probes) are installed in an injection molding machine and data collection cards are used to transform analog, discrete and temperature signals to the digital ones. Most of this data can be acquired from sensors already installed in an IMM by its manufacturer, but due to the restriction of commercial software and hardware, new sensors needed to be installed to get easy access to data and to test necessary algorithms and routines developed by researchers. Zhang, Mao (*Zhang et al., 2016*), on the other hand, are trying to use some of machine built-in sensors, namely hydraulic pressure sensor to estimate nozzle pressure values and increase controllability of the injection molding process. In (*Charest, Finn & Dubay, 2018*) pressure transducers, thermocouples, velocity and position sensors, and flow sensors are installed and "*interfaced using National Instruments data acquisition boards and a PC*" to acquire the IMM data.

Most of these reported studies use systems for data acquisition from 3d party sensors, and are not using the machine built-in sensors, due to the lack of a suitable way to acquire the data and communicate with the IMM controller. Development of an open API for the

Table 1 Comparison of commercial MES.

	Requirement	TIG authentig (*KraussMaffei, 2020*)	KraussMaffei MaXecution (*KraussMaffei, 2020*)	AURBURG Host computer system (ALS) (*ARBURG, 2020*)	MES HYDRA (*MDPV, 2020*)	SAP Manuacturing Execution (*SAP, 2020*)	Polmak Plastik QMaster (*Polmak_Plastik, 2020*)
1	Open for changes and addition of modules by the user (not producer)	No	No	No	No	Yes (mostly for business applications)	No
2	Compatibility with EUROMAP63, EUROMAP77, EMI	Yes	No	Yes	Yes	Yes	Yes
3	Acquisition of machine and process parameters	Yes	Yes	Yes	Yes	Yes	Yes
4	Set values of machine and process parameters	Yes	Yes	Yes	Yes	Yes	Yes
5	Synchronized acquisition of data from built-in and additional sensors	No	No	No	No	No	No
6	Possibility to change sampling rate	No	No	No	No	No	Yes

IMMs by either MES producers or academia would bring new opportunities for research and analysis of the machine and process data.

Requirements to an open application programming interface for IMMs

Based on the above-described challenges to the commercially available MES, as well as on the project proposal related to this research certain functional requirements were formulated:

1. The API should be compatible with different communication protocols (EUROMAP63, EUROMAP77, EMI).

2. The interface should be open so that building upon existing software and rapid algorithm prototyping is possible if/when necessary.

3. The API should be capable of providing real-time or soft real-time logging depending on the end-user's needs. Real-time scheduling communication means that missing a data sampling deadline (each 0.3 s, for example) is considered an error, while soft real-time means that it is better not to miss the deadline, but it is not an error to do so (*Lee & Seshia, 2016*). In case of the prototype further described in this article, a soft real-time logging and a 2 Hz sampling rate were required to be implemented based on the stakeholders' requirements extracted from the related project proposal.

4. The acquisition of several parameter measurements per production cycle should be ensured.

5. It should be possible to get values of up to 97 desired machine and process parameters of an IMM (depending on the IMM's model) and additionally installed sensors (if any).

6. The option of setting values of machine and process parameters, where the parameters are accessible, should be provided.

7. The setting of the values should not conflict with the IMM safety installations and non-permitted values that are too high or too low shall not be overwriting the permitted ones.

8. The developed system should ensure the synchronized acquisition of data from machine built-in sensors and 3d party sensors. In case of the prototype solution, logging of data from the IMM's built-in sensors should be synchronized with temperature and pressure mold multi-sensors.

In addition, such non-functional requirement as security needs to be considered. API implementations should ideally allow application layer security—such as authenticated access over secure HTTP via TLS, but this aspect is out of scope for this work. To sum up: a simple, open and safe application programming interface (API) which can get and set values of different parameters on the IMM, as well as values from additionally installed/external sensors, should be developed. To meet the above-listed requirements the API needs to include the functions/ methods shown in Table 2.

DESIGN AND IMPLEMENTATION OF THE API'S PROTOTYPE

An API is a software product that includes a number of clearly defined methods for communication between different components, in our case, between an IMM, a PC

Table 2 **"ParameterList" description.**

Name of method	Description of method
start_logging(sampling_rate)	By executing this method the system starts logging IMM's process parameters with specified by a user sampling rate. "sampling_rate" parameter is the user-desired sampling rate in Hz. The logging sequence is stopped by launching the *idle()* method.
get_process_param(names)	This method returns actual and set values of chosen process parameter or list of parameters. Parameter name(s) is/are given as "names" argument of the method. This method is independent of the control loop.
get_async_act_sample(params)	Returns unsynchronized current value(s) of process parameter(s). Name of one or more parameters are given as "params" argument of the method. Method is independent of the control loop.
set_process_param(name,value)	This method sets a desired value of a specified process parameter on the IMM. Arguments "name" and "value" mean respectively process parameter's name and desired value of that parameter.
get_samples(number_of_shots)	Method returns data sampled during certain number of production cycles (shots) buffered in a FIFO queue. "number_of_shots" parameter defines how many shots/cycles in the past user is interested in.
event()	Set API's state to an event, can be launched externally with method *event_sample()*.
event_sample()	Trigger an event based sampling. The state has to be an event.
idle()	Set interface's state to idle, a passive state where the connection is maintained.
disconnect()	Connection is ended and the API is shut down.

and/or a data acquisition system connected to mold sensors. To develop the proposed API prototype, the Unified Process methodology for software development has been followed. This methodology includes four main phases: inception, elaboration, construction and transition (*Jacobson, Booch & Rumbaugh, 1999*).

The Python3 programming language was chosen to program all the prototype modules, as it is easy to use, has a number of versatile features and libraries, can be utilized to practice distributed computing and be further extended in C++ or C. The IMM used in this study supports EUROMAP63 and EMI data exchange protocols. As mentioned before, EMI is capable of exchanging data significantly faster than EUROMAP63, therefore, it was selected to be used in the prototype implementation. PCMEF (presentation, control, domain and foundation) architectural framework (*Maciaszek & Liong, 2005*; *Madeyski & Sochmialek, 2005*) was applied for development of the API prototype's distributed architecture. At the same time, the OSI 7 layer model (Open Systems Interconnection model) (*Kurose & Ross, 2010*) was followed for the creation of the communication design of the API, since the EMI exchange protocol is also based on this model.

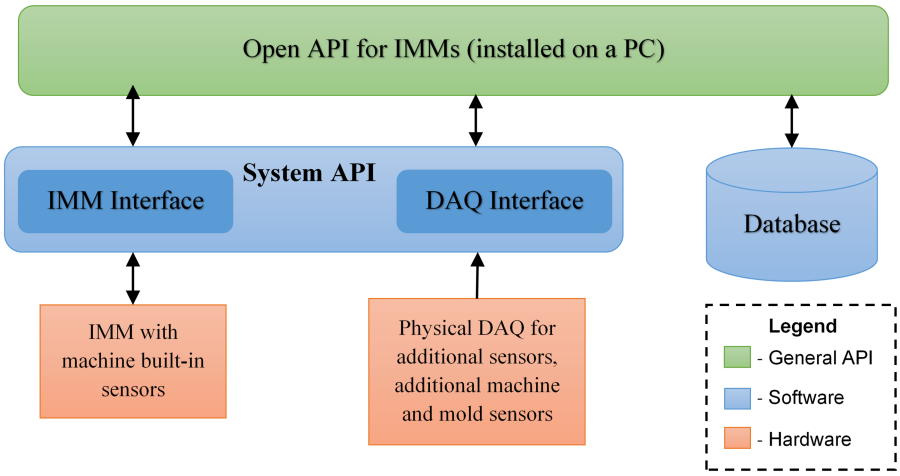


Figure 2 **Open API for IMM architecture diagram.**
Full-size DOI: 10.7717/peerjcs.302/fig-2

Description of the proposed system

The proposed prototype API includes *database* and *System APIs*, which combines *IMM interface* and *Data acquisition system for additional sensors (DAQ Interface)* modules, as shown in Fig. 2. The *System API* is used (1) for the establishment of a connection with the injection molding machine and requesting and setting the corresponding parameter values, (2) for acquisition of data from any additionally installed machine and mold sensors. Note that the data from the IMM is only accessed through establishment of connection with the IMM and not acquired directly from the IMM sensors. This is due to a fact that the IMM has its own DAQ that is connected to the installed sensors and is logging their values during the machine runs. The *database* is needed for organized storage and easy access to sampled data.

The API prototype

The API prototype uses Pyro4 (*Pyro, 2020*), xml.etree.ElementTree, socket, thread, queue and datetime Python libraries, where Pyro4 is Python Remote Objects library that is used to enable distributed computing between network-based objects.

The data is logged only when the machine mold is closed in order to minimize the amount of memory used for data storage. To detect the mold closing, a "*mold force*" parameter is used. The software unit waits until the mold force parameter reaches a value of $\geq$300 kN and then starts data collection. It stops when the mold force parameter value becomes lower than the above-specified value.

Currently, data from each production cycle (in this case defined as from mold closing to mold opening) are written to a separate file of a user-specified type (.csv, .json, .pickle) and are stored in a folder chosen by a user on the PC connected to the IMM. A proper database management unit needs to be further developed and added to the system in future versions, to provide a scalable way of storing process data.

Accessing the IMM parameters data

To access the necessary IMM parameter values the prototype uses a list of Uniform Resource Identifiers (URIs) and their correspondence to different parameters on the IMM for the establishment of machine to machine communication. The URIs list was obtained from private communication with ENGEL. A .csv file was created that included all URIs and machine and process parameters of the IMM provided by ENGEL. This file can be updated to enable or disable the collection of certain parameters depending on the necessities of a user.

The API prototype can enable three different modes "*Idle*", "*Fixed cycles*" and "*Flexible cycles*" depending on the application. In the "*Idle*" mode controller ensures that connection is up and that application is able to access the interface's methods for asynchronous interaction, while the "*Fixed cycles*" mode is a periodical sampling mechanism of parameters from the IMM that samples all of the parameters simultaneously. The sampling results are available in the queue for the application process. "*Flexible cycles*" mode takes care of the unbalanced update sampling frequencies to avoid oversampling.

Machine parameters on the IMM can have different update frequencies depending on the sampling limitations of the data acquisition system on the IMM. These frequencies can also be manipulated by the user via the IMM's general user interface. The updating frequencies typically vary from 10 to 1000 ms. At the first run in the "*Flexible cycles*" mode, the interface acquires the sampling frequency of each parameter from the IMM and groups parameters that have the same update rate. In this mode, the interface checks the last datetime, when each parameter group was updated, and determines if it is time to acquire a new portion of data from the IMM.

Acquisition of additional sensors data

When it comes to acquisition of data from sensors additionally installed on an IMM or in its molds, the data logging is synchronized. In order to enable the synchronization, the event_sample() method described in Table 2 needs to be triggered for both IMM and the DAQ used for the additional sensors data sampling.

This study included the use of a mold with two pressure and temperature multi-sensors. Figure 3 depicts a typical scenario of the API's use and interaction of its components, where the necessary parameters data is accessed on the IMM and logged from the additional sensors through the RevPi. Communication with an IMM is established through the IMMProxy module and the 3d party sensors DAQ though the DAQProxy and the requested parameter values are obtained.

EVALUATION OF THE API PROTOTYPE'S PERFORMANCE

Used hardware

In order to develop and test that the application programming interface is able to satisfy requirements specified in "Requirements to an open application programming interface for IMMs", an "ENGEL insert 130" vertical injection molding machine with CC300 control unit was used. A mold for ISO 527-2 (*ISO, 2012*) dog bone parts (15 mm thickness) with two temperature and pressure Kistler multi-sensors type 4021B10H1P1 was used to test the

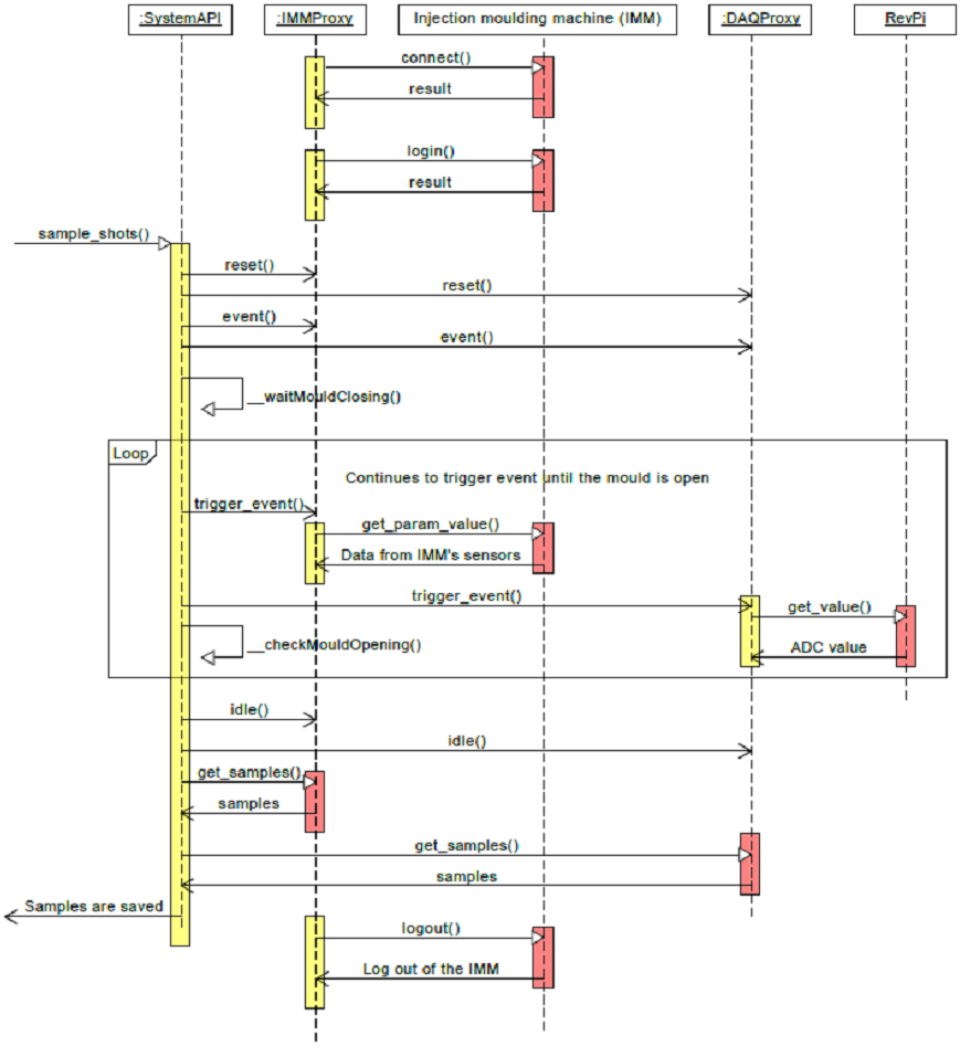


Figure 3 **Interaction Sequence Diagram showing ordering of calls to the API and underlying calls for data acquisition.**

Full-size DOI: 10.7717/peerjcs.302/fig-3

system's performance with different sampling rates. The sensors' characteristics are shown in Table 3. The dog bone specimen parts (Fig. 4) are mostly used to define mechanical properties of various materials and are used by both industry and academia for those purposes.

To transform analog signals from sensors into digital ones a RevPi (*RevPi, 2020b*)—Industrial Raspberry Pi is used, namely RevPi Core3 and RevPi Analog Input Output (AIO) modules. RevPi Core3 is a computing unit that uses Raspbian Jessie as the operating system and Preempt-RT patched kernels of version 4.9.52-rt37-v7. RevPi AIO is an analog input/output extension module for the RevPi Core3 which includes necessary analog inputs and outputs to connect to the sensors and to send converted sensor signals to the Core3 unit. The modules are connected through PiBridge (RevPi's modules connector).

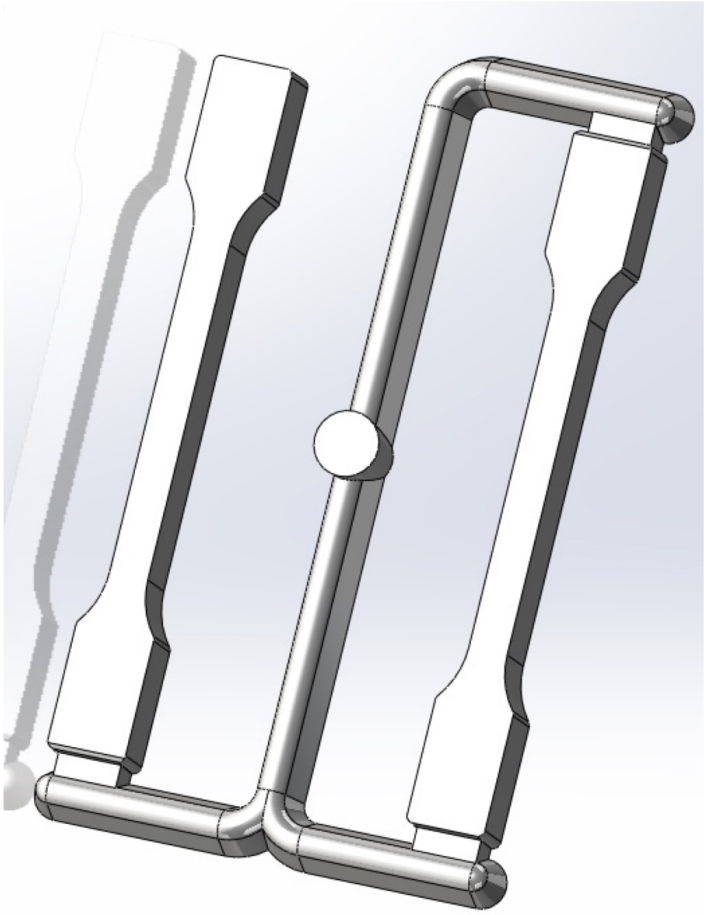


Figure 4 Dog bone specimen (ISO 527-2) with 15 mm thickness.
Full-size DOI: 10.7717/peerjcs.302/fig-4

Table 3 Pressure and temperature sensor characteristics.

Specifications	Kistler type 4021B10H1P1
Model	Measuring chain
Calibration	Calibrated by Kistler
Measuring Range pressure (Bar)	0..1000
Measuring Range temperature (°C)	0..350
Temperature accuracy (°C)	±5
Diameter (mm)	21
Height (mm)	91.5
Natural Frequency (kHz)	>165

The RevPi AIO performs analog-to-digital conversion (ADC) with a 24-bit delta-sigma converter (*Baker, 2016*) (model ADS1248) (*RevPi, 2020a*).

The reason for choosing RevPi is that, unlike commercial DAQs from manufacturers like NI and HBM, RevPi has a significantly lower cost and is an open, modular industrial

Table 4 Statistics of overhead measurement for RevPi.

Sampling rate [Hz]	Mean [s]	Std. dev. [s]	5th percentile	95th percentile	Overhead over 0.025s [%]
100	0.02532	0.02049	0.022	0.025	0.96
50	0.02611	0.02392	0.023	0.025	1.29
33	0.03001	0.02246	0.023	0.03	1.14
2	0.5006	0.00082	0.5	0.501	0.00

Table 5 Statistics of overhead measurement for IMM DAQ.

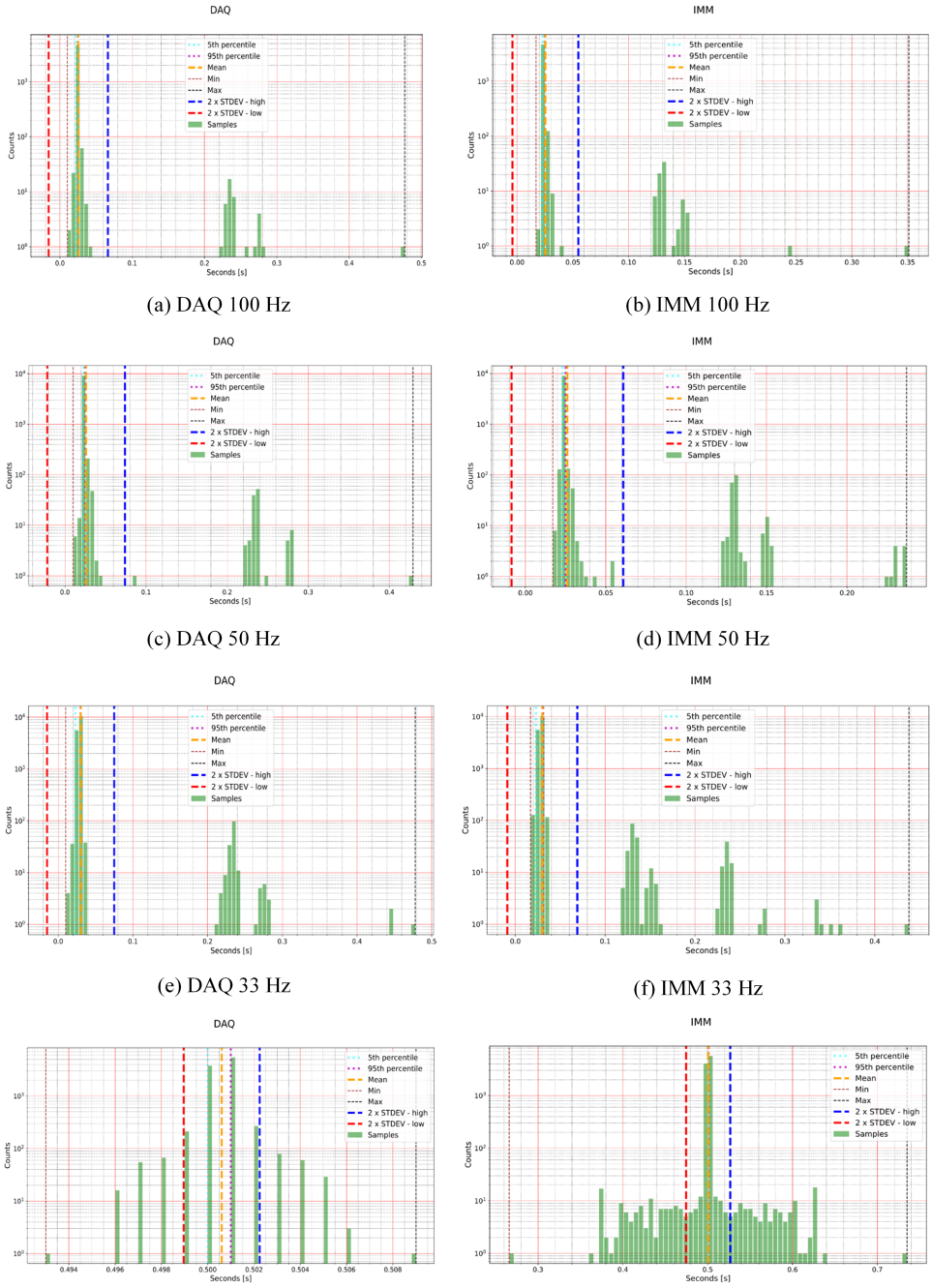
Sampling rate [Hz]	Mean [s]	Std. dev. [s]	5th percentile	95th percentile	Overhead over 0.025s [%]
100	0.02533	0.01475	0.023	0.025	1.67
50	0.02611	0.01736	0.023	0.025	2.43
33	0.03001	0.01945	0.023	0.031	1.73
2	0.5006	0.01303	0.5	0.501	2.34

PC (*RevPi, 2020b*). Due to this, it can provide flexibility in which software alternatives can be used together with it. The biggest disadvantage in comparison to commonly used commercial data acquisition systems is RevPi AIO's maximum sampling rate at 640 Hz. In practice, the sampling rate can be significantly lower due to load on the PiBridge. Further on, the update rate of the data values is reduced by a factor of 5 according to the documentation of the RevPi. As a result, if the analog-to-digital conversion sampling rate is 640 Hz, it will provide an update rate of 128 Hz on RevPi Core3. Due to this RevPi might not be the optimal solution, however, as the described API is a prototype, RevPi is a good demonstration tool.

System's performance in terms of real-time and soft real-time logging

In order to assess the system's performance in terms of its ability to comply with different sampling rates, the API was installed on the RevPi, where three different processes needed to be handled: handling the IMM, handling Kistler sensors signals and synchronization of data acquisition from the injection molding machine and mold sensors. All tests were conducted at length of 4900 samples for sampling rates at 100, 50, 33 and 2 Hz. The IMM machine was turned on and the RevPi was connected to the mold sensors, however, no IMM production runs were performed, therefore, most of the parameter values logged are constant or equal to zero. The ability of the system to set the corresponding machine parameter values was not tested during these tests. The collected data can be found in the supplemental files. Tables 4 and 5 show the mean, standard deviation, 5th and 95th percentiles, as well as the percentage of samples with overhead over 0.025 s among the logged sampling rates, while Fig. 5 depicts histograms for the corresponding sampling rates data. The overhead was calculated based on the following equation:

$$Overhead = \left(sampling\_rate(real)_{n+1} - sampling\_rate(real)_n\right) - sampling\_rate(given),$$



(a) DAQ 100 Hz (b) IMM 100 Hz

(c) DAQ 50 Hz (d) IMM 50 Hz

(e) DAQ 33 Hz (f) IMM 33 Hz

(g) DAQ 2 Hz (h) IMM 2 Hz

Figure 5 Histograms showing observed times for sampling at different rates. (A) DAQ 100 Hz. (B) IMM 100 Hz. (C) DAQ 50 Hz. (D) IMM 50 Hz. (E) DAQ 33 Hz (F) IMM 33 Hz. (G) DAQ 2 Hz (H) IMM 2 Hz.

Full-size DOI: 10.7717/peerjcs.302/fig-5

where $sampling\_rate(real)$ is a sampling rate performed by the system for samples $n+1$ and n.

Based on the results shown in Tables 4 and 5, it is easy to see that real-time restrictions of the system apply. For the test with 100 Hz sampling rate, the mean sampling rates that the system performs are around 0.025 s (39.49 Hz) on both IMM and the RevPi. This means that the system cannot comply with the given sampling rate. Examining results from sampling at 50 Hz, it is possible to see that the system performs better than with 100 Hz sampling rate. In this case, the performed mean sampling rate is around 0.026 s (38.29 Hz) (for both IMM and the RevPi).

Even better results can be seen at sampling rate at 33 Hz. Here, the average sampling is close to 0.03 s or 33.3 Hz, while the number of overhead values over 0.025 s is 1.14 and 1.73 for the RevPi and the IMM's DAQ respectively. This indicates that 33 Hz can be one of the appropriate sampling rates for the proposed API.

At the same time, 2 Hz sampling rate has the mean value for both IMM and the RevPi at around 0.5 s (1.99 Hz). On the IMM there are 2.34% of samples that have the overhead value higher than 0.025 s, while on the RevPi there are none. Regarding real-time demands, results show that a number of samples with the overhead value higher than 0.025 s are present at logging with any of the tested sampling rates except for the 2 Hz rate for the RevPi. This happens because the system tries to compensate for the sampling deviation, when launched in the "*Flexible cycles*" mode to avoid oversampling. The RevPi demonstrates that it is able to perform real-time sampling with 2 Hz or lower sampling rate and soft real-time on higher speeds. To improve the system's performance in term of the real-time performance a more powerful DAQ needs to be used.

At the same time, API's unpredictability comes, among other things, from Python's memory management mechanism. It uses reference counting collector and generational garbage collector, known as "*gc module*" for memory reclaim (*PythonSoftwareFoundation, 2020*). Unlike many other languages, it does not necessarily release the memory back to the operating system, but instead keeps some parts of already allocated memory for use in the future. It is also possible to see that the IMM process is more unpredictable than the RevPi process. This is caused by necessity to establish the server/client connection with the IMM, while there is no such necessity, when the RevPi is directly connected to the sensors and acquires the data from them. It is important to remember that the RevPi is used to access and record the IMM's parameter values, while their acquisition is performed by the IMM's internal DAQ, and only the mold sensors are directly connected and the corresponding signals are logged by the RevPi.

DISCUSSION AND FUTURE WORK

The developed API prototype complies with requirements that were defined in "Requirements to an open application programming interface for IMMs" and is a modular system open for any necessary changes. Its openness is the main difference between the proposed API and commercially available systems with similar functionality. It is possible to see from Table 1 that listed commercial MES (except SAP Manufacturing

Execution (*SAP. SAP Manufacturing Execution, 2020*)) do not include the possibility of adding any modules developed by the system's user, unless a new module is bought from the software producer. At the same time, they also do not allow to change parameters sampling rate and synchronization of data acquisition from built-in or external sensors.

The open API prototype, on the other hand, can be modified to include additional modules. The use of these modules can assist in increasing the quality yield and the overall controllability of the injection molding process. The open application programming interface is developed to simplify data collection and processing for optimization of the injection molding process and bring IMMs closer to being cyber-physical systems capable of self-regulation and self-control.

Keeping in mind operational safety, it should be noted that the prototype does not in any way hinder operation of the IMM's own control system. As a result, if a parameter value that contradicts with the IMM manufacturers' safety principles is attempted to be set, the value will not be updated on the IMM. At the same time, if a value is updated, in some cases, it might take time to reach the new stationary set point on the IMM. At the moment, the proposed prototype does not check if the IMM has reached the new value and this needs to be controlled by the IMM operator/user.

Future work should include development of the corresponding classes for the prototype's use with other data exchange protocols, such as EUROMAP77 to enable connection to IMMs of other manufacturers; testing of the system with other injection molding machines; further development of the database management unit to allow storage of all the necessary process data. Development of the prototype in terms of the real-time monitoring through implementation of the specialized process architecture (for example, non-blocking queues, concurrent read/write buffers) also needs to be continued. At the same time, a graphical user interface (GUI) or a connector to an existing Industry 4.0 process monitoring platform for the API needs to be implemented. The system's security also needs to be worked on. In addition, a question of how often the data should be sampled needs further and deeper review. Collaboration with IMM manufacturers, as well as OPC/UA working group would be highly beneficial for further development of the open API for IMMs and the injection molding process compliance with Industry 4.0 paradigms. Adding functionality of the proposed API to the commercially available MES would make them attractive to a broader range of users, including those interested in extensive research and analysis of the injection molding process.

An important aspect of Industry 4.0 is simplification of the integration and the communication across different units and machines at the shop floor (*Negri, Fumagalli & Macchi, 2017*; *Kritzinger et al., 2018*). A major motivation for the development of the API, is the possibility to enable easy access to machine tool and process parameters data during injection molding for several categories of IMM users. In order to move towards Industry 4.0, both hardware and software should become more open (without compromising systems security and operational safety) for additional modifications by users instead of staying as restricted as it is.

SUMMARY

This paper has provided requirements, description of the development and capabilities of the open API prototype for injection molding machines. The interface is open for external interaction with the machine controller, allowing logging and setting values of process parameters. The openness of the prototype also provides possibilities for rapid algorithm prototyping and testing, when developing control strategies in the laboratory. The prototype has been tested at 100, 50, 33 and 2 Hz sampling frequencies and can comply with the 33 and 2 Hz rates. Setting of the machine parameters, however, has not been performed during these tests.

As to the authors' knowledge, there are currently no such open systems available. Similar functionality might be provided by commercially available MES, but these systems often have several limitations, such as: high cost, synchronization issues of data logged from injection molding machine and additional sensors, etc.

The API prototype proposed in this paper is developed to provide soft real-time data communication between a PC and an IMM through EMI data exchange protocol, as well as allow connection with RevPi for data acquisition from sensors installed in a mold. Even though RevPi has some advantages in terms of cost, openness and high flexibility, its main disadvantage is the low sampling rate and a more powerful DAQ might be necessary to acquire data with higher sampling rates. The interface delivers the possibility of logging up to 97 machine and process parameters and allows building upon the API to integrate necessary data processing algorithms or other modules. The system has been developed using Python3 programming language and is open for changes, such as adding EUROMAP77 or other desirable data exchange protocols. Distributed computing can be practiced with the help of this API to provide additional flexibility and robustness. The prototype is inspired by concepts of Industry 4.0 and cyber-physical systems.

The developed API prototype is based on necessity to move towards implementation of Industry 4.0 and cyber-physical systems. The authors hope that it will be able to provide the necessary flexibility for rapid prototyping of methods and algorithms that can lead to enhancement of injection molding machines' capabilities related to self-optimization and control.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work was supported by the Norwegian Research Council as part of the "MegaMould" Project (project number: 256819) and through the SFI Manufacturing Project (project number: 237900). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
Norwegian Research Council as part of the "MegaMould" Project: 256819.
SFI Manufacturing Project: 237900.

Competing Interests

Mats Larsen is a research scientist at SINTEF Manufacturing and Ole Vidar Lyngstad is a research director at SINTEF Manufacturing.

Author Contributions

- Olga Ogorodnyk and Mats Larsen conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Ole Vidar Lyngstad conceived and designed the experiments, performed the experiments, authored or reviewed drafts of the paper, and approved the final draft.
- Kristian Martinsen conceived and designed the experiments, authored or reviewed drafts of the paper, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The code is available on GitHub: https://github.com/SintefManufacturing/IMM_API

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.302#supplemental-information.

REFERENCES

Ageyeva T, Horváth S, Kovács JG. 2019. In-mold sensors for injection molding: on the way to industry 4.0. *Sensors* **19(16)**:3551 DOI 10.3390/s19163551.

ARBURG. 2020. Host computer system (ALS). *Available at https://www.arburg.com/en/products-and-services/injection-moulding/production-management/host-computer-system-als/* (accessed on 25 June 2020).

Baker B. 2016. *How delta-sigma ADC work, Part 1*. Dallas, Texas, USA: Texas Instruments Incorporated.

Charest M, Finn R, Dubay R. 2018. Integration of artificial intelligence in an injection molding process for on-line process parameter adjustment. In: *Annual IEEE International Systems Conference (SysCon)*. Piscataway: IEEE.

EUROMAP. 2020. EUROMAP 77 –data exchange interface between injection moulding machines and MES. *Available at https://opcfoundation.org/markets-collaboration/plastics-and-rubber-machinery/* (accessed on 25 June 2020).

ISO. 2012. ISO 527-2:2017 Plastics –determination of tensile properties –Part 2: test conditions for moulding and extrusion plastics. *Available at https://www.iso.org/standard/56046.html* (accessed on 25 June 2020).

Jacobson I, Booch G, Rumbaugh J. 1999. The unified process. *IEEE Software* **1999(3)**:96–102.

KraussMaffei. 2020. MaXecution. *Available at https://km.kraussmaffei.com/imm-en/maxecution.html* (accessed on 24 June 2020).

Kritzinger W, Karner M, Traar G, Henjes J, Sihn . 2018. Digital twin in manufacturing: a categorical literature review and classification. *IFAC-PapersOnLine* **51(11)**:1016–1022.

Kurose JF, Ross KW. 2010. *Computer networking: a top-down approach*. Boston: Pearson Addison Wesley.

Kurt M, Kamber OS, Kaynak Y, Atakok G, Girit O. 2009. Experimental investigation of plastic injection molding: assessment of the effects of cavity pressure and mold temperature on the quality of the final products. *Materials & Design* **30(8)**:3217–3224 DOI 10.1016/j.matdes.2009.01.004.

Kurt M, Kaynak Y, Kamber OS, Mutlu B, Bakir B, Koklu U. 2010. Influence of molding conditions on the shrinkage and roundness of injection molded parts. *The International Journal of Advanced Manufacturing Technology* **46(5–8)**:571–578 DOI 10.1007/s00170-009-2149-x.

Lee J, Kao HA, Yang SH. 2014. Service innovation and smart analytics for Industry 4.0 and big data environment. *Product Services Systems and Value Creation: Proceedings of the 6th Cirp Conference on Industrial Product-Service Systems* **16**:3–8.

Lee EA, Seshia SA. 2016. *Introduction to embedded systems: a cyber-physical systems approach*. Cambridge, Massachusetts London, England: MIT Press.

Maciaszek L, Liong BL. 2005. *Practical software engineering: an interactive case-study approach to information systems development*. Boston: Pearson Addison Wesley.

Madeyski L, Sochmialek M. 2005. Architectural design of modern web applications. *Foundations of Computing Decision Sciences* **30(1)**:49–60.

Mahmoud MI, Hassan Ammar H, Hamdy MM, Hassan Eissa M. 2015. Production operation management using manufacturing execution systems (MES). In: *2015 11th international computer engineering conference (ICENCO)*. IEEE.

MDPV. 2020. Manufacturing Execution System HYDRA. *Available at https://www.mpdv.com/en/products-solutions/mes-hydra/#c2077-1* (accessed on 3 July 2020).

Negri E, Fumagalli L, Macchi M. 2017. A review of the roles of digital twin in CPS-based production systems. *Procedia Manufacturing* **11**:939–948 DOI 10.1016/j.promfg.2017.07.198.

Ogorodnyk O, Martinsen K. 2018. Monitoring and control for thermoplastics injection molding a review. *Procedia CIRP* **67**:380–385 DOI 10.1016/j.procir.2017.12.229.

Osswald TA, Hernandez-Ortiz JP. 2006. *Polymer processing modeling and simulation*. Munich: Carl Hanser Verlag, 651.

Polmak_Plastik. 2020. Polmak plastik products. *Available at https://www.polmakplastik.com/en/productsdetail-manufacturing-execution-systems-mes-products-479.html* (accessed on 6 July 2020).

Pyro. 2020. Python remote objects - 4.60. *Available at https://pypi.org/project/Pyro4/* (accessed on 25 June 2020).

Python Software Foundation. 2020. gc - Garbage Collector interface. *Available at https://docs.python.org/3/library/gc.html* (accessed on 25 June 2020).

RevPi. 2020a. How to configure analog input. *Available at https://revolution.kunbus.com/tutorials/uebersicht-aio-2/analoge-eingaenge-konfigurieren-2/* (accessed on 9 July 2020).

RevPi. 2020b. Industrial Raspberry Pi. *Available at https://revolution.kunbus.com/revolution-pi-series/* (accessed on 25 June 2020).

Saldivar AAF, Goh C, Li Y, Yu H, Chen Y. 2016. Attribute identification and predictive customisation using fuzzy clustering and genetic search for Industry 4.0 environments. In: *Software, Knowledge, Information Management & Applications (SKIMA), 2016 10th International Conference on*. IEEE.

SAP. 2020. SAP manufacturing execution. *Available at https://www.sap.com/products/execution-mes/technical-information.html#extensibility* (accessed on 3 July 2020).

Tellaeche A, Arana R. 2013. Rapid data acquisition system for complex algorithm testing in plastic molding industry. *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering* **7(7)**:1391–1395.

TIG. 2020. TIG authentig. *Available at https://www.tig-mes.com/en/software/tig-authentig/* (accessed on 25 May 2020).

Vrabič R, Kozjek D, Butala P. 2017. Knowledge elicitation for fault diagnostics in plastic injection moulding: a case for machine-to-machine communication. *CIRP Annals* **66(1)**:433–436 DOI 10.1016/j.cirp.2017.04.001.

Yin S, Ding SX, Xie X, Luo H. 2014. A review on basic data-driven approaches for industrial process monitoring. *IEEE Transactions on Industrial Electronics* **61(11)**:6418–6428 DOI 10.1109/TIE.2014.2301773.

Zhang Y, Mao T, Huang Z, Gao H, Li D. 2016. A statistical quality monitoring method for plastic injection molding using machine built-in sensors. *The International Journal of Advanced Manufacturing Technology* **85(9–12)**:2483–2494 DOI 10.1007/s00170-015-8013-2.

Zhao P, Zhou H, He Y, Cai K, Fu J. 2014. A nondestructive online method for monitoring the injection molding process by collecting and analyzing machine running data. *The International Journal of Advanced Manufacturing Technology* **72(5–8)**:765–777 DOI 10.1007/s00170-014-5711-0.

Zheng R, Tanner RI, Fan X-J. 2011. *Injection molding: integration of theory and modeling methods*. Berlin, Heidelberg: Springer Science & Business Media.

Zhou X, Zhang Y, Mao T, Zhou H. 2017. Monitoring and dynamic control of quality stability for injection molding process. *Journal of Materials Processing Technology* **249**:358–366 DOI 10.1016/j.jmatprotec.2017.05.038.