An Internet of Things Architecture for Lab-scale Prototypes of Real-Time Simulation

Giovanni Lugaresi, Vincenzo Valerio Alba, Andrea Matta

An Internet of Things Architecture for Lab-scale Prototypes of Real-Time Simulation

Giovanni Lugaresi[1], Vincenzo Valerio Alba[1], Andrea Matta [1*]

***Abstract*— Recently, new technologies for data acquisition, storing and communication enabled to improve the performances of manufacturing systems with new production management and control policies. In the next years, we may expect several architectures exploiting data exchange between a real and a digital manufacturing system. This raises the issue on how to test the frameworks since the availability of real manufacturing systems to researchers is scarce or simply costly. In this work, we propose a novel architecture which is suitable for lab-scale models of manufacturing systems. The developed architecture has been successfully applied to a test case which will be used by an Italian SME as demonstrator for ERP software capabilities.**

I. Introduction

Production enterprises are deeply involved in the Industry 4.0 revolution. This phenomenon provided a set of technologies to be exploited in the industrial context, such as Internet of Things (IoT), cloud computing, simulation, Big Data Analytics, Augmented and Virtual Reality, Radio Frequency Identification (RFID), Artificial Intelligence and Machine Learning [1], [2]. Thanks to these enabling technologies, several innovative solutions have been developed, such as Digital Twins (DT) [3], Cyber Physical Systems (CPS) [4], and Virtual Factories [5]. In general, it is possible for physical systems to be increasingly interconnected with their digital counterparts.

The system formed by the union of a physical system and a digital substrate can be called Cyber Physical System (CPS), and Cyber Physical Production System (CPPS) are CPSs featured by high interconnection between production assets and computational tools [4]. Hence, all the decisions taken within such systems can benefit from data and information coming from both shop-floor measurements and management software (e.g. Customer Relationship Management, Material Requirements Planning). Moreover, these data and information are shared in a quasi-instantaneous way, allowing managers and automated systems to take real-time data-driven decisions. When CPSs gain the possibility to sense and analyze data in real time, a set of new opportunities for production management has arisen. Data-driven decisions bring along unquestionable advantages when compared with static arrangements. Indeed, production policies defined a-priori may be optimal within a precise set of boundaries, but turn out to be detrimental on system performances when applied in actual scenarios because real systems are always subject to some degree of stochasticity. A possible countermeasure is to design solutions which are robust against many different scenarios, although never really optimal for any particular condition rather for the average performance of the system. On the contrary, online decisions can be tailored exactly for the circumstances of the very moment they are examined, hence they can suit a particular system status.

The easiness of communication between and within shop floors, enterprises managing level, suppliers and clients is generally associated with the concept of integration. Internet of Things technologies such as embedded electronics, softwares, sensors, networks and communication protocols that allow different objects to communicate and share data are at the root of systems integration [7].

The literature on architectures and frameworks related to the interplay between real and digital manufacturing systems is rich and we may expect several other contributions in the future [1]. Meanwhile, it is hard for researchers and practitioners to test these architectures on real manufacturing systems. Indeed, providing a real case validation of such algorithms and policies is not easy. The proposed algorithms and policies could be compared and validated in two ways:

1) By using a real system. In this case, it is necessary to allocate an entire production system and to change the already established production strategies. The risk is to spend a lot of time and resources in setting up the system to reproduce the desired behavior, rather than iterating the proposed logic.
2) By exploiting digital models of production systems. In this case, several issues related to the physical system might not be considered. For instance, data handling between the real and digital model must be designed properly to retrieve the desired information (e.g., current number of parts in a buffer).

In this work, a lab-scale CPPS is introduced. Both the physical and the software parts of the CPPS are developed. The physical systems are models of industrial production lines, built with LEGO® components[1]. The goal is to obtain a setting that allows for testing several iterations of real-time simulation algorithms logic, while maintaining the physical dimension and the interplay between a digital model and the real production system. The rest of the paper is organized as follows: section II discusses about the industrial context of production management; section III introduces the architecture for lab-scale prototypes, and section IV shows

*Corresponding Author: andrea.matta@polimi.it
[1] Department of Mechanical Engineering, Politecnico di Milano, 20156 Milano, Italy.

[1]LEGO, the LEGO logo, MINDSTORMS, and EV3 are trademarks and copyrights of the LEGO Group.

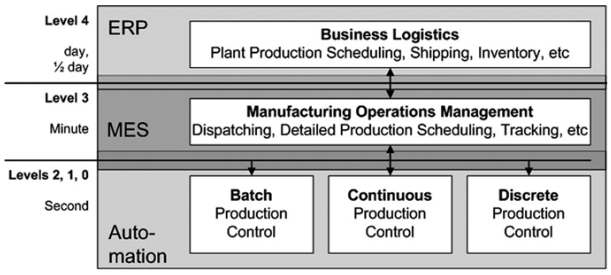

Fig. 1. ISA95 levels from [9].

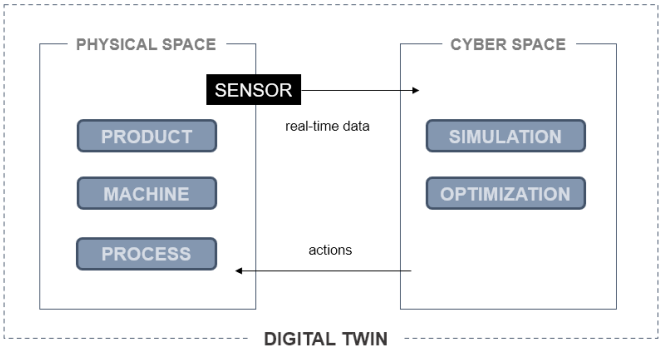

Fig. 2. Simple illustration of a DT exploiting real time simulation-optimization (adapted from [12]).

an application example; conclusions are in section V.

II. Industrial Context

One of the most established standards for production management and control is the ANSI/ISA95 [8]. It is based on five hierarchical levels as shown in Fig. 1. Level 0 is associated with the physical process of manufacturing; level 1 to actuators and sensors governing and recording the process; level 2 to the control logic and supervision of the underlying Supervisory Control And Data Acquisition (SCADA) activities; level 3 to the management of manufacturing operations (MES); level 4 to the management of the entire firm (ERP). These levels are often seen as a pyramid, with the lowest level connected to the physical process and the highest level linked to the enterprise management. According to ISA95 standard, each level is able to communicate only to adjacent levels [9], [10]. However, recent research claims that this pyramidal vision is no longer a dogma in industrial IT systems [11]. CPPSs are disrupting this hierarchical view of the production system thanks the data sharing capabilities made possible by IoT and cloud computing. The integration of functionalities is promising to bring benefits to production systems such as increased flexibility, easier capability to respond to unexpected events and higher horizontal and vertical integration of the system [10].

A high level of integration enables the creation of Digital Twins [6]. A Digital Twin is *the virtual and computerized counterpart of a physical system* [6]. Digital Twins allow to make smarter decisions and improve performances such as productivity, efficiency, flexibility, reconfigurability and integration. Digital Twins are based on automatic data exchange in both ways between the physical and the digital objects. Hence, we may state that a DT is able to: (1) mirror the status of the physical object, (2) simulate its future behavior in several scenarios, and (3) select the most promising one and directly communicate them to the physical system [3].

Automated data acquisition and communication paved the way for a real-time exploitation of simulation [12], [13]. In Real-time Simulation (RTS) applications, data are acquired from a real system and are used to synchronize a purposely built simulation model to the real system state. Then, alternative scenarios are simulated and the one that leads to best performances is applied online [14]. Fig. 2 explains the role of RTS within a CPS structure: the digital twin is the whole entity composed by both the physical and the cyber spaces; the physical space is able to share in real time the measured data with the cyber space; the cyber space contains simulation and optimization tools able to quickly implement actions on the physical space. Synchronization between the physical systems and their simulation-based DTs is necessary for obtaining the right boundary conditions and correctly simulating the system future behavior [15]. By simulating different scenarios and policies it is then possible to take those decisions that are able to optimize the performance of a production system within specific time frames [3].

In this work, we develop a lab-scale model of a production system and the related software architecture. The introduction of physical production systems model in a laboratory allows for investigating hardware-related issues such as data sharing between the various layers composing a CPPS, while being able to implement and iterate the proposed logic in reasonable times. Further, an architecture for generic lab-scale models of a production system is a tool for validating several management algorithms and rules for production planning and control which are based on data exchange with a real system.

III. Proposed Architecture

The developed architecture consists of the physical level and three software levels (Fig. 3). The physical level (section III-A) is built with LEGO MINDSTORMS, namely both structural components and sensors, actuators, and PLCs (EV3 intelligent bricks). The first two software levels are coded in *python* and are purposely built for the lab-scale environment. The choice of *python* as programming language is not restrictive and the proposed architecture can be extended to other languages. The *Execution Level* (section III-B.1) controls the PLCs. It is associated with the actuators and sensors which control the manufacturing system. The *Logic Level* (section III-B.2) dictates the system logic. The logic level is associated with SCADA functions of monitoring and supervising the process and the Manufacturing Execution System (MES) services of management of the operations and release of production orders. The fourth level has been called *Overlying Software* (section III-B.3). It can be composed

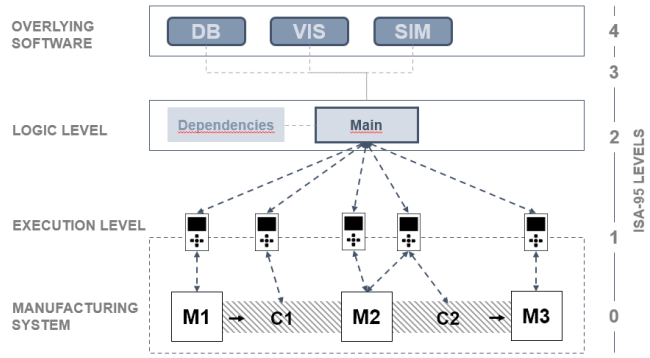

Fig. 3. The developed architecture in comparison with the ISA95 levels.

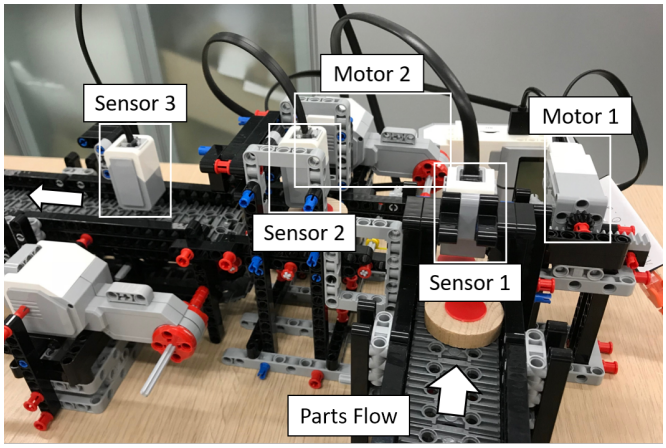

Fig. 4. Example of station model in a Lego Manufacturing System (LMS).

by several software tools, for instance dedicated to firm management and Enterprise Resource Planning (ERP), or simulation-optimization tools. All the levels are integrated thanks to messages exchanged via the Message Queue Telemetry Transport (MQTT) communication protocol (section III-C).

A. Physical System

In this work, the physical models of production systems are built with LEGO MINDSTORMS components, which include not only structural pieces such as beams, shafts and conveyor belts, but also actuators, sensors and PLCs (EV3 intelligent bricks). These models can be used to replicate the behavior of a real production line by moving parts such as spheres or discs through the different stations of the line, following the proper route and reproducing operations times by holding pieces for a specific time span. Fig. 4 shows an example of a station built with LEGO. Notice that the ability to retrieve the system state at any moment in time depends on the number and position of the sensors along the system. In this work, the open-source EV3DEV operating system (`www.ev3dev.org`) has been exploited. EV3DEV is based on Debian Linux. This OS allows the execution of *python* scripts for controlling the EV3 sensors and motors through dedicated libraries. Further details on Lego Manufacturing

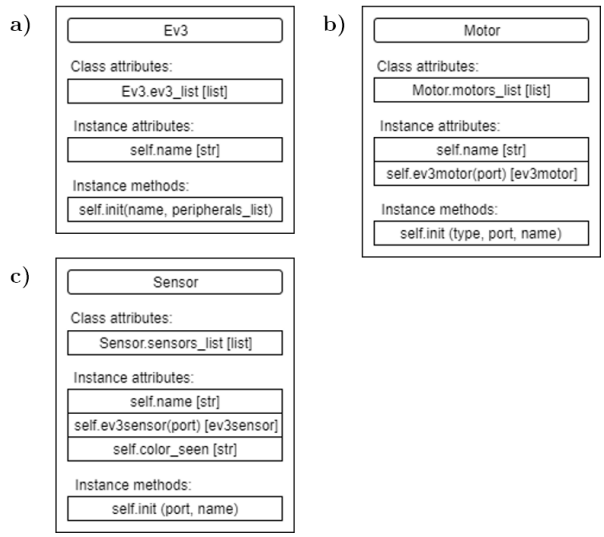

Fig. 5. The developed classes for the Execution Level: (a) Ev3, (b) Motor, and (c) Sensor.

Systems (LMS) can be found in previous related works [16], [17], [18].

B. Software Levels

1) The Execution Level: this level represents the software running on the PLCs that controls the physical system, which consists in a script running on each EV3 device. The execution level is responsible of two tasks: (1) releasing run/stop commands to the motors whenever required; (2) acquiring and communicating the sensor outputs. The motors activation is triggered by specific messages that communicate the actions that the motors should perform and releases the proper run/stop orders only when these messages are received. Similarly, the sensors outputs can be communicated "On Demand" or "On Change" via specific messages written in the JavaScript Object Notation (JSON) format. The code of the execution level is object-oriented. Three classes represent the three available pieces of hardware: the PLCs (i.e. the EV3 intelligent bricks), the motors, and the sensors. Fig. 5 summarizes the classes that have been developed:

- The **Ev3** class (Fig. 5a) represents the logic controller. At its instantiation, all the motors and sensors executed by the EV3 are contextually instantiated.
- The **Motor** class (Fig. 5b) has the attributes *name* and *ev3motor*. *ev3motor* is an object available in the EV3DEV library that allows for running the motors through *python* commands.
- The **Sensor** class (Fig. 5c) has the attributes *name*, *ev3sensor* and *color_seen*. *ev3sensor* is an object available in the EV3DEV library that allows running the motors through *python* commands. *color_seen* is an attribute that indicates the last color seen by the sensor.

2) The Logic Level: this level manages the structure, execution rules, constraints and characteristics of the manufacturing system. It is composed by the following basic parts which are always present:

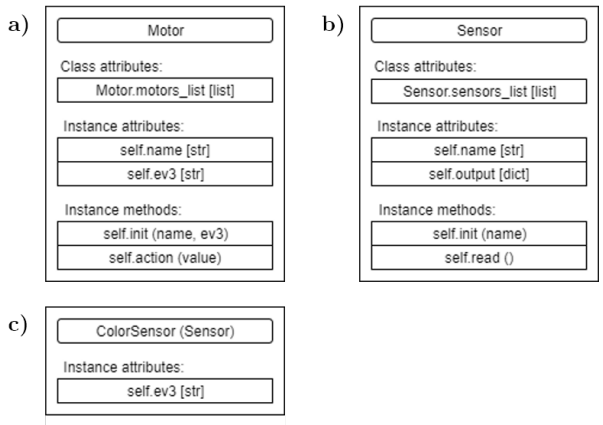

Fig. 6. The classes developed for the Logic Level: (a) Motor, (b) Sensor, and (c) ColorSensor.

- A description of the physical system layout (motors and sensors) and the configurations of the EV3s. This description is sent as a message to the EV3s in order to configure each of them at startup.
- The messages definition for being able to communicate with the execution level and the EV3s and also to export significant data towards other management services.
- The definition of the system logic. This includes the number and type of stations and the actions which should be performed. Notice that this part may differentiate a lot depending on different production systems.

In this work, the logic level is a *python* script running on a central controller (for instance, an Industrial PC). The code is object-oriented and consists of the following classes (Fig. 6):

- The **Motor** class (Fig. 6a) has the attributes *name* and *ev3*. *ev3* is the name of the PLC device that controls the motor. Motor instances also possess methods corresponding to the executable actions (section III-C). Whenever one of these methods is called, a corresponding message requesting the motor activation is published, this message will then be processed by the execution level which will perform the prescribed operation.
- The **Sensor** class (Fig. 6b) possesses an attribute which is a list of all the instantiated Sensors objects. Moreover, each Sensor instance has the attributes *name* and *output*.
- The **ColorSensor** class (Fig. 6c) is a subclass of the class **Sensor**, with the addition of an instance attribute *ev3* representing the name of the EV3 that is connected to the sensor. Hence, it is possible to locate the position of each sensor along the line.

3) Overlying Software: this level includes software which may exploit the data from the system to perform several high-level operations, such as updating a production plan or calibrating the digital twin of the physical system. Following we list some examples.

- *Data Flows Management Tools* are intermediary services to transfer data between software utilities.

TABLE I

SUMMARY OF THE MESSAGES EXCHANGED ACROSS DIFFERENT LEVELS OF THE DEVELOPED ARCHITECTURE.

Message Topic	Sent From - Received By	Purpose
data	Logic Level - Overlying SW	Data extraction
ev3_config	Logic Level - Execution Level	EV3s configuration
hello	Execution Level - Logic Level	EV3s configuration
sensor/request	Logic Level - Execution Level	Reading sensor output on demand
sensor/on_demand	Execution Level - Logic Level	Reading sensor output on demand
sensor/on_change	Execution Level - Logic level	Reading sensor output on change
motor/action/action_name	Logic Level - Execution Level	Activating motors
stop	Any - Execution/Logic Level	Stopping software execution

- *Database Management Systems (DBMS)* allow the storage, manipulation and query of the acquired data. After raw data are measured and stored, they may be manipulated and aggregated for obtaining useful information. For instance, the time series of a machine state may be used to derive both availability and reliability indicators.
- *Dashboards* are applications for the real-time visualization of data as well as custom indicators and indexes. These are useful in an industrial environment since they allow managers and technical staff to have all the relevant information they need for taking the right decisions any time they have to.
- *Simulation Models* can be exploited for building digital twins of physical systems. If properly aligned with the system state, a simulation model can be used for foreseeing the future performances of the same systems in given scenarios [4]. It is thus possible to simulate different production and management policies in order to determine which one is optimal. Simulation models need initial conditions for being able to run a simulation. These initial conditions are generally the status of the simulated system. The developed architecture allows to communicate the data measured on the shop floor, hence it is able to infer the system status.
- *Cloud Computing* is among the enabling technologies of Industry 4.0 [19]. Cloud services also enable the interface with software such as ERP, CRM, and MRP.

C. Communication

The communication between the software levels is possible thanks to an IoT infrastructure based on the MQTT protocol. Table I summarizes the messages exchanged. All messages are in the JSON format. This protocol allows all the PLCs to send and receive messages to any kind of IoT-compatible device connected to the network.

Hence, it is possible to communicate and store data from the real system while exploiting the message-based *Machine-To-Machine* communication between the single PLCs. Following we list two significant examples.

- The *sensor/request* messages are sent from the logic level to the execution level and they contain the request of reading a specific sensor output. In this case, the payload of the message contains the sensor name and

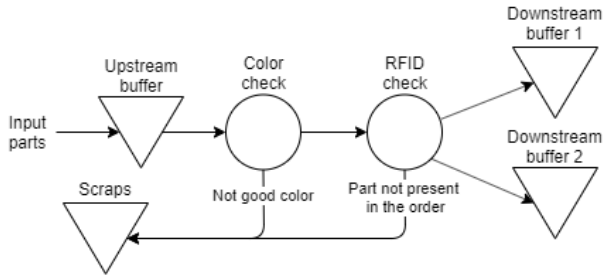

Fig. 7. The logic layout of the designed part selector.

the name of the EV3 device which is controlling the sensor.

- The *motor/action* messages are sent from the logic level to the execution level and contain the actions that should be executed by the motors. In our physical system the possible actions are the following: run forever at a certain speed; run for a certain amount of time at a certain speed; turn the axis to a specific angle value; run back-and-forth of a specific angle value; stop. Notice that other actions can be designed accordingly to specific system requirements. The payload is a JSON object containing the motor name, the name of the EV3 that executes that motor and the value that describe how to execute the prescribed action.

IV. Test Case

A. Introduction

The architecture proposed in this work has been tested within the construction of a part selector station; the system is part of a demo factory through which an Italian SME tests ERP and MES software components. The demo factory simulates a manual assembly line composed by several successive stations. The pieces to be assembled are LEGO boards on which different bricks are attached according to the specific orders. In the first station, each LEGO brick is marked with an unique RFID identifier. The bricks are inserted in cave plastic spheres of different colors. Then, they are clustered in batches of 50 pieces and stored in a warehouse. When a production order arrives, a batch is sent to the parts selector that collects the pieces needed for the order completion, according to the color of the container ball and to the brick RFID tag. Once the necessary pieces have been selected, an operator proceeds to the manual assembly of the bricks over the board and to a quality verification through image recognition of the same board. Finally, assembled boards are stored in the warehouse.

B. Part Selector System

The parts flow diagram of the Part Selector station is showed in Fig. 7. The parts are charged into an upstream buffer in groups of fifty. They are then processed, two at a time. Non-useful pieces are treated as scraps and discarded. The selector consists of two stages of verification: (1) color check, where it is controlled that the plastic sphere is of the right color. (2) RFID check, where it is verified that the brick

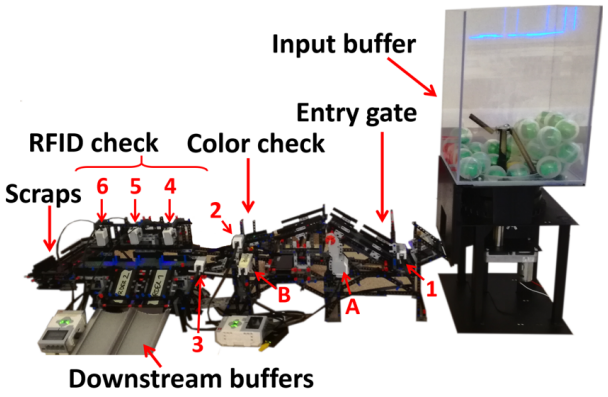

Fig. 8. The part selector physical system.

TABLE II
DATA EXTRACTED FROM THE PIECES SELECTOR MODEL.

Name	Category (Type)	Description
counter_in	Counter (int)	Number of pieces that have entered the system
counter_out	Counter (int)	Number of pieces that have leaved the system
counter_color	Counter (int)	Number of pieces that passed in Color Check station
counter_rfid	Counter (int)	Number of pieces that passed in RFID Check station
counter_color_scrap	Counter (int)	Number of pieces discarded by Color Check station
counter_rfid_ord1	Counter (int)	Number of pieces pushed into Order 1 slider
counter_rfid_ord2	Counter (int)	Number of pieces pushed into Order 2 slider
counter_rfid_scrap	Counter (int)	Number of pieces discarded by RFID Check station
color_station_state	State (bool)	State of the Color Check station (idle or busy)
rfid_station_state	State (bool)	State of the RFID Check station (idle or busy)
color_data	State (str)	Last seen color recorded by Color Check station
rfid_data	State (int)	Last seen RFID tag ID recorded by RFID Check station

contained inside the sphere is effectively present in the order to be completed.

In this work, the physical model has been built exploiting LEGO components. In particular, the pieces selector model takes advantage of the spherical shape of the pieces to transport them over slides exploiting gravity. Fig. 8 shows the selector with its components. Two EV3s control four motors (A to D) and six sensors (1 to 6). Thanks to the developed architecture, the pieces selector is controlled and supervised by the software levels (section III-B). The production plan is executed through appropriate messages published whenever required by the orders received from the management software. Moreover, useful data are extracted from the system: (1) counters are used to keep track of how many parts are in production and how many are left to complete the production plan, (2) state variables can tell the state of each selection stage (e.g., the RFID reader state), (3) work-in-progress is monitored through memory of the last piece seen by the sensor on the line. These data are saved into a database and visualized on a dashboard for monitoring the process. Table II summaries the published data. The test has been performed using a PC equipped with an i5 CPU 1.6 GHz and 8 GB memory.

V. Conclusion

Industry 4.0 introduced a set of new possibilities for manufacturing systems. For example, production planning prob-

lems may now be solved in real-time or updated with a very high frequency. This work proposed a software architecture based on lab-scale physical models which allows to study several kinds of production systems. Indeed, several different types of manufacturing systems can be turned into lab-scale models developed in very short times thanks to the ease of construction of the LEGO physical models. The introduction of such lab-scale models with an IoT Architecture can change the way manufacturing systems engineering research is done and tested, since it enables to test production planning and control algorithms exploiting real-time data communication through real industrial components (e.g., PLCs, gateways, sensors). Hence, the related research projects can reach a higher Technology Readiness Level (TRL).

Several issues still need to be solved. The architecture has been exploited for studying aggregate performance measures (e.g., average waiting times). Further work is needed to be able to investigate the influence of a specific process parameter (e.g., clamping force, workpiece temperature) on the performance of the system. In this case, a scaled physical process on the parts is needed (e.g., 3D printing). In the future, we aim to formalize the boundaries of application of this work. Further, we plan to use the proposed architecture for Real-time Simulation experiments, such as real-time scheduling. Last but not least, the architecture can be extended to prescribe comparisons among simulation results from models of real systems in similar settings, before the identification of proper corrective actions.

Acknowledgements

The construction of the physical model and the related software architecture has been partially funded by the *Sme.UP Group* (www.smeup.com).

References

[1] L. D. Xu, W. He, S. Li, Internet of things in industries: A survey, IEEE Transactions on Industrial Informatics 10 (4) (2014) 2233–2243. doi:10.1109/TII.2014.2300753.

[2] A. G. Uriarte, A. H. Ng, M. U. Moris, Supporting the lean journey with simulation and optimization in the context of Industry 4.0, Procedia Manufacturing 25 (2018) 586–593. doi:10.1016/j.promfg.2018.06.097.
URL https://www.sciencedirect.com/science/article/pii/S2351978918306255

[3] W. Kritzinger, M. Karner, G. Traar, J. Henjes, W. Sihn, Digital Twin in manufacturing: A categorical literature review and classification, IFAC-PapersOnLine 51 (11) (2018) 1016–1022. doi:10.1016/j.ifacol.2018.08.474.

[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 (2) (2016) 621–641. doi:10.1016/j.cirp.2016.06.005.

[5] T. Tolio, M. Sacco, W. Terkaj, M. Urgo, Virtual factory: An integrated framework for manufacturing systems design and analysis, Procedia CIRP 7 (2013) 25–30. doi:10.1016/j.procir.2013.05.005.

[6] E. Negri, L. Fumagalli, M. Macchi, A Review of the Roles of Digital Twin in CPS-based Production Systems, in: Procedia Manufacturing, Vol. 11, Elsevier, 2017, pp. 939–948. doi:10.1016/j.promfg.2017.07.198.
URL https://www.sciencedirect.com/science/article/pii/S2351978917304067

[7] A. Köksal, E. Tekin, Manufacturing execution through e-FACTORY system, Procedia CIRP 3 (1) (2012) 591–596. doi:10.1016/j.procir.2012.07.101.

[8] C. Johnsson, ISA 95 - How and where can it be applied?, in: Technical Papers of ISA, Vol. 454, 2004, pp. 399–408.

[9] B. S. De Ugarte, A. Artiba, R. Pellerin, Manufacturing execution system - A literature review, Production Planning and Control 20 (6) (2009) 525–539. doi:10.1080/09537280902938613.

[10] D. Rossit, F. Tohmé, Scheduling research contributions to Smart manufacturing, Manufacturing Letters 15 (2018) 111–114. doi:10.1016/j.mfglet.2017.12.005.
URL https://www.sciencedirect.com/science/article/pii/S2213846317300871

[11] R. Cupek, A. Ziebinski, L. Huczala, H. Erdogan, Agent-based manufacturing execution systems for short-series production scheduling, Computers in Industry 82 (2016) 245–258. doi:10.1016/j.compind.2016.07.009.

[12] W. Yang, K. Yoshida, S. Takakuwa, Digital Twin-Driven Simulation for a Cyber-Physical System in Industry 4.0 Era, DAAAM INTERNATIONAL SCIENTIFIC BOOK (January) (2017) 227–234. doi:10.2507/daaam.scibook.2017.18.

[13] Y. Tan, W. Yang, K. Yoshida, S. Takakuwa, Application of IoT-Aided Simulation for a Cyber-Physical System, in: Winter Simulation Conference, 2018, pp. 4086–4087.

[14] G. Lugaresi, A. Matta, Real-time simulation in manufacturing systems: Challenges and research directions, in: Proceedings - Winter Simulation Conference, Vol. 2018-Decem, 2019, pp. 3319–3330. doi:10.1109/WSC.2018.8632542.

[15] O. Cardin, P. Castagna, Proactive production activity control by online simulation, International Journal of Simulation and Process Modelling 6 (3) (2011) 177–186.

[16] D. Travaglini, Manufacturing Systems Based on LEGO-Robotics: Development of Physical and Digital Models, M.Sc. Thesis.

[17] G. Lugaresi, D. Travaglini, A. Matta, a Lego Manufacturing System As Demonstrator for a Real-Time Simulation Proof of Concept, in: Winter Simulation Conference, 2019.

[18] V. V. Alba, A lab-scale modeling architecture for testing digital twin and real-time simulation, M.Sc. Thesis.

[19] M. Bortolini, F. G. Galizia, C. Mora, Reconfigurable manufacturing systems: Literature review and research trend, Journal of Manufacturing Systems 49 (October) (2018) 93–106. doi:10.1016/j.jmsy.2018.09.005.