Advances in Science and Technology
Research Journal

Advances in Science and Technology Research Journal 2022, 16(5), 110–119
https://doi.org/10.12913/22998624/152927
ISSN 2299-8624, License CC-BY 4.0

Received: 2022.06.06
Accepted: 2022.10.17
Published: 2022.11.01

Demonstrator of a Digital Twin for Education and Training Purposes as a Web Application

Stanislaw Flaga[1], Kacper Pacholczak[2*]

[1] Faculty of Mechanical Engineering and Robotics, AGH University of Science and Technology, ul. Mickiewicza 30, 30-059 Krakow, Poland

[2] Faculty of Metals Engineering and Industrial Computer Science, AGH University of Science and Technology, ul. Mickiewicza 30, 30-059 Krakow, Poland

* Corresponding author's e-mail: stanislaw.flaga@agh.edu.pl

ABSTRACT
The concept of Industry 4.0 has now become a fact. One of its key technological solutions – the digital twin – serves as a bridge between the real and virtual worlds. Designs for both products and tools to make these products are generated in the virtual world. Thanks to the simulation capabilities of these digital replicas, it is possible to eliminate design flaws well before the creation of physical prototypes. Thus, the question naturally arises as to what degree these mathematical models of objects, processes or services replicate their physical counterparts. A correctly generated digital twin is not only a model or visualisation of its counterpart; it also reflects its dynamic behaviour. The issue of digital twins is a very broad one, and currently on the market, there are appearing an increasing number of tools available for the development of these twins. More and more often, 3D modelling software can be integrated with a control system model, facilitating the testing of newly designed objects in the virtual world. This paper presents the concept of building simplified digital twins in a web application environment. In addition to educational usage, the presented idea should find application in the design of small production lines, significantly affecting the cost of producing a digital twin.

Keywords: Industry 4.0, digital twin, web application.

INTRODUCTION

A decade has passed since the name "Industry 4.0" was first used at the Hannover Fair. This idea of widespread digitization starting around 2017 is considered one of the most critical technology trends [1]. Inherent in this idea is the technology of digital twins (DT).

When talking about digital twins and digital factories, we mean multi-layer simulation models consisting not only of the implementation of mathematical models of physical components such as e.g. drives, sensors, tools, structural elements or control devices but also of the impact of the environment by simulating various types of disturbances resulting both from the modelled technology and from emergency situations, Wagg et al. [2]. Nowadays, one can find single devices as well as entire production lines in the form of a digital twin Kutin et al. [3]. Using the virtual and augmented reality benefits in the visualisation layer already at the design stage makes it possible to conduct initial operator training based on a twin digital simulation model.

Nevertheless, the history of twin projects is long. The breakthrough was the construction of a twin capsule for the Apollo 13 program used for crew training. At a critical moment, it accelerated efforts to save astronauts' lives through operational testing of a physical model possessed by NASA. Of course, it was not a digital twin in today's sense, but its effectiveness cannot be denied. Digital twin projects are interdisciplinary projects that go far beyond IT. The technologies used to produce DTs depend primarily on their subsequent usage. Increasingly, DT is being integrated with the physical world in addition to fulfilling its native functionality to create an environment for semi-physical simulation [4].

The native functionality of DT is the real-time acquisition and analysis of data collected from a process and feeding mathematical models with this data. The massive amount of data collected forces the use of Big Data technologies for its analysis, Figueres et al. [5].

The years 2017 and 2018 proved groundbreaking in the transition from DT considerations at the theoretical and laboratory research level to implementation in real processes. Researchers Maulshree et al. [6] analyzed 13 different industries in which DT implementation will affect progress. The industries analyzed include manufacturing, agriculture, education, medicine, retail and others. The application of virtual and augmented reality tools in DT design offers many possibilities. It allows a new approach to controlling industrial equipment and making measurements in a modified image of the world, for example, distance measurement for positioning an object in space, Lalik, Flaga [7] or automation of vision-based quality control – Oborski, Wysocki [8], Gaska et al. [9]. DT technology provides new opportunities in the field of robot programming; the concept of hybrid robot programming was presented by Kuts et al. [10]. It is impossible to see the progression in the industrial application of collaborative robots. It poses new challenges for DT developers. The classic robot, most often enclosed behind an enclosure and separated from humans, is a relatively simple object to model. A cobot has direct contact with a human, making it challenging to incorporate it into a DT structure. The conditions that should be met for the introduction of DT collaborative robots for adaptation cobots for the process were presented by Pizon et al. [11]. Every production line requires interprocess transport equipment for its operation. Warehouse handling should also not be forgotten. Automated Guided Vehicle (AGV) robots are playing an increasingly important role in these areas, and by using DT, their working environment can be optimized, Staczek et al. [12].

Implementing Industry 4.0 ideas using DT technology has great potential to improve the production process efficiency, maintenance and product quality. It is important to remember that a genuine factory uses many resources in the production process: manually controlled machines, machines with built-in automation, semi-finished products with repetitive control required, people, and process procedures. The fully implemented DT is a conglomeration of all resource models and communication mechanisms. The communication mechanisms in systems with DT are relatively complex. It is due to the need to exchange information between models and their real-world counterparts. DTs in real manufacturing processes should perform calculations according to computational models that are as accurate and verified as possible. Building a full-fledged DT is costly and time-consuming, as presented by Georgakopoulos, Bamunuarachchi [13] using an example from the food industry.

The high costs of implementing Industry 4.0 with DT are reasonably quickly discounted in large corporations or government agencies. Small and medium-sized businesses can rarely afford such costs. It has been recognised, and Yasin et al. [14] have provided a roadmap for implementing DT in small and medium-sized businesses. Building and implementing DT are the costs of developing infrastructure to enable real-time data acquisition, hardware and specialised software. To a large extent, these are also the costs of highly qualified personnel. The range of tools used for creating digital twins is wide, and it is not easy to compare them. Usually, manufacturers of 3D design tools add new functionalities to integrate the model with simulation and control modules, etc. – here, an example is the Mechatronics Concept Designer Module of the NX System, Herbus et al. [15], Febronio et al. [16].

Small and medium-sized businesses, for the most part, have not yet implemented the paradigms of Industry 4.0, and the European commission's report "Industry 5.0 Towards a sustainable, human-centric and resilient European industry" (Pizon, Gola [17]) has adopted the paradigms of the Industry 5.0 idea. Such rapid industrialisation progress brings a high demand for specialised education.

Now, at the lowest level of machine control, there are digital, programmable control devices in the form of PLCs or dedicated embedded systems. PLCs are predominant in industrial practice. It is the reason why there is a high demand for education towards learning PLC programming. Effective learning to program PLCs requires access to the object or its simulator. Therefore, the problem of simulation has long been addressed. It is not uncommon that, in addition to using a model of a machine, a production line as a simulated

PLC-controlled object, attempts are made to generate source code that is later implemented into the PLC, e.g. Thapa et al. [18]. An exciting example of conveyor simulation using the Siemens Tecnomatix Process Simulator tool was presented by Ruzarovsky et al. [19].

There are several commercial solutions for simulating a control object and linking the simulation to the PLC. These include Factory i/o – Riera, Viqario [20], Famic Automation Studio, among others. There are also limited possibilities to build object operation sequences with simulators provided by controller manufacturers, e.g. Siemens PLCsim. Most object simulation tools work as a desktop application communicating with the PLC or its simulator via the chosen communication protocol – in Siemens this will be ProfiNet. The authors decided to design and realise the DT project as a web application for which the runtime environment would be a browser. The prototype of the first object became a conveyor supporting inter-operational transport between 4 assembly stations in the Industry 4.0 Laboratory (Figure 1).

This paper aims to present the design and implementation of a simplified DT implemented as a web application. The application under construction was assumed to operate in two modes. In mode 1, it is possible to work online with the devices, acquire values of process variables from the real object and analyse them in the context of the DT model. Mode 2 implements the operation of the simulator and makes it possible to work without connection to the object using the data collected when working in mode 1.

The realised project allows the modelling of interactive simple process sequences (Lalik et al. [21]). The process variables defined in the model are assigned to variables in physical or simulated PLCs on which the control algorithm is implemented. Additionally, the realised project provides real-time visualisation of the process.

CONCEPTUAL DESIGN

The concept of functioning of the designed solution is characteristic of web applications. Three main parts can be distinguished here: client application, server and external services (Figure 2). The user is in direct contact with the client application, which role is to transmit the data entered by the user to the server and the appropriate visualisation of the received responses. It does not implement any logic related to the modelling of the automation components and devices.

Communication with the server takes place through the web API (Application Programming Interface) provided by the server. All application logic, models of real devices,

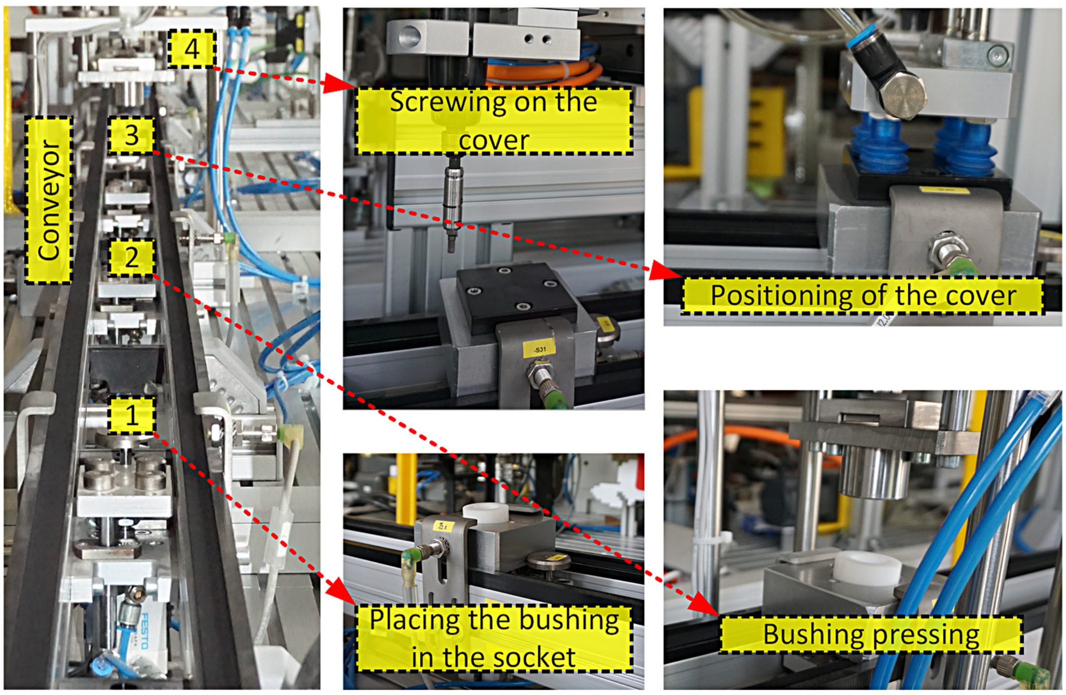

Fig. 1. View of four assembly stations with conveyor

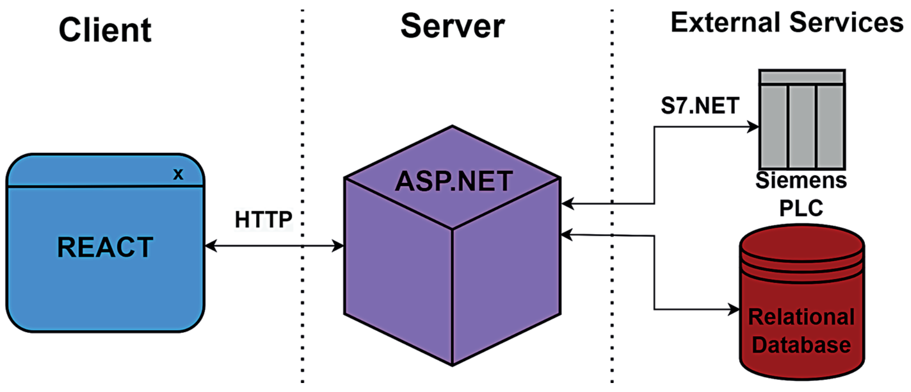

Fig. 2. Application flowchart

communication with the controller, DT system logs and user accounts are located on the server-side. Two external services were planned for the project: a database and a PLC. The database was used to store a vector of process configuration information, including user ID, PLC IDs, and the created digital twin configuration. It can be easily guessed that the driver (driver simulator) will be responsible for executing the control algorithm of the model created in the application. The server also handles the communication between the database and the controller(s).

SYSTEM ARCHITECTURE

The project was divided into two parts:

- web API – a programming interface responsible for handling the application logic,
- client application – interface used by the user.

The API is based on a layered architecture, as shown in Figure 3. This architecture ensures easy testing of the code and its modularity, i.e. the low correlation between the application and technologies that do not directly relate to the logic. It allows changing technologies if necessary. The architecture proposed by Martin [22] was followed here. The choice of technology is not critical – the chosen platform is.net, ASP. NET, and the C# language. The server uses an MS SQL engine for data storage. Communication with the controller was implemented based on the publicly available S7.NET library. At this stage, it was assumed that the designed DT would be able to cooperate with the controllers from the Siemens S7 family. To build a client application Typescript language with React library was used. Applications communicate with each other by cyclic, simulating the passage of time in the application and execution of HTTP queries by the client.

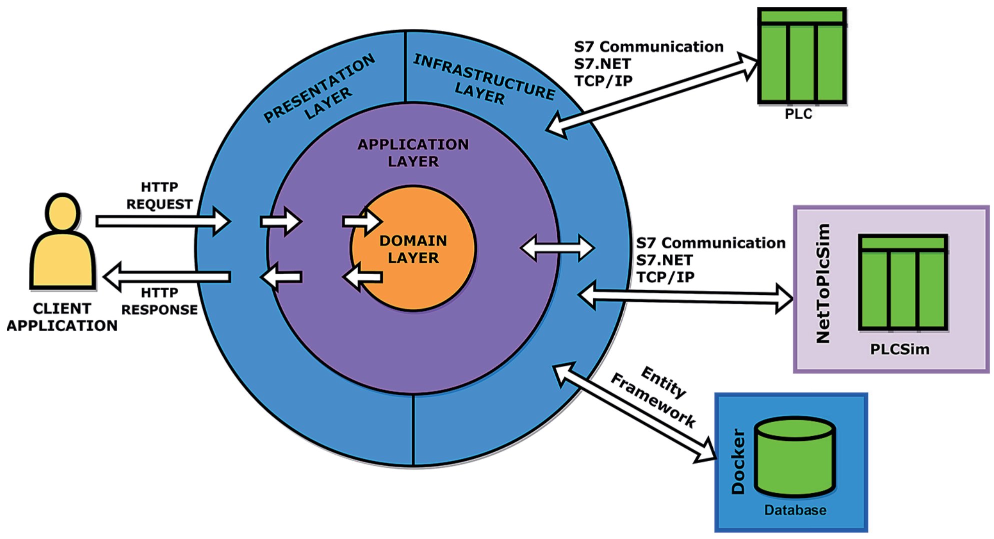

Fig. 3. Layered API structure

DOMAIN LAYER

The code at the heart of the application is called the domain layer. References to it occur throughout the application. The most important logic of the developed software is implemented here – modelling of selected automation elements and devices takes place here. In the case of the presented demonstrator, algorithms connected with moving virtual pallets on virtual conveyors and generating signals by virtual sensors are realised here.

The domain layer also contains abstractions of data sources – repositories. They define what data is to be provided by the sources without their detailed implementation. The used architecture is, therefore, domain-centric. Languages that support the object-oriented programming paradigm are well suited for modelling automation components and systems. The domain layer uses the concept of Domain-Driven Design, discussed by Evans, Evans in [23]. According to it, objects should implement logic characteristic of their physical counterparts. As simple wrappers for data, entities in the anaemic domain model should be avoided. The logic is implemented through mutations of domain objects due to executing the methods they provide.

An essential feature of the central layer is its independence – lack of connections with other layers results in complete separation of the domain rules from implementation details related to frameworks or libraries. As a result, the created models can be used in other applications. It also guarantees a high level of code testability.

APPLICATION LAYER

The layer surrounding the domain is called the application layer. Its principal function is domain orchestration – all application use cases are implemented here. This layer has data validation functionality and uses repository abstractions to retrieve relevant data and organise its flow. The domain and application layers are sometimes presented as a single core layer, defining how the developed software works.

The application layer is often created as a set of application services that handle use cases involving the same objects. The disadvantage of application services is the tendency to build extensive classes – the so-called eight-thousanders. Extensive classes are a problem both for building and executing their tests and reading the source code. They violate the principle of single responsibility.

In order to avoid these drawbacks, it was decided to use the mediator design pattern – it provides a loose connection between objects that interact with each other. Objects do not refer to each other directly; the communication takes place through the mediator object. A detailed description of this pattern was presented by Gamma et al. in their book [24] and Huda et al. [25]. In the case of the project, each of the occurring use cases was implemented in a separate class, whose methods are called precisely through the mediator. These classes were created according to the concept of Command Query Separation (Bruel [26]) – individual actions modify the system state (commands) or return values (queries). They never perform both actions simultaneously.

INFRASTRUCTURE LAYER

This layer acts as a link between the application and all the external services: it implements repositories, ensures communication with the drivers and defines the database model. Only this layer directly refers to technologies used for data access. This approach guarantees that the developed software is component-based – in case of necessity to change the used technologies, modifications should be made only in this layer.

The fundamental concept of the infrastructure layer is repositories, which are responsible for acquiring or storing data in external services. Libraries enabling object-relational mapping are usually used here. The application layer operates on abstractions contained in the domain, and therefore it is not connected to the infrastructure in any way. It guarantees the independence of the application from specific external services, such as databases or libraries providing communication with drivers. Any data source can be used if an appropriate adapter is created to provide all the behaviours required by the abstraction.

PRESENTATION LAYER

The external layer through which the server is accessed is called the presentation layer. Its role can be filled by window applications, mobile applications or, as in the case of the presented project, by the Web Application Programming Interface (Web API). API is responsible for handling HTTP requests, which are directed to the appropriate paths provided by it. They correspond to particular use cases of the application. The request is received in controllers, which results in the execution of the corresponding method. In these methods, a mediator is called, which redirects the requests deeper into the application without creating unnecessary links between objects in separate layers.

Speaking about the presentation, we should also mention the flexibility of the project. Suppose a need arose to change the type of application being created, for example, because of limitations related to the HTTP protocol. In that case, this particular layer should be replaced with one that would allow meeting business needs.

CLIENT APPLICATION

The element with which the user interacts directly in the client application running in a web browser. It cooperates with the web API by making HTTP requests. Through such queries, the client can, among other things, perform CRUD operations (Create, Read, Update, Delete) to model the automation system.

The application executes a regular (every 500 ms) HTTP request that simulates the elapsing of time. As a result of receiving this type of request, the server reads the state of the controller's outputs, makes appropriate changes to the state of the devices based on those outputs, then performs logic specific to them (e.g., Pallet Move), and then overwrites the state of the corresponding inputs to the controller. The response from this type of query is a set of data about the objects modified in the query. A detailed flow of data through the application is shown in Figure 4.

When creating the application, the focus was not only on providing support for all functionalities provided by the web API. The intuitive design is also of great importance, as it allows to start working with the application immediately after its launch.

EXPERIMENT

The experiment environment was a computer with a 4-core and 8-thread Intel Core i5 processor clocked at 1.60-3.90 GHz, 6 MB cache. The computer had 16 GB ram and Windows 10 Pro version 1903 installed. A Node.js and.net runtime environment was installed on the computer, running the client application and API. The database was run in a docker container (Nickoloff, Kuenzli [27]), while two instances of PLCSim software were used as the PLC, with which the connection was obtained using the open-source software NetToPlcSim.

As a test, a layout shown in Figure 5 was built in the application. The system consists of three conveyors – the first one (1) is activated by a button on the HMI. The second conveyor (2) starts when a pallet runs over a sensor (3). Sensor (4) switches off the conveyor (1), and sensor (5) switches on the conveyor (6). Sensor (7) switches off the conveyors (2) and (6). The conveyor (1) and sensor (2) were connected to the first instance of the PLC simulator, while the other components

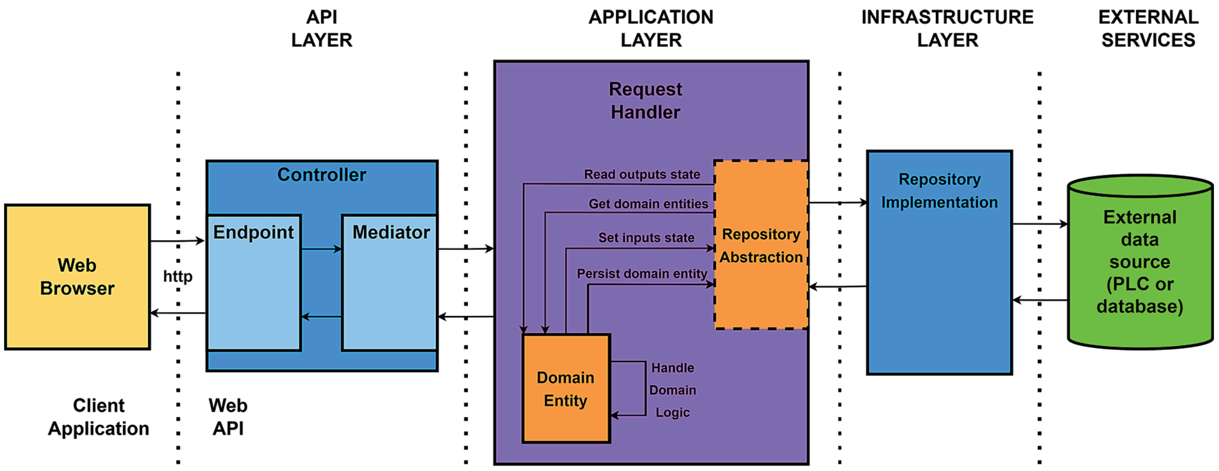

Fig. 4. Data flow diagram

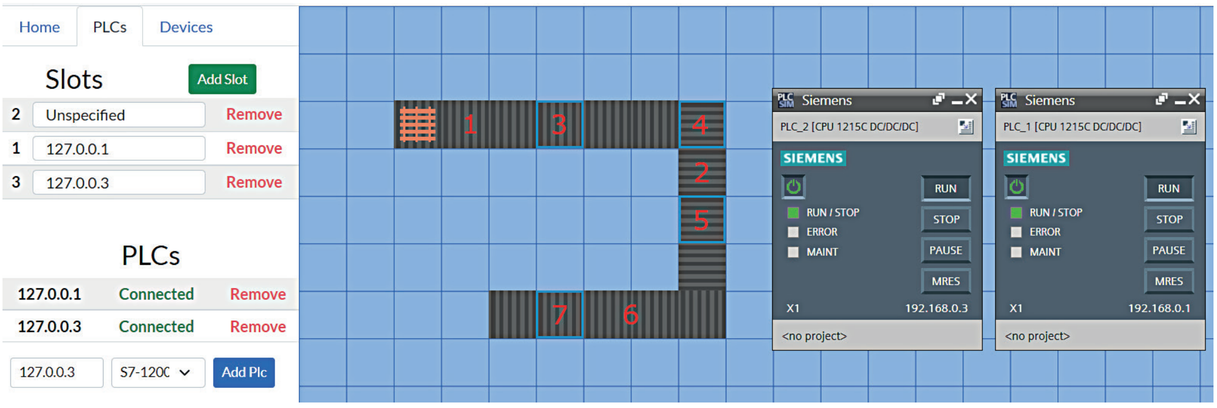

Fig. 5. Test system configuration: 1, 2, 6: virtual conveyors; 3, 4, 5, 7: virtual pallets sensors

were connected to the second instance. For the purpose of the experiment, we take the size of one field on a board as 0.5 m.

The experiment was to test the system's performance for different time intervals between HTTP requests. The time intervals chosen were 1, 0.5, 0.35, 0.2 and 0.1 seconds. The time it takes for the server to process the request and the duration of the entire request was measured. The time measurement results for several subsequent queries corresponding to the algorithm's execution are presented in Figure 6.

It was observed that the time of the HTTP data transfer itself was negligibly short and relatively constant compared to the query processing time (20ms maximum). Therefore the server processing times were used to perform the analysis. The experiment shows that an interval of one second can be taken as optimal for the correct operation of the application. The response from the API always occurs before the following query is executed (average query duration – about 750 ms). The application runs slowly but without any failures. Two times the shorter interval between queries results in slightly longer handling of the requests and their overlapping. Because answers to queries come in the order they are sent – from the user's perspective, the application continues to work correctly. It is the recommended way to run the application.

With the interval of 0.35 seconds, queries are sometimes processed even twice as long, and it also happens that the answers arrive in a different order than the queries were sent. Even shorter intervals, such as 0.2s or the interval considered by the authors to be the target interval, which is 0.1s, results

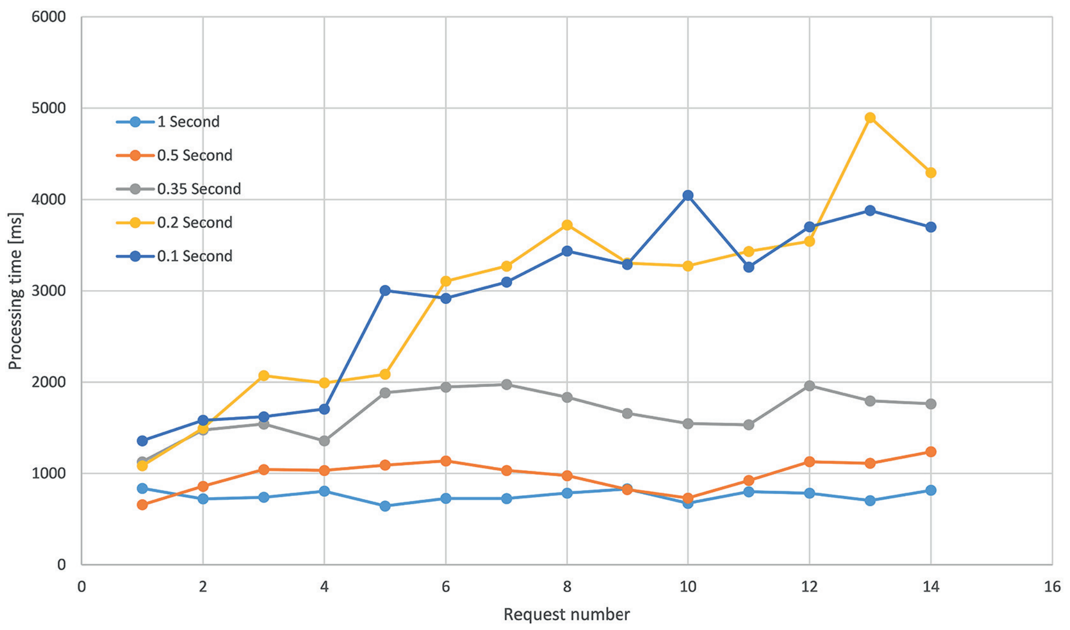

Fig. 6. Processing times for different time intervals between requests

in a drastic increase in processing time, and its high variability, so the order of responses received by the client is almost random. In both cases, this makes it impossible to use the application properly.

The experiment gives an overview of what the maximum speed of the simulated conveyor can be – taking the period of 500 ms as correct, and the maximum speed of the conveyor belt (transport by one field per query), the speed of two fields per second is obtained. The field size should be adopted considering the geometry of the simulated system. Primarily the distance through which the pallets are transported and the distance between the sensors. For the prepared system, we obtain a maximum conveyor belt speed of 1 m/s.

DISCUSSION

The performed experiment was crucial for recognizing the application's capabilities and taking a direction for its further development – thanks to the experiment; it was possible to estimate the class of problems for which the use of the application will be possible. It shows that the application is suitable at the moment only for slow-changing processes, due to query processing time. An example of a process with inter-operational transport in which the execution times are a few second each (Figure 7) lends itself very well to this. The next step in software development should be to locate Bottlenecks by measuring the execution time of specific parts of the application's code and attempting to optimize them.

Database accesses can often be a bottleneck. The software experiences a lot of database access requests. The plan is to reduce their occurrence and their time by optimizing indexes and queries and taking advantage of cache memory. Another issue worth considering is communication with the PLC – it is necessary to look at the way data is sent, make sure it is sent in an optimal way, limit the number of messages sent and retrieved, e.g. by sending whole blocks of data. Another technology worth looking at is the use of web sockets to use the optimal number of server-side threads for calculations. An additional advantage of this solution is that it allows simultaneous data transfer to several clients.

The potential use of the presented project as a support tool for training PLC programmers is envisaged. The main problem at the training stage is the lack of access to a physical object to test the implemented algorithm. The presented design can also be used to create a sketch that will be the basis for implementing a commercial digital twin or planning a new production line.

CONCLUSIONS

In the article, the authors introduced the idea of a digital twin, presented the fields in which it can be implemented, along with examples of specific implementations of it, and introduced the technology often used to create it. They also presented the proposed solution in the form of a web application, by describing the architecture and conducting an experiment.

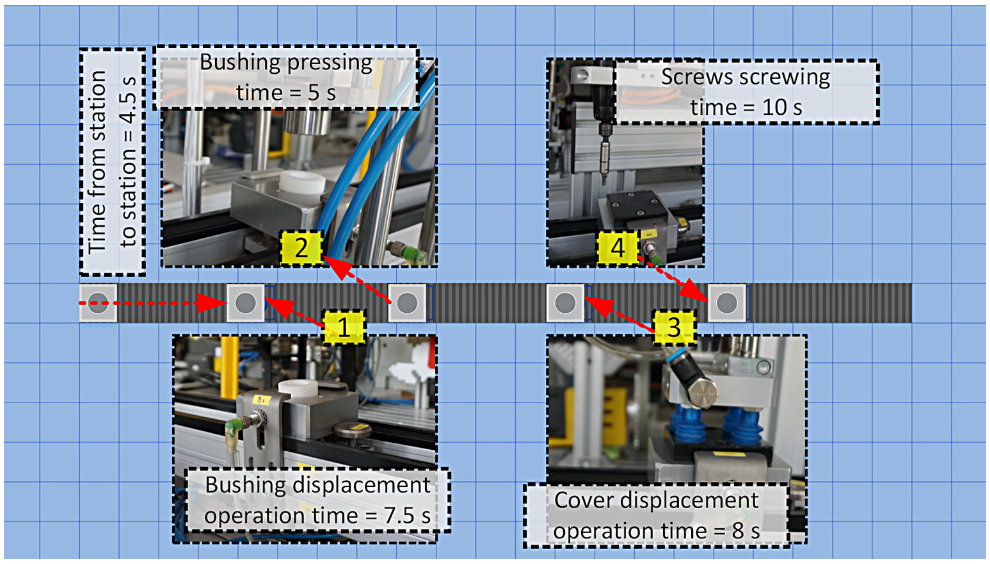
Time from station to station = 4.5 s
Bushing pressing time = 5 s
Screws screwing time = 10 s
2
4
1
3
Bushing displacement operation time = 7.5 s
Cover displacement operation time = 8 s

Fig. 7. Structure of the process model with an inter-operational transport conveyor

The current state of the project implements the following four basic types of devices, which can be used repeatedly in the project: a conveyor controlled by the controller's binary outputs, a position sensor that generates states for the controller's binary inputs, a pallet – a passive element moved on the conveyor, a machine that is a BlackBox for various devices performing operations on objects placed on pallets.

The conclusions of the article are summarized below. The application has been designed to facilitate its development. The focus was not only on implementing the assumed functionalities but also on creating an environment that allows the application to evolve in different directions, such as adding new industrial automation devices or the ability to support new PLC models. One project can contain multiple controllers or their simulators, which allows for the simulation of the operation of devices in an ICT network. Visualisation of the modelled systems is done as a top-down projection. At this stage, the application can work with popular S7 series controllers. The application is perfect for visualising simple algorithms based on devices that use digital inputs and outputs of the PLC. It can therefore be used analogously to applications such as Factory I/O or Famic Automation Studio. It might be beneficial for teaching the basics of PLC programming and even designing simple production lines. The main distinguishing feature of the application compared to selected control object simulators is the running of the application in a web environment. Calculations are performed on the server side, which must be placed on powerful hardware, while the client application can be run on much weaker hardware – even on mobile devices. Thanks to an abstract creation such as a controller slot, it is possible to swap the controlling driver even while the simulation is running. The demonstrator uses very simplified models, so one of the steps in the further development will be the implementation of the dynamics of the modelled objects. The results of the performed experiment indicate that the application needs to optimize the calculation execution time for a single timestamp to cope with more challenging processes. The standard control loop cycle time of many PLCs for industrial sequential systems is 100 ms, and this is the interval value between time steps to aim for. The conducted experiment shows that for the current solution the achievable time is 500 ms.

REFERENCES

1. Qin H., Wang H., Zhang Y., Lin L. Constructing Digital Twin for Smart Manufacturing, IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD). 2021; 638–642.
2. Wagg, D.J., Worden, K., Barthorpe, R.J., Gardner, P. Digital Twins: State-of-the-Art and Future Directions for Modeling and Simulation in Engineering Dynamics Applications. ASME. ASME J. Risk Uncertainty Part B. 2020; 6(3): 030901.
3. Kutin A.A., Bushuev, V.V., Molodtsov V.V. Digital twins of mechatronic machine tools for modern manufacturing. IOP Conference Series: Materials Science and Engineering. 2019; 568(1): 012070.
4. Cheng K., Wang Q., Yang D., Dai Q., Wang M. Digital-Twins-Driven Semi-Physical Simulation for Testing and Evaluation of Industrial Software in a Smart Manufacturing System. Machines. 2022; 10(5): 388.
5. Figueiras P., Lourenço L., Costa R., Graça D., Garcia G., Jardim-Gonçalves R. Big Data Provision for Digital Twins in Industry 4.0 Logistics Processes, 2021 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT). 2021; 516–521.
6. Maulshree S., Srivastava R., Fuenmayor E., Kuts V., Qiao J., Murray N., Devine D. Applications of Digital Twin across Industries: A Review, Applied Sciences. 2022; 12(11): 5727.
7. Lalik K., Flaga S. A real-time distance measurement system for a digital twin using mixed reality goggles. Sensors. 2021; 21(23): 7870.
8. Oborski P., Wysocki P. Intelligent Visual Quality Control System Based on Convolutional Neural Networks for Holonic Shop Floor Control of Industry 4.0 Manufacturing Systems. Advances in Science and Technology Research Journal. 2022; 16(2): 89–98.
9. Gaska P., Harmatys W., Gruza M., Gaska A., Sladek J. Simple Optical Coordinate Measuring System, Based on Fiducial Markers Detection, and its Accuracy Assessment. Advances in Science and Technology Research Journal. 2020; 14(4): 213–219.
10. Kuts V., Sarkans M., Otto T. Tähemaa T., Bondarenko Y. Digital Twin: Concept of Hybrid Programming for Industrial Robots – Use Case. Proceedings of the ASME 2019 International Mechanical Engineering Congress and Exposition. Volume 2B: Advanced Manufacturing. Salt Lake City, Utah, USA 2019.
11. Pizon, J., Gola, A., Swic, A. The Role and Meaning of the Digital Twin Technology in the Process of Implementing Intelligent Collaborative Robots.

In: Gapinski, B., Ciszak, O., Ivanov, V. (eds) Advances in Manufacturing III. MANUFACTURING 2022, Springer, 2022.

12. Staczek P., Pizon J., Danilczuk W., Gola A. A Digital Twin Approach for the Improvement of an Autonomous Mobile Robots (AMR's) Operating Environment – A Case Study, Sensors. 2021; 21(23): 7830.
13. Georgakopoulos D., Bamunuarachchi D. Digital Twins-based Application Development for Digital Manufacturing, 2021 IEEE 7th International Conference on Collaboration and Internet Computing (CIC). 2021, 87–95.
14. Yasin, A., Pang, T.Y., Cheng, C.T., Miletic, M. A Roadmap to Integrate Digital Twins for Small and Medium-Sized Enterprises. Appl. Sci. 2021; 11: 9479.
15. Herbus K., Ociepka P., Gwiazda A. Virtual activating of a robotised production cell with use of the mechatronics concept designer module of the PLM Siemens NX system. In International Conference on Intelligent Systems in Production Engineering and Maintenancel, Springer, Cham. 2018; 417–425.
16. Febronio R., Martinez G., Puga J.A.V., Coronado P.D.U., Ortigoza A.A.G., Orta-Castañon P., Ahuett-Garza H. A flexible and open environment for discrete event simulations and smart manufacturing. International Journal on Interactive Design and Manufacturing. 2021; 15(4): 509.
17. Pizon J., Gola A. The Meaning and Directions of Development of Personalized Production in the Era of Industry 4.0 and Industry 5.0. In:, et al. Innovations in Industrial Engineering II. Icieng, Lecture Notes in Mechanical Engineering,. Springer, Cham, 2022.
18. Thapa D., Park C.M., Han K.H., Park S.C., Wang G. Architecture for modeling, simulation, and execution of PLC based manufacturing system, Winter Simulation Conference, 2008, 1794–1801.
19. Ruzarovsky R., Holubek R., Delgado Sobrino D.R., Janíček M. The Simulation of Conveyor Control System Using the Virtual Commissioning and Virtual Reality. Advances in Science and Technology Research Journal. 2018; 12(4): 164–171.
20. Riera B., Vigário B. HOME I/O and FACTORY I/O: a virtual house and a virtual plant for control education. IFAC-PapersOnLine. 2017; 50(1): 9144–9149.
21. Lalik K., Kozek M., Dominik I., Łukasiewicz P. Adaptive MRAC Controller in the Effector Trajectory Generator for Industry 4.0 Machines, Advanced, Contemporary Control, Advances in Intelligent Systems and Computing, Springer, 2020.
22. Martin R.C. Clean architecture: a craftsman's guide to software structure and design. Prentice-Hall, 2017.
23. Evans E., Evans E.J. Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional, 2004.
24. Gamma E., Helm R., Johnson R., Vlissides J. Design patterns: Abstraction and reuse of object-oriented design. European Conference on Object-Oriented Programming. Springer, Berlin, Heidelberg. 1993; 273–282.
25. Huda M., Arya Y.D.S., Khan M.H. Quantifying reusability of object-oriented design: a testability perspective. Journal of Software Engineering and Applications. 2015; 8(4): 175.
26. Bruel J.M., Mazzara M., Meyer B. Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment: First International Workshop, Chateau de Villebrumier, France 2018.
27. Nickoloff J., Kuenzli S. Docker in action, Second Edition, Simon and Schuster, 2019.