Journal of Machine Engineering, 2022, Vol. 22, No. 2, 57–67
ISSN 1895-7595 (Print) ISSN 2391-8071 (Online)

Received: *29 January. 2022* / Accepted: *06 April 2022* / Published online: *18 April 2022*

digital twins, virtual reality,
robot operating system,
collaborative robots

Danyil DIACHENKO[1], Andriy PARTYSHEV[1],
Simone Luca PIZZAGALLI[1*], Yevhen BONDARENKO[1],
Tauno OTTO[1], Vladimir KUTS[3,2,1]

INDUSTRIAL COLLABORATIVE ROBOT DIGITAL TWIN INTEGRATION AND CONTROL USING ROBOT OPERATING SYSTEM

Standardized and universal solutions for industrial robot integration are increasingly urgent requirements for companies looking for machine interconnectivity, and flexibility in creating tailor made manufacturing systems. These solutions must be supported by modular and open-source components able to easily integrate new control methods and advanced Extended Reality (XR) interfaces. Robot Operating System (ROS) has proven to be a reliable standard for industrial robot integration. ROS compatibility software is provided by many producers and allows for the implementation of modular control units by unifying development practices along the same libraries and methods. Digital Twins (DT) of industrial equipment and processes offer a solid base to develop innovative digital tools relying on synchronization between physical and digital entities and to easily setup intuitive XR interfaces for teleoperation and programming. This work presents the integration of the OMRON TM5-9000 collaborative industrial robot into the IVAR laboratory DT system at Tallinn University of Technology. By using Unity3D game engine and developing a ROS package for the specific machine, the digital model of the collaborative robot is integrated into the existing twin. Synchronization with the real counterpart is provided by MQTT protocol while a robot user interface is developed in Unity and provides robot joints visualization and remote control.

1. INTRODUCTION

Collaborative robots or “cobots” are bridging the divide between human based and automated working environments, shortening the distance between operators and machines on the production floor. They are designed to allow and account for different assignments, sharing the workspace with humans and accomplishing tasks simultaneously, jointly, and possibly interchangeably with the operators.

[1] Department of Mechanical and Industrial Engineering, Tallinn University of Technology, Estonia
[2] Department of Electronics and Computer Engineering., Technological University of the Shannon: Midlands Midwest, Ireland
[3] Electronic & Computer Engineering department, University of Limerick, Ireland
[*] E-mail: simone.pizzagalli@taltech.ee
https://doi. org/10.36897/jme/148110

The implementation of Industry 4.0 (I4.0) technologies and the rapid evolution of new production models driven by flexibility and rapid adjustments to the market [1], is reflected by the interest and robots [2] towards the Operator 4.0 paradigm [3]. As the market diversifies and the number of companies developing cobots increases, the usage of Robot Operating System (ROS) [4] as the cross-platform software for robot development grows, while more leading manufacturers provide ROS drivers and integration to the new products (e.g. Universal Robots, ABB or Techman Robot) [5]. ROS has many advantages by providing high level functionalities, easy integration of different machines and sensors, and the benefits of an operating system with reusable services and development tools. Integration, reconfiguration, and interoperability become easier to handle tasks under these premises, facilitating the development of smart and highly dynamic industrial collaborative systems. Furthermore, these characteristics lead to an increased research interest for implementation of applications specifically designed in ROS and taking advantage of its possibilities. The study presented in [6], for instance, describes a new method of time synchronization for modular collaborative robots using ROS. The research presented in [7] focalizes on a ROS based integration API for the KUKA iiwa lightweight robot facilitating the development of new collaborative processes and integration with other machines. The study by Kallweit et al. [8] uses ROS to integrate a novel collision detection safety system to an industrial robotic arm. The robot operating system is used for sensor integration and prediction of the worker behavior, collision detection and robot control and path planning tasks. The study in [9] proposes a ROS based integration for electric propulsion drive Digital Twin (DT) synchronization with the real-world counterpart and aimed at autonomous driving vehicles. The system grants faster data collection and exchange from the real to the simulated testbench allowing for optimization of the propulsion system with improvements in energy consumption and path planning of the vehicle itself. Baklouti et al. [10] present an experimental work aimed at optimizing the control and teleoperation of the Motoman robots adopting the standard open-source ROS based drivers. The study proposes a new velocity control mode for the ROS driver to improve response time, delay, and motion quality. The study in [11] presents a ROS based system integrating depth sensor camera-based image recognition for a simple pick and place task using RB5 robotic arm. The work demonstrates the flexibility of ROS in integrating sensors and machines for fast collaborative production system setups and experimentation. Thanks to the system presented in [12] the authors are able to simulate a robot-based monitoring task in Unity. The monitoring and welding processes, which are performed by two different robots, are synchronized through a ROS – Unity node which operates as a main bridge between the different machines. A clear advantage of the solution is that the system is hardware agnostic as any ROS compatible machine would be able to perform the monitoring task.

The emergence of the Industry 5.0 concepts with collaborative technologies and advanced DT interface based solution for the cooperative robots in new workspaces [13], makes cobots one of the sectors that could benefit from ROS the most. This can further contribute to the industry and the developing of hybrid workspaces where multiple workers operate alongside several cobots and other machines, all in need for integration and synchronization. DT of industrial systems are already employed for testing and commissioning advanced production lines allocating human and robots on simultaneous and joint tasks

[14]. These highly dynamic operations may also frequently require quick changes and testing to determine what configuration provides the best increase of productivity between the humans and the robot. Adopting ROS in combination with DT solutions allows to optimize setup time and resources while providing easy integration for different machines [15] and safe test bench for workspace configuration. Examples can be found in the works by Bilberg et al. [16] and Perez et al. [17] were both monitor based simulations and immersive virtual reality (VR) environments are used to design, control, deploy and monitor collaborative production processes. A mixed reality (MR) solution is proposed in [18] were Unity and ROS are adopted in the implementation of an application supporting design and evaluation of different Human Robot Collaboration (HRC) environments [19]. Similarly, the research presented in [20] proposes a ROS based module to manage different machines, robots and sensors aimed at the design and reconfiguration of HRC assembly lines. The configuration process is supported by the dynamically synchronized digital counterparts of the real systems. Implementation of HRC methods with support from ROS allows for a flexible, modular and accessible systems, that does not compromise the inherent safety features of the collaborative robots (e.g., force sensors and speed limitations). Combining ROS and DT solutions is also filling the niche for a system that can enable rapid programming and development of safe collaborative practices, without the need for extensive training and potentially even a unified safety development space for multiple instances of potentially different robots.

In this study we present the integration of the OMRON TM5-9000 collaborative industrial robot using ROS and Unity with the existing integrated DT interface of the IVAR laboratory DT system at Tallinn University of Technology lab at allowing for collaboration assessment and reconfiguration of the industrial systems present in the lab. This work benefits from both the above-mentioned advantages of ROS (modularity, interoperability, universality) and the use of Unity in creating advanced interfaces for testing collaboration flows, safety methods, training operators and evaluating user performance and health state. The proposed architecture and interface provide an easy teleoperation and visualization method for the existing robot, while being the base for the integration of the user in the DT loop by means of extended reality (XR) technologies through Unity.

2. SYSTEM ARCHITECTURE

The presented solution makes use of the OMRON TM5-900 which is a multi-joint cooperative robot produced by Techman Robot [21]. This robot has 6 joints that define its possible movements and provide hardware limitations to its trajectories within the operational workspace. Figure 1 shows the robot joints and dimensions specifications. The position of the end-effector of the robot changes based on the local angles assumed by each joint and is essential for planning and execution of movements done with digital reproductions of the robot. The system natively utilizes TMflow, a Techman Robot software which is intended for programming, and allows the user to add/remove nodes, to implement logic, variables, motion types and actions which form a list of pre-defined programs/options to be used with the OMRON TM5-900. The software bears some integration features with other products designed by Techman Robot and most importantly allows for ROS set-up through a “Listen”

node. Before proceeding with the integration of the robot and connection to Unity3D a modification is made in the provided TM ROS Driver. Accordingly, a listening task is set up for the robot in TMflow by opening a listen node granting data reception.

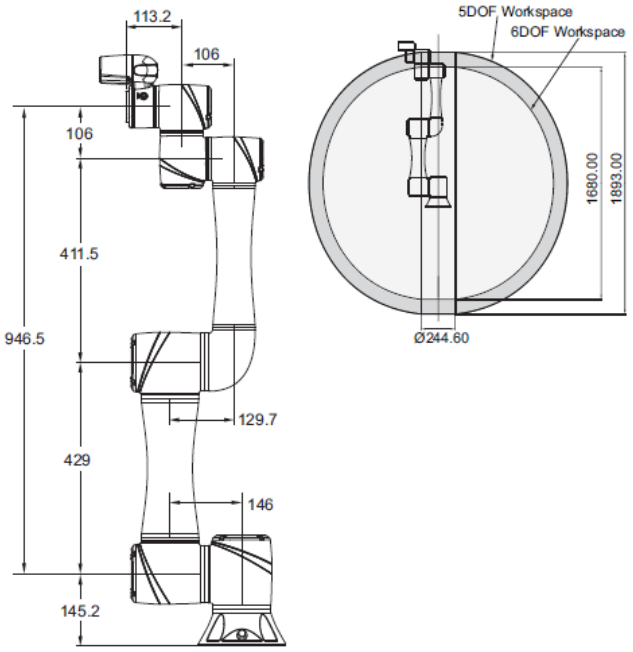

Fig. 1. OMRON TM5-900 joints and workspace dimensions

The TM ROS Driver is implemented in a dedicated Linux Virtual Machine which connects to the Listen node in TMflow via an Ethernet Slave at the driver startup. The architecture of the system is shown in Fig. 2.

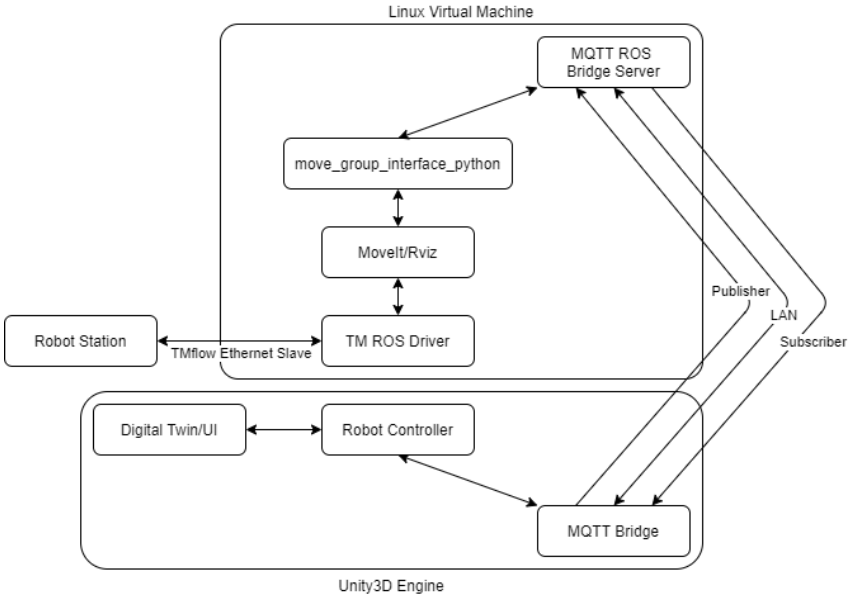

Fig. 2. Proposed architecture of the system

The ROS package created in the virtual machine transfers the received real robot parameters through the TM ROS Driver to Unity3D engine. These two are connected through an MQTT LAN Bridge modified to suit the Omron robot and based on the work presented in [22]. The MQTT ROS bridge server publishes the state and new positions of the real robot joints. The ROS Bridge Client on the Unity3D side is used by the robot controller. This is developed to control and update the DT robot joint positions based on the MQTT message subscription and publishing, and as a base for the User Interface (UI) command issuing and controls. After comparing different solutions MoveIt [23] was selected as a path planning software for this system. The following paragraphs explain more in detail the implementation of the software and DT control in Unity.

3. IMPLEMENTATION

The following paragraph will exemplify the implementation of the basic components of the proposed system including the choice of path planning software and methods, the ROS Unity communication, the creation of the robot DT, controller and UI in Unity3D.

3.1. PATH PLANNING

While Techman Robot does provide a TM ROS Driver that connects their TM Flow to ROS, the Driver has only limited pre-setups and few developed options. During the initial stage of development different solutions were evaluated including Gazebo, ROS industrial, and TM ROS Driver. MoveIt was chosen as the final approach as it allows efficient path planning, is part of the ROS ecosystem, makes use of URDF robot models provided by the TM ROS Driver including their visualization through Rviz, and easily allows to process data communication with the Unity3D controller. The *move_group_python_interface script* is created in python and aimed at path planning and execution. This script has several methods dedicated to receiving and processing the information exchanged from the MQTT bridge, storing them and calculating new trajectories. These methods are summarized in Table 1.

Table 1. Move_group_python_interface script methods

METHOD	FUNCTION
Joint_callback	Callback method that assigns received subscriber data to script position values.
all_close	Method for testing if a list of values is within a tolerance of their counterparts in another list.
__init__	Basic method which initialises MoveIt and assigns groups and interfaces
go_to_joint_state	Main method used for trajectory planning and execution; It uses the values received through the MQTT Bridge to initiate a new movement for the robot
go_to_pose_goal, *plan_cartesian_path*, *execute_plan*	MoveIt configuration supports movement planning based on end-effector pose and Cartesian path planning. These functions are not used in the scope of this work.

The script integrates a queueing capability to counterbalance the multi-axial movement deficiency of MoveIt and allowing for up to 10 movement commands to be temporarily stored. This allows each command to start after the previous goal state has been reached when the user tries to execute several publishes one after the other.

3.2. MQTT ROS-UNITY COMMUNICATION

The communication between ROS and Unity3D is provided by MQTT bridge published/subscriber scripts. The *move_group_python_interface* serves the *OmronMove* message type created to fit the specific robot requirement for transferring data referring to each joint angle. The message is sent through LAN Server to Unity3D. The two communication topics involved are:

- */omron/joints/feedback* – ROS topic sending the constructed *OmronMove* message consisting of robot current positions to Unity3D
- *omron/joints/command* – ROS topic receiving an *OmronMove* message from Unity3D consisting of the goal angles chosen by the user and to be transferred to the path calculation and real robot movement

The provided communication method can be easily modified and extended to transfer additional variables and data for other types of robots. From the Unity3D side there are two scripts written in C# managing the publication and subscription of the robot joints positions. The first, *MqttOmronMovePublisher*, packages the input joint angle values into the *OmronMove* message type and publishes them on the *omron/command* MQTT topic which is received by */omron/joints/command* at the ROS side after the conversion and JSON deserialization. The second script, *MqttOmronMoveSubscriber*, receives the *OmronMove* message type which is then passed onto each joint of the digital robot model and used by the DT controller script to reposition the robot joints.

3.3. UNITY INTEGRATION

Unity3D game engine is employed in the implementation of the robot DT, controller and UI. Unity3D is a powerful tool supporting a wide range of hardware and software development kits which enable application developments for a variety of operating systems. The first step of the DT implementation is the integration of the OMRON TM5-900 3D model in a new Unity project. One of the main requirements for the Unity robot DT integration is the correspondence and fidelity to the real machine geometry ensuring no differences existing between real and digital counterpart. The mesh and URDF definitions for the models are provided with TM ROS Driver and already employed for MoveIt-Rviz path planning and visualization. Unity3D does not support URDF files import. For this reason, we made use of the Unity3D-URDF import plugin provided by Unity Technologies. This plugin is based on ROS#, which is a ROS to Unity integration project written in C# programming language and widely employed in similar scenarios. Through the adopted plugin the model is imported in Unity along with the correct axis reference system. After importing, the OMRON TM5-

900 3D model went through an optimization process including mesh simplification and hierarchy structuring. The final robot DT is constituted by a nested tree structure of Unity *GameObjects*, with each subsequent joint being a child object to the previous one starting with the base of the robot and ending with the end-effector. Each joint carries the relevant scripts necessary for controlling and rotating the joints angles. The resulting DT robot is shown in Fig. 3.

A second robot model, *'ghost'*, was created in the scene, and differentiated from the main DT by applying a different texture material. While the purpose of the main 3D model is to represent the angle/joint values of the real OMRON TM5-900 robot and serve as the Digital Twin, the purpose of the *ghost* robot is to preview the robot position and joint angles when the user interacts with the UI and provide visual feedback to user interactions with the control application.

Fig. 3. OMRON TM5-900 DT in Unity3D

3.4. ROBOT CONTROLLER AND USER INTERFACE

The robot controller is constituted by a collection of scripts written in C# programming language and employed to manipulate both the DT and *ghost* robot in Unity. The controller accomplishes two main functions: it achieves the implementation of DT as a concept, and it provides a visual representation to the goal state of the robot by means of the *ghost* preview method, Fig. 4.

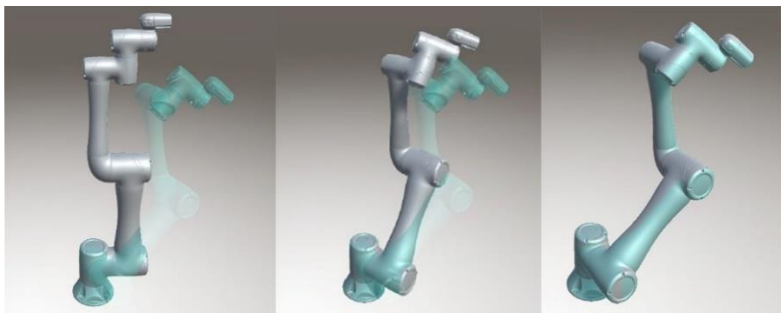

Fig. 4. DT and *ghost* robot when the position is published (left), midway (center), and having reached the goal state (right)

The *OmronMovePublisherWrapper* checks for the UI sliders inputs and transfers the values to the robot joint angles, Fig. 5. Additionally, it uses the *Mqtt-MovePublisher* to publish the values to the topic and initiating the physical robot movement.

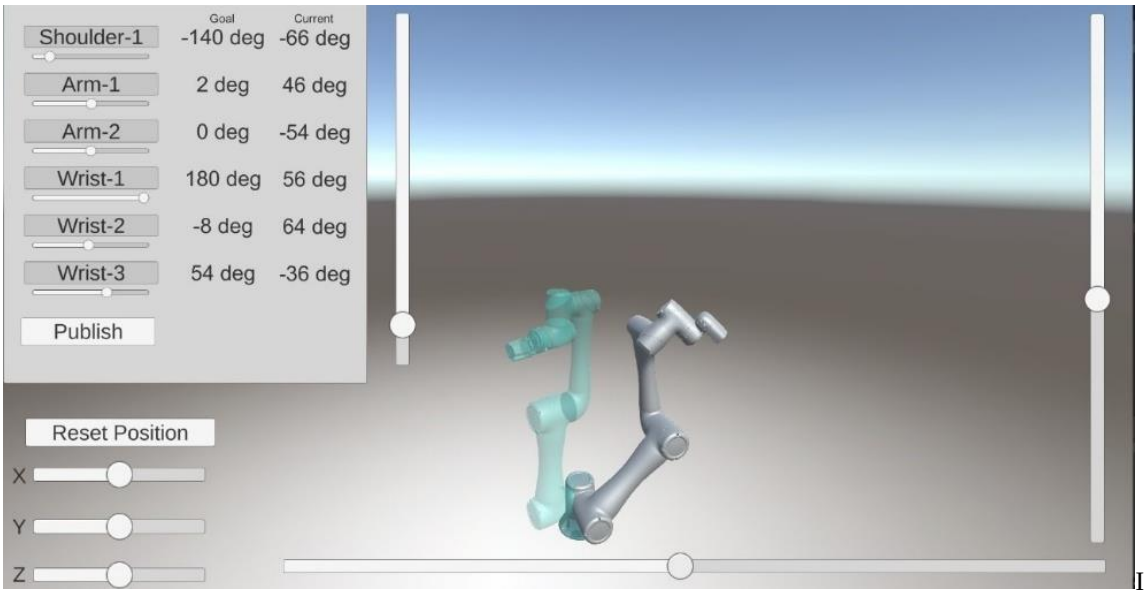

Fig. 5. OMRON TM5-900 DT remote control U

The *JointRobotROSPreviewer* script controls the preview state of each new position by sending the rotational angle value to the assigned *ghost* robot joints by means of the *ArmedRobotJoint.* The angle input is taken from the values passed by the *MqttOmron-MovePublisher*. The *JointRobotROSStateUpdater* is equivalent to the *JointRobotROSP-reviewer* for the DT robot.

This script passes the values received from *MqttOmronMoveSubscriber*, holding the joint angles of the real robot, to the assigned DT joints whom position is updated through the *ArmedRobotJoint* script. This last script is attached to each of the joints the DT robot and the *ghost* robot. The code gets the angles values from either *JointRobotROSStateUpdater* or *JointRobotROSPreviewer* and locally rotates the joint to the correct position along the selected axis. Furthermore, the script defines rotational limits for each joint that can be modified from the Unity inspector component. The monitor based remote control UI is also developed in Unity3D and provides several visualization and control functions for the DT robot and consequently its real counterpart. The interface provides the user with full camera view over the DT and *ghost* robot and the possibility of adjusting the camera position, rotation and zoom by means of user interface sliders overlayed on top of the scene view. These controls include a *Reset Position* button used to reposition the camera in its original location after being modified. The robot control interface displays the six different joint names (shoulder-1, arm-1, arm-2, wrist-1, wrist-2, wrist-3) and values for the goal position of each joint and current robot joint position, (in degrees). The values are updated dynamically by means of the UI scripts *SliderValueShow* and *CurrentPOSUI.* Six sliders are controlling the desired joint rotation angle goal position whom values are updating the *ghost* robot dynamically. Finally,

the *Publish* button sends the selected goal state values to ROS for the update of the real robot joint positions.

4. CONCLUSION AND FUTURE WORKS

The developed solution exemplifies the capabilities of ROS-Unity architecture by successfully providing a DT interface for OMRON TM5-900 and an architecture that would allow for future easy expansion of UI components and connected machines. The system provides a simple UI for remote operation and visualization of the robot. The proposed software consists of a desktop-based application and UI developed in Unity3D. The ROS package and custom-made logic are installed on the same local network of the real robot, used to execute the commands received from the UI and for the synchronization of real and DT robotic arm positions by means of MQTT communication protocol. MoveIt is used for path planning and calculation. The presented DT UI system for the OMRON TM5-900 achieves a high level of modularity and shows the potential of ROS in providing a significant degree of flexibility in integrating different automated robotic systems. The solution responds to the increasing needs of the modern production lines resulting in dynamically changing scenarios and demanding for universal software tools and methods facilitating the exploitation of human machine collaboration, limiting the costs, and optimizing the production processes.

Future developments will include the optimization of more advanced interface functions, controls, and interaction methods. Different UI controls for joint angle target position should be taken into consideration as much as the possibility of controlling and visualizing the robot in Cartesian space. The assessment of the proposed UI against the real robot control and programming methods, by evaluating their usability and design features, is another important aspect which will include the estimation of hardware-based delay and angular errors on the specific machine manipulation. The possibility of integrating remote UI controls for different end effectors should be taken into consideration and integrated as selectable list of tools in the main system interface. At last, both the planning limitation of the MoveIt libraries and impossibility of accessing the OMRON TM5-900 security force sensor parameters might be addressed to improve usability, error notification, safety, and control over the joint's movements.

The main future goal will be integrating the proposed architecture in the mobile based dashboard and full immersive Virtual Reality (VR) DT interfaces already implemented in the IVAR laboratory DT system at Tallinn University of Technology. The integration of the OMRON robot DT into a highly interactable XR environment would allow a fully immersive or augmented interface experience for different machines. The operator will be able to interact with virtual, augmented and real components and equipment both locally and remotely. The system will provide a base for advanced interaction methods, gesture or voice control, exploiting the easy XR hardware integration of Unity3D. Furthermore, by putting the user at the center of the DT loop and within the user interface through sensor integration and VR interaction and visualization tools, the system will provide a base to implement advanced I5.0 monitoring, collaboration method evaluation, and assessment of the operator health and performance in the production system.

ACKNOWLEDGEMENTS

The research was conducted using the Smart Industry Centre (SmartIC) core facility funded by the Estonian Research Council grant TT2. In addition, Vladimir Kuts has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 847577; and a research grant from Science Foundation Ireland (SFI) under Grant Number 16/RC/3918 (Ireland's European Structural and Investment Funds Programmes and the European Regional Development Fund 2014-2.

REFERENCES

[1] BASTIDAS-CRUZ A., HEIMANN O., HANINGER K., KRÜGER J., 2020, *Information Requirements and Interfaces for the Programming of Robots in Flexible Manufacturing*, Annals of Scientific Society for Assembly, Handling and Industrial Robotics, 183–192.
[2] SHERWANI F., ASAD M.M., IBRAHIM B.S.K.K., 2020, *Collaborative Robots and Industrial Revolution 4.0 (IR 4.0)*, International Conference on Emerging Trends in Smart Technologies, ICETST 2020.
[3] ROMERO D., BERNUS P., NORAN O., STAHRE J., FAST-BERGLUND A., 2016, *The Operator 4.0: Human Cyber-Physical Systems & Adaptive Automation Towards Human-Automation Symbiosis Work Systems*, IFIP Advances in Information and Communication Technology, 677–686.
[4] QUIGLEY M., CONLEY K., GERKEY B., FAUST J., FOOTE T., LEIBS J., WHEELER R., NG A.Y., 2009, *ROS: an Open-Source Robot Operating System*, ICRA Workshop on Open Source Software, 3, 5.
[5] https://robots.ros.org/.
[6] GUTIERREZ C.S.V., JUAN L.U.S., UGARTE I.Z., GOENAGA I.M., KIRSCHGENS L.A., VILCHES V.M., 2018, *Time Synchronization in Modular Collaborative Robots*, rXiv:1809.07295.
[7] MOKARAM S., AITKEN J.M., MARTINEZ-HERNANDEZ U., EIMONTAITE I., CAMERON D., ROLPH J., GWILT I., MCAREE O., LAW J., 2017, *A ROS-Integrated API for the KUKA LBR iiwa Collaborative Robot*, IFAC-PapersOnLine, 50, 15859–64.
[8] KALLWEIT S., WALENTA R., GOTTSCHALK M., 2016, *ROS Based Safety Concept for Collaborative Robots in Industrial Applications*, Advances in Intelligent Systems and Computing, 27–35.
[9] KUTS V., RASSOLKIN A., PARTYSHEV A., JEGOROV S., RJABTSIKOV V., 2021, *ROS Middle-Layer Integration to Unity 3D as an Interface Option for Propulsion Drive Simulations of Autonomous Vehicles* IOP Conference Series: Materials Science and Engineering, 1140 012008.
[10] BAKLOUTI S., GALLOT G., VIAUD J., SUBRIN K., 2021, *On the Improvement of Ros-Based Control for Teleoperated Yaskawa Robots*, Applied Sciences (Switzerland), 11.
[11] KREITZ J., LEE M., SUBPARK H., OH P.Y., OH J.H., 2020, *Implementing ROS Communications for Sensor Integration with the RB5 Collaborative Robot*, 2020 10th Annual Computing and Communication Workshop and Conference, 378–383.
[12] SITA E., HORVATH C.M., THOMESSEN T., KORONDI P., PIPE A.G., 2018, *ROS-Unity3D Based System for Monitoring of an Industrial Robotic Process*, 2017 IEEE/SICE International Symposium on System Integration, 1047–1052.
[13] MADDIKUNTA P.K.R., PHAM Q-V., PRABADEVI B., DEEPA N., DEV K., GADEKALLU T.R, RUBY R., LIYANAGE M., 2021, *Industry 5.0: A Survey on Enabling Technologies and Potential Applications*, Journal of Industrial Information Integration, 100257.
[14] KUTS V., CHEREZOVA N., SARKANS M., OTTO T., 2020, *Digital Twin: Industrial Robot Kinematic Model Integration to the Virtual Reality Environment*, Journal of Machine Engineering, 20/2, 53–64.
[15] KUTS V., OTTO T., BONDARENKO Y., YU F., 2020, *Digital Twin: Collaborative Virtual Reality Environment for Multi-Purpose Industrial Applications*, ASME International Mechanical Engineering Congress and Exposition, Proceedings, IMECE2020-23390, V02BT02A010.
[16] BILBERG A., MALIK A.A., 2019, *Digital Twin Driven Human–Robot Collaborative Assembly*, CIRP Annals, 68/1, 499–502.
[17] PEREZ L., RODRIGUEZ-JIMENEZ S., RODRIGUEZ N., USAMENTIAGA R., GARCIA D.F., 2020, *Digital Twin and Virtual Reality Based Methodology for Multi-Robot Manufacturing Cell Commissioning*, Applied Sciences, 10, 3633.
[18] SIEGELE D., STEINER D., GIUSTI A., RIEDL M., MATT D.T., 2021, *Optimizing Collaborative Robotic Workspaces in Industry by Applying Mixed Reality*, International Conference on Augmented Reality, Virtual Reality and Computer Graphics, 544–559.

[19] MARVEL J.A., BAGCHI S., ZIMMERMAN M., AKSU M., ANTONISHEK B., LI X., WANG Y., MEAD R., FONG T., BEN AMOR H., 2021, *Novel and Emerging Test Methods and Metrics for Effective HRI*, ACM/IEEE International Conference on Human-Robot Interaction, 730–732.
[20] KOUSI N., GKOURNELOS C., AIVALIOTIS S., LOTSARIS K., BAVELOS A.C., BARIS P., MICHALOS G., MAKRIS S., 2021, *Digital Twin for Designing and Reconfiguring Human–Robot Collaborative Assembly Lines*, Applied Sciences (Switzerland), 11.
[21] https://www.tm-robot.com/en/techman-x-omron/.
[22] KUTS V., MODONI G.E., OTTO T., SACCO M., TÄHEMAA T., BONDARENKO Y., WANG R., 2019, *Synchronizing Physical Factory and its Digital Twin Throughan iiot Middleware: A case study*, Proceedings of the Estonian Academy of Sciences, 68, 364–370.
[23] https://moveit.ros.org/.