Proceedings of the Estonian Academy of Sciences,
2021, **70**, 4, 392–398
https://doi.org/10.3176/proc.2021.4.04
Available online at www.eap.ee/proceedings

MANUFACTURING
ENGINEERING

ROS middle-layer integration into Unity3D as an interface option for propulsion drive simulations of autonomous vehicles

Vladimir Kuts[a,b*], Anton Rassõlkin[c], Sergei Jegorov[a] and Viktor Rjabtšikov[c]

[a] Department of Mechanical and Industrial Engineering, School of Engineering, Tallinn University of Technology, Ehitajate tee 5, 19086 Tallinn, Estonia
[b] Electronic and Computer Engineering Department, University of Limerick, Limerick, V94 T9PX, Ireland
[c] Department of Electrical Power Engineering and Mechatronics, School of Engineering, Tallinn University of Technology, Ehitajate tee 5, 19086 Tallinn, Estonia

Received 17 June 2021, accepted 19 July 2021, available online 1 November 2021

Abstract. As autonomous vehicle development continues at growing speeds, so does the need to optimize, diagnose, and test various elements of autonomous systems under different conditions. Since such processes should be carried out in parallel, it may result in bottlenecks in development and increased complexity. The trend for Digital Twins offers a promising option for the diagnosis and testing to be carried out separately from the physical devices, incl. autonomous vehicles in the virtual world. The idea of intercommunication between virtual and physical twins provides possibilities to estimate risks, drawbacks, physical damages to the vehicle's drive systems, and the physical vehicle's critical conditions. Although providing communications between these systems arises at the speed that will be adequate to represent the physical vehicle in the virtual world correctly, it is still a trendy topic. This paper aims to demonstrate the enhancement of communications by using the Robot Operating System (ROS) as a middleware interface between two twinning systems by the example of the autonomous vehicle's propulsion drive. Data gathered from the physical and virtual worlds can be exchanged in the middle to allow for continuous training and optimization of the propulsion drive model, which would lead to more efficient path planning and energy-efficient drive of the autonomous vehicle itself. Additionally, a comparative analysis of ROS and its next version ROS2 is provided, discussing their differences and outlining drawbacks.

Key words: ROS, autonomous vehicles, propulsion drives, digital twins, simulations.

1. INTRODUCTION

Simulation is an approximate or 1:1 imitation of an actual process, often taking part in the virtual environment, troubleshooting, researching, testing, training, monitoring, controlling, or educating. In the past decade, simulations have been vital in production and development as they are capable of preventing many problems related to planning and reducing bottlenecks at early stages, also during the real-time maintenance of the process [1–5]. And considering especially the increasing complexity of technology and the rise in using fully autonomous systems, simulations help to enforce and change features related to work safety. One of the simulation aspects – the concept of the Digital Twin (DT) [6,7] – is exploited in this research to develop a precise dual-way synchronized simulation interface for the propulsion drives [8,9] to be ready to be integrated into the electrical vehicles [10].

Physics simulations are very common and critical nowadays. They are used enormously in such applications as MATLAB Simulink, Simscape, CAD design, SolidWorks, etc., and in simulations of the physical processes in gamified environments. They should be considered in the planning stage of mechatronic systems [11].

* Corresponding author, vladimir.kuts@taltech.ee

Of course, all physics simulations have approximation and simplifications, since not all possible physical laws can be simulated simultaneously as yet; however, such simulations provide considerable benefits in research and testing.

In a previous study that we conducted on the electrical motors simulation in which the DT for a propulsion drive of an autonomous electric vehicle was developed [9], Unity3D was used for simulations of the DT that was exchanging messages with the Robot Operating System (ROS) node through a ROS bridge [12]. However, ROS is not only used for robots but also for various drones, self-driving vehicles, and autonomous systems. ROS enables inter-process communication; it is believed to be a quality method of interconnecting a digital twin propulsion drive system with its real counterpart. ROS was used for performance calculation applying an empirical performance model for the induction motor (IM). In this research Unity3D is used as a visualization tool, which is connected with ROS directly [13]. Even though Unity3D simulated most of the motor's physical behaviour (torque and rotation), the response and the received numerical values, unfortunately, do not suit the DT development in the long run. The reason for this is the complexity of the overall system of physics of the IM. Moreover, to make the system transferable and usable with other models (not the ones present in Unity3D but also in Gazebo or elsewhere), the physics handling must be close to standalone.

The main aim of the current research was to develop a framework and a toolkit, including a middle-layer ROS interface, connected with the physical propulsion drive workbench and its DT, which can be visualized in various simulation engines. The related paper aims to develop a methodology to connect the interface with Unity3D for the visualization, considering data exchange and feedback.

2. METHODOLOGY

2.1. Working principle of a test bench on a digital twin

The physical component of the proposed DT is being developed as a hybrid of both data-driven and process-driven modelling principles. Although most of the parameters describing the physical behaviour of a propulsion drive can be calculated using physical equations, some values can only be read from the real electrical drive. An example of such ROS node structure used in the presented case can be seen in Fig. 1, where the input current from the frequency converter to the IM was recorded, simulated in ROS, and used to calculate other ROS parameters. The infrastructure features the nodes calculating torque, power, and the efficiency of the motor as well as a node that checks motor windings for malfunctions.

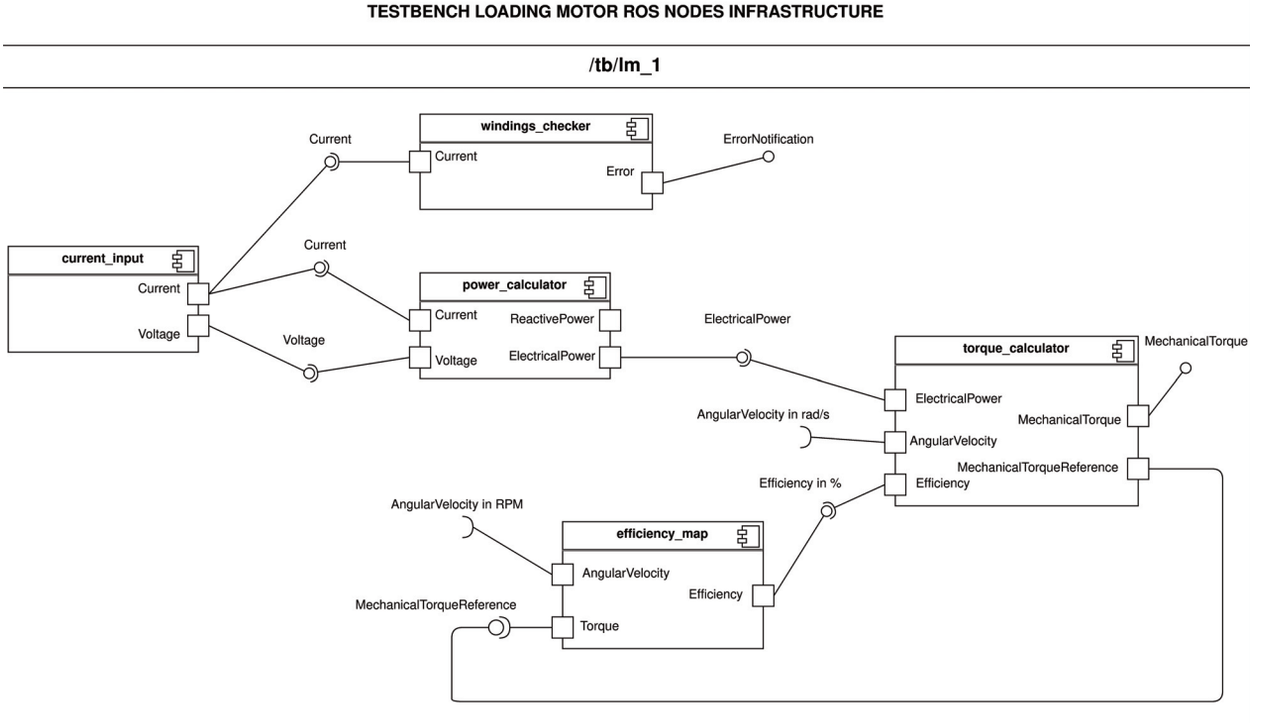

Fig. 1. Input current measurements sampled at 5 kHz frequency.

The DT operates on the simulated data, generated based on real data measured and gathered from the 7.5 kW IM (ABB 3GAA132214-ADE) for the current case study. The data was gathered using the data acquisition system (DAS) Dewetron DEWE2 and saved into files with a different extension (*.mat, *.xlsx, *.csv, *.txt). The measured data can be anything regarding the motor's operation, namely input currents and voltages, consumed and shaft powers, torque and angular velocity on data acquisition, and other additional data calculated from them. According to DAS tuning (16Hz–100kHz), the parameters can be measured with different frequencies, and the received data is relative to time. This feature enables to recreate the motor's behaviour precisely as it happened in the real case scenario with the help of the ROS server. It should be noted that the graph from the ROS package *rqt plot* is not included in this paper because it could not handle plotting messages at such high frequency.

In the proposed DT system, the ROS server acts as a data server and physics simulator. The idea behind it is the following: the server is a standalone subsystem of a test bench (TB) responsible for processing real measured data of the motor, calculating other motor parameters based on the processed data, and streaming them to the ROS topics available for models.

ROS nodes are ROS server components performing calculations, real data processing, and streaming of data. The real data is fetched to the appropriate ROS node presented in the server, processed and translated into ROS messages, and finally, sent to the DT model over the ROS bridge. The real data can be based on the empirical model (e.g., efficiency map of the motor) or the actual raw data, an example of data used for fetched ROS node is presented in Fig. 2.

Upon receiving ROS messages, the model can perform the necessary actions to simulate the mechanical, electrical, or thermal behaviour. Models can be present in any simulation environment. They are subscribed to ROS server's topics over the application programming interface (API) or the ROS bridge and configured to perform the necessary operations based on the subscribed ROS topic (e.g., rotation based on received angular speed). Furthermore, the module can feature simulated 'measurement' devices/sensors that can send back the data over the ROS bridge. In this case, the ROS nodes can process and calculate other required values, as would happen in the TB.

The current DT consists of the Unity3D model and the ROS server. The ROS server streams simulated values regarding input power (3-phase current and voltages), efficiency calculated based on measured torque, and angular velocity. The torque is calculated by the physics engine of Unity3D, whereas other values are based on the real data. This creates a problem of incorrect data calculation because Unity3D does not focus on calculating correct values on physics laws, as it is more for games, allowing developers to adjust the physics laws to the game setup. This is why the shift from the physics engine of the model environment to ROS was introduced. The ROS server would serve physical parameters based on the real TB data and independent of the modelling environment. Figure 3 depicts the above-described architecture of the TB DT. The bottom part of the figure illustrates the operation of the ROS server and the top part shows Unity3D (the visualization environment). The processed data from the ROS server is streamed over the ROS bridge to Unity, where object controllers perform actions on parts of the drive based on the received data.

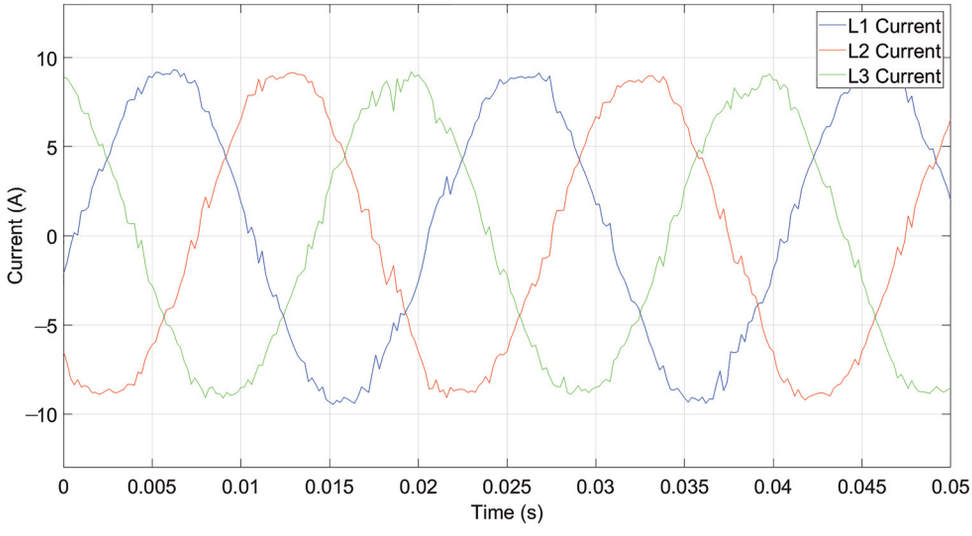

Fig. 2. Input current measurements sampled at 5 kHz frequency used for fetched ROS node.

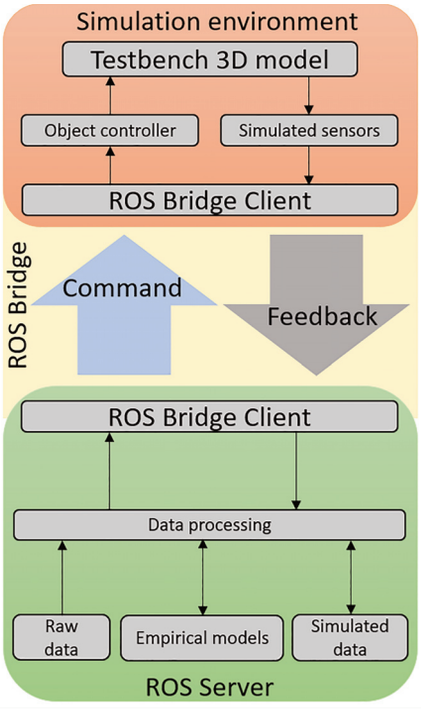

Fig. 3. Architecture of the TB DT.

Additionally, ROS can record *rosbags* – files with recorded values from topics/servers that can be played back to repeat the behaviour. Such a feature would allow us additional analytical features on the DT side.

2.1.1. ROS interfacing

To allow easy interfacing of ROS with other systems, a ROS bridge node has to be used. It converts ROS communications into a JavaScript Object Notation (JSON) file format and sends them outside the ROS ecosystem. JSON is used because of its universal format with existing libraries that support its serialization and deserialization in almost every modern programming language. Taking it one step further, the ROS bridge can be used to port specific ROS topics to and out of Message Queuing Telemetry Transport (MQTT) protocol to upscale the system and allow it to run on multiple machines around the world. The so-called MQTT bridge sends data to the remote server by taking the serialized message on a specified ROS topic and publishes it into a specified MQTT topic. The MQTT bridge is also capable of the inverse – it receives a JSON-serialized message and attempts to deserialize it into a specified ROS topic in a specific message type. Together these systems make the interfacing of ROS with any visualization solution much simpler to develop. To further simplify the deserialization process, classes that match ROS message types were created in C# for Unity3D implementation of the ROS interface. This approach can be considered the most efficient because, in this case, a ROS message delivered in the serialized form via the MQTT can be directly deserialized into an object of a matching type. This approach can be implemented similarly on the majority of the existing programming languages, making it the most straightforward and most versatile option.

Unity3D is used for visualization, see Fig. 4. Unity3D engine is connected to the physics simulator via the ROS interface, a 1:1 scale propulsion drive model with the transmission, wheel parts, and non-visible gears. The model is being assembled as the physical one, and each part is controlled by a related script, where data is fed from the middle layer.

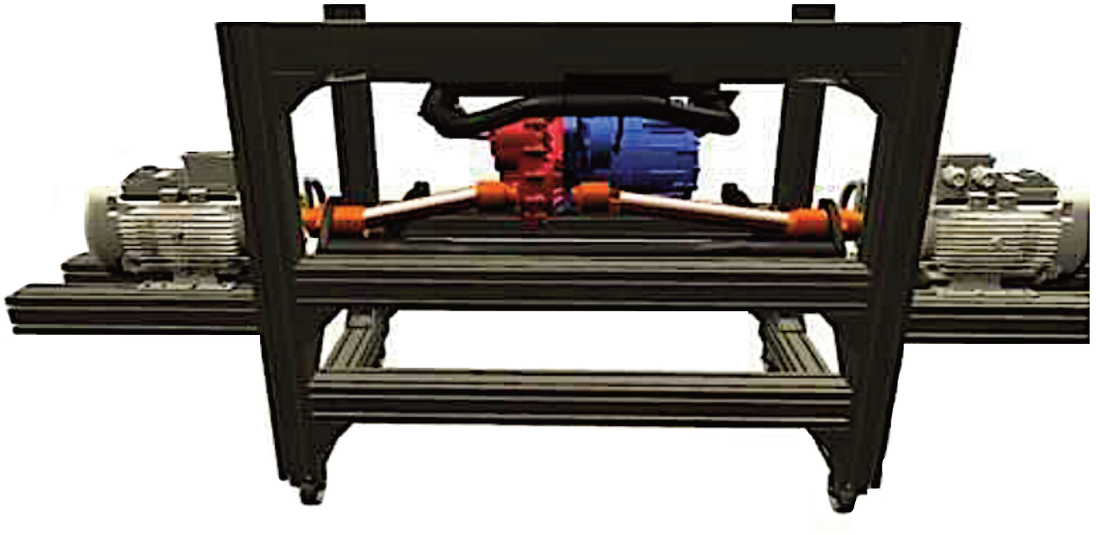

Fig. 4. Visualization of the propulsion drive test bench created in Unity3D.

2.2. Application of ROS2

In that case, the DT architecture was transferred from ROS to the ROS2 framework. ROS2 is the next version of ROS. This transfer was required for several reasons: deprecation of ROS that would follow in 2025, support of the Data Distribution Service (DDS) standard, industry-grade support [14]. Furthermore, ROS2 core libraries use the C++11 standard features and are targeting even some features of the C++14 standard, which improved the core of ROS [15]. The real-time operating systems' support was a problem from the beginning of ROS, and many researchers tried to overcome this issue, leading to the creation of several ROS "spin-offs" for real-time operating systems. However, these were still not sufficient to make ROS practical for embedded real-time systems [16]. ROS2 promises to introduce real-time support up to bare-metal microcontrollers. The ROS2 framework will become a powerful, robust, and modern middleware to be considered in any robotics/automotive vehicles project with all the implemented features.

The contribution of ROS2 to the TB DT lies in the use of DDS. In ROS2, all communication is built upon the DDS standard defined by the OMG (Object Management Group) consortium, and developers are free to switch between supported vendors. DDS supports distributed discovery that allows nodes in the distributed network to be discovered by other nodes without a broker, which is different from the custom communication protocol used in ROS, a master node. Furthermore, ROS2 provides various QoS (Quality of Service) under which the messages are delivered. As soon as our TB is directly connected over the Internet to the DT, it will feature many sensors on it, making DDS a viable option to use in such a scenario.

One of the beneficial features of ROS2 is its launch files, which are written in Python, allowing the creation of complex logic in launching ROS2 nodes. Previously, we had to create several launch files and run them in a specific sequence. Otherwise, a node responsible for the ROS bridge would crash. In ROS2, this problem is mitigated.

From the developers' point of view, ROS2 API encourages the OOP (Object-Oriented Programming) principles much stronger and helps to understand the written code better. One drawback that was observed is the lack of information about specific functionalities, and it will take time before the ROS2 community becomes as big as the one ROS has.

3. CONCLUSIONS

The primary outcome of this part of the more extensive research in developing the fully synchronized DT of the propulsion drive was the development of the ROS interface and its later transfer to ROS2. It is possible to feed it with the physical data gathered and give the data to visual simulation, which in the related use case is Unity3D. The given data simulation runs and gives logged feedback about physical interactions back to the ROS middle layer, where the model is being improved and sent back to the visual side, improving it after each data movement loop. However, some limitations were met during the development of the methodology, and more developments are in progress to reach the final research aim, see Table 1.

The ROS2 interface connected with the digital twin of the propulsion drive workbench visualized in Unity3D was introduced during the current research. Raw and simulated data as well as empirical models can be post-

Table 1. Limitations and further steps

Limitations	Further steps
The model was tested with only one type of visual simulation tool. Possible additional integrations should be performed in the middle layer to be suitable for additional software tool packages.	To establish correct torque calculations based on the real values collected from the physical TB.
	To implement a two-way connection between the physical TB and its DT.
If the DT and the TB work simultaneously over the Internet, the frequency of data acquisition may be too high to send on time and there is the possibility of lags.	The injection process flow of new components of the TB into the DT.
	To create unpredicted behaviours in the system, trigger points, and try to make the system respond to the unpredicted change, thus making it more adaptive to changes.

processed and fed to the visual simulation, where additional data is being logged and given as feedback to the middleware to improve the model and physical simulation itself. The next crucial step is to feed physical simulation directly with data from the physical drive, enabling synchronization between the real and virtual worlds through the developed interface.

ACKNOWLEDGEMENTS

The research was supported by the Estonian Research Council under grant PSG453 "Digital Twin for Propulsion Drive of Autonomous Electric Vehicle". In addition, Vladimir Kuts 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). The publication costs of this article were covered by the Estonian Academy of Sciences and Tallinn University of Technology.

REFERENCES

1. AhmadiAhangar, R., Rosin, A., Niaki, A. N., Palu, I. and Korõtko, T. A review on real-time simulation and analysis methods of microgrids. *Int. Trans. Electr. Energy Syst*., 2019, **29**(11), e12106. https://doi.org/ 10.1002/2050-7038.12106
2. Venkatesan, S., Manickavasagam, K., Tengenkai, N. and Vijayalakshmi, N. Health monitoring and prognosis of electric vehicle motor using intelligent-digital twin. *IET Electr. Power Appl.*, 2019, **13**(9), 1328–1335. https://doi.org/10.1049/iet-epa.2018.5732
3. Turner, G. Soaring through virtual aviation: The role of VR in aerospace manufacturing. *Manufacturing Global*, 2020.
4. Gevorkov, L., Rassõlkin, A., Kallaste, A. and Vaimann, T. Simulink based model of electric drive for throttle valve in pumping application. In *Proceedings of the 2018 19th International Scientific Conference on Electric Power Engineering (EPE), Brno, Czech Republic, May 16–18, 2018*. IEEE, 1–4. https://doi.org/10.1109/EPE.2018.8395996
5. Rasheed, I., Asad, B., Khaliq, H. S., Khan, M. H., Bukhari, S. Z. H. and Bukhari, S. N.-U.-H. Fast numerical techniques based analysis of electromagnetic problems using MATLAB. In *Proceedings of the 2014 12th International Conference on Frontiers of Information Technology, Islamabad, Pakistan, December 17–19, 2014.* IEEE, 2015, 115–120. https://doi.org/10.1109/FIT.2014.30
6. Kousi, N., Gkournelos, C., Aivaliotis, S., Giannoulis, C., Michalos, G. and Makris, S. Digital twin for adaptation of robots' behavior in flexible robotic assembly lines. *Procedia Manuf.*, 2019, **28**, 121–126. https://doi.org/10.1016/j.promfg.2018.12.020
7. Kuts, V., Sarkans, M., Otto, T., Tähemaa, T. and Bondarenko, Y. Digital Twin: Concept of hybrid programming for industrial robots – use case. In *Proceedings of the ASME 2019 International Mechanical Engineering Congress and Exposition, vol. 2B, Salt Lake City, UT, USA, November 11–14, 2019*. https://doi.org/10.1115/IMECE2019-10583
8. Kuts. V., Modoni, G. E., Otto, T., Sacco, M., Tähemaa, T., Bondarenko, Y. and Wang, R. Synchronizing physical factory and its digital twin through an IIoT middleware: a case study. *Proc. Est. Acad. Sci.*, 2019, **68**(4), 364–370. https://doi.org/10.3176/proc. 2019.4.03
9. Rassõlkin, A., Vaimann, T., Kallaste, A. and Kuts, V. Digital twin for propulsion drive of autonomous electric vehicle. In *Proceedings of the 2019 IEEE 60th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON), Riga, Latvia, October 7–9, 2019*. IEEE, 2020, 1–4.
10. Khaled, N., Pattel, B. and Siddiqui, A. Digital Twin development and cloud deployment for a Hybrid Electric Vehicle. In *Digital Twin Development and Deployment on the Cloud*. Academic Press, Cambridge, MA, 2020.
11. Sell, R., Coatanéa, E. and Christophe, F. Important aspects of early design in mechatronic. In *Proceedings of the 6th International DAAAM Baltic Conference Industrial Engineering, Tallinn, Estonia, April 24–26, 2008*.
12. Rassõlkin, A., Rjabtšikov, V., Vaimann, T., Kallaste, A., Kuts, V. and Partyshev, A. Digital Twin of an electrical motor based on empirical performance model. In *Proceedings of the 2020 XI International Conference on Electrical Power Drive Systems (ICEPDS), St Petersburg, Russia, October 4–7, 2020*. IEEE, 1–4.
13. Sita, E., Horváth, C. M., Thomessen, T., Korondi, P. and Pipe, A. G. ROS-Unity3D based system for monitoring of an industrial robotic process. In *Proceedings of the 2017 IEEE/SICE International Symposium on System Integration, Taipei, Taiwan, December 11–14, 2017*. IEEE, 2018, 1047–1052. https://doi.org/10.1109/SII.2017.8279361
14. Noetic Ninjemys: The Last Official ROS 1 Release. Open Robotics. https://www.openrobotics.org/blog/2020/5/23/noetic-ninjemys-the-last-official-ros-1-release (accessed 2021-05-29).
15. Thomas, D. Changes between ROS 1 and ROS 2. ROS 2 Design. http://design.ros2.org/articles/changes.html (accessed 2021-05-26).
16. Maruyama, Y., Kato, S. and Azumi, T. Exploring the performance of ROS2. In *Proceedings of the 13th International Conference on Embedded Software (EMSOFT), Pittsburgh, PA, USA, October 1–7, 2016*. ACM, 1–10. https://doi.org/10.1145/2968478.2968502

ROS-i keskmise kihi integreerimine Unity3D-ga liidese valikuna autonoomsete sõidukite jõuülekande simulatsioonideks

Vladimir Kuts, Anton Rassõlkin, Sergei Jegorov ja Viktor Rjabtšikov

Kuna autonoomne sõidukite arendamine jätkub kasvava kiirusega, suureneb ka vajadus optimeerimiseks, diagnoosiks ja erinevate autonoomsete süsteemide elementide erinevates tingimustes katsetamiseks. Kuna nimetatud protsesse tuleks teostada paralleelselt, võib see arengus põhjustada kitsaskohti ja keerukust. Digitaalsete kaksikute kontsept pakub paljutõotavat võimalust diagnoosimiseks ja testimiseks, mis viiakse läbi füüsilistest seadmetest eraldi, sisaldades ka autonoomsete sõidukite testimist virtuaalmaailmas. Virtuaalsete ja füüsiliste kaksikute vahelise kommunikatsiooni põhimõte annab võimaluse hinnata riske, puudusi, sõiduki juhtimissüsteemide füüsilisi kahjustusi ning füüsilisi kriitilisi tingimusi. Nende süsteemide vahelise side loomine toimub kiirusega, mis on piisav füüsilise sõiduki virtuaalses maailmas täpseks esitamiseks, olles endiselt trendikas teema. Selle artikli eesmärk on näidata, kuidas antud probleemi lahendada, kasutades ROS-i vahevara liidesena kahe reaalse süsteemi vahel autonoomse sõiduki tõukejõu ajami näitel. Füüsilisest ja virtuaalsest maailmast kogutud andmeid saab vahetada keskse platvormi kaudu, et võimaldada jõuseadme mudeli pidevat õpetamist ja optimeerimist, mille tulemuseks on tõhusam tee planeerimine ning autonoomse sõiduki enda energiasäästlik kasutamine. Lisaks on esitatud robotite operatsioonisüsteemi (ROS) ja järgmise versiooni ROS2 võrdlev analüüs, kus on käsitletud nendevahelisi erinevusi ning välja toodud platvormide puudused.