Jurnal Otomasi Kontrol dan Instrumentasi Vol 18 (2), 2026 ISSN: 2085-2517
https://doi.org/10.5614/joki.2026.18.2.7 E-ISSN: 2460-6340

Design and Development of an O-SEATER Simulation Platform for the BRIN Operational Area Using NVIDIA Isaac Sim

Rancang Bangun Platform Simulasi O-SEATER pada Area Operasional BRIN Menggunakan NVIDIA Isaac Sim

[1]Rifki Rochiman*), [1]Anugrah Adiwilaga, [1]Munawir Munawir, [2]Midriem Mirdanies, [2]Roni Permana Saputra

[1]Department of Computer Engineering, Universitas Pendidikan Indonesia, Bandung, 40154, Indonesia

[2]Research Center for Smart Mechatronics, Badan Riset dan Inovasi Nasional, Bandung, 40135, Indonesia

*) *corresponding email: rifkirochiman22@upi.edu*

Abstract

Developing algorithms for mobile robot systems requires a simulation environment that models operational conditions. This research develops a simulation platform for the O-SEATER autonomous vehicle in the virtual environment of BRIN KST Samaun Samadikun using NVIDIA Isaac Sim. The virtual model was connected to the physical vehicle via a wireless network using the Robot Operating System 2 (ROS 2) for Hardware-in-the-Loop (HIL) and rosbag playback tests. The playback replicated the actual movements perfectly, with a Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) of 0.0000 and an ideal Real-Time Factor (RTF) of 1.0010. HIL simulation ran 1.81 times slower (RTF = 0.5518), with a memory spike of 27.1 GiB, 214.63 ms network latency, and 15% packet loss. To compensate for this temporal deviation, we applied Dynamic Time Warping (DTW), resulting in high-precision HIL tracking (Linear X MAE = 0.0299 m/s; Angular Z MAE = 0.0224 rad/s). The high communication latency is an architectural limitation. Nevertheless, we achieved interactive communication and generated virtual sensor data (RGB and IMU). This platform remains a sufficiently valid algorithm-testing medium and the basis of the O-SEATER Digital Twin system.

Keywords: digital twin, hardware-in-the-loop, NVIDIA Isaac Sim, o-seater, ROS 2

Abstrak

Pengembangan algoritma untuk sistem robot mobil memerlukan lingkungan simulasi yang memodelkan kondisi operasional. Penelitian ini mengembangkan platform simulasi untuk kendaraan otonom O-SEATER di lingkungan virtual BRIN KST Samaun Samadikun menggunakan NVIDIA Isaac Sim*. Model virtual tersebut dihubungkan ke kendaraan fisik melalui jaringan nirkabel menggunakan Robot Operating System 2 (ROS 2) untuk uji* Hardware-in-the-Loop *(HIL) dan pemutaran ulang rosbag. Pemutaran ulang tersebut mereplikasi gerakan aktual dengan sempurna, dengan* Mean Absolute Error *(MAE) dan* Root Mean Square Error *(RMSE) sebesar 0,0000 serta* Real-Time Factor *(RTF) ideal sebesar 1,0010. Simulasi HIL berjalan 1,81 kali lebih lambat (RTF = 0,5518), dengan lonjakan penggunaan memori sebesar 27,1 GiB, latensi jaringan sebesar 214,63 ms, dan kehilangan paket sebesar 15%. Untuk mengompensasi penyimpangan waktu ini, kami menerapkan* Dynamic Time Warping *(DTW), yang menghasilkan pelacakan HIL berpresisi tinggi (MAE Linear X = 0,0299 m/s; MAE Sudut Z = 0,0224 rad/s). Latensi komunikasi yang tinggi merupakan keterbatasan arsitektural. Meskipun demikian, kami berhasil mencapai komunikasi interaktif dan menghasilkan data sensor virtual (RGB dan IMU). Platform ini tetap menjadi media pengujian algoritma yang cukup valid serta dasar bagi sistem* Digital Twin *O-SEATER.*

Kata Kunci: digital twin*,* hardware-in-the-loop*, NVIDIA Isaac Sim,* o-seater*, ROS 2*

Paper accepted July 31th, 2026 – paper revised September 7th, 2026– approved September 24th, 2026
This paper is open access with CC BY-SA *license.*

1 Introduction

Advances in robotics and autonomous vehicle technology have increased the need for reliable methods to design, test, and validate algorithms before implementing them on physical systems. Testing on physical platforms is often time-consuming, expensive, and resource-intensive, and it also carries risks. As an alternative, simulation enables algorithm testing and system configuration in a controlled, repeatable, and relatively safe virtual environment. Today, realistic simulation environments can be built to test robotic systems by replicating robotic components, environments, and sensors [1],[2].

One widely used approach to bridging the gap between virtual and physical systems is the Digital Twin concept. A Digital Twin is essentially a virtual model of a physical system. This representation enables information exchange between physical and virtual systems, supporting system monitoring, analysis, assessment, and

development [3]. Researchers have explored Digital Twin technology for various autonomous vehicle applications, including vehicle localization, algorithm development, system evaluation, and safety analysis. [1],[2]. Digital Twins enable early-stage development and testing in a controlled simulation environment through their interconnected physical and virtual components. This helps mitigate risks associated with physical system testing that stem from trial-and-error processes.

O-SEATER is a single-passenger electric autonomous vehicle developed by the Research Centre for Smart Mechatronics at the National Research and Innovation Agency (BRIN) as an outdoor variant of the SEATER, previously designed for indoor use. O-SEATER is designed to operate autonomously in restricted locations, such as the BRIN KST Samaun Samadikun area, and can be used as a micro-shuttle for inter-building travel or as an autonomous vehicle in recreational areas. O-SEATER is currently under development, informed by research on autonomous electric vehicle technology, and aligns with eco-friendly mobility system initiatives [4], [5],[6]. O-SEATER integrates many hardware and software components, so it requires a representative simulation environment to develop and test it before applying algorithms and system configurations to the actual O-SEATER.

Several simulation techniques commonly support autonomous vehicle validation and testing. Two of the most common are Software-in-the-Loop (SIL) and Hardware-in-the-Loop (HIL). SIL enables comprehensive verification of software algorithms in a virtual environment. HIL integrates real hardware components into the simulation, producing behavior more closely aligned with physical systems [7]. HIL is highly beneficial in autonomous vehicle development because it allows testing of control systems, perception, and other system components on real hardware while the environment is simulated virtually [8],[9]. Various robotics and autonomous vehicle simulators, such as Gazebo, CARLA, and other physics-based simulation platforms, have been used [10] to facilitate the development and evaluation of autonomous systems. These methods demonstrate the importance of combining virtual simulations with physical components to create a more representative testing environment.

Another crucial component for integrating virtual and physical systems is communication middleware. Robot Operating System 2 (ROS 2) has gained popularity in the robotics community for its distributed communication framework and its support for modular robotics software development [11],[12].ROS 2 can connect sensors, actuators, controllers, and simulation environments using a publish-subscribe communication model. With this support, ROS 2 serves as middleware for integrating virtual and physical components in HIL systems and Digital Twin development.

Digital Twin technology has also been applied across several smart-vehicle domains, including vehicle localization [13], algorithm development for Advanced Driver Assistance Systems (ADAS) [14], and edge-based Digital Twin architectures for connected and autonomous vehicles [15]. Researchers have explored Digital Twin techniques for trajectory tracking [16], safety and security analysis [17], and integrated representations of vehicles, drivers, and road surroundings [18]. These achievements demonstrate the potential of Digital Twin technology to support multiple stages of autonomous vehicle development, including algorithm development, localization, system control, safety evaluation, and connected mobility.

Researchers have also explored HIL systems to integrate real hardware and simulations for autonomous vehicle applications. Researchers developed a CARLA-based HIL simulation environment for real-time testing of autonomous driving algorithms, integrating physical system components with virtual environments [19]. Researchers have developed information modelling methods to standardize and manage Digital Twin simulation systems [20]. In robotics, Digital Twins have also been used to integrate physical and virtual robotic systems for testing robot capabilities [21]. Additionally, the NVIDIA Omniverse platform supports data collection, modelling, and virtual environment creation, providing a vital technological foundation for robotics and autonomous system applications [22].

To clarify this research's position relative to previous studies, we compared the simulator used, HIL implementation, ROS 2 usage, physical vehicle involvement, two-way communication between virtual and physical systems, and the use of specific operational environments. Table 1 shows this comparison.

Based on Table 1, previous studies have applied various approaches to integrate simulation environments, physical systems, and Digital Twin technology. Cardozo and Hlavác [20] developed a CARLA-based Hardware-in-the-Loop environment involving a physical vehicle; however, the research utilized the CARLA simulator and was not specifically targeted at O-SEATER or the BRIN KST Samaun Samadikun operational environment. Sekkat et al. [22] developed a Gazebo and ROS 2-based Digital Twin connected to a physical robot, while Kim et al. [21] developed a Digital Twin framework for a wheeled mobile robot using Isaac Sim and Gazebo with ROS 2 support. Nevertheless, these two studies have not demonstrated HIL integration and two-way communication between a physical vehicle and a simulator within a specific operational environment.

Table 1. Comparison of simulation platforms with previous research

Research	Simulator	HIL	ROS 2	Physical Vehicle	Two-Way Communication	Specific Environment
Cardozo & Hlavác [20]	CARLA	✓	–	✓	–	–
Sekkat et al. [22]	Gazebo	–	✓	✓	–	–
Kim et al. [21]	Isaac Sim & Gazebo	–	✓	–	–	–
This study	Isaac Sim	✓	✓	O-SEATER	✓	BRIN KST Samaun Samadikun

In contrast to previous studies, this research integrates NVIDIA Isaac Sim, ROS 2/DDS, the physical O-SEATER, and the BRIN KST Samaun Samadikun environment model into a single simulation platform. This integration enables two-way communication between virtual and physical systems through Simulation-to-Real and Real-to-Simulation mechanisms. Furthermore, using a three-dimensional reconstruction of the operational environment allows testing under virtual conditions that represent the actual operating area. Thus, this research focuses not only on developing virtual models but also on integrating physical and virtual vehicles through HIL and data synchronization to support O-SEATER system testing and development.

Although previous studies have demonstrated the use of simulators, Digital Twins, ROS 2, and Hardware-in-the-Loop approaches in developing robotic and autonomous vehicle systems, limitations remain in integrating these components into a single platform for autonomous vehicles within specific operational environments. CARLA-based research has shown HIL capabilities connecting simulators with physical vehicles [20], while Gazebo-based research has demonstrated Digital Twin and ROS 2 implementation on physical robots [22]. In contrast, Kim et al. [21] used Isaac Sim and ROS 2 to support wheeled mobile robot simulations. However, these studies have not specifically integrated the O-SEATER vehicle model, the three-dimensional representation of the BRIN KST Samaun Samadikun operational environment, two-way communication between virtual and physical systems, and HIL mechanisms in a single NVIDIA Isaac Sim-based simulation platform.

To address this gap, this research's main contribution is the development of the O-SEATER simulation platform, which integrates NVIDIA Isaac Sim, ROS 2/DDS, the virtual O-SEATER model, the three-dimensional BRIN KST Samaun Samadikun environment model, and the physical O-SEATER in a Hardware-in-the-Loop configuration. The platform enables two-way data exchange through Simulation-to-Real and Real-to-Simulation mechanisms, allowing commands and data to flow between the virtual system and the physical vehicle. Additionally, integrating a specific operational environment enables testing under virtual conditions that closely represent the actual operating area. This approach is expected to provide a safer, controlled, and repeatable testing environment to support the development and validation of O-SEATER algorithms before direct implementation on physical vehicles.

2 Method

2.1 Research Workflow

This research followed several stages, beginning with a literature review and requirements identification to inform the simulation platform design. Subsequent stages included designing the virtual O-SEATER model in SolidWorks and the BRIN KST Samaun Samadikun operational-area simulation environment in Blender. The O-SEATER model was converted to URDF, and the environment model to USD; both were then integrated into NVIDIA Isaac Sim. Next, the simulation platform was integrated with ROS 2/DDS and validated to ensure connectivity between the virtual and physical systems. Testing used a Hardware-in-the-Loop approach to represent Simulation-to-Real communication and rosbag data playback to represent Real-to-Simulation communication. The test results were then used to evaluate the simulation platform's performance and analyse the outcomes. The evaluated platform was subsequently used to support algorithm testing, sensor data acquisition, and O-SEATER Digital Twin implementation. Figure 1 provides a detailed analysis of the research process used in this investigation.

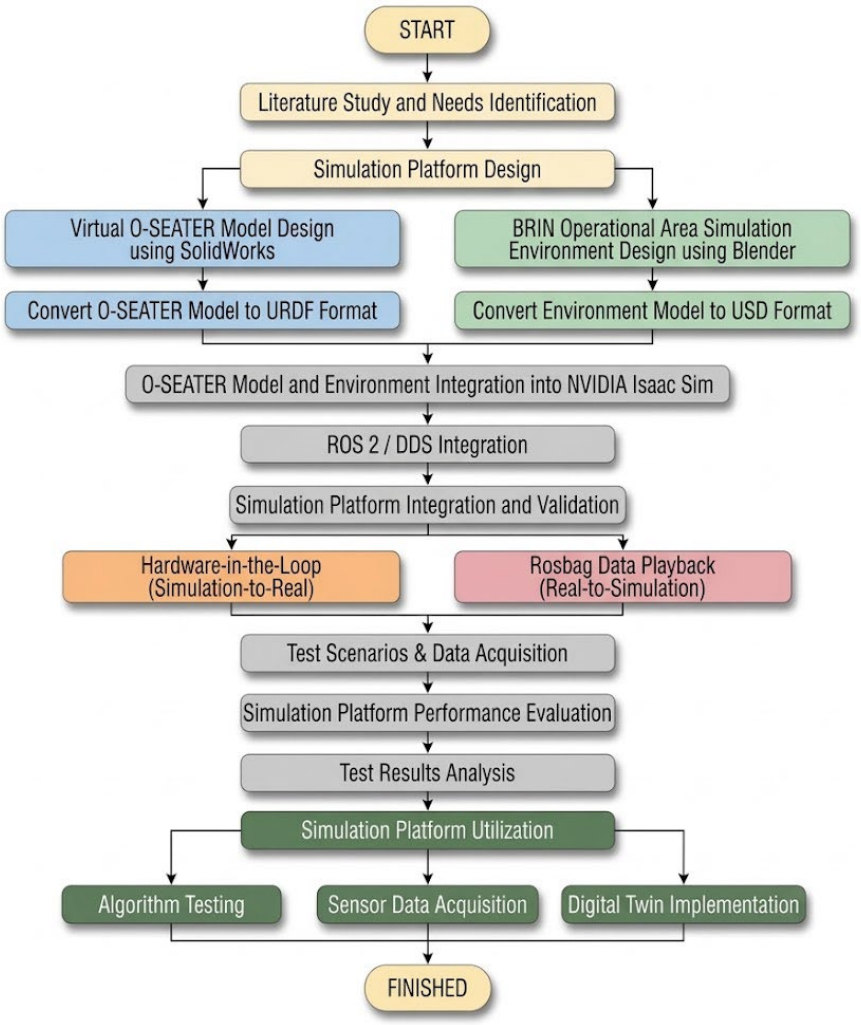


Figure 1. Research flow for designing the O-SEATER simulation platform using NVIDIA Isaac Sim.

2.2 Research Equipment

Hardware and software supported system integration, communication testing between virtual and physical environments, and the development of the O-SEATER simulation platform. The software supported communication integration, simulation environment development, and 3D visualisation on the platform. Meanwhile, the hardware served as both computing equipment and a system controller. Detailed specifications for each utilised device are shown in Tables 2 and 3.

Table 2. Hardware used

No	Device	Specification	Function
1	Computer/PC	1) Processor: Intel(R) Core (TM) i9-10900 2) Storage: 1TB SSD 3) RAM: 32GB 4) GPU: • NVIDIA Quadro RTX 4000 • VRAM 8GB	Running the process of modeling, simulation, and system integration
2	Joystick	Logitech F710 Wireless Gamepad	Controlling the virtual & actual O-SEATER in Hardware-in-the-Loop testing and physical
3	O-SEATER	Single-Passenger Electric Autonomous Vehicle	Objects used in Hardware-in-the-Loop testing and rosbag data recording

Table 3. Software used

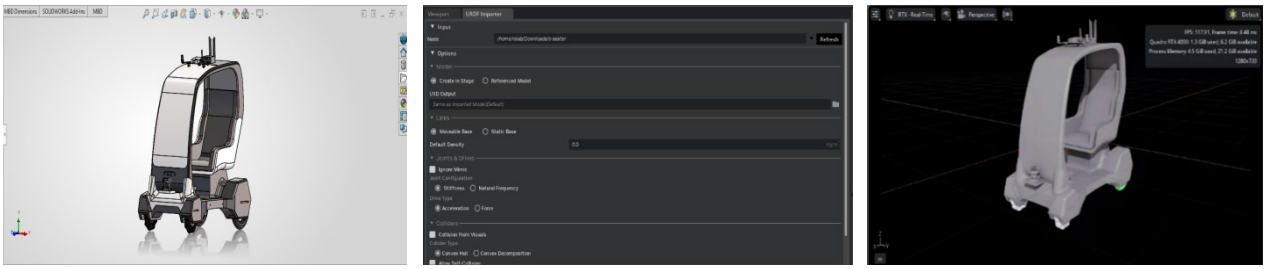
No	Device	Version	Function
1	Ubuntu	22.04 LTS	Main operating system
2	NVIDIA Isaac Sim	4.5.0	Development and visualization of the simulation platform
3	ROS 2	Humble	Communication between the virtual system and the actual O-SEATER
4	SolidWorks	2020	Design of the virtual O-SEATER model
5	Blender	4.5 LTS	Design of the BRIN simulation environment
6	Python	3.10	Development of scripts for process automation and rosbag data playback

2.3 Simulation Platform Design

This simulation platform was implemented by constructing virtual replicas of the O-SEATER and its operational environment in NVIDIA Isaac Sim. The objective of this stage was to create a simulation environment that accurately represents real-world operational conditions. This environment could then be used for algorithm testing, Hardware-in-the-Loop (HIL) implementation, and data playback. The simulation platform design process consisted of three main stages: designing the virtual O-SEATER model, designing the BRIN KST Samaun Samadikun operational environment, and integrating all components in NVIDIA Isaac Sim.

2.3.1 Virtual O-SEATER Model Design

The virtual O-SEATER model was developed from available Computer-Aided Design (CAD) designs in SolidWorks. During modelling, the dimensions and mechanical configuration of the actual O-SEATER were preserved to ensure the virtual model accurately represented the physical system used in this research. The completed CAD model was then converted to a format compatible with NVIDIA Isaac Sim using the URDF Importer, which automatically generated the model in Universal Scene Description (USD) format. Figure 2 shows the virtual model conversion process.



(a) (b) (c)

Figure 2. Stages of converting the O-SEATER virtual model: (a) CAD model in SolidWorks, (b) conversion process using URDF Importer, and (c) O-SEATER model in USD format on NVIDIA Isaac Sim.

We used the USD format because NVIDIA Isaac Sim uses it as the standard format for representing 3D objects, including their hierarchical structures, materials, and physical properties. One advantage of this format is its ease of integration, as the virtual O-SEATER model can connect directly to physics modules, virtual sensors, and ROS 2 communication.

After successfully integrating the O-SEATER model into NVIDIA Isaac Sim, the vehicle's drive system configuration was adjusted to match the physical vehicle's mechanics. O-SEATER uses two rear wheels as the main drive wheels (rear_left_wheel and rear_right_wheel), while the two front wheels (front_left_wheel and front_right_wheel) use an omni-wheel configuration as supporting wheels. Figure 3 shows the drive-wheel positions and the Differential Controller parameters used in the simulation environment.

The O-SEATER drive system uses a differential drive configuration. The wheel Distance parameter on the Differential Controller was set to 0.6 m, while the wheel Radius was set to 0.15 m. These parameters served as the primary geometric parameters determining the relationship between the drive wheels' angular velocity and vehicle movement in the simulation.

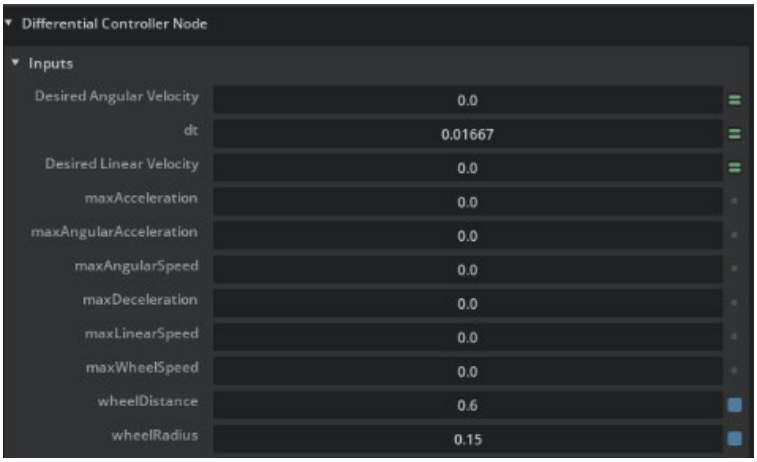


Figure 3. Drive system configuration and O-SEATER Differential Controller parameters in NVIDIA Isaac Sim.

The O-SEATER kinematics model was determined based on the angular velocity of the left and right drive wheels. The vehicle's linear velocity v and angular velocity ω were determined by the wheel radius r, the distance between drive wheels L, and the angular velocity of the left wheel ω_L and the right wheel ω_R. The relationship between wheel velocity and vehicle velocity is expressed in Equations (1) and (2):

$$v = \frac{r}{2}\left(\omega_R + \omega_L\right) \tag{1}$$

$$\omega = \frac{r}{L}\left(\omega_R - \omega_L\right) \tag{2}$$

with $r = 0.15$ m and $L = 0.6$ m. Based on these parameters, the relationship between wheel velocity and vehicle movement becomes:

$$v = 0.075\left(\omega_R + \omega_L\right)$$

$$\omega = 0.25\left(\omega_R - \omega_L\right)$$

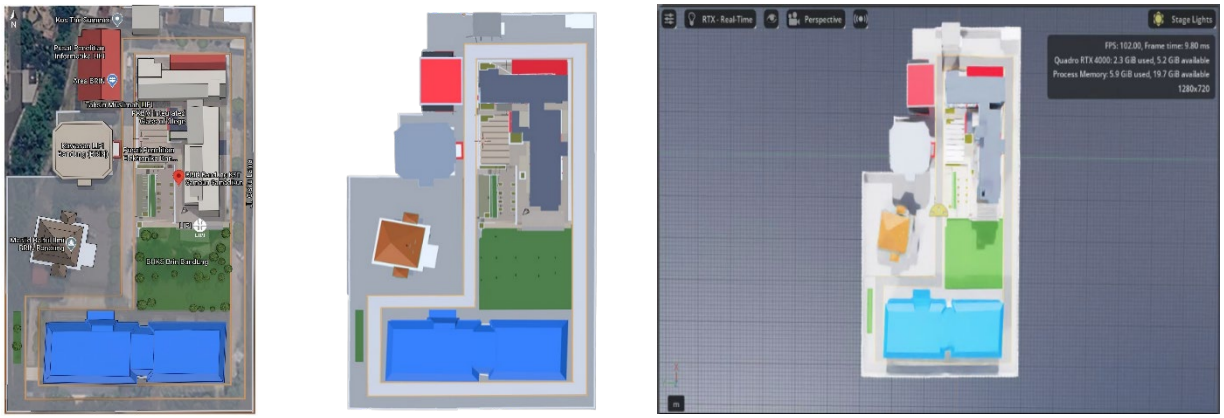
The linear and angular velocities generated by both drive wheels are subsequently used by the Differential Controller to determine the virtual O-SEATER model's movement. When both rear wheels have the same angular velocity, the vehicle moves straight; a difference in velocity between the left and right wheels changes the vehicle's angular velocity, causing it to follow a curved path. The omni-wheel front wheels act as supporting wheels during vehicle movement.

With this configuration, the virtual O-SEATER model not only represents the vehicle's geometry but also preserves the drive-system characteristics used in the simulation. This model then serves as the basis for configuring physics, sensors, actuators, ROS 2/DDS integration, and Hardware-in-the-Loop testing in subsequent stages.

2.3.2 BRIN Operational Environment Design

The BRIN KST Samaun Samadikun operational environment was modelled in Blender to simulate its surroundings. The model was based on the actual conditions at the research location, replicating the original operational area, including buildings, roads, and surrounding zones. The completed model was then exported to NVIDIA Isaac Sim as a simulation environment. Figure 4 shows the development stages.

The built virtual environment is then used in NVIDIA Isaac Sim as a simulation environment to run the virtual O-SEATER model. Using an environment that represents the operational area enables testing of vehicle movements and interactions between the virtual system and physical devices under conditions that match the actual operational location. This virtual environment also supports Simulation-to-Real and Real-to-Simulation communication scenarios, as well as Hardware-in-the-Loop implementation.

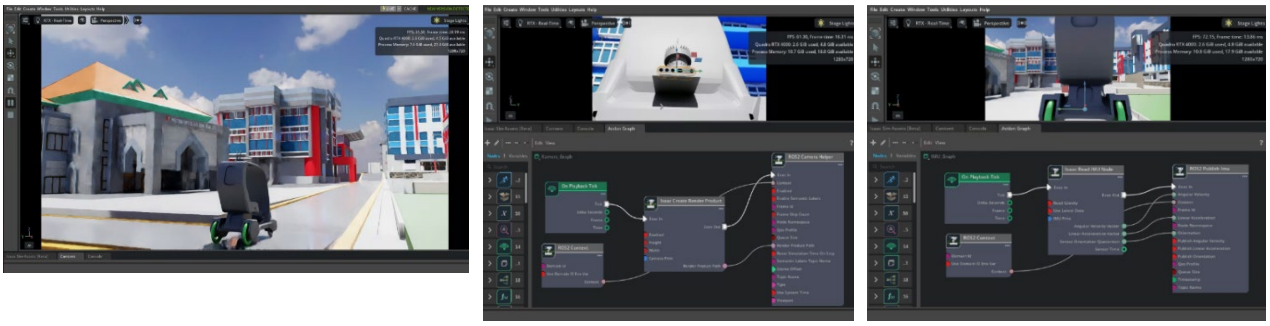


(a) (b) (c)

Figure 4. Design of BRIN operational environment: (a) Layout of BRIN KST Samaun Samadikun, (b) modeling process using Blender, and (c) result of environment integration in NVIDIA Isaac Sim.

2.3.3 Simulation Platform Integration in NVIDIA Isaac Sim

The next step was to integrate the virtual O-SEATER model and its operational environment into NVIDIA Isaac Sim to build a comprehensive simulation platform. This step involved configuring the physics engine, lighting, materials, and object placement so the virtual model could interact realistically with the simulation environment. The simulation platform also included simulated sensors, namely an Intel® RealSense™ D455 camera and an Inertial Measurement Unit (IMU). These sensors generated sensor data during the simulation. Figure 5 shows the results of the simulation platform integration.



(a) (b) (c)

Figure 5. O-SEATER simulation platform built on NVIDIA Isaac Sim: (a) Simulation platform results in NVIDIA Isaac Sim, (b) Integration of Intel® RealSense™ D455 camera, and (c) Integration of IMU sensor

The integrated simulation platform was then used as the foundation for implementing ROS 2-based communication and Hardware-in-the-Loop testing. During this phase, the simulation platform not only visualised O-SEATER movement but also served as a virtual environment for generating sensor data and testing interactions between the virtual system and physical devices.

2.4 ROS 2 Integration and Hardware-in-the-Loop Implementation

The developed simulation platform was then integrated with the Robot Operating System 2 (ROS 2) to enable real-time data exchange between the virtual and physical O-SEATER. To enable data exchange without a direct physical connection, both systems communicated via the Data Distribution Service (DDS) over Wi-Fi. Figure 6 shows the communication architecture used in this research.

On the simulation side, NVIDIA Isaac Sim ran the virtual O-SEATER model and generated simulation data, while ROS 2 served as middleware, distributing data between the virtual system and the physical device. In NVIDIA Isaac Sim, the communication configuration was set up via Action Graph by connecting nodes that receive and transmit data through ROS 2. Figure 7 shows the Action Graph configuration implemented on the simulation platform.

The control flow begins in On Playback Tick, triggering Action Graph execution during the simulation. ROS2 Context provides the ROS 2 communication context, while ROS2 Subscribe Twist receives velocity commands from a ROS 2 topic in geometry_msgs/msg/Twist format. The linear and angular components are then separated using Break 3-Vector. The resulting linear velocity on the X-axis and angular velocity on the Z-axis are forwarded to the Differential Controller, which converts them into left- and right-wheel velocity commands. The Articulation Controller then applies these commands to the virtual O-SEATER wheels, producing vehicle movement within the simulation environment.



Figure 6. Communication architecture between the actual O-SEATER and the simulation platform using ROS 2 based on DDS over a Wi-Fi network

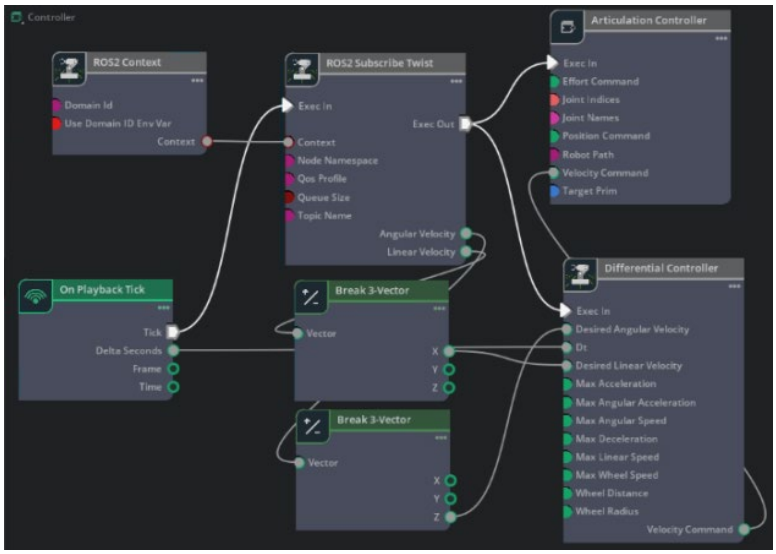


Figure 7. Action Graph configuration for controlling the O-SEATER virtual model via ROS 2 on NVIDIA Isaac Sim

The communication mechanism supports two-way data exchange. In Simulation-to-Real communication, the virtual system transmits control commands via a ROS 2 topic to the physical O-SEATER as control inputs. Conversely, in Real-to-Simulation communication, the physical O-SEATER sends data via ROS 2 to NVIDIA Isaac Sim, which serves as input for the virtual system.

The Hardware-in-the-Loop (HIL) implementation used a joystick as the operator control interface, forwarding control commands via the /cmd_vel topic in ROS 2 and distributing them via a publish-subscribe mechanism to configured systems on both the actual O-SEATER and in NVIDIA Isaac Sim. This control data moved the virtual O-SEATER model in the simulation while also serving as input to generate movement of the actual O-SEATER on the physical device. Figure 8 shows the Hardware-in-the-Loop implementation flow.

With this mechanism, the virtual O-SEATER and the actual O-SEATER receive the same control commands based on the operator's input. The real system then publishes feedback, along with the control data, via ROS 2 to monitor the vehicle's operational condition in the simulation environment. This combination allows control algorithms and configurations to be tested on the simulation platform before applying them to the physical system, so the physical system can be used once the necessary configurations have been evaluated in the virtual environment.

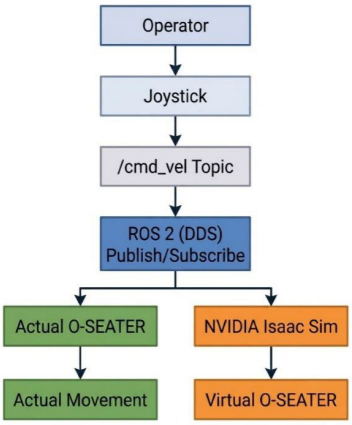


Figure 8. Hardware-in-the-Loop implementation between the virtual O-SEATER and the actual O-SEATER using ROS 2

The ROS 2 communication developed in this study forms the basis for implementing rosbag data replay, discussed in the following subsection. The same publish-subscribe mechanism is used to republish recorded data from the actual O-SEATER to the simulation platform, thereby reproducing the communication data and the vehicle's operational conditions in the virtual environment.

2.5 Rosbag Data Playback Implementation

Rosbag data playback was implemented to reproduce the O-SEATER's actual operational conditions on the simulation platform. This step used recorded data from the physical system as inputs to the virtual model, enabling the O-SEATER's behaviour in the simulation environment to mirror previously recorded operational data.

In this research, we first collected operational data from O-SEATER using the rosbag utility on ROS 2. The recorded data comprised Revolutions Per Minute (RPM) information from the wheels during operation. Subsequently, the recorded data was used as input to the simulation platform via a playback mechanism employing a Python script developed in this research. Figure 9 shows the flow of the rosbag data playback implementation.

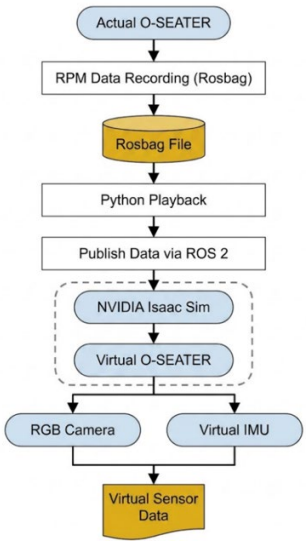


Figure 9. Flow of rosbag data playback implementation on the O-SEATER simulation platform

The Python code rpm_relay.py processes the published RPM data from the playback process. It bridges the wheel RPM data and the virtual O-SEATER model in NVIDIA Isaac Sim. The code subscribes to the /wheel_rpm/measured topic and extracts the left and right wheel RPM values from the received data. These values are converted to angular velocity in rad/s and published via the /joint_states topic using the sensor_msgs/msg/JointState message type. The converted angular velocity is mapped to rear_left_wheel_joint and rear_right_wheel_joint as wheel speed inputs for the virtual model. Figure 10 shows the data conversion mechanism.

```
START:
    Initialize ROS 2
    Create subscriber on "/wheel_rpm/measured"
    Create publisher on "/joint_states"

    WHILE node is running:
        Receive Float32MultiArray message

        IF data count >= 2:
            Get left wheel RPM
            Get right wheel RPM

            Convert left wheel RPM to rad/s
            Convert right wheel RPM to rad/s

            Create JointState message
            Set joint names:
                "rear_left_wheel_joint"
                "rear_right_wheel_joint"

            Set velocity:
                left wheel angular velocity
                right wheel angular velocity

            Publish JointState message

    Stop ROS 2 node
END
```

Figure 10. Python pseudocode for rpm_relay.

On the NVIDIA Isaac Sim side, the /joint_states data is received via an Action Graph comprising the nodes On Playback Tick, ROS2 Context, ROS2 Subscribe Joint State, and Articulation Controller. The On Playback Tick node triggers the Action Graph during simulation, while ROS2 Context initialises the ROS 2 communication context in the simulation environment. The ROS2 Subscribe Joint State node receives sensor_msgs/msg/JointState messages and provides joint names and joint velocity values. This data is then passed to the Articulation Controller, which applies it to the virtual O-SEATER wheel joints. Figure 11 shows the Action Graph configuration during playback.

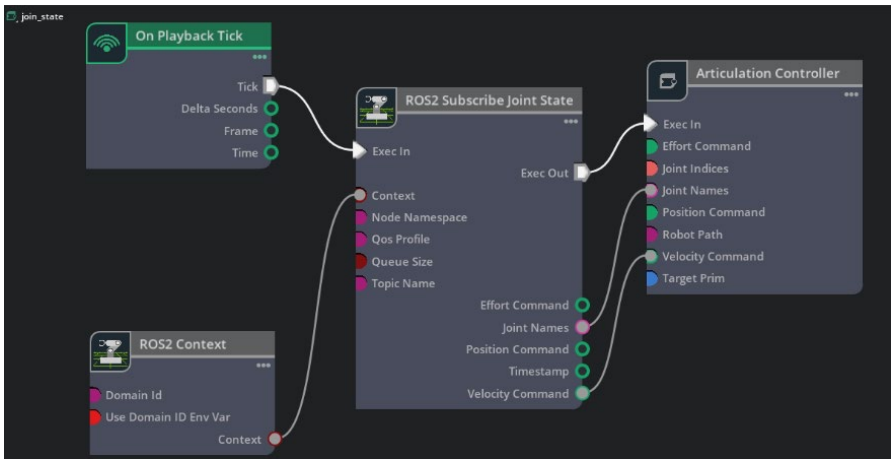


Figure 11. Action Graph configuration for receiving JointState data on NVIDIA Isaac Sim

Under this mechanism, the playback data flow begins with RPM data recorded from the actual O-SEATER and is played back via ROS 2 on the /wheel_rpm/measured topic. This data is then processed by the rpm_relay.py Python script, converted to wheel angular velocity, and published on /joint_states. The Action Graph in NVIDIA Isaac Sim then receives the JointState data and forwards it to the Articulation Controller to move the virtual O-SEATER wheel joints. This allows data from the actual vehicle operation to serve as input to the virtual model, reproducing the vehicle's movement response in the simulation environment.

During playback, the simulation platform can also generate virtual sensor data from the Intel® RealSense™ D455 camera and the Inertial Measurement Unit (IMU) integrated into the virtual O-SEATER model. This sensor data can be published via ROS 2 and used as input for algorithms or analysis in the next testing stage. The playback results are then used in test scenarios to evaluate how the virtual model responds to real O-SEATER operational data.

2.6 Testing Scenario

Testing evaluated the response suitability between the actual O-SEATER and the virtual model in NVIDIA Isaac Sim, the reliability of ROS 2/DDS-based wireless communication under standard operating conditions, and computational resource utilization. All movement tests were executed on a straight path approximately ± 50 m in the BRIN KST Samaun Samadikun operational area in a single testing cycle.

More specifically, to comprehensively validate the system's functionality, this movement evaluation includes two main testing methods:

1. Control Testing (Hardware-in-the-Loop): Testing real-time two-way communication where the operation of the virtual O-SEATER in the NVIDIA Isaac Sim environment directly transmits control data, so the actual O-SEATER also moves in response to the commands synchronously.
2. Rosbag Data Playback Testing: Testing motion reproduction using operational data (RPM) recorded from the actual O-SEATER. The data is then played back in a simulation environment using a Python script to move the virtual O-SEATER.

2.6.1 Evaluation of Kinematic Suitability and Motion Response

In the first testing method (HIL Control), motion command data via the /cmd_vel topic is evaluated to ensure that command transmission is consistent between the virtual and physical systems. In the second testing method (Playback), rear-wheel RPM data (left and right) recorded from the actual system is converted to wheel angular velocity (rad/s) by the Python relay node rpm_relay.py, published to the /joint_states topic, and applied to the virtual model using the Articulation Controller.

A direct point-by-point comparison using the original timestamps would introduce misalignment errors because the virtual system lags the real system during the HIL scenario. Thus, as a preprocessing step, we used Dynamic Time Warping (DTW). DTW finds the best match between two time series by searching for a warping path that minimises the cumulative distance between the real sequence and the virtual sequence.al sequence X and the virtual sequence Y [23]. The DTW distance is formulated in Equation (3):

$$DTW(X,Y) = \min_{W}\left[\sum_{k=1}^{K} d\,(w_k)\right] \tag{3}$$

where the contiguous set of matrix elements constituting the warping route is represented by $W = w_1, w_2, \ldots, w_K$, and $d(w_k)$ at the k-th point in the path, is the distance (such as the Euclidean distance) between the elements of the real and virtual sequences [24]. Following this temporal alignment, the response error rate is quantitatively calculated using the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics according to Equations (4) and (5):

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |\, y_i - \hat{y}_i \,| \tag{4}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (\, y_i - \hat{y}_i \,)^2} \tag{5}$$

where y_i is the actual measurement value, $\hat{y}_i$ is the simulated value, and N is the number of data pairs analyzed.

In addition to matching the values (magnitude), we evaluate temporal alignment between the actual and virtual systems using the Real-Time Factor (RTF) metric, commonly used in robotics simulation performance evaluations to indicate how fast or slow a simulation runs relative to real time [25]. In this study, RTF is calculated as the ratio of the total actual time to the total simulation time required to complete the same number of data samples, as formulated in Equation (6):

$$RTF = \frac{t_{actual}}{t_{virtual}} \tag{6}$$

where t_{actual} is the total duration of data recording on the actual system, and $t_{virtual}$ m is the total duration of data recording on the virtual system, for the same amount of data (N) the same. An RTF = 1 indicates the simulation runs perfectly in sync with real time; RTF < 1 means the simulation is slower than real time, usually due to high computational load from physics processing and rendering on NVIDIA Isaac Sim; while RTF > 1 means the simulation runs faster than real time [25].

In addition to numerical evaluation, video documentation of movements between the physical vehicle and the virtual model supports visual verification of synchronisation and qualitatively confirms alignment of movement patterns.

2.6.2 Wi-Fi Network Communication Evaluation

Data exchange between the actual O-SEATER and the simulation platform on the host computer occurs over a Wi-Fi network using the ROS 2 protocol, which is based on the Data Distribution Service (DDS). The reliability of data transmission under normal operating conditions is measured using the Packet Loss Ratio (PLR) and the average latency (L_{avg}), calculated using Equations (7) and (8):

$$\text{PLR} = \left(\frac{P_{tx} - P_{rx}}{P_{tx}}\right) \times 100\% \tag{7}$$

$$L_{\text{avg}} = \frac{1}{N}\sum_{i=1}^{N}\left(t_{rx,i} - t_{tx,i}\right) \tag{8}$$

Where P_{tx} is the total data packets transmitted, P_{rx} is the total data packets successfully received, $t_{tx,i}$ is the transmission time of the i-th packet, and $t_{rx,i}$ is the reception time of the i-th packet based on synchronized system time.

This evaluation focuses on measuring the performance of real-time data exchange under normal operational conditions in the BRIN KST Samaun Samadikun area. In addition to assessing normal data exchange, we conducted a safety (failsafe) observation by intentionally severing the Wi-Fi connection during the Hardware-in-the-Loop (HIL) manoeuvre. Upon losing the network connection, the ROS 2 communication bridge immediately stopped receiving new /cmd_vel control commands. As observed, both the actual O-SEATER and the virtual model successfully engaged their autonomous failsafe mechanism, halting all movement instantly rather than continuing at the last received velocity command. This result confirms that the platform meets the fundamental safety requirements for operating an autonomous vehicle over a wireless network.

2.6.3 Evaluation of Computational Load and Virtual Sensor Data

The simulation recorded host-computer resource usage to evaluate NVIDIA Isaac Sim's performance under integrated computational loads. The parameters observed included CPU usage (%), GPU memory (VRAM) allocation, and frame rate (FPS) stability while rendering the 3D environment and acquiring synthetic sensor data (Intel® RealSense™ D455 camera and virtual IMU). We recorded the synthetic sensor data using the rosbag mechanism to assess the platform's ability to provide a consistent perception dataset.

To provide a more structured overview of the validation stages, Table 4 summarises the testing scenarios and observed parameters.

Table 4. Testing scenarios for the O-SEATER simulation platform

No	Testing Scenario	Data/Parameters observed	Objective
1	Actual O-SEATER movement on a ±50 m path	RPM, /cmd_vel, timestamp	Obtain actual operational reference data and test real-time transmission.
2	Rosbag data playback on the virtual model	Left/right wheel RPM, timestamp	Reproduce physical O-SEATER movement conditions into the simulation environment.
3	Command accuracy and motion response evaluation	MAE dan RMSE pada /cmd_vel dan RPM roda	Evaluate the suitability of actual and virtual movement quantitatively.
4	Visual verification	MAE and RMSE on /cmd_vel and wheel RPM	Support qualitative verification of movement pattern suitability.
5	Wi-Fi connectivity testing	*Latency* (L_{avg}) and *Packet Loss Ratio* (PLR)	Evaluate ROS 2/DDS communication quality and reliability.
6	Safety (failsafe) observation	System behavior when Wi-Fi is disconnected	Ensure the vehicle stops autonomously upon losing the control signal.
7	Computational resource utilization analysis	CPU utilization (%), GPU VRAM, and FPS	Evaluate the simulation platform's computational requirements.
8	Virtual sensor recording	Camera and IMU data	Demonstrate the simulation's capability to generate synthetic sensor data.

We then used data from all testing scenarios in Table 4 to evaluate the correspondence between the actual O-SEATER and the virtual model, both quantitatively (based on motion command parameters, wheel RPM, and timestamps) and qualitatively (through visual verification). In addition, the evaluation included measurements of Wi-Fi communication quality (latency and Packet Loss Ratio), observations of safety behavior (failsafe), host

computational load (CPU utilization, GPU memory, and FPS stability), and the results of virtual sensor recordings. Chapter 3 then presents the testing results and analysis of these parameters to discuss the simulation platform's performance, reliability, and suitability for reproducing the O-SEATER's operational behavior.

3 Results and Discussion

This chapter presents and discusses the results of implementing the developed O-SEATER simulation platform. The research results include building the simulation platform on NVIDIA Isaac Sim, integrating communication using Robot Operating System 2 (ROS 2), implementing Hardware-in-the-Loop (HIL), and implementing rosbag data playback. Each result is then analyzed to evaluate the simulation platform's ability to represent the actual O-SEATER behavior as a medium for testing algorithms and collecting sensor data.

3.1 Simulation Platform Development Results

The O-SEATER simulation platform was successfully built in NVIDIA Isaac Sim by integrating the virtual O-SEATER model and the BRIN KST Samaun Samadikun operational environment into a single simulation environment. We imported the virtual O-SEATER model, converted from the CAD design, into NVIDIA Isaac Sim in Universal Scene Description (USD) format, enabling interaction with the physics engine and other simulation components. In addition, we integrated the operational environment to create a representation of the operational area that closely resembles real-world conditions. We modelled the operational environment in Blender. Figure 12 shows the results of developing this simulation platform.



Figure 12. O-SEATER simulation platform in the operational environment of BRIN KST Samaun Samadikun on NVIDIA Isaac Sim.

The integration results indicate that the O-SEATER virtual model is well configured for the simulation process, including its mechanical structure, physical systems, and ability to interact with the simulation environment. To bring the simulation closer to real conditions than the default simulation environment, this operational environment displays a graphical representation of the testing area. Developing this virtual platform foundation is a crucial prerequisite before implementing wireless communication protocols and Hardware-in-the-Loop (HIL) testing.

3.2 Results of ROS 2 Integration and Hardware-in-the-Loop Implementation

Once the simulation platform was fully developed, the team integrated communication between the physical O-SEATER and the virtual model using the Robot Operating System 2 (ROS 2) over a Wi-Fi network via the Data Distribution Service (DDS). The results show that this communication architecture enables real-time data exchange between physical devices and NVIDIA Isaac Sim. Control commands from the joystick are published on the /cmd_vel topic and received by both systems. Additionally, data distribution through supporting topics such as /joint_states, /odom, /imu, and /camera/image_raw operates stably, enabling the simultaneous synchronisation of vehicle status and sensor information.

During the Hardware-in-the-Loop (HIL) testing phase, the operator provides manual input via the joystick, which is transmitted as linear and angular velocity commands. The test results show that both the physical and virtual O-SEATER respond synchronously to these control commands, with no communication loss (disconnection) during operation. Figure 13 documents the interface and system response during this HIL testing.



Figure 13. Hardware-in-the-Loop Implementation (a) the operator controls the O-SEATER using a joystick, (b) results of ROS 2 communication integration between the actual and virtual O-SEATER, and (c) the virtual O-SEATER in the NVIDIA Isaac Sim simulation environment responding to control commands.

The successful implementation of this HIL demonstrates that the ROS 2-based communication protocol effectively bridges physical and virtual systems. For additional visual verification of real-time synchronization between both systems, recordings of this HIL test demonstration are available at: https://bit.ly/Testing_Hardware_in_the_loop. This validation proves the simulation platform is feasible for safely, flexibly, and efficiently testing algorithms before full implementation under actual operational conditions.

3.3 Results of Rosbag Data Playback Implementation

The rosbag playback implementation verifies whether the simulation platform can reproduce the actual operational conditions of O-SEATER from recorded data. In this study, the RPM data generated by the actual O-SEATER is first recorded using rosbag. Subsequently, the data is republished via ROS 2 using a Python script so that it can be used as input for the virtual O-SEATER model in NVIDIA Isaac Sim. Figure 14. Results of the ROS bag data playback implementation.

The test results indicate that the replay process runs stably, with no communication failures between the rosbag-reading Python script, the ROS 2 network, and the NVIDIA Isaac Sim environment. The virtual model receives the wheel angular velocity data sequentially, allowing the virtual O-SEATER to follow the exact trajectory pattern of the physical O-SEATER as recorded. A visual demonstration of the successful motion reproduction through this playback can be viewed directly at the following link: https://bit.ly/RPM_rosbag_data_playback_testing. This capability confirms that the simulation platform can precisely repeat operational testing scenarios, reducing the need to operate the physical vehicle in the field.

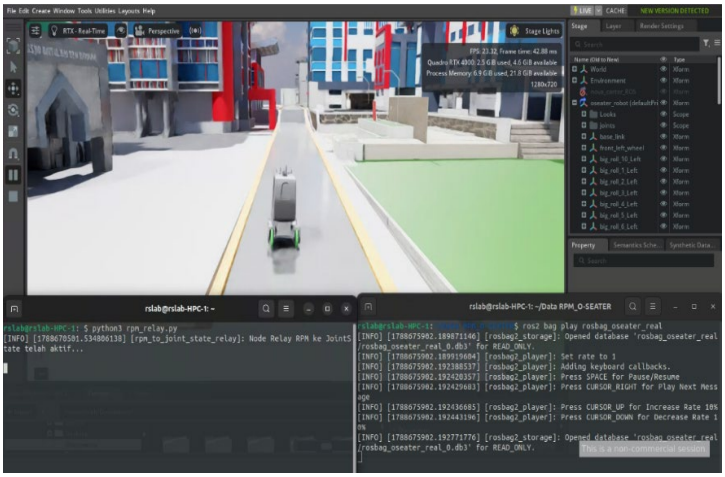


Figure 14. Implementing rosbag data playback on the O-SEATER simulation platform

3.4 Virtual Sensor Recording Results

The data streams from these virtual sensors were recorded again using the rosbag tool to support future development and validation of autonomous algorithms. The outcome of this process was a structured synthetic dataset containing the following core elements.:

1. RGB Image Sequences: Continuous frame captures from the Intel® RealSense™ D455 virtual camera illustrating the visual operational environment.
2. IMU Time-Series Logs: CSV format logs of spatial orientation, linear acceleration, and angular velocity data synchronized with the simulation timestamps.

This replay's success shows that the built simulation platform can generate and replay virtual sensor data synchronised with O-SEATER movement and replicate O-SEATER behaviour using operational data from recordings. This feature eliminates the need for ongoing outdoor data collection by enabling data acquired once on O-SEATER to be used in a variety of testing scenarios within the simulation environment. As a result, the simulation platform can support the development of safer, more effective, and reproducible algorithms. It can also provide a collection of virtual sensor data to support future research on the O-SEATER Digital Twin.

3.5 Analysis of the Suitability of Actual and Virtual Responses

We conducted a quantitative evaluation to assess the accuracy of movement responses between the actual O-SEATER and the virtual model in the simulation environment. This analysis comprises two main parts: evaluating speed control commands (/cmd_vel) during HIL testing and assessing mechanical responses, specifically wheel rotation speed (RPM), during playback. We calculated the magnitude of deviations using the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics on aligned time series. We also evaluated temporal synchronisation (time) between the physical system and the simulation using the Real-Time Factor (RTF) metric.

3.5.1 HIL Control Command Compliance Evaluation

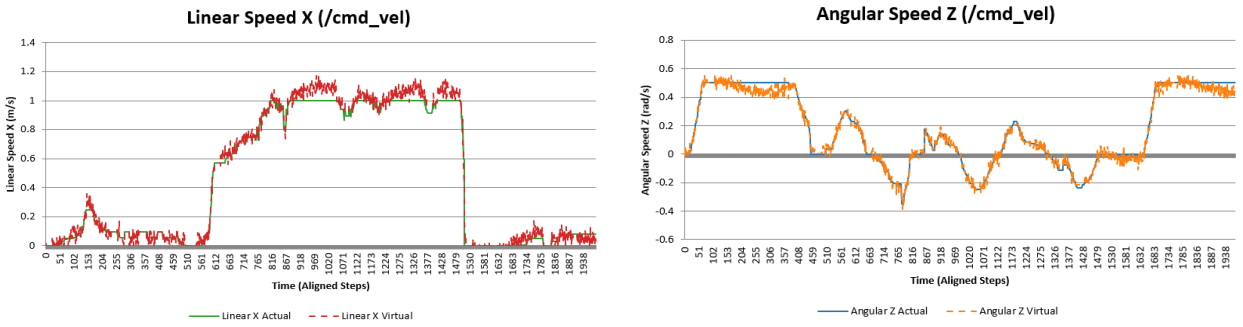
Table 5 and Figure 15 present a conformance evaluation of the virtual model against the real O-SEATER for speed-control instructions (/cmd_vel) using the Dynamic Time Warping (DTW) algorithm. For the X linear velocity component, the actual average is 0.4496 m/s, the virtual average is 0.4682 m/s, the MAE is 0.0299 m/s, and the RMSE is 0.0441 m/s. For the Z angular velocity component, the actual and virtual averages are 0.1699 rad/s and 0.1613 rad/s, respectively, with an MAE of 0.0224 rad/s and an RMSE of 0.0317 rad/s.

Table 5. Comparison of values and error evaluation on control commands (/cmd_vel)

Velocity Component	Actual Average	Virtual Average	MAE	RMSE
Linear Velocity X (m/s)	0.4496	0.4682	0.0299	0.0441
Angular Velocity Z (rad/s)	0.1699	0.1613	0.0224	0.0317

Figure 15 shows the time series of the two speed components. The speed-command pattern in the virtual system is similar to that of the actual system, with both systems receiving a series of acceleration, constant,

and braking commands in the same order. The time series shows that the virtual system takes 103.0967 seconds to execute the same series of commands, whereas the real system takes 56.8928 seconds. Using Equation (6), we obtain a Real Time Factor (RTF) of 0.5518, meaning the simulation runs 1.81 times slower than real time. To compensate mathematically for this temporal deviation caused by computational load and network transmission latency, we used the DTW algorithm to align the time series before computing the final error metrics shown in Table 5. The small residual deviations reflect signal-fidelity degradation due to the 15% packet loss and frame-rate drop during the HIL simulation.



(a) (b)

Figure 15. Comparison graph of command speed time series responses in HIL testing between the actual and virtual systems: (a) linear speed X, and (b) angular speed Z.

This finding indicates that although both the actual and virtual O-SEATER respond consistently to control command patterns, their synchronisation in the real-time domain has not yet been fully achieved. This temporal deviation likely results from a combination of data transmission latency over the Wi-Fi network and the computational load of NVIDIA Isaac Sim, which processes physics simulation and rendering simultaneously, as discussed further in Subsection 3.6.

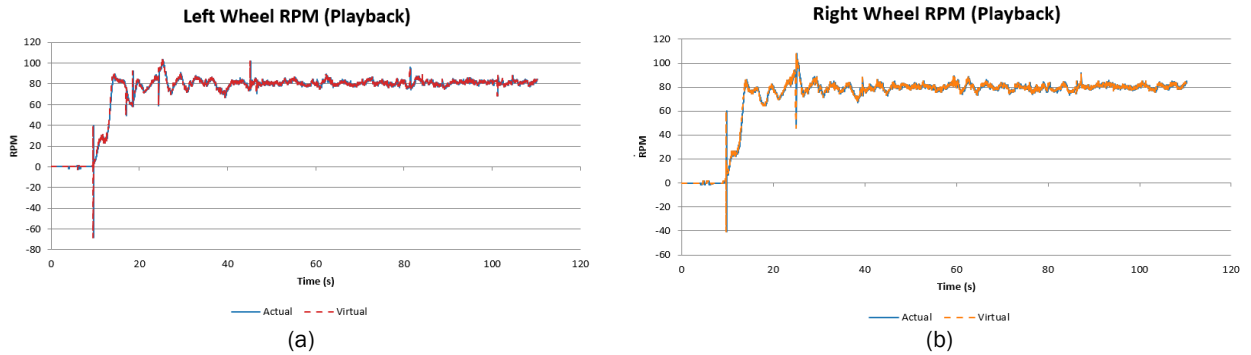
3.5.2 Evaluation of Wheel Turn Response Compatibility

Table 6 and Figure 16 present the evaluation of wheel rotation speed (RPM) response conformity during playback. Based on Table 6, Wheel 1 (Left) shows identical actual and virtual average values of 71.5911 RPM, with MAE and RMSE both equal to 0.0000. Similar results were obtained for Wheel 2 (Right), with actual and virtual average values of 70.8668 RPM and MAE and RMSE of 0.0000.

Table 6. Comparison of values and error evaluation for wheel rotational speed (RPM) response

Velocity component	Actual average	Virtual average	MAE	RMSE
Wheel 1 (Left)	71.5911	71.5911	0.0000	0.0000
Wheel 2 (Right)	70.8668	70.8668	0.0000	0.0000

The time-series curves of the left and right wheel rotational speeds in the actual and virtual systems overlap throughout the testing period. The total actual duration was recorded as 110.2195 seconds, while the virtual duration was 110.1136 seconds. Based on Equation (6), the RTF value is 1.0010, close to the ideal value of 1.0, indicating that the playback process runs almost perfectly synchronised with real time (difference <0.1%).



(a) (b)

Figure 16. Comparison time series graph of wheel rotational speed (RPM) response during playback testing between actual and virtual systems: (a) left wheel, and (b) right wheel.

A summary of the Real-Time Factor (RTF) calculation results for both test scenarios is presented in Table 7.

Table 7. Real-Time Factor (RTF) calculation results

Scenario	Actual duration	Virtual duration	RTF (Actual/Virtual)	Interpretation
cmd_vel (real-time control, HIL)/	56.8928 s (n=1636)	103.0967 s (n=1636)	0.5518	Simulation runs 1.81 slower than real-time
RPM (playback)	110.2195 s (n=5512)	110.1136 s (n=5512)	1.0010	Playback runs almost exactly in real time (difference <0.1%)

These results confirm that the playback method can accurately reproduce the mechanical responses of the virtual model actuators, both in value (MAE and RMSE are zero) and in timing (RTF is close to one). The large difference in RTF between the HIL control scenario (0.5518) and playback (1.0010) indicates that the synchronisation deviations in Subsection 3.5.1 are driven more by interactive computational load (real-time control, two-way data transmission, and physics processing) than by the virtual model's limitations.

3.6 Use of Resources and Communication Observation

The evaluation of computational resource usage and communication network performance is conducted to analyze the factors affecting the performance of the simulation platform, particularly concerning the real-time factor deviations found in the Hardware-in-the-Loop (HIL) testing scenario.

3.6.1 Evaluation of Computing Resource Usage

The computational load on the host computer while running the NVIDIA Isaac Sim environment was observed through parameters such as frame rate (FPS), graphics memory usage (GPU VRAM), and system memory usage (RAM). The system experienced significant workload fluctuations between stable conditions (such as playback) and peak-load conditions (HIL testing with active sensors), as summarized in Table 8.

Table 8. Use of computing resources in NVIDIA Isaac Sim

Metrics/Parameter	Stable condition (Playback)	Peak load condition (HIL / sensors active)
Rendering Resolution	1280 x 720	1280 x 720
GPU VRAM Utilization	2.5 GiB	2.7 GiB
RAM Utilization (Process Memory)	10.4 GiB	27.1 GiB
Frame Rate (FPS)	71.32 FPS	20.80 FPS
Frame Time	14.02 ms	48.08 ms

Based on Table 8, memory allocation on the NVIDIA Quadro RTX 4000 GPU remains stable between 2.5 and 2.7 GiB, indicating that 3D graphics rendering is not the primary limiting factor on this platform. However, system memory (RAM) usage surges from 10.4 GiB to 27.1 GiB during peak load. This critical condition leaves little available memory (around 1.6 GiB), which directly causes a drastic performance decline: the Frame Rate drops from 71.32 FPS to 20.80 FPS, and frame processing time increases from 14.02 ms to 48.08 ms.

This surge in memory load and drop in FPS confirms the anomaly findings in the previous subsection. The massive computational demands of running the physics engine, processing virtual sensor data streams in real-time, and simultaneously handling two-way ROS 2/DDS communication have placed a heavy burden on the host computer system's memory.

3.6.2 Wi-Fi Network Communication Evaluation

In addition to computational load, we evaluated the reliability of the ROS 2 communication architecture based on Data Distribution Service (DDS) over a Wi-Fi network to assess the stability of control command transmission during the HIL scenarios. Testing involved transmitting 300 data packets (ICMP ping) between the simulation host computer and the actual O-SEATER in the operational area of BRIN KST Samaun Samadikun. Table 9 summarizes the network testing results.

Table 9. Results of communication network quality testing

Scenario	Value / Result
Total transmitted packets	300 packets
Total received packets	255 packets
Packet loss ratio	15%
Minimum latency	42.11 ms
Average latency	214.63 ms
Maximum latency	521.31 ms

In addition to the average value, Figure 17 visualizes transmission stability (jitter) through a latency time-series graph.

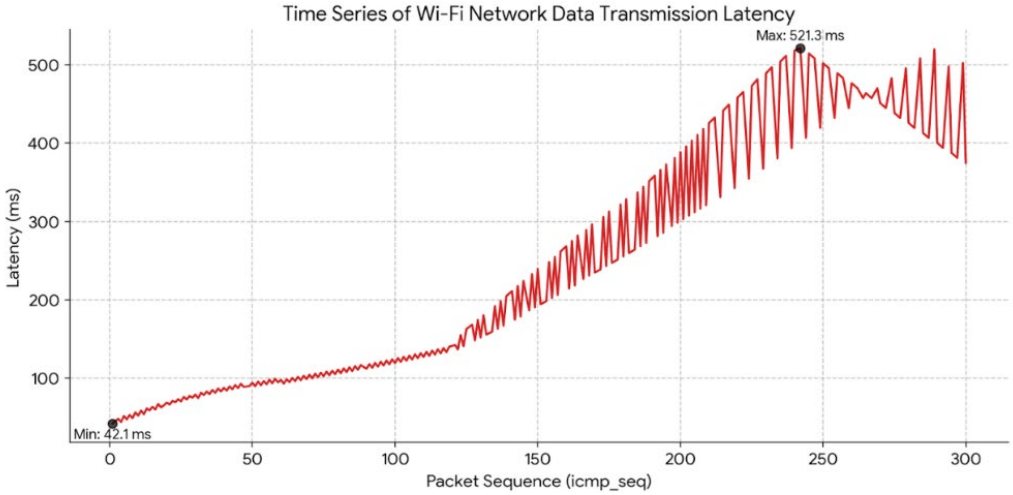


Figure 17. Time series graph of Wi-Fi network data transmission latency during HIL testing

Based on Table 9 and the calculations using Equation (7), the network's average latency L_{avg} The fluctuation range (jitter) in Figure 17 looks really wide, periodically spiking from a minimum of 42.11 ms up to a peak of 521.31 ms. Furthermore, calculations using Equation (6) confirm a fairly significant packet loss rate (Packet Loss Ratio) of 15% ($P_{tx} = 300, P_{rx} = 255$).

The average latency of 200 ms, along with a high packet-loss percentage, indicates interference or signal degradation in the wireless network in the outdoor test area. This finding explains the earlier observed timing anomaly (Real-Time Factor = 0.5518) from a technical standpoint. Delays in command packet arrivals in the actual system, plus some packets lost in the air, force the ROS 2 control system to keep adjusting (buffering), which slows the overall motion synchronization process between the real and virtual O-SEATER compared to real time.

4 Conclusion

This study successfully developed the O-SEATER simulation platform within the BRIN KST Samaun Samadikun virtual environment, using NVIDIA Isaac Sim integrated with ROS 2. Quantitative evaluation results show that the platform effectively enables two-way communication between the physical and virtual systems. In the playback scenario, the platform accurately simulates real operating conditions (MAE and RMSE are 0.0000) and runs in real time (RTF = 1.0010). The Hardware-in-the-Loop (HIL) case was 1.81 times slower (RTF = 0.5518) due to system memory (RAM) load spikes of 27.1 GiB and high latency in outdoor Wi-Fi network transmissions (Average Latency = 214.63 ms; Packet Loss = 15%). However, the platform exhibited very accurate tracking responses, which, when combined with the Dynamic Time Warping (DTW) algorithm, compensated for the time deviation caused by latency, yielding a linear velocity MAE of 0.0299 m/s and an angular velocity MAE of 0.0224 rad/s. The main drawback of this platform remains high communication latency, suggesting the need for further work to improve the communication architecture. Additionally, the platform can simultaneously generate and record synthetic sensor data (IMU and Intel® RealSense™ D455 camera). Finally, the simulation serves as a functional medium for algorithm testing and as a basic step towards the future development of the O-SEATER Digital Twin system.

Acknowledgement

The authors express their gratitude to the Badan Riset dan Inovasi Nasional (BRIN), particularly the Intelligent Mechatronics Research Center and the Robotics and Intelligent Systems Research Group, for their support, guidance, and research facilities, which have made this research possible. The authors also thank all the researchers and parties who have contributed input throughout the research process.

Nomenclature

$d(w_k)$	=	distance between the aligned elements of X and Y at the k-th point
$DTW(X,Y)$	=	minimum cumulative distance between actual sequence X and virtual sequence Y
K	=	total length or total number of steps in the warping path
L	=	distance between drive wheels (m)
L_{avg}	=	average network latency (ms)
$L_{\max}$	=	maximum network latency (ms)
$L_{\min}$	=	minimum network latency (ms)
N	=	number of samples / analyzed data pairs
P_{rx}	=	total successfully received network data packets
P_{tx}	=	total transmitted network data packets
r	=	vehicle wheel radius (m)
t_{actual}	=	total actual system data recording duration (s)
$t_{rx,i}$	=	reception time of the i-th network packet
$t_{tx,i}$	=	transmission time of the i-th network packet
t_{virtual}	=	total virtual system data recording duration (s)
v	=	linear velocity of the vehicle (m/s)
W	=	warping path sequence that aligns X and Y
w_k		the k-th element in the warping path
X	=	actual time-series data sequence
Y	=	virtual (simulated) time-series data sequence
y_i	=	i-th measured value from the actual system
$\hat{y}_i$	=	i-th measured value from the virtual system (simulation)
ω	=	angular velocity of the vehicle (rad/s)
ω_L	=	Angular velocity of the left wheel (rad/s)
ω_R	=	Angular velocity of the right wheel (rad/s)

References

[1] J. Guo, M. Bilal, Y. Qiu, C. Qian, X. Xu, and K.-K. R. Choo, “Survey on digital twins for Internet of Vehicles: Fundamentals, challenges, and opportunities,” *Digital Communications and Networks*, vol. 10, no. 2, pp. 237–247, Apr. 2024. https://doi.org/10.1016/j.dcan.2022.05.023

[2] X. Gu et al., “Digital twin technology for intelligent vehicles and transportation systems: A survey on applications, challenges and future directions,” *IEEE Communications Surveys & Tutorials*, vol. 28, pp. 3235–3271, 2026. https://doi.org/10.1109/COMST.2025.3581152

[3] A. Fuller, Z. Fan, C. Day, and C. Barlow, “Digital twin: Enabling technologies, challenges and open research,” *IEEE Access*, vol. 8, pp. 108952–108971, 2020. https://doi.org/10.1109/ACCESS.2020.2998358

[4] National Research and Innovation Agency (BRIN), “BRIN develops the SEATER autonomous vehicle that can use an application,” *BRIN News*. [Online]. Available: https://brin.go.id/brin-kembangkan-kendaraan-otonom-seater-yang-bisa-pakai-aplikasi. Accessed: Jul. 27, 2026

[5] S. Dharmaraj, “Indonesia: Sustainable, accessible and autonomous urban mobility,” *OpenGov Asia*, Nov. 11, 2024. [Online]. Available: https://opengovasia.com/indonesia-sustainable-accessible-and-autonomous-urban-mobility/. Accessed: Jul. 27, 2026

[6] “Autonomous electric vehicles in restricted areas are being developed,” *RRI.co.id*, Nov. 2024. [Online]. Available: https://rri.co.id/iptek/1111984/about.html. Accessed: Jul. 27, 2026

[7] S. M. Kargar, B. Yordanov, C. Harvey, and A. Asadipour, “Emerging trends in realistic robotic simulations: A comprehensive systematic literature review,” *IEEE Access*, vol. 12, pp. 191264–191287, 2024. https://doi.org/10.1109/ACCESS.2024.3404881

[8] H. Alghodhaifi and S. Lakshmanan, “Autonomous vehicle evaluation: A comprehensive survey on modeling and simulation approaches,” *IEEE Access*, vol. 9, pp. 151531–151566, 2021. https://doi.org/10.1109/ACCESS.2021.3125620

[9] S. D. Pendleton et al., “Perception, planning, control, and coordination for autonomous vehicles,” *Machines*, vol. 5, no. 1, Art. no. 6, 2017. doi: https://doi.org/10.3390/machines5010006

[10] J. I. Da Silva Filho, C. L. M. Fernandes, R. S. d. Silveira, P. M. Gomes, S. L. d. C. Matos, L. d. E. Santo, V. C. Nunes, H. M. Côrtes, W. A. C. Lopes, M. C. Mario, D. V. Garcia, C. R. Torres, J. M. Abe, and G. Lambert-Torres, "Process of learning from demonstration with paraconsistent artificial neural cells for application in linear Cartesian robots," *Robotics*, vol. 12, no. 3, Art. no. 69, 2023. https://doi.org/10.3390/robotics12030069

[11] A. Al-Batati, A. Koubaa, K. Gabr, M. Abdelkader, and H. Aloqaily, "ROS 2 in a nutshell: A survey," *ACM Computing Surveys*, 2025. https://doi.org/10.1145/3815113

[12] A. Bonci, F. Gaudeni, M. C. Giannini, and S. Longhi, "Robot Operating System 2 (ROS2)-based frameworks for increasing robot autonomy: A survey," *Applied Sciences*, vol. 13, no. 23, Art. no. 12796, 2023. https://doi.org/10.3390/app132312796

[13] M. A. L. Sarker, M. O. F. Sarker, and D. S. Han, "A survey on digital twin-assisted intelligent vehicle localization," *ICT Express*, vol. 11, no. 5, pp. 981–992, 2025. https://doi.org/10.1016/j.icte.2025.08.008

[14] G. S. Lutz, S. Kubica, T. Peuschke-Bischof, and C. M. Travieso-González, "Digital driving twins for scaled ADAS algorithm development: A systematic review and design proposal for co-simulation architectures, indoor localization methods, and ground truth strategies," *Applied Sciences*, vol. 16, no. 14, Art. no. 7261, 2026. https://doi.org/10.3390/app16147261

[15] C. Campolo et al., "An edge-based digital twin framework for connected and autonomous vehicles: Design and evaluation," *IEEE Access*, vol. 12, pp. 46290–46303, 2024. https://doi.org/10.1109/ACCESS.2024.3382001

[16] Y. Hu, M. Wu, J. Kang, and R. Yu, "D-Tracking: Digital twin-enabled trajectory tracking system of autonomous vehicles," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 11, pp. 16457–16469, Nov. 2024. https://doi.org/10.1109/TVT.2024.3414410

[17] S. Almeaibed, S. Al-Rubaye, A. Tsourdos, and N. P. Avdelidis, "Digital twin analysis to promote safety and security in autonomous vehicles," *IEEE Communications Standards Magazine*, vol. 5, no. 1, pp. 40–46, Mar. 2021. https://doi.org/10.1109/MCOMSTD.011.2100004.

[18] Ö. Kaya, L. Bacchiani, A. Melis, R. Presta, C.-T. Lam, G. Pau, and R. Girau, "Vehicle, driver, and road digital twins for connected mobility: A critical review and unified conceptual framework," *Future Internet*, vol. 18, no. 6, Art. no. 277, 2026. https://doi.org/10.3390/fi18060277

[19] X. Hu, S. Li, T. Huang, B. Tang, R. Huai, and L. Chen, "How simulation helps autonomous driving: A survey of Sim2Real, digital twins, and parallel intelligence," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 593–612, Jan. 2024. doi: https://doi.org/10.1109/TIV.2023.3312777

[20] S. Cardozo and V. Hlavác, "CARLA hardware-in-the-loop simulation environment for autonomous driving," *SAE International Journal of Connected and Automated Vehicles*, vol. 9, no. 2, pp. 289–309, 2026. https://doi.org/10.4271/12-09-02-0015

[21] T.-M. Kim, S.-B. Lee, C.-Y. Lee, W.-P. Yu, and S.-H. Song, "A layered information model framework for standardizing digital twin simulation," *Journal of Institute of Control, Robotics and Systems*, vol. 31, no. 12, pp. 1410–1417, 2025. https://doi.org/10.5302/J.ICROS.2025.25.0232

[22] H. Sekkat, O. Moutik, B. El Kari, Y. Chaibi, T. Ait Tchakoucht, and A. El Hilali Alaoui, "Beyond simulation: Unlocking the frontiers of humanoid robot capability and intelligence with Pepper's open-source digital twin," *Heliyon*, vol. 10, no. 14, Art. no. e34456, Jul. 2024. https://doi.org/10.1016/j.heliyon.2024.e34456

[23] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD Workshop)*, 1994, pp. 359-370. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/3000850.3000887

[24] Z. Gao, T. Yu, T. Sun, and H. Zhao, "Data Filtering Method for Intelligent Vehicle Shared Autonomy Based on a Dynamic Time Warping Algorithm," *Sensors*, vol. 22, no. 23, Art. no. 9436, Dec. 2022. https://doi.org/10.3390/s22239436

[25] NVIDIA, "ROS 2 Publish Real Time Factor (RTF)," *NVIDIA Isaac Sim Documentation*. Available: https://docs.isaacsim.omniverse.nvidia.com. Accessed: Sep. 5, 2026