# A Simulation Framework for V2X Autonomous Vehicles Based on CyberPhysical Vehicle Systems Technologies

Ming-Jui Ho and Tsung-Ying Sun
*Department of Electrical Engineering, National Dong Hwa University, Hualien, Taiwan, ROC*

Shiow-yang Wu*
*Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan, ROC*

Corresponding author. Tel.: +886-3-8905020; Fax: +886-3-8900168

E-mail: showyang@gms.ndhu.edu.tw

ABSTRACT: Autonomous vehicle technology has been advancing rapidly. Comprehensive testing is essential before road deployment. However, no test can cover every possible situation. Thus, a good simulation system is necessary. Most existing systems are restricted to limited aspects that fail to correspond with reality in complex and volatile environments. Cyber-Physical Vehicle System (CPVS) technology enables the simulation systems to integrate more closely with actual vehicles. Nevertheless, the behavior analysis and decision-making capabilities are still far from perfect. The proposed framework is based on CPVS that integrates physical sensors and mechanical components on vehicle with a ROS2-based cyber-system receiving messages through WAN for cooperative optimization. The messages are conformed to IEEE and SAE standards integrated with RealSense, Lidar, IMU sensor data and mechanical components information to display the vehicle status with Unity graphics in two modes: real-time scene rendering and vehicle monitoring mode for vehicle monitoring, and scene restoring mode for error analysis. We emphasize the cyber-physical interaction in which physical system serves feedback data for model building and scene construction while cyber system tests the controller with various simulated scenes. A scene editor subsystem is provided to visually layout static/moving objects on a virtual plane to construct any desired scene and trigger Unity to generate simulated sensor data for evaluating the reactions of simulated or physical vehicle. The CPVS is used in the design of an autonomous ground vehicle (AGV) simulated and tested using serval situations layout with the scene editor. Compared with testing the AGV in real environments, the result is highly satisfactory.

***Keywords—Cyber-Physical Vehicle System, V2X, autonomous vehicles, digital simulation, Robot Operate System***

## 1 INTRODUCTION

Computer simulations are indispensable in the design and analysis of complex dynamic systems, especially for the testing and validation of structural and functional behavior at different levels of details (L. Žlajpah, 2008). It is even more important for autonomous vehicle design since not only the vehicle status but also all possible interactions with surrounding objects must be taken into consideration. Most existing systems are restricted to limited aspects such as lane keeping, driving patterns, electronic systems, fleet control, etc. but fail to correspond with reality in complex and volatile environments (Ş. Y. Gelbal et al. 2017, P. Nilsson et al. 2017, Q. Li et al. 2012, A. Marjovi et al. 2015). We proposed a CPVS-based simulation framework for V2X and driving control in autonomous vehicles. The framework is built on top of ROS2 that integrates physical system, navigation control and sensor data on the vehicle with cyber system on Unity for scene generation, test data provisioning and vehicle status visualization. The simulation can be conducted in two modes: **real-time scene rendering and vehicle monitoring mode** for online simulation and visualization, as well as **scene restoring mode** for error analysis. A scene editor subsystem is provided to layout any desired scene and trigger the generation of simulated data for the testing and evaluation of simulated or physical vehicles. With special emphasis on cyber-physical interactions, the framework facilitates tight correspondence between simulated environment and reality which is the key to effective design, testing, and co-optimization.

The paper is organized as follows. Section 2 presents existing CPVS and related systems for autonomous vehicles, especially ROS2 and Unity-based systems. Section 3 describes the architectural design and system components of our CPVS-based simulation system. Section 4 continues with detail analysis of the key characteristics and main advantages. In Section 5, we discuss the implementation and evaluation of our framework. Section 6 concludes the paper.

## 2 LITERATURE REVIEW

CPS facilitates tight integration of virtual and physical systems on computing, communication, and

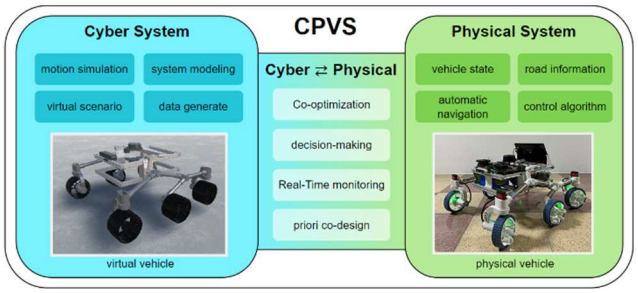


Figure 1. Cyber-Physical Vehicle System concepts.

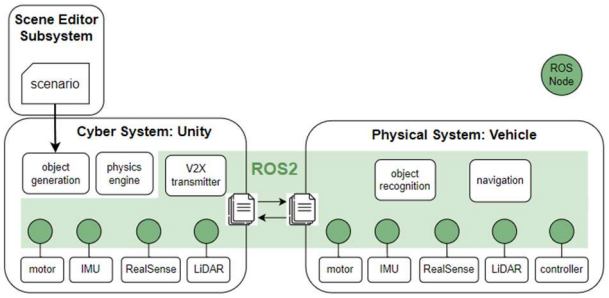


Figure 2. The proposed CPVS system architecture.

control to achieve stability, performance, reliability, robustness, and efficiency on many application domains (H. Wang, M. Xu et al., 2018). CPVS is the extension of CPS on ground, aerial, and maritime vehicles. It is more challenging due to the need to deal with not only surrounding objects but also complex issues such as weather, regulations and social interactions. The size and resource limits on vehicles make everything trickier. On-road deployment brings even more uncertainty. With tight integration of all modules and components between cyber and physical systems, CPVS facilitates close coordination and co-optimization in both design and operation to meet the demanding requirements of autonomy, adaptability, reliability, effectiveness, robustness and safety as illustrated in Figure 1 (J. M. Bradley and E. M. Atkins, 2015).

In our framework, ROS2 is chosen as the kernel of the physical system. ROS2 is a meta operating system and a set of tools for robots. It is highly versatile and very popular among roboticists ever since its first release (S. Macenski, T. Foote, B. Gerkey et al., 2022). With its concise yet comprehensive considerations of embedded systems, diverse networks, real-time processing, security and product readiness, it is now widely used in the autonomous systems such as robots and AGVs. The cross-platform support of ROS2 allows us to integrate seamlessly with Unity for the cyber system part of our CPVS framework. However, our framework is flexible enough to adopt other 3D simulation tools such as Gazebo and CoppelizSim (formely V-Rep). Most existing systems that integrate Unity and ROS2 are restricted to limited functionalities or one-way communication (E. Sita, C. M. Horváth, T. Thomessen et al,. 2017; A. Hussein, F. García, and C. Olaverri-Monreal, 2018). Our CPVS-based framework achieves tight integration of cyber and physical systems for the design, simulation, testing and optimization of autonomous vehicles.

The scope of AGV control has been extended from vehicle control to V2V (communicating with surrounding vehicles) to V2X (interacting with all objects on road such as traffic signals and pedestrians). With increasing need of V2X communication, the interoperability of vehicles and objects of different nations, brands and models must be established. International standards such as IEEE 802.11p and SEA J2735 are becoming more mature and popular (SAE, 2022). Our system abides by the international standards above to cover V2X.

## 3 SYSTEM DESCRIPTION

### A. *Framework*

The CPVS-based framework we proposed and implemented is built on top of ROS2 and Unity as illustrated in Figure 2. ROS2 is the operation platform of the physical system to support navigation control, object recognition, and all sensor modules. The cyber system is modeled and visualized by Unity. A common ROS2 substrate is used for the connection and message exchange between the two. A scene editor subsystem is provided to design test scenarios and trigger the object generator in Unity to construct corresponding simulated scenes for testing and evaluation. More details in the following sections.

#### *1) ROS2*

The physical system (vehicle) is composed of multiple ROS2 nodes to manage underlying mechanical components as well as different types of on-vehicle sensors for camera image capturing, IMU data collection and filtering, voltage measurement, wheel speed detection/control, etc. The information feedback from the physical system is crucial. After applying the control commands, the ROS2 nodes can capture vehicle status information such as wheel speed, turning angle, acceleration and other data for the cyber system to reconstruct vehicle state dynamically on the simulator GUI. This can be used for system modeling and monitoring to facilitate system design, decision making, testing and co-optimization.

#### *2) Unity Game Engine*

The cyber system is driven by the Unity game engine. Upon receiving the data captured and transmitted from ROS2, the Unity engine compares and adjusts the cyber vehicle state to truthfully represent the physical vehicle so that the simulation and analysis results can be more reliable and useful.

For more comprehensive simulation, our system can reconstruct not only the dynamic vehicle state such as coordinate, orientation, acceleration and trajectory, but also the interactions of the vehicle with surrounding objects. This is achieved by a set of virtual sensors provided by our CPVS system. For example, a virtual RealSense camera can simulate and render the image and depth data similar to a real camera. A virtual LiDAR can reflect the distance between objects in the simulated scene and render it with PhysicsRaycast in Unity.

The flexibility and versatility of Unity on scene construction enable us to achieve high diversity in simulation. Simple scenes can be constructed with built-in obstacles in Unity for initial testing. For complex scenes in real world, we can import 3D models built by tools such as AutoCAD, Blander, etc. as well as the rich 3D model resources provided by the open-source community. With model importing function, our system can potentially simulate any real world scene to significantly extend the coverage of testing with very low cost.

*3) ROS2 and UNITY Connection*

The connection between cyber and physical systems is done through message exchange between ROS2 and Unity. The message header contains information such as ID, timestamp, data format, encoding, etc. Control and sensing data such as motor control commands, data from various sensors, and calculated output are transmitted in the message payload. The message transmission is through the LAN using the official Unity-Robotics-Hub (Unity Technologies, 2022).

*B. V2X*

Compared with the information collected by sensors, the data provided by V2X is richer and more accurate, which is of great value to the prediction and decision making in AGV. V2X can also provide field of vision of obscured objects and blind spots through intermediate vehicles. Most existing V2X simulators are aimed at the testing and verification of communication protocols. To test the operation of the AGV control system in V2X environment, existing test sites are very rare and expensive, and almost all of them are closed sites. As an important feature of our framework, we introduce V2X into our system to test the AGV control responses on V2X events. Our message format is defined based on the Basic Safety Message format released by SEA, without the fields that are not used in our system, such as Brake Applied Status as presented in Table 1.

TABLE 1. V2X Message Packet Definition

| Data Element | Format | unit |
|---|---|---|
| Vehicle ID | Text | |
| Position | Vector3 | m |
| Position accuracy | Float | m |
| Liner Speed | Vector3 | m/sec |
| Liner acceleration | Float | m/sec |
| Heading | Float | rad |
| Turn angle | Float | rad |
| Turn rate | Float | rad/sec |
| Control status | Text | |
| Vehicle size | Vector3 | m |

*C. Real-Time Scenes Rendering and Vehicle Monitoring Mode*

This is one of the two primary operation modes provided by the cyber system. When Unity receives the physical- system data provided by ROS2, it restores the current state of the vehicle and surrounding scenes in the simulator. The speed, acceleration, and attitude of the vehicle are determined by the IMU, and the wheel speed, battery voltage, system power, temperature, etc. can be obtained from corresponding sensors. Driving environment data, including other vehicles, pedestrians, traffic signs, obstacles, etc., can be detected not only by cameras and LiDAR, but also from V2X information which allow us to create more accurate scenes.

It is essential to correctly model and simulate the interactions of the AGV control system with all other objects in the driving environment so as to make proper decisions. Under this mode, the cyber system can present the users with detailed physical data of the vehicle as well as all interactions with other objects in real time. In addition to facilitating remote real-time monitoring, a great opportunity for the optimization of the AGV decision-making system is at hand.

*D. Scene Restoring Mode*

The second primary mode of the cyber system is to record and store essential data during real-time simulation and rendering so that critical scenarios can be reconstructed repeatedly for controller adjustment and system optimization. The stored information can also be used on other controllers for the purposes of comparison and evaluation. System optimization is not limited to current controller on the vehicle. It can be applied to test the controllers in all stages of system design, even on the optimization of mechanical structure.

*E. Scene Editor Subsystem*

To achieve comprehensive testing of the control system, solely relying on the test data recorded by actual vehicles is not enough. The cost of collecting test data for all possible scenarios is prohibitively high. Some experimental vehicles may not even be allowed to test on the road without legal approval.

For this reason, we propose a Scene Editor Subsystem that can design and edit test scenarios. Through this system, a scene containing multiple dynamic and static objects, such as an intersection with cars and pedestrians, can be edited with a

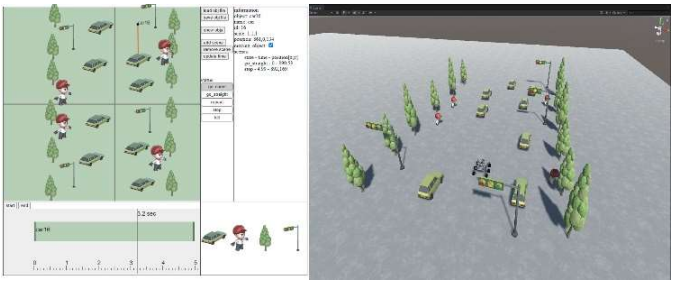

Figure 3. Scene editor subsystem (left) and scenes construction in Unity (right).

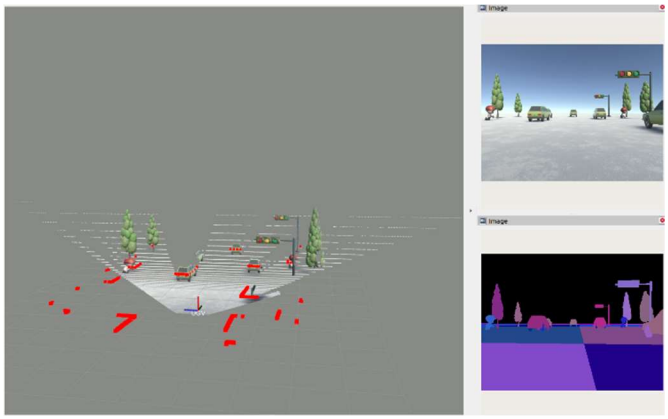

Figure 4. ROS2 visualization. Pointcloud (left), camera (top right) and object ID (bottom right).

concise and friendly user interface to generate a corresponding script. Unity engine can simulate the test scenario through running the script and analyze if the simulated AGV meets the expected control results.

As illustrated in Figure 3, after designing the layout and arranging the movement of all objects in the scene editor subsystem, the generated script can be executed by Unity to create the scene dynamically. They provide the driving environment data for the simulated vehicle on ROS2 for further simulation and control. Simulation results can be visualized in different ways as demonstrated in Figure 4.

## 4 SYSTEM CHARACTERISTICS

### *A. Control System Evaluation*

When testing on the simulator, the focus is often on whether it can truthfully reflect the actual running state and provide reference information for optimization. During the testing of controller, if the thresholds of proper responses can be determined according to the test data, then an evaluation function can be devised to measure the controller performance. To the best of our knowledge, neither the current simulation software nor the test equipment for self-driving cars can offer this function.

The advantage of CPVS is that the physical and cyber systems are closely integrated, and various information in both systems can be exchanged comprehensively. For example, if we want to test the decision-making of an AGV control system in response to a sudden obstacle in V2X environment. Existing simulation software can only simulate roads with different types of traffic flow or generate V2X information independently on the road. The proposed CPVS architecture can record the sensing time of the control system, the control commands output, vehicle movement data, etc. during the execution of the script. Upon judging whether the controller achieves the designated task, it can also evaluate the quality of the decision process. For the above mentioned case, we can evaluate if the vehicle always stops at a certain distance before the obstacle. Other metrics such as the time point when the object is recognized, the time and cost of calculation, and the vehicle deceleration behavior, etc. also need to be analyzed to set proper thresholds. By obtaining these detailed information, the system components can be optimized separately and as a whole.

### *B. Cooperative Optimization*

The main purposes of using a traditional simulator is to investigate and evaluate control system behavior for controller optimization through simulated system operation and data analysis. In our CPVS architecture, however, not only the control system and mechanical structure on the physical system can be optimized, but also the restoration of the simulated vehicle and the generation of the simulated environment in the cyber system.

Take the development process of an AGV as an example. After setting the vehicle mission, the target driving environment and terrain can be generated in the cyber system. Then the designated vehicle model can be tested and adjusted to meet the requirements. After model adjustment, various simulated sensors can also be configured and installed on the virtual model. With satisfactory simulation results, the actual vehicle can be built based on the successful model in the cyber system. In addition to cost saving, the simulations also ensure a certain level of applicability and reliability of the physical vehicle. After vehicle construction is completed, physical system can provide true data with loaded sensors and compare it with that of simulated vehicle. System identification is used to ensure the consistency between the simulated vehicle with the actual vehicle, so as to match simulation to reality. This makes the subsequent design and optimization of the control system in the simulator more compatible with the actual vehicle. After the deployment of the vehicle loaded with the control system, the operating environment data collected by the vehicle can be used to further optimize the cyber system, such as introducing real-world noise to the sensors and V2X data of the simulator, or making the characteristics of generated scenes more similar to real world environments and so on.

The above is a legitimate development process using our CPVS framework but not the only one. Our system can offer invaluable assistance to many AGV design, testing, and manufacturing tasks. In particular, the physical system and cyber system can be continuously improved at different stages and layers to achieve cooperative optimization.

### C. Decision-Making

In addition to the co-operation of the cyber and physical systems, our system can also generate possible future scenes dynamically based on certain conditions and parameter changes, then predict the decision in advance according to the situation at hand or ahead of time. This allow the system to make more resourceful and reliable decisions.

## 5 EXPERIMENTS

### A. Platform Description

#### 1) Physical System

The physical system includes the physical vehicle with ROS2 kernel for the control system. The controller consists of upper level calculation and control algorithms with lower level motor control and sensor receptors. The ROS2 employed is the Foxy Fitzroy release. NVIDIA Jetson Nano Developer Kit is chosen as the development platform. Each component is modeled by a ROS2 node including each sensor, remote control, navigation, and Unity connector. The sensors include RealSense, 2D LiDAR, IMU, wheel-speed sensor, voltage sensor, and power sensors.

#### 2) Cyber System

The cyber system consists of a Unity 20203.11f kernel on Win10 driven by Intel i7-7700 CPU with 16 GB RAM. The simulator consists of two parts, a simulated vehicle and an object generator.

The simulated vehicle is the 3D model of the target vehicle under simulation. They have the same structure and simulated components with the same set of simulated sensors: a virtual RealSense, virtual 2D LiDAR and virtual IMU.

The object generator generates static and dynamic objects based on the script exported from the Scene Editor Subsystem. The script is an XML file with object definitions, object types, coordinates, size, and the designed scenes along the timeline. Each scene contains all the objects in the scene and where/when/how each object moves. Unity executes the script and renders the objects accordingly to create the scene dynamically.

To simulate V2X environment, each object in the running scene, including pedestrians, vehicles, traffic signals, trees and obstacles, has a corresponding data publisher to publish data according to Table 1.

The simulated vehicle receives the commands from the remote control or autonomous controller on the physical vehicle to determine how to move. When both the simulated vehicle and the physical vehicle receive the commands and act accordingly, the system is in real-time monitoring mode. If only the simulated vehicle is operating without corresponding physical vehicle, it is in scene restoring mode. With various scene editing, restoration and simulation, we can test the vehicle system comprehensively with very low cost.

### B. Validation Use Cases

We design a test scene on the Scene Editor Subsystem and export the script to Unity. The simulated vehicle accepts control commands and sensor data from Unity without corresponding physical vehicle. With cyber only vehicle, we can easily and repeatedly test the controller and modify the code if necessary. The initial controller and corresponding AGV trajectory is depicted as blue line in Figure 5.

To correspond closely with physical vehicle, a calibration process is necessary so that the control properties of both systems can be consistent. In this case, we need to calibrate wheel speed, DC motor response time and the PI control parameters of the servo motor. After calibration, the cyber vehicle and physical vehicle are now in line with each other as illustrated by the orange and green lines in Figure 5.

We now apply the controller on the cyber vehicle to the physical vehicle and arrange a real-world scene which mimics the script scene. The trajectory and control output of the physical vehicle coincide closely with that of the cyber vehicle as illustrated in Figure 6. The controller designed and optimized in the cyber system can be used directly on the physical system without modification or adjustment, which demonstrates the effectiveness of our CPVS-based simulation framework.

The integration of V2X information is an important feature of our framework. As illustrated in Figure 7, upon traversing on a target path (blue line), the on-vehicle LiDAR can only detect visible obstacles within sensing range (the purple circle) and trigger dynamic obstacle avoidance locally (the green line). With V2X information (the orange dots), the control system can recognize obstacles along the target path far ahead of local sensing range to

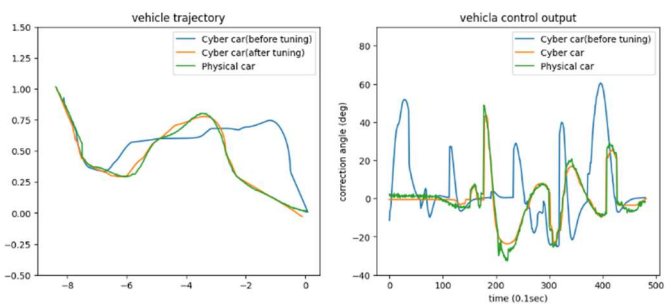


Figure 5. Simulated AGV trajectory and control output.

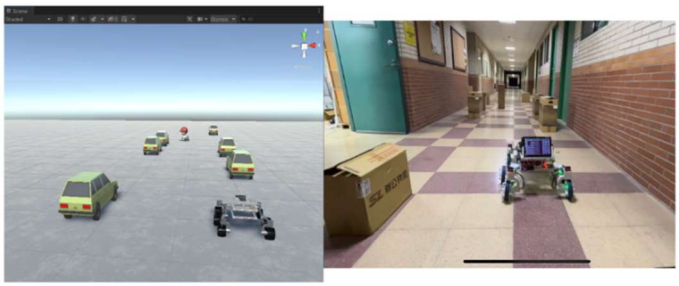

Figure 6. Scenario for demonstrating the close correspondence between cyber and physical vehicle in our system.

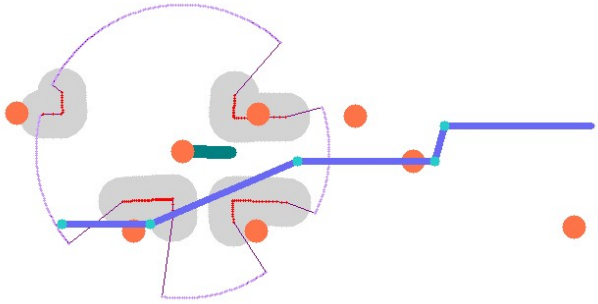

Figure 7. V2X information for the control system to plan ahead.

facilitate global optimal path planning instead of local adjustment.

## 6 CONCLUSIONS

We proposed and implemented a CPVS-based framework for autonomous vehicle development and simulation. The framework integrates ROS2-based physical system with Unity-based cyber system such that the control system developed, simulated and tested on the cyber system can be used directly on the physical system without modification yet still exhibits almost identical behavior. A Scene Editor Subsystem makes the framework even more versatile and useful.

The framework can be extended with more tools for validation and performance evaluation. We plan to devise precise metrics to measure the effectiveness of simulation and the quality of control system.

## 7 REFERENCES


L. Žlajpah, "Simulation in robotics," Mathematics and Computers in Simulation, vol. 79, no. 4, pages 879-897, 2008.

Ş. Y. Gelbal, S. Tamilarasan, M. R. Cantaş, L. Güvenç, and B. Aksun-Güvenç, "A connected and autonomous vehicle hardware-in-the-loop simulator for developing automated driving algorithms," in 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017, pp. 3397-3402.

P. Nilsson, L. Laine, and B. Jacobson, "A Simulator Study Comparing Characteristics of Manual and Automated Driving During Lane Changes of Long Combination Vehicles," IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 9, pages 2514-2524, 2017.

Q. Li, W. Chen, Y. Li, S. Liu, and J. Huang, "Energy management strategy for fuel cell/battery/ultracapacitor hybrid vehicle based on fuzzy logic," International Journal of Electrical Power & Energy Systems, vol. 43, no. 1, pages 514-525, 2012.

A. Marjovi, M. Vasic, J. Lemaitre, and A. Martinoli, "Distributed graph-based convoy control for networked intelligent vehicles," in 2015 IEEE Intelligent Vehicles Symposium (IV), 2015, pp. 138-143.

H. Wang, M. Xu, F. Zhu, Z. Deng, Y. Li, and B. Zhou, "Shadow traffic: A unified model for abnormal traffic behavior simulation," Computers & Graphics, vol. 70, no. pages 235-241, 2018.

J. M. Bradley and E. M. Atkins, "Optimization and Control of Cyber-Physical Vehicle Systems," Sensors, vol. 15, no. 9, pages 23020-23049, 2015.

S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," Science Robotics, vol. 7, no. 66, pages eabm6074, 2022.

E. Sita, C. M. Horváth, T. Thomessen, P. Korondi, and A. G. Pipe, "ROS-Unity3D based system for monitoring of an industrial robotic process," in IEEE/SICE International Symposium on System Integration (SII), 2017, pp. 1047-1052.

A. Hussein, F. García, and C. Olaverri-Monreal, "ROS and Unity Based Framework for Intelligent Vehicles Control and Simulation," in IEEE International Conference on Vehicular Electronics and Safety (ICVES), 2018, pp. 1-6.

Soc. Automotive Eng. J2735 V2X Communications Message Set Dictionary, 2022-11. [Online]. Available: http://www.sae.org

Unity Technologies. (2022). Unity-Robotics-Hub. [Online]. Available: https://github.com/Unity-Technologies/Unity-Robotics-Hub