TRACEN-X: Telemetry Replay and Analysis of CAN Bus and External Navigation Data

TRACEN-X: Telemetry Replay and Analysis of CAN Bus and External Navigation Data

Diego Gasco*, Carlos Mateo Risma Carletti*, Francesco Raviglione†, Marco Rapelli*, Claudio Casetti*

* Department of Control and Computer Engineering, Politecnico di Torino, Italy
† Department of Electronics and Telecommunications, Politecnico di Torino, Italy
E-mail: {diego.gasco, carlos.rismacarletti, francesco.raviglione, marco.rapelli, claudio.casetti}@polito.it

***Abstract*—Connectivity is a key enabler for next-generation autonomous vehicles. Developing and validating Connected and Autonomous Vehicle (CAV) services often requires extensive field testing, especially after initial pre-deployment tests. However, multiple field tests introduce high costs and logistical challenges, with no guarantee of consistent environmental conditions.**

This paper presents TRACEN-X (Telemetry Replay and Analysis of CAN bus and External Navigation data), the first open framework for collecting and reproducing navigation, sensor, and Controller Area Network (CAN) bus data from real-world scenarios in a controlled lab environment. By recreating real-world conditions, TRACEN-X streamlines development and validation, reducing costs and complexity. We validated our framework using data from a connected vehicle with ADAS sensors and a V2X-equipped stroller, demonstrating its potential in combination with a vehicular network simulator and an open ETSI C-ITS stack implementation.

***Index Terms*—V2X, Vehicular networks, CAN bus, GNSS, serial interface, sensors, telemetry**

I. Introduction

Validating Connected and Autonomous Vehicle (CAV) software under real-world conditions remains critical after simulations, but reproducing navigation[1] traces in the lab is challenging. Real-world tests are costly, hard to replicate, and subject to environmental variability. Accurate replay of navigation data, essential for developing V2X services, typically requires expensive GNSS spoofing equipment, limiting practical lab testing. Connected vehicle services depend on multiple inputs, including GNSS-based Position-Velocity-Time (PVT) data, internal sensor and ECU information, and V2X messages. Yet, there is a notable lack of open frameworks capable of aggregating and replaying these inputs offline for controlled development and testing.

To address this gap, we propose TRACEN-X (Telemetry Replay and Analysis of CAN bus and External Navigation data), the first Linux-based comprehensive open-source[2] framework capable of simultaneously collecting and reproducing *(i)* navigation data (from GNSS + inertial sensors) and *(ii)* Controller Area Network (CAN) bus data from a field test environment, all within a single tool. TRACEN-X creates virtual serial and CAN devices, allowing real automotive services to interface as if running on the original vehicle. This enables real-time, accurate trace replay in the lab. In addition to replay, TRACEN-X supports transparent trace analysis, offering insights into collection conditions, message types, and update rates. It can also export traces as log files for use in simulation frameworks like ms-van3t [1].

TRACEN-X has been developed in Python with a minimal number of dependency packages, and validated through field data collection using both a connected vehicle and a "smart stroller", i.e., a stroller equipped with an On-Board Unit (OBU) supporting V2X connectivity [2]. We further demonstrate its potential by integrating it with the ms-van3t simulator and the OScar framework [3], an open ETSI C-ITS implementation designed to run on Linux-based OBUs. In particular, we showcase how TRACEN-X can facilitate the study of standardized message dissemination, allowing researchers to create a real-time virtual field test environment, enabling controlled and repeatable testing scenarios.

The remainder of this paper is organized as follows: Section II reviews related work and existing tools for vehicle data recording, while Section III details the architecture of TRACEN-X. Section IV presents the validation results and experimental findings, and, finally, Section V summarizes the conclusions and outlines directions for future work.

II. Related Works

Accurate data collection and reproduction in lab settings are crucial for CAV development, since field testing is often costly and impractical, making testing in controlled environments essential. To address this, various tools have been developed for data collection and emulation. Table I compares TRACEN-X with the existing solutions in the literature, which essentially fall into three categories: digital twins for vehicle modeling, software tools for recording and replaying navigation data or software tools for recording and replaying CAN data.

Despite existing tools, no open framework is able to simultaneously collect real-world data and provide it to automotive services in a laboratory setting as if running on a real vehicle. TRACEN-X fills this gap by enabling real-time collection and reproduction of navigation, sensor, and CAN data, for CAV development and testing.

[1]Navigation data includes Position-Velocity-Time (PVT) from GNSS and kinematic data from IMUs.

[2]https://github.com/DriveX-devs/TRACEN-X

A. Tools for Vehicular Digital Twin

Digital twins replicate real-world systems in a virtual environment to test and optimize new solutions. In CAV development, they bridge the gap between real-world data and simulation. A relevant example is CarLab [4], an open vehicular data-collection platform designed for seamless API integration. As opposed to CarLab, TRACEN-X focuses on *collecting*, *replaying*, and *analyzing* real-world traces in a controlled lab setting, all within a single tool. Another example is ART/ATK [5], which combines high-fidelity simulation with real-world experiments using the "Chrono" engine but does not capture or replicate real field test data with virtual devices, an aspect TRACEN-X specifically addresses.

B. Tools for Recording and Reproducing Navigation Data

Reliable navigation data replay is essential for in-lab CAV testing. Tools like LabSat [6], [7], GNSS hardware simulators with multi-constellation and multi-frequency support, offer precise, repeatable signal reproduction. gpsfeed+ [8] provides NMEA-based GPS data via socket or serial port to emulate movement without dedicated hardware. However, these tools focus solely on navigation data and often require costly equipment.

C. Tools for Recording and Reproducing CAN Bus Data

The CAN bus enables low-latency communication between ECUs and is central to automotive systems. Tools like Vector CANoe [9] and CANalyzer [10] support ECU and network testing, offering simulation and real-time analysis, but are proprietary and expensive. An open-source alternative, Savvy-CAN [11], provides real-time CAN data capture and transmission across multiple buses, supporting efficient analysis and reverse engineering.

III. TRACEN-X Architecture

The primary objective of TRACEN-X is to provide a system that can accurately collect and reproduce real vehicle data, enabling the development, testing, and optimization of CAV services under controlled conditions. To develop such a service, it is essential to examine the vehicle's internal data sources. The internal vehicle data required by CAV services can be primarily divided into two categories:

- Navigation data, obtained through: (i) GNSS systems, using National Marine Electronics Association (NMEA) sentences via serial interface; (ii) Inertial Measurement Units (IMUs), using U-blox (UBX) messages or similar binary protocols, via serial interface; or (iii) a combination of both GNSS and IMU for enhanced accuracy;
- Data from internal Electronic Control Units (ECUs) and perception sensors obtained both via CAN bus and serial interfaces.

There are therefore two main *internal* data sources: serial and CAN bus. A third, critical *external* source is V2X data, which originates from other connected entities on the road. However, since we currently focus exclusively on internal data sources within a CAV, it is from these that TRACEN-X has been designed to collect and replicate data. A future extension of TRACEN-X may incorporate external sources, such as V2X messages, to further enhance its capabilities.

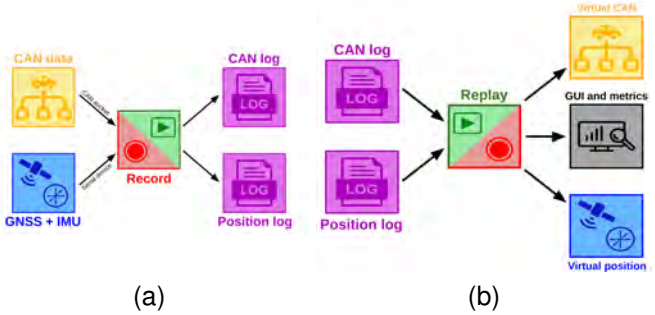


Fig. 1: "X-shape" architectures of TRACEN-X. (a) Record function. (b) Replay function.

The data collection and reproduction is achieved through the two core functions of TRACEN-X: the *Record* function and the *Replay* function, implemented as separate modules, which are detailed in the following subsection. Both functions follow an "X-shape" architecture, as illustrated in Fig. 1.

A. The Record function

The Record function is the first crucial component of TRACEN-X, with its architecture illustrated in Fig. 1a. Its primary goal is to read data from the internal sources of the vehicle and store them in corresponding log files. Specifically, it takes as input navigation data (GNSS + possibly IMU) and CAN data.

The navigation data of a vehicle typically originates from a GNSS receiver, an IMU system, or a combination of both. This data is commonly represented by PVT information, which is included in NMEA sentences. When an IMU system is also integrated, additional data may be provided via binary messages. A relevant example supported by TRACEN-X is the U-blox (UBX) protocol, which is widely used in automotive navigation systems. All these data is usually provided via serial device from a GNSS (plus, possibly, IMU) system connected to the vehicle and can be retrieved using the Python `serial` library.

To read data from the CAN bus, we follow a similar process. Once connected to the CAN bus, TRACEN-X creates a dedicated CAN socket using the Python `cantools` library. It is important to note that, in order to properly decode and interpret the content of CAN frames, a CAN database file is also required.

As the incoming messages are read from a serial device and a CAN socket, they are written to a *serial log file* and a *CAN log file*, respectively. These files are in JSON format and contain all the relevant information extracted from the NMEA sentences and, if available, UBX messages (for the serial log), and from the CAN frames (for the CAN log). A particularly important information included in both logs is the timestamp field, which records the time elapsed since the beginning of the

TABLE I: Literature works comparison.

Tool	Data collection	Data reproduction	CAN bus	Navigation data	No dedicated HW	Open source
TRACEN-X	✓	✓	✓	✓	✓	✓
CarLab [4]	✓	*X*	✓	✓	✓	✓
ART/ATK [5]	✓	✓	*X*	✓	partial*	*X*
LabSat 4 [7]	✓	✓	*X*	✓	*X*	*X*
gpsfeed+ [8]	*X*	✓	*X*	✓	✓	✓
Vector CANoe [9]	✓	✓	✓	✓	offline-only**	*X*
Vector CANalyzer [10]	✓	✓	✓	✓	offline-only**	*X*
SavvyCAN [11]	✓	✓	✓	*X*	✓	✓

*live tests need an ART scale rover or a Chrono-compatible robot
**real-time CAN bus interface required Vector hardware

capture. This timestamp allows us to determine the exact time at which the message should be reproduced when employing the Replay function.

Note that the format of log files can be converted to be compatible most mobility simulators, like MATLAB and SUMO simulator [12]. Thanks to this, they can be easily reproduced on vehicular network simulators, like OMNeT++ [13] or ms-van3t [1].

B. The Replay function

The Replay function in TRACEN-X enables the reproduction of recorded navigation and CAN data in a controlled lab environment, as shown in Fig. 1b. Using the recorded *serial log file* and *CAN log file*, the Replay tool creates a virtual serial device and a virtual CAN bus. These virtual devices facilitate the testing and development of vehicle functionalities that rely on navigation and CAN bus data, such as V2X services. Navigation data reproduction is managed using `socat`, which creates a virtual serial device for data transmission, while CAN data replay is handled via a virtual CAN bus interface generated with the Python `cantools` library.

To preserve the original inter-arrival times of messages, TRACEN-X dynamically adjusts waiting intervals between transmissions. It calculates the difference between simulated elapsed time (relative to the trace's initial timestamp) and actual wall-clock time (measured via the Python `time` package) since replay initiation, adjusting sleep durations accordingly to compensate for processing overhead. The Replay module also offers GNSS position interpolation, enabling smoother and more continuous motion tracking. This feature is especially useful when a higher update rate is needed than that of the original GNSS device and can be activated as required.

The result is a multi-process, accurate reproduction of both CAN and serial data simultaneously, using virtual serial and/or virtual CAN bus devices. Users can connect to these virtual interfaces and access data in real time, as if generated by an actual navigation system or CAN bus. This is particularly beneficial for testing and developing automotive navigation systems, CAN implementations, and CAV services, including V2X applications. The Replay function includes tools for data visualization and analysis. A web-based GUI supports inspection of navigation and CAN data for preliminary checks. It also enables synchronized replay of multiple, separately recorded traces to generate multi-actor scenarios.

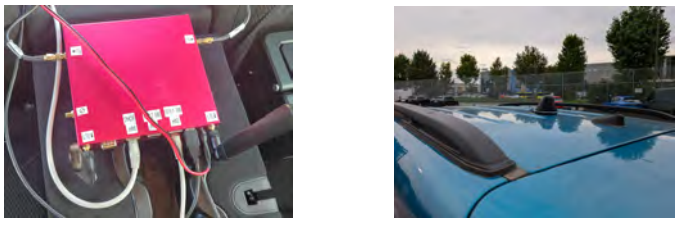
IV. Experimental Results

This section first covers the collection and replay of CAN bus data, followed by the recording and reproduction of navigation data for both vehicles and pedestrians. Finally, we validate TRACEN-X by comparing collected traces when replayed using either the ms-van3t simulator [1] or the OScar [3] implementation of the ETSI C-ITS stack for real OBUs.

A. CAN bus data

To retrieve data from the vehicular CAN bus, we set up a test vehicle equipped with an On-Board Unit (OBU) based on a PC Engines APU2 board, as shown in Fig. 2. This OBU, depicted in Fig. 2a, can read internal CAN bus data and enable V2X communication, following the platform in [14]. Directly connected to the CAN bus, it decodes vehicle messages using a manufacturer-provided database, granting access to front sensor data for detecting objects ahead. Additionally, the OBU generates ETSI C-ITS messages and transmits them via IEEE 802.11p using a multi-band magnetic antenna on the vehicle roof (Fig. 2b).

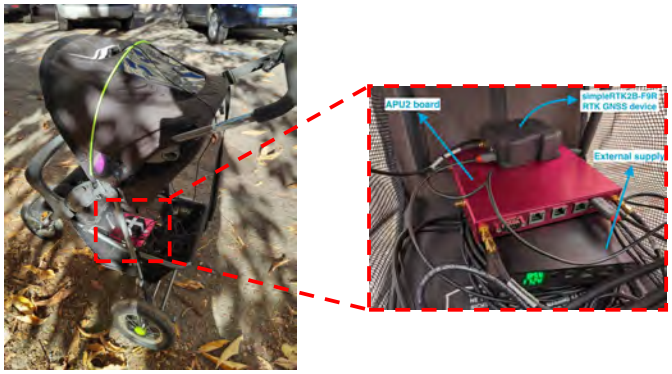
Thanks to this setup, we recorded sensor perception data as a pedestrian moved in front of the car. Once the trace was collected using the Record module of TRACEN-X, we could then reproduce it in the laboratory using the Replay module, in conjunction with the OScar framework, connected to the virtual CAN bus generated by TRACEN-X. We obtained the results in Fig. 3, where the vehicle icon represents the stationary position of the vehicle, while the purple dots indicate the position of the perceived pedestrian moving in



(a) (b)

Fig. 2: CAN perception field test setup. (a) OBU on the vehicle, connected to CAN bus and DSRC antenna. (b) Magnetic GNSS and DSRC antenna on vehicle roof.

Fig. 3: Replay of a CAN trace with a perception of a pedestrian sensed by the equipped vehicle.



(a) (b)

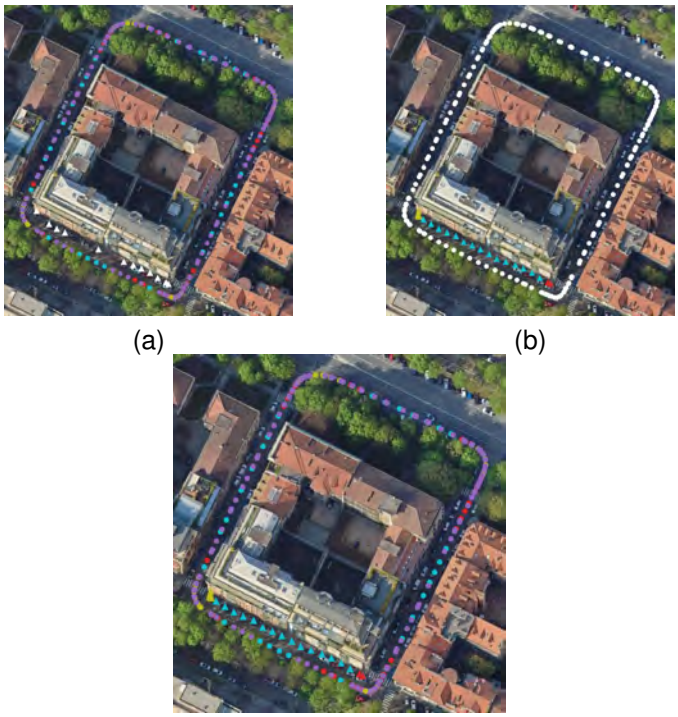
Fig. 4: Pedestrian field test setup. (a) The equipped stroller, front view, with our "VRU" and the DSRC and GNSS antenna. (b) Our hardware boards in the bottom bag of the stroller.

front of the vehicle, as recorded from the CAN bus. Note that this CAN data reproduction process can be executed multiple times in a laboratory environment, allowing for fine-tuning of communication parameters.

B. Navigation data

For the GNSS/IMU tests, we recorded data from both a vehicle and a pedestrian. The pedestrian traces were collected using a "smart stroller" equipped with an OBU supporting V2X connectivity and interfaced with a GNSS + IMU receiver (Fig. 4). A similar setup was installed on a vehicle to capture vehicular traces. Both the stroller and the vehicle followed a similar path at different times while TRACEN-X recorded their navigation data.

This setup allowed us to accurately reproduce, in a laboratory environment, the Cooperative Awareness Message (CAM) trace transmitted by the vehicle and the Vulnerable Road User Awareness Message (VAM) trace sent by the pedestrian, both derived from their recorded navigation data. Notably, despite



(a) (b) (c)

Fig. 5: Replay of vehicular and pedestrian traces. (a) Vehicular CAMs transmitted with pedestrian VAMs received in white. (b) Pedestrian VAMs transmitted with vehicular CAMs received in white. (c) Vehicular and pedestrian traces together.

being recorded at separate times, TRACEN-X enabled us to reproduce both traces simultaneously, effectively creating an emulated interaction between the two entities (see Fig. 5c).

Specifically, in Fig. 5a, the colored dots represent the positions of the CAMs transmitted by the vehicle, while the white triangles indicate the VAMs received by the vehicle from the pedestrian. Conversely, Fig. 5b illustrates the positions of the VAMs transmitted by the stroller, represented by colored triangles, while the white dots denote the CAMs received by the stroller from the vehicle.

This capability highlights the potential of TRACEN-X in generating realistic multi-actor traffic scenarios for testing and validating complex interactions between connected road users.

C. Traces validation

Finally, we validated the recorded traces, shown in Fig. 5, by replaying them for both a simulator, ms-van3t, and a real V2X service, OScar. The pie charts in Fig. 6 illustrate the dissemination of CAMs and VAMs, with different colors representing distinct triggering conditions, according to ETSI standards [15], [16], as also indicated in Fig. 5.

Comparing Fig. 6a, which illustrates the dissemination of CAMs generated by replaying a recorded vehicular trace using the ms-van3t simulator [1], with Fig. 6b, where the same trace was replayed using the OScar framework [3] for real-world

Fig. 6: Comparison of CAMs and VAMs triggering conditions when using TRACEN-X in conjunction with ms-van3t and OScar. (a) CAMs generated from Replay on ms-van3t. (b) CAMs generated from Replay on OScar. (c) VAMs generated from Replay on ms-van3t. (d) VAMs generated from Replay on OScar.

field testing, we observe a close correspondence in the results. Similarly, Fig. 6c and Fig. 6d show the dissemination of VAMs replayed by ms-van3t and OScar, respectively, exhibiting a comparable distribution. This validation demonstrates the reliability and consistency of the recorded traces when reproduced across different testing environments, confirming the robustness of TRACEN-X in faithfully replicating real-world vehicular scenarios.

V. Conclusions

Ensuring reliable performance of Connected and Autonomous Vehicle (CAV) services in real-world conditions remains challenging, even after extensive simulations. A key difficulty lies in accurately replicating real-world navigation and vehicle dynamics in the lab. We presented **TRACEN-X** (Telemetry Replay and Analysis of CAN bus and External Navigation data), the first open-source, Linux-based framework for collecting and replaying real-world *(i)* navigation data (GNSS + inertial) and *(ii)* CAN bus traces in a unified environment. TRACEN-X supports fine-tuning of CAV systems, merging of separately recorded actors into shared virtual scenarios, and generation of V2X-enriched traces.

Future work will focus on improving robustness during trace replay with incomplete data, adding PCAP-based V2X traffic emulation, and enabling more complex interactions such as platooning and vehicle–pedestrian dynamics.

Acknowledgments

This work was partially supported by the European Union - Next Generation EU, through the National Center for Sustainable Mobility (Grant No. E13C22000980001) and was partially funded by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of Next Generation EU, partnership on “Telecommunications of the Future” (PE00000001 - program “RESTART”). This publication is also part of the PNRR–NGEU project, funded by the Italian Ministry of University and Research (MUR) under Decree DM 117/2023.

References

[1] F. Raviglione, C. R. Carletti, M. Malinverno, C. Casetti, and C. Chiasserini, “ms-van3t: An integrated multi-stack framework for virtual validation of V2X communication and services,” *Computer Communications*, vol. 217, pp. 70–86, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366424000227

[2] A. Genovese, M. Rapelli, F. Raviglione, and C. Casetti, “ETSI-Compliant Protection of Connected Vulnerable Road Users: Simulation and Field Trial,” in *2024 IEEE Vehicular Networking Conference (VNC)*, 2024, pp. 343–350.

[3] M. Rapelli, F. Raviglione, and C. Casetti, “OScar: An ETSI -Compliant C- ITS Stack for Field- Testing with Embedded Hardware Devices,” in *2024 22nd Mediterranean Communication and Computer Networking Conference (MedComNet)*, 2024, pp. 1–4.

[4] M. D. Pesé, A. Ganesan, and K. G. Shin, “CarLab: Framework for Vehicular Data Collection and Processing,” in *Proceedings of the 2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services*, ser. CarSys ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 43–48. [Online]. Available: https://doi.org/10.1145/3131944.3133940

[5] A. Elmquist, A. Young, T. Hansen, S. Ashokkumar, S. Caldararu, A. Dashora, I. Mahajan, H. Zhang, L. Fang, H. Shen, X. Xu, R. Serban, and D. Negrut, “ART/ATK: A research platform for assessing and mitigating the sim-to-real gap in robotics and autonomous vehicle engineering,” 2022. [Online]. Available: https://arxiv.org/abs/2211.04886

[6] RACELOGIC, “LabSat GNSS Simulators,” 2010, accessed: [3rd June 2025]. [Online]. Available: https://www.labsat.co.uk/index.php/en/

[7] RACELOGIC, “LabSat 4,” 2010, accessed: [3rd June 2025]. [Online]. Available: https://www.labsat.co.uk/index.php/en/products/labsat-4

[8] SOURCEFORGE, “gpsfeed+,” 2005, accessed: [3rd June 2025]. [Online]. Available: https://gpsfeed.sourceforge.io/

[9] Vector Informatik, “CANoe Family,” 2010, accessed: [3rd June 2025]. [Online]. Available: https://www.vector.com/canoe-family/

[10] Vector Informatik, “CANalyzer — ECU & Network Analysis,” 2010, accessed: [3rd June 2025]. [Online]. Available: https://www.vector.com/se/en/products/products-a-z/software/canalyzer/

[11] CASSIOPEIA, “SavvyCAN,” 2017, accessed: [3rd June 2025]. [Online]. Available: https://www.savvycan.com/

[12] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker-Walz, “Recent Development and Applications of SUMO - Simulation of Urban MObility,” *International Journal On Advances in Systems and Measurements*, vol. 3&4, 12 2012.

[13] A. Varga and R. Hornig, “An overview of the omnet++ simulation environment,” in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools ’08. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[14] F. Raviglione, C. Casetti, and F. Restuccia, “Edge-V : Enabling Vehicular Edge Intelligence in Unlicensed Spectrum Bands,” in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, 2023, pp. 1–5.

[15] ETSI EN 302 637-2 V1.3.2 (2014-11), “Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service,” ETSI, Tech. Rep., 2014.

[16] ETSI TS 103 300-3 V2.2.1 (2023-02), “Intelligent Transport Systems (ITS); Vulnerable Road Users (VRU) awareness; Part 3: Specification of VRU awareness basic service; Release 2,” ETSI, Tech. Rep., 2023.