ROS2-based Small-Scale Development Platform for CCAM Research Demonstrators

Joshwa Pohlmann, Maximilian Matthé, Tobias Kronauer, Paul Auerbach, Gerhard Fettweis

Barkhausen Institut gGmbH, Germany
{firstname.lastname}@barkhauseninstitut.org

***Abstract*—This work proposes an architecture and platform for researching and demonstrating use-cases for connected cars based on small-scale vehicles. The proposal bridges the gap between a lab setup to test individual algorithms and deployments on real cars. It allows researchers to communicate their results with a small-scale indoor demonstrator. The platform employs ROS2 and MicroROS to allow for a modular and scalable hardware and software setup. Moreover, it allows running all control algorithms in a graphical simulation to ease development of complex scenarios. We successfully apply the platform to build a demonstrator for a platooning use-case and point out limitations such as lacking photorealism of the simulation and limited processing power of the platform. Our results indicate that using a well-designed platform and architecture can significantly reduce required effort for implementing connected cars use-cases.**

***Index Terms*—science communication, demonstrator, platform, CPS, ROS2, CCAM, LGSVL, real-time, V2X, V2V, DDS, micro-ROS, DDS-XRCE, simulation, digital twin**

I. Introduction

Connected cooperative and automated mobility (CCAM) algorithms and applications for improving efficiency and safety of future mobility are currently being abundantly researched [1]–[3]. Despite having used accurate vehicle and traffic models, all research results need to be verified and tested in reality. However, today's modern vehicles are a complex collection of software as well as electrical and mechanical components, tightly integrated against each other [4]. Therefore, after building a first functional proof of concept in a research lab, it is often far too early for integration with real cars, as their interfaces are strict, complex and have to comply with safety requirements [5]. The same problem arises when building demonstrators for fairs or public science communication. A real car is impractical to be used on fairs or similar events due to cost, weight, size and safety. Simply relying on the proof of concept however, researchers can only analyze the newly developed concept or algorithm isolated but ignore its interactions to system boundaries. Therefore, we identified the need for an intermediate step in research result verification and demonstration. Such solution shall be cost-effective, mobile, safe and enable fast development while still being realistic enough to provide valuable data for research result verification.

To address this need, the contribution of the present work is two-fold. First, we propose an architecture and evaluate a platform for small scale autonomous cars for testbeds and demonstrators in research. The proposed architecture supports simultaneous development within both a computer simulation and the real world without requiring adaptions to the developed control algorithms. Second, we use the proposed platform to develop a demonstrator for the CCAM use-case *platooning* and point out advantages and drawbacks of our proposal.

The remainder of this paper is organized as follows: In the following section we refine our desired framework with a set of requirements. Current and past studies regarding similar research platforms for CCAM will be reviewed in section III. We explain our suggested solution in section IV where we also discuss tooling for development and testing. The proposed architecture has been implemented and will be evaluated in section V followed by a discussion about the limitations in section VI. Finally, we present possible future work and improvements in section VII.

II. Requirements

To fulfill its purpose, we formulated the following qualitative requirements for a platform or framework, which enables verification and demonstration of research results in the field of connected driving or CCAM:

- R1 Modularity: The architecture should allow re-use and re-arrangement of existing parts and seamless integration of new parts.
- R2 Scalability: CCAM scenarios are usually complex, e.g. platooning on highways or smart crossings. The architecture should hence scale to support numerous instances like cars and road side units but also allow a detailed study of driving behaviours of a single unit.
- R3 Transportability: Platform dimensions shall be small enough for easy transportation to fairs or indoor events.
- R4 Extendability: The architecture shall allow to include new functionality without interfering with existing modules. New functionality includes both hardware like sensors or actuators and software like different control algorithms or user interfaces.
- R5 Time saving: The platform shall be easy to integrate and reduce required development effort and time compared to implementing a demonstrator from scratch.
- R6 Vehicle-to-Vehicle (V2V) communication: The platform shall enable message exchange between vehicles.
- R7 Link-quality control: Provide a way to control the link-quality of the Vehicle-to-Everything (V2X) communication link for controlled experiments.

R8 Computer Vision: The platform shall integrate at least one camera module and enable computer vision for navigation and localization purposes which is required by many self driving algorithms.

Quantitative performance requirements depend on the actual algorithms used to implement a specific use-case on the platform. Therefore we chose performannce values most likely to support a wide range of possible CCAM-scenarios. Details on mechanical and electrical requirements of the vehicle housing the platforms components are omitted on purpose.

Q1 End-to-End (E2E) latency for communication within a single car shall be equal or $<$1ms to support a wide range of CCAM control algorithms. The value is inspired by latencies entailed by the CAN bus [6].

Q2 E2E latency for V2X communication shall be $<$10ms to support a wide range of CCAM control algorithms. [7]

Q3 The physical dimensions of computation and communication components should not exceed 14x20x6cm (WxLxH) to fit a car of scale 1:10. This scale is suitable for demonstration purposes on fairs or exhibitions.

III. State of the Art

Numerous small-scale cars have already been built with varying purposes. La et al. built a small-scale research platform in [8], but being only equipped with a small microcontroller and a scale of 1:14 it is too small with insufficient processing power. Its architecture was not devised to be extended by other sensors or software components on the vehicle. Wireless communication was realized using XBee which does not meet the latency or bandwidth requirements of CCAM applications.

In [9], Pandi et al. use miniature vehicles to demonstrate the behavior of 5G connected cars at an intersection. The vehicles are small and can follow a line autonomously but the architecture also doesn't provide any ways to extend or adapt the vehicles to other use-cases. It is also not open-source and not available for further use.

Valtl et al. presented a data collection platform for autonomous cars in [10]. It integrates common sensors like a LiDAR, but has no concept for V2V communication. All sensors are connected through interfaces provided by a Raspberry Pi (RPi) 3 which limits extendability and real-time capabilities.

An affordable and modular platform was proposed by Quartey and Korsah [11]. It has a modular software architecutre relying on MQTT as a publish/subscribe communication bus. As MQTT requires a broker and is not designed for low-latency, high-bandwith applications [12], it doesn't scale as required. Further, no simulation or other debugging tools were considered to aid development.

Finally, the open-source project DonkeyCar [13] is an affordable, small-scale car with a modular, Python-based software framework. A simulator for development is available. However, the platform is not designed with V2X communications in mind and hence does not scale well when multiple vehicles are involved.

Therefore, we propose a cost-effective architecture and platform which is extendable, scalable, modular and supports V2X by design adhering to the aspects of development effort and tooling.

IV. Architecture and Design

In this section, we will briefly elaborate on different aspects of our architecture and on design decisions. We will start by outlining the communication bus in the first subsection, followed by a detailed description of the hardware platform and the required changes. We will conclude our description by pointing out our modularity approach and the benefits of using a simulation to further ease the development process.

A. Communication Bus

To achieve modularity, scalability and extendability while maintaining real-time capability we decided to employ a publish-subscribe communication bus. Known options, which provide low-latency real-time publish-subscribe connectivity are ZeroMQ [14] and Data Distribution Service (DDS) [15]. Within the context of CCAM we found several vendors [16] [17], which work on a certified DDS software stack for autonomous vehicles. We couldn't identify comparable efforts on the market for ZeroMQ. DDS realizes low-latency communication using UDP, TCP or shared memory. It can automatically discover participants on the network, allows for complex role definitions, provides type safe interface definitions and is an international standard with multiple commercial and open source implementations. Although it is a non-trivial protocol, it can be performant [12]. Further, the Robot Operating System (ROS) [18] project decided to use DDS as a middleware in version 2 of the framework. This led to the conclusion that we satisfy the requirements R1, R4, Q1 and also remain close to real autonomous car architectures if we choose DDS as our communication bus. Complete and good documentation as well as available tools are an important factor to reduce development time. The aforementioned framework ROS 2 provides tools to inspect the bus, manage software launching and interface libraries. However, being a wrapper around DDS it introduces overhead into the system, increasing latency [19] and reducing flexibility by abstracting away vendor specific behavior. Nonetheless, our architecture employs ROS 2 to ease the transition to DDS if necessary and reduce development time by profiting from compatible open-source software developed by the community (see R5). Fig. 1 shows an overview of the layers contained in the used communication bus.

B. V2X communication

The choice for a DDS based framework enables native V2X communication as demanded by R6. Since publishers and subscribers are location transparent, they only require a working IP connection. This posed the challenge to separate communication on each car instance from V2V traffic to avoid collisions. We solved this problem using different DDS domains for each traffic type. Gateway nodes were programmed, which can be configured to forward any topic to the V2X domain and vice versa. The same approach was used to

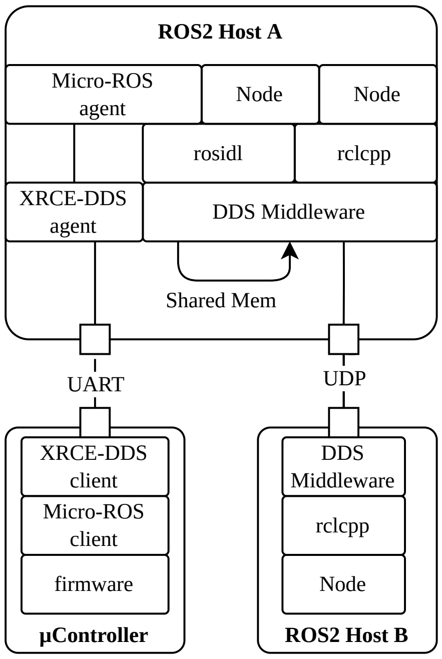

Fig. 1. Layers involved in a ROS2 setup.

establish a domain for control and management data of the system. This logical seperation is highlighted in Fig. 7.

With total control of message forwarding between domains, we implemented link-quality control for packet drops and latency (R7).

C. Hardware platform

a) Processing: Spare processing resources for additional software increase the flexibility of the platform. Especially the camera sensor requires considerable CPU power. Therefore, we decided to select suitable hardware small enough to remain transportable and battery-powered (see R3). The RPi 4 is a single board computer (SBC) with 8GB RAM, a Quad core Cortex-A72 (ARM v8) processor and built-in 802.11 WLAN chip. Any other SBC can be chosen, as long as it supports Linux and provides a UART connection to interface with the ESP32 on the sensor aggregation board (SAB). For example, the NVIDIA Jetson SBC family can be a viable alternative when more processing power is needed.

b) Sensor and Actuator Connectivity: A custom designed PCB was used as a sensor aggregation board (SAB) which provided connectivity to all sensors and actuators. An ESP32 microcontroller is mounted to act as a proxy, providing timing sensitive communication required by some sensors or actuators. Data is exchanged using a client-server extension to ROS 2 called Micro-ROS [20]. It is based on an official DDS extension for extremly resource constrained environments (XRCE), called DDS-XRCE. The ESP32 acts as a client to transmit sensor data over UART to the SBC, using the same data types and topics as the normal ROS 2 communication bus. No custom translators or gateways are required, therefore eliminating error sources and reducing development time (see R5). Further, peripherals can be relocated to other hardware components without any interface changes, since data consuming nodes have no concept of data origin. The combination of ROS 2 and Micro-ROS makes this architecture truly scalable (see R2), only limited by the number of available UART interfaces on the main SBC. Separating software processing on the RPi and the hardware connectivity on the custom designed PCB for sensor connectivity reduces project specific hardware and code. As a result, we expect shorter development times and easier re-use of existing components (see R1).

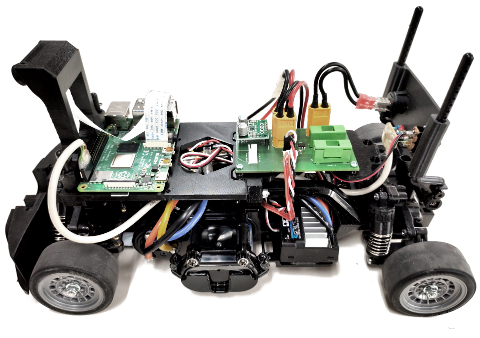

Fig. 2. The base platform containing PDB, camera and RPi.

c) Chassis: A radio controlled model car with a 1:10 scale was used as a basis. It houses a 6:1 transmission and a brushless motor. The model can reach speeds of up to 25 km/h. A power distribution board (PDB) provides power to all chassis and platform components. Fig. 2 shows a picture of the chassis containing the PDB and the RPi and the contained hardware components are illustrated in Fig. 3.

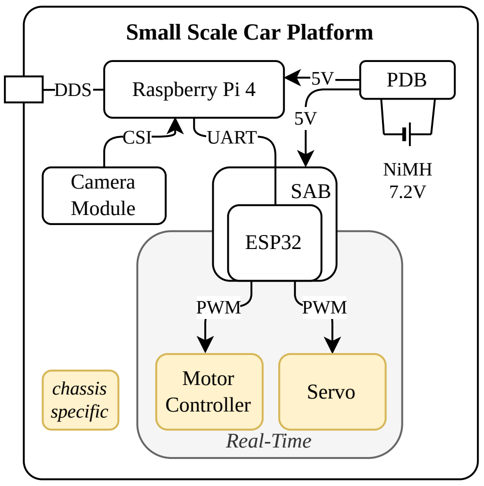

Fig. 3. Hardware components contained in the base platform.

D. Modularity & Simulation

Developing a complex cyber-physical system (CPS) like an autonomous, connected car - even a scaled down, simplified version - requires system integration tests to catch errors early, improve quality and enable early verification of system components, e.g. control algorithms. While unit tests verify components behave as intended, system integration tests verify that components are modelled correctly and interfaces match. Further, system level tests verify and reveal emerging characteristics of the system as a whole like E2E latency. Manual tests with the physical car are too time consuming and not deterministic enough to fulfill the requirement of fast development (R5). Manually measuring system behavior is error-prone and expensive, since appropiate measurement equipment has to be used. As a result, a 3D simulation was introduced into the architecture. LGSVL [21] is an open source simulation based on Unity3D, specifically developed to develop and test autonomous cars. We chose LGSVL over other simulations like Gazebo, which is directly targeted towards ROS, because of the underlying engine and therefore, achievable graphics. This was considered to be important for two reasons.

1) Computer vision algorithms require the most realistic visual model of reality to be tested properly.
2) When used for demonstrating purposes, aesthetics of the presentation are valued by the visitors.

In the simulator, vehicle models, physical properties and environments can be modelled within the Unity3D editor. The sophistication of the models depend on the project requirements and testing scope. Testing the major part of our software required minimization of hardware dependant code. Therefore we realized the sensor interfacing nodes as a thin hardware abstraction layer which we called ultra thin layer (UTL). In this layer each node only reads and publishes raw sensor values without any processing. The same principle applies to motor and steering control. Their behavior has to be cloned in the simulation. This UTL concept was only possible due to the modularization enabled by ROS 2. For all other software components outside this abstraction layer, the source of sensor values and destination of actuator data was transparent.

V. Evaluation

Here, we briefly describe a use-case for our platform, elaborate on implementation details and conclude by pointing out some lessons learned during the deployment of our platform.

a) Concept: We verified our proposed platform and architecture by implementing a CCAM use-case. A suitable scenario covering most of the formulated requirements is platooning, where autonomous vehicles coordinate their maneuvers through wireless communication to achieve a stable convoi in close distance to each other.

In our scenario, 3 cars drive on a lane on a track at varying speed. They can be instructed to drive freely, control distance and speed using the ultra sonic distance sensor only (ACC) or use communication to coordinate their maneuvers (CACC). Controlling and monitoring the demonstrator was done with a touch interface next to the track. Successful implementation of this concept depends on the low-latency communication bus for a stable control algorithm (Q1, Q2). Usage as an indoor demonstrator requires small scale cars (R3). Highlighting the effects of an unreliable V2X link required R7. A photo of the track and touch control unit is shown in Fig. 4.

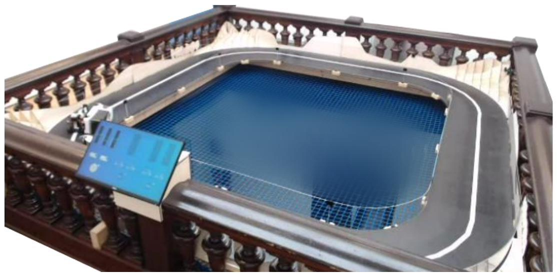
Fig. 4. A one-lane track was setup for the 3 vehicles of the platoon. Users can interact with the installation using a dashboard which directly sends control commands on the ROS 2 DDS bus.

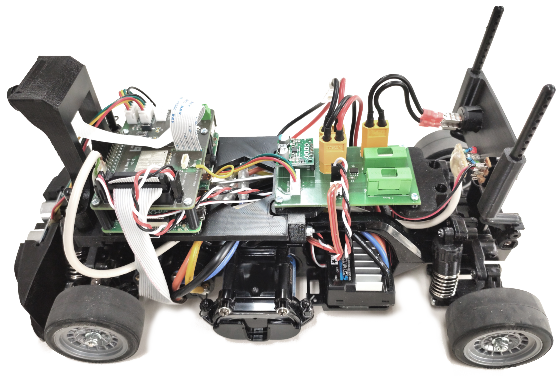
Fig. 5. The platform was extended with the SAB mounted on top the RPi, a line sensor and an ultra sonic distance sensor.

b) Implementation: Four sensors were attached to the chassis:

1) Wide-angle camera module
2) Motor encoder Hall sensor
3) Ultra sonic distance sensor
4) Line array sensor

Camera and line array sensor work in conjunction to implement a stable line following. The motor encoder sensor is required for speed control and the ultra sonic sensor measures the distance to the preceding vehicle. Except for the camera, which requires a high bandwidth connection directly to the processing unit (RPi), all sensors are connected to the adapted SAB. Fig. 5 shows a picture of all components of the chassis for the platooning use-case, and the hardware components and connections are illustrated in Fig. 6. After these modifications the car weighs 1700g without the decorative case mounted and measures 34x16cm (LxW).

A digital twin of the target environment and models of the used sensors were implemented into LGSVL to speed up development and tests. LGSVL already had a camera sensor built-in. A configurable, ideal ultra sonic sensor and motor encoder were developed for the digital twin to mimic the behaviour of the UTL sensor nodes of the real chassis. The communication architecture of the system is shown in Fig. 7.

For presentation purposes, a LED strip and a speaker were attached to the RPi. Their integration followed the same principle as any other component with a ROS node responsible

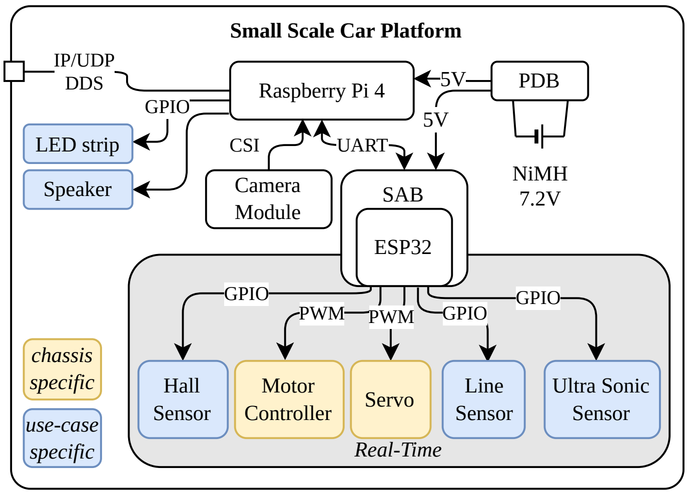

Fig. 6. Hardware components and connections contained in the platooning platform.

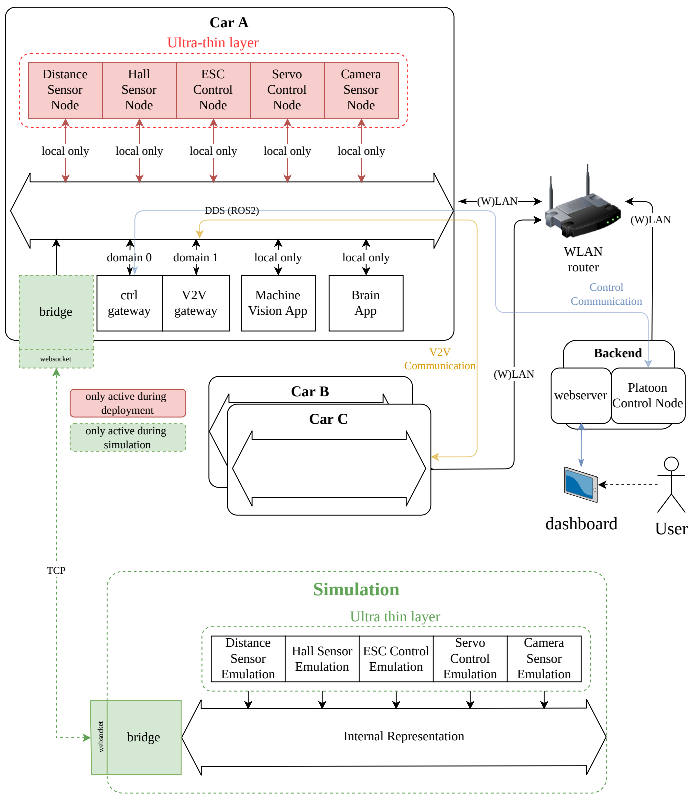

Fig. 7. Detail overview of component communication for the evaluation demonstrator. V2V, control and local node communication were separated using different DDS domains. The simulation interfaces with the internal bus by a bridge using websockets.

controlling them. Although the LED strip protocol requires correct timing and therefore should be added as an ESP32 peripheral we broke the consistency of this architecture for the sake of using spare GPIOs on the RPi.

The track was 4m wide and 4m long with 50cm wide lanes. The corners have a radius of 75cm.

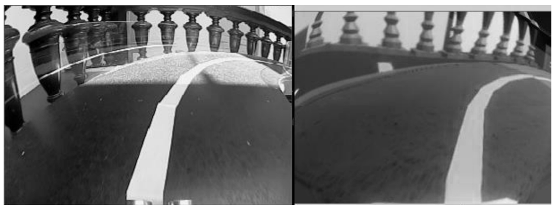

Fig. 8. Difference between reality and simulated camera images. Such inhomogeneous lighting was not modelled in the simulation and hence the image processing algorithm needed to be adjusted in the field tests.

c) Evaluation Results: Besides system integration tests of all nodes, especially the lane detection and distance control algorithm relied on the simulation for testing and development. Labour-intensive manual resets of the car to test the basic functionality of these functions were eliminated. The simulation was also integrated in our continuous integration pipeline, which automatically tested for a working, driving car whenever a new software version was commited.

User interaction was realized with a dashboard with a web-interface to enable demo control. The modular nature of ROS 2 made integration possible without any interface changes, reducing development time. During the development of the computer vision lane following algorithm, the RPi reached its performance limits. We were able to mitigate this problem by offloading this process to a workstation connected over WLAN without any software changes, showing again the benefit of the modular software architecture. Using a feature complete SBC on the cars with Linux and built-in WLAN showed to be a valuable advantage during development. Updating code or configurations could be done using SSH. Moreover, WLAN connectivity was also used for inter-car communication.

The decision for Micro-ROS made integration of the line sensor late in the development with minor interface changes possible. Three steps were required: First, appropiate connectors had to be added to the SAB. Second, the data type for the ROS 2 bus was to be specified. Last but not least, the firmware of the ESP32 had to be adapted to read the sensor values.

VI. Platform Limitations

In section V we verified that the architecture design decisions had the expected effects and benefits. As with any system, limitations apply, either because certain aspects were not considered during design or they were not prioritized.

a) Processing Power: Our evaluation use-case relied on image processing using OpenCV. During development we discovered that the processing was too slow for our real-time control loop. The platform's architecture allowed to offload the image processing to an edge cloud. However, a more suitable hardware for image processing with hardware accelerators should be used. Also, some nodes written in Python were re-written in C++ to comply with the latency requirements of our control loop. This increased development time. Although Python is known to be comparibly slow, more processing power will help reducing the optimization effort in a project.

b) Simulation: The digital twin in the 3D simulation was a helpful addition to development. It should be noted that maintaining a digital twin is time consuming. The environment has to be kept in sync as well as the physical properties of the vehicle model. Therefore, a tradeoff analysis should be made, if a project exhibits a complexity justifying the effort. In the evaluation use-case, the lane detection algorithm failed in field tests, whenever sunlight casted harsh shadows on the track (see Fig. 8). This highlights that the simulation is no replacement for properly specified system verification tests.

c) Chassis: The chassis, designed as a toy, has an unfavorable transmission ratio for small speeds, resulting in unpleasant operation noise levels. We discovered that the abdundance of brakes do also limit the use-cases which can be realized, e.g. emergency braking. Fortunately, switching to another chassis only requires designing a new PDB.

d) Latency: We measured a minimal achievable round-trip time of 2ms from a sensor attached to the SAB and the RPi using a simple ping-pong method. This translates to an E2E latency $<$1ms which satisfies Q1. The bottleneck of this communication chain is the maximum UART speed but can be mitigated by using either multiple microcontrollers as well as multiple or faster UART connections.

The same measurement method yielded an E2E latency of $<$2ms with sporadic jumps to 15 to 20ms which shows that the V2X communication via WLAN mostly satisfies Q2, but reveals the typical unreliability of any ISM band based standard. A dedicated and managed spectrum would be required to completely fulfill Q2 in any scenario.

VII. Outlook

In future, we plan to improve the platform and verify architecture design decisions.

a) Chassis: Using a chassis with brakes and a transmission suitable for low-speeds will contribute to a wider range of possible CCAM use-cases. Not being a feature of normal RC cars, this will increase the costs considerably. Another solution would be to use the braking feature of the motor controller, which applies negative torque to slow down the car. It has yet to be determined if the achievable deceleration is sufficient.

b) Sensors: Additional and more sophisticated sensors would also increase versatility of the platform. For example, while measuring the distance to the front vehicle in our demonstrator, one could not differentiate between preceding cars and the track boundaries, due to the simple ultra sonic sensor. When equiped with a LiDAR sensor, the platform could reduce the effort to robustly implement such a use-case.

c) AI capabilities: CCAM is strongly linked to autonomous cars. Hence, better AI/ML capabilties could help building meaningful demonstrators and tests. Therefore, we plan to replace the RPi with a suitable SBC with AI/ML acceleration, like the Nvidia Jetson.

d) Evaluation: We will use the platform for other demonstrators, proof of concepts and prototypes for CCAM and other wireless applications. This will further verify if the chosen technologies and architecture meet our requirements.

References

[1] H. Bagheri, M. Noor-A-Rahim, Z. Liu, H. Lee, D. Pesch, K. Moessner, and P. Xiao, "5G NR-V2X: Toward Connected and Cooperative Autonomous Driving," *IEEE Communications Standards Magazine*, vol. 5, no. 1, pp. 48–54, 2021.

[2] J. Liu and J. Liu, "Intelligent and Connected Vehicles: Current Situation, Future Directions, and Challenges," *IEEE Communications Standards Magazine*, vol. 2, no. 3, pp. 59–65, 2018.

[3] M. Centenaro, S. Berlato, R. Carbone, G. Burzio, G. F. Cordella, R. Riggio, and S. Ranise, "Safety-Related Cooperative, Connected, and Automated Mobility Services: Interplay Between Functional and Security Requirements," *IEEE Vehicular Technology Magazine*, vol. 16, no. 4, pp. 78–88, 2021.

[4] S. H. Leilabadi, N. Katzorke, M. Moosmann, and S. Schmidt, "Systematic Test Case Design for Autonomous Vehicles," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[5] "Road vehicles - Functional safety," International Organization for Standardization, Standard, 2018.

[6] H. Daigmorte, M. Boyer, and J. Migge, "Reducing CAN latencies by use of weak synchronization between stations," in *16th International CAN Conference*, Nuremberg, DE, 2017, pp. 1–8. [Online]. Available: https://oatao.univ-toulouse.fr/23021/

[7] M. H. C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. ahin, and A. Kousaridas, "A Tutorial on 5G NR V2X Communications," *IEEE Communications Surveys Tutorials*, vol. 23, no. 3, pp. 1972–2026, 2021.

[8] H. M. La, R. S. Lim, J. Du, W. Sheng, G. Li, S. Zhang, and H. Chen, "A small-scale research platform for intelligent transportation systems," in *2011 IEEE International Conference on Robotics and Biomimetics*, 2011, pp. 1373–1378.

[9] S. Pandi, F. H. P. Fitzek, C. Lehmann, D. Nophut, D. Kiss, V. Kovacs, A. Nagy, G. Csorvasi, M. Toth, T. Rajacsis, H. Charaf, and R. Liebhart, "Joint Design of Communication and Control for Connected Cars in 5G Communication Systems," in *2016 IEEE Globecom Workshops (GC Wkshps)*, 2016, pp. 1–7.

[10] J. Valtl, J. Mendez Gomez, M. Cullar, and V. Issakov, "Autonomous Platform based on Small-Scale Car for Versatile Data Collection and Algorithm Verification," 04 2021.

[11] B. Quartey and G. Ayorkor Korsah, "Affordable Modular Autonomous Vehicle Development Platform," in *2018 IEEE 7th International Conference on Adaptive Science Technology (ICAST)*, 2018, pp. 1–8.

[12] S. Profanter, A. Tekat, K. Dorofeev, M. Rickert, and A. Knoll, "OPC UA versus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols," in *2019 IEEE International Conference on Industrial Technology (ICIT)*, 2019, pp. 955–962.

[13] An opensource DIY self driving platform for small scale cars. [Online]. Available: https://web.archive.org/web/20220224103151/https://www.donkeycar.com/

[14] "Zeromq," https://zeromq.org, Accessed: 23rd Feb., 2022.

[15] "DDS - Data Distribution Service," https://www.dds-foundation.org/, Accessed: 23rd Feb., 2022.

[16] Real-Time Innovations. (2022) RTI Connext Drive. [Online]. Available: https://web.archive.org/web/20220114191550/https://www.rti.com/drive

[17] ADLINK. (2022) Connected Autonomous Vehicle Solutions. [Online]. Available: https://web.archive.org/web/20220216174229/https://www.adlinktech.com/en/connected-autonomous-vehicle-solutions

[18] D. Thomas, W. Woodall, and E. Fernandez, "Next-generation ROS: Building on DDS," in *ROSCon Chicago 2014*. Mountain View, CA: Open Robotics, sep 2014. [Online]. Available: https://vimeo.com/106992622

[19] T. Kronauer, J. Pohlmann, M. Matth, T. Smejkal, and G. Fettweis, "Latency Analysis of ROS2 Multi-Node Systems," in *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2021, pp. 1–7.

[20] S. Dehnavi, D. Goswami, M. Koedam, A. Nelson, and K. Goossens, "Modeling, implementation, and analysis of XRCE-DDS applications in distributed multi-processor real-time embedded systems," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021, pp. 1148–1151.

[21] Guodong Rong et. al., "LGSVL simulator: A high fidelity simulator for autonomous driving," *CoRR*, vol. abs/2005.03778, 2020. [Online]. Available: https://arxiv.org/abs/2005.03778