An ETSI ITS-enabled Robotic Scale Testbed for Network-Aided Safety-Critical Scenarios

José M. Pinheiro[*], Ênio V. Filho[†‡], Pedro M. Santos[†‡], L. Almeida[*‡]
[*] *Faculdade de Engenharia – Universidade do Porto*
[†] *Instituto Superior de Engenharia do Porto – Instituto Politécnico do Porto*
[‡] *CISTER Research Center in Real-Time & Embedded Computing Systems, Portugal*
up201705172@edu.fe.up.pt, {enpvf,pss}@isep.ipp.pt, lda@fe.up.pt

***Abstract*—Vehicle-to-Everything (V2X) communications allows new and exciting relevant Intelligent Transportation Systems (ITS) applications such as cooperative perception. In some vehicular use-cases, infrastructure can have a crucial role in safeguarding safety, e.g. in an intersection in which inflowing vehicles do not have Line-of-Sight (blind corner) and an emergency braking action is required. Performance evaluation of vehicular communications systems (end-to-end delay, packet loss ratio, etc.) often resorts to static wireless testbeds, but there tends to be little consideration about the other subsystems involved in the time-sensitive goal, namely sensors, processing and decision-making, and vehicle actuators. In this paper we present a laboratorial testbed in which ETSI ITS/IEEE 802.11p-based On-Board and Road-Side Units (OBU/RSU) are deployed on a 1/10-scale autonomous robotic vehicle and on the road-side respectively, the latter being part of a road-side infrastructure that includes a camera and an edge processing node. We target a use-case of collision avoidance supported by the network, in which the infrastructure detects an impending collision and issues a DEN message to prevent it. We aim to take a step beyond the traditional end-to-end delay characterization that is limited to the communication subsystem, by proposing a tool that enables realistic characterization of the entire end-to-end (detection-to-action) delay in safety-critical use-cases. Results show that the end-to-end latency of the system (camera-edge processing node-RSU-OBU-vehicle actuators) is under 100ms.**

***Index Terms*—Vehicle safety, ETSI ITS, DENM, Testbed**

I. Introduction

According to [1], 92.7% of new vehicles available in the US have at least one Advanced Driver Assistance Systems (ADAS) feature available. Yet, such systems are not perfect and may fail in complex scenarios, such as intersections or adverse weather conditions [2]. In [3], Lane Keep Assist performed well in a closed course with well-defined lane markings, while Emergency Brake with a simulated disabled vehicle contact made in 66% of tests. V2X communications may complement in-car systems to enable a variety of relevant ITS applications, most notably to expand the situational awareness of the vehicle through *cooperative perception*. The ETSI ITS stack [4] defines a suite of services oriented for vehicular scenarios with emphasis on safety-related applications, such as Cooperative Awareness (CA) and Decentralized Environmental Notification (DEN). The U.S. counterpart DSRC/WAVE defines similar services (e.g., Basic Safety Messages – BSM), and both regional stacks rely on the IEEE 802.11p standard as access technology (refered to as ITS-G5 [5] in the ETSI ITS stack). Infrastructure can play a major role in enabling cooperative perception. In circumstances such as intersections with *blind corner*, i.e., approaching vehicle do not have Line-of-Sight (LoS) to other inflow roads, the vehicles' sensors range becomes limited and *ad hoc* communication performs poorly due to shadowing. Infrastructure can alleviate this problem; judicious placement of radios and sensors may enable such a collision avoidance application.

In this paper we propose a scale testbed equipped with ETSI ITS-capable devices that allows to prototype and develop network-supported vehicular applications. The goal is to enable realistic characterization of the entire end-to-end (detection-to-action) delay for safety-critical use-cases in laboratory conditions. Previous works relied heavily on simulation [6], [7], and V2X wireless testbeds that are static or limited to the communications system [8], [9]. In [8], a Mobile Edge Computing (MEC)-enabled collision avoidance service is proposed. The authors implemented and validated the service using OBUs with OpenC2X, but provide no latency evaluation. A Vehicle-to-Infrastructure (V2I) testbed for cybersecurity testing is presented in [9]. The evaluation of the impact of the network performance on the timeliness of the intended physical action/reaction is less explored. An hybrid solution has been the hardware-in-the-loop approach [10], in which actual communications hardware is connected to a vehicular simulator so that there can be a manifestation of the network performance on the vehicle behaviour. Some works have carried out extensive experimental characterization using full size vehicles [11], but involved cost and logistics make such works rare. The current work proposes a middle ground between a simulation-based evaluation and full-size implementation. Previous work has showcased the benefits of this approach [12], that we now extend to have a road-side infrastructure.

We propose a testbed composed of a 1/10-scale autonomous

This work was supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDP/UIDB/04234/2020); by the European Regional Development Fund (ERDF) through the Operational Competitiveness Programme and Internationalization (COMPETE 2020) under the PT2020 Partnership Agreement and the Portuguese Foundation for Science and Technology (FCT) and the Portuguese National Innovation Agency (ANI), under the CMU Portugal partnership, within project POCI-01-0247-FEDER-045912 (FLOYD); and by ERDF through the Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, within project NORTE-01-0145-FEDER-000062 (RETINA).

robotic vehicles, inspired by the F1Tenth competition [13], equipped with a ETSI ITS/IEEE 802.11p-based On-Board Unit (OBU), and a road-side infrastructure that includes a Road-Side Unit (RSU), a camera and a computing node to carry out object detection. We focus on a use-case of collision avoidance/emergency braking triggered by the infrastructure, as the road-side camera detects a potential collision and send an ETSI ITS DEN message, causing the vehicle to come to an halt. This testbed can be seen in the link: https://youtu.be/psUvqg3xltA. The contributions of this work are:

- Integration of an ITS V2X wireless platform with the control logic of a 1/10-scale robotic vehicle;
- Development of a road-side infrastructure capable of object detection and integration with V2X platform;
- Characterization of end-to-end temporal performance of the hazard detection and communication system.

The remainder of this document is as follows. The use-case and application of the ETSI ITS stack are described in Section II. Section III presents the V2X-enabled scaled robotic and road-side testbed. Results on end-to-end communications latency and braking distance are shown in Section IV. Section V draws final remarks.

II. Collision Avoidance Use-case & ETSI ITS Stack

We describe the collision avoidance use-case and application, and introduce the relevant ETSI ITS elements.

A. Collision Avoidance Use-Case

We showcase the use of the testbed in a specific application: collision avoidance supported by the infrastructure. This application is useful in many real-world scenario. Similar use-cases have been proposed in the context of ETSI ITS [14, scenario C.1.3.2] (*Stationary vehicle warning*) and C-V2X [15, Use-case 4.10] (*Obstructed View Assist*). In our use-case, we can focus on the scenario of two simultaneous vehicles entering an intersection where a blind corner exists, and thus the vehicles do not have Line-of-Sight visually nor wirelessly.

Judiciously placed road-side infrastructure can play a role in safeguarding passengers' and road users' safety in such scenario. We assume that the infrastructure is equipped: (i) with ETSI ITS-capable Road-Side Units (RSU), and (ii) sensors, particularly a camera that monitors a region-of-interest. Note that the road-side infrastructure could also receive information from a centralized transit management service (we assume this not to be available) or from CA Messages (CAM) broadcast by vehicles (although not all vehicles may be ETSI ITS-capable, thus motivating the need for road-side sensors). As the road-side infrastructure receives the status updates of a protagonist vehicle, via the regular CAMs issued by its ETSI ITS-capable transceivers (the On-board Unit – OBU), and the infrastructure also detects (via camera) another non-ETSI ITS capable vehicle or road-user of other kind is entering the region-of-interest, it will issue a DEN Message (DENM) to the protagonist vehicle about an impending collision. This is represented in Figure 1.

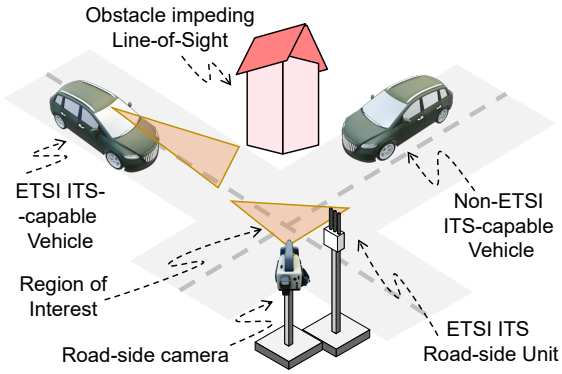


Fig. 1: Collision Avoidance use-case & system

This application can be implemented by a ETSI ITS Collision Avoidance System. In terms of hardware and software, it has the following requirements: ETSI ITS-capable transceivers on both protagonist vehicle (OBU) and infrastructure (RSU), and a road-side infrastructure that is able to capture a video stream of a given area, perform object detection, identify a potential collision, and prepare and send a DENM that will lead the protagonist vehicle to carry out emergency braking. Prior to discussing the architecture and implementation of such a system (Section III), we introduce the ETSI ITS stack and application of DEN messages to this particular scenario.

B. ETSI ITS Architecture

ETSI ITS specifies a communication stack for Intelligent Transportation Systems in the domain of road transportation The stack is composed of four fundamental layers – *Access* (PHY/MAC), *Networking & Transport*, *Facilities* and *Applications* –, with the Facilities Layer providing some of the most noteworthy services, namely the Cooperative Awareness (CA) and Decentralized Environmental Notification (DEN) services. The CA service cyclically broadcasts (with variable periodicity) Cooperative Awareness Messages (CAM) containing relevant information of an ITS station, while the DEN service specifies messages to warn road users of events. These services are connected to the Local Dynamic Map (LDM) facility, that is responsible for receiving and storing the messages from the underlying services. The LDM builds a digital map of all dynamic objects and road details, such as traffic lights, that may be sensed by the own station or through near-by road users through messages like CAM. The relevance of these services comes from the fact they provide *facilities* for upper layer ITS applications to achieve their goals.

C. Decentralized Environmental Notification Messages

DENMs [16] are used to advertise events. After an event is detected by an ITS station, it transmits a DENM to disseminate the occurrence to other stations that are nearby. The transmission is initiated by the application layer. Contrary to CAMs, DENMs can be forwarded by the receiving stations as the objective is to warn a certain region of interest. Every event needs to be characterized by its type, position, the time of detection

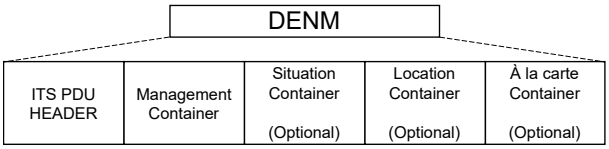


Fig. 2: Structure of a DENM (from [16])

and duration. Every DENM follows a structure made of a common header and various containers, some of which optional, as seen in Figure 2. The **Header** contains information about the protocol version, type of message and station ID from the origin of the message. The **Management container** includes information about the event the DENM refers to; fields are ActionID, time of detection, position of event and the type of station that sent the message, among others. The DENM **Situation Container** is optional and characterizes the detected event. Obligatorily, it must include *informationQuality* and *eventType*, that provides a description for the detected event using a number correspondent to a certain type. This identifier unfolds in two, the *causeCode* and *subCauseCode* describe the event even more accurately. For example, a DENM indicating a stationary vehicle detected in the road would contain a *causeCode* of 94; in addition, a *subCauseCode* of 1 would indicate a human problem and 2 a vehicle breakdown. The **Location container** provides information about the location of the signaled event. It must include *traces*, a field that indicates one or more itineraries one could follow to find the event. The **À la carte container** provides optional information such as *lanePosition*, *externalTemperature* and *stationaryVehicle*.

D. ETSI ITS DENMs in this Use-case

The *Situation Container* contains a slot dedicated to information about the type of event that the DENM refers to, called *eventType*. The *eventType* field is divided into two data elements, the *causeCode* and the *subCauseCode*, that allow to more accurately draw the situation through the message. Examining Table 10 of [16], one can see the variety and abundance of codes that represent the most specific of situations, replicated in Table I. The basic set of applications of ETSI ITS [14] describes a *Cause code* named *Stationary Vehicle Warning*. Code 10 warns of a hazardous location due to an obstacle on the road, which can include a stopped vehicle. If a vehicle continues approaching that location without taking due measures, the edge node may identify the risk of a collision and use Code 97 and warn that a collision may be imminent.

TABLE I: Some available cause codes (from [16])

Direct Cause Code	Cause Description	Sub cause Code	Sub cause Description
9	Hazardous location - Surface condition	0	Unavailable
		1 to 9	As specified in tec109 of clause 9.18 inTISA TAWG11071 [i.10]
10	Hazardous location - Obstacle on the road	0	Unavailable
		1 to 7	As specified in tec110 of clause 9.19 in TISA TAWG11071 [i.10]
97	Collision Risk	0	Unavailable
		1	Longitudinal collision risk
		2	Crossing collision risk
		3	Lateral collision risk
		4	Collision risk involving vulnerable road-user
99	Dangerous Situation	0	Unavailable
		1	Emergency electronic brake lights
		2	Pre-crash system activated
		3	ESP(Electronic Stability Program) activated
		4	ABS (Anti-lock braking system) activated
		5	AEB (Automatic Emergency braking) activated
		6	Brake warning activated
		7	Collision risk warning activated

III. ETSI ITS Scale Testbed

We describe the testbed that implements the ETSI ITS Collision Avoidance System in laboratory context.

A. Architecture & Implementation Overview

From a physical deployment point of view, the ETSI ITS Collision Avoidance System can be broken down into two main parts: the road-side infrastructure, and the in-vehicle system. The road-side infrastructure encompasses a *video source* (camera) that monitors permanently an area of interest, a *Object Detection Service* that performs object detection from the video stream and determines the dynamics of the vehicles (motion direction vector); a *Hazard Advertisement Service* that identifies a potential collision and triggers the sending of a message via the OBU, and ultimately the *ETSI ITS stack*, that sends a DENMs that properly described the event.

At the vehicle, the following components exist: the *ETSI ITS stack and radio*, that receives DENMs and the *Message Handler* module, to which the received DENMs is delivered to interpret the content of the message; and the *Motion Planner*, a module of the vehicle that decides the next actions of the vehicle on the short/medium term and takes into consideration, besides its own sensors and navigation information, the data received from the network. Figure 3 presents the end-to-end sequence of components that make up the system.

If an on-coming vehicle crosses a point of the road, the *Object Detection Service* identifies it and contacts the *Hazard Advertisement Service* to assess a potential collision from consulting the LDM. If so happens, the *Hazard Advertisement Service* instruct to *ETSI ITS stack* to send a DENM. At the vehicle, a *Message Handler* checks for new messages and if a DENM is detected, the information is transmitted to the vehicle's *Motion Planner*. Then, if the DENM contains such information that an emergency brake is needed, the *stop* procedure is sent to the engine actuator in order to stop the vehicle. This behavior is further exemplified in the sequence diagram in Figure 4, where we can see the expected flow of the information and messages along the system.

We implemented the above software architecture in actual communications, sensing and processing equipment that do not present relevant differences from the one that would be used in real-world conditions. Furthermore, we deploy this equipment in a 1/10-scale robotic testbed that may be

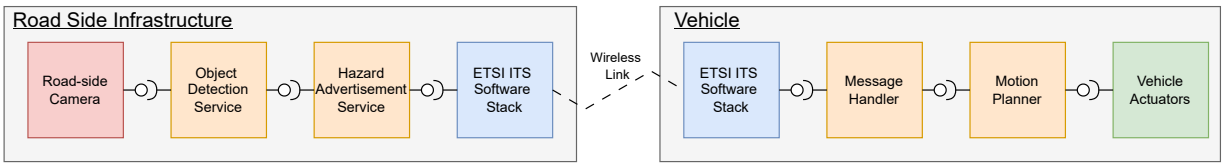


Fig. 3: Components of the ETSI ITS Collision Avoidance System

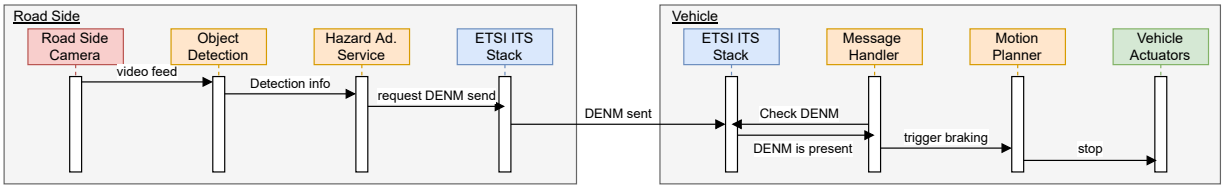


Fig. 4: Sequence diagram of the ETSI ITS Collision Avoidance System

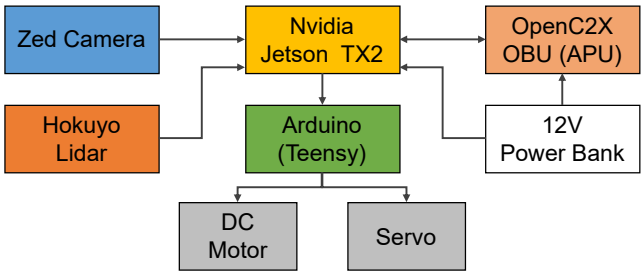


Fig. 5: HW architecture of robotic vehicles (updated from [10]).

leveraged to experiment and prototype wireless vehicular applications prior to deployment in actual vehicles. We provide further description of our implementation of the testbed in the next sections. For convenience, the following breakdown is observed: scale vehicles, road-side sensor and computation infrastructure, and OpenC2X-capable devices.

B. Scale Robotic Vehicles

A 1/10 scale vehicle testbed dedicated to prototyping autonomous driving protocols, named CopaDrive [10], is used as a physical manifestation of the scenario. It is based on the F1/10 platform [13], an open-source autonomous vehicle project that describes the complete platform, including chassis, sensors and computation, power board and the communication architecture. The vehicle is based on a Traxxas Rally car 1/10 scale model, uing its chassis and power train, including the batteries. It has 4-wheel drive, a top speed of up to 60 km/h and rubber tires with tread that further approximate it to a real scenario. The electric motor is controlled by an Electronic Speed Controller (ESC) using Pulse Width Modulation (PWM). Computational power is guaranteed by a Nvidia Jetson TX2 computer, that is installed on top of the Traxxas chassis, running ROS (Robotic Operating System). A Teensy Microcontroller Unit (MCU) interfaces the Jetson with motor and steering. The following sensors are installed: a ZED depth camera capable of stereo vision, an Hokuyo scanning LiDAR and an Inertial Measurement unit (IMU). Figure 5 describes the hardware architecture.

The vehicle is currently configured to perform a simple line following procedure. However, the available equipment allow

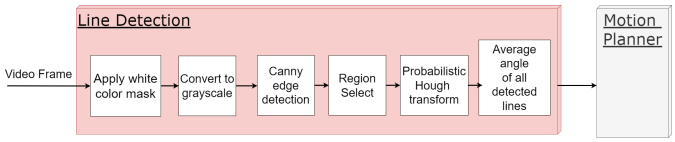


Fig. 6: Line decision algorithm structure.

it to navigate a closed-circuit fully autonomously [10], a feature that was deemed unnecessary in this context. The vehicle's internal operation is as follows. The ZED camera captures video that is sent to the Jetson through a Robot Operating System (ROS) topic, to the Line Detection algorithm. We use Canny edge detection function from OpenCV[1] to detect the edge between the line and the floor. After detecting the edges and applying a region filter to only receive the center of the image, we apply a probabilistic Hough Lines Transform [17] with the objective of receiving the coordinates of the detected lines. These lines and their respective coordinates are then transmitted to the *Motion Planner* module as ROS topics. The *Motion Planner* computes two pairs of coordinates from the line coordinates: its own position and the endpoint of the line where the car aims to be. In order to calculate the steering angle to be transmitted from the Motion Planner to the module, a Proportional-Integral-Derivative (PID) controller is implemented. The steering angle is then transmitted to the Control module, that uses Universal Synchronous/Asynchronous Receiver Transmitter (USART) to make a PWM signal reach the DC motor and servo through the Teensy module. The sequence of steps of the line decision algorithm is shown in Figure 6.

C. Road-Side Camera and Image Processing at Edge Node

The edge infrastructure is meant to detects events and issue DEN messages when necessary. Hardware-wise, the software components of Figure 3 are provided or hosted by a Zed camera[2] as *video source*, a Nvidia Jetson NX as an edge computing node (hosting *Object Detection Service* and *Hazard Advertisement Service*). The overall edge infrastructure is shown in Figure 9.

[1]https://opencv.org/
[2]https://www.stereolabs.com/

1) Image Processing Setup: For the purpose of learning its surroundings, we use the object detection library YOLO [18] at the computing device. In a glimpse, given a figure, YOLO predicts multiple bounding boxes and object class probabilities for those boxes. YOLO does this resorting to a single convolutional network that analysis the whole figure at once; to accomplish this, YOLO is built on the open-source neural network framework Darknet[3]. YOLO also offers a distance estimation functionality for the objects identified by the bounding boxes.

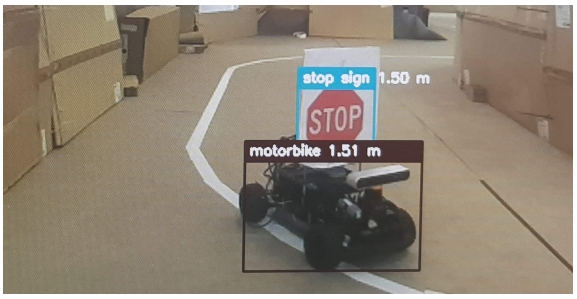
For YOLO to run at real-time speeds, the computing device needs to support NVidia CUDA acceleration. To meet this requirement, we use a Nvidia Jetson Xavier NX running Ubuntu 18.04 LTS and CUDA 10.2[4] to that end. On this device, we had to install Stereo Lab's own ZED development kit, CUDA Deep Neural Network Library (cuDNN)[5] (a GPU-accelerated library of primitives for deep neural networks) and OpenCV 4.6.0, an open source computer vision library were also installed. The *GPU* and *CUDNN* options were activated to use GPU acceleration with CUDA and cuDNN, respectively. OpenCV allows the detection on live video feeds from cameras and the libso option builds a darknet library *libdarknet.so* necessary for the Python implementation used.

2) Event Detection at Edge Node & Challenges: The road-side infrastructure of the ETSI ITS Collision Avoidance System will be monitoring a Region of Interest to detect road-users that enter it and issue a DENM to the protagonist vehicle in order to prevent a collision. The selected event was that the road-user would reach a certain distance to the camera, using the distance estimation functionality of the YOLO library. On preliminary tests, we identified that YOLO can only detect objects up to approximately 75 cm; under this value, estimated distance defaults to 1.73m. Hence the threshold distance was set to this value.

A particular challenge was the identification of the robotic vehicle as such by YOLO. The lack of common features such as bodywork and headlights on our vehicle hindered the accuracy of the detection. From a 3/4 view of the front of the vehicle and at less than 2 meters of distance, YOLO classified the vehicle as a *motorbike* (as seen in Figure 7a). However, this identification was inconsistent and varied from each analysed frame. In an attempt to approximate this work to a real world scenario, we experimented with adding the original Traxxas rally car body shell to the autonomous vehicle. This shell includes recognizable features of a real car like headlights and the overall shape. This new version of the vehicle was recognized by YOLO (Figure 7b), but remained unreliable: identified object class oscillated between *car* and *truck*, it was very sensitive to the angle w.r.t. the camera, and the range of recognition was very short. Finally, a cardboard *stop sign* placed on top of the car proved to be the most resilient option. The stop sign does not cause doubt to the recognition software. Figure 7c shows how the software detects the sign and assigns a box to it, even in a case where it also associates the vehicle to a *motorbike* tag.

(a) Vehicle recognized as *motorbike*.

(b) Vehicle recognized as *truck*.



(c) *STOP sign* detection proved to be the most resilient.

Fig. 7: Explored options to achieve steady, reliable detection.

D. OpenC2X deployments in the OBUs

OpenC2X [19] is an open-source implementation of the ETSI ITS stack. It also offers several applications, notably to trigger the transmission of CAMs/DENMs, that are accessible to the user via an HTTP API. In our testbed, OpenC2X was deployed in two PCEngines APU2 single-board computers equipped with a Compex WLE200NX wireless module, capable of operating in the frequency range needed for IEEE 802.11p [20]. There are several commercial options of OBU/RSU, such as Commsignia V2X OBU unit[6], UNEX OBU-301x[7], Codha's Mk5 OBU[8], and Ettifos V2X OBU[9]. These impose, however, some barriers to simple prototyping goals, such as cost, proprietary/closed stacks, and development

[3]https://pjreddie.com/darknet/
[4]https://developer.nvidia.com/about-cuda
[5]https://developer.nvidia.com/cudnn

[6]https://www.commsignia.com/
[7]https://www.unex.com.tw
[8]https://www.cohdawireless.com/solutions/hardware/mk5-obu/
[9]https://www.ettifos.com/

environment (SDKs) lock-in. This motivated our option for the open-source alternative OpenC2X, for which multiple testimonies of successful usage exist [20]–[22].

The software structure of OpenC2X (described in detail in [19]) relates closely to that of the ETSI ITS stack. A *DEN App* triggers the sending of a DENM message when executed (with a similar The CAM service for periodic messages transmissions). The *Local Dynamic Map (LDM)* is a digital map that represents static road-related information (curbs, pedestrian walking, bicycle paths and road furniture such as traffic signs and traffic lights) and dynamic entities such as road-users, either directly sensed or indicated by other road users, thus enabling cooperative perception. Of noteworthy relevance are the *Server*/Web Interface, that represents graphically the georeferenced information contained in the LDM, with the two being interfaced by the *Server*, and allows the sending of DENMs and CAMs.

1) Updating Message Types & Compilation Procedure: The contents of each container of the CAM and DENM can be found in the *common/asn* folder of OpenC2X, in the form of .ASN files. Modifying these files and recompiling with the *asn1* compiler alters the content and the shape of the containers. Inspecting the LDM showed us that the DENM information was being stored statically in a *sqlite3* database. Adding the ability of reading/writing the information of the container identified in Section II-D requires editing the LDM file. Due to time constraints this was not done; the testbed here presented has used solely DENMs with the mandatory structure (Header and Management Container).

OpenC2X has an embedded branch that integrates the software into OpenWRT, including the necessary patches to enable 802.11p. The testimony of [21] proved valuable to get it working correctly.In the end of the build, a flashable image of OpenWRT with OpenC2X included is available.

2) Integration of OpenC2X in Vehicle and Road-Side Infrastruture: At the vehicle, a Python script running at the Jetson TX2 is constantly communicating with the OpenC2X's HTTP API hosted at the OBU, through *POST* requests sent to *http://<OBU IP>:<OpenC2X Web port>/request_denm*. If no DENM is found, it only returns an HTTP 200 success status code. If a DENM was received by the OBU, a response to the request is sent and power to the wheels is interrupted by the control logic at the Jetson, stopping the car.

On the road-side infrastructure, another instance of OpenC2X is running on the RSU, and likewise to the OBU, its HTTP API is also used to transmit CAM or DEN messages. If an event is detected through the camera, a *POST* request is sent to *http://<OBU IP>:<OpenC2X Web port>/trigger_denm*, which in turn will trigger the OpenC2X instance to transmit a DENM.

IV. System Delay & Braking Performance

In this section, we report the temporal response of the end-to-end system. Although less relevant, we also report the braking distance in our testbed. A discussion on the pertinence and applicability of this testbed concludes the section.

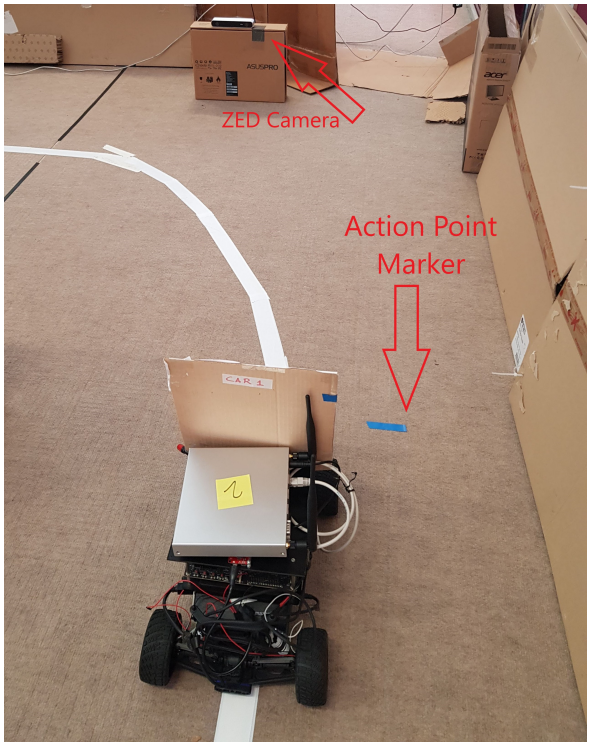


Fig. 8: Testing conditions.

The experimental setup is shown in Figure 8. For convenience, we use the same vehicle as both the road-user that will be approaching or entering a critical area, as well as the protagonist vehicle that is informed wirelessly to stop. While this does not follow closely the intended use-case, it nevertheless showcases all the required functionalities. The vehicle (bottom) begins the test by following the line on the floor autonomously. The ZED camera (top) monitors the Region of Interest, and once the Action Point (shown as the blue line at the right), set as a threshold distance to the camera, is crossed, a DENM is sent. The core of the edge infrastructure (shown separately in Figure 9), composed of the Nvidia Jetson NX and OBU, detects the vehicle's position from the camera's videostream and issues the DENM. When the vehicle's control logic is made aware that a DENM has arrived, all power supply to the wheels is cut.

A. Latency of Emergency Braking System

1) Measurement Setup: We characterize the time it takes since the hazard is detected by the road-side infrastructure until activation of the actuators at the vehicle. To that end, timestamps were collected through the various steps of the chain of action, in order to measure the time elapsed intervals between the stages/components shown in Figure 4.

1) Step 1: the vehicle reaches the Action Point (the point at which the brakes must be applied as soon as possible).
2) Step 2: the YOLO software outputs an identification of the vehicle and that it is at the Action Point.
3) Step 3: the RSU sends a DEN message, on request by the edge node.

Fig. 9: Edge system components assembled.

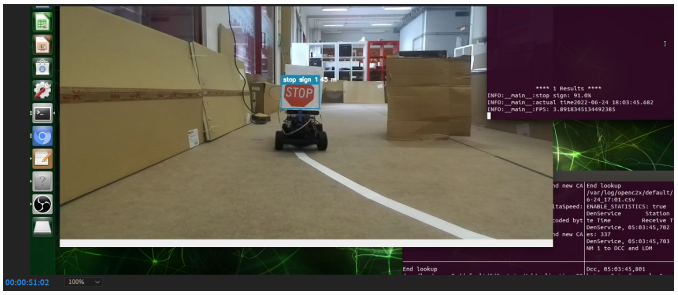


Fig. 10: Video frames to obtain detection-to-stop period.

4) Step 4: the OBU receives the DEN message.
5) Step 5: power to the wheels is cut.
6) Step 6: the vehicle comes to a halt.

All platforms were connected to a Network Time Protocol server to reliably collect timestamps. Timestamps of each step were collected in the following manner:

- Step 2: the YOLO software registers the time the vehicle is crossing the Action Point.
- Step 3: the RSU registers the time of sending of DENMs.
- Step 4: the OBU registers the time of DENM reception.
- Step 5: the vehicle ECU registers the time at which a command is sent to the physical actuators.

As for the overall time between step 1 and 6, one can look at the video recording of the road side camera and measure the interval between the crossing of the action point and the complete stop of the vehicle. Figure 10 depicts the instant the vehicle is reaching the Action Point. The processing is done at approximately 4 Frames per Second (FPS), so a small error margin on detection exists. In the video for run #4 [10], at 51:02 (S:ms), the vehicle crosses the 1.52m action point and is detected at 1.45m; and at instant 51:22 (S:ms), the car stops.

2) End-to-end Latency: The results of five test runs are shown on Table II. Communication between RSU/OBU represents a minimal part of the total time between the activation point and the deployment of the actuators, with an average time of 1.6 ms. On average, the time between step 2 and 5 was 58.4 ms and in none of the five runs exceeded 100ms, making it a very responsive system. The total time values from Figure II can be observed in Figure11, on the Empirical Distribution Function (EDF) graph. We observe that 60% of the samples occur between 44 and 55 ms, whereas the remaining 40% occur between 70 and 71 ms.

[10]https://youtu.be/0rC8FvmGA24https://youtu.be/0rC8FvmGA24

TABLE II: Time interval measurements

Interval between Steps	Run #					Avg.	Units
	#1	#2	#3	#4	#5		
#2: Action Point Detection #3: RSU sends DENM	34	27	27	21	29	27,6	(ms)
#3: RSU sends DENM #4: OBU receives DENM	1	2	2	1	2	1,6	(ms)
#4: OBU receives DENM #5: Vehicle Actuators	36	41	23	22	24	29,2	(ms)
Total Delay	71	70	52	44	55	58,4	(ms)

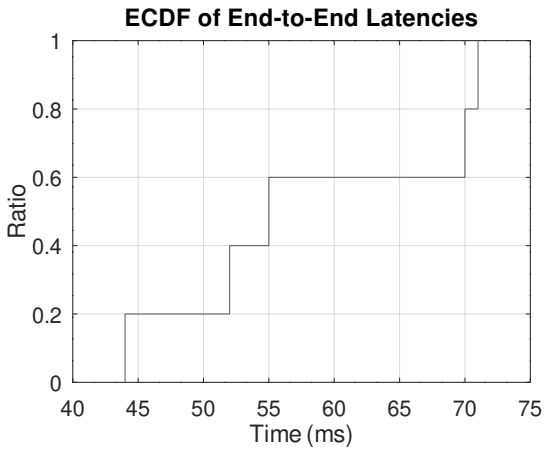


Fig. 11: Empirical distribution function of total time samples

TABLE III: Distance travelled from detection to halt

Run	#1	#2	#3	#4	#5	#6	#7
Braking Dist. (m)	0.43	0.37	0.31	0.42	0.31	0.36	0.36

B. Braking Distance

In the real world, a compelling metric for the collision avoidance application would be the braking distance. The previously mentioned *Action Point* is set as a reasonable threshold distance at the vehicle should initiate braking to avoid a collision; accordingly, the vehicle's *actual* braking distance should be within that threshold distance. In the context of the presented testbed, the relevance of these values may be limited but they can, in future, be related to the stopping power of full size vehicles, through dedicated models of vehicle inertia, floor-wheel friction and air drag.

The values for the seven test runs can be seen in Table III, The actual measurements were made with a measuring tape, forming a straight line from the ZED camera lens and the stop sign on the vehicle. The braking distance, based on the real measurements, was on average 36 centimeters with a variance

of 0.0022. The average braking distance is less than one vehicle length, that measures approximately 53 centimeters.

C. Discussion & Outlook

The measured end-to-end delay between the detection by the infrastructure and the actuation of the vehicle stopping systems is consistently under 100ms. Overall, the designed system proved to be a reliable and effective way to deliver an emergency message to a vehicle. The wireless equipment used in this context is the same as it would be used in the real-world, hence this work serves as a proof-of-concept ready to be deployed. Further work is required to properly model attenuation, either by interference or shadowing caused by own vehicle or others. The braking distance reveals, at this point, much less actionable information. Mapping the braking performance observed from the scale to real-world vehicles would allow to extract more insights. Using parameters of the full-size vehicles, such as stopping power, weight and frontal area, models can be drawn to map braking distances observed in the testbed to real-world ones.

The present exercise also shows the complexities of bringing together subsystems of different nature. We highlight particularly the object detection service, that brought challenges of its own such as the accurate identification of the scale vehicle. The end-to-end delay is, thus, not only dependent on the communication equipment but also of other subsystems that will be deployed on location and also require temporal characterization. Finally, observing the vehicle actually stopping (or not) brings immediate visual feedback as to whether the full processing chain is operating properly, thus conferring value to the testbed.

V. Conclusions and Future Work

The present testbed allows to test and validate a V2X application using actual communications equipment that mimics a real world scenario through the form of an emergency braking procedure informed by the network. A road-side infrastructure monitors a region-of-interest, and once a vehicle is detected in that region, ETSI ITS messages are sent by the infrastructure to trigger an emergency braking action by the vehicle. The obtained results demonstrate a system that is capable of detecting and communicating with a vehicle in a very short time span.

Future work will involve improving various aspects. We will carry out more measurements to produce a more comprehensive CDF of end-to-end latency, and possibly model it with an appropriate distribution so that it can be used by the community. We are currently installing a 5G module in the robotic vehicles, to compare the same detection-to-action delay over a different interface and network. We also plan to extend the testbed to support connected platoons (i.e., more robotic vehicles that are following each other), and evaluate the detection-to-action delay for the entire platoon. There is room to explore multi-technology solutions in this later case (e.g., platoon leader is 5G-capable while intra-platoon message forwarding is based on IEEE 802.11p), leading to a more complex arrangement that can impact detection-to-action delay.

References

[1] American Automobile Association, "Advanced Driver Assistance Technology Names," Tech. Rep., 2019.

[2] S. Hecker, D. Dai, and L. Van Gool, "Failure Prediction for Autonomous Driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2018, pp. 1792–1799.

[3] American Automobile Association, "Evaluation of active driving assistance systems," Tech. Rep., 2020.

[4] ETSI, "[Arch] ETSI EN 302 665 V1.1.1 - Intelligent Transport Systems (ITS); Communications Architecture," ETSI, Tech. Rep., Sep. 2010.

[5] ——, "[ITS-G5] ETSI EN 302 663 V1.3.1 - Intelligent Transport Systems (ITS); ITS-G5 Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band," Tech. Rep., Jan. 2020.

[6] S. Hasan, A. Balador, S. Girs, and E. Uhlemann, "Towards Emergency Braking as a Fail-Safe State in Platooning: A Simulative Approach," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. Honolulu, HI, USA: IEEE, Sep. 2019, pp. 1–5.

[7] B. Vieira, R. Severino, E. V. Filho, A. Koubaa, and E. Tovar, "COPADRIVe - A Realistic Simulation Framework for Cooperative Autonomous Driving Applications," in *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. Graz, Austria: IEEE, Nov. 2019, pp. 1–6.

[8] A. García Olmos, F. Vázquez-Gallego, R. Sedar, V. Samoladas, F. Mira, and J. Alonso-Zarate, "An Automotive Cooperative Collision Avoidance Service Based on Mobile Edge Computing," in *Ad-Hoc, Mobile, and Wireless Networks*, M. R. Palattella, S. Scanzio, and S. Coleri Ergen, Eds. Springer International Publishing, 2019, vol. 11803, pp. 601–607.

[9] D. Grewe, A. Tan, M. Wagner, S. Schildt, and H. Frey, "A real world information-centric connected vehicle testbed supporting ETSI ITS-G5," in *2018 European Conference on Networks and Communications (EuCNC)*, 2018, pp. 219–223.

[10] E. V. Filho, R. Severino, J. Rodrigues, B. Gonçalves, A. Koubaa, and E. Tovar, "CopaDrive: An Integrated ROS Cooperative Driving Test and Validation Framework," in *Robot Operating System (ROS): The Complete Reference (Volume 6)*, A. Koubaa, Ed. Springer International Publishing, 2021, pp. 121–174.

[11] L. M. Castiglione, P. Falcone, A. Petrillo, S. P. Romano, and S. Santini, "Cooperative Intersection Crossing Over 5G," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 303–317, Feb. 2021.

[12] E. V. Filho, N. Guedes, B. Vieira, M. Mestre, R. Severino, B. Goncalves, A. Koubaa, and E. Tovar, "Towards a Cooperative Robotic Platooning Testbed," in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. Ponta Delgada, Portugal: IEEE, Apr. 2020, pp. 332–337.

[13] M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio, and M. Bertogna, "F1/10: An Open-Source Autonomous Cyber-Physical Platform," no. arXiv:1901.08567, Jan. 2019.

[14] ETSI, "[Apps] ETSI TR 102 638 V1.1.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions," Tech. Rep., Jun. 2009.

[15] 5GAA, "C-V2X use cases volume II: Examples and service level requirements," Tech. Rep., 2020.

[16] T. I. ETSI, "[DENM] ETSI EN 302 637-3 V1.2.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service," ETSI, Standard, Sep. 2014.

[17] J. Matas, C. Galambos, and J. Kittler, "Robust Detection of Lines Using the Progressive Probabilistic Hough Transform," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 119–137, Apr. 2000.

[18] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Tech. Rep., Apr. 2018.

[19] S. Laux, G. S. Pannu, S. Schneider, J. Tiemann, F. Klingler, C. Sommer, and F. Dressler, "Demo: OpenC2X — An open source experimental and prototyping platform supporting ETSI ITS-G5," in *2016 IEEE Vehicular Networking Conference (VNC)*, 2016, pp. 1–2.

[20] B. Bloessl, "A Physical Layer Experimentation Framework for Automotive WLAN," Ph.D. dissertation, Paderborn University, 2018.

[21] H. Sand, "802.11p V2X hunting," 2019. [Online]. Available: https://harrisonsand.com/posts/802-11p-v2x-hunting/

[22] F. Raviglione, "Implementation of the IEEE 802.11p protocol on embedded systems," laurea, Politecnico di Torino, Dec. 2018.