Challenges of Remote Driving on Public Roads Using 5G Public Networks

Adrien Bellanger[a], Michael Klöppel-Gersdorf[b], Joerg Holfeld[c], Lars Natkowski and Thomas Otto[d]
Fraunhofer IVI Institute for Transportation and Infrastructure Systems, Dresden, Germany

Keywords: Teleoperation, Remote Driving, 5G Standalone Public Network, 5G Non-Standalone Public Network.

Abstract: Teleoperation, in the form of remotely controlling a vehicle (remote driving), is an important bridging technology until fully autonomous vehicles become available. Currently, there are manifold activities in this area driven by public transport companies, which implement solutions to offer first commercial teleoperation activities on the road. On the other hand, scientific reports of these solutions are hard to come by. In this paper, we propose a potential implementation for remote driving in 5G based public networks. We describe our insights from real world test drives on public roads and discuss possible challenges and suggest solutions.

1 INTRODUCTION

Teleoperating a vehicle (or remote driving) means remotely controlling its speed and steering. From our point of view, remote driving has two main application scenarios: The first one is to control a non-automated car over a whole ride e.g. redistribution of car sharing fleets, yard automation or allowing individual mobility for people which are not able or willing to drive (mobile work, child, alcohol consumption, disabilities, etc.), (Domingo, 2021). On the other hand, automated driving won't be able to cover all situations in the upcoming years. Remote driving potentially allows unmanned vehicles to continue their journey after entering a risk-minimal state. After we demonstrated in (Klöppel-Gersdorf. et al., 2023a) and (Klöppel-Gersdorf. et al., 2023b) the viability of remote driving in a 5G standalone (SA) campus network, this paper focuses on challenges of remote driving on public roads with 5G public networks.

The paper is organized as follows: after introducing the state of the art in remote driving, section 3 presents the hardware used by our demonstrator, and section 4 the software architecture. In section 5, we discuss our results and key findings. The paper concludes with an outlook in the last section.

2 STATE OF THE ART OF REMOTE DRIVING

In 2013, (Gnatzig et al., 2013) failed to show the feasibility of remote driving in public with latencies higher than 1 s due to the 3G networks. And also (Liu et al., 2017) proved that innovations introduced by Long Term Evolution (LTE) were not enough to drive remotely on public areas. (Kakkavas et al., 2022) provided a first show case on public roads using the current 5G technology. (Saeed et al., 2019; Kim et al., 2022) confirmed that 5G remote driving is possible at least if a 5G base station provides excellent coverage, and if the remote operators are positioned at locations with low latency network access (Zulqarnain and Lee, 2021).

Since a few years, demonstrations on sites with great coverage were shown from car manufacturers using tier-1 technology (Valeo, Bosch, etc.) in research projects as well as from car sharing providers (vay.io, halo.car, Elmo, etc.).

The consensus for commercialization aims at the integration of multiple networks to reduce congestion risks. According to (Ralf Globisch, 2023) using Low-Latency, Low-Loss and Scalable Throughput (L4S) would be enough to consider a single cellular provider, if L4S would be widely deployed.

After conducting an ISO 26262 (ISO 26262-10:2018, 2018) assessment, Vay launched at the beginning of the year 2024 its service commercially and

---

[a] https://orcid.org/0009-0002-8345-427X
[b] https://orcid.org/0000-0001-9382-3062
[c] https://orcid.org/0000-0002-1618-4241
[d] https://orcid.org/0000-0003-0099-9363

Bellanger, A., Klöppel-Gersdorf, M., Holfeld, J., Natkowski, L. and Otto, T.
Challenges of Remote Driving on Public Roads Using 5G Public Networks.
DOI: 10.5220/0012738600003702

In *Proceedings of the 10th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2024)*, pages 506-512
ISBN: 978-989-758-703-0; ISSN: 2184-495X

remotely drives vehicle of its car sharing fleet to customers in certain areas of Los Angeles[1].

Last, but not least, our own demonstrator showed the feasibility of remote driving on a closed area using a 5G SA campus network (Klöppel-Gersdorf. et al., 2023a) and (Klöppel-Gersdorf. et al., 2023b).

3 DEMONSTRATOR

The demonstrator's purpose is to control a car (see 3.1) using a remote driving station (see 3.2), where the car is connected over a 5G SA public network. The communication between both is facilitated by a server hosted in a Virtual Machine (VM) (see 3.3), which is accessible from both sides as described in Figure 1.

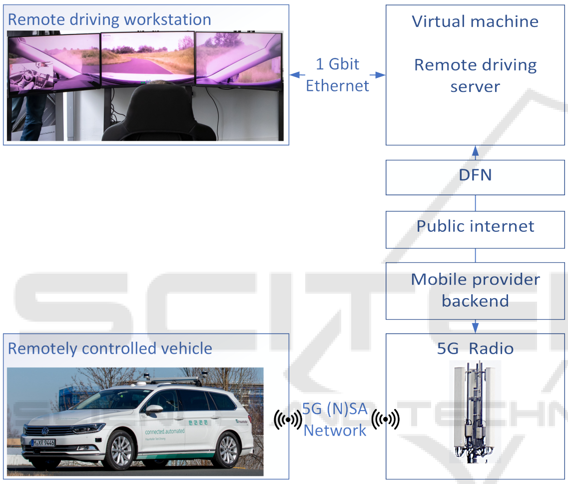


Figure 1: Main components of the remote driving demonstrator. The communication between radio and the VM is outside our sphere of influence and is depicted to show the general flow of information, especially in comparison with the remote driving demonstrator using campus network (Klöppel-Gersdorf. et al., 2023b). DFN denotes the German Science Network, a separate backbone connecting science institutions in Germany. This network interfaces the public internet at certain exchange points.

3.1 Remotely Controlled Vehicle

The test vehicle is the same as in (Klöppel-Gersdorf. et al., 2023b), where it was used on a test track in a 5G SA campus network. It is a Volkswagen Passat with automatic transmission modified by IAV GmbH to be able, among others, to control the set Adaptive Cruise Control (ACC) speed and the steering wheel angle using a custom Controller Area Network (CAN) interface.

The car computer (a Nuvo-9160GC PoE) is connected to the Internet over a 5G router (Mikrotik Chateau 5G). This computer features an Intel Core i7-13700TE processor, 32GB of RAM and an NVIDIA Quadro RTX A2000 graphics card.

The customer CAN (vehicle control) as well as the vehicle data CAN (telemetry) interfaces are accessed via a USB CAN Bus interface connected to the car computer. The transmission of those control and telemetry information is explain in section 4.2.

The visual information is captured by four AIDA Imaging HD-NDI-MINI cameras with a resolution of 1920x1080 pixels each, which were used in (Klöppel-Gersdorf. et al., 2023b) too. Three cameras are placed below the rear mirror to capture above 240° in the front direction, whereas the fourth is placed close to the rear window to provide a rear-view image.

We decided to use those cameras for their ability to encode videos streams on their own, such that the old car computer (see (Klöppel-Gersdorf. et al., 2023b)) was not a bottleneck anymore within the video stack.

The transmission of the video streams is detailed in section 4.1.

3.2 Remote Driving Station

The remote driving station consists of a workstation with three gaming monitors and a racing wheel (see Figure 2a).

The workstation consists of an Intel i7-12700 with 12 physical cores clocked at 4.90 GHz, 32 GB RAM and a NVIDIA RTX A5500 graphics card.

To facilitate remote driving, a Logitech G29 racing wheel (including pedals) is used, the wheel's force feedback is configured to center automatically.

3.3 Virtual Machine - Server

The VM is needed to forward video signals to the remote driving station (see 4.1), as well as receiving and distributing messages (see 4.2) in both directions between car and remote driving station.

As explained in the next section the VM runs MediaMTX, a TURN/STUN server and an MQTT broker. All of them do not need a lot of resources. In our case a VM with only one core 8 GB RAM is oversized, where the processor is an *AMD EPYC 7542* clocked at 2.9 GHz.

To reduce the whole latency, we place the VM as close as possible to the remote driving station (ping from remote driving station to VM < 1 ms).

---

[1] https://vay.io/press-release/vay-launches-commercial-driverless-mobility-service-with-remotely-driven-cars-in-las-vegas-nevada/

(a) Remote driver perspective.

(b) Driver perspective.

(c) Placement of the car on the track.

Figure 2: Challenges estimating car width and placement as a remote driver.

4 SOFTWARE ARCHITECTURE

In order to drive remotely, visual information has to be transmitted from the vehicle to the remote driving station (see 4.1) and the control command has to be transmitted from the workstation to the car (see 4.2). Figure 3 shows the data flow between all components.

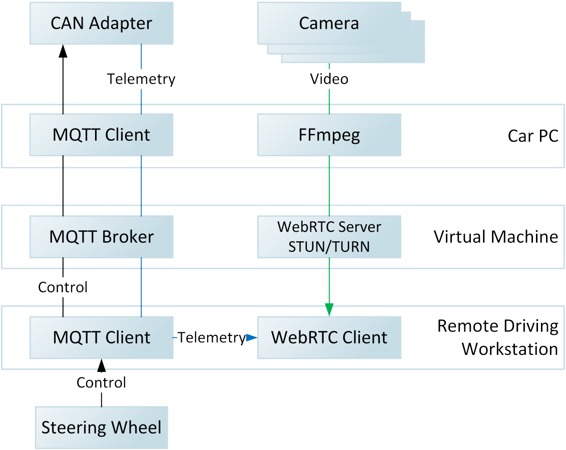


Figure 3: Data flow. Telemetry data is printed over the video stream (see Figure 2a), which requires that the WebRTC client is connected to an MQTT client.

4.1 Video Transmission

According to (Neumeier et al., 2019), remote driving with a glass-to-glass latency higher than 300 ms will not occur in decent conditions. Video streaming under such latency is usually named ultra-low latency, notice that a few publication use this term for latency under one second.

AIDA Imaging HD-NDI-MINI camera provides video streams encoded using H.264 or H.265, only Real Time Streaming Protocol (RTSP) allow acceptable latency. Notice that RTSP streams are only accessible in local network, and the remote driving station has no access to the car computer. To avoid an extra latency, the stream is forwarded to the VM as an RTSP stream over Transmission Control Protocol (TCP) using FFmpeg[2].

The RTSP port is managed by MediaMTX[3] which encapsulate the streams using WebRTC (Sredojev et al., 2015). MediaMTX provides one web page per camera. To allow the WebRTC to be displayed from other subnets the VM also run a TURN/STUN server (Coturn[4]).

To reduce the number of screens, we customized

---

[2] https://ffmpeg.org

[3] https://github.com/bluenviron/mediamtx

[4] https://github.com/coturn/coturn

the web page for the center camera with an overlay containing the current speed of the vehicle and the status of the acceleration and steering CAN interfaces.

A web browser is sufficient to display the visual information provided by the cameras with a decent latency, except when it is sandboxed (Klöppel-Gersdorf. et al., 2023b). Note, that not all of the tested RTSP players were able to play those streams without buffering (ffplay, VLC, totem...).

4.2 Control and Telemetry

The transmission of control and telemetry information is in its architecture close to how it was implemented in (Klöppel-Gersdorf. et al., 2023a) and (Klöppel-Gersdorf. et al., 2023b), where the Edge Cloud is replaced by the VM.

The remote driver station runs a Python script which monitors the Logitech G29 driving wheels state using PyGame 2[5]. It converts the position of both acceleration and brake pedal to a relative acceleration in the interval $[-1; 1]$, where negative values are for deceleration, the steering wheel angle is also converted to the interval $[-1; 1]$. Buttons on the wheel are also used to activate remote driving as well as turn lights.

Each state has its own Message Queuing Telemetry Transport (MQTT) topic, the messages are published to the MQTT broker installed on the VM (Eclipse Mosquitto[6]).

The car computer uses a Python script to control the vehicle and read telemetry information.

To get control messages, it subscribes to the control MQTT topics. The car is controlled by modifying CAN messages on a custom CAN interface, whereas telemetry information are read mainly from the vehicle data CAN interface. The interaction with the CAN bus occurs using python-can[7] and an USB CAN interface.

The ACC CAN interface allows to set a target acceleration in m^2/s, thus the script transforms the input normalized acceleration to an absolute acceleration. A linear transformation is used to get values in the interval $[-3\,\mathrm{m}^2/\mathrm{s};\ 2\,\mathrm{m}^2/\mathrm{s}]$. Consequently, when gas and brake pedals are released, the vehicle keeps its current speed. It would be possible to modify the transformation to simulate engine braking or one pedal driving, but our remote drivers appreciate the ability to stay at the current speed easily. It compensates the difficulties to assess the vehicle's speed on a video basis.

The steering CAN interface controls the steering wheel angle using the servo motors employed by parking and lane assistants. As the CAN interface is expecting values in degree in the interval $[-460°;\ 460°]$, like for acceleration a linear transformation is used.

Note that even the basic transformations from relative acceleration and steering values to absolute acceleration in m^2/s and steering wheel angle in degrees provide an acceptable remote driving feeling. One of the limitations of our implementation is that the acceleration interface will be disabled when the Anti-lock Braking System (ABS) has to prevent the wheels to block or Electronic Stability Program (ESP) detects loss of steering control. The feeling was good enough that after a few brake tests, the remote drivers were able to drive on our test track with black ice and snow to drive without activating the ABS and ESP including starting and stopping.

As explained in (Klöppel-Gersdorf. et al., 2023b) we try to reduce the cost of a future implementation, that's why we limit our telemetry usage to the current vehicle speed, and the state of acceleration and steering CAN interfaces. The telemetry is read using the Python script used to control the vehicle and published on different MQTT topics.

Figure 2a shows how it is displayed on the web page dedicated to the center camera.

5 RESULTS AND REMAINING CHALLENGES

Tests on public roads in Dresden using the 5G public network point unexpected and underestimated challenges to us, which we discuss in this section.

5.1 Location of the Remote Driving Work Place

As already noted by (Saeed et al., 2019), the remote driving work place should be close to the 5G communication infrastructure. This is not the case in our implementation, where the internet entry point of the 5G network is located in Frankfurt/Main (Germany), whereas the VM is hosted in Dresden (Germany) and the network traffic is routed via the Deutsches Forschungsnetzwerk[8]. This leads to an additional latency between 35 ms and 50 ms.

5.2 Cellular Radio Network Properties

According to our mobile network provider the area where we tested remote driving is fully covered by

---

[5] https://github.com/pygame/pygame/releases/tag/2.0.0

[6] https://mosquitto.org

[7] https://github.com/hardbyte/python-can

[8] https://www.dfn.de/en/network/

5G[9]. According to our tests (see Figure 4) a noticeable part of the track on public roads (about 5 km) was not decently covered by 5G by considering the Reference Signal Received Power (RSRP) of less than $-100\,\mathrm{dBm}$). Even when restricted to LTE, the limiting coverage during the test drives became obvious.

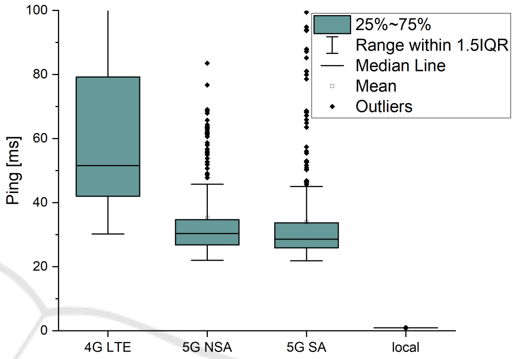
(a) 5G standalone coverage. (b) LTE coverage.

Figure 4: Measured RSRP[dBm] as network coverage in a range between $-120\,\mathrm{dBm}$ and $-50\,\mathrm{dBm}$ sketched from red to green.

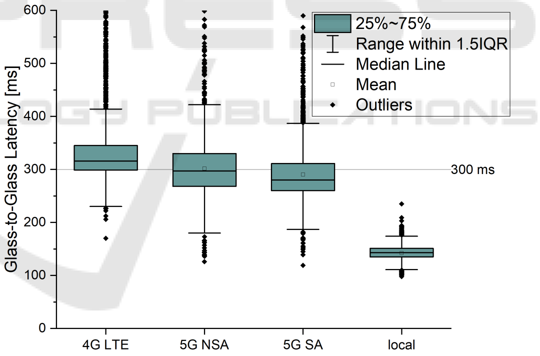
The used router from Mikrotik has besides an automatic choice the ability to force the radio access technology to LTE, 5G non-standalone (NSA) or 5G SA. This gives us the ability to separate the test drives with identical trajectories into each cellular radio technology. By traversing through the spanned radio cells, rapid handovers occur which are normally initiated by the radio network. But due to the higher frequencies of 5G cells, where our routers have been assigned to the n78 band at 3.5 GHz band, the entire coverage or cell size is smaller compared to lower frequencies. That results in more intra/inter-site handovers or even terminal induced and slow reconnections. Furthermore, this results in a higher number of large delays on ping level and applications level.

As shown in Figure 5a, the median ping latency has been observed at 50 ms while the 5G based radio access technologies delivered similar lower levels at 30 ms. The corresponding box plot is given to show the distributions.

Higher latency levels arise when additionally considering application or glass-to-glass latency (see Figure 5b). To measure the glass-to-glass latency, the rear-view camera was temporarily directed to a screen displaying a synchronized clock (using chrony[10]). The remote driving station runs a Python script which calculates the difference between the displayed remote clock and the workstation's synchronized clock. The script takes screenshot, and user optical character recognition (OCR) (tesserocr[11]) to retrieve the remote clock's value. Thereby, the entire median line can be considered at about 300 ms across all radio technologies. The local latency is presented for completeness in the outer right column to offer the lower possible bound by using a cable-based LAN. This means for our camera setup, at least 150 ms come from the video en-/decoding in conjunction with the TCP-based data packets.



(a) The ping statistics split into the radio technologies.



(b) The glass-to-glass latencies by radio technologies.

Figure 5: The ping and glass-to-glass latency statistics for the covered area, reduced to the north of rail tracks.

Finally, it should be mentioned that the required video bandwidth occupies the cellular uplink significantly, while most network provider optimize the traffic flow for streaming applications in the downlink. The vehicle's front camera consumes about 16 Mbit/s and two side cameras and the rear camera add three 4 Mbit/s streams.

---

[9] https://www.vodafone.de/hilfe/netzabdeckung.html, date: 21.02.2024

[10] https://chrony-project.org/

[11] https://pypi.org/project/tesserocr/

5.3 Camera Placement

The cameras directed through the left and right front windows, and which are displayed on the side screens are placed parallel to the windshield (see Figure 6). Its placement under the rear-mirror induce a smaller blind-spot for the left A-pillar, but when human drivers can move the head to move the blind-spot the remote driver can not move the cameras.

Figure 6: Placement of the 3 front cameras.

A second consequence of this placement is that under certain light conditions, and even using polar filters, the light reflection displayed to the remote driver can be disturbing (see side screens in Figure 2a). The problem does not appear on the center display because the camera is close to the windshield.

The solution used by Vay with cameras outside of the car has the advantage to tackle both problems but needs outdoor equipment.

5.4 HUD Necessity for Remote Driving

A Head-Up Display (HUD) is composed of graphic overlays on top of the remote driving station screen that present useful information to the remote driver. The already used overlays (see Figure 2a) are the vehicle speed and the status of steering and acceleration CAN interfaces, where the last two are mainly important for the startup routine.

Vehicle path guidelines should be added to the HUD as a new overlay. As shown in Figure 2, not leaving the lane while driving on public road is challenging for the remote drivers. It is mainly because the current lane takes just a small proportion of the entire screen in front of the car, whereas the vehicle fill the whole bottom of the center display. Thus the vehicle width is harder to estimate than for sitting behind the vehicle's steering wheel. Therefore, guidelines which indicate the future vehicle path would be a useful HUD overlay as demonstrated by Vay[12].

5.5 Encoding Latency

Even if the cameras used here are way faster than those we used in our first demonstrator (Klöppel-Gersdorf. et al., 2023a), the current glass-to-glass latency in a local network (ping <1 ms) is around 150 ms as shown in Figure 5b. The latency induced by encoding and decoding task using a stable 5G campus network was enough to stay under the 300 ms, but with the performance of our network provider, we probably need to upgrade the camera once again.

This latency can be considered as encoding time since the high-end graphic card of the decoding computer as well as its processor usages are very low. As it is not possible to access the RAW video output of these cameras, reducing the encoding latency means using other camera with an external encoder. Our new car computer's high-end graphic card can be used for this task (see 3.1). If it is not sufficient an upgrade to an expensive hardware encoder allowing glass-to-glass latency about 10 ms should be considered.

6 CONCLUSION

This paper presents a 5G public network remote driving demonstrator with its limits. The performance on our test track is satisfying, as the 5G public network connection from our network provider is sufficient there. But even on our test track we encountered some lags due to insufficient public network coverage. In areas with a sufficient coverage and a non saturated network a 5G SA campus network is not mandatory. The authors were not able to find a 5 km circuit on public roads in Dresden with a decent coverage, at least using our network provider.

The feasibility of remote driving on public roads using public networks is already shown with the first commercial usage. But neither Vay nor its competitors communicate the limits of their systems. From our point of view exploring the limits of an affordable demonstrator and communicating them is still needed to ensure feasible Operational Design Domain (ODD) for remote driving as well as to encourage the acceptance of commercially viable remote driving.

Areas of application for remote driving result from the sum of mobility needs from a transport perspective, the infrastructure design of roads and urban districts, which is described by the ODD, and last but not least the performance indicators of the mobile network to ensure a reliable communication between technical supervision and vehicle. Further research should define a set of network Key Performance Indicator (KPI) required to drive remotely in specific con-

---

[12] https://youtu.be/hcnRSedBDgU

ditions (weather, speed, city, highways, etc.) and certain ODD. That's why we plan to upgrade our demonstrator with new cameras (see 5.5) as well as extend our tests to other network providers and places (cities as well as countryside).

ACKNOWLEDGEMENTS

This research is financially supported by the German Federal Ministry for Digital and Transport (BMDV) under grant numbers FKZ 45FGU141_B (DiSpoGo) and FKZ 19FS2020_F (LAURIN) as well as co-financed by the Connecting Europe, Facility of the European Union (C-ROADS Urban Nodes).

REFERENCES

Domingo, M. C. (2021). An overview of machine learning and 5g for people with disabilities. *Sensors*, 21(22).

Gnatzig, S., Chucholowski, F., Tang, T., and Lienkamp, M. (2013). A system design for teleoperated road vehicles. In *ICINCO (2)*, pages 231–238.

ISO 26262-10:2018 (2018). ISO 26262-10:2018 Road vehicles – Functional safety – Part 10: Guidelines on ISO 26262. Standard, ISO.

Kakkavas, G., Nyarko, K. N., Lahoud, C., Kühnert, D., Küffner, P., Gabriel, M., Ehsanfar, S., Diamanti, M., Karyotis, V., Mößner, K., and Papavassiliou, S. (2022). Teleoperated support for remote driving over 5g mobile communications. In *2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pages 280–285.

Kim, J., Choi, Y.-J., Noh, G., and Chung, H. (2022). On the feasibility of remote driving applications over mmwave 5 g vehicular communications: Implementation and demonstration. *IEEE Transactions on Vehicular Technology*, pages 1–16.

Klöppel-Gersdorf., M., Bellanger., A., Füldner., T., Stachorra., D., Otto., T., and Fettweis., G. (2023a). Implementing remote driving in 5g standalone campus networks. In *Proceedings of the 9th International Conference on Vehicle Technology and Intelligent Transport Systems - VEHITS*, pages 359–366. INSTICC, SciTePress.

Klöppel-Gersdorf., M., Bellanger., A., and Otto., T. (2023b). Identifying challenges in remote driving. *Submitted to 9th International Conference, VEHITS 2023, Revised Selected Papers*.

Liu, R., Kwak, D., Devarakonda, S., Bekris, K., and Iftode, L. (2017). Investigating remote driving over the lte network. In *Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '17, page 264–269, New York, NY, USA. Association for Computing Machinery.

Neumeier, S., Wintersberger, P., Frison, A. K., Becher, A., Facchi, C., and Riener, A. (2019). Teleoperation: The holy grail to solve problems of automated driving? sure, but latency matters. pages 186–197.

Ralf Globisch, J. G. T. (2023). Deep dive: How to break the congestion barrier – achieving low latency with high throughput for safe teledriving. Technical report, Vay Technology GmbH. https://vay.io/how-to-break-the-congestion-barrier-achieving-low-latency-with-high-throughput-for-safe-teledriving.

Saeed, U., Hämäläinen, J., Garcia-Lozano, M., and David González, G. (2019). On the feasibility of remote driving application over dense 5g roadside networks. In *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*, pages 271–276.

Sredojev, B., Samardzija, D., and Posarac, D. (2015). Webrtc technology overview and signaling solution design and implementation. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1006–1009.

Zulqarnain, S. Q. and Lee, S. (2021). Selecting remote driving locations for latency sensitive reliable teleoperation. *Applied Sciences*, 11(21).