https://doi.org/10.31891/csit-2025-3-6
UDC 004.942:656.1:519.876.5

Oleksandr BYZKROVNYI, Serhii SMELYAKOV
Kharkiv National University of Radio Electronics

THE SOFTWARE DESIGNING METHOD FOR THE MODEL OF PREDICTION POTENTIAL ROAD TRAFFIC ACCIDENCE

This research focuses on developing a software designing method capable of delivering real-time notifications about potential car accidents to Internet of Things (IoT) devices installed in vehicles. Since vehicles are mobile and frequently change location, there is a need to have stable and timely notifications. To address this, the proposed designing method leverages cloud technologies, which offer scalability, low latency, and high availability for real-time data transmission.

The main objective is to design an architecture that enables effective real-time messaging between cloud services and moving IoT devices. The system is tested across various simulated driving scenarios to evaluate its performance and reliability. A key aspect of the methodology is measuring the time interval that begins when the message is sent and ends when it is received by the IoT device, followed by assessing whether the driver has adequate time to respond appropriately.

The experimental setup is based on test cases created using Python, the BeamNG.tech simulation environment, and AWS IoT Core as the cloud service provider. The results demonstrate that the proposed designing method can reliably handle real-time messaging in dynamic conditions. As a conclusion, the research confirms the potential of the proposed designing method for real-life applications in connected vehicle safety systems.

Keywords: Software designing method, real-time messaging, cars accident prediction, IoT, computer vision, machine learning, artificial intelligence, message routing optimization.

Олександр БИЗКРОВНИЙ, Сергій СМЕЛЯКОВ
Харківський національний університет радіоелектроніки

МЕТОД РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОГНОЗУВАННЯ ПОТЕНЦІЙНИХ ДОРОЖНЬО-ТРАНСПОРТНИХ ПРИГОД

Ціль дослідження – створення методу проектування програмного забезпечення, здатного надсилати сповіщення про потенційні дорожньо-транспортні пригоди до IoT пристроїв, встановлених у транспортних засобах, в режимі реального часу. Враховуючи мобільність об'єктів та часту зміну їхнього просторового положення, необхідно підтримувати стабільне та своєчасне надання повідомлень про можливі ДТП. Для цього запропоновано використання хмарних технологій, які забезпечують масштабованість, низьку затримку та високу доступність передавання даних у реальному часі.

Основна мета полягає в розробці архітектури, яка дозволяє ефективно здійснювати обмін повідомленнями між хмарними сервісами та рухомими IoT-пристроями. Систему протестовано в різних змодельованих дорожніх ситуаціях з метою оцінки її продуктивності та надійності. Використана методологія — вимірювання інтервалу часу між моментом надсилання повідомлення та його отриманням IoT-пристроєм, а також аналіз, чи мав водій достатньо часу для своєчасної реакції.

Експериментальне середовище базується на тестових сценаріях, створених за допомогою Python, симулятора BeamNG.tech і хмарних сервісів AWS IoT. Результати підтверджують, що запропонований метод до проектування інформаційної технології здатний працювати в умовах реального часу. Як висновок, дослідження засвідчує перспективність використання цієї розробки в реальних умовах для підвищення безпеки руху автомобілів.

Ключові слова: Розробка архітектури програмного забезпечення, обмін повідомленнями в режимі реального часу; передбачення ДТП, IoT, комп'ютерний зір, машинне навчання, штучний інтелект, оптимізація маршрутизації повідомлень.

Received / Стаття надійшла до редакції 07.06.2025
Accepted / Прийнята до друку 22.07.2025

Introduction

Car road crashes are a big challenge in the European Union (EU), significantly impacting human lives and incurring considerable economic costs. Recent EU statistics indicate that while there has been significant drop in road fatalities over the past 20 years, the road safety situation remains critical, pushing ongoing research and intervention efforts. In 2021, approximately 19,800 people lost their lives in road traffic accidents across the EU, which is about 44 deaths per million inhabitants [1]. Despite various road safety measures implemented, certain demographics, particularly younger males, still have high crash rates, highlighting the need for targeted strategies aimed at these vulnerable groups [2].

The multifactorial nature of road traffic accidents includes human behavior, environmental conditions, and infrastructure quality. Recent studies illustrate the role of advanced driver assistance systems (ADAS) in mitigating human errors, which are the cause of up to 90% of crashes [3]. Notably, the efficiency of such technologies in enhancing roadway safety is supported by empirical data, as evidenced by research indicating that effective ADAS implementations lead to decrease in incidents involving vulnerable road users such as pedestrians and cyclists [3]. Furthermore, ongoing analyses using geographic information system (GIS) methodologies reveal how the spatial distribution of accidents can inform infrastructure improvements and targeted interventions, particularly in urban environments where crash hotspots have been identified.

Highlighted above points reveal the necessity of creating a software, which have possibility to predict the possible car collision, and as crucial point here: the timely and correctly deliver this finding to the drivers which are identified by system as involved in potential collision. The most crucial point here is delivering such notification to the IoT devices (cars) in real-time, which frequently change the location.

Related Works

Real-time communication with Internet of Things (IoT) devices is an important part of building smart and responsive environments. This type of communication depends on different technologies and protocols that help devices share data instantly, making sure they work well together.

One key method that improves real-time communication is called Device-to-Device (D2D) communication. This allows devices to talk directly to each other without needing a central system or server. This is especially useful in smart cities, where many devices are moving and growing in number [4]. D2D helps reduce delays in sending data and uses network resources more efficiently, which makes communication faster and more stable.

Another important factor is wireless communication. In many IoT systems, devices use very little power, so the wireless connection must be strong and steady. Studies show that good wireless channels help devices send and receive data without breaks, which is important for systems that run automatically [5]. If communication is interrupted, the system might not work properly, and the data might become incorrect [6]. To solve this, some researchers use methods like delta encoding. This compresses and protects data before sending it, which helps save energy and speeds up communication [7].

The MQTT (Message Queuing Telemetry Transport) protocol is another helpful tool for IoT communication. It is made for simple, fast messaging in systems with limited power or memory. MQTT uses a publish-subscribe model, where devices can send or receive messages quickly without always asking each other directly [8]. This model improves real-time messaging and helps devices work together smoothly.

Data security is also very important in real-time communication. When devices send messages to each other, they must make sure the data stays private and correct. New methods like using blockchain help protect this data and confirm that messages are real [9]. These solutions are especially important today, as security problems can cause serious damage to IoT systems.

In conclusion, real-time messaging in IoT depends on fast communication methods, strong wireless connections, and safe data handling. Using D2D communication, smart data management, and secure protocols like MQTT helps build better systems. These systems allow smart devices to connect and work effectively in different areas, such as homes, cities, or industries. But at the same time none of the proposed methods can fully cover the needs of real-time messaging from one IoT device to another ones, where receivers frequently change own location. Additionally, information technology should have high horizontal scaling, convenient management of thousands of IoT receivers' devices and organic messages routing mechanisms to avoid usage of additional services, which may reduce performance of message delivery.

Purpose

The purpose of this article is to create a designing method of software, which allows delivery of real-time messages to the end users in a way that end user (driver) have enough time to react on the received warning-message. Additionally, proposed designing method should support high horizontal scaling, and the message routing should be done with minimum usage of different third-party services to decrease the time for notifications delivery.

Methodology

The proposed designing method of software works in couple with the model for detecting potential cars' collision, which is deployed on single board computer (Nvidia Jetson Orin). Information technology in that case performs the role of mediator or warning messages delivery system. Fig. 1 explains the general idea of information technology use case.

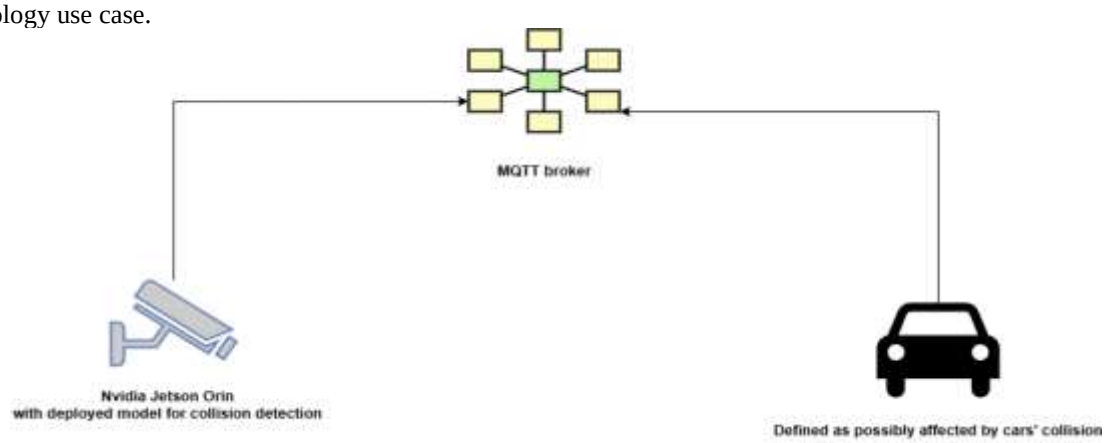

Fig. 1. Information technology is used as a warning messages delivery system

There is the question: “How to identify which car (IoT) the warning-message should be sent to?”. The Nvidia Jetson device knows nothing about cars which are in the field of its view. It is just the vehicle-like objects which can be intersected, due to the trajectory projections of these cars are intersected.

The preliminary answer may be the storing of car’s plate number and then send the warning message directly to this car’s driver. That method may not work well, due to the additional complexity of necessity to have additional model for plate number recognition, then sending message directly to IoT device is a task which adds unnecessary performance degradation. Also, the plate number may be not visible in all cases, then driver will not be notified at all about the possible crash.

Consequently, the method which allows publishing and subscribing to some intermediaries is the best solution, because the warning message will be received in any case, if the IoT device is subscribed. For that purposes the MQTT broker can be used as the mediator between Nvidia Jetson devices and end users – IoT devices installed in cars. Fig. 2 displays the simplified components diagram.

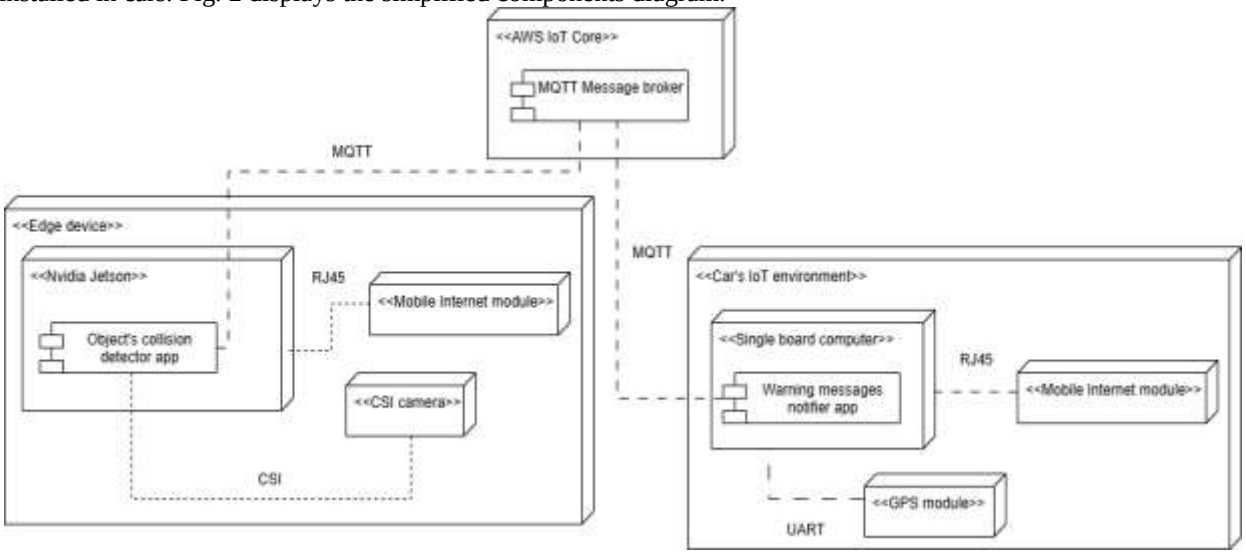

Fig.2. Components diagram of the designing method for proposed software

At the same time another issue appeared, and the main question is not resolved. How the warning-messages should be routed to the end user? The MQTT broker topics usage may help on that. The end-user IoT devices can subscribe to different topics and route messages there. The topics may be in that case some location: city, route, building number. Such method is also not convenient because it creates a lot of complexity in message routing. Different countries may have different schemas in naming principles and this lead to usage of different MQTT brokers or topics naming schemas for different countries or even cities. The topics management in that case is a very inconsistent and difficult task.

Therefore, the usage of unified method for location encoding which allows route the message to the right topic should be used. The GeoHash algorithm [10] is the place to use. The GeoHash allows divide the total map into equal cells, by specified precision, this allows to define the rectangle by any scale on any place in the world and do not depend on city, village, highway naming conventions.

Additionally, this gives the ability to decrease the number of messages received by particular end users’ IoT. The IoT device subscribes to the specific topic which denotes the exact location area, where IoT device is on. So, the other messages, which come from another area, which is out of captured location cells, go into different topic and will be received by the IoT devices, which are subscribed there. This method provides the native way of messages routing without usage the third-party services or rules in MQTT message broker. Additionally, such method gives the mechanism for reducing message delivery latency, so fewer users are subscribed to the topic, then less latency for message delivery will be. The Fig.3 provides the flow chart of subscribing and receiving warning messages by IoT device installed in the car.

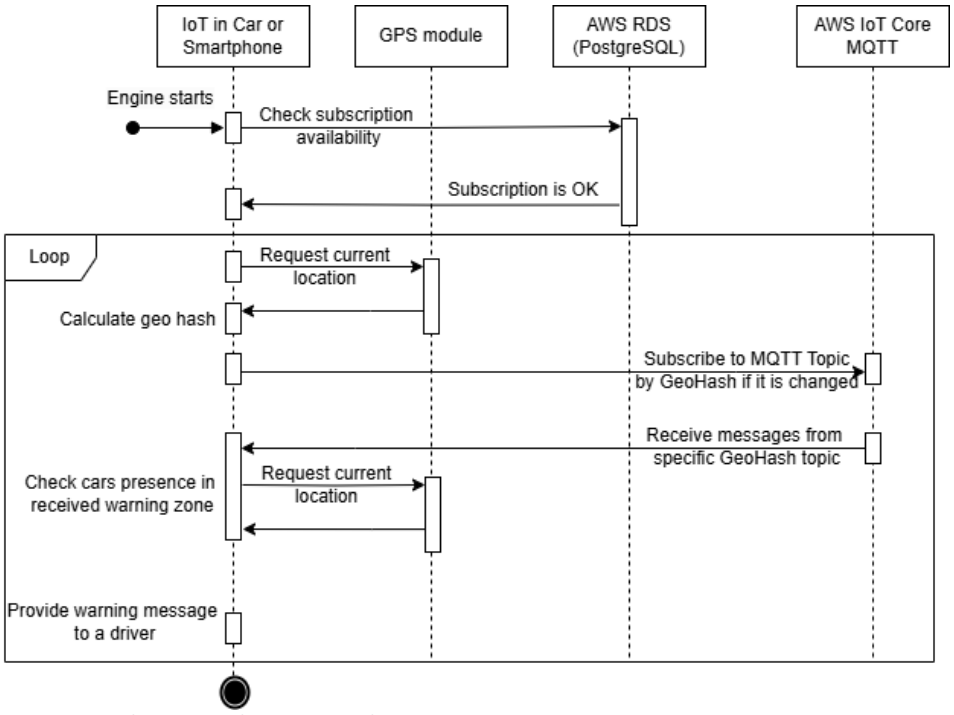

Fig.3. The flow chart of the process of subscribing and receiving warning messages by IoT device

As can be seen in fig. 3 the step for checking exact location of IoT device is present. This is the step which allows displaying message to the exact cars' drivers which are considered as potentially impacted by cars' collision. Once the message appeared in the MQTT topic (fig.4).

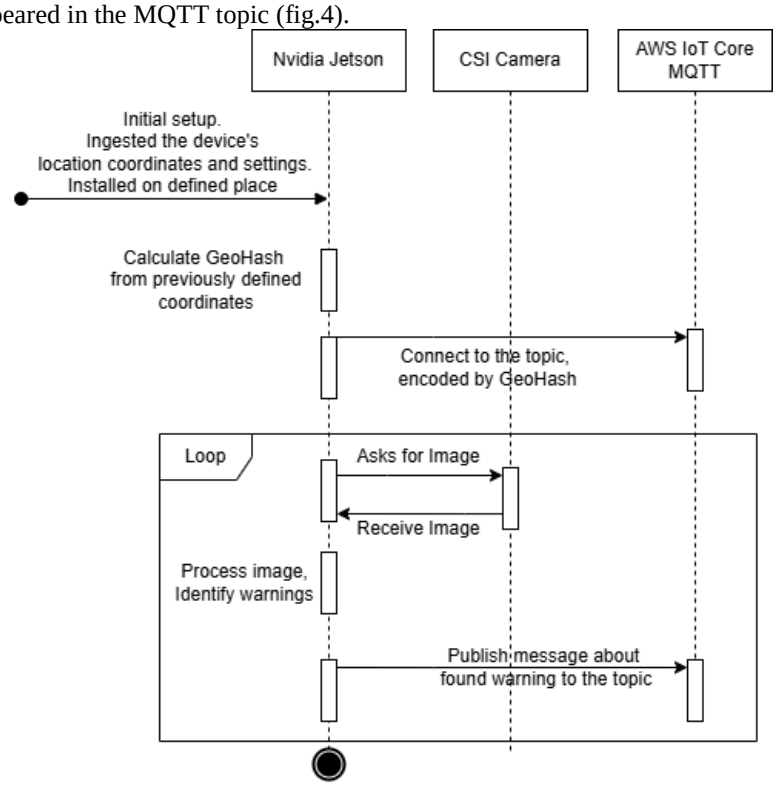

Fig.4. The flow chart of the process of publishing warning message to the MQTT topic

This message is read by all IoT devices which are present and subscribed to the topic of the location which can be considered as single cell on the grid by GeoHash algorithm.

After that, the message parsing happened on the IoT device. The received message contains the declared warning-polygon, which is denoted as a set of coordinates latitude and longitude values. In case if IoT device is within the specified polygon, then message is displayed to driver. The algorithm by which IoT device calculates the presence inside of the polygon explained in fig.5.

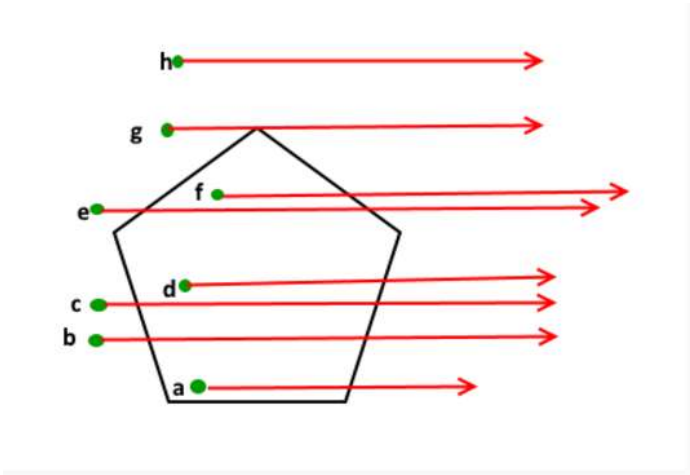

Fig.5. Ray-Casting algorithm explanation [11]

Using that designing method, extremely fast messages routing and highly scalable software system is designed.

Results

The experiment is based on simulated scenarios of potential road accidence created using BeamNG.tech platform [12] and Python, Jetson Nvidia, AWS IoT Core, AWS MQTT message broker.

The test scenario explains the case when two cars are going in the same direction, but due to some obstacles or landscape form, both drivers do not see each other with direct view. The case explained in Fig. 6.

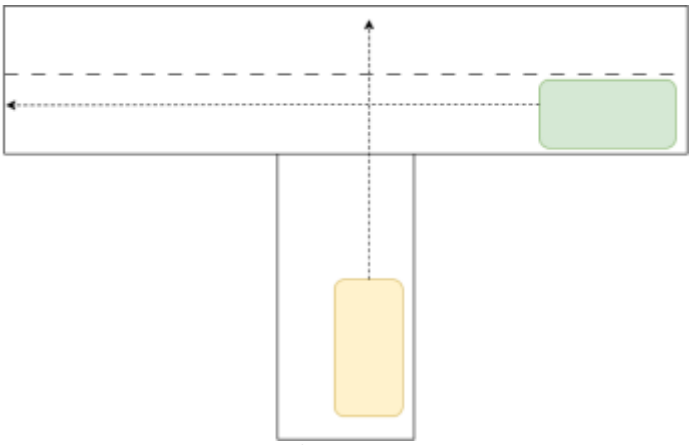

Fig. 6. Basic scenario of car accidence on the crossroads

Tables 1 and 2 represent the results of the experiment, including the recorded time of each stage and a binary indication of whether a crash occurred. The vehicle used in the study described in Table 1 was a 1425-kilogram sedan equipped with a conventional braking system.

Table 1

Experiment result

Time before message is sent	Time, when message is received by application in car	Time, when car fully stopped	Speed at the moment of message receiving; km/h	Did a car accident happen?
2025-01-05T12:40:02.709622	2025-01-05T12:40:02.933484	2025-01-05T12:40:05.114334	42	TRUE
2025-01-05T13:08:56.834051	2025-01-05T13:08:57.076655	2025-01-05T13:08:58.772655	37	FALSE
2025-01-05T13:09:27.781746	2025-01-05T13:09:28.030199	2025-01-05T13:09:29.102009	*51.38*	FALSE
2025-01-05T13:09:57.748407	2025-01-05T13:09:57.994106	2025-01-05T13:09:59.253262	52.67	TRUE

The results of the first experiment indicate that the system works reliably when the vehicle speed remains at or below 50 km/h for the tested vehicle type. The average delay in publishing and receiving the warning message was about 0.3 seconds, which is considered as low enough to provide the driver additional time to react. Under these

conditions, observed braking reaction times ranged from 1.7 to 2 seconds. Despite this, collisions occurred in approximately 50% of the attempts. The main reasons for these collisions were the delay in object recognition by the model and the temporary loss of target tracking during the threat detection phase [13]. The example of how system works is displayed in Fig. 7.

Fig. 7. Displaying of how possible car accidence classification works

The next scenario involves network delays and an additional one-second delay during car's location identification. This reflects the situation in which the GPS module within the vehicle's IoT device operates at a maximum frequency of 1 Hz. The tested vehicle was a station wagon with a weight of 1660 kg, equipped with a high-performance braking system featuring three-piston calipers.

Table 2

Test case with delay in message delivering and cars' position negotiation

Time before message is sent	Time, when message is received by application in car	Time, when car's IoT found its own location	Time, when car fully stopped	Speed at the moment of message receiving km/h	Did a car accident happen?
2025-01-23T20:12:06.217046	2025-01-23T20:12:08.000562	2025-01-23T20:12:09.405625	2025-01-23T20:12:11.594017	33	FALSE
2025-01-23T20:12:22.577780	2025-01-23T20:12:24.374097	2025-01-23T20:12:25.695778	2025-01-23T20:12:27.176777	44	FALSE
2025-01-23T20:12:51.205166	2025-01-23T20:12:52.986657	2025-01-23T20:12:54.387525	2025-01-23T20:12:57.265166	60	FALSE
2025-01-23T20:14:06.713777	2025-01-23T20:14:08.538826	2025-01-23T20:14:09.900823	2025-01-23T20:14:13.012524	62	TRUE
2025-01-23T20:15:51.263816	2025-01-23T20:15:53.064392	2025-01-23T20:15:54.459391	2025-01-23T20:15:57.064110	64	FALSE
2025-01-23T20:17:07.964313	2025-01-23T20:17:09.752505	2025-01-23T20:17:11.133271	2025-01-23T20:17:12.562308	68	TRUE

This experiment takes about 2 seconds to send and receive a message, 1 second to locate the car, and 2-3 seconds to stop the car. As it can be seen, if the speed of the car is more than 60 km/h, the probability of an accident is much higher. Also, during the testing of this scenario, it was quite noticeable that the warning message is delayed, and the car that should respond to the message, stops far beyond the limits of the case when there are no delays (case 1). Fig 8 displays the difference between cases.

Fig. 8. Explanation of differences between cases with no delays and added delay

Fig.8 displays the lines on which the cars are stopped for two cases.

The stop line which marked as 1 – this is the case, where delay for message publishing and receiving is 0.3 seconds, and the car's speed is lower than 60 km/h. The car's location finding takes there 0.1 seconds.

The stop line which marked as 2 – this is the case, when delay between message publishing and receiving is 2 seconds, the car's speed is lower than 60 km/h and car's location finding takes 1 second.

So, based on the experiment, that delay for the second case is significant for information technology workability.

Conclusions

The article proposes the software designing method for real-time messaging between IoT devices for the task of notifying drivers about possible car accidence. This technology involves usage of publishers/subscribers in AWS MQTT message broker and native way for messages routing based on GeoHash algorithm. Experiments show fair results, even if the additional delay is in place. The real-time notification software system or method in general can be used also for other needs, for example, medical. There are cases when the timely notifications may have a crucial role on the patient's life. Diabetic patients require real-time glucose level monitoring, which helps in prediction of the crisis and notify patients and his doctor about that. The usage of optimized neural network for low resource devices should be in place. Current research is based on optimized neural network for cars classification, which can be taken in consideration, and such optimization can be applied for low resource devices too. Under the term of low resources devices, I mean smartphones, some IoT devices, which should process data from sensors in real-time and send notifications to doctor and user about short coming crisis.

References

1. Karkalakos S., Tselekounis M. Road safety and sustainability performance: A cross-country analysis. E3S Web of Conferences. 2023. Vol. 436. P. 11005. URL: https://doi.org/10.1051/e3sconf/202343611005 (date of access: 18.05.2025).
2. Evaluating the effectiveness of speed humps and rumble strips in improving pedestrian safety in Addis Ababa, Ethiopia / G. S. Tulu et al. PLOS One. 2025. Vol. 20, no. 4. P. e0318088. URL: https://doi.org/10.1371/journal.pone.0318088 (date of access: 18.05.2025).
3. Isaksson Hellman I., Lindman M. Estimating the crash reducing effect of Advanced Driver Assistance Systems (ADAS) for vulnerable road users. Traffic Safety Research. 2023. Vol. 4. P. 000036. URL: https://doi.org/10.55329/blzz2682 (date of access: 18.05.2025).
4. A Multi-Constraints Routing Scheme for MANET-assisted IoT in Smart Cities / Q. Vu Khanh et al. EAI Endorsed Transactions on Industrial Networks and Intelligent Systems. 2023. Vol. 10, no. 2. P. e5. URL: https://doi.org/10.4108/eetinis.v10i2.3388 (date of access: 18.05.2025).
5. Arif S., Khan M. A., Rehman S. U. Wireless Channel Estimation for Low-Power IoT Devices Using Real-Time Data. IEEE Access. 2024. P. 1. URL: https://doi.org/10.1109/access.2024.3359170 (date of access: 18.05.2025).
6. Shen L. Exploring the application and performance of extended hamming code in IoT devices. Applied and Computational Engineering. 2024. Vol. 32, no. 1. P. 71–76. URL: https://doi.org/10.54254/2755-2721/32/20230186 (date of access: 18.05.2025).
7. Jindal R., Kumar N., Patidar S. IoT streamed data handling model using delta encoding. International Journal of Communication Systems. 2022. URL: https://doi.org/10.1002/dac.5243 (date of access: 18.05.2025).
8. Interactive Task Assignment Model for Internet of Things Devices / T.-C. Hsu et al. Sensors and Materials. 2019. Vol. 31, no. 6. P. 1815. URL: https://doi.org/10.18494/sam.2019.2254 (date of access: 18.05.2025).
9. Blockchain based security protocol for device to device secure communication in internet of things networks / P. Chandrakar et al. SECURITY AND PRIVACY. 2022. URL: https://doi.org/10.1002/spy2.267 (date of access: 18.05.2025).
10. GeohashTile: Vector Geographic Data Display Method Based on Geohash / C. Zhou et al. ISPRS International Journal of Geo-Information. 2020. Vol. 9, no. 7. P. 418. URL: https://doi.org/10.3390/ijgi9070418 (date of access: 18.05.2025).
11. GeeksforGeeks. How to check if a given point lies inside or outside a polygon? - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/how-to-check-if-a-given-point-lies-inside-a-polygon/ (date of access: 18.05.2025).
12. BeamNG.tech C. BeamNG.tech. BeamNG.tech. URL: https://beamng.tech/ (date of access: 23.03.2025).
13. Comparison of Object Detection Algorithms for the Task of Person Detection on Jetson TX2 NX Platform / O. Byzkrovnyi et al. 2024 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream), Vilnius, Lithuania, 25 April 2024. 2024. URL: https://doi.org/10.1109/estream61684.2024.10542592 (date of access: 18.05.2025).

Oleksandr Byzkrovnyi **Олександр Бизкровний**	PhD Student of the Department of Software Engineering. Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, e-mail: oleksandr.byzkrovnyi@nure.ua. https://orcid.org/0000-0001-9335-442X	асп. каф. програмної інженерії, Харківський національний університет радіоелектроніки, Харків, Україна.
Serhii Smelyakov **Сергій Смеляков**	Doctor of Mathematics, Professor, Professor of Software Engineering department, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, e-mail: serhii.smeliakov@nure.ua http://orcid.org/0000-0002-5791-2479	д-р фізико-математичних наук, проф., проф. каф. програмної інженерії, Харківський національний університет радіоелектроніки, Харків, Україна