 https://doi.org/10.24191/jeesr.v27i1.019

Telemetry Website for Vehicle-to-Vehicle and Vehicle-to-Infrastructure Communication

Muhammad Hakimi Aiman Hadri, Fazlina Ahmat Ruslan and Juliana Johari*

Abstract— **Intelligent Transportation Systems (ITS) highlights the need for efficient Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication. This research establishes a real-time telemetry framework based on Message Queuing Telemetry Transport (MQTT) operating through WebSocket to enable rapid data transmission between vehicle units, traffic light units, and a web-based monitoring system. The system combines GPS data processing, real-time vehicle tracking, and traffic light status updates to display information through a web-based monitoring system that integrates Google Maps API for map interface. The backend of the web development utilizes GPS coordinates to determine vehicle speed before storing structured telemetry data in an online database by using InfluxDB, a time-series database for historical analysis. Performance evaluation of the telemetry system focuses on MQTT protocol efficiency, WebSocket latency, and GPS integrity to ensure a robust and reliable V2I telemetry system. Existent literature indicates that MQTT communication protocol, particularly with QoS 1, ensures efficient and reliable telemetry with minimal delay, it is expected to deliver the benefit to this telemetry system. Additionally, the study highlights that increased vehicle speeds correlate with higher GPS positioning errors, necessitating improved filtering techniques for enhanced positional accuracy. Given these findings, this research explores the requirement for robust V2I communication frameworks in ITS, paving the way for future advancements through machine learning-based predictive modeling, 5G connectivity, and edge computing for enhanced processing efficiency. By addressing critical gaps in real-time telemetry, this study offers a roadmap to the development of scalable and efficient ITS solutions that support urban traffic optimization.**

Index Terms— **GPS, intelligent transportation systems, telemetry, MQTT, V2V, V2I, vehicle tracking.**

I. INTRODUCTION

Real-time communication in Intelligent Transportation Systems (ITS) enhances traffic management and road safety by enabling instant data transmission between vehicles and infrastructure through Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication that allows vehicles and traffic control systems to exchange telemetry data, optimizing traffic flow and accident prevention. However, ITS currently faces challenges in transmitting real-time telemetry data due to network reliability issues and limitations in integrating with traffic infrastructure. Mari et al. [2] categorized traffic control systems into fixed-time and adaptive operation, with adaptive systems requiring reliable and low-latency connections to operate effectively. Sensor-based controllers which power adaptive traffic management experience rising communication errors as time progresses [7] while static traffic control faces difficulties in adapting to changing traffic patterns. Real-time ITS applications require communication methods that deliver fast responses while maintaining reliability which conventional methods struggle to achieve.

Conventional ITS systems depend on cellular networks, Wi-Fi, and DSRC for communication, yet these technologies create performance challenges related to latency, scalability, and reliability. The MQTT messaging protocol addresses these challenges through its ability to provide low-latency and reliable telemetry communication across any network [5]. Amelia et al. [1], Kaskatiiski and Boyanov [3], and Selimović et al. [4] demonstrated MQTT efficiency in managing multiple connections, delivering rapid messages, and scaling effectively in large networks. The research of Živić et al. [5] and Gruener et al. [6] demonstrated how MQTT delivers reliable low-latency performance for ITS applications. The combination of MQTT protocol features makes it an optimal solution for ITS systems that need effective vehicle-to-traffic infrastructure data exchange. Beyond MQTT's role in ITS telemetry systems, researchers have explored both web development methodologies and vehicular communication technologies for data visualization, vehicular positioning, and behavior analysis. The authors Antunes and da Fonseca [8] developed a flexible web framework that combined React and Node.js to deliver real-time data visualization through GraphQL-based data structure management. This research demonstrates how to develop telemetry interfaces that scale for traffic monitoring systems. Mendes et al. [9] investigated V2X communication through 5G networks to show how hybrid 5G-GNSS systems improve vehicular positioning accuracy and cooperative driving performance. Shushkova et al. [10] developed an algorithm to extract synthetic accelerometer and gyroscope data from GPS readings which enhances driver behavior analysis capabilities. Their approach delivers better tracking results for vehicle systems that do not use motion sensors in ITS applications.

Although prior studies have examined various aspects of

This manuscript was submitted on 25th February 2025, revised on 19th April 2025, accepted on 23rd April 2025, and published on 31st October 2025.
Muhammad Hakimi Aiman Hadri, Fazlina Ahmat Ruslan and Juliana Johari are with the Faculty of Electrical Engineering, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia.

*Corresponding author
Email address: juliana893@uitm.edu.my

ITS and communication technologies, these papers mainly focused on independent elements such as traffic algorithms, communication protocols, data visualization, and positioning techniques. Research regarding the MQTT protocol mostly examines its performance against HTTP and CoAp alternatives and broker performance across various operational parameters. While these studies demonstrate MQTT's low-latency and scalable capabilities, its integration into real-time ITS telemetry systems remains insufficient in the existing literature and lacks comprehensive implementation frameworks. For ITS-related papers, studies that propose fully integrated systems—combining real-time data exchange, visualization, and long-term data handling—remain relatively uncommon. The current research on web frameworks for traffic monitoring focuses on frontend scalability but lacks integration with live V2X data streams. The domain lacks proper utilization of real traffic data such as Google Map API and cloud-based time-series storage solutions like InfluxDB. The literature shows a significant research gap because there is a requirement for a single telemetry system that combines MQTT over WebSockets with modern web and database technologies to achieve effective V2V and V2I communication within ITS environments.

In the context of integrating ITS telemetry systems into the current technology, several challenges need to be resolved—especially the communication network. Widely available cellular networks are susceptible to congestion in dense urban environments. Wi-fi has a limited range and does not meet reliability standards for ITS settings. DSRC, while specialized for vehicular communication, has limitations in range and interoperability [9]. These limitations will limit the long-term progression of real-time telemetry systems in dynamic traffic environments. In response, adopting MQTT over Websockets provides the fundamental communication method that enables low-latency, scalable, and reliable communication across various network infrastructures. These limitations highlight the significance of this research, which incorporates recent advancements in ITS telemetry communication and web development methodologies to develop a real-time telemetry system using MQTT over WebSockets for efficient GPS tracking and traffic coordination. The system leverages MQTT's low-latency and high-reliability data exchange, ensuring continuous vehicular telemetry communication in ITS environments. Additionally, InfluxDB, a time-series database, is integrated to store and manage historical telemetry data, enabling efficient retrieval and analysis of vehicle movement trends. The system provides real-time traffic monitoring and coordination through a web platform integrating Google Maps API, facilitating dynamic visualization of vehicle movements and traffic light status while achieving both low latency and precise telemetry updates.

This work presents a novel real-time telemetry system that leverages the MQTT messaging protocol over WebSockets to enable efficient GPS tracking and traffic coordination. The system integrates InfluxDB, a time-series database, for managing historical telemetry data and utilizes the Google Maps API for real-time visualization of vehicle movements and traffic light status on a web platform. This combination of technologies aims to achieve low-latency and precise telemetry updates, enabling dynamic traffic monitoring and improved responsiveness to changing traffic conditions. The key contribution of this research is the specific integration of MQTT over WebSockets with InfluxDB and the Google Maps API, offering a robust and scalable solution for real-time data exchange in ITS, ultimately contributing to safer and more efficient transportation networks.

II. METHODOLOGY

A. System Architecture

The telemetry system enables vehicles and infrastructure to exchange data in real-time through MQTT over WebSocket connections. The system connects vehicles to traffic light infrastructure to share data in real-time, which assists in the advancement of smart traffic control systems. The system includes six main structures.

1) Vehicle Units

Vehicle units gather GPS data in real-time and send it through onboard MQTT clients. Each vehicle sends its position and movement data via MQTT topics, which other system components can access to get updated information.

2) Traffic Light Units

Traffic light controllers receive control commands from the control system which will be used to determine the traffic light status and timer based on the control algorithm for traffic management.

3) MQTT Broker

The MQTT broker acts as a central distribution point that establishes data communication in real-time via subscriber and publisher topic structure. The system lets vehicles send telemetry updates to establish traffic lights and the web platform communication.

4) Database Storage

The Storage system adopts InfluxDB as its preferred time-series database solution as it specializes in handling telemetric data in real-time. The database stores vehicle movement paths alongside speed changes and traffic light activity for historical data analysis.

5) Telemetry Web Platform

The web platform shows vehicle and traffic updates in real-time for monitoring purposes. Through WebSockets, the system fetches live telemetry data from the MQTT broker to show vehicle data and traffic updates in real-time.

6) Data Processing Module

A Node.js system processes raw GPS data before sending it to storage or transmission. The system reads National Marine

Electronics Association (NMEA) format sentences to convert coordinate values into decimal degrees and calculate vehicle speed using Haversine while preparing JSON data for MQTT broker and database integration.

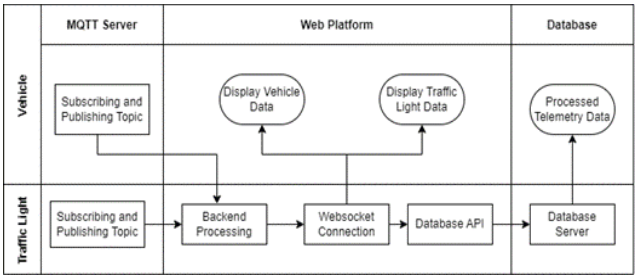

Fig. 1. Communication flow.

In Figure 1, the system starts when vehicles send GPS data through the MQTT broker, which then sends it to the data processing module for preparation before storage or display. Traffic light units operate through control algorithms that give commands for signal changes and are displayed on the Web Platform.

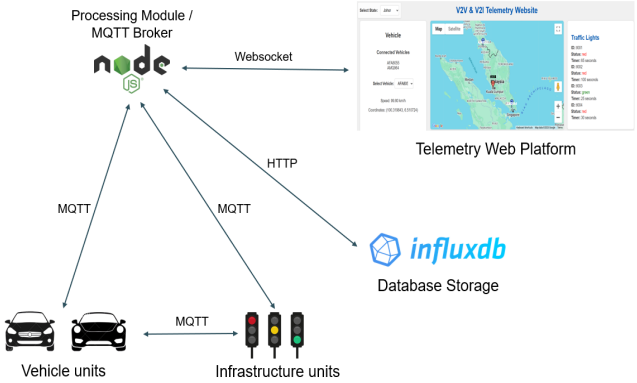

Fig. 2. Communication framework.

The system communication framework, as in Figure 2 demonstrates the real-time telemetry capabilities of MQTT and WebSocket and HTTP protocols working together. The Node.js processing module integrates an MQTT broker, which functions as the core component for establishing quick and dependable lightweight communication between vehicle units and infrastructure units through its publish-subscribe model. The system distributes GPS positions and traffic light statuses to the system without additional overhead, which ensures quick and essential updates needed for ITS environments. The WebSocket protocol creates a permanent two-way connection that connects the processing module to the telemetry web platform, which delivers immediate updates for displaying vehicle positions and infrastructure statuses on the Google Maps interface. When telemetry data needs to reach the InfluxDB time-series database, HTTP becomes the protocol responsible for this transfer while keeping data structured for traffic pattern analysis and monitoring of trends. The system achieves a smooth, time-sensitive, scalable communication flow through its integrated components.

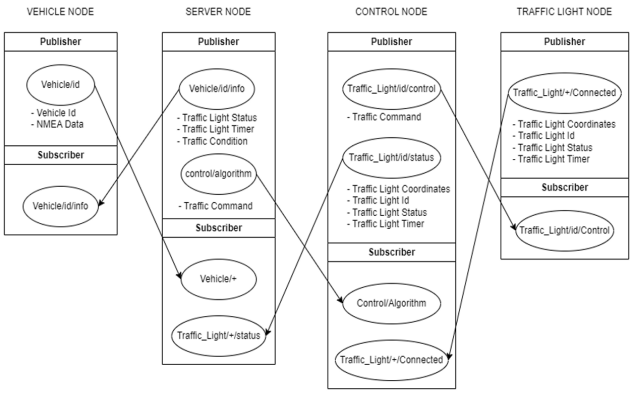

Fig. 3. MQTT topic structure.

Figure 3 illustrates the topic structure of the MQTT-based communication framework, where each node functions as a publisher, subscriber, or both. The server and control node are the processing modules that perform data processing as well as configure the MQTT Broker to handle communication from all the connected sources. The vehicle nodes send GPS and identification data through NMEA format to the /vehicle/id topic and use /vehicle/id/info to subscribe to processed infrastructure information, which enables dynamic navigation adjustments. The server node serves as the data processor and distributor, which collects information from all vehicles and traffic lights through vehicle/+ and traffic_light/+/status subscriptions. The server node distributes processed vehicle information through the vehicle/id/info topic while sharing control strategies through control/algorithm. The control node operates traffic management algorithms by receiving control logic and traffic light connectivity information from control/algorithm and traffic_light/+/connected subscriptions. The system sends control instructions to traffic_light/id/control and distributes status information to traffic_light/id/status with the signal state, countdown timer, and position details. Through traffic light nodes, vehicles and control systems obtain real-time updates and signal status information by subscribing to traffic_light/+/connected and receiving traffic_light/id/status messages.

Through this method, the system architecture can be effectively scaled by leveraging MQTT's topic hierarchy and broker clustering capabilities, along with edge-layer processing and adaptive traffic control coordination as demonstrated in [2], [5], allowing for reliable message dissemination across thousands of vehicles and multiple infrastructures in large-scale deployments.

B. Data Collection and Processing

The telemetry system receives and processes GPS data from vehicle units to show vehicle positions and control traffic signals in real-time. The GPS modules deliver NMEA sentences that provide data about vehicle location and movement. This system takes NMEA messages and processes them to eliminate invalid readings to maintain precise data quality. Subsequently, the system parsed and converted all GPS coordinates from degrees-minutes format into decimal degrees format to create a consistent location reading. The

Haversine formula works between GPS points to find the distance traveled, which helps the system measure vehicle speed in kilometers per hour. The processed data uses JSON structure for real-time MQTT transmission and database storage.

1) MQTT Topic Structure

The system implements a structured MQTT topic hierarchy to enable real-time data exchange between vehicles, traffic lights, and backend servers and control algorithms. The system components use topic-based messaging to subscribe and publish events, which enables both state synchronization and efficient event handling.

TABLE I. MQTT TOPIC STRUCTURE

Topic Name	*Publisher*	*Subscriber*	*Purpose*
vehicle/{vehicle_id}/nmea	Vehicle Unit	Backend Processing	Raw GPS Data (NMEA)
vehicle/{vehicle_id}/processed	Backend Processing	Web Platform	JSON-formatted GPS Data
vehicle/{vehicle_id}/info	Server.js	Vehicle Unit	Sends processed vehicle info
vehicle/{vehicle_id}	Vehicle Unit	Vehicle Unit	Publishes telemetry updates
traffic_light/{traffic_light_id}/status	Traffic Light Unit	Web Platform	Publishes traffic light states
traffic_light/{traffic_light_id}/control	Control.js	Traffic Light Unit	Sends control commands
traffic_light/{traffic_light_id}/connected	Traffic Light Unit	Control.js	Traffic light connection status
traffic_light/+/status	Traffic Light Unit	Server.js	Aggregates traffic light status updates
traffic_light/+/connected	Traffic Light Unit	Control.js	Registers newly connected traffic lights
vehicle/+	Vehicle Units	Server.js	Subscribes to all vehicle updates
control/algorithm	Server.js	Control.js	Sends control decisions for traffic light adjustments

In Table I, the hierarchical structure of MQTT topics shows the message exchange between vehicle units, traffic lights, backend processing, and web platforms. The topic structure operates dynamically through identifiers {vehicle_id} and {traffic_light_id}, which identify vehicles and traffic light units respectively. Through its publish-subscribe model, the system achieves efficient data transmission by ensuring that system components only receive relevant updates without wasting bandwidth. Vehicle telemetry receives processing before real-time visualization begins, and traffic light status updates are transmitted continuously for monitoring needs.

2) Database Storage and Retrieval

The system stores processed telemetry data within InfluxDB through a time-series database that specializes in continuous data operations. The system persistently records database information on vehicle GPS data along with speed measurements and traffic light statuses through its efficient real-time monitoring and historical analysis capabilities.

TABLE II. TIME-SERIES DATABASE STRUCTURE

Field	Description
Timestamp	The time when the data was recorded
vehicle_id	Unique identifier for each vehicle
Latitude	Processed GPS latitude in decimal degrees
Longitude	Processed GPS longitude in decimal degrees
speed_kmph	Vehicle speed in kilometers per hour
traffic_light_id	Unique identifier for a traffic light
light_status	Current traffic light state (Red, Yellow, Green)

Table II shows the InfluxDB time-series database structure for storing vehicle telemetry and traffic light status. The database uses timestamps as its main index to provide quick access for real-time performance analysis and visualization needs.

Through MQTT, the broker enables real-time data sharing among vehicle units, traffic lights, and the telemetry web platform. Vehicle telemetry data moves to specific MQTT topics for traffic lights and the web platform to receive vehicle location updates in real-time. The system saves processed data in the InfluxDB database, which handles time-series data effectively to support both past performance checks and real-time monitoring.

C. Web Development

The telemetry web platform functions as a real-time interface to display vehicle movements alongside traffic light status and system analytics. The platform implements Node.js for backend processing alongside JavaScript with WebSockets for frontend data visualization. Real-time data streaming becomes possible through the combination of WebSockets with MQTT, which is optimized for telemetry website updates.

1) Frontend Implementation

The web application implements HTML CSS and JavaScript development while using Google Maps API to display real-time GPS tracking through an interactive map interface. Through the Google Maps API users can view vehicle positions, speed indicators, and traffic light status in real-time for precise location tracking. Through a WebSocket connection, the front end receives the processed MQTT data, which provides continuous live telemetry data updates. The interface improves user experience by sending real-time event notifications that display important changes, including new vehicle detections and traffic light status updates.

2) Backend Implementation

The backend system functions as an intermediary through Node.js development to connect the MQTT broker and the database with the frontend WebSocket clients. The backend system subscribes to MQTT topics, which it processes to deliver structured telemetry data to the frontend. The database InfluxDB stores processed data, which enables users to track historical data for analysis purposes.

The backend analyzes MQTT-based NMEA GPS vehicle data by first extracting latitude and longitude positional information before converting the degrees-minute measurement system to decimal degrees format and removing any inconclusive measurement readings. The system determines vehicle speed through the Haversine formula, which evaluates the distance traveled during specific periods. The system stores processed data in InfluxDB through JSON format, which includes a timestamp and vehicle ID alongside coordinates and speed measurements. The system provides real-time tracking precision and enables quick access to historical data for additional analysis.

3) Real-Time Data Flow

The data flow within the web platform follows this structured sequence:

1. Vehicle Units publish real-time GPS data to the MQTT broker.
2. The Node.js backend subscribes to MQTT topics, processes raw data, and formats it before transmission to the frontend.
3. The Frontend WebSocket connection retrieves real-time updates and dynamically updates the Google Maps API-based interface and analytics dashboard.
4. The Database (InfluxDB) stores structured telemetry data, allowing historical retrieval via API queries.

The telemetry web platform functions as a real-time interface to track vehicle movements and traffic light status, and system analytics. The platform utilizes Node.js for backend processing alongside JavaScript with WebSockets to display frontend data visualization. WebSockets integrated with MQTT enable low-latency telemetry updates that stream real-time data without needing manual refreshes.

III. Results And Discussion

A. MQTT Performance Analysis

MQTT communication performance was evaluated by measuring message transmission delays across QoS levels 0, 1, and 2. The results indicate that better Quality of Service helps messages reach their destination more reliably while affecting delay time differently.

The graph in Figure 4 shows how MQTT delays change over time for different Quality of Service settings. The lowest delay occurs with QoS 0, while QoS 2 shows the highest delay because it requires message acknowledgments. The delays for QoS 2 show how the system needs to handle more processing to guarantee one-time message delivery.

1) MQTT Delay Trends Over Time

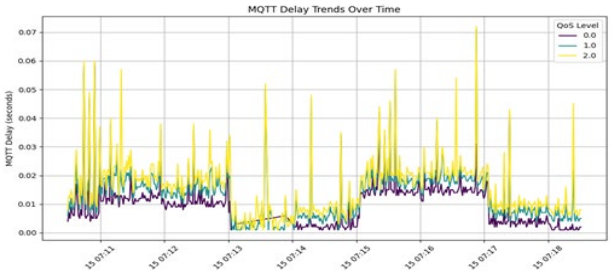

Fig. 4. Trend of MQTT delays over time.

Figure 5 shows how often MQTT delays fall into specific time ranges. The histogram shows that most delays stay below 0.02 seconds, but higher QoS levels experience sudden increases in delay that make their performance more unpredictable. The probability density estimation reveals that QoS 0 and QoS 1 produce consistent delay results, while QoS 2 shows more variation in its delay values.

2) Distribution of MQTT Delays

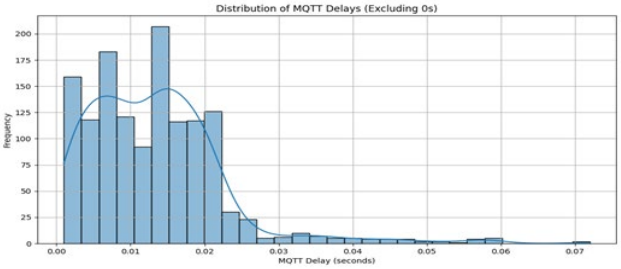

Fig. 5. Frequency of delays across different ranges.

3) Comparison of MQTT Delays by QoS Level

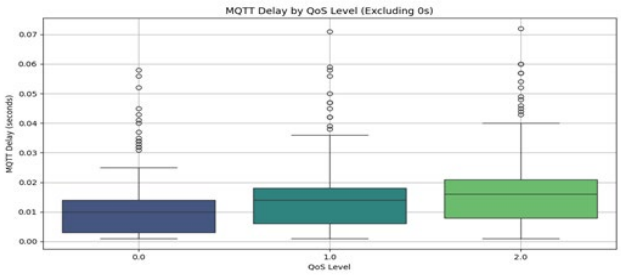

Fig. 6. Box plot comparing MQTT delays for QoS level.

Figure 6 displays a box plot that shows MQTT delay performance at QoS levels 0, 1, and 2. The results demonstrate that QoS 0 produces the shortest median delay with fewer outliers than QoS 2, which has the highest median delay. The larger interquartile range in QoS 2 shows that delay values vary more widely, which confirms the performance trade-off between reliability and latency in MQTT messaging.

B. MQTT Resilience Evaluation

This section evaluates MQTT performance through delay analysis and studies how the protocol behaves under naturally

fluctuating network conditions that were recorded during system operation. The experimental setup used a vehicle node as an MQTT publisher and a backend server as a subscriber, which connected through a Wi-Fi-based local area network. The system operated with naturally changing wireless conditions, even though artificial congestion was not introduced. The recorded timestamps and delay metrics allow post-analysis simulation of dynamic traffic load and network congestion effects on MQTT reliability.

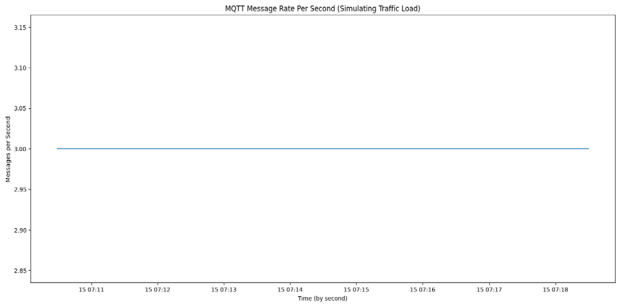

Fig. 7. MQTT message rate per second

The MQTT message rate per second is displayed in Figure 7 during the entire experiment duration. The experiment results demonstrate a steady message transmission speed of 3 messages per second, which remains constant throughout the observation period. The publishing process functioned without interruption, indicating that no messages were lost. The protocol shows its ability to maintain delivery frequency stability without QoS level 2 implementation, which proves essential for time-sensitive vehicular communication systems. Throughput degradation does not occur because MQTT demonstrates its ability to maintain continuous delivery throughout time.

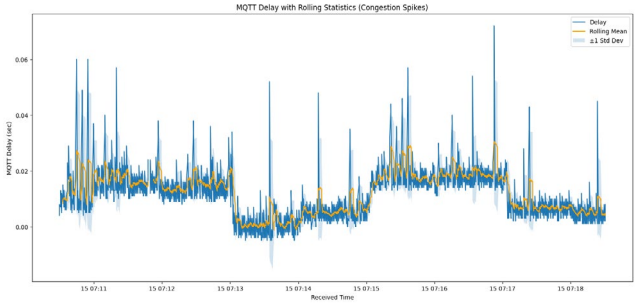

Fig. 8. MQTT Delay with Rolling Statistics

The rolling analysis in Figure 8 shows the MQTT delay through moving average and standard deviation calculations. These calculations help track brief delay changes. The delay graph shows significant variations and occasional spikes reaching values up to 0.06 seconds. Temporary queuing delays or jitter spikes occur in the system, probably due to network interference, broker congestion, or wireless channel contention. The system coped with transient latency fluctuations because MQTT successfully maintained operation without packet loss despite the random variations in delay amount.

The experimental setup was conducted in a semi-controlled environment that used steady traffic generation, it does not represent actual deployment scenarios with extensive urban traffic and fluctuating signal conditions. Nonetheless, the experimental findings of this study are consistent with previous research outcomes [1], [3], [4], [6], which demonstrated MQTT's reliable message delivery alongside minimal overhead and network disruption tolerance. Furthermore, Gruener et al. [6] examine MQTT broker stability and fault tolerance during stress tests, which confirms its suitability for IoT and V2X applications that need reliable, lightweight communication. The evaluation results demonstrate that MQTT would be suitable for connected vehicle systems as it provides stable throughput and limited delay despite specific network and broker configurations.

C. GPS Performance Evaluation

GPS performance was tested by looking at how measurement errors change over time and by comparing the consistency of position readings between multiple devices. The analysis shows how environmental factors, movement speed, and GPS refresh rate impact location precision.

1) GPS Error Trends Over Time

Fig. 9. GPS error trends.

Figure 9 presents GPS error patterns when comparing phone GPS with Emlid Reach RS GPS, using Emlid Reach RS as the ground truth. Initially, the phone's GPS achieves accuracy within 2 to 5 meters. However, periodic fluctuations cause error spikes reaching 17-18 meters, likely due to movement speed, environmental interference, and GPS update delays. Under ideal conditions, phone GPS measurements improve to nearly one meter. The sudden spikes indicate possible data loss or timestamp inconsistencies caused by irregular GPS updates.

Figure 10 illustrates GPS coordinate variability over time for both devices. Initially, both GPS units maintain similar stability, but variations increase when signals weaken or movement changes. The parallel fluctuation patterns indicate that environmental conditions impact performance more than device hardware limitations.

2) GPS Stability Analysis

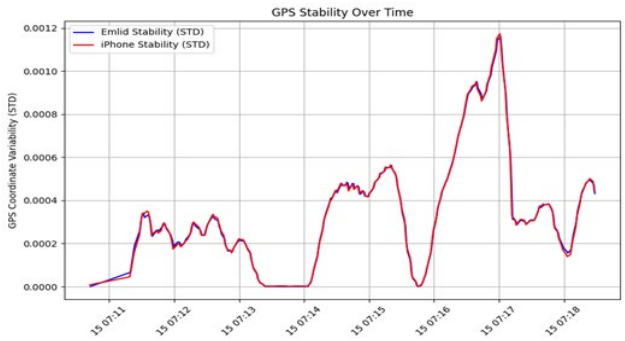

Fig. 10. Standard deviation of GPS coordinates.

Figure 10 illustrates GPS coordinate variability over time for both devices. Initially, both GPS units maintain similar stability, but variations increase when signals weaken or movement changes. The parallel fluctuation patterns indicate that environmental conditions impact performance more than device hardware limitations.

3) Correlation Analysis of GPS Error, Speed and MQTT Delay

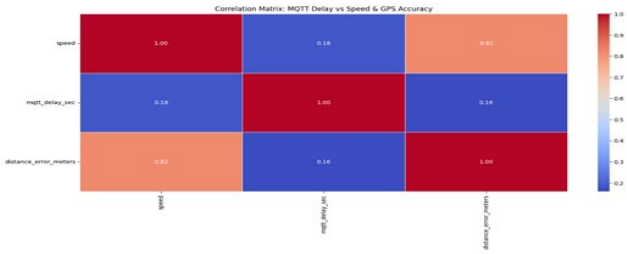

Fig. 11. Correlation matrix between MQTT delay, speed and GPS accuracy.

Figure 11 illustrates the correlation between GPS error, vehicle speed, and MQTT delay. The strong correlation (0.82) between speed and GPS error suggests that higher vehicle speeds reduce GPS accuracy, likely due to motion effects and update delays. In contrast, MQTT delay shows minimal correlation with both speed (0.18) and GPS error (0.16), confirming that network latency does not significantly impact GPS precision or real-time telemetry updates.

The analysis highlights that GPS accuracy decreases significantly at higher speeds, making motion-induced errors a major concern for real-time telemetry. Additionally, the MQTT delay remains stable, ensuring consistent data transmission. These findings emphasize the need for GPS filtering techniques to enhance location accuracy and confirm MQTT's suitability for real-time ITS applications.

D. Real-Time Data Visualization

The telemetry web platform displays vehicle movements and traffic light status updates in real-time through an interactive Google Maps API-based interface for continuous monitoring. The system combines MQTT and WebSockets to deliver quick updates through an interface that tracks vehicles and traffic signals automatically without requiring manual page reloads.

1) Visualization of Vehicle and Traffic Light Data

Fig. 12. Real-time visualization of vehicle and traffic light data.

Figure 12 shows the telemetry web interface that displays vehicle positions on a map and shows system metrics, including speed and location. The platform enables dynamic traffic light state updates through incoming MQTT messages. Through its Google Maps interface, users can perform both zooming and panning operations to examine particular locations in greater detail.

2) WebSocket-Based Live Updates

The platform depends on WebSockets to obtain real-time telemetry data from the MQTT broker for smooth updates. The system uses this mechanism to eliminate periodic polling, which results in reduced network overhead while maintaining a high response speed.

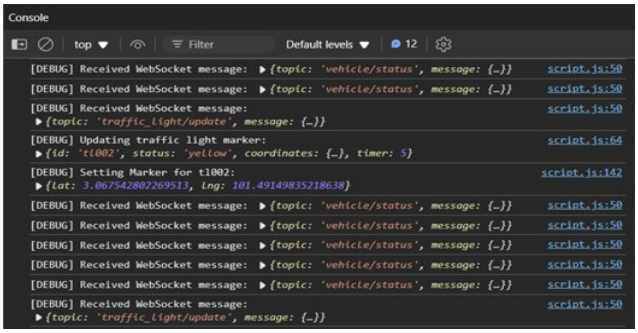

Fig. 13. Websocket-based live updates developer console

In Figure 13, the browser developer console displays real-time WebSocket messages that demonstrate vehicle and traffic light update reception. The system records an entry in the log every time vehicles modify their positions or traffic lights switch their status. The WebSockets system delivers live telemetry updates with minimal latency which enables immediate web interface updates. Through MQTT communication the system optimizes data transmission tasks which reduces network utilization and ensures message reliability.

E. Database Management and Historical Data Retrieval

This system integrates InfluxDB, a time-series database, to store and manage historical telemetry data for efficient retrieval and analysis. While WebSockets handle real-time updates, InfluxDB ensures long-term storage of vehicle speed, coordinates, and timestamps, enabling traffic trend analysis and decision-making.

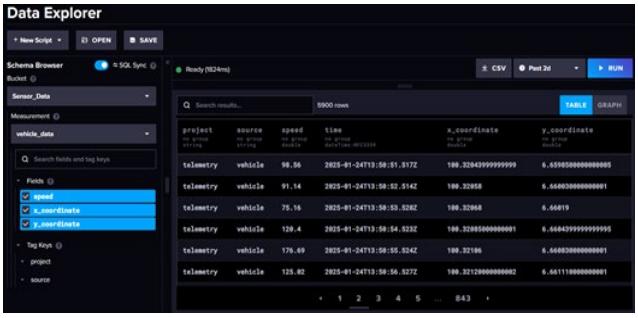

Fig. 14. InfluxDB data explorer.

Figure 14 illustrates the InfluxDB Data Explorer, showing stored telemetry data for vehicle tracking and speed monitoring. The system allows query-based data retrieval, ensuring access to past telemetry for performance evaluation and traffic optimization. This data persistence supports ITS decision-making, preventing information loss during network disruptions.

Overall, the system delivers an efficient and scalable solution for real-time visualization needs in V2V and V2I telemetry applications. The system will benefit from future improvements that optimize WebSocket performance for handling larger data volumes and enhance visualization capabilities for dense traffic conditions.

IV. CONCLUSION

Exploring ITS and integrating a real-time telemetry system that connects vehicles and traffic lights through MQTT over WebSocket to share data in real-time has revealed further possibilities and challenges in this dynamic field. The system processes GPS telemetry data, calculates vehicle speed, and transmits structured data for real-time visualization. The web platform leverages WebSockets for continuous updates on vehicle movements and traffic conditions while storing telemetry data in InfluxDB for historical analysis. The evaluation outcomes demonstrate that MQTT communication achieves both low latency and reliable messages, especially with QoS 1 settings, balancing between speed and reliability while maintaining message integrity. The test conducted shows that the data processing pipeline works well because Emlid GPS and Phone GPS show similar vehicle tracking results. However, results also reveal that higher speeds introduce greater GPS errors, demonstrating the need for filtering techniques to improve real-time positioning accuracy. The ongoing development of this field will benefit from machine learning traffic prediction models combined with edge computing data processing techniques to enhance system efficiency. The implementation of 5G networks will boost system responsiveness and decrease latency, which enables large-scale deployment of the system.

ACKNOWLEDGMENT

The authors would also like to thank and acknowledge the Faculty of Electrical Engineering, Universiti Teknologi MARA, 40450 Shah Alam for their support.

REFERENCES

[1] A. Amelia, F.N. Roslina, H. Pranoto, B.V. Sundawa, I.S. Hutauruk, & A. Arief, "MQTT Protocol Implementation for Monitoring of Environmental Based on IoT," 2020 International Conference on Applied Science and Technology (iCAST), Padang, Indonesia, 2020, pp. 700-703, doi: 10.1109/iCAST51016.2020.9557694.

[2] N. M. Mari, S. Arrigoni, F. Braghin, S. Mentasti and M. Filippini, "A V2I communication framework of adaptive traffic lights and a prototype shuttle," 2022 AEIT International Annual Conference (AEIT), Rome, Italy, 10.23919/AEIT56783.2022.9951792.

[3] N. Kaskatiiski and L. Boyanov, "Efficiency of data exchange of IoT communication protocols," 2021 International Conference Automatics and Informatics (ICAI), Varna, Bulgaria, 2021, pp. 358-361, doi: 10.1109/ICAI52893.2021.9639627.

[4] D. Selimović, A. Salkanović and M. Tomić, "Application of MQTT Based Message Brokers for IoT Devices Within Smart City Solutions," 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 2022, pp. 428-433, doi: 10.23919/MIPRO55190.2022.9803388.

[5] M. Živić, D. Nemec and Ž. Bojović, "MQTT protocol in IoT environment: Comparison with CoAP and ZeroMQ protocols," 2023 31st Telecommunications Forum (TELFOR), Belgrade, Serbia, 2023, pp. 1-4, doi: 10.1109/TELFOR59449.2023.10372710.

[6] S. Gruener, H. Koziolek and J. Rückert, "Towards Resilient IoT Messaging: An Experience Report Analyzing MQTT Brokers," 2021 IEEE 18th International Conference on Software Architecture (ICSA), Stuttgart, Germany, 2021, pp. 69-79, doi: 10.1109/ICSA51549.2021.00015.

[7] J. Rouyer, A. Ninet, H. Fouchal, and A. Keziou, "A road intersection control in urban intelligent transportation systems," ICC 2022 – IEEE International Conference on Communications, Seoul, Republic of Korea, 2022, pp. 3562-3567, doi: 10.1109/ICC45855.2022.9838267.

[8] H. Antunes and I. d. S. A. da Fonseca, "Advanced web methodology for flexible web development," 2021 16th Iberian Conference on Information Systems and Technologies (CISTI), Chaves, Portugal, 2021, pp. 1-4, doi: 10.23919/CISTI52073.2021.9476295.

[9] B. Mendes, M. Araújo, A. Goes, D. Corujo, and A. S. R. Oliveira, "Exploring V2X in 5G networks: A comprehensive survey of location-based services in hybrid scenarios," Vehicular Communications, vol. 52, p. 100878, 2025. [Online]. Available: https://doi.org/10.1016/j.vehcom.2025.100878.

[10] V. Shushkova, A. Kashevnik and L. Bakeeva, "Accelerometer and Gyroscope Synthetic Data Calculation based on Driver Smartphone GPS," 2024 36th Conference of Open Innovations Association (FRUCT), Lappeenranta, Finland, 2024, pp. 748-755, doi: 10.23919/FRUCT64283.2024.10749939.