Acta Montanistica Slovaca
ISSN 1335-1788
actamont.tuke.sk

# Application of computer modeling software for mining vehicle fleet telemetry monitoring

*Fares ABU-ABED[1]** and *Aleksei IVANOV[2]*

**Authors' affiliations and addresses:**
[1] Tver State Technical University, Russia, Tver, Embankment Afanasiya Nikitina, 22
e-mail: aafares@mail.ru

[2] Tver State Technical University, Russia, Tver, Embankment Afanasiya Nikitina, 22
e-mail: lexuzieel@gmail.com

***Correspondence:**
Fares ABU-ABED, Tver State Technical University, Russia, Tver, Embankment Afanasiya Nikitina, 22
tel.: +7 4822 52-48-30
e-mail: aafares@mail.ru

**How to cite this article:**
Abu-Abed, F. and Ivanov, A. (2021). Application of computer modeling software for mining vehicle fleet telemetry monitoring. *Acta Montanistica Slovaca*, Volume 26 (4), 593-602

**DOI:**
https://doi.org/10.46544/AMS.v26i4.01

**Abstract**
This study introduces a computer modeling approach for mining vehicle fleet monitoring. Computer modeling helps to reduce prototyping costs and lower the risks of initial launch failure by analyzing and configuring the prototype in order to test various options and find the most fitting ones. We show in the first part of the study that using a computer modeling method, it is possible to test numerous combinations of metrics acquired during vehicle monitoring in the simulation rather than adding equipment to vehicles during prototyping Usage of real hardware during the prototype phase adds downtime to the vehicle fleet and reduces productivity. Along with those drawbacks, it also introduces the possibility of additional costs if the configuration needs to be changed later. By leveraging modern time-series data storage solutions, we present an easier approach to analyze real-world data retrieved using a simulation as a proxy. In the second part of the paper, we then propose a workflow of integrating SUMO with a time-series storage database through an Application Programming Interface (API) called TraCI, allowing for aggregation of vehicle fleet data over time and visualizing that data on the dashboard. At the end of the paper, we show a measuring methodology and provide a viable solution for the efficient transfer of telemetry data.

**Keywords**
Mining, vehicle, fleet, telemetry, monitoring, internet of things, computer, modeling

## Introduction

Mining is a capital-intensive sector that necessitates the purchase of big equipment worth hundreds of millions of dollars. Mine trucks are the most commonly utilized pieces of equipment for material hauling in surface mining operations. On the other hand, their maintenance costs account for a major amount of the total operating cost – usually more than 50% of the operational cost (Subtil et al., 2011). Maintenance costs cannot be reduced because currently existing costing methods and models do not account for all critical restrictions.

The amount of ore and waste that needs to be hauled each year over the mine's life is determined by the mine production schedule of an open pit. Mine haulage trucks play an important role in transporting these materials due to their flexibility and efficiency. Mine trucks, on the other hand, are expensive pieces of machinery that require routine maintenance in order to operate safely and efficiently. In addition, mine trucks must undergo a major maintenance service after a certain number of working hours, which primarily includes engine replacement. (Topal et al., 2010) estimates that the cost of these maintenance services can account for 30–50% of the overall haulage cost of a surface mining operation. As a result, it is critical to optimize the mine truck fleet's schedule so that the fleet's utilization is maximized while maintenance costs are kept to a minimum.

With the current improvements in the Internet of Things (IoT) technologies, more industries are incorporating these technologies into their operations in order to obtain a better understanding of their processes and find new methods to improve them. In conjunction with big data technology and cloud computing, the Internet of Things enables continuous monitoring of nearly any equipment, including vehicles. Because of the higher hazards to human health and the costly damage to equipment in the mining business, analyzing that data can be valuable.

While there appears to be a multitude of solutions in the field of logistics and long-distance general goods transportation, transport monitoring using Global Positioning System (GPS) has been extensively used previously (Greenfeld, 2002). Despite the fact that there are numerous solutions for equipment and staff monitoring, there appears to be a lack of developments in vehicle fleet monitoring, particularly in the mining industry. Recently, IoT has been increasingly used for transportation monitoring (Čaušević et al., 2018), utilizing less expensive computer devices and transport protocols such as LoRaWAN (Navarro-Ortiz et al., 2018).

Because of the industry's characteristics, as well as variances in the vehicles themselves, the requirements for data transferring security regulations, and telemetry gathering device interfaces with the vehicles, the bulk of solutions available is not applicable to the mining industry. In their publication, Chaulya & Prasad (2016) show a method for mine transport surveillance that includes obtaining camera feeds and detecting intrusions. While their work focuses on the production management aspect, we believe it is also worthwhile to explore a solution for an automated method of telemetry gathering, which will allow monitoring parameters such as humidity, tire pressure, ambient temperature, and other variables that are unique to the industry segment in order to ensure process security and reliability.

It is beneficial to employ computer modeling to produce a dependable outcome while building a new solution in order to reduce the cost of integration and limit the chances of failure. Nowadays, computer modeling software is widely used in a wide range of businesses, particularly those with a higher potential of hazard. In this paper, we propose an approach for mining vehicle fleet monitoring using Simulation of Urban MObility (SUMO) (Krajzewicz et al., 2002) vehicle traffic modeling software applied to the mining industry. We are using SUMO since it is one of the most well-established road traffic modeling tools (Krajzewicz et al., 2012), and it has proved to be a reliable and convenient software package that we used in our previous work proposing a method for traffic control system configuration validation (Ivanov & Abu-Abed, 2019).

The first section of the paper discusses the modeling software we want to employ and shows how to simulate a fleet of mining vehicles. Then we show how, in the early stages of a prototype, such simulations can substitute costly testing hardware such as real-life automobiles. Then, using an Application Programming Interface (API) called TraCI, we offer a method for connecting SUMO with InfluxDB, a time-series data storage system, to enable the aggregation and visualization of vehicle fleet data over time.

## Overview of vehicle fleet computer modeling software

Modeling traffic flows is the basis of this work. Because of this, it is important to choose the most appropriate tool that would solve this problem. Currently, there are many solutions for various kinds of computer simulations, including simulations of traffic flows. In this area, there are three most common solutions

- *PTV Vissim* – a leader in the market of software systems for modeling traffic flows and has many different functions, but is available only as a paid version
- *SUMO* (***S**imulation of **U**rban **Mo**bility)* – is an open-source portable traffic flow simulator that allows you to simulate large road networks
- *Quadstone Paramics Modeller* – a product of Quadstone Paramics, which allows visualizing the modeling process using 3D graphics, but only the paid version and demo version are available – the latter, in turn, does not allow you to create and edit models

PTV Vissum (Fellendorf & Vortisch, 2010) can be considered one of the most advanced solutions for traffic computer modeling currently. However, while providing many useful features specific to urban traffic modeling, those features could not be used in the context of vehicle simulation in the mining industry. Hence, for our proof-of-concept, we decided to opt for a smaller solution that still covers our needs in terms of functionality, which are:

1. Ability to control simulation from an external program
2. Ability to read individual vehicle state in the simulation
3. Microscopic level of simulation

For our case of vehicle telemetry data gathering, we require a software solution that could simulate mining vehicles and their environment on the *microscopic* level, primarily because of the small travel distances inside the mining plant. Several traffic simulation tools out there fall into one of these categories depending on which level of abstraction they provide during modeling (Lopez et al., 2018): *macroscopic* when only traffic flow dynamics are simulated, only generic aspects of the flow can be analyzed, such as traffic density; *microscopic*, when each vehicle and its dynamics are modeled individually; *mesoscopic* – a combination of macroscopic and microscopic approaches; *submicroscopic* – each internal function of the vehicle simulated.

Microscopic level modeling is sufficient for our purposes since it gives adequate information during the simulation of even slow vehicles in a smaller area, which the mining operation has in abundance. It should be feasible to reproduce a given mining site using a microscopic traffic computer modeling approach. This might be accomplished, for instance, by using publicly available satellite photos of a given location and reproducing existing monitoring transport routes in the model.

Even though it might not be as feature-rich, we are using Simulation of Urban MObility software for demonstration purposes, which still provides microscopic traffic modeling capabilities (Krajzewicz et al., 2003), and it is freely available to use because of the open-source distribution model. Another important feature of SUMO is a continuous simulation of traffic flow. This type of simulation allows improving accuracy of simulation (Fig. 1):

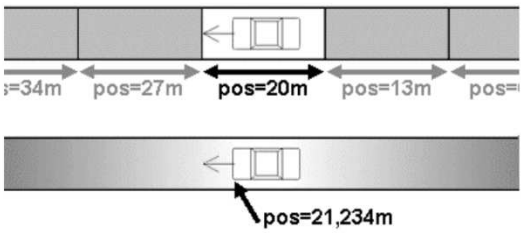


*Fig. 1. Continuous simulation of traffic flow compared to discrete simulation*

SUMO is made up of many software modules that allow for the pre-processing of simulation data, the customization of the model environment, and the visualization and simulation of the data. Following that, we will go through a few modules that are essential for creating a simple simulation: *SUMO* – the simulation program itself which does not have a graphical interface and could be used separated by calling it from the Command Line Interface (CLI); *SUMO-GUI* – a graphical interface for the simulation program which is cross-platform and can be run on any widely used operating system; *NETEDIT* – a model editor that allows to create and edit model environments by defining road networks, vehicle routes and flows and making it easy to configure complex models. Fig. 2 shows an example of a typical road network model created in NETEDIT.

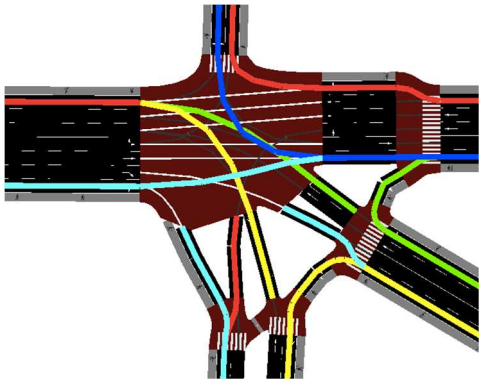

*Fig. 2. Typical road network created with NETEDIT*

**Telemetry time-series data processing and storage**

A time series is an ordered succession of values of a variable at evenly spaced time intervals. As a result, it is a series of discrete-time data. Time-series can include data that has been associated with date and time, such as measurements from IoT devices.

A time series is made up of measurements that are sorted on a timeline to give information about underlying trends. Because there is a relationship between time and measurements, changing the order can affect the interpretation of the data (Joshi et al., 2017). Hourly temperature recordings at a specific weather station, daily measurements of the closing price of a certain stock, and so on are examples of time series.

The majority of telemetry data is recorded as time-series data, consisting of data points spread over time at regular intervals. Air temperature collected at regular intervals, stock price, and so on are examples of time series data. There are metrics in the mining sector that may be of interest to a certain sub-branch of the industry. However, when it comes to vehicle fleet monitoring, there may be certain common indicators that may be used to better understand how they operate. Time-series format is also convenient for monitoring because it allows to detect outliers in the data (Jagadish et al., 1999) more easily, and by employing current advancements in machine learning, it becomes possible to detect anomalies in time-series data more quickly and accurately (Kanarachos et al., 2017).

There are numerous time-series database solutions available today; some are tailored to specific needs, such as software application monitoring, while others are general-purpose databases that can be used in a variety of projects, ranging from simulation data logging (as we do) to IoT telemetry logging. In fact, there are so many different types of time-series databases that researchers are conducting a study in both specific sectors (Fadhel et al., 2019) and more broad surveys (Bader et al., 2017).

A TSDB is a database type that is designed to store time series or data that has been time-stamped. It's designed specifically to deal with time-stamped metrics, events, or measures that change over time. A TSDB enables users to build, enumerate, update, destroy, and manage diverse time series more efficiently. The main distinction between time series and normal data is that you usually ask questions about it across time. Nowadays, the bulk of businesses generates an enormously big stream of measurements and events (time series data), necessitating the need for a TSDB. For our purposes, we will be using InfluxDB since it is the most top-ranking (Fig. 3) time-series database (TSDB) recently (Naqvi et al., 2017).

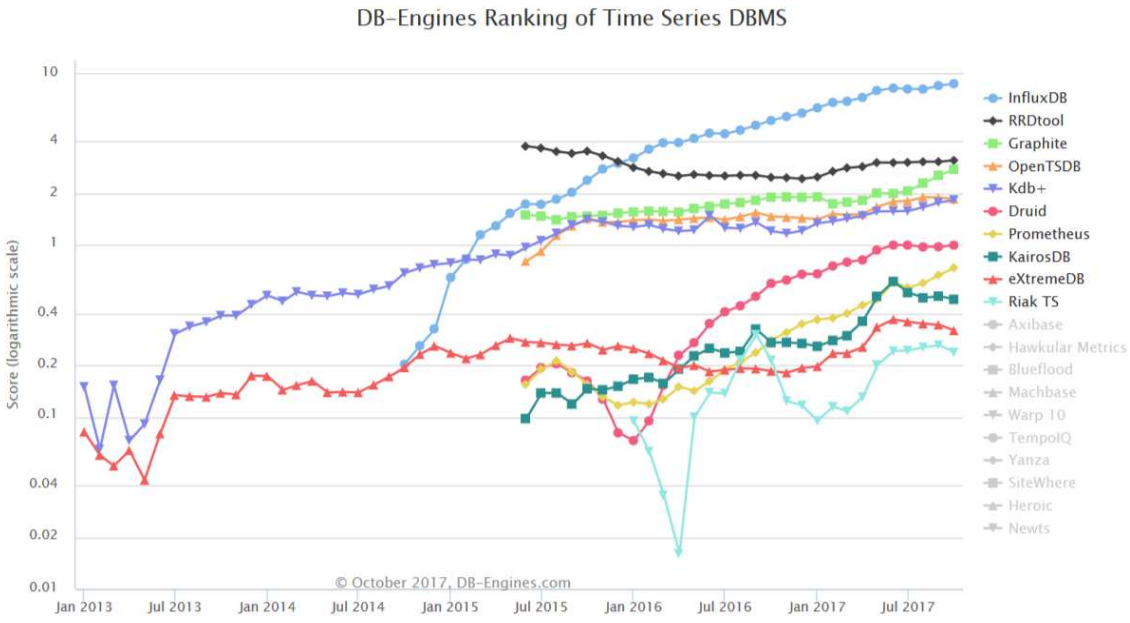


*Fig. 3. Ranking of the top-10 time-series databases (Source: Naqvi et al., 2017)*

InfluxDB does not depend on any other DBMS and uses a SQL-like language called InfluxDB Query Language (InfluxQL) (Bader et al., 2017) and has proven to work in IoT applications (Nasar & Kausar, 2019).

Our justification for using InfluxDB is that, while it is newer on the market than OpenTSDB, it has more capabilities and – more significantly – has an integrated dashboard, which eliminates the need for us to set up a separate dashboard solution such as Grafana (Grafana Labs, 2020). Of course, in real-life applications choice of the database is not purely on the looks and functionality of the dashboard and the like, but that is a separate extensive topic outside of the scope of this paper.

## Material and Methods

In this portion of the article, we detail the tools and procedures we used to generate a digital model of a mining site using the SUMO traffic modeling software solution. Following that, we concentrate on developing a simple software system that can collect telemetry data from automobiles, emulating the real-world application of such a system. Finally, we show how to use InfluxDB and its interactive time-series database and exploration tool to visualize gathered telemetry data.

### Creating mining site computer model

Our proposed model consists of two parts: SUMO road network with the graphical representation of a mining site and a Python program that would use TraCI to control the simulation. This will give us easy access to the simulation data and make it possible to use it for telemetry. Because it is time-consuming to set up a realistic traffic simulation scenario, which has been done more in-depth by (Bieker et al., 2015) and others, we will take some liberties since our aim is to present a proof-of-concept for a way to use computer modeling in mining vehicle fleet monitoring without relation to a real mining site.

Using NETEDIT software that is a part of the SUMO package, our model will describe road networks and routes which vehicles can take using those roads. NETEDIT allows using a background image for the model. This will help to create a graphical representation of a mining site that is being simulated. For demonstration purposes, we will be using part of a Bingham Canyon in Utah, USA, as an example representation of a mining site by using satellite image freely accessible from Google Maps (Fig. 4).

*Fig. 4. Bingham Canyon, Utah (Copyright Google, Copyright Maxar Technologies, State of Utah, USDA Farm Service Agency)*

By importing this image as a background image, it is possible to closely recreate a road network segment for this particular mining site. This map can be larger for real-life applications and can consist of multiple images stitched together, similar to how satellite images are stitched together for maps. By placing junctions and connecting them with the edges, a road network is created (Fig. 5), which can then be used to define transport routes.

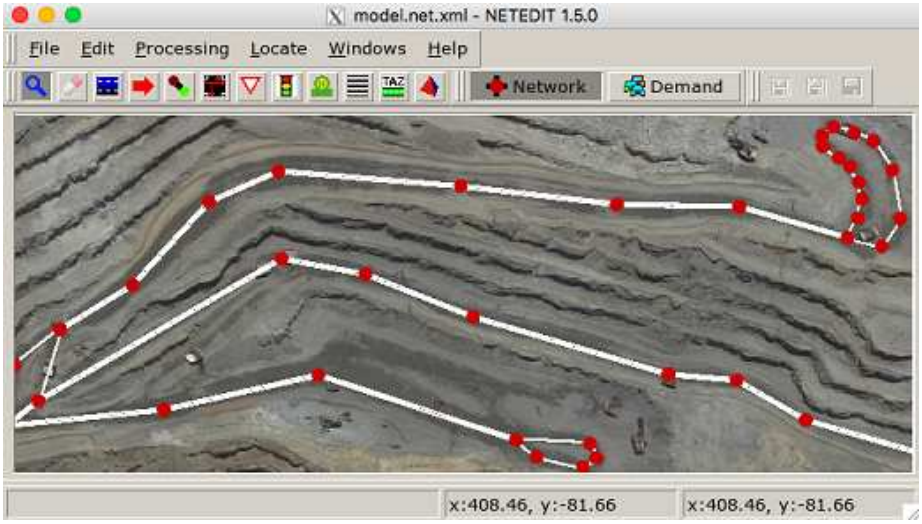


*Fig. 5. Mining site transport network*

**Set up of simulation software**

In this part of the paper, we develop the model by creating a small Python program that would launch and interact with the simulation created in NETEDIT.

After the model in SUMO has been created, the next part of the simulation would reside in a Python program that would use TraCI to launch and control the simulation. Therefore we will not be using SUMO to launch the simulation, but a Python program that will launch SUMO through TraCI. In the same program, the management of the vehicles will be done. Our development environment is as follows: Python 3.7 on macOS High Sierra 10.13 using pipenv virtual environment manager and a TraCI library that is included with the SUMO installation (German Aerospace Center (DLR), 2020).

This package enables us to run external simulations in server mode and connect to them automatically from within the program. We may alter the simulation using the TraCI API by adding and removing vehicles, as well as assigning any newly formed vehicle to a route that was previously defined in the model. Following the creation of the vehicles, we would query them for telemetry data at regular intervals. We will ask for acceleration measurements in meters per second squared (the default SUMO unit for acceleration) and angle measurements in degrees in this example. We're utilizing InfluxDB as our time-series database. Thus, we are also using the InfluxDB Python module. On the same computer, an InfluxDB instance is started using binaries from the official site (InfluxData Inc, 2020).

The design of our simulation involves three steps: data acquisition, data transmission, and storage of acquired data (Fig. 6), which closely relates to a real-life implementation of the fleet monitoring system. Our Python program will help us complete all of the three components (Prinsloo & Malekian, 2016).

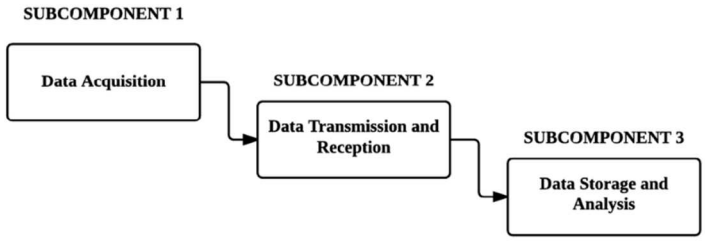


*Fig. 6. General overview of a monitoring system (Source: Prinsloo & Malekian, 2016)*

When the simulation has been initialized using TraCI, we will insert five vehicles into the simulation using the procedure shown in Fig. 7. This program will add a new vehicle into the simulation following one of the predefined routes in the simulation that use the following name conversion: **route_#**, where **#** is the index number of the route starting from zero.

```
routes = 3

def add_random_vehicle(vehicle):
    traci.vehicle.add(
        vehicle,
        f'route_{random.randint(0, routes-1)}',
        typeID="vType_0"
    )

if __name__ == "__main__":
    sumoBinary = 'sumo-gui'
    traci.start([sumoBinary, "-c", "model.sumocfg", ])

    for i in range(5):
        add_random_vehicle(f'truck{i}')

    vehicles = []

    while traci.simulation.getMinExpectedNumber() > 0:
        for vehicle in traci.simulation.getDepartedIDList():
            vehicles.append(vehicle)

        for vehicle in traci.simulation.getArrivedIDList():
            vehicles.remove(vehicle)
            add_random_vehicle(vehicle)

        traci.simulationStep()

    traci.close()
```

*Fig. 7. Python program that controls SUMO simulation*

Running the program from Fig. 7 will launch SUMO simulation that has been defined in the configuration file "model.sumocfg". This program will add five mining vehicles (Fig. 8) and run in an infinite loop:

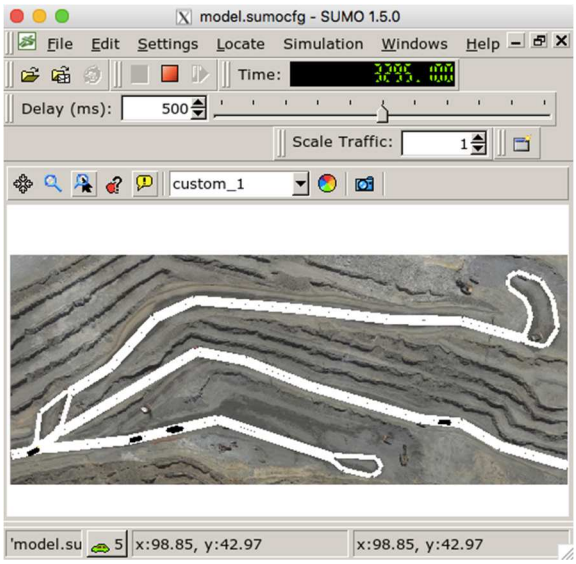


*Fig. 8. SUMO simulation launched using TraCI*

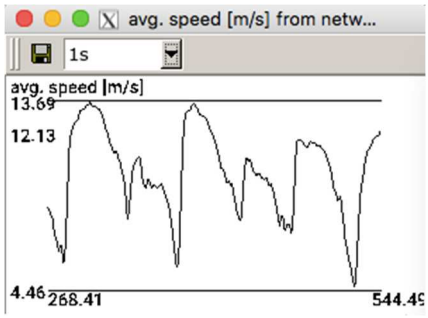


*Fig. 9. The average speed of vehicles in the simulation*

While running, the SUMO has access to all the information in the running simulation, such as vehicle positions, their speed, angle, etc. It is possible to either manually read this data using a graphical interface in SUMO (Fig. 9) or, since we are using TraCI, we have the opportunity to read all this data programmatically – effectively simulating telemetry data aggregation from a fleet of vehicles.

**Collecting telemetry data and storing it in InfluxDB**

After the model has been set up and operating, the next step is to integrate it with InfluxDB for time-series data collecting and storage. As previously stated, for this example, we will use the data point write procedure to write the acceleration and angle of each vehicle in the simulation. Using the Python client library makes this possible. We can bypass network configuration procedures like NAT configuration and port forwarding because the InfluxDB server is hosted on the same system as the simulation.

It is possible to store every measurement from every vehicle at some predefined period as a time-series using the InflixDB client Python API. Furthermore, in order to simulate the implementation of a real-life embedded system device that collects telemetry frequently and then averages the results, we will aggregate measurements each frame and, at the time of writing, averaging the sum of all recent measurements over the number of samples collected, effectively simulating real-world behavior. It is important to stress such an approach since it allows to bring simulation closer to what would happen in real-life simulation and will help avoid unexpected results when the prototype is transferred from the model into a real-life scenario.

Figure 10 shows a modified version of the program that now collects vehicle acceleration and angle measurements while also publishing the average value of the values to the InfluxDB database every five seconds using Python client API.

```
for vehicle in vehicles:
    measurements['acceleration'][vehicle].append(
        traci.vehicle.getAcceleration(vehicle)
    )
    measurements['angle'][vehicle].append(
        traci.vehicle.getAngle(vehicle)
    )

    # Every 5 seconds
    if traci.simulation.getTime() % 5 == 0:
        for key in measurements:
            write_point(vehicle, key,
                        np.average(measurements[key][vehicle]))
            measurements[key][vehicle].clear()

traci.simulationStep()
```

*Fig. 10. Modified Python program which now collects measurements from the simulation and sends them to InfluxDB*

The next step would be to log into the dashboard and configure the visualization of the measurements once the simulation has been running and writing measurements to the database for some time. Since InfluxDB comes with an already integrated graphical interface, we will be using that instead of some additional solution. Given the data we have, we will set up two graphs for each of our measurements.

First, we are going to set up *truck median acceleration*. This metric allows to the analysis of characteristics of vehicle movement. We will display this metric as a stacked histogram for each of the five trucks (Fig. 12a). InfluxDB has a data query graphical user interface that allows to easily select and filter time-series data entries while also applying aggregate functions to the result. In the case of truck acceleration, we are interested in median value over time (Fig. 11):

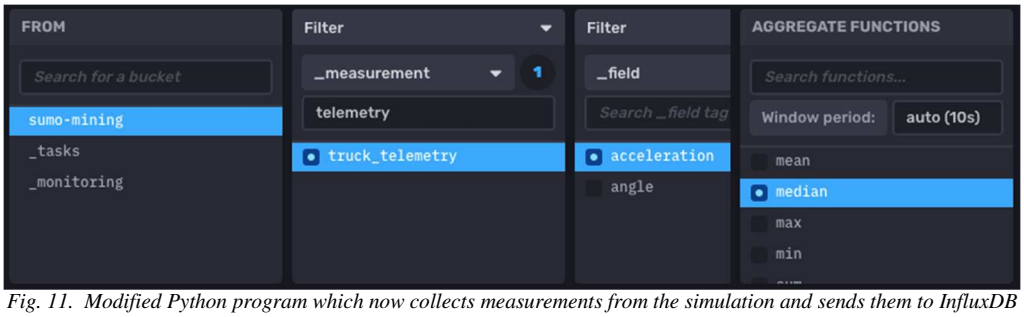


*Fig. 11. Modified Python program which now collects measurements from the simulation and sends them to InfluxDB*

Another metric we are going to display is *truck angle derivate* (Fig. 12b). This metric will show the rate at which the angle of each individual vehicle changes its angle. It can help monitor drivers' behavior and prevent hazardous situations. It is worth noticing that we got rather high values for this metric in our experiments primarily because of rather sharp corners in paths in the model. Therefore it is advised to increase the resolution of the model in order to mitigate this.

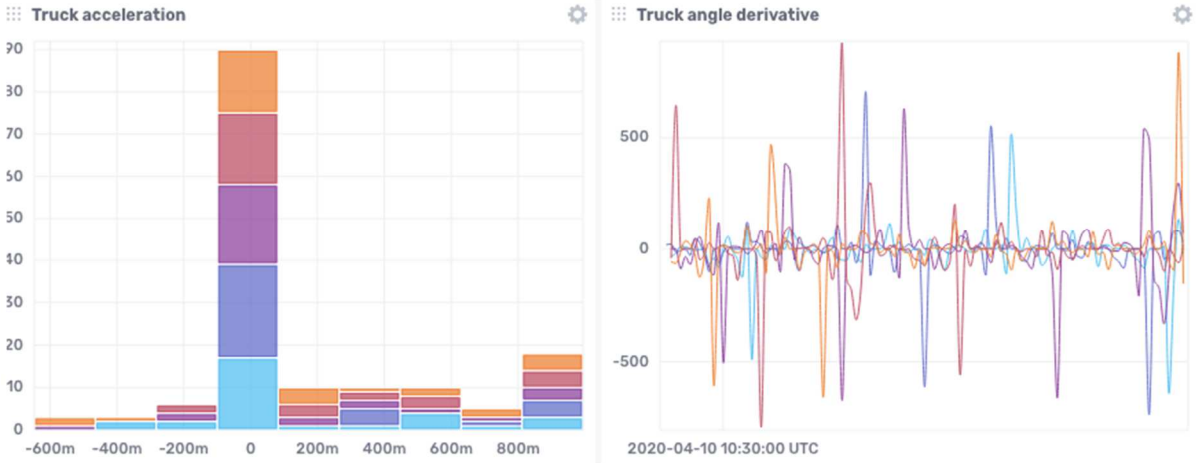


*Fig. 12. (a) individual truck median acceleration in m/s², (b) individual truck angle derivative in degrees*

## Discussion

We have shown that it is possible to use computer modeling to get vehicle telemetry sample data. However, a few unanswered questions remain. In this section, we discuss alternative implementations of the measurement process utilizing real hardware in this section of the study, as well as the problem of efficient data transport.

RQ1: **How to transfer results acquired during the modeling process?**

Here the term "results" refers to the telemetry data model that has been obtained, the data format, and the post-processing procedures that are performed in the dashboard for full data display. Even though it is possible to sample raw location data from the simulation directly, we believe it would be of no use because the values of x and y coordinates on a plane in the simulation environment are raw, which means they follow the internal simulation format and are thus meaningless to the user. One solution would be to utilize coordinate mapping to convert the simulation's Cartesian coordinates to the geographical coordinates of the mining site used in the model. However, while this would add extra complexity to the simulation system, it would still be doable. The coordinate mapping would match the vehicle's GPS position. However, as Prinsloo & Malekian (2016) point out, this method of transportation localization is not always accurate enough in complicated situations. In this scenario, RFID

markers are recommended since they can aid in vehicle tracking. Furthermore, if there is a requirement to monitor internal vehicle parameters, it is worth mentioning and maybe using an on-board diagnostic system (OBD II) (Malekian et al., 2016) while building an embedded system for vehicle monitoring is worth a try. However, due to the unique nature of vehicles in the mining industry, there is no certainty that such a system will be installed. In this situation, simply tracking the location of the vehicle and other external sensor measurements (such as temperature, vibration, and so on) may be sufficient to offer adequate information for fleet management decisions.

RQ2: **How to transfer telemetry data efficiently?**

In general, an efficient data transport method is critical for vehicle telemetry in particular. Currently, LoRaWAN appears to be a promising data transfer method for IoT systems. LoRaWAN is a low-power wide-area network technology that has shown to be successful in industrial IoT applications that employ mobile data networks (Navarro-Ortiz et al., 2018). Furthermore, large-scale applications require the use of an efficient time-series database. Choosing the correct time-series storage solution depends on the enterprise and its existing infrastructure and a generic demand on the volume of data flow to be handled (Nasar & Kausar, 2019). As a result, we recommend looking into research that focuses on examining and comparing alternative time-series database solutions, such as Bader et al. (2017) and Fadhel et al. (2019), since this question goes well beyond the scope of this work.

## Conclusion

In this work, we used a computer modeling technique to demonstrate a proof-of-concept for a telemetric data gathering system for a mining vehicle fleet. Because both vehicles and data collection can be simulated, this strategy might significantly minimize integration costs by minimizing vehicle downtime - which is especially crucial in the early stages of fleet management system integration. In the post-processing step of data analysis in a time-series database dashboard, this approach can also help uncover possible metrics that can be utilized for fleet monitoring and adjustments of these metrics.

We further identified three components in the overall process of a monitoring system (see Fig. 6) and proposed various implementations for each of them — the data obtained from the simulation can be employed in the early stages of the creation of additional system components. The data transmission and reception components can then test against the requirements for data integrity during transmission using simulated data. This step could also incorporate various data corruption recovery options (for example, via a custom protocol) — this component could be implemented as a separate software unit in some circumstances. Finally, the ability of a human operator or an algorithm to make control decisions requires data storage and analysis.

A sample modeling environment of a mining site was constructed using the SUMO transport traffic computer modeling system. This simulation was performed externally by a program using the TraCI API, which provides greater flexibility and the ability to obtain telemetry data from the model environment. After that, the telemetry was saved in an InfluxDB time-series database. We demonstrated that this approach might be utilized to model telemetry gathering scenarios that can aid in mining vehicle fleet management by presenting the test data received from the simulation.

One topic not covered in this study is a way of providing feedback on a driver's decision-making process. Because it is extremely dependent on the company policy and implementation details of the telemetry collection system, it might theoretically be generated from telemetry data (i.e., which vehicles are being monitored). Potential for feedback based on retrieved metrics could improve operational efficiency (e.g., by picking more routes) or even worker safety (i.e., important event notification broadcast towards personnel with even data derived from the telemetry data automatically).

In our future research, we would aim to add an evaluation framework for measuring metrics, which would aid in the decision-making process for fleet management duties. We would like to add a more robust set of vehicle metrics, such as vibration and humidity, to that framework, as well as more complex SUMO aspects of regions and extra objects that could aid vehicle tracking during simulation. Aside from those measures, another significant enhancement would be the addition of geographical position tracking in SUMO via the imitation of GPS tracking mentioned in previous sections utilizing the coordinate mapping approach.

## References

Abu-Abed, F. E3S Web of Conferences 278, 01017 (2021) SDEMR-2021.

Abu-Abed F.N. IT-solutions for transport monitoring systems in the mining industry / F.N. Abu-Abed, A.V. Ivanov // Bulletin KuzSTU (2020) (pp. 69-76).

Bader, A., Kopp, O., & Falkenthal, M. (2017). Survey and Comparison of Open Source Time Series Databases. Datenbanksysteme für Business, Technologie und Web (BTW2017) (pp. 249-268). Gesellschaft für Informatik e.V. (GI).

Bieker, L., Krajzewicz, D., Morra, A., Michelacci, C., & Cartolano, F. (2015). Traffic simulation for all: a real world traffic scenario from the city of Bologna. Modeling Mobility with Open Data, 47-60.

Čaušević, S., Čolaković, A., & Haskovic, A. (2018). The model of transport monitoring application based on Internet of Things. Opatija, Croatia: International Scientific Conference on Science and Traffic Development.

Chaulya, S., & Prasad, G. (2016). Mine Transport Surveillance and Production Management System.

Fadhel, M., Sekerinski, E., & Yao, S. (2019). A Comparison of Time Series Databases for Storing Water Quality Data., (pp. 302-313).

Fellendorf, M., & Vortisch, P. (2010). Microscopic Traffic Flow Simulator VISSIM. In Fundamentals of Traffic Simulation. International Series in Operations Research & Management Science (Vol. 145, pp. 63-93). Springer, New York, NY.

German Aerospace Center (DLR). (2020). TraCI - SUMO Docmentation. Retrieved 2020, from https://sumo.dlr.de/docs/TraCI.html

Grafana Labs. (2020). Grafana: The open observability platform. Retrieved from https://grafana.com

Greenfeld, J. (2002). Matching GPS Observations to Locations on a Digital Map.

InfluxData Inc. (2020). InfluxDB: Purpose-Built Open Source Time Series Database. Retrieved 2020, from https://www.influxdata.com

Ivanov, A., Abu-Abed, F. (2019). Usage of SUMO Computer Modeling Software for Road Traffic Control System Validation. 8(2), 4662-4666.

Jagadish, H., Koudas, N., & Muthukrishnan, S. (1999). Mining Deviants in a Time Series Database. VLDB, 99, 7-10.

Joshi, P., Massaron, L., & Hearty, J. (2017). Python: Real World Machine Learning. Packt Publishing.

Kanarachos, S., Christopoulos, S., Chroneos, A., & Fitzpatrick, M. (2017). Detecting anomalies in time series data via a deep learning algorithm combining wavelets, neural networks and Hilbert transform. Expert Systems with Applications, 292-304.

Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012). Recent development and applications of SUMO-Simulation of Urban MObility. International Journal On Advances in Systems and Measurements, 128-138.

Krajzewicz, D., Hertkorn, G., Feld, C., & Wagner, P. (2003). An Example of Microscopic Car Models Validation Using the Open Source Traffic Simulation SUMO. 14th European Simulation Symposium, (pp. 318-322).

Krajzewicz, D., Hertkorn, G., Rössel, C., & Wagner, P. (2002). SUMO (Simulation of Urban MObility)-an open-source traffic simulation. Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002), (pp. 183-187).

Lopez, A. P., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y., Hilbrich, R., Wiessner, E. (2018). Microscopic Traffic Simulation using SUMO. 21st International Conference on Intelligent Transportation Systems (ITSC), (pp. 2575-2582).

Malekian, R., Moloisane, N., Nair, L., Maharaj, B., & Chude Okonkwo, U. (2016). Design and Implementation of a Wireless OBD II Fleet Management System. IEEE Sensors Journal.

Nasar, M., & Kausar, M. (2019). Suitability Of Influxdb Database For Iot Applications. International Journal of Innovative Technology and Exploring Engineering, 1850-1857.

Navarro-Ortiz, J., Sendra, S., Ameigeiras, P., & Lopez-Soler, J. (2018). Integration of LoRaWAN and 4G/5G for the Industrial Internet of Things. IEEE Communications Magazine, 56, 60-67.

Naqvi, SNZ. & Yfantidou, S. (2017). Time Series Databases and InfluxDB.

Subtil, R. F., Silva, D. M., & Alves, J. C. (2011). A practical approach to truck dispatch for open pit mines, 35th Apcom Symposium, Wollongong, NSW, Australia.

Topal, E. & Ramazan, S. (2010). A new MIP model for mine equipment scheduling by minimizing maintenance cost, Eur. J. Oper. Res., 207, 1065–1071.