Tracking and reconstructing vehicles routes

Érick de Oliveira Teixeira * **Rafael de Magalhães Dias Frinhani** *
Luiz Olmes Carvalho *

* *Federal University of Itajubá – Itajubá - MG*
(e-mail: {erick.teixeira, frinhani, olmes}@unifei.edu.br)

Abstract: The tracking of vehicles and cargo plays an important role in the globalized world. There are several solutions to that task, but some present limitations and others are expensive. In this context, this paper develops a low-cost device to the traceability problem. Our solution implements a mechanism to reconstruct the real traveled route even in the absence of network connection. In addition, we designed not only the hardware but also the firmware and a client server. The communication of the device and the server is performed by using distinct data transmission modes. Our experimental evaluation identified the main features of each transmission mode in terms of resource consumption and a correct trajectory reconstruction in areas with no connection, proving the proposed solution consistently surpasses those issues.

Keywords: traceability, tracking, embedded, firmware, hardware, route.

1. PRELIMINARIES

Nowadays, one of the impacts resulting from the evolution of production technologies is the increasing in the number of vehicles circulating in roads and highways. According to data from the Brazilian Institute of Geography and Statistics, in 2022 this land fleet exceeded 115 million vehicles in Brazil (IBGE, 2022). As the amount of vehicles increases the same occurs with the number of crimes such as robbery or theft of vehicles and cargo. For instance, in the State of São Paulo, Brazil, there were 40,673 car robbery, 6,371 cargo robbery, and 92,912 car theft, only in 2022, according to the Secretary of Public Security of the State of São Paulo (SSP-SP, 2022).

One possible solution to that problem is to purchase a coverage insurance, aiming at protecting vehicles and assets. This option, however, is not always feasible, especially for low-income people, who often prefer (or are forced) to take risks associated with damage or loss rather than hiring a service that they may eventually never use. To illustrate, in the capitals of Brazil, the average price of car insurance in 2023 is R$ 3,805.29 for the male profile and R$ 3,035.07 for the female profile (Minuto Seguros, 2023).

An alternative approach is to employ tracking methods by using technologies for communication and positioning. However, companies providing such a service are focused on the corporate market, and the cost of joining ranges from R$ 300.00 to R$ 600.00, in addition to monthly fees between R$ 50.00 and R$ 150.00.

Nowadays, the available solutions for vehicle and cargo tracking are based on two main technologies: Radio Frequency (RF) and Global Positioning System (GPS). RF trackers work by emitting radio signals that are captured by antennas, allowing to obtain the location of the vehicle. GPS, on the other hand, is a positioning system that uses satellites to provide the geographic position and current time of a receiver (e.g. navigation center of a car, smartphone, microcontroller), regardless of climate conditions.

In addition to the advances in tracking technologies, the miniaturization of components and the evolution of the Internet of Things (IoT) have enabled smaller and cheaper devices to be connected to the Internet. Those devices employ most varied types of connections, such as Bluetooth Low Energy (BLE), General Packet Radio Service (GPRS), Global System for Mobile Communications (GSM), Wi-Fi, Long Range (LoRa), Near Field Communication (NFC), as well as the aforementioned RF and GPS (Oliveira, 2017). Since such technological alternatives enable the development of low-cost solutions for traceability, they have been employed in both the academia and industry environments (Chabi et al., 2021).

The main drawbacks regarding to those approaches are that solutions based on RF and GPS often need to be coupled to the vehicle due to the dimension of the device and the requirement for energy. In the case of vehicle substitution, the device must be removed and installed in other vehicle, and it demands for specialized service. Therefore, they are not portable.

Solutions based on IoT partially solve the portability issue, but they also present shortcomings. The first problem is most devices require an external module or equipment, such as electronic shields or smartphones, in order to establish network connections. Therefore, they are not standalone devices. Moreover, when the vehicle crosses roads and other rural areas where there is no connection to the internet or even mobile range, hereafter referred to as *shadow areas*, the correct trajectory of the vehicle is lost.

For instance, in Figure 1 it is possible to observe part of a trajectory crossed by a monitored vehicle. Until the point marked as “A” the trajectory was correctly traced, once the tracker device continuously sent to a central system its current position. Ahead the point “A”, the lack of

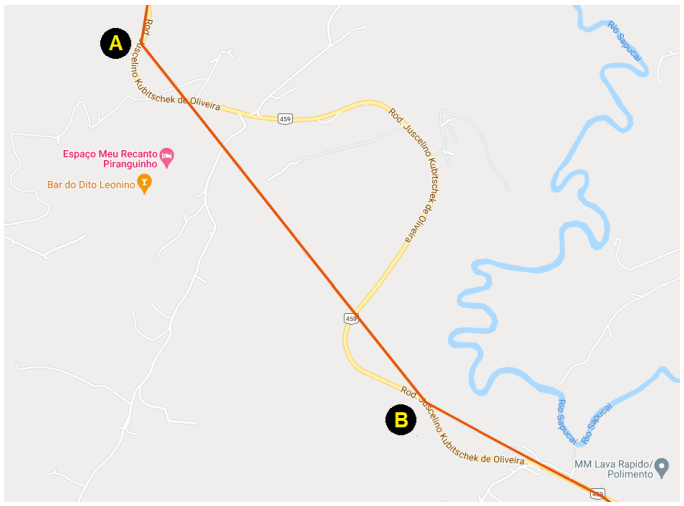

Figure 1. Problem of shadow areas in trajectories.

connection causes a trajectory loss: between the points "A" and "B" the device did not transmit its current location to a monitoring central since it is offline. The connection is restored only in point marked as "B", but the positions between "A" and "B" were not received by the central. Thus, when the connection is up, the trajectory showed in the map becomes a straight line linking the points "A" (where the connection was missed) and "B" (where the connection is reestablished). The *real route*, however, was not a straight line. Clearly, the vehicle performed the curve observed in the map.

Another common problem is related to the behavior of the equipment responsible for the tracking. Regardless of the device, they usually gather the current position considering fixed time intervals. As an example, if the device is configured to capture its location every 2 seconds and the vehicle is passing through a traffic jam, then representing each gathered coordinate as a point in the map results in a area containing several overlapped points, closer to each other, because the vehicle is moving too slow. That cluster of points does not bring valuable information to analyze the route, as well as it means a misuse of connection, bandwidth and, mainly, battery energy. An ingenuous solution is to increase the time interval. That approach, however, becomes problematic when the vehicle is at high speeds, because the traveled distance between two consecutive time intervals can reach hundreds of meters. Therefore, the sparsity of the points in the map may lead to erroneous interpretation for a route analysis.

Some approaches to vehicle monitoring can be found in the literature. The study presented in Chabi et al. (2021) proposed a tracking device based on the LoRaWAN protocol. In Lee et al. (2014) and Jacobsen et al. (2017), the developed prototypes communicates through SMS messages, which requires a smartphone, whereas our proposal does not use external devices. Similar to ours, Ya'u Gital et al. (2023) employed GPS/GSM technologies to build an intelligent speed assistance system, based on the vehicle coordinates. Following the same approach, GPS/GSM are also used in Mustafa et al. (2019) to solve the tracking problem, but that proposal relies on a full time connection in order to transmit information, while our implementation is able to work offline.

Considering the several possibilities for improvements identified in this context, the main goal of the present paper is to develop an IoT solution based on hardware and software for the problem of vehicle traceability. Regarding the hardware design, the main contributions of the proposed solution are:

- *Standalone*: the proposed architecture does not require external modules or equipment.
- *Portability*: the small weight and dimensions enable to easily move the device from a vehicle to another.
- *Free of installation*: the device does not require coupling or modifications in the vehicle, nor the handle of its electric/electronic parts to operate.

From the software (firmware and application) perspective, which is more interesting once its design can be replicated in other distinct hardware platforms, the main contributions are summarized as follows:

- *Customized transmission modes*: the information about location and timestamp can be sent to a server considering not only time intervals but also the traveled distance or speed ranges, so as to save energy.
- *Hybrid mode*: the software layer was designed to automatically alternate among the available transmission modes in order to best represent the real trajectory.
- *Asynchronous clock*: the server clock does not need to be synchronized with the device clock in order to reconstruct the route in the correct timestamp. Actually, the IoT device does not even have a Real Time Clock component.
- *Trajectory correction*: the route distortion problem, as depicted in Figure 1, is overcome by using a logical scheme of time synchronization in the software layer. Therefore, even in the presence of shadow areas, it is possible to reconstruct the real route.
- *Client-server architecture*: a server application receives the data gathered by the device through the `http` protocol. The hardware requirements for the server are minimum, like a domestic personal computer or laptop afford.

The remainder of this paper is organized as follows: Section 2 describes the proposed solution in terms of hardware, firmware and software layers. Section 3 presents the experimental evaluation of the proposal. Finally, Section 4 concludes the paper.

2. MATERIAL AND METHODS

The design and implementation of the proposed solution are divided into three parts: (i) the hardware assembly, i.e., the physical device; (ii) the firmware programming, i.e., the code in charge of controlling the device operation, and (iii) the software design, i.e., a server application.

2.1 Hardware

For the prototyping of the vehicle tracker hardware, the NodeMCU was chosen as the main controller. NodeMCU is a development board focused on the IoT, open-hardware, and low cost when compared to similar alternatives, with

prices ranging between R$ 35.00 and R$ 50.00. This board combines the ESP8266 chip developed by Espressif Systems, with a USB-Serial interface, a 3.3 V voltage regulator and several input and output terminals.

The NodeMCU can be powered via USB or batteries, and has a consumption of only 26 mA, according to its technical specifications. This is an advantage for the type of application for which it is being designed, as the low power consumption allows the tracker to operate for longer periods of time, also reducing the cost with power supply.

Once a tracking device requires a connection to send location information to a server where the data will be processed, in the proposed prototype, this operation is performed through an internet connection. The ESP8266 microchip has a 2.4 GHz 802.11 b/g/n Wi-Fi connection, with WPA/WPA2 support, which is one of the ways to use the tracker. However, it is important to highlight our goal is to build a device with no need for a smartphone or similar external devices in order to achieve communication. Thus, the adopted solution to ensure device communication employs a GPRS and GSM module: the SIM800L, which supports a quad-band GSM/GPRS network using a SIM card. That module has a compact size, low current consumption (up to 1 mA), is able of making and receiving voice calls, sending and receiving SMS and receiving GPRS data (e.g. TCP/IP, HTTP). Communication with the microcontroller is performed by a serial TTL interface.

The prototype uses the GPS system to obtain the geographic location of the tracked vehicle. For this purpose, the module Neo-6m V2 was chosen. The Neo-6m V2 is a GPS module which enables to gather information such as the latitude and longitude values, date, time and the speed of the vehicle. Also, the module has a compact layout and independent memory, making it ideal for applications involving mobile devices and microcontrollers. It is possible to receive location data and store them in the module's EEPROM memory, as it has an integrated battery for backup of the data and an integrated LED indicating whether the module is on or off. Communication with the microcontroller is performed through a serial interface.

To power the microcontroller and allow the tracking device to work without dependence on a USB connection, a 9 V battery was used, together with an LM2596 voltage regulator, responsible for ensuring that the input voltage on the board is 3.3 V. Lastly, the total cost of the hardware modules was about R$ 146.00.

2.2 Firmware

The firmware is a piece of software responsible for controlling the hardware modules. Basically, it gathers information from the device modules and take some action.

The solution's firmware was developed in the C++ programming language. The main code was combined with additional libraries in order to establish communication and control the modules. Those libraries are: `ArduinoJson`, which enables the use of JavaScript Object Notation (JSON); `SoftwareSerial`, for communication with the GPS and GSM modules by using the Serial protocol; `TinyGPS++` for handling data; and `sys/time.h` for operations with date and time.

The firmware was designed so as to decode the data received by the GPS module, produce a transmission message with those data and send it to the server. In the moments when the vehicle is crossing a shadow area, there is a failure to send the data. Thus, those messages are enqueued and saved in the memory, until the connection is reestablished. Then, all the saved messages are sent to the server, in a queue. It is important to observe the firmware may send messages in any order. The ordering of events in the server is logical, and in the case of receiving the enqueued messages in a random order, the server is able to sort them.

The message implementation was performed by a data type comprising three information: latitude, longitude and a timestamp (integer value). Aiming at achieving a better performance, the queue was designed as a static array of messages. The array size depends on the maximum capacity of the internal memory in the device.

Although the gathered timestamp may not correspond to the correct time, it can be used to calculate the time difference between one point and another. Such a difference is important to consistently allow the trajectory tracing when analyzing the route. The queue, on the other hand, represents a set of points.

The user for which the points will be collected is also configured in firmware, so that there is confidentiality, since users in the client application can only view the routes traveled by themselves and only if they are logged in. To help with security, an API key is also configured in firmware and the server only responds to requests if this key is valid and belongs to that specific user.

The firmware layer is also in charge of controlling how the vehicle location coordinates are transmitted to the application server. The traditional approach is to send those data in a *fixed time* interval. To the best of our knowledge, the studies in the literature employ only that strategy. In order to surpass the problems of the fixed time interval alternative (overlapped points, bandwidth and energy spending, described in Section 1), our implementation also performs the sending according to a fixed traveled distance, and a distance based on speed, as well as a hybrid mode which alternates among these modes automatically.

The *transmission by fixed distance* consists in collecting coordinates, calculating the distance between the current coordinate and the previous one, and sending data to the server if the accumulated distance is at least a previously defined distance δ, i.e., $distance(Coord_1, Coord_2) \geq \delta$, where δ can be customized by the user.

In an empirical evaluation, the higher the speed of the vehicle, the lower the frequency of sending the coordinates. This occurs because if the car is at a higher speed, due to inertia, the vehicle does not make abrupt changes in direction. Therefore, based on the observation of the maps, an ideal interval distance can be found for sending the coordinates according to the speed of the route traveled. In such a way, data is sent if $distance(Coord_1, Coord_2) \geq f(speed)$, where f is a linear function. In Section 3 (Experiments) a linear function f is interpolated using two different speeds. That approach is the *speed-controlled transmission*.

The *hybrid transmission* consists in using the fixed time sending when the vehicle is traveling at a speed limited to ν km/h, and the speed-controlled when the vehicle is traveling at more than ν km/h. This allows obtaining more accurate tracings in small routes and roads with many curves, where the amount of points sent will naturally be higher, due to the lower speed at which vehicle travels. In addition, this criterion enables network and memory savings on fast expressways and highways, where fewer points should be sent due to the higher speed.

2.3 Software

The software layer corresponds to the application server which receives and stores the geographic coordinates gathered by the device. The server works as an API, i.e., a set of definitions and protocols used for the development and integration of applications. The server was designed following the Representational State Transfer (REST) - a type of software architecture - and can be hosted in any cloud service such as Google Cloud.

The server receives data from the tracking device and performs validations to verify that the received messages are complete with all the necessary data fields. Than, the address of the point collected through latitude and longitude can be obtained using the LocationIQ API. The data is stored in a MySQL database, in a table containing login, latitude, longitude, real timestamp (from the server), street, neighborhood (or district), city, state, country and logical timestamp (from the device).

In order to infer the *real* timestamp of each point when the server receives a queue, it needs to treat the data in a different way. Initially, the server computes the time difference between the first element in the queue and the last point saved in the database. Thus, it obtains the real time for the first enqueued element. Once the real time of one enqueued element is known, based on the logical timestamp of the other elements, the real timestamp of every element can be computed, as depicted in Figure 2. The process is presented in Algorithm 1.

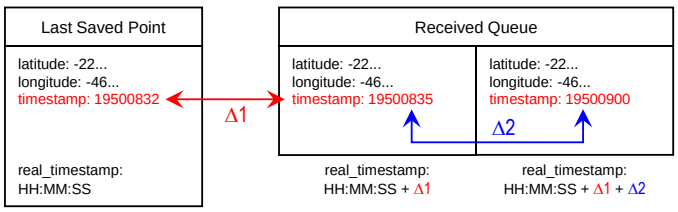

Figure 2. Real timestamp computation.

In the case a new route starts within a shadow area, the real timestamp can be inferred in the following way: there is no previous point saved in the server. Thus, while the device is offline, the gathered points are enqueued. When the connection is reestablished, the server receives a queue as its first information. The last point in the queue (which is also the first captured point when the device is online again) is synchronized with the real time in the server. So, the real timestamp for all previous elements in the queue can be also inferred based on the difference of their logical timestamps regarding the last element.

Algorithm 1: Real timestamp inference

Input: *queue*[ ], *login*
last ← `last_saved_point()`;
for *point* ∈ *queue*[ ] **do**
  $\Delta \leftarrow point.timestamp - last.timestamp$;
  $point.real\_timestamp \leftarrow last.real\_timestamp + \Delta$;
  `save_point`(*point*, *login*);
  last ← *point*;

3. EXPERIMENTS

For the experimental evaluation of the developed hardware, firmware and software, three routes were defined:

- Route 1 (small): this route comprehends just a city block, whose distance is about 312 m.
- Route 2 (average): this route crosses some streets around the Federal University of Itajubá, whose distance is about 4.1 km.
- Route 3 (large): this route comprises several neighborhoods and also includes a shadow area. The total distance is about 10 km.

Each of the routes was traveled with a car, and at one or more strategic points a stop was made to collect the correct time using a smartphone, just for purpose of comparison. For the experimental validation, those stops should be visible on the map (evidenced by several points close to each other) and the obtained time of the points at the stop should be sufficiently close to the correct time. Also, the route presented on the map must be the same route traveled by the vehicle. The tolerance value between the real time and the gathered time of the points was set by 1 minute, considering that the time of communication between the tracker device and the server may result in this small time difference.

We crossed the Route 1 on March 29 at 7:31 PM. For the Route 1, all the times matched when checked against the time collected by the smartphone. The main goal of traversing a small route like Route 1 was to validate the accuracy of the collected coordinates, that is, if the traversing of a small and geometrically defined route (such as a rectangular block in this case) would correspond to the real trajectory of the car.

The Route 2 was traveled on March 30 at 8:22 AM. There were two stopping points in the path: the first one at the Municipal Park of Itajubá and the other at the entrance of Federal University of Itajubá. The stop at the Municipal Park can be observed on the map by several points closer to each other. The collected time of the points also corresponded to the real time. The Route 2 is depicted in Figure 3, where the letter "E" indicates the start/end of the route.

The Route 3 started on April 2 at 4:48 PM. There was a single stop point in this route, in front of a supermarket, which is highlighted on the map. Figure 4 presents the Route 3 and shows that even in the presence of a shadow area (points at the top right corner), the points were consistently traced. The sections where the vehicle traveled faster (highway, first quadrant of the image) due to the greater distance between the collected points, and the

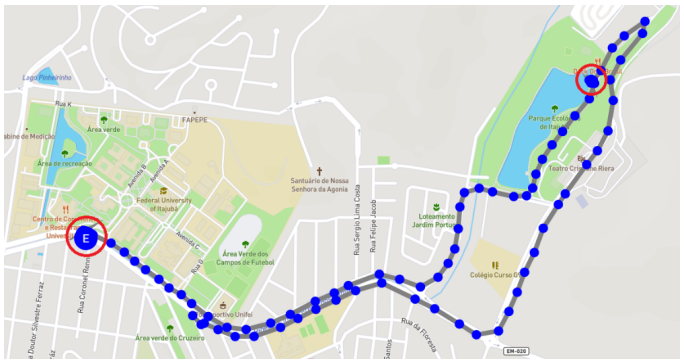

Figure 3. The average traveled route: 4.1 km.

sections where the car traveled slowly since it is in a urban area (closer points) can also be observed.

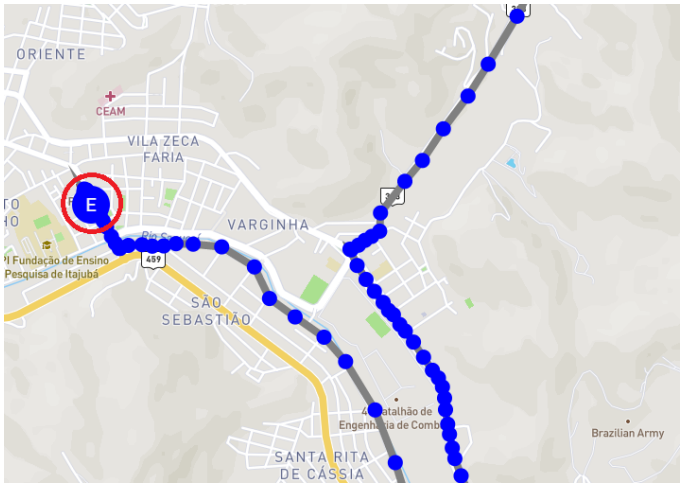

Figure 4. The large taveled route: 10 km.

In all the performed routes, the tracking device sent the collected data to the server every 10 s, using the fixed-time transmission mode. Although with this strategy it was possible to understand the path taken just observing the maps, additional experiments were performed with the three other transmission modes, in order to evaluate a possible optimization of the route on the maps.

Whereas the previous experiments evaluated the device regarding hardware calibration, time synchronization and route tracing, we also performed extensive experiments aiming at analyzing the distinct transmission modes and the route correction in shadow areas.

The first experiment evaluates the transmission by fixed distance. That method consists in collecting coordinates whenever possible, calculating the distance between the current coordinate and the previous one, and sending data if the accumulated distance without sending is equal to or greater than a previously defined distance. Figure 5 shows the resulting map of Route 1, where the sending distance was set to 10 m.

Comparing the route tracing using fixed time interval and fixed distance, the latter is satisfactory for short journeys, as it increases the accuracy of the tracing by sending more points. Such a strategy, however, may not be efficient for any route size, once there is not a fixed distance

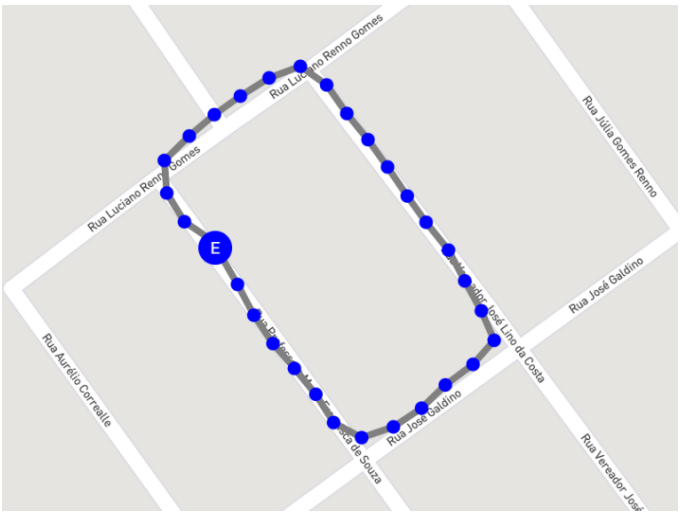

Figure 5. Transmission by fixed distance in Route 1.

parameter that can be generically applied. Therefore, the transmission by fixed distance is interesting to be applied in short routes.

The next transmission mode is the speed-controlled one. In that strategy, the higher the speed of the vehicle, the lower the frequency of data transmission. Thus, after several experiments in the Route 3, an ideal interval distance was established for sending the coordinates according to the speed of the path. For places where it is generally traveled at a speed of up to 25 km/h, sending every 10 m proved to be satisfactory. In the same way, on roads where the speed is close to 50 km/h, sending every 50 m has proved to be sufficient. Therefore, based on those two values, a linear function to control the sending distance is expressed such as in (1), where ν is the speed.

$$f(\nu) = \begin{cases} 10 & , \text{if } \nu \leq 25 \\ 1.6\nu - 30 & , \text{otherwise} \end{cases} \quad (1)$$

The obtained trajectory on the map to the Route 3 using speed-controlled transmission is depicted in Figure 6. Figure 6 also shows that the regions where the car traveled with a reduced speed, there is a concentration of points, which we intend to avoid due to their redundancy. In the areas where the speed is higher, however, the speed-controlled strategy became feasible to be applied.

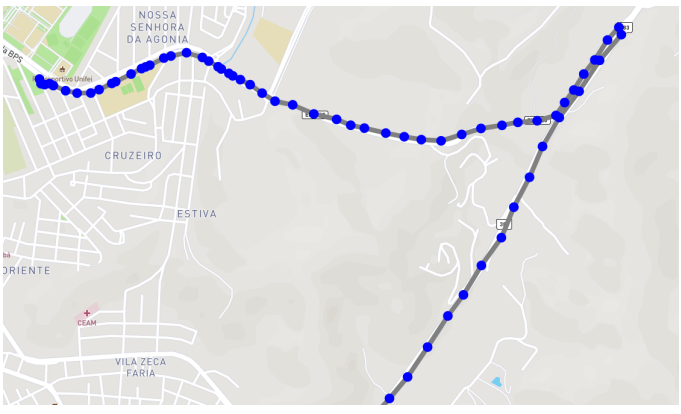

Figure 6. Transmission by speed control in Route 3.

Since fixed-time, fixed-distance and speed-controlled distance transmission methods present their advantages and disadvantages, a hybrid strategy was investigated, so as to combine the best characteristics of each method and establish the strategy to be used by the tracking device. The hybrid strategy consists in transmitting data by a fixed-time interval when the vehicle is traveling at a speed inferior to 60 km/h, and by speed-controlled distance when the vehicle is traveling at 60 km/h or more. That strategy allows to obtain more accurate tracings in small paths, where the amount of points sent will naturally be higher, due to the lower speeds. In addition, that criterion also enables network and memory savings on fast transit routes and highways, where fewer points will be sent due to the higher vehicle speed. Once Route 3 contains urban and highway sections, it was chosen to test and validate the hybrid solution. The result is illustrated in Figure 7.

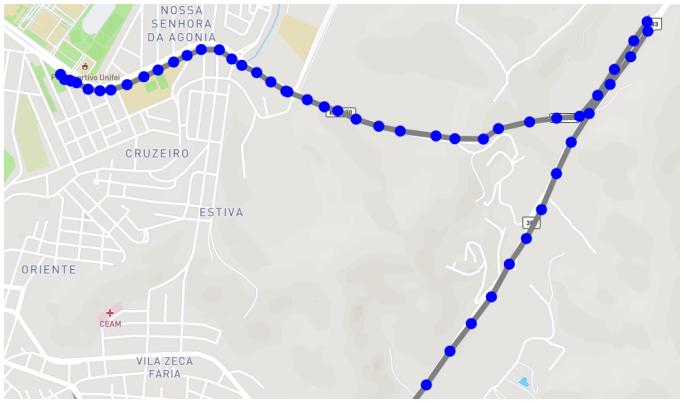

Figure 7. Hybrid transmission in Route 3.

The hybrid strategy proved to be interesting for the tracing on the map to be precise, and for the amount of points transmitted to the server to be reduced, which leads to better results with lower connection and energy consumption. In addition, it consistently manages small, medium and large routes.

Finally, in order to evaluate the implementation of the queue of points, the same trajectory illustrated in Figure 1 was traveled. Such a trajectory contains a shadow area, without cellular network coverage. The trajectory correction (green line) for that route is presented in Figure 8. Considering Figure 8, the queue implementation proved to be efficient and reconstructed the route. The time synchronization was also satisfactory, as the points kept increasing times, different from each other, and a delta value closer to the value configured on the device.

4. CONCLUSION

The present study delineated a device to solve the traceability problem. The device was constructed in terms of hardware, firmware and software, and each part presents specific aspects that contributes to achieve the solution. Our proposal does not require external devices, such as smartphones, and is able to reconstruct a traveled route even in the areas where the cellphone connection is lost. The route correction was implemented by using a queue scheme, where the points are stored and the entire queue transmitted to a server when the network connection is reestablished. That mechanism was indeed able to reconstruct the real trajectory of a vehicle even in the presence of shadow areas. Additionally, we also evaluated several distinct data transmission strategies aiming at saving connection and power energy. As a future work, we intend to implement the alternation among the transmission strategies by using fuzzy logic.

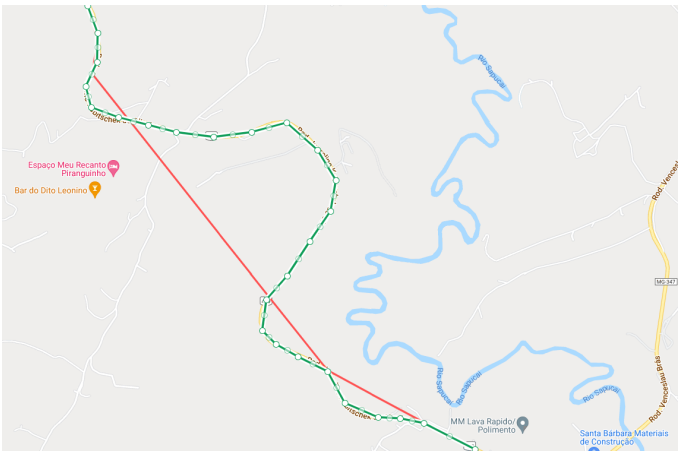

Figure 8. Trajectory correction in shadow area.

REFERENCES

Chabi, A.F., Guimaraes, J.C.B., Pinto, J.C., Furtado, R.S., Júnior, W.S., Carvalho, C.B., Santos, A.E., and Luiz, D.F. (2021). A IoT system for vehicle tracking using long range wide area network. In *IEEE International Conference on Consumer Electronics*, 1–2.

IBGE (2022). Frota de veículos. Available at: `https://cidades.ibge.gov.br/brasil/pesquisa/22/28120`; Access: 27/02/2023.

Jacobsen, R.H., Aliu, D., and Ebeid, E. (2017). A low-cost vehicle tracking platform using secure SMS. In *International Conference on Internet of Things, Big Data and Security*, 157–166.

Lee, S., Tewolde, G., and Kwon, J. (2014). Design and implementation of vehicle tracking system using GPS/GSM/GPRS technology and smartphone application. In *IEEE World Forum on IoT*, 353–358.

Minuto Seguros (2023). Qual o valor de seguro dos carros mais vendidos em 2023? Available at: `https://www.minutoseguros.com.br/blog/valor-seguro-carros-mais-vendidos`; Access: 03/05/2023.

Mustafa, A., A al Nouman, M.I., and Awad, O.A. (2019). A smart real-time tracking system using GSM/GPRS technologies. In *International Conference of Computer and Applied Sciences*, 169–174.

Oliveira, S. (2017). *Internet das Coisas com ESP8266, Arduino e Raspberry Pi*. Novatec, Brazil.

SSP-SP (2022). Estatísticas trimestrais. Available at: `http://www.ssp.sp.gov.br/Estatistica/Trimestrais.aspx`; Access: 28/02/2023.

Ya'u Gital, A., Abdulhamid, M., Abdulhameed, M., Zambuk, F.U., Nehemiah, M., Lawal, M.A., and Yakubu, Z.I. (2023). Review of GPS-GSM based intelligent speed assistance systems: Development and research opportunities. In *International Conference on Intelligent Communication and Computational Techniques*, 1–8.