IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 2 October 2024, accepted 16 October 2024, date of publication 21 October 2024, date of current version 5 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3483959

RESEARCH ARTICLE

P. I. E. N. O.–Petrol-Filling Itinerary Estimation aNd Optimization

MARCO SAVARESE, ANTONIO DE BLASI, CARMINE ZACCAGNINO, AND CARLO AUGUSTO GRAZIA, (Member, IEEE)
Department of Engineering "Enzo Ferrari," University of Modena and Reggio Emilia, 41125 Modena, Italy

Corresponding author: Marco Savarese (marco.savarese@pieno.dev)

This work was supported by the MOST–Sustainable Mobility National Research Center and received funding from the European Union Next-GenerationEU Piano Nazionale di Ripresa e Resilienza (PNRR)–Missione 4 Componente 2, Investimento 1.4–D.D. 1033 17/06/2022, under Grant CN00000023.

ABSTRACT The recent rise of intelligent transportation systems (ITS) has challenged the integration between different data sources. Reaching the goal of sustainable mobility requires properly managing and merging information coming from the vehicle (intra-) and information coming off the vehicle (inter-). In this paper, we provide a proof-of-concept leveraging on data merging between intra- and inter-networking presenting our framework: Petrol-Filling Itinerary Estimation aNd Optimization (PIENO). PIENO is a system that not only automates the search for the best fuel station but also paves the road to significant reductions in fuel consumption, making eco-driving a practical reality from a user perspective. The PIENO framework is designed to be fuel-type independent, ensuring its adaptability to different vehicles and conditions. It achieves this by merging data from the vehicle through a CAN Access Module (CAM) and data outside the vehicle through a mobile application connected to the internet. Different domains are stressed to reach the goal: microcontroller and OEM to retrieve the fuel level from the car, national authorities to retrieve the daily fuel price, AI models to predict the price trend for the next days, and algorithms to compute the best fuel station and the best time to fill. The modularity of PIENO allows it to adapt to different OEMs by modifying the intra-network interface to properly collect the fuel level, as well as to adapt to different markets and countries, retrieving the station's locations and fuel prices by modifying the inter-network interface.

INDEX TERMS Artificial Intelligence, cloud computing, edge computing, intelligent transportation systems, microcontrollers, mobile communication, mobile applications, smart mobility, smart transportation, time series analysis, vehicle-to-everything.

I. INTRODUCTION

These recent times, characterized by uncertainty, are the source of arduous challenges, among which, especially the environmental ones, make everyday life more difficult. The cause is to be sought, in particular, in excessive human-caused emissions, that between poor management of transportation and power generation represent the second highest risk factor for noncommunicable diseases [1]. Many solutions can be found in the transportation field, like transitioning to low-emission public transportation or switching to fully electric vehicles [2]. Yet, these solutions require time and funds that can not be invested in the short term. What can we do in the meantime? Thanks to modern communication technology, we have unprecedented opportunities to reduce our carbon footprint and optimize our trips. In particular, V2X communication technologies integrated with cloud computing enable the development of Intelligent Transportation Systems that can acquire data from vehicles and run optimization algorithms based on vehicle parameters and the surrounding area.

This is where PIENO fits in. PIENO is a framework that integrates existing technologies, such as OBD-II, CAN Bus, and cloud services, to enable the implementation of future evolutions of Intelligent Transportation Systems. In particular, PIENO intends to help drivers optimize their refueling routes. By acquiring the data from the vehicle and sending it to the cloud, PIENO creates a digital twin [3] of

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamed M. A. Moustafa.

the car, where information about fuel type, tank capacity, and fuel level are stored in order to guarantee a more precise and personalized optimization. Starting from this proof of concept, the end goal is to create a fully integrated and easy-to-use system accessible by every vehicle owner, regardless of their technical proficiency.

This paper is organized as follows. Section II introduces works related to PIENO. Section III presents PIENO's features and mission. Section IV shows the actual version of PIENO in a real-world scenario. Section V shows the results of the proof of concept and compares them to an early development version of PIENO (PIENO V0). Section VI is about future developments of PIENO. Section VII concludes the article.

II. RELATED WORK

One of the main emerging topics in the field of ITS is the so-called Software Defined Vehicles,[1] which is a trend in which software massively shape the customers experience integrating the vehicle into their digital lives. Despite the formal definition still missing, the SDV paradigm has already been exploited to figure out the characteristics of the future OS of new SDV-based vehicles [4], as well as renew existing applications like Firmware Over The Air (FOTA) [5]. The SDV approach has a key role in harmonizing the different E/E standardization among different Original Equipment Manufacturers (OEMs), reducing the complexity of future vehicle applications. SDV must not be confused with Software Defined Internet of Vehicles (SDIV) or, generally speaking, Software Defined Networks (SDN) solutions applied to inter-vehicle networks. The former has been exploited to analyze optimal routing protocol [6] or design novel RSU applications [7], while the latter has been exploited to address aspects like QoE in multi-radio-access technologies [8], delay-tolerant data transmission [9], stream management for time-sensitive vehicular networks [10], and cooperative applications [11] to name a few.

These papers take different state-of-the-art paradigms (SDIV, SDN, or generally Internet of Vehicles), modeling the vehicle as a network node without focusing on the challenges between the electronic-based intra-vehicle and computer-based inter-vehicle domains. From this perspective, PIENO is an SDV-ready framework. This means that the system can take advantage of the advancing SDV paradigm, simplifying data retrieval from the intra-vehicle network, where different OEM standardization plays a crucial role.

Existing works matching the same transition domain between the intra-vehicle data and the inter-vehicle collaboration typically deal with platooning [12], [13] where, based on the same OEM system and company goal, trucks are maneuvered together with the goal of sustainability in [12], or fuel-efficiency based on route speed in [13]. The former approach has been touched through a cooperative approach, outside the platooning use-case, recently in other papers more general on the ITS system [14], [15], [16], [17]. At the same time, the latter is applied to vehicles and infrastructure in [18], also adding real-time constraints in another paper dealing with general-purpose ground vehicles [19]. Continuing, monitoring the fuel level and the fuel consumption has been matched to AI algorithms considering the driving behavior by Sun et al. in [20].

Despite significant advancements in Intelligent Transportation Systems (ITS) and Software-Defined Vehicles (SDVs), existing works have primarily focused on either inter-vehicle communication through SDN-based routing protocols or specific use cases like platooning and cooperative vehicle systems. While these studies address inter-vehicle dynamics or intra-vehicle data management separately, they do not sufficiently tackle the integration challenges between the electronic-based intra-vehicle systems and the cloud-based inter-vehicle networks. PIENO bridges this critical gap by providing a holistic SDV-ready framework capable of real-time optimization of fuel consumption through the seamless integration of intra-vehicle data collection and inter-vehicle communication. Unlike existing solutions that focus on niche applications or specific OEM systems, PIENO's modular design ensures broad adaptability to diverse vehicle types and fuel systems, enabling personalized route optimization based on both fuel consumption patterns and real-time fuel prices. Moreover, PIENO goes beyond traditional route planning by incorporating predictive analytics for fuel prices, a feature absent in previous frameworks. This makes PIENO a comprehensive and scalable solution for sustainable mobility, offering practical eco-driving benefits to users while contributing to the broader goals of reducing emissions and optimizing transportation efficiency.

III. PIENO IN A NUTSHELL

PIENO stands as a versatile and fuel-type independent system that aims to adapt to any type of vehicle and to address key challenges faced by vehicle owners with a suite of features:

- **Real-Time Fuel Level Monitoring**: accurate, real-time updates of the vehicle's fuel level help users manage their fuel consumption more effectively;
- **Optimal Pump Locator**: purpose-built algorithms identify the optimal fuel pumps with the best prices and quickest routes, ensuring cost-effective and time-saving refueling;
- **Seamless Connectivity**: Bluetooth integration facilitates easy connectivity between the user's smartphone and vehicle, enabling effortless data exchange and control.

FIGURE 1. General structure.

As seen in the figure, PIENO's architecture is built on three main blocks:

[1]The SDV definition proposed by BOSCH: https://www.bosch-mobility.com/en/mobility-topics/software-defined-vehicle/

- **Intra-Vehicle Data Collection**: collection of fuel level data.
- **User Interface and Bridge**: retrieval of data from the car and communication with the cloud.
- **Cloud**: where the aforementioned algorithms are run.

A. INTRA-VEHICLE DATA COLLECTION

For what concerns the data collection from the car, we employ a personalized CAN Access Module (CAM) based on three main elements:

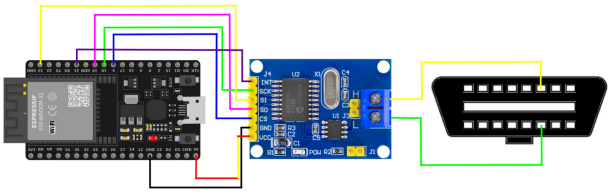


FIGURE 2. CAN access module.

- **OBD-II PORT**: acronym for On-Board Diagnostic II, OBD-II is a diagnostic tool that provides a communication interface with the vehicle that enables the reading and transmission of data measured by sensors and actuators of the vehicle in real-time [21]. Furthermore, OBD-II exposes a standard port, enabling anyone to access all kinds of messages exchanged on the various protocols that are built on vehicles. Specifically to PIENO, OBD-II is vital to interact with the CAN Bus protocol [22].
- **MCP2515**: a stand-alone CAN controller developed to simplify applications that require interfacing with a CAN Bus [23]. This module, by connecting to the CAN High and CAN Low pins of the OBDII port, works as a CAN Transceiver, facilitating the reading of vehicle data, particularly the fuel level, which is crucial for the PIENO system. It interfaces with microcontrollers via the Serial Peripheral Interface (SPI).
- **ESP32**: a single 2.4 GHz Wi-Fi-and-Bluetooth chip designed with the TSMC low-power 40 nm technology [24]. It plays a pivotal role in transmitting fuel-level data to the cloud. In particular, the PIENO system utilizes Bluetooth Low Energy (BLE) for efficient, low-power communication, ensuring continuous and real-time data flow from the vehicle to the cloud infrastructure by interfacing with a mobile device.

This approach was chosen due to its open-source nature, reproducibility, and accessibility. Furthermore, it helped solve one of the most arduous challenges during the development of PIENO: different OEMs have different yet proprietary methods to exchange messages on the CAN Bus. Normally, OEMs should follow the SAE standard J1979 [25], which defines a set of standardized codes (also known as OBD PIDs) to use in the communication between OBD and diagnostic tools. Yet, this is not the case for some of the car manufacturers. Taking that into account, the retrieval of the fuel level data might be different from vehicle to vehicle. Therefore, making use of solutions that rely on standardized diagnostic tools has been revealed to be impractical. In contrast, after proper reverse engineering sessions, the CAM is able to adapt to almost all kinds of message formats and standards of communication.

B. USER INTERFACE AND BRIDGE

Like a bridge between two worlds, the PIENO mobile application works as an intuitive interface to interact with the system, serving as the central hub where users can easily access all features, from checking their car status to locating the best fuel pumps. Additionally, it acts as the bridge between the CAM and the cloud infrastructure.

1) TECHNOLOGY CHOICE

To develop the mobile application, we used Flutter, a framework that allows for quick development of cross-platform apps without degrading performance and quality [26]. This allows our system to be used by as many people as possible since the app runs on both Android and iOS, but it could also run on embedded devices running Linux or other operating systems. It is also currently the most commonly used cross-platform mobile application framework since it has overtaken React Native [27].

As for the protocol used to connect the CAM and the app, we chose Bluetooth Low-Energy (BLE), since it is the most suited and optimized for the sort of communication our system requires, and also since it is widely supported, both by commonly available hardware and by widely-used software libraries.

2) FEATURE SET

- **Experience Customization**: registering an account allows the user to add specific information about their vehicles, making the calculation easier and more specific.
- **Vehicle Status**: the user can access their vehicles' information, including the fuel level, from their smartphone.
- **Price Trend Discovery**: every Sunday, a trained AI model provides the average fuel prices for the following week.
- **Reserve notification**: if a vehicle goes into reserve, the application notifies the owner by sending a push notification.

C. CLOUD

All of the features provided by PIENO are powered by a cloud-native backend service, which collects the information coming from all sources:

- Through HTTP requests made by the app for the information coming from the user or the CAM module
- From the Web, for the daily fuel price data provided by the government.

1) BACKEND INFRASTRUCTURE AND ARCHITECTURE

We followed a microservice architecture by splitting the operations performed on different kinds of data into different individual but interconnected software programs. For ease of

deployment and scaling, we containerized the application into OCI-compliant [28] containers, which we deployed on our own rented Oracle Linux 9 cloud instance using Docker.

Most of the app API, and most importantly, the fuel level monitoring part, is written in the Rust programming language. Rust is a compiled language that allows you to write programs that are as fast as C or C++ programs [29], [30] while being more modern and, most importantly, type-safe [31]. Therefore, Rust is perfect for dealing with the real-time constraints related to processing the data from the CAM and for integrating the information from various external sources, ensuring the system remains reliable under heavy loads.

Other parts, mostly those that run daily or weekly and that perform data-crunching or prediction, are written in Python, since it is more widely used for this sort of task, and allows for quicker development than Rust.

2) PATH FINDING AND NAVIGATION

For path-finding and navigation, many options were explored, from proprietary solutions like Google Maps and Mapbox API to other open-source alternatives like Open Source Routing Machine (OSRM). After many tests, OSRM stood out: by being a high-performance routing engine that leverages OpenStreetMap (OSM) data, it provides efficient and scalable routing solutions [32]. As an open-source platform, OSRM offers extensive customization options, allowing routing functionalities to be tailored to specific needs. In the context of PIENO, OSRM's flexibility and performance make it an invaluable tool. It enables the system to handle large volumes of routing requests efficiently, ensuring real-time and accurate routing for users. This is particularly beneficial for logistics and navigation within the app, enhancing the overall user experience without incurring high costs.

3) TIME SERIES FORECASTING

To perform the prediction, we used the Prophet, an open-source forecasting tool developed by Meta [33]. The choice fell on this model since it is specifically designed to work with time series data and it is particularly effective for datasets with daily observations that exhibit strong but varied seasonal effects.

To train Prophet, a publicly available dataset of fuel prices was used, made accessible by the Italian Ministry of Enterprises and Made in Italy.[2] This dataset contains comprehensive information on fuel prices from 2015 to 2024 divided by quarter and provided as CSV. For each dataset instance, we retained details on the region, fuel type, fuel price, and date.

The purpose of the prediction is to enable the user to understand whether, in the following days, the price will fall or rise and thus decide when to refuel. Multiple training sessions of the model were conducted using various time intervals. The best performance was achieved with data from 2021 to 2024, since it mitigates biases introduced by the COVID-19 pandemic, ensuring a more reliable and representative model performance.

[2]Accessible at https://www.mimit.gov.it/it/open-data/elenco-dataset/carburanti-archivio-prezzi

D. WORKFLOW

Given all the elements, we define a concise architecture (Figure 3) and workflow:

- **Data Acquisition**: fuel level data is acquired from the vehicle using the ESP32 and MCP2515 CAN module. This data is then transmitted to the mobile application via BLE.
- **Interface and Bridge**: the Flutter mobile application enables the user to interface with the system, check their vehicles' fuel level, and get information about the next day's price trends. Furthermore, it provides the backend with all the needed data about the vehicles.
- **Data Collection and Processing**: the high-performance backend, based on Rust and Flask, collects data on current fuel prices and station locations from national authorities and processes them. In addition, it uses OSRM to compute the best route to nearby fuel stations and Prophet to forecast future fuel prices.

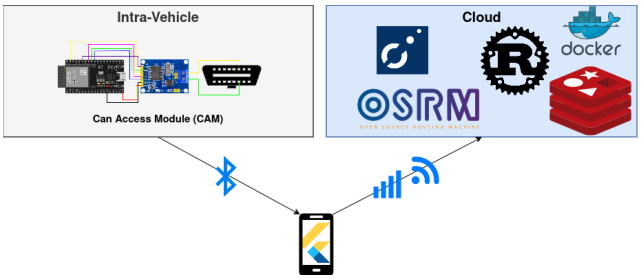


FIGURE 3. Final PIENO Structure.

The source code for the parts can be found in our repository on GitHub [34].

IV. PROOF OF CONCEPT

For this proof of concept, two cars were chosen: a Citroën C3 from 2007 and a Citroën C3 Aircross from 2017 (Figure 4).

FIGURE 4. Test subjects.

A. INTRA-VEHICLE DATA COLLECTION

First of all, we set up the CAM, following the official pin layout of Espressif [35] to identify the pins that drive the SPI protocol (Figure 5).

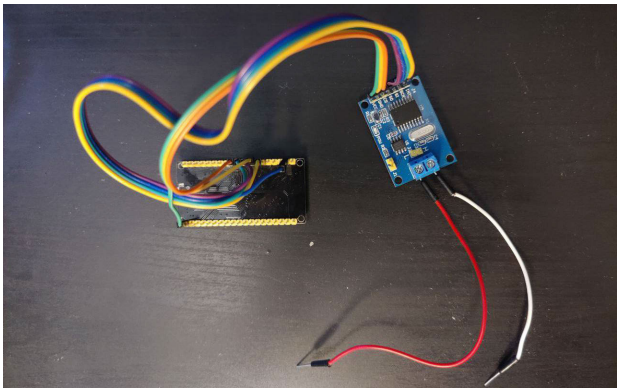


FIGURE 5. ESP32 - MCP2515 wiring.

For what concerns microcontroller programming, we started with a simple code that lists all the data acquired from the CAN Bus.

```
void loop() {
  struct can_frame frame;
  if (mcp2515.readMessage(&frame) == MCP2515::
    ERROR_OK) {
    Serial.println("Message received");
    Serial.print("ID: ");
    Serial.println(frame.can_id);
    Serial.print("Byte1: ");
    Serial.println(frame.data[0]);
    Serial.print("Byte2: ");
    Serial.println(frame.data[1]);
    Serial.print("Byte3: ");
    Serial.println(frame.data[2]);
    Serial.print("Byte4: ");
    Serial.println(frame.data[3]);
    Serial.print("Time: ");
    Serial.println(millis());
  }
}
```

LISTING 1. First general code.

To get the fuel level data, at first, we tried the specific SAE-J1979 OBD PID: 01 2F (from Hex to Dec: 303 in Big Endian, 12033 in Little Endian). Unfortunately, not having found any instance of CAN Bus messages with this ID, we concluded that Citroën does not follow the standard, so we needed to reverse-engineer the value.

The first test was conducted on the Citroën C3 Aircross since it exposes the OBD-II port without taking apart the car's dashboard. First of all, we decided to capture all the traffic on the CAN Bus while the car was stationary. Then, starting from a value of ∼10% tank, we went to the nearest gas station and filled up the tank a little bit (∼25%).

By excluding data with low variance and those whose maximum value was zero, we came down to three IDs (Figure 7 - 9).

By analyzing the data, we concluded that the value that better corresponded to the fuel level was ID 1554 - Byte 4. In particular, it seems that Citroën transmits a value between 0 and 100 on Byte 4 directly, as opposed to the SAE-J1979

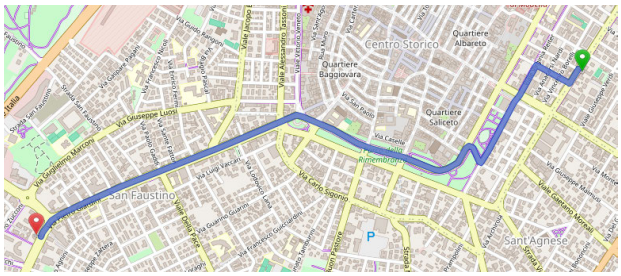


FIGURE 6. C3 2017 route to the nearest gas station.

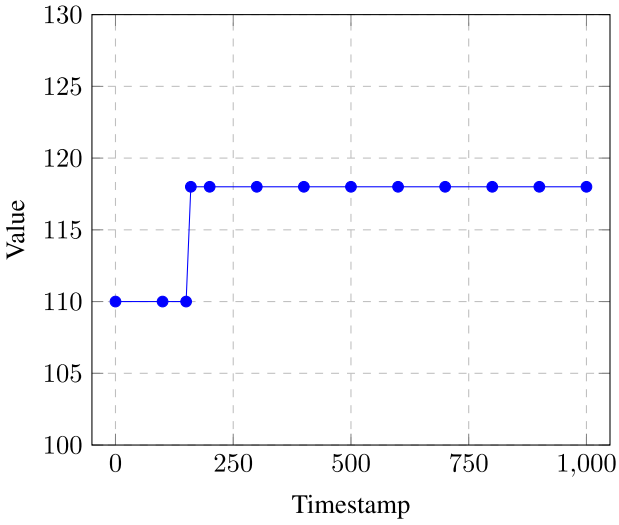


FIGURE 7. ID 1426 - Byte 2.

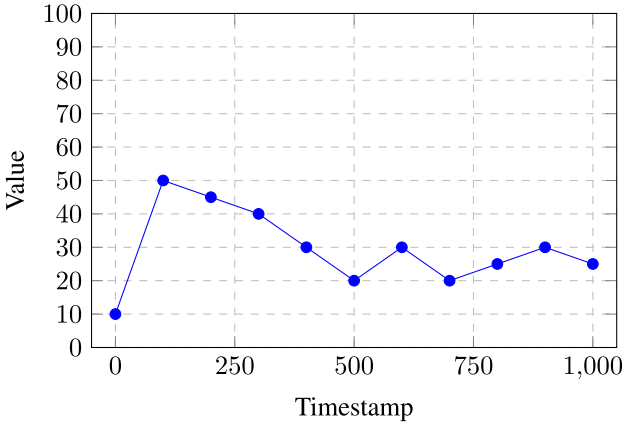


FIGURE 8. ID 1554 - Byte 4.

standard definition that specifies that the data in the fourth byte has to be transformed as follows:

$$\frac{100}{255} \cdot A. \tag{1}$$

Finally, having found the data, we repeated the test on the other Citroën, but this time, we filtered only the ID 1554 messages. Additionally, we started from 40% and filled it all the way up.

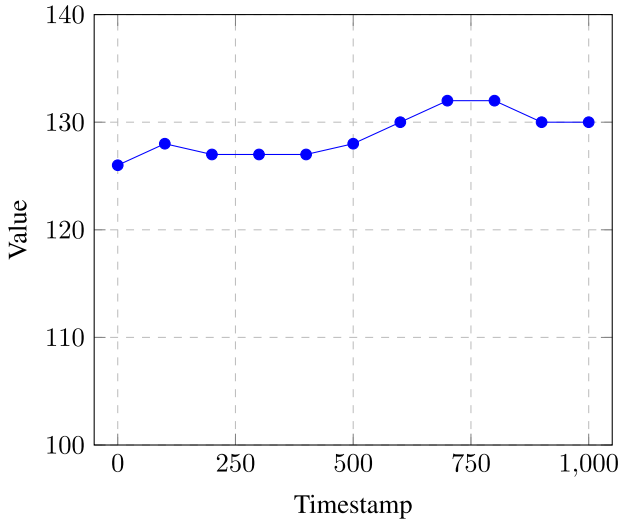


FIGURE 9. ID 813 - Byte 1.

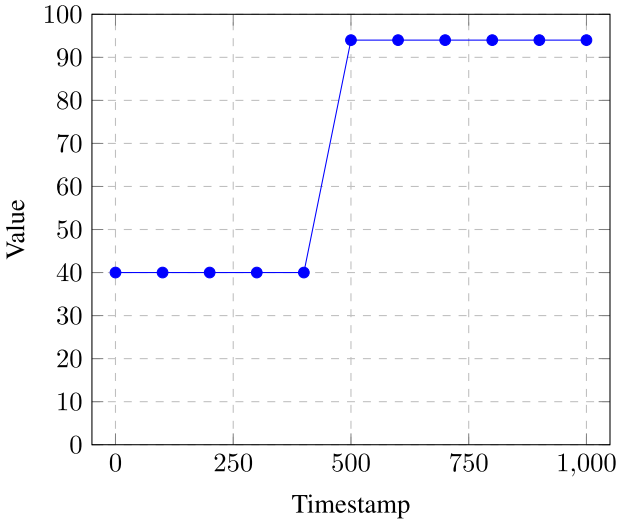


FIGURE 11. ID 1554 - Citroën C3 2007.

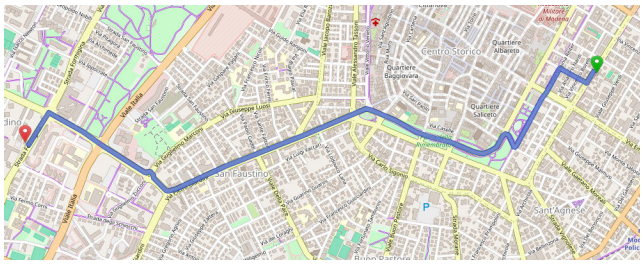


FIGURE 10. C3 2007 route to a cheaper gas station.

Using that logic and adding some code to implement BLE, the code turns into:

```
void loop(){
  struct can_frame frame;
  if (mcp2515.readMessage(&frame) == MCP2515::
    ERROR_OK) {
    Serial.println("Message received");
    if(frame.can_id == 1554) {
      /* same debug prints ... */
      int currFuelLevel = frame.data[3];
      if (currFuelLevel != fuelLevel){
        fuelLevel = currFuelLevel;
        pCharacteristic->setValue(fuelLevel);
        pCharacteristic->notify();
      }
    }
  }
}
```

LISTING 2. Specific code for Citroën.

As expected, the car followed the same pattern: ID 1554, and the byte value was always between 0 and 100 without any transformation needed.

As seen in Figure 11, the values are more stable since the Citroënn C3 2007 uses a digital meter. In contrast, the C3 Aircross from 2017 uses an electrical and more precise one, resulting in a consequent fluctuation around the threshold value.

The process we described to acquire data from these two cars could be extended to other makes and models to create a universal system by adding a configuration/calibration function to the CAM and the bridge. Further tests are being made, including various Lancia Ypsilon models and Ford Kuga 2010 as compatible vehicles. This would allow users of known cars to have the correct data automatically and simplify the reverse engineering process for users of new vehicles.

B. USER INTERFACE AND BRIDGE

The PIENO mobile application needs the user to register an account (Figure 12a). This is done to link any vehicle to the specific owner.

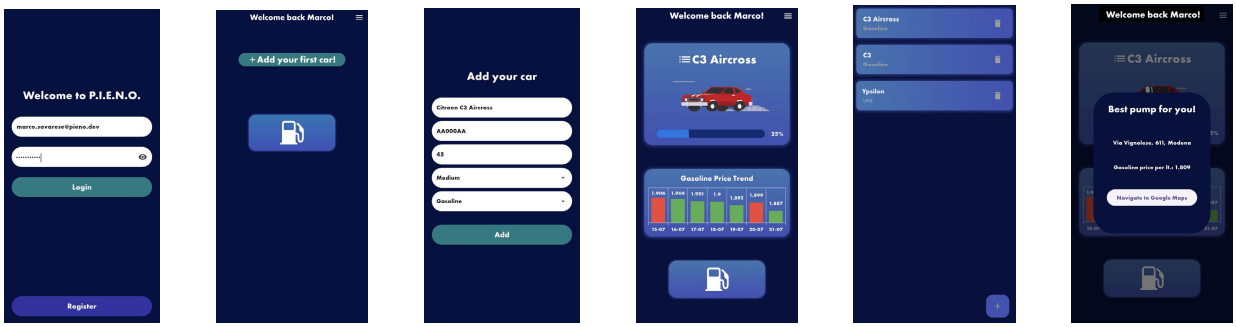
If the user has not added any vehicle to the system, the page in Figure 12b is shown. To function properly, the user must provide information about their vehicle. The information in Figure 12c is critical to the system:

- **Name** and **Plate Number** are useful to identify the vehicle and bind it to the user uniquely;
- **Tank Capacity**, **Car Size**, and **Fuel Type** help to provide a calculation based on the vehicle's parameter.

Once the car is added, the Homepage changes, as seen in Figure 12d. We can recognize three main blocks or containers:

- **First container**: it contains information about the name and fuel level of the vehicle. By clicking on the name, the user will be taken to the car list page (Figure 12e). Here, the user can edit his vehicle list by adding or removing them. Furthermore, by clicking on the name of the vehicle, the user can change the **active car** and, consequently, change the vehicle that is seen on the homepage.
- **Second container**: it shows the forecast is done by the prophet for the following week. It changes depending on the active car.
- **Third container**: with this button, it begins the calculation of the best fuel station. Each fuel station is graded by a score calculated as follows:

$$Score = A + B. \tag{2}$$



(a) Login Page. (b) Homepage without cars. (c) Form for adding vehicles. (d) Homepage with active car. (e) Vehicle List. (f) Instance of best fuel station found.

FIGURE 12. Application screenshots.

where

$$A = \text{cons\_per\_km} \cdot \text{distance}. \tag{3}$$

$$B = (100\% - \text{fuel\_level}) \cdot \text{price}_{lt}. \tag{4}$$

This way, we can always provide the best solution for refueling, making it an efficient process (Figure 12f).

C. CLOUD

The backend implementation is a rather standard REST API. Nonetheless, we note one aspect: we paid significant attention to ensuring potentially frequent fuel level updates from many cars don't cause issues. In order to do that, we developed two different decoupled microservices to handle that (Figure 13):

- The **Fuel Meter**, which exposes an HTTP API and pushes all requests to a queue in a Redis in-memory key-value database, is an inexpensive operation in terms of computation.
- The **Fuel Notifier**, which retrieves fuel level data from Redis with a given frequency, saves the data to the main SQL database and notifies the user to refuel through Firebase Cloud Messaging if the fuel level is low.

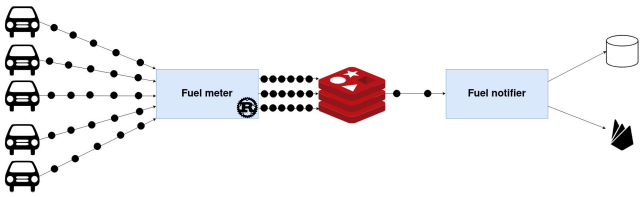


FIGURE 13. Decoupled fuel level acquisition architecture overview.

V. RESULTS

This section presents the results of the PIENO system's latest iteration and compares them with the performance and capabilities of its predecessor, PIENO V0. Through a series of rigorous tests and real-world implementations, we aim to highlight the advancements and improvements made in PIENO.

A. AI MODEL

When evaluating the prediction model, we compared our approach to the Random Forest [36] used in PIENO V0. A Random Forest is an ensemble machine-learning algorithm that builds multiple decision trees during training and merges their results to improve prediction accuracy. In particular, a decision tree for each fuel type was created.

As said in Section III-C3, our goal is not to predict the exact future value of fuel but solely to predict whether the price will increase or decrease.

For this purpose, the models were evaluated not by considering an exact match with ground truth but as a binary classification with two classes: decreasing price and increasing price. The predictions of both models were then compared with the actual values to see whether the predicted trend was correct or not.

TABLE 1. Accuracy of different models.

Model	Highest Accuracy (%)
Average accuracy of time series prediction	58
Random forest (PIENO V0)	67
Prophet (PIENO)	75

As shown in Table 1, Prophet has the highest accuracy at 75%, demonstrating superior reliability in forecasting trends.

To further stress the features of Prophet, in the graph in Figure 14 we compared a prediction with the actual average price per liter of gasoline for every day of the week from the 15^{th} July to 21^{st} July.

TABLE 2. Comparison with ground truth.

Metric	Score
Mean Square Error	0.00136
Average difference from ground truth	1.90%
Precision	80%
Recall	80%

As we can see, the values predicted by Prophet slightly differ from the actual values, as shown also in Table 2. Still, being exclusively interested in the trend, we can see that Prophet correctly predicts the rise or fall in price six times out of seven, confirming once more that it is the most suitable model for our purpose.

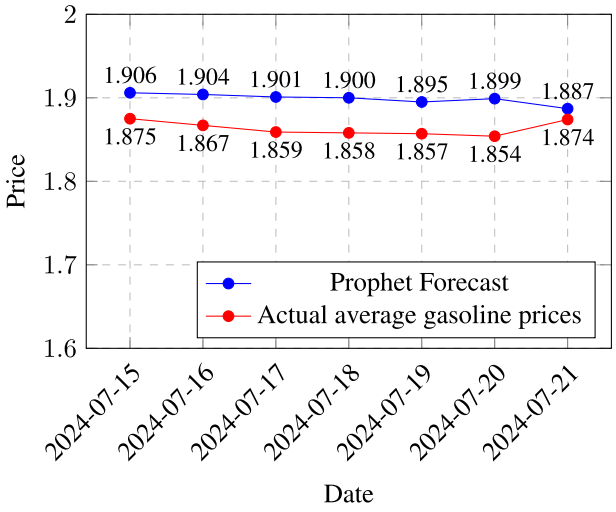


FIGURE 14. Prophet's forecast related to the actual average gasoline price.

B. BACKEND PERFORMANCE

To evaluate the effectiveness of the optimizations we described in Section IV-C, we used an HTTP load-tester[3] to test the number of requests the fuel meter microservice can handle in a second, and compared it with the number of requests that can be handled in the same time by the login page, which instead performs an SQL query and a hash computation to function, and we use as a proxy of a hypothetical fuel meter microservice that also acts as a fuel notifier.

The results in Table 3 show that the described optimizations indeed more than double the throughput.

TABLE 3. Backend performance evaluation.

Microservice	Average Req/sec
Login	974.2
Fuel-meter	**2084.81**

VI. FUTURE WORK

As of today, PIENO serves as a proof of concept that relies on specific datasets and is tested primarily on Citroën vehicles for intra-vehicle data collection. The next phases of development will focus on expanding the system's compatibility and ease of use. One significant area of expansion is integrating PIENO with various OEMs and international datasets. By collaborating with OEMs and working on expanding our APIs, PIENO can be adapted to a wider range of vehicle makes and models. Additionally, incorporating data from international fuel price databases will enable the system to function effectively in various countries, making PIENO a globally viable solution.

The CAN Access Module is currently a prototype requiring technical skills in the configuration process. Our goal is to transform this module into an all-in-one solution that is user-friendly and accessible to individuals with low technical proficiency. This involves developing a plug-and-play device with a more intuitive interface to ensure that any vehicle owner can benefit from PIENO's capabilities without needing specialized knowledge.

Lastly, ongoing improvements will focus on enhancing the AI models used for fuel price prediction and route optimization. By continuously training these models with larger and more diverse datasets, we aim to improve their accuracy and reliability. This will ensure that users receive the most efficient and cost-effective recommendations for their refueling needs.

[3]https://github.com/mcollina/autocannon.git

VII. CONCLUSION

PIENO represents a leap forward in vehicle management, trip optimization, and, by extension, the sustainability and effectiveness of intelligent transportation systems. Integrating real-time data, advanced analytics, and user-friendly interfaces offers a holistic solution that aims to reduce emissions, promote eco-friendly driving practices, and reduce the stress related to driving. Its robust architecture, leveraging state-of-the-art technologies such as ESP32, MCP2515, Flutter, Rust, Redis, OSRM, and Prophet, ensures adaptability, efficiency, and user-centric functionality. Through PIENO, sustainable driving becomes a practical and achievable goal for users worldwide. Moreover, PIENO is an SDV-ready framework, which means that the system can easily integrate the SDV paradigm and helps to close the gap between the intra-vehicle network, where different OEM standardization plays a crucial role, and inter-vehicle networks not completely deployed yet.

ACKNOWLEDGMENT

The authors Marco Savarese, Antonio De Blasi, and Carmine Zaccagnino thank Emiliano Maccaferri who provided the cloud backend for PIENO V0, which was the basis for the PIENO backend, Stefano Politanò and Francesco Zampirollo for their help in developing PIENO V0 and professor Roberto Vezzani who gave the first push to PIENO development. This manuscript reflects only their views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

REFERENCES

[1] WHO. (2022). *Air Pollution*. Accessed: Jul. 20, 2024. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health

[2] *Aria Tossica: Il Costo Dei Combustibili Fossili*, CREA, Jangpura, India, 2020.

[3] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108952–108971, 2020.

[4] J. Becker, "Operating system for software-defined vehicles," *ATZelectronics Worldwide*, vol. 17, no. 5, pp. 40–45, May 2022.

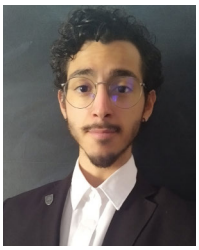
[5] A. W. Malik, A. U. Rahman, A. Ahmad, and M. M. D. Santos, "Over-the-air software-defined vehicle updates using federated fog environment," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 4, pp. 5078–5089, Dec. 2022.

[6] K. Z. Ghafoor, L. Kong, D. B. Rawat, E. Hosseini, and A. S. Sadiq, "Quality of service aware routing protocol in software-defined Internet of Vehicles," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2817–2828, Apr. 2019.

[7] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Software-defined networking for RSU clouds in support of the Internet of Vehicles," *IEEE Internet Things J.*, vol. 2, no. 2, pp. 133–144, Apr. 2015.

[8] W. Huang, L. Ding, D. Meng, J.-N. Hwang, Y. Xu, and W. Zhang, "QoE-based resource allocation for heterogeneous multi-radio communication in software-defined vehicle networks," *IEEE Access*, vol. 6, pp. 3387–3399, 2018.
[9] B. Xia, F. Kong, J. Zhou, X. Tang, and H. Gong, "A delay-tolerant data transmission scheme for Internet of Vehicles based on software defined cloud-fog networks," *IEEE Access*, vol. 8, pp. 65911–65922, 2020.
[10] S. Nam, H. Kim, and S.-G. Min, "Simplified stream reservation protocol over software-defined networks for in-vehicle time-sensitive networking," *IEEE Access*, vol. 9, pp. 84700–84711, 2021.
[11] D. Dias, M. Luís, P. Rito, and S. Sargento, "A software defined vehicular network using cooperative intelligent transport system messages," *IEEE Access*, vol. 12, pp. 93152–93170, 2024.
[12] F. Giannini, G. Franzè, F. Pupo, and G. Fortino, "A sustainable multi-agent routing algorithm for vehicle platoons in urban networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 14830–14840, Dec. 2023.
[13] G. Guo and Q. Wang, "Fuel-efficient en route speed planning and tracking control of truck platoons," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 8, pp. 3091–3103, Aug. 2019.
[14] C. Luo, J. Chen, X. Feng, J. Zhang, and J. Li, "BSL: Sustainable collaborative inference in intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 15995–16005, Dec. 2023.
[15] B. Fong, A. C. M. Fong, and G. Y. Hong, "Sustainable micromobility management in smart cities," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 15890–15896, Dec. 2023.
[16] H. Li, Y. Chen, K. Li, C. Wang, and B. Chen, "Transportation Internet: A sustainable solution for intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 15818–15829, Dec. 2023.
[17] M. Xia, J. Lin, L. Ying, J. Sun, K. Chi, K. Gao, and K. Yu, "Toward sustainable transportation: Robust lane-change monitoring with a single back view cabin camera," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 15414–15424, Dec. 2023.
[18] P. Typaldos, I. Papamichail, and M. Papageorgiou, "Minimization of fuel consumption for vehicle trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1716–1727, Apr. 2020.
[19] J. Kim and C. Ahn, "Real-time speed trajectory planning for minimum fuel consumption of a ground vehicle," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2324–2338, Jun. 2020.
[20] H. Sun, F. Tao, Z. Fu, A. Gao, and L. Jiao, "Driving-behavior-aware optimal energy management strategy for multi-source fuel cell hybrid electric vehicles based on adaptive soft deep-reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 4127–4146, Apr. 2023.
[21] G. Signoretti, M. Silva, A. Dias, I. Silva, D. Silva, and P. Ferrari, "Performance evaluation of an edge OBD-II device for industry 4.0," in *Proc. II Workshop Metrol. Ind. 4.0 IoT (MetroInd4.0&IoT)*, Jun. 2019, pp. 432–437.
[22] *CAN Specification Version 2.0*, Robert Bosch GmbH, Stuttgart, Germany, Sep. 1991.
[23] "Stand-alone CAN controller with SPI interface," Microchip Inc., Tech. Rep., 2003.
[24] "ESP32 ser. Datasheet," Espressif, Shanghai, China, Tech. Rep., 2023.
[25] *E/E Diagnostic Test Modes*, Standard J1979_201702, SAE Int., Feb. 2017.
[26] S. Gowri, C. Kanmani Pappa, T. Tamilvizhi, L. Nelson, and R. Surendran, "Intelligent analysis on frameworks for mobile app development," in *Proc. 5th Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Jan. 2023, pp. 1506–1512.
[27] C. Zaccagnino, "Programming flutter," Pragmatic Bookshelf, Raleigh, NC, USA, Tech. Rep., 2020.
[28] Open Container Initiative The Linux Foundation. *OCI Image Format Specification*. Accessed: Jul. 20, 2024. [Online]. Available: https://github.com/opencontainers/image-spec/?tab=readme-ov-file
[29] M. P. Rooney, N. Rao, N. Liebers, A. S. Leger, and S. J. Matthews, "A comparative study of programming languages for a real-time smart grid application," in *Proc. IEEE Green Energy Smart Syst. Conf. (IGESSC)*, Oct. 2023, pp. 1–6.
[30] J. Rotter and M. C. Lewis, "N-body performance with a kD-tree: Comparing rust to other languages," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2022, pp. 457–462.
[31] N. D. Matsakis and F. S. Klock, "The rust language," in *Proc. ACM SIGAda Annu. Conf. High Integrity Lang. Technol.* New York, NY, USA: ACM, Oct. 2014, pp. 103–104.
[32] D. Luxen and C. Vetter, "Real-time routing with OpenStreetMap data," in *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, New York, NY, USA, Nov. 2011, pp. 513–516.
[33] S. J. Taylor and B. Letham, "Forecasting at scale," *Amer. Statistician*, vol. 72, no. 1, pp. 37–45, 2018, doi: 10.1080/00031305.2017.1380080.
[34] M. Savarese, A. D. Blasi, and C. Zaccagnino. (Jul. 2024). *P.I.E.N.O.* [Online]. Available: https://github.com/P-I-E-N-O
[35] Espressif. (2016). *ESP32-DevKitC V4 Getting Started Guide*. Accessed: Jul. 17, 2024. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/hw-reference/esp32/get-started-devkitc.html
[36] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, Oct. 2001.

MARCO SAVARESE received the bachelor's degree in computer engineering from UNIMORE, in 2023, where he is currently pursuing the master's degree in cloud and cyber security. He is developing research interests in the smart mobility and automotive connectivity field.



ANTONIO DE BLASI received the bachelor's degree in computer engineering from UNIMORE, in 2022, where he is currently pursuing the master's degree in artificial intelligence engineering–large scale. He is developing research interests in the analysis, development, and training of multimodal large-scale architecture.

CARMINE ZACCAGNINO received the bachelor's degree in computer engineering from UNIMORE, in 2022, where he is currently pursuing the master's degree in computer engineering. He has multi-year professional software engineering experience and has written a book about Flutter, which was published in two languages. His current main research interests are in AI and computer vision, particularly in the field of document analysis.

CARLO AUGUSTO GRAZIA (Member, IEEE) received the Ph.D. degree in ICT from UNIMORE, in 2016. He is currently a tenure-track Assistant Professor with DIEF, UNIMORE, where he teaches "Automotive Connectivity" in the M.S. of electronic engineering for intelligent vehicles, computer engineering, and electronics engineering. His research interests include computer networking, with an emphasis on queueing algorithms and V2X.

Open Access funding provided by 'Università degli Studi di Modena e Reggio Emilia' within the CRUI CARE Agreement