



System for Monitoring and Control of Vehicle's Carbon Emissions Using Embedded Hardwares and Cloud Applications

Tsvetan Tsokov^(✉) and Hristo Kostadinov

Institute of Mathematics and Informatics,
Bulgarian Academy of Sciences, Sofia, Bulgaria
hristo@math.bas.bg

Abstract. Today, the electronic devices such as sensors and actuators, forming the Internet of Things (IoT) are presented naturally in the people's day to day life. Billions of devices are sensing and acting upon the physical world and exchange information. All major industries like transportation, manufacturing, healthcare, agriculture, etc. adopt IoT solutions. Not only people and industry are affected in a positive way by IoT, but also the nature and environment. The IoT is recognized as a key lever in the urge to save the climate. It has a major potential in reducing air carbon emissions and pollution. Taking into account the promising sectors of IoT application, this paper proposes a solution for monitoring and control of carbon emissions from vehicles. It consists of hardware device that ingests data related to vehicles' carbon emissions and cloud based services for data storage, analysis and representation. It controls the carbon emissions via notifications and vehicle's power restrictions.

Keywords: Clustering · Internet of things · Reduction of carbon emissions · Sensor data processing

1 Introduction

Nowadays, the smart physical devices are part of the everyday people life and generate huge amount of data, which drives applications and services on top of them, establishing the Internet of Things (IoT) ecosystem [1]. Applications are developed to ingest, store and analyze large amounts of data generated by the IoT devices in all economic sectors such as transportation, agriculture, health and education. According to CISCO Internet Business Solutions Group (CISCO IBSG), the total number of interconnected devices by 2015 was 25 billions and it's expected to be 50 billion by 2020 [2]. For the same time period it is expected 5.8 billion IoT endpoints and global economic revenue from endpoint electronics to total 389 billion US dollars.

One of the most impacted sectors by the IoT is the automotive industry. Recently, tens of millions of cars are said to be connected to the Internet and their number is expected to become hundreds of millions in the near future.

According to a global industry analysis, the number of connected vehicles will become 125 million by 2022. In order to enable such a massive amount of connected vehicles to communicate with other IoT endpoints in real time, new network bridges and protocols are developed [3]. At the same time, mobile network technology is recognized to have considerable potential to enable carbon emissions reduction across a wide range of sectors. Currently, 70% of the carbon savings come from the use of machine-to-machine (M2M) technologies, according to Global e-Sustainability Initiative. The survey data shows that 68% of smartphone users are willing to adopt behaviours that could result in even more future reductions to personal carbon emissions. According to another study the power consumption of the radio access networks (RAN) and the production of mobile devices are major contributors in the carbon emissions [4]. There it is concluded that the usage of green technologies to reduce the power consumption offer big potential for reduction of the emissions. IoT is pointed as a key lever to reduce the carbon emissions.

The current hardware and software solutions for tracking vehicles give evidence for the efforts of using IoT technologies in automotive industry. Geotab provides a service for monitoring and analysis of vehicles using integrated hardware module that sends data to private cloud. The hardware module is connected to the onboard diagnostic bus of the vehicle and collects data about fuel consumption, traveled distance and other parameters. The data analysis, which is made in the cloud allows identification of vehicles with not optimal fuel consumption. Unfortunately, Geotab solution does not detect increased rate of carbon emissions and provide control over vehicle's parameters. Madgetech is a data logger, which provides functionality for regular monitoring of carbon dioxide levels (Data Loggers). It measures the carbon emissions by exhaust gas sensors in vehicles and sends data to private cloud. The measured data is visualized by mobile application, but analysis and control of the carbon emissions are not supported. In addition to these existing solutions, there are mandatory emissions inspection and certification procedures in many countries. But these procedures usually are done only one or several times per year, which is a small frequency and can not guarantee that the vehicles are working with optimal emissions for big period of time.

Inspired by the low-carbon roadmap of European union and the grate potential provided by the IoT technologies for reducing the carbon emissions, in this paper we propose a solution for real-time monitoring and detection of rising levels of carbon emissions from vehicles, called EcoLogic. The proposed solution includes hardware module, which collects sensor data related to vehicle's carbon emissions such as air pressure, air temperature and fuel mixture. The data is transferred to a cloud-based applications, where it's stored and analysed. The results from the analysis are used to control the carbon emissions through driver notifications and vehicle's power limitations. The source code of the main software components of the solution is publicly available and can be accessed and downloaded from the following locations: <https://github.com/ttsokov/vehicle-monitor-controller-hw-controller>, <https://github.com/ttsokov/vehicle-monitor-controller-hw-proxy>, <https://github.com/ttsokov/vehicle-monitor-controller-backend>. The repositories contain information on how to setup the projects and deploy them on hardware device and

server or in the cloud. The source code and hardware modules are with open-source licenses, because the purpose of the solution is to be freely reviewed, used and extended by everyone who is interested to track and reduce emissions at global scale. Additionally the system can be integrated with the countries tax system supporting drivers with small emissions footprint to pay smaller taxes.

The rest of the paper is organized as follows. Section 2 presents the architecture of EcoLogic with its components, interconnections and protocols. Section 3 describes its components in implementation details and why the concrete technologies are used. Section 4 shows a case study that validates the feasibility of the proposed solution by comparing the results of the algorithm from two datasets: test dataset and real dataset. Finally, Sect. 5 concludes our research by giving the pros and cons of the system and points out directions for future work.

2 System Design

This section presents the architecture of EcoLogic system with its components, communication paths and protocols.

The EcoLogic system is composed of two big parts: hardware modules and cloud-native microservice applications. The hardware modules are located in vehicles and the microservice applications are deployed on a cloud platform. The architecture of the system is shown in Fig. 1. The history of physical computing technology showed us that each physical computing system contains at least one hardware device, but nowadays the Internet of Things paradigm, which extends the physical computing technology, is showing us that except hardware devices each IoT system should use also Cloud or Fog platform, which enables massive scale, reliability and efficiency. It becomes a standard for IoT systems to incorporate Cloud and Fog computing platforms in its architectures [5] as main components that give a lot of benefits like enormous computing power for artificial intelligence algorithms, including machine learning and data mining, high availability, disaster recovery, big storage and near real-time speeds.

The hardware module of the system has sensors and measures several physical parameters. It can also extract parameters from the onboard diagnostic system of the vehicle. The data is sent to cloud-native applications in the cloud platform. The measured physical parameters are:

- *Air/fuel ratio*, which is measured by lambda sonde sensor, which is located into the exhaust system of the vehicle.
- *Absolute pressure of the air* that is consumed by the engine, which is measured by sensor located in the air intake manifold of the vehicle.
- *Temperature of the air* that is consumed by the engine, which is measured by sensor located in the air intake manifold of the vehicle.

The cloud-native applications are implemented as microservices, which are designed in a platform agnostic way in order to have the possibility for deployment on different cloud platforms. The cloud applications store data in a relational database, which is represented by backing service from the cloud platform. They process the incoming data, store it into the database and analyse it.

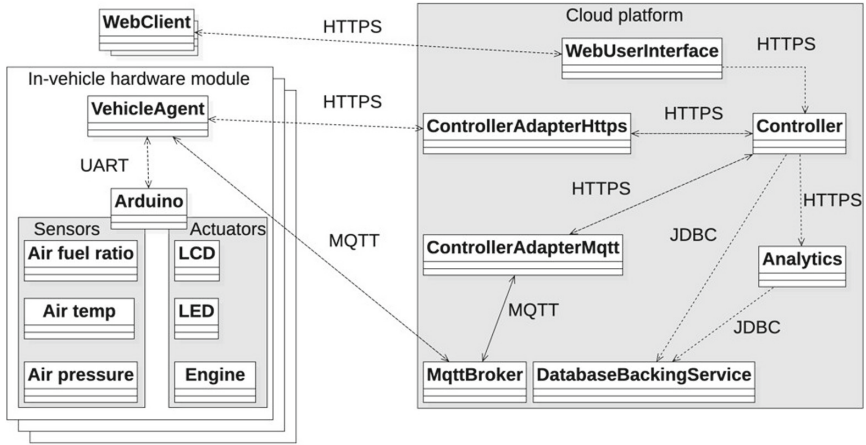


Fig. 1. EcoLogic general architecture

The hardware modules communicate with the cloud with wireless network via HTTPS or MQTT protocols. The following physical parameters are calculated on top of the incoming sensor data:

- Mass of the consumed air by the engine;
- Mass of the consumed fuel by the engine;
- Mass of the carbon dioxide emissions, exposed into the atmosphere.

The database contains all measured and calculated physical parameters. A cloud-native Analytics application executes an anomaly detection on the streamed data and outlines vehicles that have not optimal amount of carbon dioxide emissions or system failures. Clustering analysis is made in order to detect anomalies. The hardware module is notified when some vehicle is detected by the system as an anomaly, with suboptimal amount of emissions. Then the hardware actuator is started automatically to reduce the amount of emissions. This comprises a feedback control loop. In this way the system monitors and controls the amount of carbon dioxide emissions in the atmosphere in real time. The hardware modules are equipped with three actuators:

- *Liquid crystal display (LCD)*, which visualize the measured and calculated physical parameters to the driver.
- *Light-emitting diode (LED)*, which indicates to the driver that the amount of carbon dioxide emissions is not optimal or there is a system failure (not optimal parameters).
- *Actuator*, which controls the amount of injected fuel in the engine and regulates the amount of emissions.

Currently, only the display and LED actuator are implemented in the EcoLogic. The purpose of the LED actuator is to notify the driver to intentionally reduce the acceleration and change the driving behaviour, which leads to reduction of the amount of burned fuel and emissions.

The cloud applications provide web user interface, on the base of HTML5, JavaScript and CSS resources. It's endpoint is publicly available and accessible by clients via HTTPS protocol.

The user management of the system is composed of two roles: driver and operator. The lifecycle of the system is the following:

- Dealer sells a hardware module to a driver.
- The driver installs the hardware module into vehicle. This step can also be accomplished by an authorized service.
- The driver registers the vehicle with the hardware module and sensors via the web user interface. All components have unique identifiers.
- Drivers are authorized to monitor and control their own registered vehicles.
- Operators are authorized to monitor and control all registered vehicles by regions.
- Each driver gets score points proportional to the amount of carbon dioxide emissions exposed in the atmosphere from their vehicles.
- The score points can be integrated with tax systems of countries and city halls. In this way the taxes can be reduced proportionally to the score points. Drivers can also participate in greenhouse gas trading schemes with their score points.

Important aspect of the design of the system is it's security. Security is major problem in all IoT applications, because of the big number of heterogeneous devices, the data, which is distributed across many locations and the need for high-speed communication. New protocols and schemes that incorporate symmetric and asymmetric cryptography are developed [6] in order to resolve these problems. The current system uses TLS v1.2 (TLS protocol) with it's built-in asymmetric and symmetric cryptography for the network communication between components.

3 System Implementation Elements

This section outlines the main components of EcoLogic system. First, the hardware module with it's components and the algorithm for calculation of carbon emissions are presented. Later, the cloud applications are described and the algorithm for data analysis.

3.1 Embedded Hardware System

The embedded hardware system contains two embedded subsystems: Arduino Uno and Raspberry Pi B+. The Arduino operates on the low-level hardware sensors and actuators in the vehicles, while the Raspberry Pi works on higher level. It aggregates the data from the low-level hardware and communicates with the cloud platform in bidirectional way.

Arduino Embedded Hardware System. The Arduino Uno embedded system has the following capabilities:

- Measure physical parameters of internal combustion engine.
- Visualize the measured parameters on 4×16 liquid crystal display.
- Control of actuator (light emitting diode).
- Communication with Raspberry Pi embedded system.

The physical parameters are measured in two ways: by sensors or extracted from the onboard diagnostic system (OBD2), which is provided by the electronic control module of the vehicle. If the onboard diagnostic interface exposes the necessary parameters in the Application Programming Interface (API), no additional sensors will be installed in the vehicle. If the onboard diagnostic interface API does not expose the necessary parameters, additional sensors, which measure these parameters, will be integrated. There is also a possibility to reuse the already existing default sensors of the vehicle. In this way the hardware module is flexible and can be installed on huge amount of vehicles.

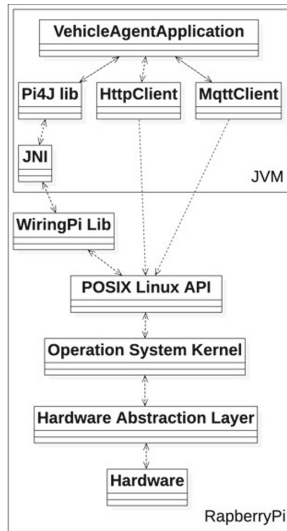


Fig. 2. Raspberry Pi architecture

The Arduino embedded system has deployed application in its local storage and executes it. The application is implemented on the C++ programming language. It is publicly available and can be downloaded from the following location: <https://github.com/ttsokov/vehicle-monitor-controller-hw-controller> (vehicle-monitor-controller-hw-controller source code). The repository from the link contains information for the build, deployment and execution on local Arduino-based hardware module. The application consumes Wiring

library, which is part of the Arduino SDK (Software Development Kit). The Wiring library executes on the concrete Arduino microcontroller via drivers. Currently, the Arduino Uno hardware is used, which has Microchip ATmega328 microcontroller based on RISC architecture. This hardware module executes a critical calculations from physical sensors in real time. This is the reason why it's a separate module with own independent computational resources and with software application written in fast low-level programming language like C++.

Raspberry Pi Embedded Hardware System. The Raspberry Pi B+ embedded system delegates the communication between the Arduino embedded system and the applications in the cloud platform. Its architecture is shown in Fig. 2.

The communication between the Raspberry Pi B+ embedded system and the Arduino embedded system is via serial UART (Universal Asynchronous Receiver/Transmitter) protocol with a custom application messaging protocol on top of it. The Raspberry Pi B+ embedded system is connected to the cloud platform via 4G broadband cellular network. The measured physical parameters by the Arduino embedded system are ingested into the Raspberry Pi B+ embedded system, which caches them in a local cache storage for further processing and sends them to the cloud platform. It sends the data to the cloud platform via HTTPS or MQTT protocol. The Adapter cloud-application is the facade, which is the only endpoint seen by the Raspberry Pi B+ embedded system. When it sends the data to the cloud platform, it receives response that contain information about the state of the vehicle, including the amount of the carbon dioxide emissions. There are two possible states: optimal (eco) and not optimal (not eco). The Raspberry Pi module notifies the Arduino embedded system once the emissions are not optimal. Then the Arduino embedded system activates the low-level hardware actuator in order to reduce the quantity of emissions. The main component inside the Raspberry Pi embedded hardware is the System on a Chip (SoC), which is using ARM architecture and Linux based operation system, currently Raspbian. It executes the VehicleAgent application, which is implemented in the Java programming language and it runs on top of a Java Virtual Machine. The VehicleAgent application is publicly available and can be downloaded from the following location: <https://github.com/ttsokov/vehicle-monitor-controller-hw-proxy> (vehicle-monitor-controller-hw-proxy source code). The repository from the link contains information on how to build the application locally, how to deploy it on any Raspberry Pi-based hardware module and how to execute it.

The VehicleAgent application depends on the Pi4J and WiringPi libraries, which support the serial communication between the two embedded systems. An USB WiFi adapter is used for the 802.11n communication between the Raspberry Pi embedded hardware and a hotspot, which provides the 4G broadband cellular network to the cloud platform. The adapter is connected to one of the USB ports of the Raspberry Pi. The communication with the cloud platform can be achieved by several application layer protocols: HTTPS or MQTT.

The decision which protocol will be used is taken according to the supported protocol by the cloud platform. The modular architecture of the application enables easy extension with other application layer protocols. Because the communication network to the cloud platform is constrained it is expected to exist glitches and that's why the application caches the latest data in the local cache storage. After successful reestablishment of the connection with the cloud platform, the application retries to send the locally cached data. The application can be configured by configuration file (config.xml), which is located into the local file system storage. It contains unique identifiers for the vehicle, sensors and type of the communication with the cloud platform (HTTPS, MQTT, etc.). The driver or operator, who registers the vehicle into the system, should populate the configuration file and save it.

This hardware module serves as a communication proxy and need to have capabilities to work with a lot of communication networks and protocols. This is the reason why it's a separate hardware module, which can work with many hardware adapters or antennas supporting different communication networks. The software application is written in high-level programming language like Java, because it needs to work with many communication protocols and serialize data in different formats.

3.2 Cloud Software Applications

The EcoLogic is composed of several cloud- native microservice applications, namely Controller application, Adapters applications, Web user interface and Analytics application. Each application is described in the following subsections.

Controller Application. The Controller application is the main cloud application, which manages all vehicles with their hardware modules and sensors, make calculations, persists data into database, executes artificial intelligence algorithms on the data and provides HTTP REST API. It is implemented using the Java Enterprise Edition programming language using JPA and Apache CXF services framework. The Controller application is publicly available and can be downloaded from the following location: <https://github.com/ttsokov/vehicle-monitor-controller-backend> (vehicle-monitor-controller-backend source code). The repository from the link contains information on how to build the application locally, how to deploy it on a web container in any cloud platform or local server and how to execute it. The application has the following functionality:

- Compose a data model, which has the following relations: users, which have vehicles, which have sensors, which have measurements of physical parameters.
- Orchestrates the lifecycle of all users, vehicles, sensors and measurements.
- Persists the data into a relational database, which is provided as a backing service exposed by the cloud platform. The application is database-agnostic, which means that it can work with any relational database, which provides Java connectivity. If it does not provide Java connectivity, an adapter can be used. The database is composed of four tables: User, Vehicle, Sensor and Measurement.

- Calculates the total mass of the carbon dioxide emissions disposed into the atmosphere by vehicles.
- Manages the state of each vehicle: optimal (eco) state and not optimal (not eco) state. This is a simple state machine with two states.
- Calls the API of an Analytics application, which executes clustering analysis and anomaly detection. The purpose is to find vehicles, which have not optimal amount of carbon dioxide emissions per region.
- Provides HTTP REST API which is used by the Adapter applications and web user interface.

Adapter Application. The data coming from the vehicle hardware modules is intercepted and adapted by the Adapter cloud-native applications. Then it is routed to the Controller application. The Adapter applications are implemented using the Java programming language. Currently there are two types of Adapter applications that handle HTTPS and MQTT network protocols: `ControllerAdapterHttps` and `ControllerAdapterMqtt`, respectively. `ControllerAdapterMqtt` application communicates with MQTT broker. It is broker independent, so: Mosquitto, HiveMQ, Mosca or other type of MQTT broker can be used. The MQTT broker and the Adapter applications have publicly available URL endpoints, which are called by the vehicle hardware modules. The traffic is routed by the cloud platform to the concrete application depending on the application layer protocol that is used. The traffic is routed to the `ControllerAdapterHttps` application if HTTPS protocol is used. The traffic is routed to the MQTT broker if MQTT protocol is used. This routing behaviour of the cloud platform is based on TCP routing mechanism. TCP routing enables cloud platforms to support applications, which communicate with different non-HTTP protocols. TCP routing is used by one of the industry standard cloud platforms, called Cloud Foundry (Cloud Foundry TCP Routing).

The flow of the MQTT traffic is the following:

1. On creation of new vehicle, the Controller application registers two topics with names `vehicles/{id}/sensors/{id}/measurements` and `vehicles/{id}/state`. The `ControllerAdapterMqtt` application subscribes for that topics.
2. The concrete hardware module also subscribes to both topics and start to publish new measurements to `vehicles/{id}/sensors/{id}/measurements` topic.
3. On receiving of new message with measurement by the `ControllerAddapterMqtt` application, it adapts the measurement and sends it to the Controller application via the HTTP protocol.
4. The `ControllerAddapterMqtt` application receives the state of the concrete vehicle and publishes it to the `vehicles/{id}/state` topic.
5. The concrete hardware module is subscribed to the state topic and receives a response with the state of the vehicle, whether it is in optimal state or not. In this way vehicle is notified.

This flow represents how `ControllerAddapterMqtt` application adapts the data from MQTT to HTTP in both directions.

Web User Interface. A web server serves static HTML5, JavaScript and CSS resources, which are assembling the web user interface. The web resources do not contain any back-end logic, but only front-end code, which assembles a responsive and user-friendly interface. This functionality for serving of static web resources is lightweight and it's supported by the most cloud platform providers. The concrete web user interface in the EcoLogic is based on the open-source JavaScript front-end web application framework OpenUI5. It makes many simultaneous AJAX (Asynchronous JavaScript and XML) requests to the HTTP REST API endpoint exposed by the Controller application. Public access is provided and the drivers and operators of the system are using it. The web user interface and the Controller application have different origins, because they have different domain names. The problem of the same-origin policy (OpenUI5), which postulates that one web application can access web resources from the same origin or only permitted web resources from another origin is resolved by most of the cloud platforms, usually by tokens.

The web user interface is based on the model-view-controller architecture and supports the following capabilities:

- Handles the whole lifecycle of the users, vehicles, sensors and measurements: supports create, read, update, and delete operations.
- Display all historic and live measured parameters in real-time.
- Static manual control of the state of all vehicles by appropriate user interface controls, which leads to control of the vehicle's actuator. This is achieved with manually configured threshold.
- Automatic dynamic control of the state of the vehicles, which is based on the value of emissions or parameters that are not optimal per region.
- Visualization of all vehicles with not optimal emissions or parameters - anomalies (outliers).

Analytics Application. The Analytics application executes a clustering analysis algorithm on the stored data. The algorithm takes two parameters as input: engine capacity of the vehicles and total mass of carbon dioxide emissions exposed in the atmosphere. In this way it partitions vehicles that have similar engine capacity and emissions amount in clusters and finds the vehicles which are anomalies. The analytics application uses K-Means algorithm for clustering analysis. The number of engine capacities of the registered vehicles represent the number of clusters (K) in the algorithm. The application is connected to the backing service and consumes the stored relational data.

4 Algorithm Results

This section proves the relevancy of the EcoLogic algorithm by presenting a case study with two separate datasets: test dataset and real dataset. The goal of the case study is to test the capability of the algorithm to find outliers in the dataset, which represent vehicles' with not optimal carbon dioxide emissions

in the context of a region with many vehicles. The experiment is carried out with one real vehicle with installed hardware module. The cloud platform and services that are configured for the case study are as follows:

- All described cloud-native applications are deployed into SAP Cloud platform.
- The data is stored in a HANA relational database, provided as a backing service from the cloud platform [7].
- The Analytics application is represented by a SAP cloud platform predictive service, which provides an algorithm for clustering analysis used for anomaly detection.

4.1 Algorithm Results on a Dataset with Known Anomalies

First for validation of the correctness of the anomaly detection algorithm a test dataset, which contains known anomalies is used. The algorithm is executed on the test dataset and the returned result is compared with the known result. An official test dataset is used for that purpose (Mugglestone 2014). It has data for customers with the following parameters: id, name, lifespan, newspend, income and loyalty. The test dataset contains 152 rows. The returned results from execution of the clustering analysis with anomaly detection on the test dataset are shown in Fig. 3.

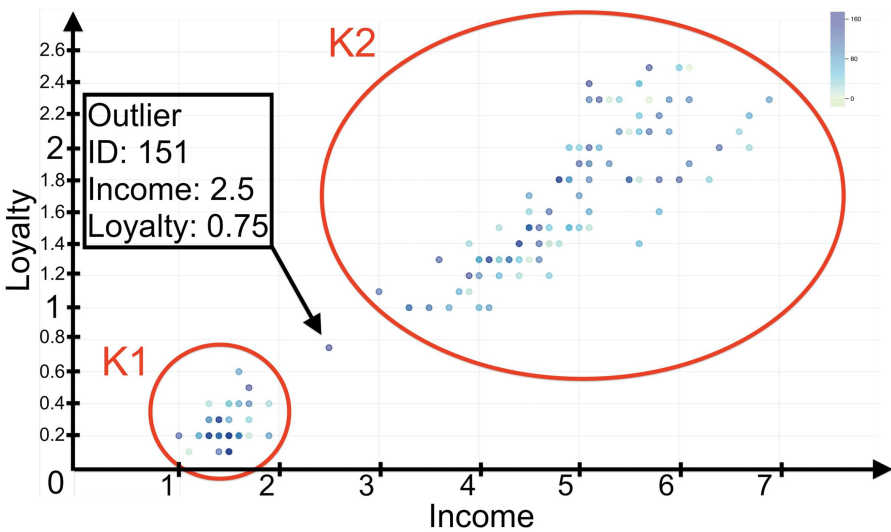


Fig. 3. Clusters of dataset with known anomalies

The x-axis contains the income of the customers. The y-axis contains the loyalty of the customers. The algorithm successfully make two clusters (K1 and K2) and detects one anomaly. The clusters represent two kinds of customers:

customers with low income and low loyalty and customers with high income and high loyalty. The anomaly, marked in Fig. 3 as Outlier, represents a data point, which is located too far from the centers of the clusters K1 and K2, according to other data points. The customer, which is detected as an outlier has ID: 151, income: 2.5 and loyalty: 0.75. The result obtained from the algorithm is the same as the known result from the test dataset, which proves the correctness of the algorithm for this dataset.

4.2 Algorithm Results on a Real Dataset

The hardware module of the EcoLogic is installed into a real vehicle with internal combustion engine with a capacity of 1800 cubic centimetres, working on petrol fuel. In order to acquire bigger dataset, the collected real data is expanded with proportional simulated data. The final operative dataset contains data for vehicles with many engine capacities and carbon emission amounts. The x-axis contains the engine capacity measured in cubic centimetres (cc). The y-axis contains the mass of the carbon dioxide emissions, measured in milligrams (mg). The outcome clusters obtained after execution of the clustering algorithm are shown in Fig. 4. The data points, which have unique IDs, corresponds to the vehicles.

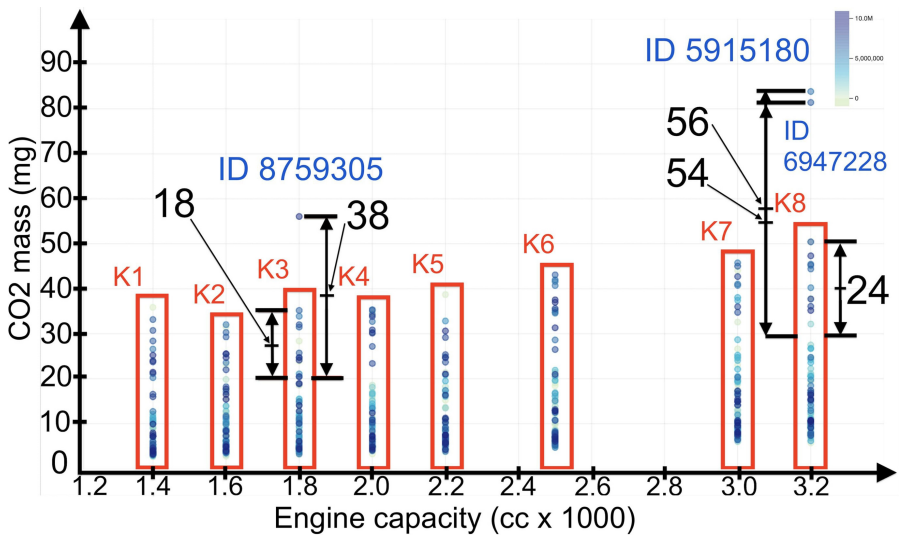


Fig. 4. Clusters of real dataset

The clustering algorithm should place vehicles, which have equal engine capacity and different amount of emissions preferably in the same cluster. In this way, the vehicles with not optimal emissions will not be placed in any cluster and should be detected as outliers. The algorithm returns 8 clusters (K1-K8)

for the current dataset, which represent vehicles grouped in 8 different engine capacities – 1400, 1600, 1800, 2000, 2200, 2500, 3000 and 3200 cubic centimetres respectively. They have different carbon dioxide emissions values, which vary between 2.70 and 50.44 mg. Vehicle with ID 8759305 has engine capacity of 1800 cubic centimetres and emissions of 56.07 mg. The nearest cluster center located to vehicle with ID 8759305 is those of cluster K3, with distance of 38 mg. The maximal internal cluster distance in cluster K3 is 18 mg. Vehicle with ID 6947228 has engine capacity of 3200 cubic centimetres and emissions of 81.42 mg. Vehicle with ID 5915180 has engine capacity of 3200 cubic centimetres and emissions of 83.86 mg. The nearest cluster center located to vehicles with ID 6947228 and 5915180 is those of cluster K8, with distance of 54 and 56 mg respectively. The maximal internal cluster distance in cluster K8 is 24 mg. The vehicles with IDs 8759305, 6947228 and 5915180 are detected as outliers, because the distance from them to their nearest clusters is bigger than the internal cluster distance. These vehicles don't have optimal amount of carbon dioxide emissions in the context of the rest of the vehicles, which are located into clusters K1-K8. Important notice for the outlier with ID 8759305 is that this is the real vehicle with parameter values measured during the idle period on cold engine. The outliers with IDs 6947228 and 5915180 have simulated parameters on the base of the real vehicle's parameters on cold engine. The result obtained from the algorithm is feasible and proves it's correctness on the real dataset. The hardware modules of the detected anomalies are successfully notified and actuators are activated.

5 Conclusion and Future Work

In this paper we presented an IoT platform, called EcoLogic, for real-time monitoring, analysis and control of carbon dioxide emissions exposed into the atmosphere by vehicles with internal combustion engines.

The platform has the following advantages:

- High availability and velocity during work with big amounts of data due to cloud-native architecture.
- Platform-agnostic – possibility to work with different vehicles and clouds. The hardware modules are flexible and easy extendable to work with variety of sensors and/or collect data from the default onboard diagnostic bus or sensors of the vehicles. The implemented cloud-native applications are microservices, which are not vendor locked and can work on different clouds.
- Fully completed and validated solution for monitoring and control of carbon dioxide emissions from vehicles, which is ready to contribute in the fight against climate change and reduction of the carbon dioxide emissions in the atmosphere.

The following routes for further improvements are identified and will be addressed in the latter stages of the project:

- The engine capacity and carbon dioxide emission parameters are not adequate to appropriately regulate the amount of fuel injected. The solution could

be extended to use also other parameters like vehicle weight and performance requirements for better control.

- The measured parameters on cold and hot (with normal working temperature) engine are not deviate between each other. The measurements on a cold engine leads to detection of the outliers. The solution could be optimized to use two separate data sets: data, which is measured on normally working engine and data, which is measured on cold engine.

- Different algorithm for anomaly detection or classification can be used. It can be a supervised machine learning algorithm, which takes the data measured on cold engine as a training dataset, which defines not optimal amount of carbon dioxide emissions.

- Implementation and usage of more lightweight application network protocols such as CoAP, DDS and AMQP.

- Implementation of predictive maintenance algorithm, which makes notifications for potential future failures in vehicles, based on the current and historical data.

- Integration of the project with other third party systems such as emissions trading systems, transportation tax institutions and smart cities. For example, the more eco-friendly drivers could pay smaller taxes in tax systems of countries and city halls. The amount of carbon dioxide emissions could determine the traffic routing by the traffic lights in concrete regions of smart cities.

Acknowledgements. This work was partially supported by the National Science Fund of Bulgaria under Grant KP-06-N32/2-2019.

References

1. Xia, F., Yang, L., Wang, L., Vinel, A.: Internet of things. *Int. J. Commun. Syst.* **25**(9), 1101–1102 (2012)
2. Evans, D.: The Internet of Things. How the Next Evolution of the Internet Is Changing Everything, Cisco IBSG (2011)
3. Carignani, M., Ferrini, S., Petracca, M., Falcitelli, M., Pagano, P.: A Prototype Bridge Between Automotive and the IoT. *IEEE World Forum on Internet of Things*, Milan, Italy (2015)
4. Fehske, A., Fettweis, G., Malmudin, J., Biczok, G.: The global footprint of mobile communications: The ecological and economic perspective. *IEEE Commun. Mag.* **49**(8), 55–62 (2011)
5. Munir, A., Kansakar, P., Khan, S.: IFCIoT: Integrated Fog Cloud IoT: a novel architectural paradigm for the future Internet of Things. *IEEE Consum. Electron. Mag.* **6**(3), 74–82 (2017)
6. Henriques, M., Vernekar, N.: Using symmetric and asymmetric cryptography to secure communication between devices in IoT. In: *International Conference on IoT and Application*, India (2017)
7. Färber, F., May, N., Lehrer, W., Grosse, P., Rause, H., Dees, J.: The SAP HANA database - an architecture overview. *IEEE Data Eng. Bull.* **35**(1), 28–33 (2012)