FACTA UNIVERSITATIS
Series: **Automatic Control and Robotics** Vol. 19, Nº 3, 2020, pp. 163 - 174
https://doi.org/10.22190/FUACR2003163B

IMPROVING INTERNET OF THINGS PARKING SYSTEMS

UDC (004.8:004.451.2)

Dušan Bogićević

University of Niš, Faculty of Electronic Engineering,
Department of Computer Science, Niš, Republic of Serbia

Abstract. *Edge data processing represents the new evolution of the Internet and Cloud computing. Its application to the Internet of Things (IoT) is a step towards faster processing of information from sensors for better performance. In automated systems, we have a large number of sensors, whose information needs to be processed in the shortest possible time and acted upon. The paper describes the possibility of applying Artificial Intelligence on Edge devices using the example of finding a parking space for a vehicle, and directing it based on the segment the vehicle belongs to. Algorithm of Machine Learning is used for vehicle classification, which is based on vehicle dimensions.*

Key words: *Machine learning, IoT, car segment, Arduino, Raspberry Pi*

1. INTRODUCTION

In most cities today there is limited space for parking spaces. In addition, there are more and more vehicles which are of different sizes. Increasing the number of vehicles requires a better optimization of parking spaces. The problem of finding a parking space is one of the problems addressed by IoT researchers. It is necessary to develop an IoT application that will effectively save space (by determining the class of vehicle using its dimension), find the closest parking spot and provide navigation instructions to it. Different sensors may be used to collect parking occupancy information. The data sent from the sensor is raw, and needs to be processed and analyzed. The number of devices connected to the network as well as data processed on the servers is increasing.

Fourth industrial revolution requires increased data rates. The problem which is still present is processing all data coming from the sensors in real time. One of the new trends in the architecture of the Internet is to bring processing closer to the source, that is to the devices where the data is generated. The two architectures that are proposed to process data information closer to the origin are Edge and Fog computing. The most common devices used to process information are gateway devices. It is possible to implement streaming data processing on these devices in order to achieve better information processing.

Received September 25, 2020
Corresponding author: Dušan Bogićević
University of Niš, Faculty of Electronic Engineering, Department of Computer Science, Aleksandra Medvedeva 14, 18000 Niš, Republic of Serbia
E-mail: dusanbogicevic@elfak.rs

One of the common problems of drivers having larger vehicles is that they cannot park in some individual parking spaces, and smaller vehicles take up more space than they need. The difference in the required parking space is shown in Fig. 1.

Fig. 1 The difference in needed space

The auto industry in Europe has accepted the classification of vehicles into multiple segments A, B, C, D, E, F, S, M, J depending on the vehicle dimensions. With the development of the auto industry, there are changes in dimensions, and they overlap in terms of dimensions in some segments. Classification of vehicles in the parking lot can contribute to better organization and use of space. Depending on the segment they belong to, vehicles have greater or less maneuverability that determine the space required for successful parking. The user who would enter a parking lot equipped with vehicle classification and smart routing needs to be provided with information on where to park in the shortest time with the most efficient spare of parking space.

Vehicles could be classified based on dimensions, but it would lead to a problem because it is not possible to clearly draw a line between segments. Vehicle segments in Europe do not have formal characterization or regulations. Models segments tend to be based on comparison of well-known brand models. For example, a car such as the Volkswagen Golf might be described as being in the Ford Focus size class, or vice versa. The VW Polo is smaller, so it belongs to the segment below the Golf, while the bigger Passat is one segment above [1].

Determining which segment the vehicle belongs to is made on the basis of all dimensions, and only then it should be concluded which segment it is. IoT collects the data and the AI processes the data to give them meaning [2]. In order to keep the previous models and their belonging to the segments from the period when they were created, as well as the new models and their belonging to the current segments, we need to use of Machine Learning mechanisms and approach.

According to these observations, a suitable architecture for processing streaming data close to the source of where it is created is analyzed. An attempt was made to find a fitness ML algorithm which will get the best results for classification of a car by segments. In this research both real sensors and simulated ones were used.

The paper is organized as follows. In section 2, is presented architecture of IoT systems with technologies which will be used in research, also in this paper a literature survey in this domain is made. Section 3 presents the proposed solution for improving the parking system. The next section shows Implementation of the proposed solution with results based on data from sensors, processed using ML algorithms. Concluding remarks are listed in the final section.

2. BACKGROUND AND RELATED WORK

2.1. Architecture of IoT

IoT devices are spread around the edge of the network, where data generation is present. This data requires real-time processing without delay. Sending data to servers remotely located is not the best solution for systems with these requirements. Bringing the server to the edge of the network was first proposed by Cisco in 2012 and was called Fog computing [8]. As devices are now more capable of performing complex computer processing and data analysis, certain tasks can be performed on the devices themselves without sending data and request to the server. This kind of architecture where processing is done on Edge devices (smart phone, smart object ...) is called Edge computing.

Edge devices can serve other devices with weaker computing and communication capabilities to connect to them and communicate with servers through them. A device that would perform this functionality would be the Gateway. A gateway device could solve the connectivity problem, e.g. if wireless technologies such as Bluetooth or Zigbee on the one hand were to be used, the gateway on the other would provide communication with servers (services, IoT platforms, Middleware ...) and other devices located on the other network. Differences in the Cloud, Fog and Edge computing architectures are shown in Fig. 2 [8].

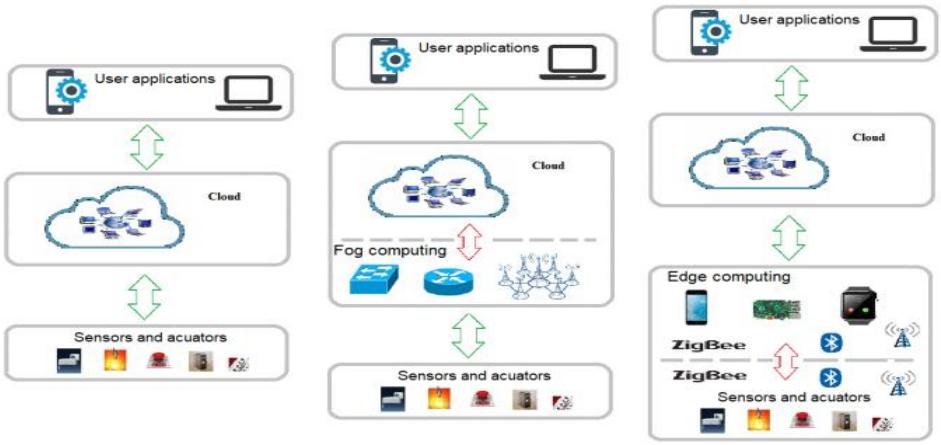

Fig. 2 Moving from Cloud computing over Fog to Edge computing [8]

Edge sensors and actuators are specific devices that most often do not have an OS, but they are connected directly or through radio networks to the Edge devices or gateways [9]. The goal to be achieved is that all IoT devices can use Edge to integrate into the Cloud and uniformly utilize all the resources available. In addition to the uniform use of available resources, it is also possible to:

- ensure privacy: Data released from the device may be confidential, so it is better to process that data at the point where it originates (closest to the data source), and only those data that is security compliant (not security sensitive) are sent to the server for processing.
- delay reduction: Machine learning algorithms that work directly on IoT devices can be implemented on Edge.

- connectivity: IoT devices can be equipped with different communication interfaces, and by using Edge that has multiple communication interfaces, it is possible to extend the communication capabilities of all devices.
- filtering: It is also possible to filter the data obtained from the sensors.
- pre-processing: It is possible to perform data processing and preparation for sending.

2.2. Streaming analytics

Streaming analytics is an approach to processing data, in which hidden information is extracted from the data before it is sent to the Cloud. Small analytical algorithms can be applied to this data to better utilize network resources [10]. They work on: router, switch, gateway, server [11].

In IoT systems, there are a large number of events that need to be processed and make an appropriate decision based on the analysis of the data obtained from those events. Processing such data must be faster than their generation as events. Data analysis aims to find events in the flow of data that is coming. Some of the features that streaming analytics should provide are: Preprocessing, Alerting, Windowing, Joins, Errors, Database, Temporal events and patterns, Tracking, Trends and etc. [4].

The features expected from data analysis and processing on the Edge [12] [13] are:

- to reduce the cost of sending data
- to reduce the amount of data being stored
- to determine when the data should be sent further for processing using analytical model
- not to send data continuously, but at appropriate moments when certain conditions are met
- to ensure data authorization and security
- to provide predictive analytics

Data processing closer to the source allows reduction of network load and delay. The processing can be done on the devices themselves if they have the capability for analytical processing, or on the closest device that can perform analytical processing of the received data. The architecture of streaming data processing closer to the source is shown in Figure 3. Companies such as Cisco, Intel, Dell propose their gateways as Edge data processing devices [14]. The Edge side usually has an agent, which is configurable from a server (e.g. IoT platform) [15].

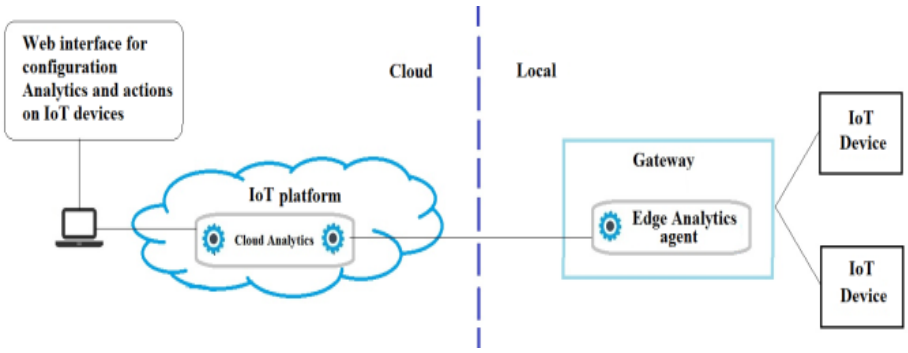

Fig. 3 Edge streaming analytics architecture

2.3. Apache Edgent

APACHE Edgent (former Quarks) [16] [17] is an open source Edge analytics model for programming and execution on Edge devices that enables real-time analysis of data flows on Edge devices or gateways. It is an open source project that represents a tool for environment programming (lightweight runtime environment) for analytical data streaming on Edge [18].

Use of this model achieves data analysis that starts from the end devices and goes to the central parts. It is possible to send alerts to end devices from the central unit. It can be run on devices such as the Raspberry PI (requires devices with a minimum of 512MB of memory). It is made in Java but can be extended to other programming languages like Swift, Scilab, Python.

Edgent accelerates your development of applications to push data analytics and machine learning to Edge devices. (Edge devices include things like routers, gateways, machines, equipment, sensors, appliances, or vehicles that are connected to a network.) Edgent applications process data locally—such as, in a car, on an Android phone, or on a Raspberry Pi—before it sends data over a network.

2.4. Smart parking: Related works

The problem of finding a parking space today is one of the problems that is common in most cities. This issue is being addressed by a number of researchers exploring the application of the IoT concept. The solution to the problem of occupying a parking space can be grouped into one of four categories, namely: counting based, wired based, wireless based and image processing based [19].

Sarker at al. [20] propose an architecture for parking with gateway and enable pre-processing, compression, encryption, dynamic pricing on edge gateway device. They proposed IR sensor, ultra-sonic rangefinder and magnetometer sensors to prove accurate information in any weather conditions. The sensors send data via radio networks like BLE, Nordic Semiconductor's nRF end etc. All sensors send data to Raspberry Pi3 B+ edge gateway device. Raspberry Pi3 B+ has an extra module to send data over LoRa gateway.

Misra et al. [21] designed solution for management of parking space. The focus of this research was checking is the vehicle parked correctly and monitoring a number of vacant spots. They used NodeMCU ESP8266 with ultrasonic sensor (HC-SR04) and IR sensor (FC-51). In case that the car is obstructing the space of another vehicle, the system will produce an alert to inform the gatekeeper or owner to take some action. All information about occupation are stored in the database and send to a central server.

Khanna and Anand [22] presented a solution for booking a parking slot. Parking place in this case is equipped with ESP8266 modules. Each module can be connected to sensors like IR, PIR and Ultrasonic Sensors. The modules are sending data over a WiFI network to a Raspberry Pi3 gateway which communicate over the MQTT protocol with the IBM MQTT server. This server is a public server, and enable mobile application to exchange data, and booking a parking slot.

Thangam et al. [23] used Raspberry Pi with camera in Facial Recognition and recognition of car license plate in their parking systems. They developed a mobile application for booking parking slot which enable to parking slot be saved for driver who booked it. In case that a driver's face and license plate are detected, then it has enabled the driver to park a car. They used an OpenCV library for creating data sets.

There are researchers like Fedchenkov et al. [24] who do research on implementing AI to IoT Parking Solution. They used AI to forecast the availability of parking slots. Using data set with information about parking occupation and timestamp they created trained neural networks with root-mean-square error of 3.5%. Such a trained network is used to give drivers information where is the best parking place. This information is based on distance from the parking place and prediction system of occupying. Researchers used the Keras library which provides tools for building and learning neural networks, and the software product TensorFlow as a backend.

3. PROPOSED SOLUTION OF THE IMPROVED PARKING SYSTEM

Most smart parkings focus on the number of vacancies, as well as on the problem of booking them. This paper demonstrates the possibility of deploying Machine learning in IoT systems by using data streaming processing closer to the data source. The paper describes a possible solution for entering and arranging vehicles in a parking lot with a large number of sensors used to identify the occupancy of the parking space.

Data is collected from sensors at the entrance and from sensors at each parking space. The sensors located at the entrance are appointed as in Figure 4. Based on these data, the length (A), width (C) and height (D) of the vehicle are obtained.

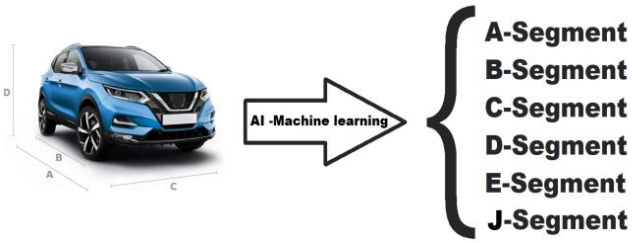

Fig. 4 Parameters used for classification

By using these dimensions, classification is made and the vehicle is directed to the appropriate parking lot. Some of the ways to check parking space occupancy are described in chapter two of this paper. The focus of the paper is not on processing the information from the sensors that are in parking places.

Vehicles can be identified by image processing. License plates can be obtained by image analysis. Based on the picture, it would also be possible to identify the vehicle by recognizing the manufacturer and the model of the vehicle. There are libraries that can be used to identify vehicle license plates such as OpenALPR.

The solution proposed in this paper is based on the processing data from sensors located at the entrance of the parking lot. Sensor data is sent when the read value is different from the initial value (initial values are l, d, h as shown in Figure 4). Information about the length, width, height of the vehicle is derived from the formulas C = d-sl-sr, A = l-lf-lr, D = h-ht. On the basis of these data, the classification of vehicles is made as in Figure 5.

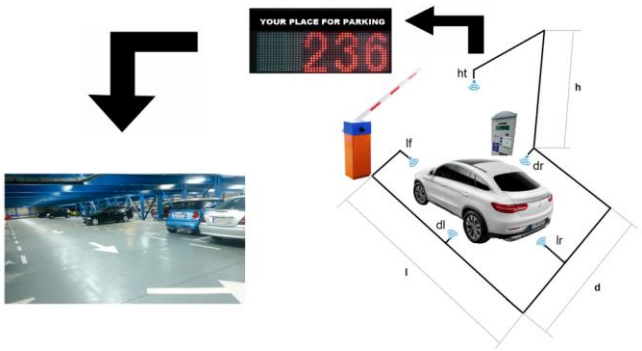

Fig. 5 Determining the vehicle segment and directing it to packing

All data obtained from the parking lot is sent to the local gateway, which performs their processing. The gateway processes streaming data from the sensors to detect events of interest and further processing. After the event is detected, the vehicle is classified into one of the official European segments (in this example, a shortened list of vehicle segments will be used). When the segment the vehicle belongs to is recognized, the place where the vehicle should be parked is shown. An illustrated flow of information is shown in Figure 6.

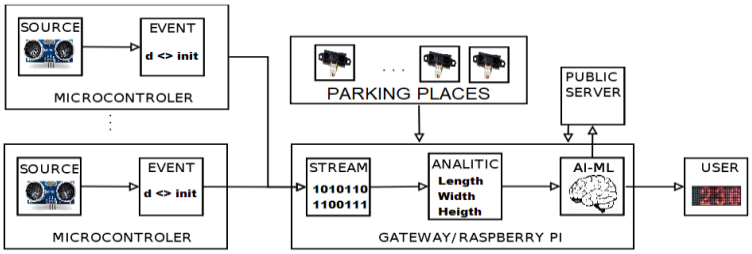

Fig. 6 Data flow

This processing should be fast in order to immediately inform the driver of which parking space to go to. In order to achieve the highest possible speed, information processing is done as close as possible to the source. The necessary information is sent to the server through which the occupancy information is available to the users. In this example, a shortened list of vehicle segments will be used.

In proposed solution we create system which knows how much space it has and to be aware what kind of car is coming. According to that information we want to route the car to suitable parking slot. In related works we can see that the main focus for the most research was to find parking slot and booking it. Some authors use approaches like AI, EDGE computing, optimization of parking slots. Comparison of related works with proposed solution is shown in Table 1 and it is made on the following criteria:

- Occupancy - does the system have information about each parking slot?
- AI - does the system use AI?
- EDGE architecture - is the system organized as EDGE computing architecture?
- Optimization of parking slots - does the system have any optimization mechanism for parking space?

Table 1 Comparison of proposed solution with relation works

	Occupancy	EDGE	AI	Optimization
Sarker at al. [20]	+	+	-	-
Misra at al. [21]	+	-	-	+
Khanna at al. [22]	+	+	-	-
Thangam at al. [23]	+	+	+	-
Fedchenkov at al. [24]	+	+	+	-
Our proposed solution	+	+	+	+

Form Table 1 we can make some conclusions. Every paper takes care about occupancy, and there is a tendency to EDGE architecture. There are also AIs, but with lower frequency. Optimization is supported only in one paper but in that paper there is no EDGE architecture and there is no usage of AI. The solution proposed by this paper connects EDGE architecture, AI and optimization of parking space, in what is step forward from related papers.

The research is divided into two phases. The first phase involved preparing data and selection of a convenient machine learning algorithm. During this phase, one-dimensional (length), two-dimensional (length and width) and three-dimensional (length, width and height) classification was attempted. In the second phase the selected algorithm was used to process the data from a real sensor.

4. Implementation of the Proposed Solution

To train machine learning model, we used data set with 154, 1867, 3313 car models that were classified into 7 segments. As the segments to which the vehicle may belong are defined, we have chosen a group of supervised machine learning algorithms. Available information for training are: the segment to which the vehicle belongs, the length, width and height of the vehicle. The dimension range by each segment is shown in the Table 2, and the overlapping of the dimensions is graphically presented in Figure 7.

Table 2 Maximum and minimum dimension by each segment

		A	B	C	D	E	F	J
LENGTH	MAX	3828	4549	4702	504	5113	5462	5205
	MIN	2500	3575	4025	4262	4490	4984	3962
WIDTH	MAX	1720	1850	2033	1877	1911	1949	2030
	MIN	1475	1610	1680	1726	1707	1855	1658
HEIGHT	MAX	1700	1678	1667	1526	1559	1501	1983
	MIN	1455	1397	1350	1369	1365	1398	1460

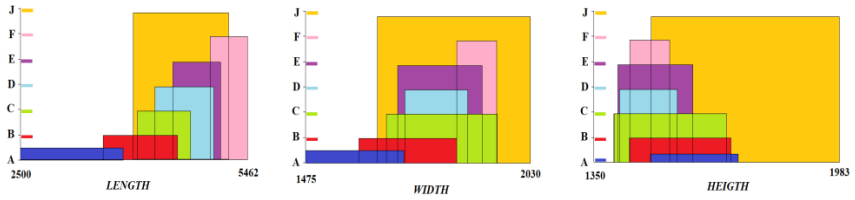

Fig. 7 Overlapping of the dimensions

A value for length can occur in 3 segments, and there is no segment that has a unique value range. For width we have an overlap which for some values can be 5 segments. Height can appear in all 7 observed segments.

For data analysis, we opted for Supervised learning algorithms. Collected data were adapted to machine learning algorithms, and corresponding datasets were formed. As Supervised algorithms were used, training of each algorithm was performed. The next step was testing. The results are presented in the Table 3.

Table 3 Results of tested algorithms

Algorithms	dimensional	Training set	Training set	Measurement error (9335 items)		
				max-10mm	max-20mm	max-30mm
Decision tree (C 4.5)	one	154	82.47% (127)	65.20% (6086)	63.89% (5964)	62.67% (5850)
		1867	92.29% (1723)	66.76% (6232)	60.62% (5659)	57.99% (5413)
		3313	78.06% (2586)	58.18% (5431)	54.59% (5096)	52.60% (4910)
	two	154	92.86% (143)	72.84% (6800)	71.37% (6662)	69.53% (6491)
		1867	97.64% (1823)	80.44% (7509)	73.44% (6884)	68.58% (6402)
		3313	92.85% (3076)	70.36% (6568)	62.28% (5814)	58.01% (5415)
	three	154	96.10% (148)	80.63% (7527)	79.29% (7402)	77.80% (7263)
		1867	98.87% (1846)	90.12% (8413)	85.88% (8017)	81.18% (7578)
		3313	97.01% (3214)	83.74% (7817)	79.96% (7464)	76.69% (7159)
Random forests	one	154	100% (100)	63.01% (5882)	60.47% (5645)	58.54% (5465)
		1867	95.45% (1782)	62.67% (5850)	58.31% (5443)	56.10% (5237)
		3313	83.55% (2768)	53.26% (4972)	50.17% (4684)	47.88% (4470)
	two	154	100% (100)	73.94% (6902)	72.43% (6761)	70.83% (6612)
		1867	99.68% (1861)	83.32% (7778)	75.17% (7017)	70.18% (6551)
		3313	99.18% (3286)	73.37% (6849)	64.30% (6002)	59.54% (5558)
	three	154	100% (100)	81.06% (7567)	80.06% (7474)	79.19% (7392)
		1867	99.89% (1865)	94.57% (8829)	89.30% (8337)	85.26% (7959)
		3313	99.64% (3301)	90.11% (8412)	83.89% (7831)	80.45% (7510)
Logistic regression	one	154	74.02% (114)	57.76% (5392)	57.87% (5402)	57.68% (5384)
		1867	65.72% (1227)	66.26% (6185)	66.36% (6195)	66.35% (6194)
		3313	83.55% (2768)	66.26% (6185)	50.18% (4684)	47.88% (4470)
	two	154	85.06% (131)	71.18% (6645)	69.94% (6529)	67.84% (6333)
		1867	75.95% (1418)	75.85% (7081)	74.80% (6983)	73.66% (6876)
		3313	63.00% (2087)	63.18% (5898)	62.38% (5823)	61.38% (5730)
	three	154	92.86% (143)	78.39% (7318)	77.40% (7225)	76.48% (7139)
		1867	86.12% (1608)	85.35% (7968)	84.27% (7867)	82.41% (7693)
		3313	75.28% (2494)	77.41% (7226)	77.27% (7213)	76.09% (7103)
Mlp	one	154	74.68% (115)	59.88% (5590)	59.80% (5582)	59.53% (5557)
		1867	68.24% (1274)	68.20% (6366)	68.40% (6385)	68.23% (6369)
		3313	55.24% (1830)	65.80% (6142)	65.11% (6078)	64.90% (6059)
	two	154	85.71% (132)	72.96% (6810)	72.06% (6727)	70.37% (6569)
		1867	78.52% (1466)	77.25% (7211)	75.00% (7001)	72.89% (6804)
		3313	63.08% (2090)	64.72% (6042)	63.52% (5929)	62.69% (5852)
	three	154	92.21 % (142)	74.52% (6956)	74.72% (6975)	73.76% (6885)
		1867	87.47% (1633)	86.71% (8094)	84.50% (7888)	82.22% (7675)
		3313	76.76% (2543)	81.30% (7589)	79.25% (7398)	76.89% (7178)

The second phase was providing streaming data processing and spotting a complex event. The problem to be solved was how to obtain the dimensions representing the dimensions of the vehicle from the flow of data from all the sensors located at the

entrance. Apache Edgent was used to process streaming data and detect complex events. Data came from an Arduino Nano board to which a HC-SR04 sensor for distance measuring was attached.

5. TESTING AND EVALUATION OF THE PROPOSED SYSTEM

For research purposes, an application has been developed that simulates data from sensors used to determine length and width. Height information is obtained from the physical device. Data from the physical sensor is sent to the application that simulates other sensors. Simulated data is sent via WiFi to another application located on RPi. For the sake of simplicity, the data from the sensor goes to the mid-application in the example. In real use, the data from the sensor would go directly to the gateway. The layout of the application from which the real and simulated data is sent is shown in Figure 8.

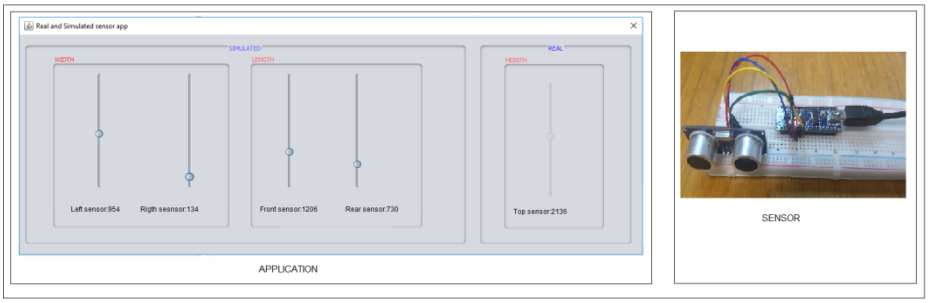

Fig. 8 Application for generating sensors data and pass real data

Data from all sensors arrive at the gateway where the processing takes place. Streaming data processing gives information about the length, height and width of the vehicle. The gateway application determines to which segment the vehicle belongs, and directs the vehicle to the appropriate parking lot. The layout of the application is shown in Figure 9. The application runs in the background and graphically displays the data coming from the sensors and the result of their processing. When the vehicle in question is concluded based on the dimensions, data is sent to the lower devices. The lower device shows the segment to which the vehicle belongs, and where the vehicle should be parked.

Fig. 9 Gateway application for streaming processing data, calculating segment of vehicle and device for informing user

In the research, a classic programming technique was used based on data from Table 2 and Figure 7. The result of accuracy when using the classic programming goes from 65% to 73% depending on the measurement error. Soft computing gets better results, compared with the result from the classic programming. In both techniques there is a probability of mistake.

6. CONCLUSIONS

Processing data in real-time is one of basic requirements of IoT. The one solution is to use Edge computing, which is described in the paper. Using Process engine and analytics on Edge devices such as gateways can reduce server latency and processing power. The cost of the entire system is reduced. An architecture suitable for processing large amounts of data and processing in approximately real time is achieved. By using RPi as a gateway device, we can increase the connectivity reflected in connecting multiple types of networks. By adding artificial intelligence, these devices become aware of received data and can learn from them. By further improving the training set, it is possible to increase accuracy.

Further research could go towards the use of genetic algorithms. By applying genetic algorithms, it is possible to increase the utilization rate of parking lots and improve their organization. Also, another direction for further research could be applying this approach to highway tolling.

REFERENCES

[1] Vehicle size class. [Online]. Available: https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Vehicle_size_class.html [Accessed on June 2020].

[2] Why the Internet of Things needs Artificial Intelligence. [Online]. Available: https://www.iot-now.com/2019/03/12/93908-internet-things-needs-artificial-intelligence/, [Accessed on June 2020].

[3] Ryszard Tadeusiewicz, Rituparna Chaki, Nabendu Chaki, "Exploring Neural Networks with C# ", *CRC Press*,US,2015.

[4] Perry Lea, "Internet of Things for Architects", *Packt Publishing Ltd*., Birmingham, UK, 2018.

[5] Nishith Pathak, Anurag Bhandari, "IoT, AI, and Blockchain for .NET", *Apress*,2018.

[6] Aurélien Géron, "Hands-On Machine Learning with Scikit-Learn & TensorFlow", *O'Reilly*, US,2017.

[7] Stuart J. Russell and Peter Norvig, "Artificial Intelligence A Modern Approach Third Edition", *Pearson Education*, US, 2010.

[8] Redowan Mahmud, Ramamohanarao Kotagiri, Rajkumar Buyya, "Internet of Everything", *Springer*, pp.103-130, 2017.

[9] Roy Want, Bill N. Schilit, and Scott Jenson, "Enabling the Internet of Things", *IEEE*, 2015.

[10] Foteini Beligianni, Miltiadis Alamaniotis, Athanasios Fevgas, Panagiota Tsompanopoulou, Panayiotis Bozanis, Lefteri H. Tsoukalas, "AN INTERNET OF THINGS ARCHITECTURE FOR PRESERVING PRIVACY OF ENERGY CONSUMPTION", *IET*, 2016

[11] Rajkumar Buyyaa, Amir Vahid Dastjerdi, "Internet of Things Principles and Paradigms", *Elsevier*, 2016.

[12] Apache Edgent Overview, [Online]. Available: https://Edgent.apache.org/docs/home [Accessed on June 2020].

[13] Julian Ereth, Edge Analytics in the Internet of Things, [Online]. Available: https://www.eckerson.com/articles/Edge-analytics-in-the-internet-of-things [Accessed on June 2020].

[14] Edge Analytics – The Pros and Cons of Immediate, Local Insight, [Online]. Available: https://www.talend.com/resources/Edge-analytics-pros-cons-immediate-local-insight/ [Accessed on June 2020].

[15] Getting started with Edge Analytics in Watson IoT Platform, [Online]. Available: https://developer.ibm.com/recipes/tutorials/getting-started-with-Edge-analytics-in-watson-iot-platform/ [Accessed on June 2020].

[16] Getting started with Apache Edgent, [Online]. Available: https://Edgent.apache.org/docs/old-Edgent-getting-started [Accessed on June 2020].

[17] Quarks Renaming Discussion - Now Apache Edgent, [Online]. Available: https://cwiki.apache.org/confluence/display/EDGENT/Quarks+Renaming+Discussion+-+Now+Apache+Edgent [Accessed on June 2020].
[18] Apache Edgent Overview, [Online]. Available: https://Edgent.apache.org/docs/home [Accessed on June 2020].
[19] Luis Ostiz Urdiain, Carlos Pita Romero, Jeroen Doggen, Tim Dams, Patrick Van Houtven, "Wireless Sensor Network Protocol for Smart Parking Application Experimental Study on the Arduino Platform", *The Second International Conference on Ambient Computing, Applications, Services and Technologie*, AMBIENT 2012.
[20] Victor Kathan Sarker, Tuan Nguyen Gia, Imed Ben Dhaou, Tomi Westerlund, "Smart Parking System with Dynamic Pricing, Edge-Cloud Computing and LoRa", *Sensors*, 2020.
[21] Rohit Misra, Shekhar Jain, Mihir Bonde, Harish Motekar, "IoT Enabled Smart Parking System", *International Research Journal of Engineering and Technology*, 2019.
[22] Abhirup Khanna, Rishi Anand, "IoT based Smart Parking System", *International Conference on Internet of Things and Applications*, Pune, India, 2016.
[23] Cassin Thangam, M. Mohan, J. Ganesh, C.V. Sukesh, "Internet of Things (IoT) based Smart Parking Reservation System using Raspberry-pi", *International Journal of Applied Engineering*,2018.
[24] Petr Fedchenkov, Theodoros Anagnostopoulos,Arkady Zaslavsk, Klimis Ntalianis, Inna Sosunova and Oleg Sadov, "An Arti fi cial Intelligence Based Forecasting in Smart Parking with IoT", *Springer Nature Switzerland*, 2018.
[25] Eben Upton and Gareth Halfacree, “Raspberry Pi User Guide, Wiley”, United Kingdom, 2016.
[26] RPiSpec, [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/ [Accessed on June 2020].
[27] The Arduino Playground , [Online]. Available: https://playground.arduino.cc/ [Accessed on June 2020].