Appeared in Proceedings of Thirteenth International Conference on Digital Information Management (ICDIM 2018), pp. 18-23, 2018, Berlin, Germany, ISBN 978-1-5386-5244-2

Smart parking system - another way of sharing economy provided by private institutions

Simona-Daniela Marcu, Adrian Florea
Computer Science and Electrical Engineering Department
"Lucian Blaga" University Sibiu
Sibiu, Romania
Email: marcu.simona95@yahoo.com, adrian.florea@ulbsibiu.ro

***Abstract*— This paper presents our smart parking solution implemented at "Lucian Blaga" University of Sibiu (LBUS) Romania, which consists in a hardware / software embedded system for managing the institution's parking lots, namely sharing the parking places in excess for people who are in traffic in neighbourhood of LBUS and are looking to park. Our solution is flexible, universal, applicable to all faculties that have car parks from Romania and not only, in university cities where the crowd is bigger, but also to other private institutions that own parking spaces inefficiently exploited. The advantages introduced are primarily economic, then social and even environmental. The first benefit is economic - for institutions which exploit their free parking spaces during the year, at different moments of time (holiday, afternoon, or time when people left from work) offering for people who are searching for. From a social point of view, ensuring the convenience of drivers, reducing crowding especially at rush hours and reducing the time spent in search of a parking space, is also an important advantage. Reducing the cost of fuel required by cars that are looking for a parking space contributes to reducing pollution and improving air quality, having a positive impact on the environment.**

Keywords - parking spaces; embedded system; MQTT; web application; smart parking

I. INTRODUCTION

An important goal of a smart city is to find intelligent solutions that help improve the quality of services provided to citizens. The lack of parking spaces and other infrastructure problems are the consequence of city crowding [1]. Moreover, in Sibiu, during the annual 25-year-old theatre festival or other events, the authorities restrict parking places, offering parking slots for guests from abroad or in the country. This year the number of visitors and participants to theatre festival was 525,000 people, around 3 times more than the Sibiu city's population that creates an additional pressure on local drivers. Looking for a parking space has become a routine for most drivers around the world. Some statistics from Boston University [2] shows that more than 30% of the drivers spent about 8 minutes surrounding the areas near their target to park their vehicle. Our experience, into a small town like Sibiu, Romania, reveals that finding a parking space (e.g. around the Engineering Faculty) involved every morning losing at least 10 minutes of searching, as well as fuel costs, or coming to Faculty with at least 20 minutes before the class starts. A smart parking system would be the first step in the right direction. This helps to reduce the time with searching a parking spot, reduce pollution and helps to fluidizing traffic, and even reduces the risk of fines for irregular parking.

In these days, the Internet and mobile devices can be used by any person. Thus, implementation of a web application is the best way to achieve this goal. A web application has many advantages as portability, compatibility between different platforms and accessibility. All this advantages made us to reduce the problems mentioned above, using a web application implemented in ASP .NET and C#.

This solution was chosen due to the fact that finding a parking space represents one of the biggest problems of a driver. An urban smart mobility solution for congestion control might be the use of parking charges, congestion time pricing for downtown parking areas. However, this would be apply by local public administration. Our proposal is dedicated for private institutions who may share their parking spaces with every driver at certain moments of time, without affecting their own employees.

In our previous work [3] we forecasted the "the occupancy rate of a building" based on embedded systems that collected data from sensors and, using artificial intelligence tools we have modelled the energy consumption in buildings based on alternative data sources, such as the number of vehicles in a parking lot. It was determined the correlation between the number of vehicles over time and time-related parameters, like day of week, hour of day, and type of day, in order to validate the potential of using a machine learning model of parking lot occupancy. This paper presents our smart parking solution implemented at LBUS. During one year, this parking space is occupied around 40 weeks, the rest being holiday. There are around 120 parking spaces. The institution time scheduler shows that after 17 o'clock are much less classes than during the morning. In conclusion, there is a rather high degree of vacancy in the parking which could generate additional income for private institutions.

The proposed system consists of the hardware system (camera, Raspberry Pi) for license plate recognition number, the opening / closing the barrier, three led that form the traffic light, and software application for determining the free places, requesting (booking) a parking space for a time interval through a web application, the communication module between Raspberry Pi microprocessor and the WEB application being made through the Node-RED program that

use the MQTT services to transmit the real-time information to the application without the need to refresh the webpage.

The remainder of paper is organized as follows: The second part analyses the related works. The introduction of our solution with its advantages follows in the third part. The fourth part gives us the application architecture whilst the tools and programming environments used are described in fifth section. Some scalability issues and a possible solution provided by our embedded system are tackled in sixth section. Finally, conclusions and future work ideas are presented in the last section.

II. Related Work

In [4] the authors presented a reservation-based parking system used at Nebraska University, Lincoln, USA. The application is managing parking spaces and monitor real-time the parking spaces. The proposed solution is a reservation-based system, so the driver reserves the most convenient parking space for him, depending on the walking distance to the destination and the cost of the car park. This paper further elaborates on one of the criteria for choosing the parking space by the driver, namely the cost. The disadvantage of this work is that, when more users apply for a reservation in the same time, occur synchronization problems, and there is a "bottleneck" within the system and increases the update time because the system have to handle serial each requests. In our work we solve this problem at the session level of the OSI reference model [5] of communications protocols. We are using a different name for every session, and the messages are received and send only on that session.

In [6] the authors describe an embedded system design for a real-time parking. This work, implemented at University of South Florida, has the main goal to inform the drivers, using a mobile application, about availability of parking spaces in real-time. An ultrasonic sensor is used to detect free parking spaces. When a car is detected, reads the ultrasonic sensor, and updates the counting of cars both on the display at the entrance to the parking lot and on the mobile application. The disadvantage is that the sensor detects golf carts or people as cars. This is not a problem to our solution because the entrance to the parking is based on license plate number.

In [7] the authors present a smart parking system based on embedded system implemented in Pune, India. The solution consists of a smart parking system based on reservation, QR code generation technology and a webcam that reads at parking entrance the QR code and checks in database if it is the one expected. Our hardware components are newer (Raspberry Pi model 3B) than their solution and we use license plate recognition algorithm instead QR technology.

III. Solution

A. Reservation

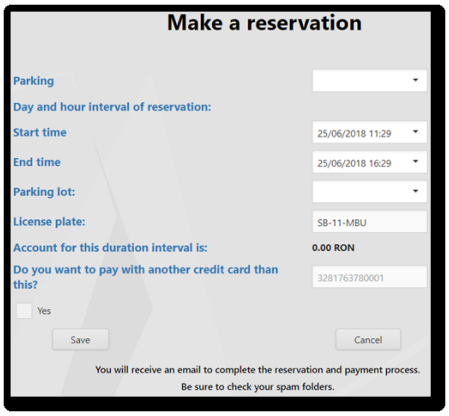

Fig. 1. Reservation form

This solution consists of a smart parking system based on reservations. Anyone who is registered on application can make a reservation. Reservation contains the name of parking, parking space (can be choose by user), price, arrival and departure estimated date and time.

B. Real time parking status view

Anyone can view in real time the parking status, how many spaces are free, how many are busy and by which car (the license plate number is placed on parking space chosen from reservation page). The busy spaces are represented with red color, and free spaces are represented with green color.

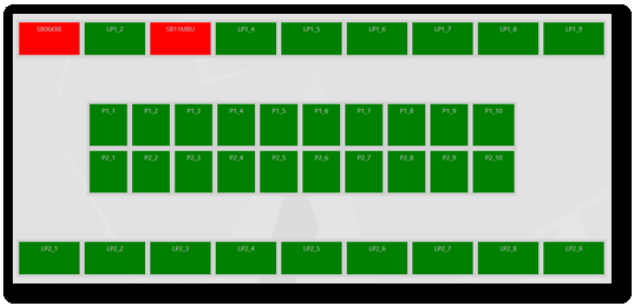

Fig. 2. Real time status parking view

C. License plate recognition

The entrance to the parking is based on license plate number. When driver arrives in the front of the barrier, he has to push the button from home page to send instruction to camera for checking his license plate number. If the drivers have a reservation that is +/- 15 minutes around estimated arrived date and time he has access into the parking. If not, we have to check if exist empty spaces, that are not reserved in the next 4 hours. So, if there are still empty spaces, that car has access in the parking, otherwise not.

IV. Application Architecture

The purpose of this project is to create a hardware-software application that resolve the parking problem described above: booking a place and allowing the entrance

and of course the reverse problem – exiting and updating the number of free places.

The database is essential. We designed a MySQL database which contains few tables with foreign keys. We keep in database all reservations, user information, prices, history and the number of free parking spaces.

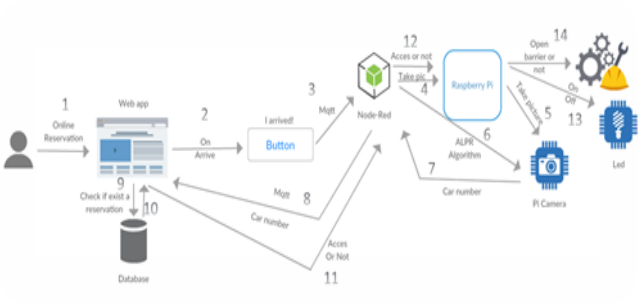

Fig. 3. Application architecture

In fig. 3 are presented all steps (14) of operations' flow of the project in order to allow a car to enter into a parking space. The first step is making an online reservation. When the driver arrives in front of the barrier, he has to push the *<I arrive>* button from home page. After all this, the Node-Red is sending using MQTT, messages and instructions from/to application.

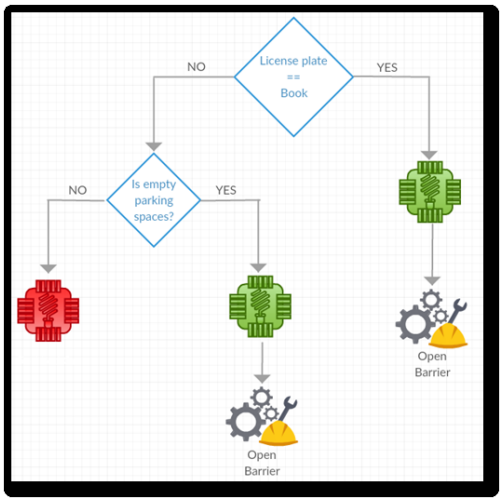

Fig. 4. Checking access

In fig.4 is present the checking access in parking. So, if the car witch arrives in the front of barrier has a valid reservation, the green led is turning on, and the barrier opens. Otherwise, we check for the empty spaces with the following formula. Difference between final number of spaces for parking available, busy spaces at that time, reservations for the next 4 hours and 2 spaces used for safety. Applying this formula, if there are still empty spaces, led green is on and the barrier opens, otherwise the red led is turning on.

In this project are used some new tools for IOT programming. The application flow is the following:

A. Online Reservation

The first step is making an online reservation from web application. For this, we have to register into the application and access the Reservation page.

B. Arrival Home button

When driver arrives in the front of barrier, he has to push the arrival button from Home page. This button sends the instruction to camera to take a picture, get license plate number from that and check in database if exist a reservation with that license plate number.

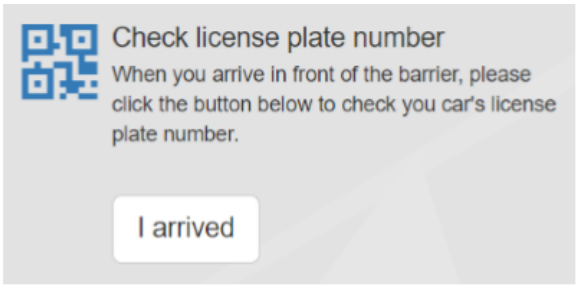

Fig. 5. The arrival button

C. Barrier and traffic lights

After that license plate number is checked, web application sends instructions to turn on / off the lights and open the barrier or not.

While all this checks are made, yellow light is on. At the end, if that car has access in the parking, green light is turning on, and the barrier opens. Otherwise, red light is turning on, and the barrier does not open.

V. TOOLS AND PROGRAMMING ENVIRONMENTS USED

This project is divided into two parts. First part contains hardware components used, while the second contains software programming environment used.

A. Hardware components

The main component used is a Raspberry Pi model 3B microcontroller witch has a Linux Raspbian operating system. Together with it, we used the following compatible components:

1) Pi camera: Is a small camera that has 5MP. We use a cost-effective camera with reasonable performance (around 90%) taking into account that some commercial and professional camera cost around 10000 dollars but with accuracies between 93-95% [2].

2) Leds: Together form the traffic light.

a) Yellow: Waiting

For turning on / off the leds we used some Python scripts which are executed as follow:

$sudo python xLedOn.py
$sudo python xLedOff.py

Where xLedOn.py / xLedOff.py represents the Python file that contains the script. We made 2 files for each led (1 for turning on and another for turning off the led). So, there are totally 6 python files to act the leds.

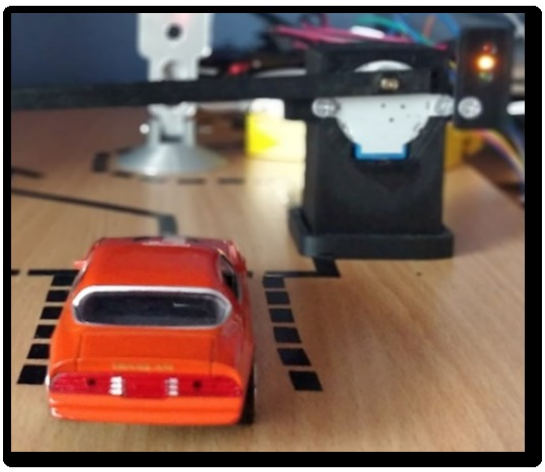

Fig. 6. Waiting

b) Green: Access

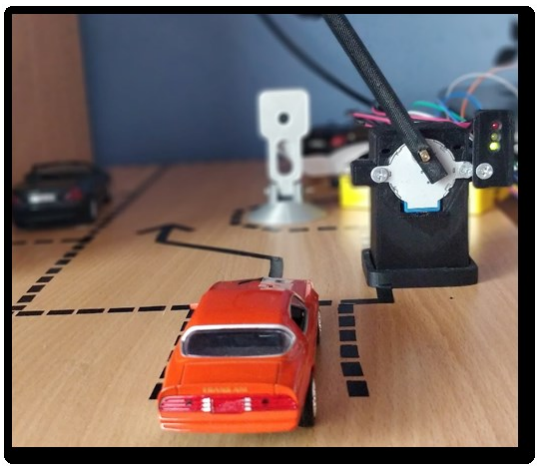

Fig. 7. Access in the parking

c) Red: No Access

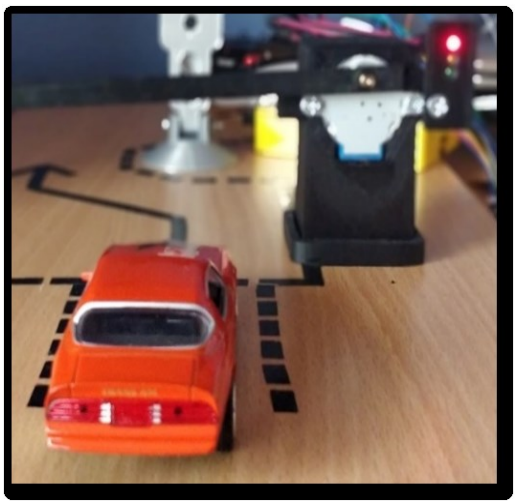

Fig. 8. No access

3) Stepper motor: Open and close the barrier.

To operate the barrier we use a stepper motor along with a python script.

$sudo python motor.py 1 60 cw – open the barrier (spinning clockwise with 60°)

$sudo python motor.py 1 60 ccw – close the barrier (spinning counter clockwise with 60°)

Where *motor.py* is the file that contains python script for operating the barrier, 1 represents the speed of spinning, 60 represents with how many degrees is spinning, cw and ccw means the direction in which it moves.

B. Software

1) OpenALPR: is an open source library writen in C++, Java, Node.js, and Python that analyzes video images and streams to identify the license plate numbers [8]. The computer vision algorithm is in Python implemented. The output returned is the text represented by the characters of the license plate. This library can be compiled and run on Linux, Mac and Windows. OpenALPR requires the following additional libraries:

- Tesseract OCR: it is one of the most accurate optical character recognition (OCR) engine for various operating systems.
- OpenCV: it is a library dedicated to computer vision applications enhanced with machine learning algorithms.
- Leptonica: contains useful software, generally for image processing and analysis applications. The license plate region detector uses the Local Binary Pattern (LBP) algorithm [9]. LBP is a simple but effective texture operator that labels the pixels of an image by comparing it with neighboring pixels, and considers the result to be a binary number. The most important property of the LBP operator in real-world applications is its robustness to monotonic changes in gray level, especially caused by lighting variations.

We simply call the algorithm on the picture created when the button is pressed, and the algorithm returns the car's number. To access the camera to take a picture we use the following command, where cam.jpg represents the image name.

$raspistill -o cam.jpg

To apply the *ALPR* algorithm to a picture, we use the following command:

$alpr -c eu cam.jpg

This command can be used only on the cars with European license plate number, that's why the first parameter is "eu". If we want to apply on American license plate numbers, we change the "eu" parameter with "us".

Fig. 9. ALPR algorithm

2) MySQL: Is used for database. The database architecture can be viewed on fig. 10.

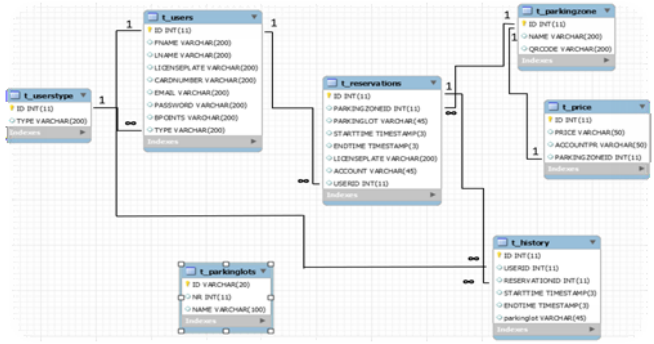

Fig. 10. Database architecture

The users table contains all information about users, including user type which can be "User", "Visitor" or "Admin". Reservations table contains all reservations made using the online reservation, while history table contains only the cars that actually entered in the parking. Prices table contains the cost/hour for every parking. The parking lots table contains the number of parking spaces from the application because those are made dynamically. We choose to do that because those numbers can be changed occasionally, from example during a certain event, some places can be reserved by institution or other parking spaces are built.

This number of parking spaces changed can be made only by admins, from configuration page.

3) Node-Red: represents a programming tool used in general for IOT programming that connects hardware components with API's and online services in new and interesting ways.

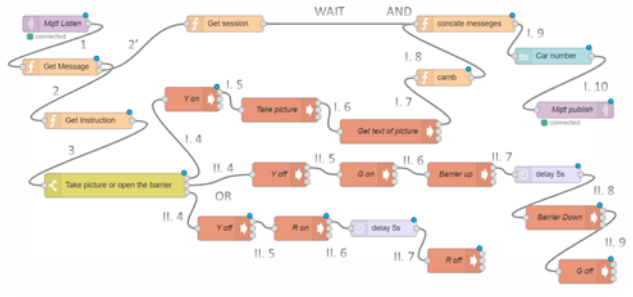

Fig. 11. Access in the parking actions

In fig. 11 is presented the whole flow of Node-Red. The purple nodes are MQTT nodes. "MQTT in" is waiting for messages from web application while "MQTT out" is sending messages to web application. The nodes with an "*f*" are functions, the yellow node is a switch node, the nodes with an arrow are execution nodes, delay nodes represent a delay of 5 seconds, and the blue node represents a text output that displays the final message that has to be sent.

When a message arrives, using function nodes we get the session and instruction from message. If instruction is "new Car" we go on the first value of switch. This value determine the execution on few commands such as turn yellow led on, take a picture, get the license plate number from picture, concatenate with session and send message. The second value of switch determine the fact that the car has access in parking and send the commands to turn off yellow led, turn on the green one, open the barrier, wait 5 seconds, close the barrier and turn off the green led. Last value of switch represents the fact that the car doesn't have access and turns off yellow led, turn on red led, wait 5 second and turn off the red led.

4) DevExpress: Is software developing company which, in present has very much ASP .NET controls and more others.

The web application was divided into several modules presented as finite functionalities to have a good functionality, an easy user understanding, but also to achieve ultimate goal.

This web application is hosted on a free host (*gear.host*), and the link where this application can be seen is: http://sibiuparking.gear.host/Pages/Home

We implemented the 3 menu types that can be seen in the following figure.

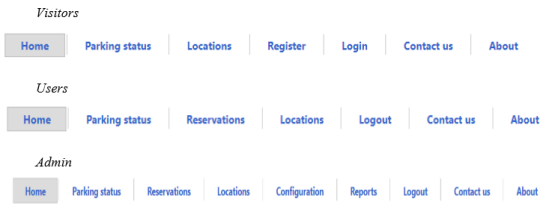

Fig. 12. Menu type depending of the user type

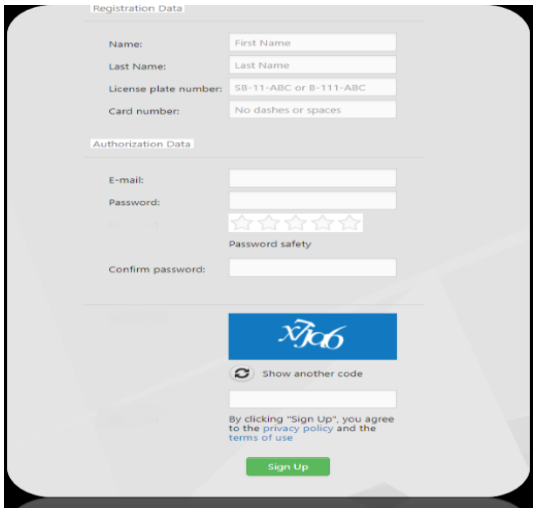

Fig. 13. Register form

The Register form from web applications is completely made from DevExpress controls. This form can be viewed on fig. 13.

5) MQTT: (Message Queuing Telemetry Transport) is a connectivity protocol M2M (machine-to-machine).

The messages are sent and received using an online MQTT broker. For this thing we need two topics which will be used with MQTT. A topic is used for sending messages such as "*testSibiuSend*" and the other one "*testSibiuReceived*" is a topic where we have to subscribe to receive messages. So, on driver action to push the arrive button, a message will be sent using MQTT to a topic. The Node-Red receives the message using the MQTT in node. After all executions from fig. 11, a final message that contains license plate number is sent to another topic. The web application is subscribing to this topic and the message will be received to further operations.

As trial session for this project, we used the same camera to update the parking spaces in the moment when the car leave the parking. So, if a car want to leave the parking is applying the same *ALPR* algorithm. In the web application we check if that license plate number is inside the parking or not. If it is, we update the parking spaces and open the barrier along with green led. Otherwise, means that the car wants to get in the parking. Another solution for this would be using ultrasonic sensors that detects present or absence of the car in front of those. When the car leaves, sensors detects this thing and update the database.

We choose the first option for this project because is simpler, following to adopt the second option witch is safer.

VI. Scalability Issues

Even we designed a prototype tested for Sibiu case study (a city with less a million inhabitants) we consider that our embedded system solution is scalable at least to a medium level. First, our idea aims private institutions (generally universities but also others which would share excessive parking places) and in such cases even the city is bigger the number of parking place is no more than few thousands.

According to [10] the top 10 biggest parking lots in the world (most of them belonging to airports) have between 8000 and 20000 places with maximum 60 entry points. The solution is not applying to airports parking lots because they are always open not like previously mentioned institutions which have periods of time during the day / month / year when would share their exceeding parking space.

From hardware viewpoint a scalable solution means multiplication of all resources required by one entry with the number of entries in the park. Also, we propose using a camera that embedded the license plate recognition algorithm and works regardless of the weather conditions. If there are, e.g. 10000 parking spaces, we would need a web server that offers a higher storage capability and RAM. Also, we will need a larger storage capacity for the database as well. The database could be modified, it can be adapted to the linear database for a higher speed of the web application.

At the software level, the parking number can be changed at any time. The problem is that we use an online *mqtt* broker that supports maximum 1024 connections. Within the reservation process the client chooses the parking space in case someone keeps booking faster, when the other person wants to select that parking space, a message can be displayed to choose another location and is updated with the remaining parking spaces. The *mqtt* broker is used only on the main page where, when the driver reaches the barrier, press the button to start the camera to read the registration number (which means there is no need for more than one connection). In the real-time parking viewing page, parking spaces should be made smaller, spaced as many as possible (about 20-25 per row, with more than one row), but this means that this page will have many parking places, their viewing will be done with a scroll and its loading will be a bit harder.

VII. Conclusions and Further Work

Our developed smart parking system is functional, scalable and presents many advantages. It will also extended to other institutions.

ALPR algorithm can be applied for any picture that has a license plate, even those with US license plate numbers. Usually, this algorithm returns 10 results, but first, which has the biggest confidence percentage is more often the right one.

As momentary limitations is that camera used is not a professional one and the license plate numbers are not always correct because this depends of many conditions such as brightness. Another problem that we need to tackle is that we have no certainty that the driver is parking on the right parking space.

As a further work we intend to implement the payment system using the application. Differentiated pricing according to the number of parking hours or even days. Thus, if a car is parked for a long time, the price will be lower than the product of the number of hours and the hourly cost.

Implementing the concept of "Gamification" is one of the most important because by awarding the user with bonus points for reservations made as accurately as possible, also determine the correct management of the parking spaces.

Another further work is to view real-time parking spaces with a camera and an algorithm of cars detection to see of which parking space every car is parking.

REFERENCES

[1] Tomaszewska E.J., Florea A., "Urban smart mobility in the scientific literature - bibliometric analysis", Engineering Management in Production and Services, Volume 10, Issue 2, 2018, ISSN 2543-6597, Poland, DOI: 10.2478/emj-2018-0010.

[2] Y. Geng and C.G. Cassandras. "A new smart parking system infrastructure and implementation". Procedia-Social and Behavioral Sciences, 54:1278–1287, 2012.

[3] Oliveira-Lima, J. A., Morais, R., Martins, J. F., Florea, A., Lima, C. (2016). "Load forecast on intelligent buildings based on temporary occupancy monitoring". Energy and Buildings, 116, 512-521.

[4] Hongwei Wang, "A reservation-based Smart Parking System" University of Nebraska-Lincoln, July 2011.

[5] International Telecommunication Union ITU-T X.200, "Information technology – Open Systems Interconnection – Basic Reference Model: The basic model", July 1994.

[6] Omkar Dokur, "Embedded System Design of a Real-time Parking Guidance System", University of South Florida, October 2015.

[7] Falz Ibrahlm Shaikh, Pratlk Nirnay Jadhav, Saldeep Pradeep Bandarkar, Omkar Pradip Kulkarni, Nikhilkumar B. Shardoor, "Smart Parking System Based on Embedded System and Sensor Network" JSPM's Jayantrao of Sqwant College of Engineering, Pune, India, April 2016.

[8] https://github.com/openalpr/openalpr, Retrieved at 05.07.2018.

[9] DC. He and L. Wang (1990), "Texture Unit, Texture Spectrum, and Texture Analysis", Geoscience and Remote Sensing, IEEE Transactions on, vol. 28, pp. 509 - 512.

[10] Rajkamal Narayanan, "Top 10 Biggest Parking Lots In The World", 6 April 2015, https://www.drivespark.com/off-beat/top-10-biggest-car-parking-in-the-world/articlecontent-pf18346-010286.html, Retrieved at 19.08.2018.