Simulation of the Automatic Parking Assist System as a Method of the Algorithm Development Thinking

PETR COUFAL, TOMAS HORNIK, STEPAN HUBALOVSKY, MICHAL MUSILEK

Abstract - The teaching of algorithm development and programming should develop not only practical skills, theoretical knowledge and techniques, but algorithmic thinking in particular. [1] The ability of algorithmic thinking can be cultivated through implementation of modeling and computer simulation of real phenomena into the teaching of algorithm development not only at universities, but also at high schools and elementary schools. [2, 3] The process of complex problem solving leads the students to implement problem analysis in order to figure out the given problem. Whole process of the problem analysis, including its solution, strategy, algorithm development and creation of the computer simulation, is presented in the article in a step by step form of the case study simulating a real process - the Automatic Active Parking Assist System. Computer simulation of this process is presented by means of robotic system LEGO® Mindstorms® and programmed in the integrated development environment of LEGO® Mindstorms® NXT-G.

Keywords - Active Parking Assist, Algorithm Development, Algorithmic Thinking, LEGO® Mindstorms®, Robot Model, Sensors.

I. INTRODUCTION

The ever rising complexity of various educational aids, such as for example LEGO® Mindstorms®, allows teachers to easily include different intricate real-world systems into the common curriculum. There is no longer need for a purely theoretical explanation, that can be substituted by working knowledge of the given system.

Enormous progress has been made in the area of autonomous cars, mostly because of Google self-driving car project, which started in 2009 and was renamed to Waymo in December, 2016. [4] One of the most important parts is the company modification of already existing LiDAR technology Light Detection and Ranging), that is mounted on the car and provides the system with data necessary for making all the decision regarding the driving.

LiDAR itself is a system of lasers measuring ranges from obstacles, which creates precise 3D model of the given area. On top of that Google/Waymo created algorithms that discern exact type of obstacle (pedestrians, cyclists, animals, garbage cans, other cars, buses, and so on) in real time and give the information to the driving program.

Autonomous cars are the future of all the car industry, and as such it is important for people to at least broadly understand the technology behind it. Simulation of a system so complex in school conditions and on a budget is near impossible. However, there is a direct predecessor called Active Parking System or Active Park Assist.

Active Parking Assist also takes control of the whole car, yet it happens only during a parking process. When the driver wants to park and the circumstances are assessed as optimal, the car is able to execute parallel parking all on its own. This system is regularly used by companies like Ford or Mercedes. [5], [6]

In this paper we introduce a case study illustrating both construction and programming of a LEGO® Mindstorms® model, including problem solution, and algorithm development of the Active Parking Assist system. Computer simulation of this process is presented in visual programming system for LEGO® Mindstorms® NXT-G 2.0.

II. PROBLEM FORMULATION

The article shows the possibility of a simulation of a real-world system within the confines of a school class. The creation of similarly complex tasks, that links the World of Work (WoW) with the Inquiry-Based Learning (IBL), has been traditional at our department. One of the authors was a member of the

Petr Coufal, Tomas Hornik,stepan Hubalovsky, Michal Musilek are with the Department of Cybernetics, University of Hradec Kralove Rokitanskeho 62, Hradec Kralove, CZECH REPUBLIC petr.coufal@uhk.cz, tomas.hornik@uhk.cz stepan.hubalovsky@uhk.cz, michal.musilek@uhk.cz

international team in MaSciL project [7] oriented on connecting WoW and IBL. To imitate the system, specific problems have to be solved regarding the features and the limitations resulting from the use of LEGO® Mindstorms® construction set:

- What are the minimal necessary dimensions of a parking place for the given car model?
- What is the specific construction of the car?
- Which sensors will be mounted on the car?
- What are the sensors limitations?
- Which algorithm will control the car?

All these questions are connected, because the choice of the car construction defines the minimal dimensions as well as the positioning of the selected sensors. These decisions determine exact form of the final algorithm.

Parallel Parking with Active Parking Assist

In a car with the Active Parking Assist the driver is in full control of the car until he or she reaches a parking lot. Then there are several different versions of Active Parking Assist implementation based on the given car manufacturing company as well as particular car model. The versions vary in the level of car autonomy and the amount of driver's involvement in the parking process.

Our version emulates one of the most advanced variants - the driver completely hands control over to the Active Parking Assist, which continues moving forward while using side ultrasound sensor to determine whether there is necessary amount of space for parallel parking. [8] If such a place is found, the car itself starts the parallel parking maneuver, as depicted in Figure 1.

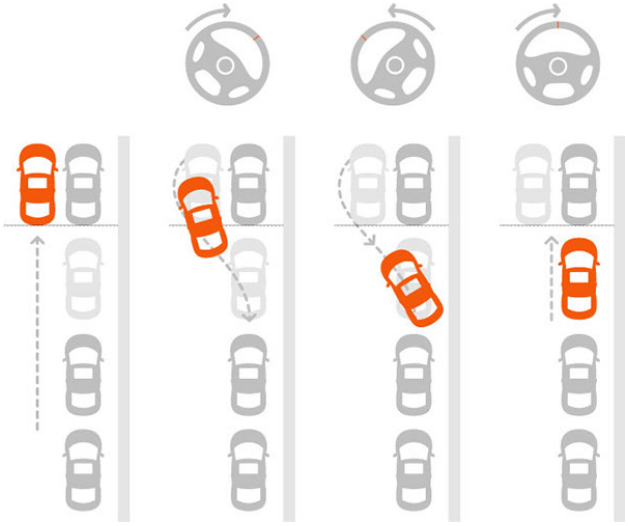

Fig. 1 Parallel parking scheme [8]

III. Problem Solution

This section demonstrates the actual construction, analysis of individual problems and possible states during the parking process and the algorithm itself. The section is concluded by the flowchart and the program created in the development environment LEGO® Mindstorms® NXT-G 2.0.

A. Construction of the car model

The underlying construction of the robotic car model is based on the original instructions for the "Intelligent Car" created by LEGO® company. The model is from the second generation of LEGO® Mindstorms® referred to as NXT (standing for NeXT) and two construction sets are necessary for its assembly, 9695 and 9797. [9] The basic design was modified and enhanced in order to meet the requirements placed upon the robotic car model.
The requirements include:

- addition of ultrasonic sensor facing the right side of the car model;
- addition of the EOPD sensor (Electro Optical Proximity Detector) facing the right side of the car model;
- addition of light bulbs as signaling devices for reverse car movement;
- efficiency improvement of gears used for model steering;
- maintaining the model compactness.

The resulting construction complies with all the requirements.

There were several different reasons for choosing specifically the "Intelligent Car" model. The design uses only basic construction sets 9695 and 9797. The model utilizes wide range of sensors - an ultrasonic sensor, a light sensor and two touch sensors. The interactive servo motors are located in the central section near the center of gravity. One of the motors controls direct power rack and pinion steering utilizing rubber band transfer, while the other motor procures model movement via rear differential driving axle. Whole model is compact with easy access to the control unit.

B. Enhancements made on the construction

The first model improvement is the replacement of a rubber band transfer located in the steering section. The rubber band was replaced by a set of two cog gears while keeping the same transfer rate 1:1, but changing the resulting rotation direction of the whole steering system. This issue is addressed by a simple change of the motor rotation direction in the control program.

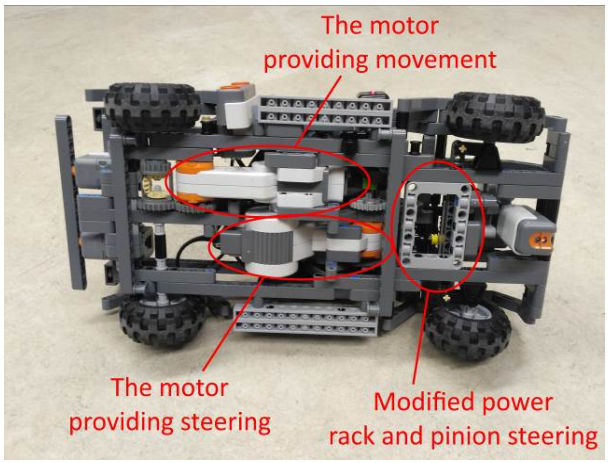

Fig. 2 Model chassis with the modification

Another improvement is located on the right side of the model. Two sensors were added there pointing perpendicularly to the car model. In the back section between the cab and the rear right wheel there is an ultrasonic sensor for making distance measurements. The EOPD sensor incorporated into the front section behind the front right wheel is used for exact measurement of shorter distances. Positioning of sensors in these places is convenient for easy access to input ports of the control unit as well as for leaving the truck deck empty for further expansions of the model. These positions are also particularly suitable for distance measuring needed for the Active Park Assist.

Reverse car movement is signalized by a pair of mutually connected light bulbs placed in the rear section of the truck deck. The bulbs are placed on the upper side of the deck for better visibility, however in case of necessity, they can be moved to the lower side leaving the whole truck deck empty. Reverse car movement signalization has obviously no impact on the functionality of the Active Parking Assist system, but it is an element strengthening the connection to the real world, i.e. WoW.

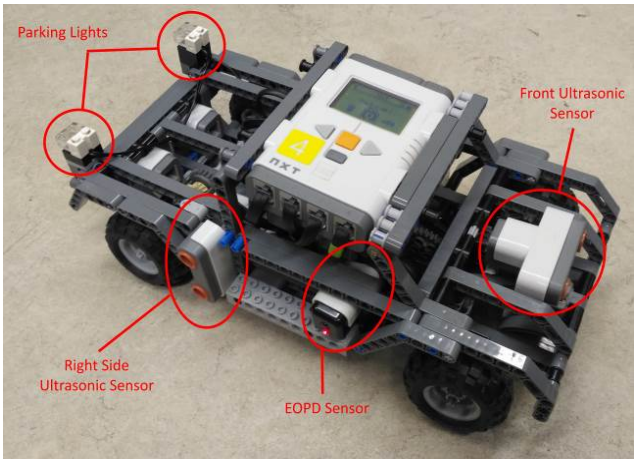

Fig. 3 Enhancements made on the model

The compactness of the model remained intact even after so many changes and enhancements. Keeping the truck deck empty and thus leaving the option for further model expansions was considered as one of the main goals.

C. Sensors

The total number of utilized sensors is six, which is more than the control unit can link. Thus our model uses only selected sensors. In the rear section there are two touch sensors, which can be used for impact monitoring. In the front section there is a light sensor located so that it can follow the surface on which the model is moving. This sensor can be used for tracing and following a black line. None of those sensors was used in the program and remained in the model as a possibility for future expansion.

Fig. 4 Different types of sensors used in the construction

Ultrasonic sensor located in the front section measures the distance in front of the model. Analogously the ultrasonic sensor located on the right side along with the EOPD sensor are both used for measuring the distance from the side of the car.

Incorporated NXT motors can be used as drive units or as rotation sensors. This feature is used in our program for the motor propelling the rear differential driving axle.

Ultrasonic Sensor

The ultrasonic sensor is based on sonar principle and serves for measuring the distance up to 250 cm. The measuring is done by means of an integrated source of ultrasonic waves, which sends the waves towards the observed item. The waves bounce back from the item and return back to the sensor that measures the distance based on the time necessary for the whole process.

The distance values are within the interval of 0-100 inches or 0-250 cm. The measurement deviation is ± 3 cm. Measurement quality is dependent on the distance, size and material of the item, that reflects the waves. Big items from hard materials reflect the waves considerably better than small items from soft materials.

EOPD Sensor

Electro Optical Proximity Detector (EOPD) serves for exact measurements of short distances up to 20 cm. Measuring is done by an integrated source of pulse modulated light, which emits light towards the observed item. The sensor records reflected light intensity value, which it compares with established value of background light intensity, which is in our case the place where the measuring is being done.

The sensor makes 350-400 measurements per second. It works in two different modes labeled as 1x and 4x, which differ in measuring distances and in possibilities of item detection. The item detection accuracy depends on its distance, size, shape and above all the surface structure that reflects the light back to the sensor.

Unlike the ultrasonic sensor, the EOPD sensor doesn't work with the time necessary for the return of reflected waves, but with the intensity of reflected light. The raw readings range from 0 to 1024, measurement deviation is 12. From these readings the exact distance is calculated by the following formula:

$$distance = \frac{kScale}{\sqrt{rawsensorvalue}} - kError$$

Fig. 5 EOPD Distance Calculation Formula [10]

In the formula kScale is a constant that sets the scale of the value and kError is a constant defining the difference between the calculated distance and the actual distance. [10] Quoted values are only illustrative and the sensor has to be calibrated for accurate readings and calculations.

Rotation Sensor

NXT motors can be used as a source of rotation movement or as a rotation sensor. Motors include integrated rotation sensor and a sequence of gears, that have resulting gear ration 1:48. The turn of the motor can be regulated in degrees, rotations or in seconds. While using the motor as a rotation sensor the output values are stated either in degrees or in rotations. Positive and negative signs indicate the rotation direction.

Other motor parameters, such as rotational mechanical force, torque moment and rotation speed, are directly dependent on battery voltage powering the control unit.

D. Parameters of the Employed Sensors

The NXT control unit has four input ports for sensors, which is a limiting factor for our model incorporating six sensors in total. In order to plug in all the built-in sensors, or to add other sensors, it is necessary to supplement the model with a sensor multiplexor powered by a separate power source.

Three motor ports on the control unit are filled with a motor for movement, a motor providing the steering and the third port contains the light bulbs for signaling the reverse car movement. In order to extend the amount of motor ports it is again necessary to add a multiplexor for NXT motors.

E. Possible Future Expansions of the Model

The model design can be supplemented with more sensors, which would allow precise parking and also expand the amount of different parking methods. In order to do so, it is necessary to add ultrasound and EOPD sensors also on the left side of the model as well as on the rear side. Such an expansion however requires use of the aforementioned multiplexors.

Another option for further expansion of the model is to use different sensors altogether, e.g. an accelerometer, a compass or a gyroscope.

Apart from the model expansion through addition of more sensors, the option to fully utilize the sensors already mounted on the vehicle by means of the control program enhancement remains. The light sensor situated in the front section of the vehicle model allows the car to follow a black line on the surface, which can be used for fully autonomous driving in a given area and subsequent parking chosen according to the available space.

IV. ALGORITHM DEVELOPMENT

An algorithm can be defined as a sequence of instructions describing the procedure of solving given problem. The generally accepted norm for algorithm presentation is a flowchart. When the problem is analyzed and the algorithm is created, it can usually be written in more than one programming language. LEGO® created its own integrated development environment called LEGO® Mindstorms® NXT-G 2.0. In following subchapters, we will describe individual problems and solutions.

A. Analysis of the possible states

Fully autonomous system has to be able to deal with all the possible exceptions, that can happen during the course of its execution. For the educational purposes the solution has been simplified and certain exceptions have been left out.

The distance measuring in our model remained very close to the original, however the parking process itself is reduced into precisely given steps that are repeated exactly in the same way. In other words, where the real car uses external sensors for precise parking, our model parks always in the same way, not taking into account any sensor data. This leaves the option for future alteration of the program with more complex parking procedure utilizing aforementioned addition of a sensor multiplexor.

The starting point

The beginning of the program execution is created just like the real Active Parking Assist, i.e. the driver arrives at the parking lot, sets the car along the line of already parked cars and then gives over the control to the computer. In our case the model is set at the starting point manually and then the program is started.

Identification of an obstacle in front of the car

While searching for an empty parking slot, the car has to continually check, whether there is still space ahead. In reality, the car has to check either for a wall, a curb or some other small obstacle that would prevent the parking process. Our model expects, that the parking lot ends with a wall. With the use of an input port multiplexor, the front light sensor could be checking for a line imitating curb. In case of an obstacle in front of the car in a distance of 35 cm or less, the car stops.

Measuring the depth and length of a parking space

The car continually checks with the ultrasonic sensor (or EOPD) distance from the nearest obstacle on the right side. If there is none or if it is further than 20 cm, it means that the car has enough depth to be parked there. From this point, the motor starts behaving also as a rotation sensor counting the length of the parking space. If the space is long enough, the car stops moving forward along the line of parked cars. If there is an obstacle and the distance on the right side of the model becomes smaller than 20 cm, the variable counting the length of the parking space is zeroed.

Parallel parking maneuver

If a parking space big enough for the parking maneuver is found, the car parks itself. The real systems are still using the external sensors to check distances on all sides of the vehicle during the whole parking process. Our simplified version jumps out of all the loops checking distances and gives control only to the motors. They are procedurally synchronized to park the car in the same way as shown in figure 1.

B. Analysis of the robot motion

Move forward

detection:
- front ultrasonic sensor detects no obstacles in front of the car within the distance of 35 cm;

action:
- the model keeps steering motor centered, while the propulsion motor moves the car forward;
- the side sensor is activated and measures the depth and length of potential parking spaces.

Start counting the length of a potential parking slot

detection:
- front ultrasonic sensor detects no obstacles within the 35 cm ahead of the vehicle;
- side ultrasonic sensor detects no obstacles within the 20 cm from the side of the vehicle;

action:
- the propulsion motor begins to serve also as a rotation sensor storing the parking spot length value in a variable.

Parking slot is too short

detection:
- front ultrasonic sensor detects no obstacles within the 35 cm ahead of the vehicle;
- side ultrasonic sensor starts detecting an obstacle within the 20 cm from the side;

action:
- the variable holding the parking spot length is zeroed and searching for the beginning of a new parking slot starts again.

Parallel parking

detection:
- a parking slot longer than 35 cm is found;

action:
- front and side ultrasonic sensors are stopped;
- the motor stops acting as a rotation sensor;
- sequence of parking instructions is commenced.

Stop the car, terminate the program

detection:
- no appropriate parking slot detected;
- the front ultrasonic sensor detects an obstacle ahead of the car in a distance of 35 cm or less;

action:
- front and side ultrasonic sensors are stopped;
- the motor stops acting as a rotation sensor;
- stop the forward movement of the car;
- write error message "No parking space".

C. *Algorithm of solution*

The solution of the problem is presented in the form of a flowchart (shown on figure 6) as well as a screenshot of the program itself created in the integrated development environment of LEGO® Mindstorms® NXT-G 2.0 (figure 7).

Despite the fact, that it is possible to rewrite the final algorithm in a form of a flowchart and the program itself seems to be procedural, in reality the LEGO® Mindstorms® NXT-G 2.0 belongs under the event-driven programming paradigm.

Continuous distance control from the nearest obstacle in front of the car is difficult to represent in a traditional flowchart. The event of achieving the set limit distance from the obstacle in front calls immediate interruption of the program and switch into the program termination part.

V. CONCLUSION

There is a vide variety of different approaches for training and cultivation of algorithm development thinking of pupils and students.

The paper offered insight into one of the possibilities, which is the implementation of computer simulation principles of real world phenomena and processes into the education process of algorithm development and programming. Authors described in detail the problem analysis of an Active Parking System, the problem solution, strategy, algorithm development and creation of the computer simulation within the frame of the case study. Similar examples are incorporated into the programming education at the UHK. [11, 12]

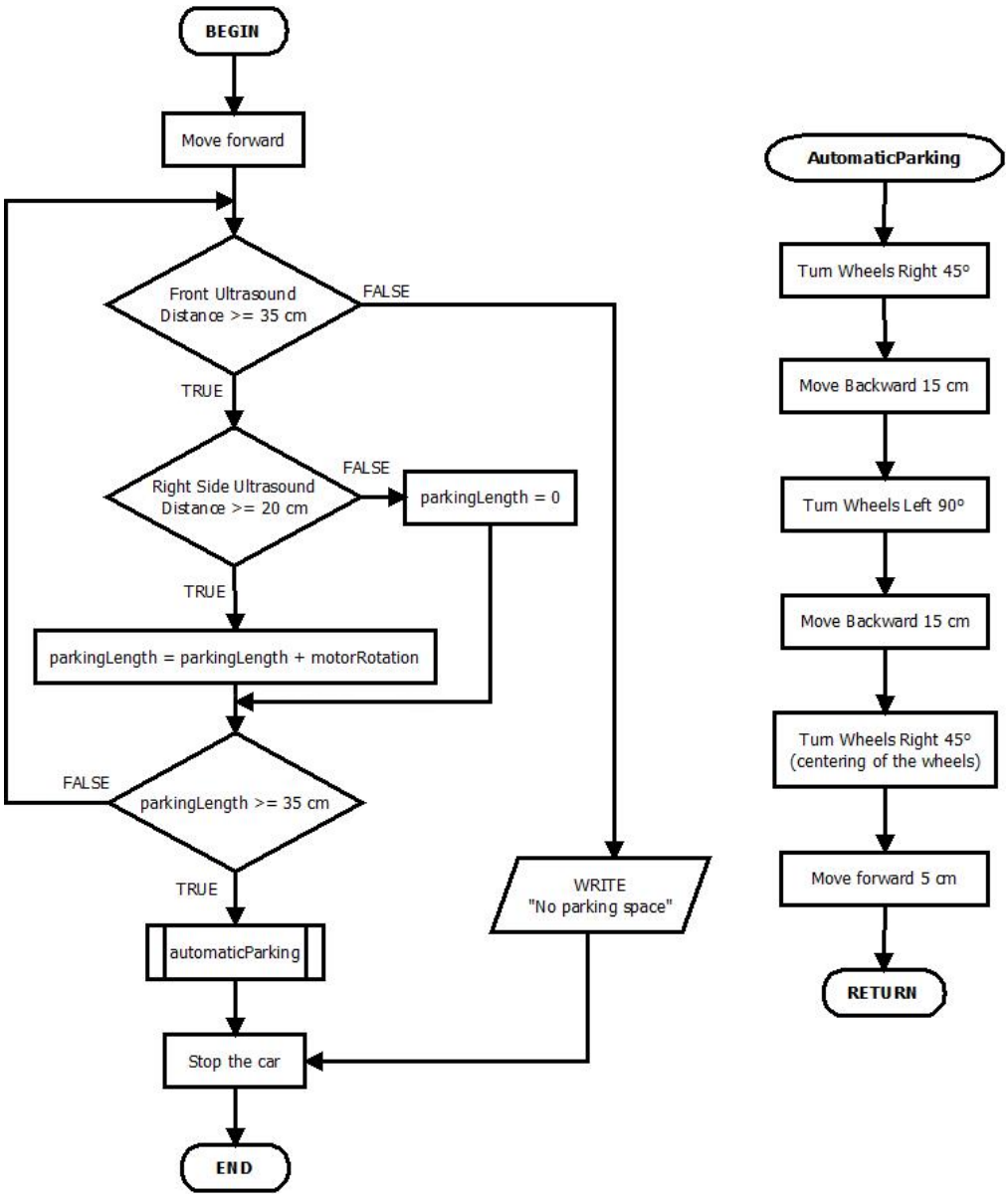

Fig. 6 Flowchart describing the program solution

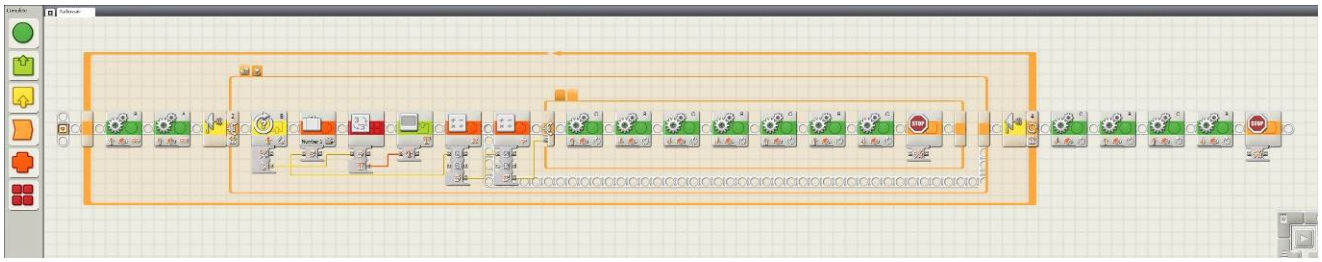

Fig. 7 The NXT-G program for Automatic Parallel Parking

Acknowledgment
This research has been partially supported by the Specific research project of the Faculty of Education of the University of Hradec Kralove 2017 and Specific research project of the Faculty of Science of the University of Hradec Kralove 2017.

References:

[1] Hubalovsky, S., Modeling and Simulation of Real Process – Passing through the Labyrinth as a Method of Development of Algorithm Thinking and Programming Skills. *International Journal of Mathematics and Computers in Simulation*. 2013, Vol. 7, No. 2, p. 125-133. ISSN 1998-0159.

[2] Hubalovsky, S., Modeling and computer simulation of real process – solution of Mastermind board game. *International Journal of Mathematics and Computers in Simulation*. 2012, Vol. 6, No. 1, p. 107-118. ISSN 1998-0159.

[3] Hubalovska, M., Hubalovsky, S., Implementation of the Systems Approach in Mathematical Modeling, Dynamic Simulation and Visualization Using MS Excel Spreadsheet. *International Journal of Mathematics and Computers in Simulation*. 2013, Vol. 7, No. 2, p. 267-276. ISSN 1998-0159.

[4] Krafcik, John. (Dec 13, 2016). "Say hello to Waymo: What's next for Google's self-driving car project." In: *Medium*. [online]. Accessed March 9, 2017. Available: <https://medium.com/waymo/say-hello-to-waymo-whats-next-for-google-s-self-driving-car-project-b854578b24ee#.26txvbbsg>

[5] Sauser, Brittany. (Jan 6, 2009). "Ford's New Car Parks Itself." In: *MIT Technology Review*. [online]. Accessed March 4, 2017. Available: <https://www.technologyreview.com/s/411563/fords-new-car-parks-itself/>

[6] "Active Parking Assist." In: *Mercedes Benz*. [online]. Accessed March 4, 2017. Available: <http://techcenter.mercedes-benz.com/cs_CZ/active_parking/detail.html>

[7] Michiel, D. et al. 2016. *MaSciL Mathematics and Science in Life: Inquiry Learning and the World of Work*. Freiburg: University of Education. ISBN: 978-90-70786-35-9.

[8] Biello, Rachel. (Jan 1, 2014). "Parallel parking instructions." In: *Behance*. [online]. Accessed March 4, 2017. Available: <https://www.behance.net/gallery/13434875/Parallel-parking-instructions>

[9] The LEGO Group. 2010. "Intelligent Car: 9695 + 9797." In: *Lego Education: Building Instruction.* [online]. Accessed March 6, 2017. Available: <http://lego.sasbadi.com/cms/upload_files/download/g000021.pdf>

[10] "EOPD – How to measure distance." In: *HiTechnic Blog.* [online]. Miami: HiTechnic Products, 2010. Accessed March 6, 2017. Available: <http://www.hitechnic.com/blog/eopd-sensor/eopd-how-to-measure-distance/>

[11] Hubalovsky, S., Musilek, M., Modeling, Simulation and Visualization of Real Processes in LOGO Programming Language as a Method of Development of Algorithm Thinking and Programming Skills. *International Journal of Mathematics and Computers in Simulation*. 2013, Vol. 7, No. 2, p. 144-152. ISSN 1998-0159.

[12] Musílek Michal, Hubálovský Štěpán, Hubálovská Marie. Mathematical Modeling and Computer Simulation of Codes with Variable Bit-Length. *International Journal of Applied Mathematics and Statistics*, Vol. 56; Issue # 1, p. 1-12. ISSN 0973-1377.