[6] Krupnyk L. I., Pavlova N. P., Fedorchenko A. N. ta in. Hazofaznyi katalitychnyi reactor/ Patent Ukrainy № 11808, MPK: B01J 8/02, zaiavl 11.07.89 № 4717489/26, opubl. 25.12.1996.
[7] Stepanov A. V. Trubchastyi katalitychnyi reactor/ Patent Ukrainy №4573, MPK: B01J 8/02, zaiavl 07.02.91 №4929007/26, opubl. 28.12.94. Biul № 7-1.
[8] Zardi U., Pahani D. Sposib provedennia ekzotermichnoho heterohennoho syntezu ta reaktor dlia yoho zdiisnennia/ Patent Ukrainy №1854, MPK: B01J 8/04, zaiavl. 27.06.88 № 2457/80, opubl. 20.12.1994.
[9] Kheihveish D. P., Brain P. F., Trambol S. Ye. Sposib katalitychnoho ryforminhu vuhlevodniu ta reaktorna systema dlia katalitychnoho ryforminhu/ Patent Ukrainy №51609, MPK: C10G 9/16, C10G 35/00, zaiavl. 06.03.1992 № 93004174, opubl 16.12.2002.
[10] Milberger E. C. Automatic catalytic screening unit/ Patent US 4099923A IPC G01N31/10, claimed 1977-01-17 №US05/760,198, publication 1978-07-11.
Отримана в редакції 01.11.2023. Прийнята до друку 04.12.2023. Received 01 November 2023. Approved 12 December 2023. Available in Internet 03 January 2024

**УДК 004.896**

# AUTOMATED NAVIGATION FOR UNMANNED GROUND VEHICLES IN LOGISTICS

**[1]Oleksandra Danylova, [2]Olha Horishna, [3]Vitalii Onatskyi, [4]Ivan Burlachenko, [5]Volodymyr Savinov**

Petro Mohyla Black Sea National University, Mykolaiv, Ukraine
ORCID: [1]https://orcid.org/0009-0005-7208-2847, [2]https://orcid.org/0009-0005-7627-8337,
[3]https://orcid.org/0009-0007-7207-8375, [4]https://orcid.org/0000-0001-5088-6709,
[5]https://orcid.org/0000-0002-0862-5879
E-mail: [1]danylova.o@chmnu.edu.ua, [2]gorishna.o@chmnu.edu.ua, [3]vitalii.onatskyi@chmnu.edu.ua,
[4]ivan.burlachenko@chmnu.edu.ua, [5]volodymyr.savinov@chmnu.edu.ua




**DOI: https://doi.org/10.15673/atbp.v15i4.2720**

***Abstract.*** *The use of ground-based courier robots is widespread in the systems that integrate the complex automation of transportation management. Most often, the robot's main problems are finding and avoiding obstacles on the way during cargo transportation. Many computing resources are used to model the route taking into account the features of the terrain during logistic navigation. In research, the choice of gear motors, motor drivers, microcontrollers, and microcomputers was justified to ensure movement. To ensure effective autonomous navigation, the robot must contain ToF, LiDAR, ultrasonic sensors, GPS modules, cameras, an accelerometer, and a magnetometer. The selected combination of the investigated hardware components will allow for accurately determining the location of the vehicle. The article analyses the components of the robot. The principle of operation of drivers and engines is defined. Two models of drivers and two models of engines were considered. A schematic of the device with all connected sensors, a controller, and a microcomputer was developed. The principle of interaction of all system components to ensure autonomous navigation was developed. For the long-term operation of the system with autonomous navigation, the power supply system and the capacity of the necessary battery were calculated. A neural network was created and used for object detection. For neural network calculations, a server-side was created that receives frames from the device's camera and returns data on detected objects to the autonomous vehicle.*

***Анотація.*** *Використання наземних роботів-кур’єрів є дуже розповсюдженим в процесах систем, які забезпечують комплексну автоматизацію управління транспортними перевезеннями. Найчастіше основними проблемами робота є знаходження та оминання перешкод на шляху під час вантажоперевезення. Багато обчислювальних ресурсів займає моделювання маршруту з урахуванням особливостей рельєфу місцевості під час логістичної навігації. Для забезпечення руху було обгрунтовано вибір моторів- редукторів, драйверів двигуна, мікроконтролерів та мікрокомп’ютерів. Для забезпечення ефективної автономної навігації робот повинен містити ToF, LiDAR, ультразвукові датчики, GPS-модулі, камери, акселерометр, магнітометр. Обрана комбінація досліджених апаратних компонентів дозволить точніше визначати місцезнаходження*

*транспортного засобу. У статі проаналізовано комплектуючі робота. Визначено принцип роботи драйверів та двигунів. Було розглянути дві моделі драйверів та дві моделі двигунів. Було розроблено схему пристрою зі всіма підключеними датчиками, контролером та мікрокомп'ютером. Був розроблений принцип взаємодії всіх компонентів системи для забезпечення автономної навігації. Для довготривалої роботи системи з автономною навігацією було розраховано систему живлення та ємність необхідного акумулятора. Була створена та використана нейронна мережа для виявлення об'єктів. Для обчислень нейронної мережі була створена серверна частина, яка отримує кадри з камери пристрою та повертає дані про виявленні об'єкти до транспортного засобу.*

**Keywords:** Autonomous navigation, robotics, computer vision, neural networks.
**Ключові слова:** Автономна навігація, робототехніка, комп'ютерний зір, нейронні мережі.

## I. INTRODUCTION

Ground mobile robots are most often used in the delivery segment. The effectiveness of the use of ground-based robot couriers is quite high since robot couriers can reduce the cost of delivery compared to the use of human couriers, which means that the cost of providing the service will decrease, which will serve as an additional reason for the robotics solution integration. Ground mobile robots will have to compete with delivery by flying drones. Before delivery by flying drones, ground delivery by robots has a clear advantage in that neither the cargo nor the courier will injure people, even in the event of a critical situation. In 2016, several companies are testing street robot couriers for commercial use. One of the difficult tasks for creating ground mobile robots is the development of an algorithm that would allow the robot to find its way in the environment with people but in a way as to exclude the risk of collision with them, at least through the fault of the robot. The robots are equipped with sensors, including machine vision cameras, radars, and ultrasonic sensors that detect curbs and walls.

To develop hardware and software for ground mobile robots that autonomously calculate the correct route to destinations, analyse the environment, and bypass obstacles, it is necessary to solve the task of analysing modern autonomous navigation technology step by step. It is necessary to develop software for the Arduino microcontroller to control the navigation modules using the ultrasonic sensor and driving control modules that will interact with the motor driver. An application will be developed to receive data from the navigation modules and build a route for the microcomputer. The application will integrate the functions of building a route to the destination based on the location information obtained from the GPS module and ensuring the necessary control of the power supply system.

## II. LITERATURE ANALYSIS

One of the most important requirements for mobile robotics is the ability to move freely in an environment. Avoid obstacles and respond in time to the occurrence of unusual situations of collision with other moving or stationary objects and respond to the peculiarities of the environment.

### 2.1. Avoidance of obstacles

Automatic control of the robot's movement involves detecting obstacles and making decisions about further actions. In the context of this task, obstacle detection can be conditionally divided into stages. First, direct detection of potential obstacles in the way of moving the mobile device is performed. Then the distance to the obstacle is estimated. Time Of Flight (ToF) methods are used to recognize obstacles in autonomous navigation. All ToF sensors measure distances using the time it takes photons to travel between two points, from the sensor emitter to the target and then to the sensor receiver. 3D ToF cameras can capture depth data in three spatial dimensions. For ranging and distance sensing, ToF is very effective at emitting light rather than sound. Compared to ultrasound, it provides a much longer range, faster readings, and higher accuracy, while maintaining a small size, lightweight, and low power consumption. ToF sensors are used for a number of applications, including robot navigation, vehicle monitoring, people counting, and object detection. Light Detection and Ranging (LiDAR) is a remote sensing technology that uses a laser pulse to collect measurements that can then be used to create 3D models, and maps of objects and the environment. LiDAR works similarly to radar [1, 2] and sonar but uses light waves from a laser instead of radio and sound waves.

The LiDAR system calculates how long light takes to hit the object and reflect back to the scanner. Distance is calculated using the speed of light. The systems can generate about 1,000,000 pulses per second. Each of these measurements can be converted into a three-dimensional visualization as a cloud of points [3, 4]. A cloud of points is a collection of points that represent a three-dimensional shape or object. The data received from the LiDAR are processed by the central processor that controls the autonomous vehicle [5 − 7]. Also, based on the location, thanks to the GPS system, it lays out a route and moves to the destination. An unmanned vehicle needs data on dozens of objects around it. Therefore, the Lidar rotates around its axis, emitting many light flashes, and thus forms a three-dimensional 360-degree image of the surrounding environment from the cloud of points.

Ultrasonic sensors are characterized by high reliability and incredible versatility of use in robotics. They can be used to solve more complex tasks, including object recognition or measuring levels with millimetre accuracy, thanks to the reliability of their measurement method, practically, in any conditions. High-frequency ultrasonic pulses emitted by the sensor are used to measure the distance. These pulses are emitted in the form of a conical beam in the wave frequency range above 20 kHz and are reflected from any surface when they meet. This allows you to detect objects and measure the distance from the ground mobile robot's sensor to the obstacle based on the signal transferring time.

You can also use a video camera to analyse surrounding objects, through which the computer will recognize various objects using a neural network [8] and, based on information about these objects and their position on the frame, will

determine in which direction to move or not to move at all.

**2.2. Determination of position and orientation in space for mobile ground robots**

Also, one of the main requirements for controlling the movement of a robot is to recognize its location and direction [9].

GPS module can be used for location recognition. GPS modules communicate with satellites and receive location coordinates. The most common GPS module [10, 11] is NEO-6m-001. Communication with satellites takes time. The cold start of this module is from 5 to 20 minutes, provided that it is outdoors. Hot start of about 1 second. This module helps to determine the location, but not precisely enough. With the help of the accelerometer onboard the ground mobile robot, you can determine the position of the device in space, and with the help of the magnetometer, you can determine the direction of the autonomous vehicle.

**III. OBJECT, SUBJECT, AND METHODS OF RESEARCH**

Autonomous vehicles can perform their functions as accurately as possible, have efficient performance, and operate for a long time, which will lead to a shorter payback period. A detailed study of autonomous vehicle components helps to improve product quality.

**Purpose of the research**: Development of hardware and software for autonomous devices that calculate the correct route in logistics processes, analyze the environment and avoid obstacles.

**Objective**: Autonomous vehicle navigation system in logistics.

**Subject:** Technologies for autonomous navigation of unmanned ground vehicles.

To achieve the goal, the following tasks must be solved:

- Analyze modern technologies of autonomous navigation;
- Develop software for the Arduino microcontroller to control the navigation and movement modules (ultrasonic sensor, motor driver, etc.);
- Develop an application for receiving data from navigation modules and building a route for a microcomputer;
- Develop an application for building a route to the target based on the location information received from the GPS module;
- Provide a power supply system with the necessary energy reserve for all components.

Autonomous devices can be used in the field of lightweight product delivery. It can also be used to study neural networks with computer vision.

Autonomous vehicles use GPS modules that find satellites and connect to them and, based on the received signals, calculate the current location of the device in space. The module provides information about the location in the form of longitude and latitude coordinates. The primary source of GPS coordinate errors is the distortion of the satellite signal by tall buildings. In large cities with tall buildings, it is difficult for the GPS module to pick up the signal. The signal reaches the receiver having already been reflected from the surface of the buildings. Thus, the path length increases, and the measurement error increases with it. Also, the module will not be able to work in a house because it will not be able to receive a signal.

The GY-511 module uses the LSM303DLHC chip to detect and direct magnetic fields. The communication protocol of this module is I2C, and it can be connected to various processors, including the Arduino board, using the SCL and SDA pins is shown in Figure 1 (a). Microcontrollers are suitable for this purpose, but microcontrollers cannot provide enough current to start the motor and thus can damage the microcontroller. For example, Arduino pins are limited to 40 mA, and motors require a current of at least 100 mA. The l298n driver is used for this purpose. We can change the direction of the current by activating two specific switches at the same time, so we can change the direction of rotation of the motor. A schematic diagram of the module is shown in Figure 1 (b).

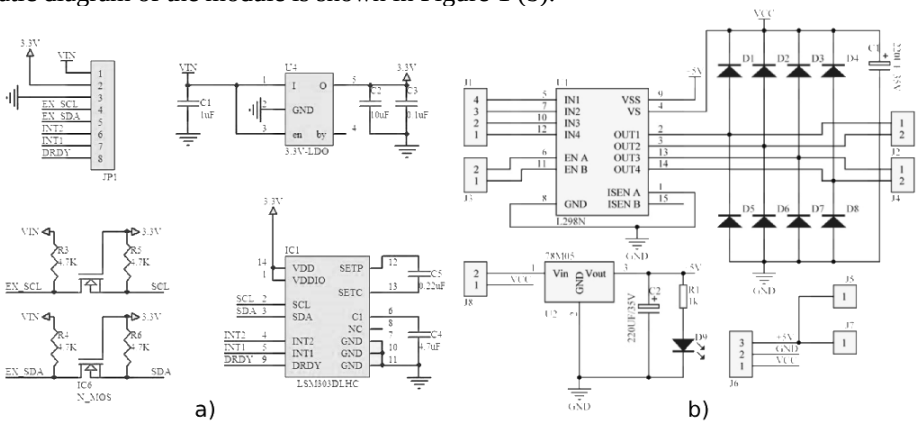



**Fig. 1 (a) − Schematic diagram of the GY-511 module (b) − Schematic diagram of the L298n module**

The main element of the robotic system is the Raspberry Pi 3 B microcomputer. To turn on the entire device, it is enough to turn on only the microcomputer, as it starts a chain reaction of turning on all other components. This can be done by connecting a power source to the Raspberry Pi via Micro-USB or by connecting to the 5V and Ground pins. The Raspberry Pi requires a special Linux-based operating system, which is located on a 16 GB memory card. You can

download the OS on the official Raspberry website and then install it on the memory card. The memory card can be easily inserted into the Raspberry Pi and will store all the information.

The HC-SR04 ultrasonic sensor is connected as follows: Trig connects to pin 5, Echo connects to pin 6, and uses the free power and ground inputs of the Arduino. The GY-511 LSM303DLHC module is connected to the analogue inputs: SCL to A5, and SDA to A4. The module is connected to these inputs because the library for the module works with them. The inputs for the left motor of the L298n driver IN1 and IN2 are connected to pins 8 and 7, respectively, and IN3 and IN4 to pins 3 and 4, respectively. All the jumpers on the module are removed, so ENA and ENB need to be connected to pins 11 and 10 respectively. The l298n module is powered by the battery, but it is used for the motors. For the module circuit to work, you need to connect 5V and GND to the Arduino. Although the module is connected to the power supply, it will not work until the microcontroller is turned on. A schematic diagram of the entire device is shown in Fig. 2.

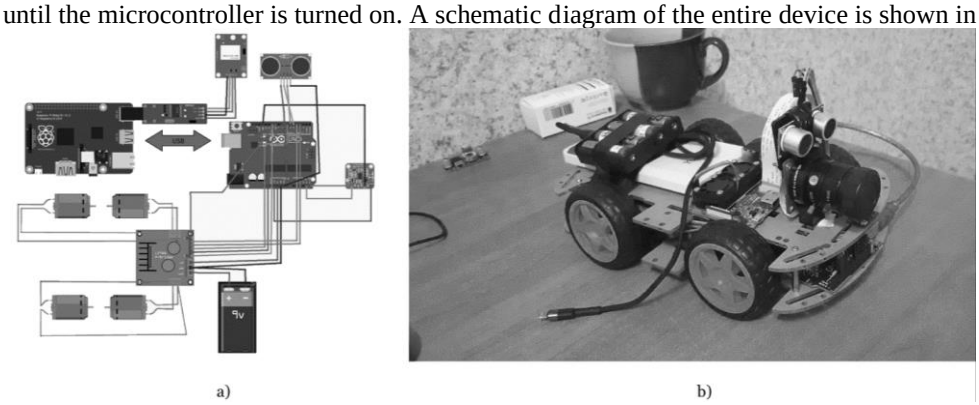



**Fig. 2 (a) − schematic diagram of the device, (b) − constructed prototype**

After the whole system is up and running, the Raspberry Pi connects to the Internet and runs the script to execute the program part. First, the microcomputer tries to connect to the server, which provides information about the destination and will perform additional calculations. At this time, the GPS module searches for satellites. The rest of the software will start running only after the connection with the server and satellites is established.

All other modules are already working. The microcontroller constantly receives information from the modules and transmits it to the microcomputer via the serial port. The microcomputer, in turn, constantly receives more information from the GPS module and the camera that is connected to it. The frames from the camera are transmitted to the server to detect objects that may be obstacles. The server detects objects and transmits information about them to the computer. The Raspberry Pi doesn't have a powerful enough processor for such frame computations, so the server performs this task. Using all this data, the microcomputer calculates the speed of the motors and transmits it to the microcontroller. This is how the route is plotted, and obstacles are recognized and avoided. The principle of the system is shown in Fig.3.

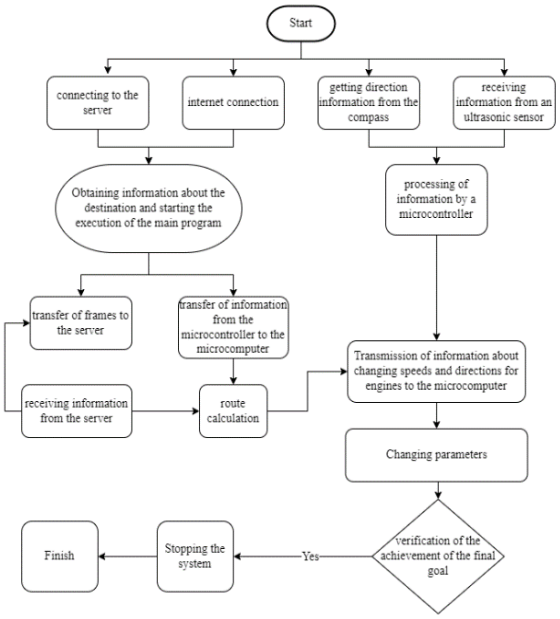



**Fig. 3 − algorithm of the vehicle control system**

The ground mobile robot will need at least 12 hours of battery life for autonomous operation, and a reserve of 24 hours is required for unforeseen situations. Now you can calculate the capacity of the required battery 7131×24=171144 mAh. The system requires a battery of at least 180 Ah. However, these were calculations with not very powerful gear motors.

If we take the motor wheels from the gyroscope and the BLD-300B driver. BLD-300B driver (2 pcs.): maximum current 15A (how much current the chip itself consumes is not specified). Since the wheels-motors consume 2-4 A each,

their number can be up to 3 pieces for each driver. But if you take four 2A wheels and two drivers, whose chips will consume about 1A each, you will get the following calculations:

$$3000 + 40 + (4 \times 2000) + (1000 \times 2) + 15 + 20 + 20 = 13095 \text{ mA}$$

The BLD-300B driver with motor wheels needs a more powerful battery because now the system consumes much more current. The capacity of the required battery is 13095×24=314280 mAh. The robot requires a battery of at least 320 Ah.

## IV. RESULTS

### 4.1. Learning a neural network model

A camera is also used for object recognition. However, recognition requires the creation of a neural network that takes a vector of input data and returns a vector of output data. A neural network has an input layer, one or more hidden layers, and an output layer. A layer can have many nodes. Also, a neural network can have many layers.

Image classification involves predicting the class of a single object in an image. Object localization refers to determining the location of one or more objects in an image and drawing a large rectangle around their boundaries. Object detection combines these two tasks and localizes and classifies one or more objects in an image.

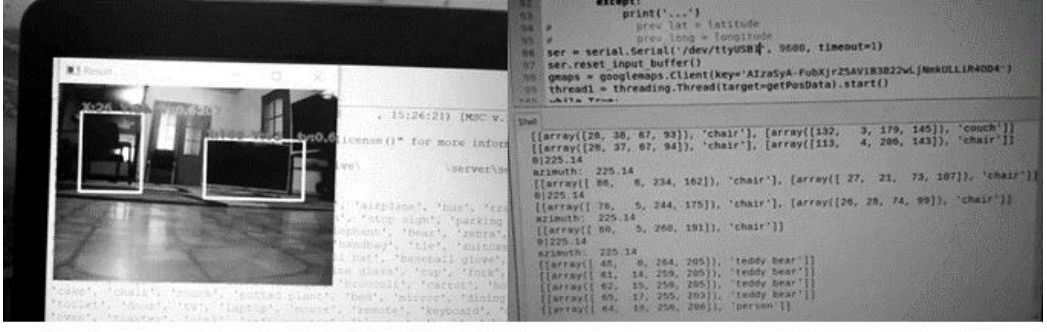
One of the key components of most computer vision applications based on deep learning is a neural network (CNN). A CNN is a type of neural network that efficiently captures patterns in multidimensional spaces. Each convolutional neural network consists of one or more convolutional layers - a software component that extracts meaningful values from the input image. And each convolution layer consists of several filters and square matrices that slide the image and register a weighted sum of pixel values at different locations. Each filter has different values and extracts different features from the input image. The output of the convolutional layer is a set of "object maps". When superimposed, convolutional layers can reveal a hierarchy of visual patterns. For example, lower layers will create feature maps for vertical and horizontal edges, corners, and other simple patterns. As you move deeper into the network, the layers will detect complex objects such as cars, houses, trees, and people.

A region-based convolutional neural network (R-CNN) consists of important key components. First, the region selector uses selective search, an algorithm that finds regions of pixels in an image that may represent objects, also called regions of interest (RoIs). The region selector generates about 2000 regions of interest per image. Then the RoIs are deformed to a given size and fed into a neural network. CNN processes each region separately, extracting features using a series of convolution operations. CNN uses connected layers to encode object maps into a one-dimensional vector of numerical values. Finally, the machine learning model of the classifier maps the encoded features obtained by the CNN to the original classes. The classifier has a separate output class for the background that matches anything that is not an object. Fast R-CNN is a new architecture that solved some of the problems of its predecessor. Fast R-CNN combines feature extraction and region selection into a single machine-learning model. Fast R-CNN takes an image and a set of RoIs and returns a list of bounding boxes and feature classes detected in the image.

One of the key innovations in Fast R-CNN was the region-of-interest pooling layer, an operation that takes feature maps of the CNN and the region-of-interest for an image and provides corresponding features for each region. This allowed Fast R-CNN to obtain features for all image regions in a single pass, unlike R-CNN, which processed each region separately. As a result, speed has been increased. You Only Look Once (YOLO) is a family of neural networks that have increased the speed and accuracy of object detection using deep learning. The main improvement of YOLO is the integration of the entire process of detection and classification of objects into a single network. Instead of extracting features and regions separately, YOLO does everything in one pass through a single network.

YOLO can perform object detection at the frame rate of streaming video and is suitable for applications requiring real-time logic output. There are already many Python libraries for neural network training, such as Keras and TensorFlow. The YOLOv4 architecture will be used to train the neural network. YOLOv4 is a real-time object detection model released in April 2020 that achieved state-of-the-art performance on the Common Objects in Context (COC) dataset. It works by breaking down the task of object detection into parts: regression to determine object positioning using bounding boxes, and classification to determine object class. This implementation of YoloV4 uses the Darknet platform.

However, training a neural network in laboratory conditions is problematic (Fig. 4.) as a powerful processor or GPU is needed for training. How much time it takes to train a neural network depends on the power of the processor. Neural network training can be done in Colab. Also, this service provides processors that are suitable for neural network training.



a) b)

**Fig. 4 (a) − areas of recognized objects in laboratory conditions, (b) − obstacle classification results**

To train a neural network, data is needed on which it will be trained. Such data can be downloaded from COC. A dataset for object detection, segmentation, key point detection, and signatures consisting of 328,000 images.

After downloading the dataset to Google Drive, next, establishing the connection between Google Drive and Colab is needed. The learning architecture can be cloned from the GitHub repository. We need variables in *.makefile to enable GPU and OpenCV. Next, we need to build the project using the !make command. For training, loading pre-prepared scales is needed. There are many different pre-prepared weights. In our case, the data will contain small and large objects, and the official repo advises using yolov4.conv.137 for initialization. The final dataset can be downloaded and unzipped for use as we need additional dataset description files. These files will need a *.txt file with the file paths in the train and validation partitions. A backup folder on Google Drive can be created to store the neural network weights there. Next, we need to change the configuration. Data description files must be created. We need to create two files on Google Drive: obj.data and obj.names. In obj.names we will write the names of the classes that will be provided. In obj.data, we will write information about classes and perform neural network training. This process takes quite a long time, the execution time depends on the number of classes and the size of the dataset. After the training, the weights of the neural network are saved in the backup folder.

### 4.2 Software for ground mobile robot's hardware

Researchers need to install a number of Python libraries on the microcomputer: opencv-python, opencv-contrib-python, GPS, numpy, pickle, imagezmq, geographiclib, googlemaps, imutils. Global variables used for navigation and connection to the controller using the Serial library need to be created. Next, a flow with the reading data function from modules is started. Now we start the cycle, which will end when the connection to the server takes place and the GPS module establishes communication with the satellites.

It is necessary to start the streams of transferring frames from the camera to the server and receiving data about objects from the server. To get the coordinates of the destination point of the route, the Google Maps API is used. To get an API key, we need to create a project in google cloud, go to the credentials tab and create a key (Fig. 5.).

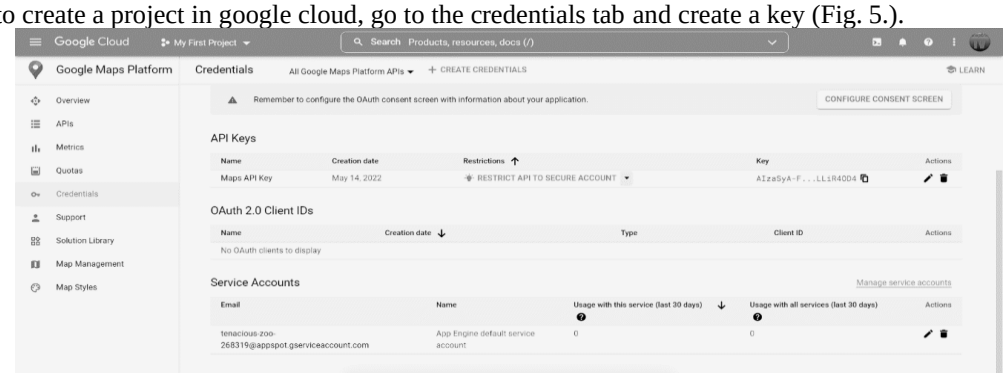



**Fig. 5 − obtaining a key to use the Google Maps API**

With the help of Google Maps, we lay out the route. First, the coordinates of the destination are obtained by its name. Then obtaining the device's location using coordinates from the GPS module. Google Maps functions help automatically calculate the route to the destination. As a result, all waypoints and information about them are available to the autonomous vehicle software.

### 4.3 Creation of the server part

The operation of a neural network is a very resource-consuming task. The Raspberry Pi processor will not be able to cope with such a large number of calculations. Therefore, it is necessary to create a server on which the calculations will be performed. The Raspberry Pi will send the images from the camera to the server, which will be processed by the neural network. The received data from the neural network will be sent to the Raspberry Pi. A number of python libraries will be required to create the server. The pip install command is used to install libraries. The configuration, neural network weights, and classes must be loaded. The imagezmq library is used for image transfer. Images can also be transferred via sockets, but this is not very efficient and the library will cope with this task better. It is necessary to start the imagezmq server and connect the configured neural network model. Then specify the destination, which is transmitted to the autonomous vehicle. Variables are created where the address of the autonomous vehicle will be stored. We start the server monitoring cycle. The main tasks will be performed in the body of the cycle.


**Fig. 6 − setting initial coordinates and available classes for obstacle recognition**

First, check the connection to the server. If there is no connection, then the main tasks are not performed and the next iteration of the cycle begins immediately. If there is a connection, the name of the destination is sent to the connected client, and code execution continues. Receiving images from the client part of the autonomous vehicle is in progress. We perform calculations with a neural network that records the found classes, the result is shown in Fig. 6., their probability, and location coordinates in the image.


**Fig. 7 − areas of recognized objects (a) in road traffic conditions**

If the object array is not empty, the received data is processed, and the processed data is added to the object array. The data is archived using the pickle library so that it can be sent and then sent to the client side of the autonomous vehicle. If it fails to send, the server disconnects the autonomous vehicle.

**4.1. Development of the software for driving through the crossroads**.

A decision tree is one of the methods that will allow classification and grouping for machine learning. The decision tree tries to reproduce the logic of solving the problem by an expert.

In the study, decision trees will be used to classify the movement manoeuvres of an unmanned ground vehicle. Decision trees for driving operations are easy to understand and interpret because they are visualized. Decision trees require little or no data preparation. Other methods often require normalizing the data, creating dummy variables, and removing null values. However, the software module with decision tree functions does not support missing values. The cost of using a tree or predicting data is equal to the logarithm of the number of data points used to train the tree. Both numerical and categorical data can be processed. However, the scikit-learn implementation does not currently support categorical variables. Other methods usually specialize in the analysis of data sets with one type of variable and can solve problems with multiple outputs and use white-box models. If a certain situation is observed in the model, then the decision-making processes can be easily explained using Boolean logic. On the other hand, in "black box" models such as artificial neural networks, the results of which can be difficult to interpret. The model should be able to be tested using statistical tests. This allows for the robustness of the model to be taken into account and works well with the actual models from which the data were derived, even if their assumptions were slightly violated.

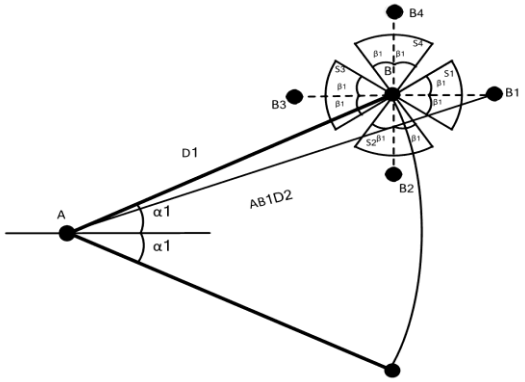



**Fig. 8 − Geometric representation of traffic sector analysis based on LIDAR data**

In vehicle control software development, decision tree theory is used to develop a model that can make decisions based on traffic data at intersections. Input data contains information about the state of roads, pedestrians, traffic lights, public transport stops, and road signs.

In Fig. 8, the unmanned ground vehicle is at point A, and the obstacle detected by LIDAR is at point B. After measuring the distance from point A to the conditional point, which is an obstacle, we automatically assume that the obstacle can move in one of four directions , which correspond to the $S_1$, $S_2$, $S_3$ and $S_4$ sectors. Sectors are built relative to the hypothetical directions of the obstacle's movement. These hypothetical vectors are denoted as $B\mathrm{B}_1$, $\mathrm{B}B_2$, $B\mathrm{B}_3$ and $\mathrm{B}B_4$. The sectors take into account the corresponding angles $\beta_1$ displayed symmetrically relative to the axis of the vector $\mathrm{B}B_i$.

At the next moment in time, the LIDAR takes another measurement and fixes the obstacle. Accordingly, the robot

fixes the speed and direction of movement of the obstacle. Suppose that the obstacle point has moved to the position of point $B_1$, so we calculate the distance between points A and $B_1$, which will be called $AB_1D_1$.

The probable intersection area formula (1) of the obstacle sector with the robot movement sector will be described as follows.

$$S_A \cap S_1 = D_{S_AS_1}, S_A \cap S_2 = D_{S_AS_2}, \\ S_A \cap S_3 = D_{S_AS_3}, S_A \cap S_4 = D_{S_AS_4} \quad (1)$$

We can calculate the cross-section of the sectors using the following algorithm.

Let the circles have radii r_1 and r_2 and their centres be separated by a distance d. If the circles intersect at two points, then the radical line is the line passing through the points of intersection. If not, then any two circles which cut each original circle twice should be drawn. Draw lines through each pair of points of intersection of each circle. The line connecting their two points of intersection is then the radical line.

Given two circles formula (2) with trilinear equations:

$$(l\alpha + m\beta + n\gamma)(\alpha\alpha + b\beta + c\gamma) + k(a\beta\gamma + b\gamma\alpha + c\alpha\gamma) = 0$$

$$(l'\alpha + m'\beta + n'\gamma)(\alpha\alpha + b\beta + c\gamma) + k'(a\beta\gamma + b\gamma\alpha + c\alpha\gamma) = 0\,, \quad (2)$$

their radical line formula (3) has equation

$$(k'l - kl')\alpha + (k'm - km')\beta + (k'n - kn')\gamma = 0 \quad (3)$$

The radical line is located at distances formula (4,5):

$$d_1 = \frac{d^2 + r_1^2 - r_2^2}{2d} \quad (4)$$

$$d_2 = \frac{d^2 + r_2^2 - r_1^2}{2d} \quad (5)$$

along the line formula (6) of centres from $C_1$ and $C_2$, respectively, where

$$d \equiv d_1 - d_2 \quad (6)$$

The radical line of any two polar circles formula (7,8) is the altitude from the third vertex.

$$\|D\| \equiv \iint_D 1dx\,dy, \quad (7)$$

$$D \equiv \{(x,y) \in R^2 : x^2 + y^2 \le 1, |y|\, cos\, cos\, \frac{\alpha}{2} \le x\, sin\, sin\, \frac{\alpha}{2}\,, (x - x_C)^2 + (y - y_C)^2 \le r^2\,,$$

$$|(y - y_C)\ cos\, cos\, \gamma - (x - x_C)\ sin\, sin\, \gamma\,|\, cos\, cos\, \frac{\beta}{2} \le ((x - x_C)\ cos\, cos\, \gamma + (y - y_C)\, sin\, sin\, \gamma)$$
$$sin\, sin\, \frac{\beta}{2}\} \quad (8)$$

with $x_C$, $y_C \in R$ , $r \ge 0$ and $0 \le \alpha, \beta, \gamma \le 2\pi$ known parameters.

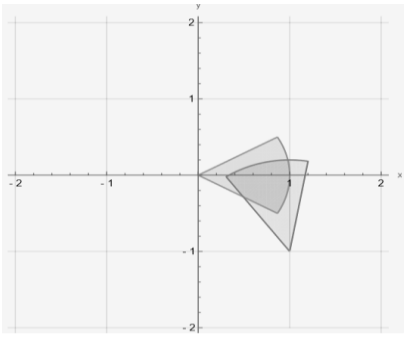


**Fig. 9 − visualization of the intersection of sectors during the calculation of their area**

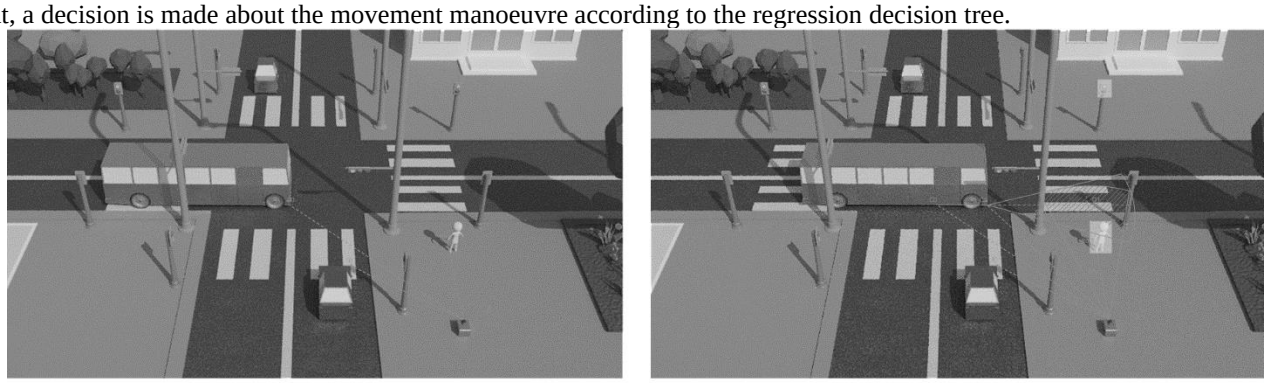
Having analysed the process of calculating the areas of intersections of sectors, we will proceed to the next step - calculating the probability of the collision itself $P(S_i)$, which we will calculate according to the following formula (9).

Where $D_{S_AS_i}$ is the calculated area of the likely intersection of sectors, and SA is the area of the robot movement

sector.

$$P(S_i) = \sum_{i=1}^{n=4} \frac{D_{S_A S_i}}{S_A} \quad (9)$$

Next, we will consider the situation at the intersection simulated in 3D. In Fig. 10 (a) shows that the unmanned ground vehicle finds the points of possible obstacles such as the extreme point $J_0$ of the green car, the extreme point $C_0$ of the bus, and also the extreme point of the person $P_0$, which is in front of the robot itself. At the next moment in time Fig. 10 (b) the LIDAR installed on the robot again finds the distances to these obstacles and conducts an analysis. New points $J_1$, $C_1$ and $P_1$ are being built. Points $J_1$ *and* $P_1$ are in the same place as the initial ones, from which the robot, based on regression decision trees, concludes that the objects are stationary, but the position of the point $C_0$ has shifted to the position of the point $C_1$. After that, the robot builds movement sectors and calculates the intersection areas of the movement sectors of obstacles for these same objects and determines the probability of collision with obstacles. After that, a decision is made about the movement manoeuvre according to the regression decision tree.



a) b)

**Fig. 10 − a simulated situation in 3D at the intersection**

Decision trees are a machine learning method that allows you to create decision models from data. The basis of this approach is the construction of a tree, where each branch corresponds to a possible solution and each leaf is the final result. Decision trees are used in many fields such as finance, medicine, commerce, and transportation.

A visual representation of what the decision tree looks like for the robot movement data set at the intersection. Each of these rectangles represents a certain decision that will be performed by the robot movement algorithm depending on the situation. Each solution looks at several variables, checks whether the specified variables acquire boolean values. If a certain variable is true, then the scenario from the right branch is chosen. At the second level, the decision tree begins to analyse the current situation.

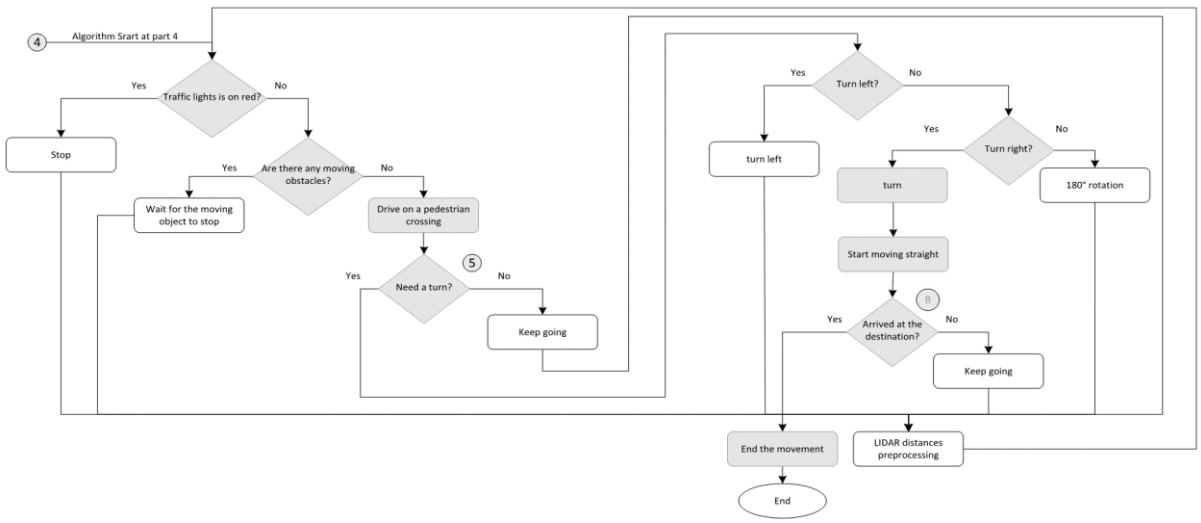



**Fig. 11 − the diagram of the part of the algorithm is built on the basis of the decision tree of the intersection**

An analysis of the environment in relation to moving objects on the path of the robot is performed. In the absence of moving objects, an analysis of the transition to the execution of a specific action of the next circuit block is performed in the decision tree. Next, the tree, based on the robot's navigation, considers the option of movement: straight, back, left or right. The algorithm predicts which path the robot will take. Having chosen the best direction option, the robot begins to move along it. Next, the location of the robot is compared with the destination. If the points match, the algorithm goes into the environment scanning state. The main goal for the robot is to reach the destination in the minimum time while

using the shortest route, if possible, in order to spend less energy resources on the trip.

Driving vehicles on the city's road network is a difficult task. The movement of vehicles using decision trees and demonstrated by the algorithm Fig. 11, which requires high accuracy and speed in making decisions on the road. To achieve this goal, software was developed using the theory of decision trees in machine learning.

The construction of the decision tree of the program takes place in several stages. First, the main branch is determined, which corresponds to the possible decisions regarding the movement of the vehicle. The next step is to determine the different possible options for the development of the situation on each branch of the tree for traffic at the intersection. For this, data from the robot's sensors, as well as information about road conditions and potential obstacles on the road, are used. Each variant of the development of the event has its own weighting factor, which reflects the probability of its occurrence, namely the probability of collision with an obstacle.

Based on traffic data at the intersection with the help of a machine learning algorithm, a model was developed that can independently make decisions about the further movement of the robot based on the current situation on the road. The model is based on the decision tree created in the previous step.

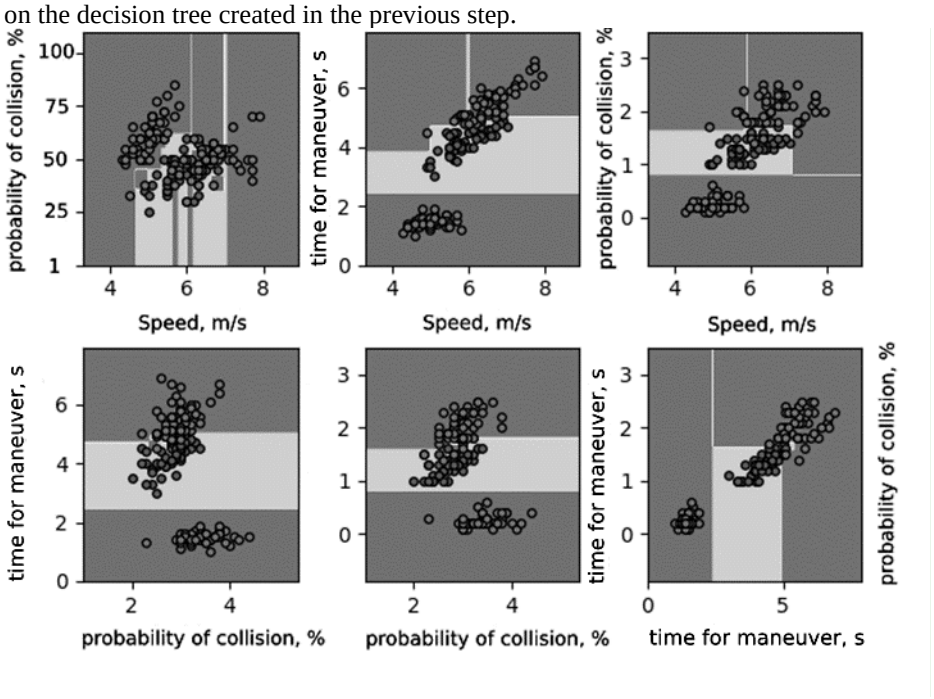



**Fig. 12 − decision surface of decision trees trained on pairs of features**

The diagram in Fig. 12 shows the decision surface of decision trees trained on the data set of the movement of an unmanned vehicle. For pairs of collision probability, manoeuvre time, and speed, the decision tree learns decision boundaries composed of combinations of simple threshold rules derived from training samples using combined manoeuvres. The class of combined manoeuvres covers the combination of different types of manoeuvres to achieve more complex movement tasks. Manoeuvre types can include a sequence of rectilinear, rotational, and curvilinear manoeuvres to reach a specific location or to perform a specific action.

To support the operation of the software, a system was developed that constantly monitors the road situation and interacts with other road users, providing additional information for the robot to make decisions.

Thus, the development of vehicle traffic software for the urban road network is based on the application of decision tree theory in machine learning. This makes it possible to create a system that can independently make decisions based on the current situation on the road, thereby ensuring safety and efficiency of traffic.

## V. CONCLUSIONS

In this work, the necessary components for creating an autonomous navigation system were analysed and selected. Many of the required navigation modules with more accurate measurements have a higher price, so devices with sufficiently high accuracy and a satisfactory price were chosen.

The main task of the robot is to find a way to the goal, as well as bypass obstacles. For this, the necessary components were selected. A microcontroller and a microprocessor are used to control the robot. A microcontroller is needed to control the various modules and engines, and a processor is needed to calculate the route. Special modules were selected for robot navigation. A LiDAR system uses a laser to find the distance to objects, while ultrasonic sensors use high-frequency sound pulses. These sensors can detect obstacles. A GPS module is used to find the location. All these components allow you to create an autonomous navigation system.

It was determined which modules are necessary for a system with autonomous navigation. It was determined how each module works separately and how it interacts with the entire system.

The principle of operation of drivers and engines is defined. Two models of drivers and two models of engines were considered. They differ in capacity and price. Therefore, one driver is used for research and development, and the second

can already be used for cargo transportation.

A schematic of the device with all connected sensors, a controller and a microcomputer was developed. The principle of operation of the entire system with autonomous navigation was developed.

For the long operation of the system with autonomous navigation, the power system was calculated, namely the capacity of the required battery. Since the engines consume the most energy, a battery with a capacity of at least 180 Ah is required. The software was written for the Arduino to control the various modules and exchange data with the microcomputer. The Python programming language was used for the main calculations. A description of the used functions and software libraries is provided.

A neural network was created and used for object detection. For neural network calculations, a server part was created that receives frames from the device's camera and returns data about detected objects to the client. Software for the calculator was developed. With its help, the device receives data from sensors and also builds a route to the target. The Google Maps API was used to map the route.

Decision trees have been used to model vehicle traffic processes at intersections, where decisions must be based on multiple factors and take into account collision probabilities, manoeuvre times, and speeds of alternative traffic situations. Decision trees made it possible to take into account not only objective criteria but also subjective parameters of combined manoeuvres, which provides a more reasonable process of movement prediction. Due to their structuredness and ability to model complex systems, decision trees help solve intersection traffic problems by systematically considering options, performing risk analysis and assessment, and choosing the optimal course of action for safely crossing an intersection with an unmanned vehicle. Decision trees in complex traffic situations with 12 combined manoeuvres take into account 82.4% of the variation of the unmanned vehicle movement at the intersection to avoid collisions. At the same time, the time to travel through the intersection increases by no more than 13–20 seconds to the destination.

Regression decision trees have several advantages in the context of analysis and prediction. In addition, their use requires minimal data preparation, as they do not require normalization, creation of dummy variables, or deletion of missing values. Regression trees can handle both numerical and categorical data. Compared to other methods, regression trees have the advantage of easy interpretability, comprehensibility of the model, the possibility of checking the model using statistical tests and checking the reliability of decision-making.

**VI. REFERENCES**

1. Hirz, M., & Walzel, B. (2018). Sensor and object recognition technologies for self-driving cars. Computer-Aided Design and Applications, 15(4) https://doi.org/10.1080/16864360.2017.1419638
2. Min, K., & Choi, J. (2012). Vehicle positioning technology using infra-based laser scanner sensors for autonomous driving service. Lecture Notes in Electrical Engineering, 114 LNEE. https://doi.org/10.1007/978-94-007-2792-2_48
3. Szeliski, R. (2021). Computer Vision: Algorithms and Applications 2nd Edition. Springer. https://doi.org/10.1016/j.arcontrol.2022.11.001
4. Watzenig, D., & Horn, M. (2016). Introduction to automated driving. In Automated Driving: Safer and More Efficient Future Driving. https://doi.org/10.1007/978-3-319-31895-0_1
5. Zhou, X., Weber, C., & Wermter, S. (2017). Robot localization and orientation detection based on place cells and head-direction cells. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10613 LNCS. https://doi.org/10.1007/978-3-319-68600-4_17
6. Uçar, A., Demir, Y., & Güzeliş, C. (2017). Object recognition and detection with deep learning for autonomous driving applications. Simulation, 93(9). https://doi.org/10.1177/0037549717709932
7. Meyer, G., & Beiker, S. (2016). Road Vehicle Automation - Reducing Conflict Between Vulnerable Road Users and Automated Vehicles. Lecture Notes in Mobility Road Vehicle Automation, August. https://doi.org/10.1007/978-3-319-19078-5
8. Shim, V. A., Tian, B., Yuan, M., Tang, H., & Li, H. (2014). Direction-driven navigation using cognitive map for mobile robots. IEEE International Conference on Intelligent Robots and Systems. https://doi.org/10.1109/IROS.2014.6942923
9. Clement, P., Veledar, O., Könczöl, C., Danzinger, H., Posch, M., Eichberger, A., & Macher, G. (2022). Enhancing Acceptance and Trust in Automated Driving through Virtual Experience on a Driving Simulator. Energies, 15(3). https://doi.org/10.3390/en15030781
10. Salimpour Kasebi, S., Seyedarabi, H., & Musevi Niya, J. (2021). Hybrid navigation based on GPS data and SIFT-based place recognition using Biologically-inspired SLAM. ICCKE 2021 - 11th International Conference on Computer Engineering and Knowledge. https://doi.org/10.1109/ICCKE54056.2021.9721522
11. Shved A. V. , Davydenko Y. O. (2022). Outlier detection technique for heterogeneous data using trimmed-mean robust estimators. Radio Electronics, Computer Science, Control. (3), 50. https://doi.org/10.15588/1607-3274-2022-3-5

Отримана в редакції 01.11.2023. Прийнята до друку 04.12.2023. Received 01 November 2023. Approved 12 December 2023. Available in Internet 03 January 2024