Crane Movement Modulator for Web Application of Electric Steel Melting Shop Mnemonic Scheme with Crane Movement Visualisation System

Rauza R. Abdulveleeva
Mathematics and Science dept.
Novotroitsky branch of the federal state autonomous educational institution of higher education "National Research Technological University MISIS" Novotroisk, Russia
rashitovna-2011@mail.ru

Viktor A. Korsakov
Novotroitsky branch of the federal state autonomous educational institution of higher education "National Research Technological University MISIS" Novotroisk, Russia
prettytoddlergtm@gmail.com

Ildar R. Abdulveleev
Power supply of industrial enterprpises dept.
Nosov Mangnitogorsk State Technical University
Magnitogorsk, Russia
i.abdulveleev@magtu.ru

2024 International Ural Conference on Electrical Power Engineering (UralCon) | 979-8-3503-8602-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/URALCON62137.2024.10718976

***Abstract*—The paper proposes a method of modulating crane movement in electric steel melting shop for web application of electric steel melting shop mnemonic scheme with the system of visualisation of cranes and steel ladles movement. The relevance of mnemonic scheme creation is caused by the problem of low transparency of cranes operation for the dispatcher, leading to downtime at the units and corresponding violations of steel production technology. Modulator is a Django application developed using noSQL DBMS Redis. With the help of the crane motion modulator, developers have an opportunity to debug the work in the web application without the need to install hardware at the stage of construction and testing of the web application. The file structure and web page of the modulator, the structure of JSON data passed to Redis are presented. This approach allows to increase the transparency and efficiency of crane operation, to increase the speed of decision making on the management of production operations in the shop floor within the shift.**

Keywords—modulating crane movement, mnemonic scheme, RFID, Drag-and-Drop, JSON, Redis

I. INTRODUCTION

Visualisation plays a key role in modern industrial manufacturing, providing tools to better manage and optimise production processes. A challenge faced by many businesses is that production data often remains siloed, unstructured and unavailable for real-time analysis. This leads to a lack of transparency in processes, increased time and costs, as well as lower productivity and product quality.

In this context, visualisation becomes a hot topic, as it allows data on production processes to be presented in a visual form, which helps personnel better understand the current state of production and make prompt, more informed decisions [1, 2].

For example, Novolipetsk Steel (NLMK) faced the challenge of visualising the processes taking place in the BOF shop to reduce deviations of the actual process from the standard melting processing time in order to save production resources. To solve this problem, NMLK, together with the team of Data-Centre Automation LLC, developed a converter shop visualisation system with the ability to monitor the movement of ladles, the status of melts in them, as well as the status of steel ladles and cranes in real time. As a result of the system implementation, the economic effect is 4% savings in electricity, electrodes and aluminium rod.

Ural Steel JSC faced a similar problem in the electric steelmaking shop (ESSP). According to the ESSPP Half Year Report, from July 2022 to November 2022, an average of 1,204 minutes of waiting time for melting were recorded in the casting span of the electric steelmaking shop, which resulted in a corresponding violation of technology, and over 18 hours of downtime at the HMP were identified due to the absence of steel ladle and crane occupancy. It is assumed that up to 20 per cent of the downtime was due to low transparency of crane operations.

In the light of this problem the mnemonic scheme of ESPPP with the system of visualisation of cranes and steel ladles movement was developed.

Purpose of the mnemonic scheme:

- visual control of the situation in the ESPPP in terms of cranes and steel ladles movement in real time;
- saving and accumulation of information in the database on the movement of cranes and steel ladles, implementation of main and secondary operations by cranes;
- integration of readings from the main production units (GMP, MCP, UHS, CCM) with information on current movements of cranes and steel ladles.

Problems to be solved:

- reduction of unit downtime due to crane occupancy or unavailability of steel ladles;
- increased transparency and efficiency of crane operations;
- increased speed of decision-making on the management of production operations within the shift;
- control of steel ladle location at any time.

II. DESCRIPTION OF THE APPLICATION

In the area of out-of-furnace treatment technological processes of metal transfer in steel ladles from unit to unit are

carried out by means of supporting overhead travelling cranes No. 7, No. 8, No. 9 and No. 10 (Figure 1).

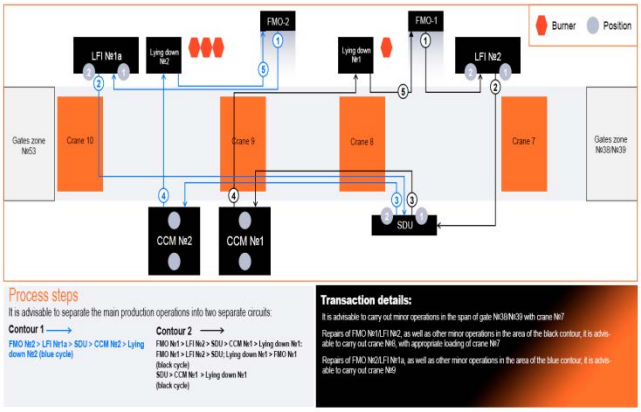

Fig. 1. Stages of the crane movement process

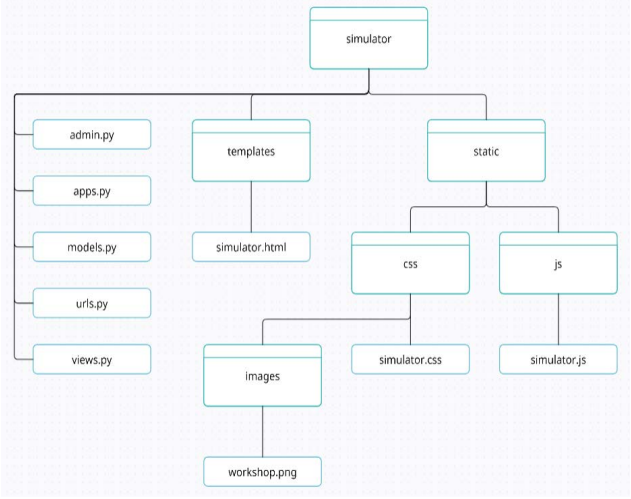

Fig. 2. File structure of the crane motion modulator

The sequence of technological operations in the specified span is as follows: GMP - UFP - UWS - CCM (Figure 1).

To ensure control over the movement of cranes, in the course of the literature review it was decided to use RFID tags, because, in comparison with Wi-Fi modules, GPS modules, laser range finders, they are suitable for work in aggressive conditions of the workshop. The work of RFID tags will not be affected by vibration, high dustiness of the air of the working area and high temperature [3–7].

It is assumed that the tags will be located near the crane tracks, this is due to the fact that they will be far from places where they may be damaged or fail. The fact that the temperature at their location is significantly lower than the temperature of the working area is also taken into account. Industrial high-temperature marks can withstand temperatures up to three hundred degrees.

If the tags are placed at a certain frequency, they will form a coordinate system along which the cranes will move. The coordinate system provides convenience in determining the location of the crane in the workshop. This was taken into account in the development of the mnemonic diagram for the purpose of the most accurate visualization.

In the absence of the possibility to install the necessary equipment in the workshop, it was decided to develop a debugging site - a crane movement modulator.

The mnemonic scheme is a web application developed using Django, an open-source, high-level Python web framework that promotes clean and pragmatic design [8–12].

Within the specifics of the Django framework, the crane motion modulator is one of two applications. The file structure of the application is shown in Figure 2.

The application is located in a separate directory called ‘simulator’. It contains the standard Django framework python files ‘admin.py’, ‘apps.py’, ‘models.py’, ‘urls.py’ and ‘views.py’, which are necessary for the application to work, and two more directories ‘templates’ and ‘static’. In the 'admin.py' file, the settings for the Django Admin Panel are stored. In the 'apps.py' file, the application's configuration is defined. The 'models.py' file describes the entities used in the application. The 'urls.py' file contains the URL address used for the application's page. Finally, the 'views.py' file configures the application's page. The ‘templates’ directory is needed to store templates of the application web pages, but since the modulator is a single page, only one template ‘simulator.html’ is located here. The ‘static’ directory is needed to store static files: images, CSS files and JavaScript files. The file ‘workshop.png’ contains an image of the workshop drawing. The file ‘simulator.css’ describes the rules for displaying page elements. The file ‘simulator.js’ implements the functionality of the crane movement modulator.

III. Operation of the Crane Movement Modulator

On the page with the modulator there is a drawing of the ESPC with the key units involved in the metal melting process, crane controls and the cranes themselves, which move along the drawing. The Crane Modulator page is shown in Figure 3.

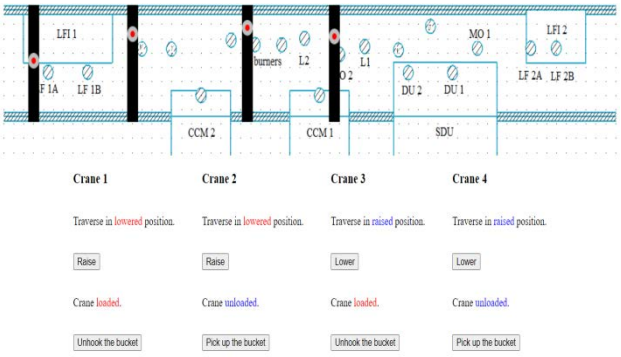

Fig. 3. Crane motion modulator

The cranes are moved using Drag-and-Drop, a way of operating the interface elements [13–15].

The Drag-and-Drop functionality is implemented using JavaScript on the client-side. The crane movement area is represented by a canvas. Geometric shapes, such as circles and rectangles, are drawn on the canvas. The page is continuously

waiting for user actions, such as mouse button clicks and mouse movement.

When the user clicks the mouse button, the browser handles this event and retrieves the current mouse position, stores it, and checks if the mouse is within any of the shapes on the canvas. If the mouse is inside a shape, the user can then move that shape by moving the mouse. The other shapes remain stationary.

As the mouse is moved, the browser handles this event, calculates the distance the mouse has moved since the button was clicked, erases all the shapes on the canvas, and redraws them with their new calculated positions.

After the mouse button is released, the positions of the shapes are saved. To move another shape, the process must be repeated.

Since the movement of the cranes in the drawing is very fast, to simulate the movement of the cranes, a red circle is created, which will follow the crane at the approximate speed of the crane in the shop, updating its position once per second (red point in Figure 4). And it is the coordinates of the red circle that will be transmitted to the mnemonic diagram.

Also for more detailed modulation, the position of the traverse and the bucket load of the cranes must be taken into account. This is done by means of the controls. This data is also transmitted to the mnemonic. On the mnemonic screen, the crane picture will change depending on the position of the crosshead and the ladle load of the crane. Since the functionality of the crane motion modulator, including the collection and formation of information about the positions of the cranes, is implemented on the client-side, there is a need to transfer the information to the server for further processing and use.

The method of data transmission is the JSON text format of data exchange [16]. The use of the JSON format is determined by its simplicity of use, lightweight nature, highest performance, and widespread support in the Python and JavaScript programming languages. Browser every second forms JSON-string with current information about the cranes and transmits it to the server using AJAX POST request [17-19].

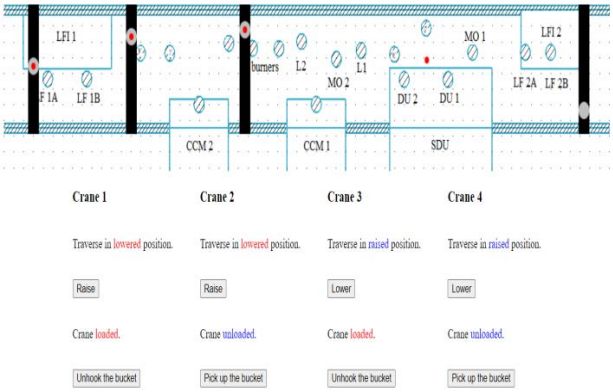

Fig. 4. Crane movement


```
{
    "cranes_pos":
        {
            "Crane_7": {
                "position_x": 7,
                "position_y": 4,
                "with_ladle": false,
                "is_upper": true
            },
            "Crane_8": {
                "position_x": 13,
                "position_y": 7,
                "with_ladle": true,
                "is_upper": true
            },
            "Crane_9": {
                "position_x": 45,
                "position_y": 2,
                "with_ladle": false,
                "is_upper": false
            },
            "Crane_10": {
                "position_x": 60,
                "position_y": 2,
                "with_ladle": false,
                "is_upper": true
            },
        },
}
```


Fig. 5. Structure of the transmitted JSON string

AJAX uses asynchronous data transfer, which eliminates the need to refresh the entire page each time to send another request to the server. POST is a standard method for sending data in the HTTP protocol, in which data is sent in the body of the request. Parameters define the URL and request method, data processing settings, the data to be transferred, and functions to be executed upon success or failure of the request. The function for sending an AJAX POST request is shown in the Figure 6.

The structure of the transmitted JSON string has three nesting levels. The first level is the key ‘cranes_pos’, the value of which is an array of keys. The second level is the keys ‘Crane_7’, ‘Crane_8’, ‘Crane_9’, ‘Crane_10’, the values of which are the keys with the crane state variables. The third level is the keys ‘position_x’ and ‘position_y’, which store the variables of the crane coordinates along the horizontal and vertical axes, the key ‘with_ladle’, the value of which is a variable with information about the crane load with a bucket, and the key ‘is_upper’, the value of which is a variable with information about the position of the crosshead. The structure of the transmitted JSON-string is shown in Figure 5.

After the server has received data from the browser, it is written to the Redis server. Being a non-relational database management system and storing data in the server's RAM, Redis provides high performance [20-22]. The data in Redis is stored for 10 seconds, but is updated to current data every second after the POST request is processed.

The function for processing a POST request by the server is shown in Figure 7. The function accepts the request, gets a JSON string from it, writes it to the Redis server, and returns an object with the visualization result.

Then in the cranes page view of the mnemonic scheme, the Redis server is accessed and a JSON string is received, the

received data is processed and the cranes are displayed on the mnemonic scheme according to it.

```
$.ajax({
    url: '',
    type: 'POST',
    processData: false,
    contentType: false,
    data: formdata,
    headers: {'X-CSRFToken': getCookie('csrftoken')},
    success: function(){
        console.log('Ajax post success');
    },
    error: function(response){
        console.log(response);
    }
});
```

Fig. 6. Function for sending AJAX POST request

Since the ESPC drawing used in the modulator is inconvenient and difficult to read, it was decided to modify it by stretching it vertically and compressing it horizontally, as well as replacing the representations of the units with more visual ones. After the changes, the mnemonic scheme, with which the dispatcher will interact, has the form shown in Figure 7.

```
class SimulatorView(TemplateView, RedisCacheMixin):

    template_name = 'simulator/simulator.html'

    def post(self, request, *args, **kwargs):

        data: dict = json.loads(request.POST.get('cranes_pos'))

        SimulatorView.set_key_redis_json( key_name: 'cranes_pos:1', data, ttl: 60)
        return render(request, self.template_name)
```

Fig. 7. POST request processing function

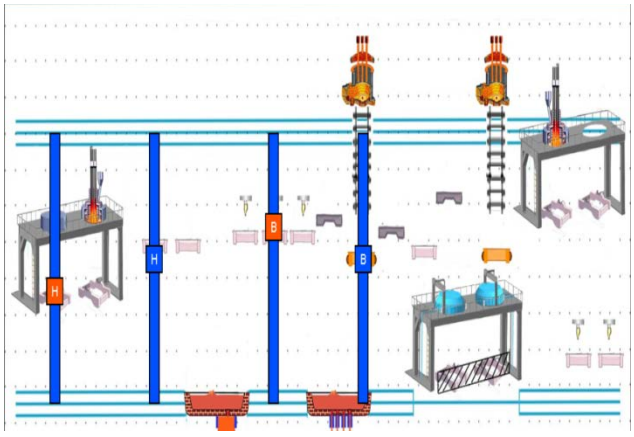

Fig. 8. Display of crane positions on the mnemonic diagram

Figure 9 shows the appearance of the mnemonic diagram after crane №4 has moved from the position shown in Figure 3 to the position shown in Figure 4. The image of crane №4 shown in Figure 8 will gradually move to the right and then down, in accordance with the movement of the red dot (Figure 4), until it reaches the position shown in Figure 9.

The function for displaying cranes on the mnemonic diagram is shown in Figure 10. The function receives information about the cranes and determines which carriage images will be used for each crane. The function then creates image objects and displays them on the mnemonic diagram.

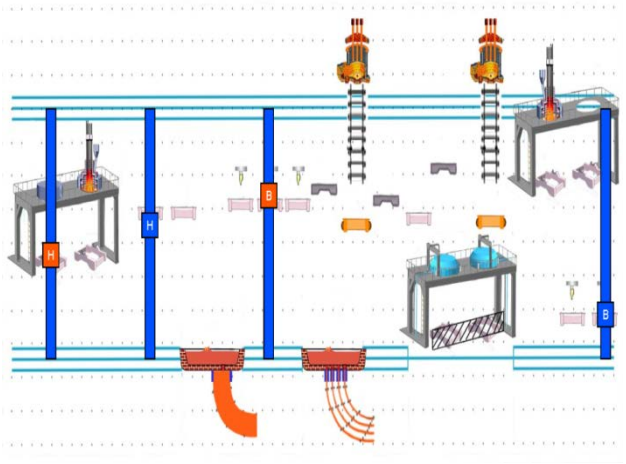

Fig. 9. Result of crane movement

```
function draw_crane(crane, crane_info, cranes_info) {
    let crane_body = cranes_info[`${crane}`];
    if (!crane_info['is_upper']){
        if (crane_info['is_ladle']){
            var ladle = cranes_info['Ladle_d_w'];
        } else if (!crane_info['is_ladle']){
            var ladle = cranes_info['Ladle_d_wo'];
        }
    } else if (crane_info['is_upper']){
        if  (crane_info['is_ladle']){
            var ladle = cranes_info['Ladle_u_w'];
        } else if (!crane_info['is_ladle']){
            var ladle = cranes_info['Ladle_u_wo'];
        }
    }
    let crane_body_x0 = crane_info.x * x_point + x_indent - crane_body['size_x']/2;
    let crane_body_y0 = y_indent;
    let crane_body_photo = new Image();
    crane_body_photo.src = crane_body['photo'];
    canvas_context.drawImage(crane_body_photo, crane_body_x0, crane_body_y0);
    let ladle_x0 = crane_info.x * x_point + x_indent - ladle['size_x']/2;
    let ladle_y0 = y_indent + crane_info.y * y_point + 60;
    let ladle_photo = new Image();
    ladle_photo.src = ladle['photo'];
    canvas_context.drawImage(ladle_photo, ladle_x0, ladle_y0);
};
```

Fig. 10. Function of displaying cranes on the mnemonic diagram

Thus, the algorithm of the crane motion modulator in the form of an event driven process (EPC) model is shown in Figure 11.

The implementation of the mnemonic diagram is an important step in the digitalization of production and the subsequent implementation of advanced analytics models:

- models for retrospective analysis of crane movement trajectories to create an optimal route for their movements;

- models for creating production schedules and optimizing the operation of the EAF taking into account the load in real time;
- models for tracking for crane management.

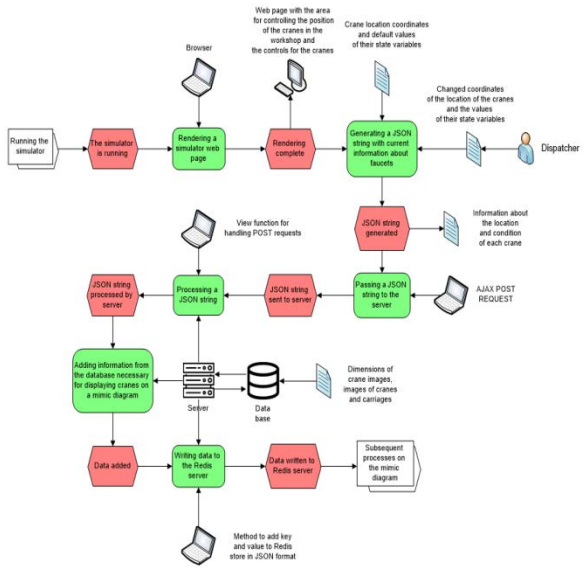

Fig. 11. Algorithm of crane motion modulator operation

IV. CONCLUSIONS

In the course of the work we have developed a crane motion modulator for the web application of electric steel melting shop mnemonic scheme with the system of visualisation of cranes and steel ladles motion. The proposed method of modulating the motion of cranes makes it possible to debug the work of the web application without the need to install equipment at the stage of construction and testing of the web application.

The developed mnemonic scheme with the system of visualisation of cranes and steel ladles movement is a promising solution to the problem of downtime at the units caused by low transparency of cranes operation for the dispatcher and is taken for consideration for introduction into production by the management of Ural Steel JSC.

REFERENCES

[1] K. Healy, Data Visualization: A Practical Introduction, Publisher, 2018, p. 240.
[2] H. Shoppa, Technology of Steel Production Processes, Publisher, 2015, p. 400.
[3] P. P. Petrov, “Development of a web-based visualization system for steel production using Django,” Industrial Informatics, 2022, vol. 14, no. 2, pp. 33–47.
[4] K. Finkenzeller, RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication, Publisher, 2010, p. 450.
[5] I. I. Ivanov, “Application of RFID technology in harsh industrial conditions: a case study of steel manufacturing,” Automation Journal, 2023, vol. 15, no. 3, pp. 45–59.
[6] Y. Zhang, et al. “Real-time crane monitoring system based on IoT and RFID technology,” IEEE Internet of Things Journal, vol. 8, no. 4, 2021, pp. 2951–2960.
[7] I. I. Ivanov and P. P. Petrov, “Application of RFID technologies for tracking crane movements in metallurgical production,” Automation in Industry, 2023, no. 2, pp. 15–22.
[8] R. R. Abdulveleyeva and V. Korsakov, “Comparative analysis of methods and means of visualisation of cranes movement of electric steelmaking shop of JSC ‘Ural Steel’ for use in web application,” Digital systems and models: theory and practice of design, development and application, Kazan, 10-11 April 2024, pp. 270–274
[9] P. P. Petrov, “Development of a web-based visualization system for steel production using Django,” Industrial Informatics, 2022, vol. 14, no. 2, pp. 33–47.
[10] S. S. Sidorov, “Development of web applications using Django for visualization of production processes,” Software Products and Systems, 2022, vol. 35, no. 3, pp. 452–461.
[11] Li X., et al., “A web-based visualization platform for industrial process monitoring using django and websocket,” IEEE Transactions on Industrial Informatics, vol. 17, no. 9, 2021, pp. 6379–6388.
[12] S. S. Sidorov, “Development of web applications using Django for visualization of production processes,” Software Products and Systems, 2022, vol. 35, vo. 3, pp. 452–461.
[13] S. Liu, et al., “A Drag-and-Drop based interactive data visualization framework for industrial big data,” IEEE Access, vol. 8, 2020, pp. 13734–13745.
[14] S. S. Sidorov, “Implementation of Drag-and-Drop interface for industrial process simulation,” Control Systems and Information Technologies, 2021, vol. 12, no. 1, pp. 101–115.
[15] A. A. Kuznetsov and V. V. Smirnov, “Use of Drag-and-Drop technology in interfaces of industrial control systems,” Bulletin of Computer and Information Technologies, 2021, no. 6, pp. 20–28.
[16] J. Wang, et al., “An efficient data exchange method based on JSON for industrial internet of things,” IEEE Access, vol. 7, 2019, pp. 82328–82341.
[17] D. D. Sokolov, “Application of JSON and AJAX for real-time data exchange in industrial automation web applications,” Modern High Technologies, 2020, no. 8, pp. 59–64.
[18] K. K. Kuznetsov, “Utilizing JSON and AJAX for real-time data exchange in industrial applications,” Information Technologies and Systems, 2020, vol. 11, no. 4, pp. 123–137.
[19] R. R. Abdulveleeva and V. Kravchenko, “Vulnerability analytics of web-application mnemonic scheme of electric steel-smelting shop with visualisation system of cranes and steel ladles movement,” Digital systems and models: theory and practice of design, development and application, Kazan, 10-11 April 2024, pp. 275–279
[20] S. S. Smirnov, “Redis as a high-performance solution for real-time data processing in industrial applications,” Bulletin of Information Technologies, 2019, vol. 10, no. 2, pp. 88–102.
[21] H. Chen, et al., “Redis-based Real-time data processing architecture for industrial internet of things,” IEEE Transactions on Industrial Informatics, vol. 16, no. 7, 2020, pp. 4769–4778.
[22] M. M. Volkov, “Redis as a high-performance solution for data processing in industrial visualization systems,” Scientific and Technical Journal of Information Technologies, Mechanics and Optics, 2019, vol. 19, no. 4, pp. 714–721.
[23] M. Vlasov, RFID: 1 technology – 1000 solutions: Practical examples of RFID use in various fields, Moscow: Alshina Publisher, 2014, p. 218.
[24] D. Forsyth and J. Pons., Computer vision. Modern approach, Trans. from English, Moscow: Williams Publishing House, 2004, p. 926.
[25] C. D. Knuckles and D. S. Yuen, Web applications: Concepts & real world design, Wiley, 2005, p. 833.
[26] A. C. Telea, Data visualization: Principles and practice, CRC Press, 2014, p.617.