Improved Graphical User Interface for Crawler Robot Servosila Engineer

1st Alexandra Dobrokvashina
Laboratory of Intelligent Robotic Systems (LIRS)
Intelligent Robotics Department
Institute of Information Technology and Intelligent Systems
Kazan Federal University
Kazan, Russia
dobrokvashina@it.kfu.ru

2nd Ramil Safin
Laboratory of Intelligent Robotic Systems (LIRS)
Intelligent Robotics Department
Institute of Information Technology and Intelligent Systems
Kazan Federal University
Kazan, Russia
safin.ramil@it.kfu.ru

3rd Yang Bai
Information Science and Engineering Department
Ritsumeikan University
Kyoto, Japan
yangbai@fc.ritsumei.ac.jp

4th Roman Lavrenov
Laboratory of Intelligent Robotic Systems (LIRS)
Intelligent Robotics Department
Institute of Information Technology and Intelligent Systems
Kazan Federal University
Kazan, Russia
lavrenov@it.kfu.ru

2021 International Siberian Conference on Control and Communications (SIBCON) | 978-1-7281-8504-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/SIBCON50419.2021.9438926

***Abstract*—Robots are widely used in many areas of modern life, and a teleoperated control plays a significant role in critical mission applications. For such applications Graphical User Interface (GUI) is one of the most popular interfaces for robot navigation and manipulation manual control. In this article, we present an improved GUI for a Russian crawler robot Servosila Engineer, which provides an operator with images from four cameras of the robot, a 3D view of a current configuration and an ability to control every joint of the robot. In addition to minor adjustments and extensions of a previous version of the GUI, a special attention is paid for modelling and control of the robot gripper.**

***Index Terms*—crawler robot, modeling, Servosila Engineer, GUI, graphical user interface, 3D model, controller, video stream, QT**

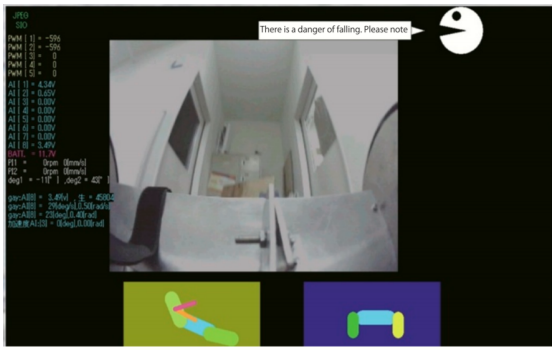

Fig. 1. Operator screen with support information from "Scott 1" [11].

I. Introduction

Today robots could be found in a different part of our life. Robots learning to drive cars [1], they clean our houses, work on manufactures [3], help surgeons to do complicated surgeries in more safe and easy ways [2]. They are used in places and areas where humans are forbidden to be, such as dangerous manufactures or radioactive zones. Manipulators present productivity as no human alive could. They could work through all day and night seven days a week, easy lift cars and their parts.

Different types of robots are used for various purposes in different degrees of autonomous [4], [5]. Some of them are used in fully autonomous mode, for example, robots house cleaners. On the other hand, there are teleoperated robots. They are used in conditions of important tasks, where it is necessary to coordinate every movement manually according to the situation. For example, in surgeries or urban search and rescue tasks [6]. In such conditions, it is very important to have a stable connection to the robot and a comfortable way to communicate with it. Our goal is to make operator job maximally productive. That is why receiving data from sensors and cameras of the robot and sending commands to move should be easy to process.

One of the most popular ways to interact with robots is a computer or mobile app with a graphical user interface (GUI). It is one of the cheapest and the easiest solutions which suits most of the cases. There are many examples of such interfaces, such in Fig. 1-3.

These examples are used for different tasks. Some of them show only information about the current configuration of the robot. Others also send a notification if there are any problems, e.g. danger of falling, as in Fig. 1. Another doesn't receive any information from the robot but allows controlling robot movement, for instance, Fig. 2. Third combines all of the previous statements, so that operator receives the ability not

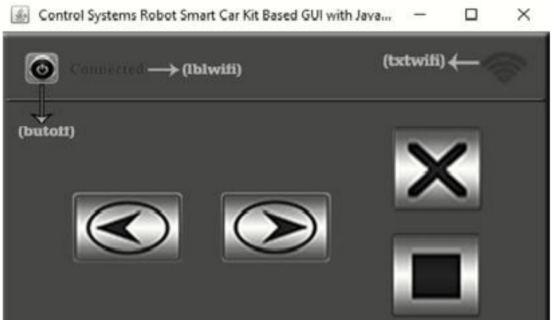

Fig. 2. GUI for operator of 4WD Smart Car Robot [10].

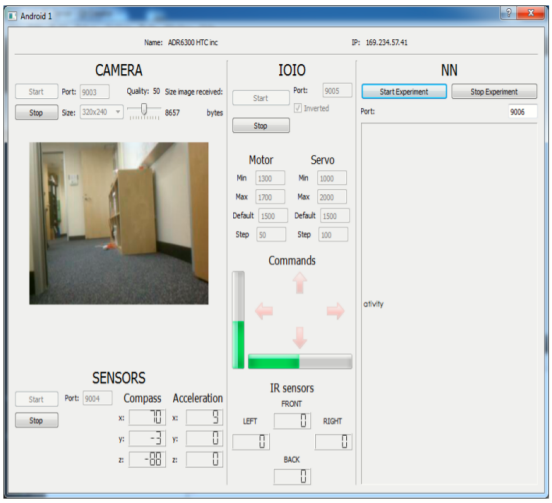

Fig. 3. GUI for the operator of Android-based Car Robot [9].

only to move the robot but to see everything that the robot can sense with its sensors and cameras e.g. Fig. 3.

This article presents the next step of work [8] on a graphical user interface for the Russian crawler robot Servosila Engineer. It already gives the ability to control the robot by sending data with commands for every joint. Also, it shows the current robot configuration via a 3D model and data from built-in cameras.

II. Crawler robot Servosila Engineer

A. System setup

Mobile crawler-type robot Servosila Engineer (Fig. 4) was made by Russian company Servosila [12]. It was designed to be exploited in dangerous for human environments. Having an insulated and waterproof shell it could be used in situations with different weather conditions e.g. in extreme circumstances such as floods or even firefighting. Due to built-in sensors and cameras Servosila Engineer could be used rather in autonomous mode, so in teleoperation mode. It has an IMU sensor and 4 cameras: stereo pair, one camera with zoom and one camera in the back. Also head of the robot contains a lantern. It could be useful while having operation during the night or in half-light rooms. More than that Servosila Engineer is equipped with laser scan on moving platform. It could measure distance not only for positive obstacles but by

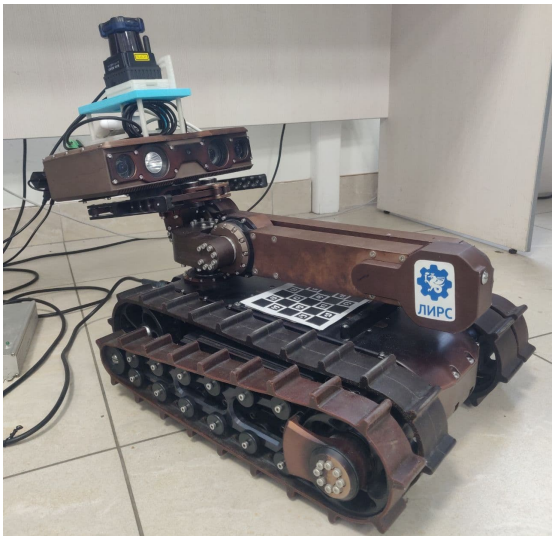

Fig. 4. Servosila Engineer crawler-type mobile robot.

changing angle it also could give information about negative obstacles [13].

The modular construction of the robots shows them as a good education platform. With the help of cameras, Servosila Engineer could be applied in tasks of computer vision. A manipulator with a gripper allows interacting with the environment around it. And tracks with flippers give good traversability on uneven terrains [14]. All of these features give a lot of possible ways to use this robot.

B. Previous GUI version

As already been mentioned this article continued the theme of the user interface for Servosila Engineer. A big amount of work was done already, such as controlling the robot via remote control packages [7]. Basic things in the graphical user interface also were done. These are sliders for controlling joints and tracks movements, the first version of the 3D view of the model of the robot. The 3D model shows its current configuration [8].

In this article, we continued this development and improved several elements of GUI such as the 3D model and its mobility. Also, we improved video widgets with pictures from cameras.

III. GUI improvements

A. Model improvement

The first version of the 3D model of the robot in our GUI has several differences from the real one. And that may become not only a visual issue but practical. That is because the size of the model was smaller than the present robot. It could cause some misspelling in the current configuration or possible workspace. That is the reason why changing the 3D model was one of the first tasks in this research.

The new model was created using the same meshes that were also used for simulation of the Servosila Engineer in the Gazebo environment. The final version of the model with comparing to the first version of the model shown in Fig. 6.

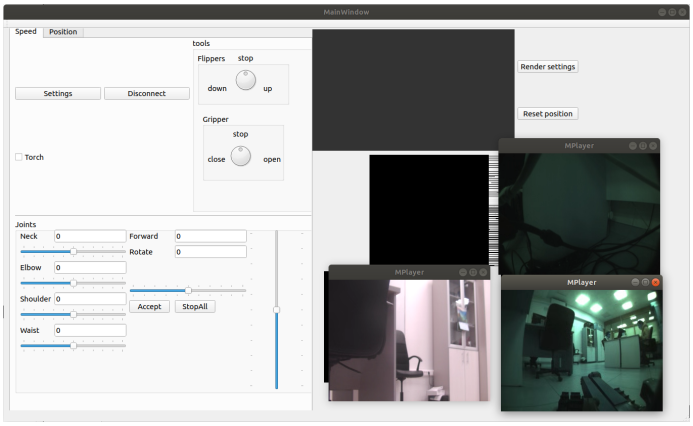

Fig. 5. Previous version of GUI for Servosila Engineer.

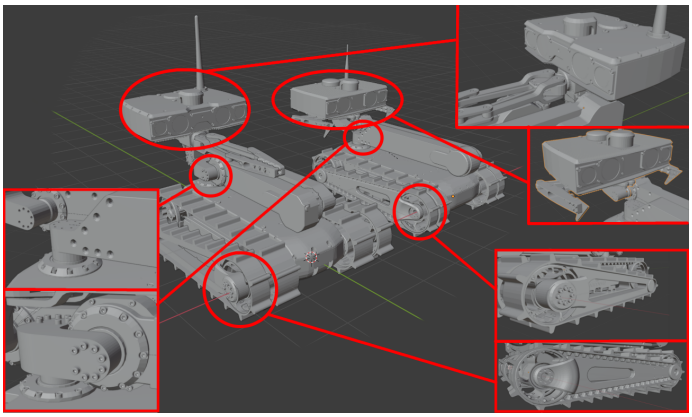

Fig. 6. Comparing old 3D model(right) for GUI and new(left).

The other problem that appears with the model was about the 3D view itself. The first version of GUI was made with Qt in Ubuntu 16 via a self-written OpenGL widget. After updating Ubuntu to 18 version there was found an issue that OpenGL in some cases shows nothing. To workaround that problem 3D view was replaced with the new widget which based on built-in Qt3D modules [15]. It presents a simple way to create a 3D scene with the model inside and by simple manipulations move any part of that model in the necessary position [16], [17] .

The created solution makes the program more universal because of fewer side modules inside it. And manipulation with the object on the 3D scene became much simpler than before.

B. Gripper movement

Another problem with the 3D model was the gripper. In the old model, it was connected with the model of the head of the robot so it couldn't move. So one of the tasks was to split the mesh of the head into several different parts to give the gripper ability to move.

Finally model of the head turned into seven parts. These are the head itself and three parts for each of two sides of the end-effector. To move that part simultaneously the way they move on the real robot it is necessary to move all six parts of the gripper in one moment. To open end-effector with some unknown angle β parts need to move according to the scheme illustrated in Fig. 7.

The next step was to make the gripper move the same way as on the robot. As already been mentioned created user interface provides operator information about the current state. For that purposes it use telemetry packet (Table I). These packets contain information about cameras and servo drives. Every motor sends information about itself with the structure of Motor Data (Table II). This information is used for moving the 3D model the same way as the Servosila Engineer itself.

TABLE I
SERVOSILA ENGINEER TELEMETRY PACKET.

Field	Type	Size
Frame type ID	**unit8**	**1 byte**
Tick number	**unit64**	**8 bytes**
Number of motors	**unit8**	**1 byte**
Motor data #0	**struct**	**24 bytes**
...	**...**	**...**
Motor data #9	**struct**	**24 bytes**
Not used	**-**	**25 bytes**

TABLE II
SERVOSILA ENGINEER MOTOR DATA STRUCTURE.

Field	Type	Size
Device ID	**unit8**	**1 byte**
Device state	**unit8**	**1 byte**
Operation mode	**unit8**	**1 byte**
Position	**unit32**	**4 bytes**
Speed	**int16**	**2 bytes**
Electric current (amperes)	**int16**	**2 bytes**
Status bits	**int16**	**2 bytes**
Position command	**unit32**	**4 bytes**
Speed commands	**int16**	**2 bytes**
Electric current command (amperes)	**int16**	**2 bytes**

The gripper also sends information about its position, and it presents as a good way to know about the current end-effector configuration. In this step, we faced the problem with the data. The parameter server on the robot sends bad data about gripper position. The position of the gripper on the server is measured with the wrong boundaries. They are

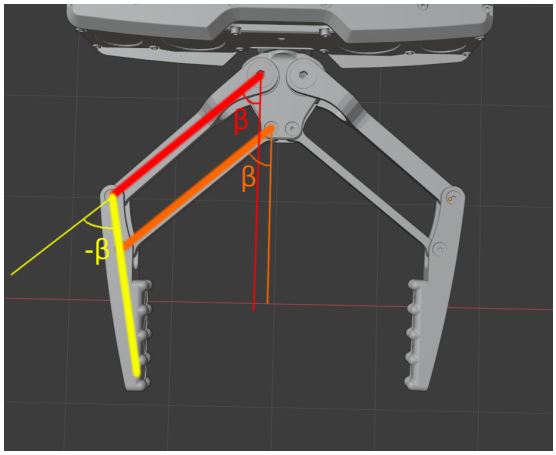

Fig. 7. Scheme of gripper movement by angle.

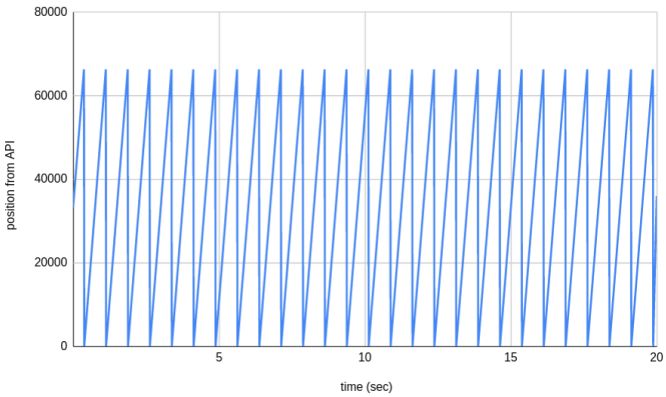

Fig. 8. Correlation of position of the gripper from time.

smaller than the real position interval. In a such situation information about positions of the end-effector through the time of moving between extreme points makes about twenty circles in the boundaries of the parameter server. Correlation of position from the time shown in Fig. 8. And there were several ways to solve this problem.

The first way is to count the current position of the gripper on the operators computer. If the position of the gripper is known from frame to frame the new position could be counted by addition or deduction from previous. So if the position steps over server boundaries we just compute the difference from the previous position to one bound and add the difference from the second bound to the new position. That also means that in the moment of launch gripper firstly needs to move to the start point, for example in a fully open state.

The described way is shown as a good math solution for the designated problem. But while trying this method we faced the next problem. Telemetry packages that were received by the operator were sent too rare. This causes jumping through server position intervals and losing data, which makes the gripper on the model desynchronize with the real one.

On the other hand, we have another possible solution that doesn't correlate with position data but uses time stamps. Now we count the current position of the gripper from the time of moving. Maximal time is the time needed for the gripper to move from the fully opened position to closed. And the current time is calculated from the time of sending the command to open or to close. Again as in the previous method in the beginning we need to move the end-effector in an extreme position as the start point. So based on the time of changing state we increase the angle if closing the gripper or decrease the angle if opening it. Now gripper is better synchronized with the real robot because of not appealing to the position data of the gripper. But this solution still could have issues.

Working on the parameter server and fixing the bug of the gripper position seems like a better solution, but this is a complicated task that could be exposed in another work.

C. Video streaming

As mentioned before Servosila Engineer has four onboard cameras: three in the front and one rear. Images from them sending via video server [18] with hardware encoding [19]. One of the abilities of our GUI is to show images from all cameras. Also, it gives the operator information about the robots surrounding. Part of the work has been already done in the previous version.

In the first version of the program, there were some problems with connecting views to the main window. Videos from the cameras are always opened in separate windows as in Fig. 9. Working with video stream carried out by side application - MPlayer [20], which is widely used for working with audio and video stream [21]. It can be launched in so-called "slave mode". In that mode, all of the commands are sending with the command line. In the program, we launch this player with several arguments to improve the quality of the picture and draw its picture as part of the main window for each of four streams. But still, pictures are not always shown properly.

To solve this problem we improved this widget, changed the window hierarchy so that every instance of player has its own instance of the widget with proper id. So the final view of the created graphical user interface with views from cameras is shown in Fig. 10 The same window in full-screen mode presented in Fig. 11.

IV. Conclusion

Creating a graphical user interface for the robots is a very complicated task that is including graphical part on the operator side, transfer of the data from robot to operator, and sending commands to the robot.

In this paper, we presented an improved graphical user interface for the crawler-type robot Servosila Engineer. We created a more realistic model, improved the 3D viewport using Qt3D modules, added a moving gripper to the model, and synchronize its position with the robot. We added some changes in the visualization of the video stream from the cameras of Servosila Engineer. Also, there were done little improvements in layout.

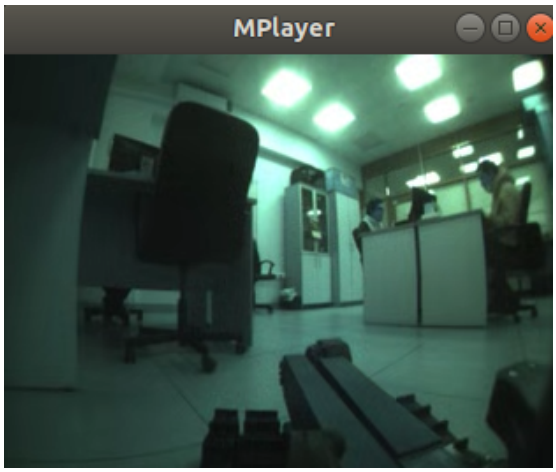

Fig. 9. Video from a camera in a separate window.

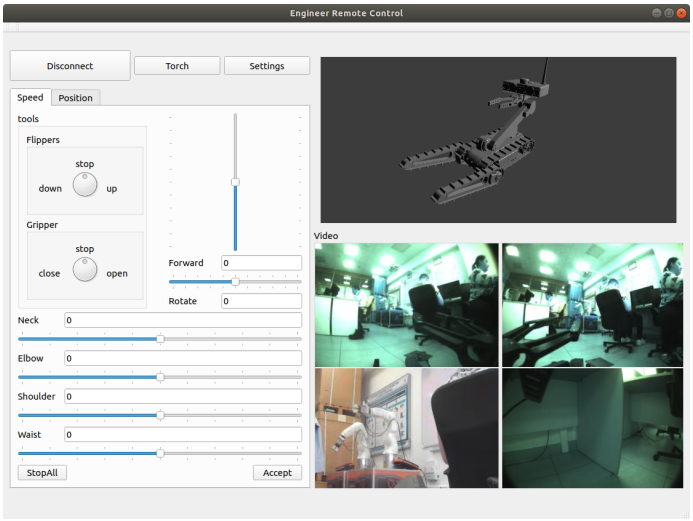

Fig. 10. New GUI with four streaming cameras view.

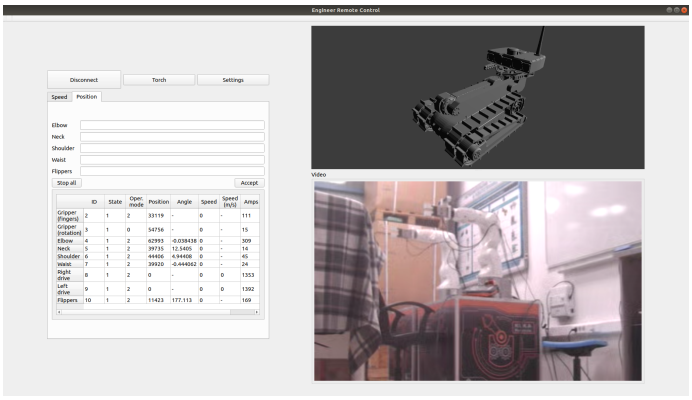

Fig. 11. New GUI in a full-screen mode of a single streaming camera view (selection by an operator).

ACKNOWLEDGMENT

This work was supported by the Russian Foundation for Basic Research (RFBR), project ID 19-58-70002. The third author acknowledges the support of the Japan Science and Technology Agency, the JST Strategic International Collaborative Research Program, Project No. 18065977.

REFERENCES

[1] A. Paolillo, P. Gergondet, A. Cherubini, M. Vendittelli and A. Kheddar, Autonomous car driving by a humanoid robot, Journal of Field Robotics 35(2),pp. 169-186, 2018.
[2] J. Isogaki, S. Haruta, M. Man-i, K. Suda, Y. Kawamura, F. Yoshimura and K. Taniguchi, Robot-assisted surgery for gastric cancer: experience at our institute, Pathobiology 78(6), pp. 328-333, 2011.
[3] T. Martinec, J. Mlýnek and M. Petrů, Calculation of the robot trajectory for the optimum directional orientation of fibre placement in the manufacture of composite profile frames, Robotics and Computer-Integrated Manufacturing 35, pp. 42-54, 2015.
[4] Pecka, M., Zimmermann, K. and Svoboda, T. Fast simulation of vehicles with non-deformable tracks. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 6414-6419). IEEE.
[5] Shimchik, I., Sagitov, A., Afanasyev, I., Matsuno, F. and Magid, E. Golf cart prototype development and navigation simulation using ROS and Gazebo. In MATEC Web of Conferences, Vol. 75, p. 09005, 2016. EDP Sciences.
[6] J. Wang, M. Lewis, and J. Gennari, USAR: A game based simulation for teleoperation, In Proceedings of the Human Factors and Ergonomics Society Annual Meeting Vol. 47 No. 3, pp. 493-497, 2003.
[7] I. Mavrin, R. Lavrenov, M. Svinin, S. Sorokin, and E. Magid, Remote control library and GUI development for Russian crawler robot Servosila Engineer, MATEC Web of Conferences, vol. 161, 2018, pp. 03016.
[8] I. Mavrin, R. Lavrenov, and E. Magid, Development of a Graphical User Interface for a Crawler Mobile Robot Servosila Engineer, 11th International Conference on Developments in eSystems Engineering (DeSE), IEEE, 2018.
[9] Oros N., Krichmar JL., Smartphone based robotics: Powerful, flexible and inexpensive robots for hobbyists, educators, students and researchers, IEEE Robotics & Automation Magazine, 2013.
[10] R. F. Siregar, R. Syahputra and M. Y. Mustar, Human-Robot Interaction Based GUI, Journal of Electrical Technology UMY, 1(1), pp. 10-19, 2017.
[11] S. Suzuki, S. Hasegawa and M. Okugawa Remote control system of disaster response robot with passive sub-crawlers considering falling down avoidance, Robomech J 1 (20), 2014.
[12] Servosila official site. https://www.servosila.com/en/index.html
[13] J. Larson, and M. Trived, Lidar based off-road negative obstacle detection and analysis, 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 192-197, 2011.
[14] E. Magid and T. Tsubouchi, Static Balance for Rescue Robot Navigation: Translation Motion Discretization Issue within Random Step Environment, International Conference on Informatics in Control, Automation and Robotics (ICINCO), Portugal, 2010, pp 415-422.
[15] Qt 3D Overview. https://doc.qt.io/qt-5/qt3d-overview.html
[16] S. Harmer, Qt 3D: a data-driven renderer for mortals, ACM SIGGRAPH 2016 Talks, 2016, pp. 1-2.
[17] G. Lazar, and R. Penea, Mastering Qt 5. Packt Publishing Ltd, 2016, pp.196-222.
[18] R. Safin, R. Lavrenov, T. Tsoy, M. Svinin and E. Magid, Real-Time Video Server Implementation for a Mobile Robot, 11th International Conference on Developments in eSystems Engineering (DeSE), pp. 180-185, 2018.
[19] Safin, R., Garipova, E., Lavrenov, R., Li, H., Svinin, M. and Magid, E. Hardware and software video encoding comparison. In 2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE) (pp. 924-929). IEEE.
[20] M. Team, Mplayer – the movie player. https://mplayerhq.hu
[21] R. Dantas, C. Exton, and A. Le Gear, Comparing network performance of mobile voip solutions, 6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp. 43-50, 2018.