Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 100 (2015) 1475 – 1484

Procedia Engineering

www.elsevier.com/locate/procedia

25th DAAAM International Symposium on Intelligent Manufacturing and Automation, DAAAM 2014

Control of the Mobile Robots with ROS in Robotics Courses

Khassanov Alisher, Krupenkin Alexander, Borgul Alexandr*

St. Petersburg National Research University of Information Technologies, Mechanics & Optics, 197101, 49 Kronverkskiy av., Saint Petersburg, Russian Federation

Abstract

The paper describes implementation of mobile robots programming process with Robot Operating System (ROS) in student robotics courses. ROS provides different tools for data analysis, facilities of multiple robots and their sensors, teleoperation devices interaction thereby targeting engineering education. An example with the multiagent interaction between agent-evader and agent-pursuer were taken as the basic navigational task. The computed behavior of the virtual agents were successfully transferred to the quadcopters, Lego Mindstorms NXT based and Robotino robots. Diverse experimental tests were conducted using the algorithms on virtual agents and robotic platforms.

Peer-review under responsibility of DAAAM International Vienna

Keywords: ROS; robotics; education; multiagent system; remote control

1. Introduction

We should think about the future of robotics and control science in close connection with the present of control education. Advanced educational tools are used to make study more illustrative and therefore more attractive for the young specialists. Current technological achievements allow us to develop high-quality courses in the modern control theory and robotics courses. For example, mechatronic and robotic research equipment is very popular now. Such experimental setups represent compact high-tech tools, which are the great substitution for the traditional in control workshops computer simulation software like Matlab and Simulink. From educational point of view, it has several advantages. First of all, with such laboratory equipment students have opportunity to intuitively understand

* Corresponding author. Tel.: +7-951-664-47-88.
E-mail address: borgulalexandt@gmail.com, 148586@niuitmo.ru

Peer-review under responsibility of DAAAM International Vienna
doi:10.1016/j.proeng.2015.01.519

the basic control theory principles and feel how formulas really work in practice. On the other hand, students can obtain additional knowledge in the related areas such as robotics, computer science, information theory, programming, and electrical and circuit engineering. Moreover, young engineer can better understand the critical value of the real technical systems constraints during experimental validation. Thereby, using mechatronic and robotic laboratory setups, we can provide students possibility to follow the robotics and control system development process from the formulas to the implementation, excluding the risk to damage expensive equipment. We use this equipment primarily in the bachelor student's workshops as the first touch on the more advanced master courses [1]. At the same time we do understand that the deeper we dive in the more we can explore. Hence, the described systems can be used even in the doctoral research.

But to be able to control different robotic and mechatronic setups we should provide appropriate programming and developing tools to the students. Commonly used Matlab and Simulink allow making researches and controlling robots only having proper described mathematical model. And there are only few robots with the free spread models. Though every robot can be described and programmed with common programming languages such as C/C++ or Python, but it's not comfortable to learn and use different languages for every part of the complex project or the new model of the robot. Meanwhile there are diversity of the robots should be studied during the robotics course for good preparation. And ROS (Robot Operating System) looks like the best solution for this problem.

Standards developed by Willow Garage and realized as ROS infrastructure allow combining in unified development environment both the software and hardware components. It is completely comfortable to use in case of students society where they can be targeted on engineering problems. Students should only know the programming language and ROS infrastructure to be able to use diverse instrumental set for different educational and practical tasks. ROS is the worldwide open source project with the huge database of available robots' models, sensors, and guidebooks, supported programming languages, simulation environments. There are a lot of advantages described in [2], [3] and [4]. That's why we implemented ROS in robotics courses at the Control Systems and Informatics department of the ITMO University.

Let us present how we solved two tasks and developed complex projects with the students during the course and studying process. You can find the description and step-by-step solution process of the remote control of the mobile robots via the Internet in the second part of the paper. And the solution of one of the most popular tasks about the multiagent system developing with navigation providing and task manager presented in the third part of the paper. The simplicity and convenience of using ROS instruments are shown in the both tasks. The practical experiments are conducted and presented as well.

2. Teleoperation interface gateway for ROS

One of the common tasks during the Robotics course is developing the robotic system with the remote control. There are some robots under ROS should be provided with the remote control via the Internet. Control of robots will be independently exercised from various places. It is necessary to provide high level of safety, flexibility and performance. There is a solution for this very task proposed and described earlier in our papers and similar papers [5]. There is a complex task, which allows students to study almost all the main parts of the ROS environment. Let us present more or less detailed example explanation.

2.1. Introduction to the task

Let's start from the discovering of available tools. At the moment there is a software providing possibility of interaction with the robot from the browser, it is called as ROSLIBJS (http://wiki.ros.org/roslibjs) and ROS Bridge Suite (http://wiki.ros.org/rosbridge_suite). ROSLIBJS uses Web Socket for connection with Bridge Suite that is carried out in namespace of ROS and can interact directly with nodes. In that case the solution of start of Bridge Suite on the separate server for circumvention of restrictions of NAT and connection of the robot to the server through a virtual private network looks logical. In our case OpenVPN is chosen as safe and cross-platform decision.

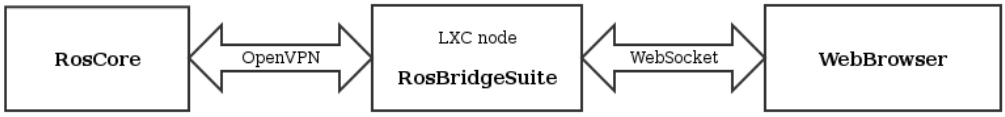


Fig. 1. Simple gateway.

As shown in fig. 1 the robot connects to the server through the virtual private network, with the server the web browser through WebSocket also interacts, such diagram provides high performance at the expense of a complete support from Bridge Suite event - the oriented approach and safety at the expense of data encryption on all transit. LinuX Containers LXC (http://linuxcontainers.org) are used here for Bridge Suite start in an isolated surrounding with limited resources, but without overhead. The approach described above can be expanded for independent control of a set of robots to start with several LXC-containers. However there are also some difficulties:

- Automatic service of containers (creation, start, stop)
- Traffic routing from the robot to the container and from the container to a web browser
- Authorization system for access to Bridge Suite from the robot and a web browser

2.2. Server side

Students study the main organizational structure and protocols at this step. The server part of the system provided by two large modules: Web server and system of containers. Binding of these two systems is carried out by means of a database.

We propose the Web server to be based on Yesod (http://www.yesodweb.com) framework and provides project business logic: authorization of users, creation, deleting and configuring of robots: adding and setup of plug-ins, loading of a configuration and OpenVPN keys. The set of scripts of easy-rsa (http://openvpn.net/easyrsa.html) is used for keys of the X509 standard developing. Let's skip the part of the database's model description used by a Web server to not make the paper too long.

The system gives opportunity to customize the control interface of the robot by adding of plugins, with the robot the set of the extensions possessing unique parameters for each robot communicates. The more detailed information about plugin system presented to the students provided further.

2.3. Containers

The system of containers is engaged in service tasks such as creation, start and stop of LXC containers, conjugation of containers to the OpenVPN channel, conjugation of the container to external WebSocket port. The part of the database models' description file used by a system of containers presented below as an example.

```
Node                    # Some server is a Node
    name Text
    address Text
Container               # LXC container
    robot RobotId
    node NodeId Maybe
    address Text Maybe  # Container address on virtual ethernet
    UniqueContainerRobot robot
NewContainer            # New container queue for Creator thread
    ink ContainerId
Connection              # Table of current connections for stats
```

```
container ContainerId
node NodeId
since String Maybe
vaddress String Maybe      # Virtual address
raddress String Maybe      # Remote address
sent Int Maybe             # Sent bytes
eceived Int Maybe          # Received bytes
```

The management system is partitioned by containers into two independent threads: Creator and Connector. Main objective of a Creator thread is creation of the new container. But it continuously monitors queue in a database and creates the new container in case of request appearance. The modified Ubuntu Core 12.04.3 version packed into SquashFS (http://squashfs.sourceforge.net) is used as core file system of the container. The system is mounted in the read-only mode. It makes possible to use the same root for an unlimited number of containers. Main objective of a Connector thread is monitoring of the new OpenVPN connections and their binding with appropriate containers from the both launches of the new connections as well as stops in case of connection comes to the end. Binding is carried out by gateway facilities of packets as a standard firewall of Linux.

2.4. Routing

Transits of packets both from the specific robot to the container and from the container to a web browser are presented in fig. 2 and fig. 3. Thin lines present data streams; network interfaces are drawn by squares and circles shows the software.

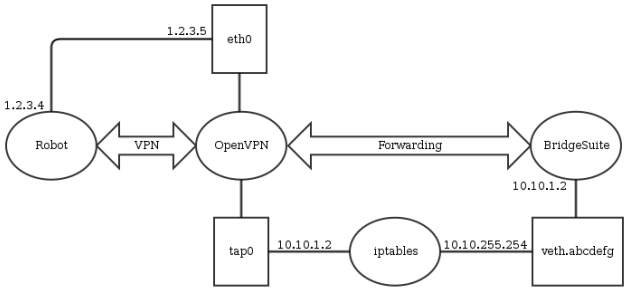


Fig. 2. Robot connection.

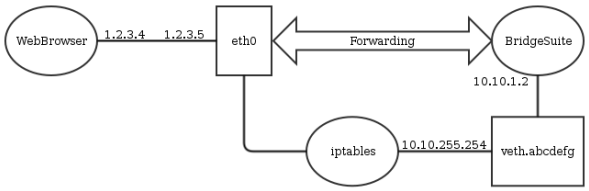


Fig. 3. Web browser connection.

2.5. Security

According to the modern trends it's very important to study and implement security solutions, especially in the complex systems using Internet. ROS provides good infrastructure for the developing of such kind of a solutions. In this case the high level of security is provided with the help of TLS encoding applied to all transit data and with the

isolated containers for Bridge Suite. It protects the software with the direct access from outside within the virtual surroundings.

2.6. Browser side

This part of the system is implemented in the JavaScript language using ROSLIBJS for representation of the abstraction of interaction with ROS nodes. In very common case, with the switched off plugins all the code in the system consists of instant connections creation of the container and pointer located on it in term of global variables which then all connect plugins and make them available for execution.

```
var ros = new ROSLIB.Ros(),host="46.47.232.219",port=7009,proto="ws";
ros.connect(proto+'://'+host+':'+port);
```

IP address of the container and port for the connection can be turned out from the database in case of dynamic generation of the page. Further extension of functionality is supposed to be performed at the expenses of the connection of unique set of plugins for each separate robot.

2.7. Conclusion of the part

As the result of this task performing students study not only how to use ROS but have the safe and productive system of remote control of several independent robots. This system presents full-fledged possibility of adding and deleting the robot, archive loading with settings on the local computer. The server part is fully implemented and is available for the check under the open source license at the address: https://github.com/akru/tigro-lxc. The web interface has only two plugins. There are ImageView and KeyboardTeleop. The prototype of TIGRO is located at the address http://tigro.akru.me:3000. There is a huge possibility and amount of available work for improving and developing in this system. Though the system itself represent fully functional model with a lot of advantages. So, there are a lot of tasks for the student developers in the Python, JavaScript and Haskell languages for extension of functionality, fixing of errors.

3. Developing of mobile robots multiagent system with ROS

An interest to the developing of multiagent systems significantly increases last decades. There are a lot of researches and papers on this topic with the different proposed approaches and algorithms. Developing of such kind of a system is one of the important complex tasks, which allows students to study simultaneously methods of modern control theory, mathematical modelling, system identification and programming languages. Let us present our approach to this task solving with the theoretical description and practical implementation.

Mobile robots is a class of devices appropriate for problems solving which require the presence of operation unit and sensing system in the difficult to access areas. In everyday life the main reason of this situations appearance is technogenic accident and catastrophe, elemental calamity (fast acquisition, search and delivery of assistance is requires), celestial bodies surface research, the hard labor in explosive atmosphere or different dangerous areas. So the tasks of trajectory searching and following in uncertain environment have a big practical validity. Complex decision of that task allows finding the optimal trajectory for all machines in formed circumstances. This task may be solved in a distributed manner with duty of few robots, which called agents and has ability to work together with some problems simultaneously. We based our research on the previous works such as [6].

Leader-follower formation control is applied as the basis for multiple wheeled robots and unmanned aerial vehicles (UAVs). This approach's difference is in a simple way of algorithms description. It makes easier understanding by the students and easier realization. But at same time it allows adding different optimization criterions. The control objective for the follower UAV is to track its leader at a desired- separation, angle of incidence, and a bearing by using an auxiliary velocity control.

3.1. Theoretical description

The technical challenge for the control designer is facing an autonomous collaborative operations system in real-time sensing, computing and communication requirements, environmental and operational uncertainty, hostile

threats and the emerging need for improved robots team autonomy and reliability. Let us introduce an approach for the coordinated control of multiple robots and formation control with the collision avoidance. The hierarchy connected with global situation awareness and team mission planning, local knowledge, and formation control and obstacle avoidance.

Let us consider some group of several interacting agents with dedicated functions. The formation control problem is viewed as an interaction of *n* pursuers and *n* evaders. Stability of the formation of vehicles is guaranteed if the vehicles can reach their destination within a specified time. Assume that the destination points are avoiding the vehicles in an optimal way. A two-vehicle example is shown to illustrate the approach. Vehicle model is simplified to point mass with acceleration limit for the simplicity. Collision avoidance is achieved by designing the value function so that it ensures that the two vehicles move away one from another when they come too close to each one. The problem of control algorithm developing is multiple autonomous vehicles should maintain a formation during traversing a desired path and avoiding inter-vehicle collision. It may be written as the formation control problem. Each individual robot in the team is considered as an agent with particular capabilities engaged in executing a portion of the mission. The primary task of a typical team of robots is to execute faithfully and reliably a critical mission while satisfying local survivability conditions. Let us consider the group of two agents *U1* and *U2* as an interaction between pursuer and evader. The next solutions are appropriate for the whole group of robots in the team as well. The obstacles for this system is static solid objects like trees or columns in the uncertain area. The sensors for obstacles detection are placed only on the base of the evader agent *U1*. It allows detecting the barrier on the distance *r*. Pursuer agent *U2* moves with the help of data achieved from the agent *U1* by wireless communications. Also agent *U2* should optimize the trajectory of agent's *U1* movement. This approach is convenient due the one agent resolves the tasks of obstacles avoiding for all the other agents. Other vehicles could solve different tasks and don't lose time and energy for impassable objects searching. The issue of the trajectories optimization is minimization of the cost function *f(x)* [7]:

$$f(x) = \max_i \left[\frac{\partial \omega}{\partial t}\Big|_{\omega=\omega_i}\right], \tag{1}$$

where the argument ω is controlled variable of the angular velocity of an agent at the moment t_i. In control system agent-pursuer should has a big part of the agent-evader's trajectory for the analysis. Control algorithm for robot movement described further. Agent *U1* equipped with the distance sensors. It moves with the collisions avoiding and sends the coordinates to the agent-pursuer. In case of obstacle detection agent *U1* sends the signal to agent *U2* and makes a maneuver. After the maneuver it sends another signal. Agent-pursuer receives the coordinates of the trajectory of the agent *U1* during the maneuver and after that executes path-smoothing algorithm and continues movement with the new coordinates. Evaluating flight path error shouldn't be above the range of vision of distance sensors during the new path. Let's consider the trajectory of the agent-evader movement as a sequence of *n* points with coordinates (x_i, y_i), where $i \in \{1..n\}$. Gradient descent with fixed step size is used for the cost function evaluating. Associate every pair of coordinates of the first agent trajectory (x_i, y_i) with the new pair of coordinates (x'_i, y'_i). The minimum of angular velocity achieves with the next minimization criterion:

$$((x'_i, y'_i) - (x_i, y_i))^2 \to min, \tag{2}$$

$$((x'_{i+1}, y'_{i+1}) - (x'_i, y'_i))^2 \to min, \tag{3}$$

Here (x_i, y_i) - coordinates of point of the required trajectory on the step *(i+1)*. Equation (2) minimizes the distance between the points on the initial trajectory and according point on the smoothed trajectory. Equation (3) minimizes the distance between two closest points on the desired trajectory. Let's provide for every criterion some weight of functions in range from 0 to 1. It means the correspondence level of initial trajectory to optimized trajectory for criterion (1). With weight of criterion (3) increasing the length of trajectory decreasing and degree of scaling becomes higher. Weight functions selects of thumb and complies with requirement $\alpha > \beta > 0$, where α -

minimization coefficient for (2) and β - weight of criterion (3). According to the (3) some incrementation should be added to the desirable point (x'_i, y'_i) on the trajectory in the direction of the next desirable point:

$$(x'_i, y'_i) \leftarrow (x'_i, y'_i) + \beta((x'_{i+1}, y'_{i+1}) + (x'_{i-1}, y'_{i-1}) - 2(x'_i, y'_i)) \quad (4)$$

The resulted gradient descent algorithm for our task:

$$\begin{cases} (x'_i, y'_i) \leftarrow (x'_i, y'_i) + \alpha((x_i, y_i) - (x'_i, y'_i)) \\ (x'_i, y'_i) \leftarrow (x'_i, y'_i) + \beta((x'_{i+1}, y'_{i+1}) + (x'_{i-1}, y'_{i-1}) - 2(x'_i, y'_i)) \\ \alpha > \beta > 0 \end{cases} \quad (5)$$

That coordinates transformation repeats until the difference between the value of the last element of array and desirable value becomes less than error margin. Fig. 4 shows the behavior model for group of agents in case of different obstacles positioning.

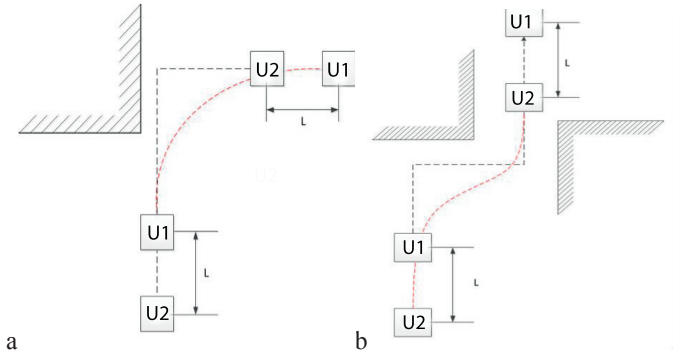


Fig. 4. Model of group behavior in case of: (a) – single obstacle, (b) – multiple obstacle.

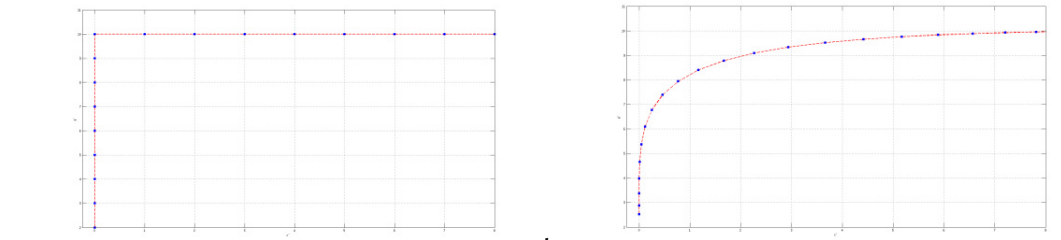
3.2. Realization in ROS

We propose the realization in ROS to the students right after the theoretical description of the system. Let us describe one of the tasks. We kindly advise to use the Haskell language for realization as the most convenient for this kind of tasks with a lot of advantages. Let us not to provide you with example cause it takes 30 rows of code. But it is the minimal available amount for this problem solving.

So, the global planner in Haskell was written without transformation it into the node. Describing the desirable behavior we have the next scenario:

- Leader agent starts moving on the fixed distance from the follower. Leader moves until it meets the obstacle.
- Leader sends the command about stop to the follower and starts turning.
- Leader moves fixed time after the turn while saving path points.
- Then it sends saved trajectory to the global planner. Global planner optimizes it according to the algorithm and then follower agent receives optimized trajectory.
- Local planner in the follower agent allows controlling velocity of the robot and traversing the path on the optimized trajectory.

The simple case of simulation plots illustrated in fig. 5.



a b

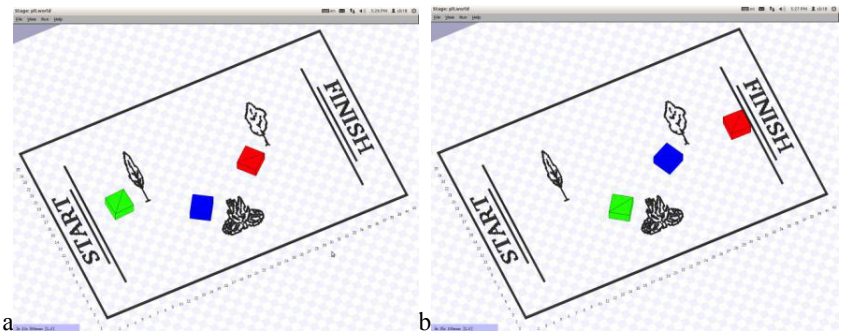
Fig. 5. Simulation plots of the trajectory: (a) – leader agent, (b) – follower agent.

Students supposed to realize the planning algorithm in Haskell and the node of the robot in Python. We have it but don't show on purpose. At the final student receives file with the two simple and comfortable for further applying steps. Step one is a step of algorithm from the previous part of the paper and second step is execution of the optimization algorithm.

Meanwhile we should remember that for the algorithm realization supposed *leaderpath[n+1] = leaderpath[n], leaderpath[n-1] = leaderpath[0]*. And all the points of the optimized path converge to *leader_path[n-1]* in case of increasing of the iterations number.

3.3. Simulation and practical implementation

ROS provides very well balanced simulation environment Stage [8]. It's useful tool to see how the calculated behavior of the robots will look like. And it's also very powerful tool cause you can see if you have mistaken somewhere. So, the next task in the course is to make simulation in Stage right after realization of the algorithm. Simulation screenshots of our algorithm with the 3 robots are shown in fig. 6.



a b

Fig. 6. Multiple virtual agents collision and obstacles avoiding process. (a) – time 1, (b) – time 2.

After the simulation if everything goes fine the algorithm can be easily ported on every single platform using ROS native supported platform drivers. The well-known platforms Lego Mindstorms NXT, quadcopters and Festo Robotino were taken as a basis for the practical experiments. The same algorithm from the Stage was uploaded in the both platforms via the wireless channel. Experiments were conducted in the laboratory of the Department of Control System and Informatics of the ITMO University. The task was to go through the set of obstacles and reach the finish line with the formation holding during the movement. The images of the experiment are shown in fig. 7.

Fig. 7. Screenshots of the experiment. (a) – Lego Mindstorms, (b) – Robotino, (c) – quadcopters.

The quadcopters and Robotino showed very good results according to their embedded navigational system and quality of the odometry sensors. And some additional tuning was required for the Lego Mindstorms NXT because of the equipment quality. In general all the computed virtual agents behavior were successfully realized with the real robots. All of them avoided all the obstacles and finished the distance. It can be very comfortable for students to do any changes and add some sensors or machine vision to this task and make it more complicated. Majority of sensors and platforms has native support in ROS.

Conclusion

Exploration of the new ways of control study is very important for efficient training of the next generation of engineers. New software and experimental tools that can be used in student's workshops and research projects are only one of the many directions to attend.

The problem of the students' preparation according to the modern tendencies and purposes are presented. And some examples with the practical validation are shown. We presented our approach to preparing students during the robotics course. It mainly based on the ROS step-by-step studying as elements of the complex tasks. It allows students to learn different programming languages, software tools, learn project work. Our developed algorithms can be easily uploaded in the real robots doesn't depend on model of the robot. We demonstrated one step-by-step approach to creation a teleoperation interface and the process of creation real-time multiagent system with the global planner. There is a unique solution with easy implementation in every system. The developed teleoperation interface can be used with every model as it written in the global level and separated in the node. It is an important decision allows using it in the both global and local navigational tasks. The simulation results for the proposed multiagent system match the theoretical calculations and used as laboratory work. Afterwards students often use this tool in there usual tasks. Both simulation and practical experiments were conducted. The results of theoretical and practical results are very close. It proves the stability of the solutions and advantages of using simulation software in educational process. Useful tools were created and shared for everybody according to the open source policy. Further we are planning to add more nodes and tools developed by our students, and ourselves and write a textbook based on our approach.

Acknowledgements

This work was partially financially supported by Government of Russian Federation, Grant 074-U01.

This work was supported by the Ministry of Education and Science of Russian Federation (Project 14.Z50.31.0031).

References

[1] Bobtsov, A.A., Kolyubin, S.A., Pyrkin, A.A., Borgul, A.S., Zimenko, K.A., Evgeniy, R.Y. Mechatronic and robotic setups for modern control theory workshops, (2012) IFAC Proceedings Volumes (IFAC-PapersOnline), 9 (PART 1), pp. 348-353.

[2] Ruiz, E., Acuña, R., Certad, N., Terrones, A., Cabrera, M.E. Development of a control platform for the mobile robot Roomba using ROS and a Kinect sensor, (2013) Proceedings - 2013 IEEE Latin American Robotics Symposium, LARS 2013, art. no. 6693270, pp. 55-60.

[3] Bayar, V., Akar, B., Yayan, U., Yavuz, H.S., Yazici, A. Fuzzy logic based design of classical behaviors for mobile robots in ROS middleware, (2014) INISTA 2014 - IEEE International Symposium on Innovations in Intelligent Systems and Applications, Proceedings, art. no. 6873613, pp. 162-169.

[4] Mendonça, R., Santana, P., Marques, F., Lourenço, A., Silva, J., Barata, J. Kelpie: A ROS-based multi-robot simulator for water surface and aerial vehicles, (2013) Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013, art. no. 6722374, pp. 3645-3650.

[5] Hold-Geoffroy, Y., Gardner, M.-A., Gagné, C., Latulippe, M., Giguére, P. Ros4mat: A matlab programming interface for remote operations of ROS-based robotic devices in an educational context, (2013) Proceedings - 2013 International Conference on Computer and Robot Vision, CRV 2013, art. no. 6569209, pp. 242-248.

[6] Bobtsov, A.A., Borgul, A.S. Multiagent aerial vehicles system for ecological monitoring, (2013) Proceedings of the 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems, IDAACS 2013, 2, art. no. 6663037, pp. 807-809.

[7] I. G. Chernorutsky Methods of decision making. Saint-Petersburg, BHV – Petersburg, 2005, 416 p.

[8] Pinho, T., Moreira, A.P., Boaventura-Cunha, J. Framework using ROS and SimTwo simulator for realistic test of mobile robot controllers (2015) Lecture Notes in Electrical Engineering, 321 LNEE, pp. 751-759.