# Mobile application for controlling multiple robots

Daniel Kiryanov
*Laboratory of Intelligent Robotic Systems (LIRS)*
*Intelligent Robotics Department*
*Institute of Information Technology and Intelligent Systems*
*Kazan Federal University*
Kazan, Russia
danielkiryanov@gmail.com

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

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

Mikhail Svinin
*Information Science and Engineering Department*
*College of Information Science and Engineering*
*Ritsumeikan University*
Kyoto, Japan
svinin@fc.ritsumei.ac.jp

Evgeni Magid
*Laboratory of Intelligent Robotic Systems (LIRS)*
*Intelligent Robotics Department*
*Institute of Information Technology and Intelligent Systems*
*Kazan Federal University*
Kazan, Russia
magid@it.kfu.ru

2021 IEEE 16th Conference on Industrial Electronics and Applications (ICIEA) | 978-1-6654-2248-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICIEA51954.2021.9516091

***Abstract*—Smartphones are becoming more and more high-performance devices every year, which allows to use them in new types of applications, including control and automation. This paper presents an Android OS based control tool that unifies a remote control of heterogeneous robotic systems. The current version of the tool controls the humanoid ROBOTIS OP2 robot, the wheeled differential drive TIAGo Base mobile robot, and the crawler Servosila Engineer mobile robot. Communication between a mobile device and the robots employs the RosJava library and the TCP/IP network protocol stack. The controls were implemented for compound kinematic systems with 3D robots' models displaying. Sensory data visualization and robots' movements controls were added to the application.**

***Index Terms*—Android OS, GUI, ROBOTIS OP2, TIAGo Base, Servosila Engineer**

## I. Introduction

The approaches to solving human-robot interaction problems are largely based on their areas of application in real life, from industry to education [1]. Different fields of robotics require different scenarios for the behavior of robots, however, several requirements overlap in many areas - these are safety and mobility. In the case of the latter, the requirement may apply not only to the controlled robot but also to the operator himself.

This task can be solved using conventional mobile devices capable of providing sufficient computing power. This is possible because their hardware is not inferior to a high-end computer from only 5 years ago and significantly exceeds the power of microcontrollers [2]. Given these characteristics as well as built-in communication protocols (Wi-Fi, Bluetooth, 3G/4G) and built-in sensors (GPS, camera, accelerometer, gyroscope, etc.), it makes sense to use the smartphone as an operation control unit [3].

Such a solution will be useful when interacting with a robot in conditions where operator mobility is required since control of the robot through a mobile device is more convenient during its operation in contrast to stationary computers or laptops. This approach will be useful in search and rescue operations when control in a hard-to-reach location is required. This kind of remote control can be applied in other areas as well [4], [5].

Such variability in the use of this software becomes possible if we add support for various programming interfaces to it. With the advent of ROS (Robotic Operating System), there was no need to create an implementation for each system separately. However, it remains to enable support for non-ROS systems that use the network protocol stack [6]. Today, such communication technologies between robots and mobile platforms already exist.

In our research, we develop an Android OS-based control tool for interacting with robots. The software implements mechanisms for displaying information received from sensors. For systems with a large number of links, visualization of a three-dimensional model is added with the ability to control individual servo drives of robots. This article presents devel-

978-1-6654-2248-2/21/$31.00 ©2021 IEEE

opments for the wheeled differential drive TIAGo Base mobile robot and the humanoid ROBOTIS OP2 robot as for systems with ROS and the crawler Servosila Engineer mobile robot as for a non-ROS system [7].

## II. Communication

This section describes the implementation of various specifications for networking between an Android device and robots. The RosJava library is used as the mechanism for connecting to ROS systems, and the TCP/IP network protocol stack is used as the mechanism for connecting to non-ROS systems [8].

### A. RosJava library

RosJava is a client library that is a pure Java implementation of ROS [9]. This technology enables work with ROS themes, services, and parameters. It includes the `rosjava_core` module - the main core of the library as well as the `android_core` module - for creating programs for Android. The `android_core` module includes some ROS application implementations such as displaying sensor data, robot motion control, etc.

### B. TCP/IP protocols

TCP/IP is a network model that describes the process of transferring digital data. Named after two main protocols of inter-networking:

1) IP (Internet Protocol), which provides routing of network packets,
2) TCP (Transfer Control Protocol), which ensures the establishment of a reliable connection between two machines and, in fact, data transfer, controlling the optimal size of the transmitted data packet and re-sending it in case of failure.

The TCP/IP family also has User Datagram Protocol (UDP) - which, unlike TCP, is a simple data transfer model without sequence and integrity checks.

## III. Robots

This section describes the robotic platforms involved in our research. Here are descriptions of the general-purpose robot TIAGo Base, the anthropomorphic robot ROBOTIS OP2 and the tracked robot Servosila Engineer.

### A. TIAGo Base

TIAGo Base is a mobile base designed for the minimum footprint and maximum payload used in logistics, industry, or research. The robot is propelled by two main motorized shock-absorbed wheels and four roller wheels. Its primary sensors are a laser rangefinder and an IMU, but add-ons include ultrasonic sensors and microphones. The TIAGo base is equipped with built-in speakers.

### B. ROBOTIS OP2

The robotic kit ROBOTIS OP2 (also called DARwIn-OP2 - Dynamic Anthropomorphic Robot with Intelligence Open Platform) is an open platform designed for educational and research activities in the field of robotics, the study of vision algorithms, object stabilization algorithms, control algorithms for robotic systems with complex kinematics. The robot is equipped with 20 servo modules, a programmable controller for operating with the robot's actuators, an onboard computer with Linux OS, a vision camera, a microphone and speaker, a 3-axis gyroscope, and a 3-axis accelerometer.

### C. Servosila Engineer

Servosila Engineer is a crawler robot designed for search and rescue operations in natural and man-made disasters in conditions where remote video image acquisition from a hazardous room is required [10]. The robot is equipped with a large set of built-in sensors - a laser scanner, stereo vision, a GPS/GLONASS receiver, an inertial sensor unit, a powerful camera with zoom and a thermal imager [11].

## IV. Mobile Application

At the stage of formulating the requirements for the application, the idea of creating improved software played an important role. During the design, we set out to create a unified robot control tool, despite the difference in remote control specifications [12]. This approach is impossible without proper code organization. This problem is solved through the use of an architectural pattern that will allow the development of individual modules when designing application components.

### A. Architecture

MVP (Model-View-Presenter) was chosen as a design pattern. The Figure 1 demonstrates how this architecture works. It shows how the development process is organized by dividing the code into three layers of abstraction. The first layer (View) uses the code for working with the UI (User Interface), the second layer (Presenter) is an intermediary between the first layer and the layer of business logic (Model). In our case, the business logic layer is the communication layer. This separation avoids code overload and provides little dependency on components. It's possible not to re-create the business logic layer when re-creating the view layer unlike other patterns when using the mobile app [13]. To summarize, MVP gives us the following benefits:

1) easily adding new network specifications by dividing components into areas of responsibility,
2) keeping the state of the connections open when recreating the view layer.

### B. Technologies

It was necessary to adapt the dependencies used to the requirements of the architecture before implementing the MVP design pattern. We have reorganized the work of RosJava library. The `android_core` library module is implemented

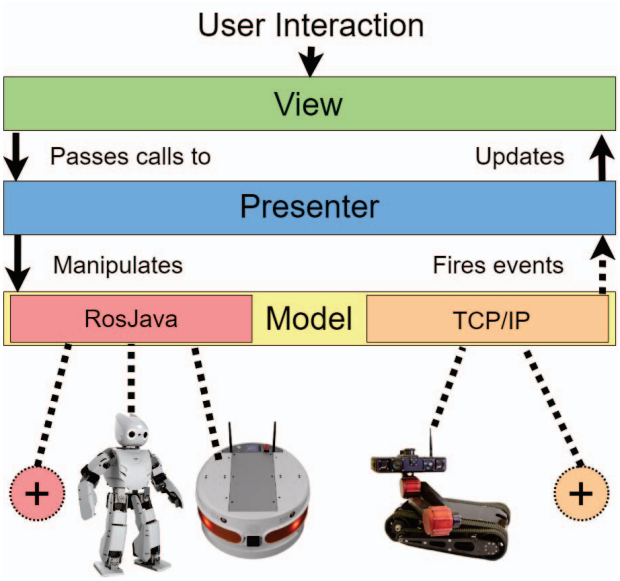


Fig. 1. MVP project architecture.

in such a way that the control mechanisms combine the presentation layer and the communication layer, which does not correspond to the chosen architecture. Therefore, we decided to separate the implementation of the view layer from the communication layer using the RxJava.

RxJava is a library that takes a functional approach to programming using the Observer and Iterator design patterns. It's based on handling events and data streams approach [14]. In the first case, the observable and observer concepts are used. The observable is the source of the events sequence, while the observer is its receiver. Observable's job is to generate events as a sequence (Observable), once (Single), with probability (Maybe), etc. The observer's job is to receive and process these events. This approach works well when combined with the ROS publisher and subscriber format. Also, the creation of observable and observer is functionally accompanied by mechanisms for changing the execution flows of operators. This means that the data can be processed asynchronously.

Moxy is a library for Android the main purpose of which is to implement the architectural MVP pattern taking into account the main system components' life cycle peculiarities [15]. The library provides functionality that preserves the state of the view layer at the time of its recreation or deletion. This makes it possible to keep received and transmitted information streams lossless.

### C. Visualization and control

This section describes the implementation of interaction tools with TIAGo Base, ROBOTIS OP2, and Servosila Engineer robots. The following solutions are demonstrated here: laser scanner data visualization, robot links' control using a 3D model, robot motion control.

*Laser scan viewer* : OpenGL ES (Open Graphics Library for Embedded Systems) technology is used to implement the laser scan viewer. The basic principle of OpenGL operation is to obtain sets of vector graphics primitives in the form of points, lines, and triangles, followed by mathematical processing of the received data and building a raster image on the screen and/or in memory [16]. A triangle-based vertex calculation is used to correctly display the data that comes in the form of a range of points. Here, each subsequent triangle uses the first and last vertices of the previous one. Figure 2 (left) shows a laser scan data visualizer on an Android device. The data comes from the Tiago Base robot. The robot movement controls are located at the bottom of the user interface.

*3D model viewer* : We have implemented links' control for robots with complex kinematics using 3D model visualization. Here, the relative position and orientation of parts in space are determined by solving the problem of direct kinematics [17]. DH parameters are determined from the robot's corresponding URDF file [18]. The graphics library OpenGL ES version 2.0 is used as a 3D technology [19]. Its difference from older versions is the presence of programmable shaders. Figure 2 (right) shows an example of the 3D model viewer, where the robot model ROBOTIS OP2 is displayed. The robot links' control is located at the bottom of the user interface. Active links are colored differently and are driven by signed floating-point values.

*Links and movement controls* : We have also added a simplified links' control option in cases where the 3D model viewer cannot be used.

Figure 5 shows the ROBOTIS OP2 robot links' control. Figure 5 (left) shows the robot's servo modules control panel.

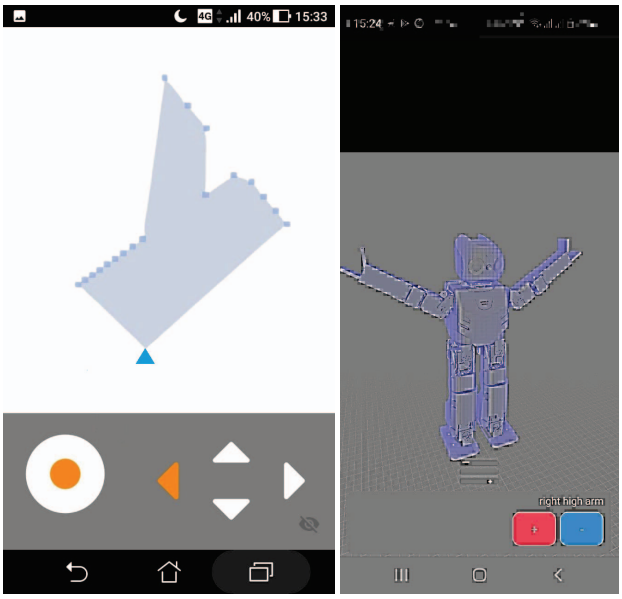


Fig. 2. Laser scan viewer (left) and 3D model viewer (right).

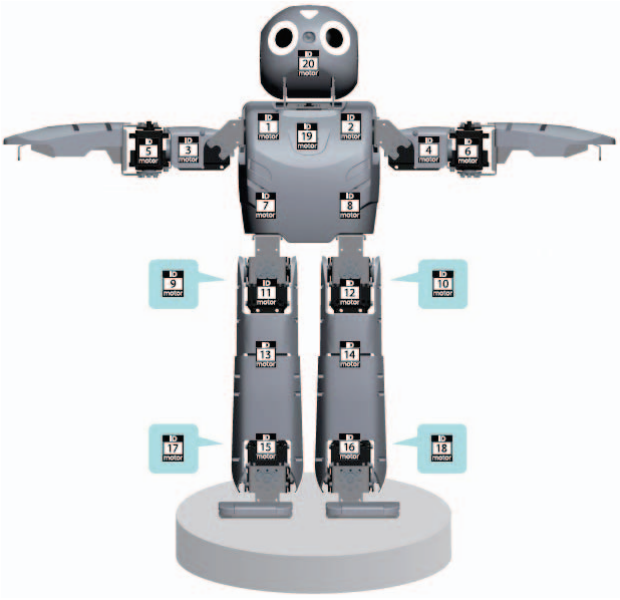

Fig. 3. ROBOTIS OP2 servo modules scheme.


Fig. 4. Servosila Engineer servo modules scheme.

The scheme of their location is taken as a basis (Fig. 3). The program switches to control mode by clicking on a certain servo module - the rotating kinematic chain is highlighted and buttons are added indicating the direction of the links' movement (Fig. 5 right).

Figure 6 shows the Servosila Engineer robot links' control. Figure 6 (right) shows the robot's servo modules' and motors' control panel. A diagram of their location is shown in Figure 4. The servo modules' rotation speed is set in a separate tab (Fig. 6 left). The link is controlled by clicking on the button

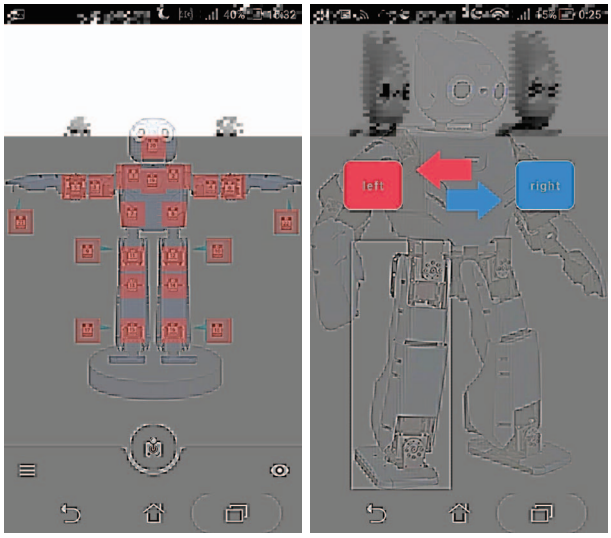


Fig. 5. ROBOTIS OP2 servo modules control panel (left) with transition to their management (right).

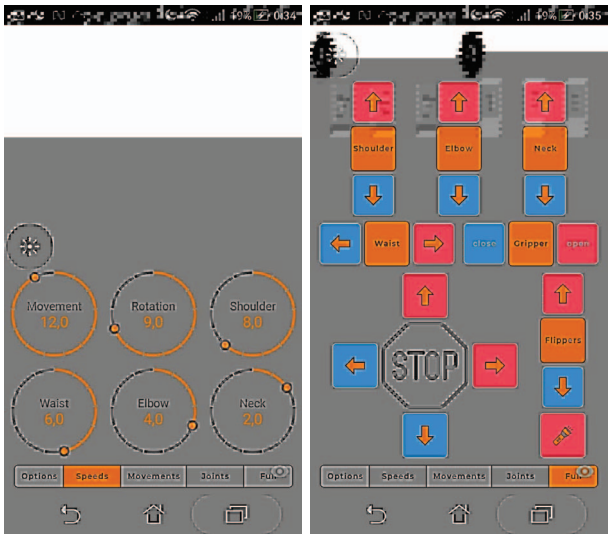


Fig. 6. Servosila Engineer speeds' tab (left) and servo modules control panel tab (right).

with its name, after which the movement direction buttons appear.

## V. Conclusions and Future Work

In this paper, we presented an Android OS-based control tool for operating multiple heterogeneous robots. We implemented network specifications for connecting to ROS and non-ROS systems using the RosJava library and the TCP/IP network protocol stack, respectively. The tool allows remote control of the humanoid ROBOTIS OP2 robot, the wheeled differential drive TIAGo Base mobile robot, and the crawler Servosila Engineer mobile robot. As part of our future work,

we plan to add support for other robots such as TurtleBot2, ROBOTIS OP3 [20], PR2, and others.

## ACKNOWLEDGMENT

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

## REFERENCES

[1] A. Karpov, B. Kryuchkov, A. Ronzhin, and V. Usov, "Designing human-robot interaction in a united team of cosmonauts and autonomous mobile robots on the lunar surface," in *Proceedings 26th International Conference Extreme Robotics (ER-2016), St. Petersburg, Russia*, 2016, pp. 71–75.

[2] B. C. Florea, "Smartphone controlled autonomous robotic platform," in *2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. IEEE, 2018, pp. 620–623.

[3] R. Meshcheryakov, "Control of hyperlinked cyber-physical systems," in *Proceedings of 14th International Conference on Electromechanics and Robotics "Zavalishin's Readings"*. Springer, 2020, pp. 27–33.

[4] J. C. Yepes, J. J. Yepes, J. R. Martinez, and V. Z. Perez, "Implementation of an android based teleoperation application for controlling a kuka-kr6 robot by using sensor fusion," in *Health Care Exchanges (PAHCE)*, 2013, pp. 1–5.

[5] N. Fung, "Light weight portable operator control unit using an android enabled mobile phone," in *in Proceedings of the SPIE Unmanned Systems Technolog XIII*, 2011, pp. 44–48.

[6] J. Nadvorník and P. Smutny, "Remote control robot using android mobile device," in *2014 15th International Carpathian Control Conference (ICCC)*, 2014, pp. 373–378.

[7] D. Kiryanov and R. Lavrenov, "Remote control application for "servosila engineer" on android mobile devices," in *The 2020 International Conference on Artificial Life and Robotics (ICAROB2020)*, vol. 25, 2020, pp. 440–443.

[8] I. Mavrin, R. Lavrenov, and E. Magid, "Development of a graphical user interface for a crawler mobile robot servosila engineer," in *2018 11th International Conference on Developments in eSystems Engineering (DeSE)*. IEEE, 2018, pp. 192–197.

[9] D. Kohler and K. Conley, "Rosjava – an implementation of ros in pure java with android support," 2011.

[10] I. Moskvin, R. Lavrenov, E. Magid, and M. Svinin, "Modelling a crawler robot using wheels as pseudo-tracks: model complexity vs performance," in *2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA)*. IEEE, 2020, pp. 1–5.

[11] S. Ramil, R. Lavrenov, T. Tsoy, M. Svinin, and E. Magid, "Real-time video server implementation for a mobile robot," in *2018 11th International Conference on Developments in eSystems Engineering (DeSE)*. IEEE, 2018, pp. 180–185.

[12] Z. Shao, M. Li, Y. Qu, G. Song, Y. Guan, J. Tan, and H. Wei, "Reliable communication mechanism design for interaction between android and ros," in *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2018, p. 92.

[13] T. Lou *et al.*, "A comparison of android native app architecture–mvc, mvp and mvvm," *Eindhoven University of Technology*, 2016.

[14] Z. Jovanovic, R. Bacevica, R. Markovic, and S. Randjic, "Android application for obsrving data streams from buitin sensors using rxjava," in *23rd Telecommunications Forum Telfor (TELFOR)*, 2015, pp. 918–921.

[15] R. Verdecchia, I. Malavolta, and P. Lago, "Guidelines for architecting android apps: A mixed-method empirical study," in *2019 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2019, pp. 141—-150.

[16] H. Lee and N. Baek, "Implementing opengl es on opengl," in *Proc. of the 13th IEEE ISCE*. IEEE, 2009, pp. 999—-1003.

[17] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," vol. 22, 1955, pp. 215–221.

[18] D. A. Yousuf, M. C. C. Lehman, D. M. A. Mustafa, and D. M. M. Hayder, "Introducing kinematics with robot operating system (ros)," in *In 122nd ASEE Annual Conference and Exposition*, 2015, pp. 1–18.

[19] Y.-S. Wang, Y.-X. Gai, and F.-Y. Wu, "A robot kinematics simulation system based on opengl," in *in IEEE Conf. Rob. Autom. Mechatronics RAM Proc.* IEEE, 2011, pp. 158—161.

[20] G. Vasilyev, A. Sagitov, and L. Gavrilova, "Walking algorithm for robotis op3 humanoid robot with force sensors," in *2019 12th International Conference on Developments in eSystems Engineering (DeSE)*. IEEE, 2019, pp. 20–23.