Architecture of a software system for robotics control

Aleksey V. Shevchenko
Information Systems Technologies Dept.
Stavropol City, NCFU
luckyleo769@mail.ru

Oksana S. Mezentseva
Information Systems Technologies Dept.
Stavropol City, NCFU
28mos05@mail.ru

Dmitriy V. Mezentsev
Information Systems Technologies Dept.
Stavropol City, NCFU
dmezentcev@ncfu.ru

Konstantin Y. Ganshin
Information Systems Technologies Dept.
Stavropol City, NCFU
magnuskos@gmail.com

Abstract

This paper describes the architecture and features of RoboStudio, a software system for robotics control that allows complete and simultaneous monitoring of all electronic components of a robotic system. This increases the safety of operating expensive equipment and allows for flexible configuration of different robotics components without changes to the source code.

1 Introduction

Today, there are no clear standards for robotics programming. Each manufacturer creates their own software and hardware architecture based on their own ideas for optimizing the development process. Large manufacturers often buy their software systems from third parties, while small ones create theirs independently. As a result, the software often targets a specific platform (more often, a specific modification of a given platform). Thus, even minor upgrades of an existing robotic system often require a complete software rewrite. The lack of universal software products imposes large time and resource costs on developers, and often leads to the curtailment of promising projects.

Some developers of robotic systems software partially solve the problem of universal software using the free Robotics Operation System (ROS), which also has limitations and is not always able to satisfy all the requirements of the manufacturers.

2 Equipment

The research was conducted using two robots produced by the SPA "Android Technics" the "Mechatronics" stand and the full-size anthropomorphic robot AR-601E (figure 1). The first is a manipulator arm with 10 degrees of freedom. The second is an anthropomorphic robot having 13 degrees of freedom in each arm, 6 in the legs, 1 in the torso, 3 in the neck.

In: Marco Schaerf, Massimo Mecella, Drozdova Viktoria Igorevna, Kalmykov Igor Anatolievich (eds.): Proceedings of REMS 2018 – Russian Federation & Europe Multidisciplinary Symposium on Computer Science and ICT, Stavropol – Dombay, Russia, 15–20 October 2018, published at http://ceur-ws.org

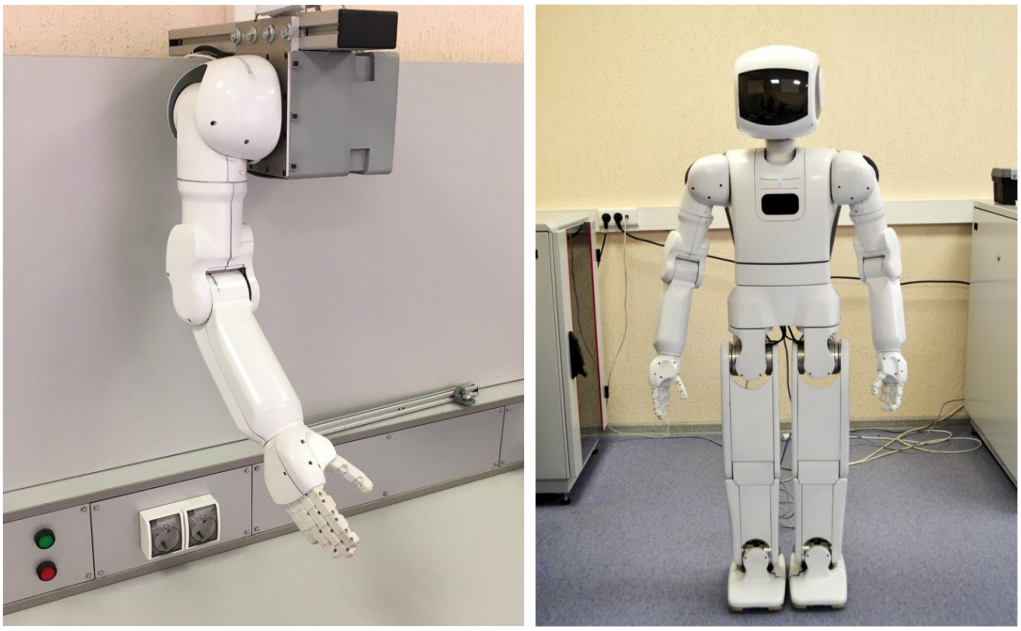

Figure 1: Mechatronics stand (left) and full-size anthropomorphic robot AR-601E (right)

The robots are very similar from a technical point of view: the control is performed by sending commands to the robot's motherboard via the LAN communication channel and receiving readings from the sensors. However, due to the use of different motor controllers, two different software systems based on ROS are used.

3 Software

The standard offering of the SPA "Android Technics" robots includes an ROS-based control system, the features and functionality of which informed the development of the system described in this paper.

ROS (Robot Operation System) is a software platform for programming robotic systems that provides functionality for distributed operation. Initially, ROS was developed for the STAIR project. The project included the creation and programming of two robots, STAIR-1 and STAIR-2[Qui07].

The STAIR project imposed the following requirements on ROS:

1. modular architecture since the project separated the tasks into fine motor control, pattern recognition, and movement in space, a monolithic architecture would significantly increase the implementation time and reduce development efficiency (increased number of errors and number of experiments) (figure 2) [New15];

2. parallel mathematical calculations the computing power of STAIR-1 and STAIR-2's on-board computers was insufficient for image recognition and inverse kinematics problems, so the most complex calculations needed to be transferred to more powerful computers via the LAN protocol [Len18];

3. hardware independence the STAIR-1 and STAIR-2 robots had different hardware architectures, so ROS was required to create a single programming interface to allow executing the same program on both robots without changing the program code [Wil17].

This architecture is optimal for research in robotics, but in education, teaching robotics is complicated by the lack of the following features in ROS:

- support for Microsoft Windows operating systems;

- basic functionality not requiring configuring and installing third-party software (graphical user interface, program modules for solving direct and inverse kinematics problems, image recognition or voice command libraries, etc.)

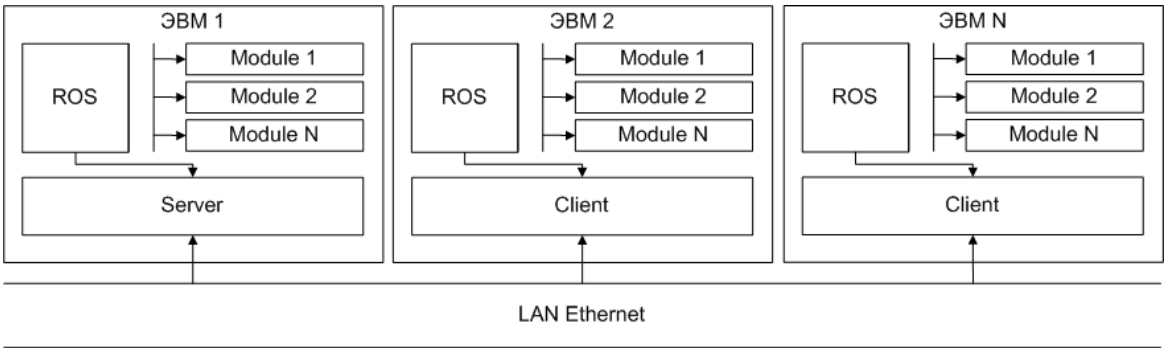

Figure 2: Connections structure between the ROS system modules

3.1 Alternative Software

As ROS contained a number of shortcomings and features that did not allow the full use of the existing equipment, alternative software systems for programming and controlling robotic systems were analyzed.

Marie (Mobile and Autonomous Robotics Integration Environment) the integration environment for mobile and autonomous robotic systems, developed in the Research Laboratory on Mobile Robotics and Intelligent Systems of the Universit de Sherbrooke, Canada [Cot07, Bea05]. It consists of three software "layers" [Bea05]: "Application", "Components" and "Kernel". The "Application" and "Components" layers provide basic tools for developing control programs, the "Kernel" contains tools for exchanging data between layers and connected equipment, and the central "Mediator" module performs the control. Marie does not meet our requirements for the software system due to the lack of built-in tools for modeling and emulation of robotic systems.

Microsoft Robotics Developer Studio (MRDS). The MRDS platform includes Visual Programming Language (VPL) and Visual Simulation Environment. For describing algorithms of the robots' behavior, the Visual Programming Language (VPL) is offered for beginners and the C# language for professional programmers [Jac07, Gad05, Joh08, Mor08]. The main disadvantage of MRDS is the lack of fault tolerance and power control, as the system is not able to prevent potential damage to the expensive equipment in the presence of program errors.

There exist commercial software products for the development of robotic systems software (Gostai Urbi, Evolution Robotics ER1, etc.), as well as proprietary solutions of the military and law enforcement agencies, analysis of which was not the purpose of this research.

The analysis of software systems has shown that all existing solutions do not fully satisfy the set of functional requirements required for working with robotic equipment: modularity, hardware independence, redundancy, the presence of a graphical user interface and robotic systems emulation. An alternative solution is the presented RoboStudio, a software platform for the control of robotic systems. Experimental studies of the system functionality were conducted using the full-size anthropomorphic robot AR-601E and the "Mechatronics" stand.

4 Developed software

4.1 Advantages of RoboStudio, the developed software system for robotics control

Modularity as with ROS, the developed system allows for quick expansion of the functionality and separation of the tasks between the modules of the system. The connection between the modules is performed by direct method calls using Win32 and the .NET platform, and not through the local network.

Hardware independence the system implements the ability to manage any type of robotic systems.

Graphical user interface for improving the convenience of working and teaching with the system, the system is managed with a graphical user interface.

Redundancy the system has the entire set of basic tools for working with robotic systems: the development environment, debugging and testing, emulation of the robotic system, solving the main tasks of kinematics and dynamics.

4.2 Architecture

RoboStudio includes the following modules (figure 3):

1. kernel the control module for all pluggable software parts of the system;
2. driver dynamically replaceable element connecting the system with the connected robotic equipment. Compiles commands based on the architecture of the hardware. Normalizes the readings from sensors to a common format. In addition to exchanging data with the equipment, the module contains detailed information about the robotic system for constructing it in the virtual emulator;
3. script interpreter executes the action programs, sending commands to a single programming interface for their further execution by add-ons, emulator, or connected equipment;
4. mathematical modeling module software module solving direct and inverse problems of kinematics;
5. emulator software module for experimenting and testing different scenarios on a virtual robot model;
6. add-ins software modules that extend the functionality of the system.

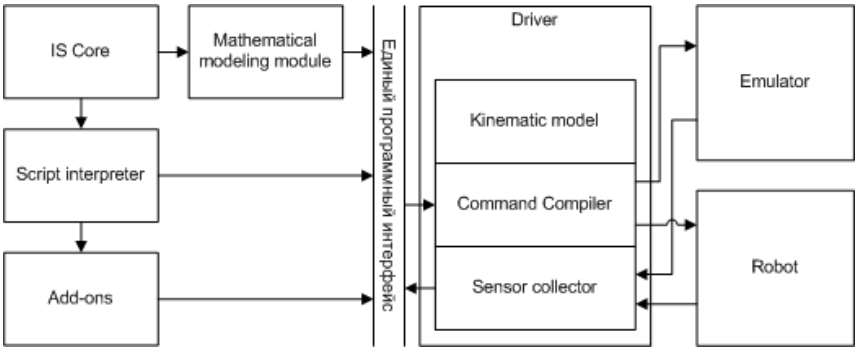

Figure 3: Architecture of the software system

4.3 Practical solutions

Each RoboStudio module, except for the kernel, is a dynamically connected Windows library (DLL) containing a specific set of commands responsible for the operation of this module. The commands are called from the interpreter. If necessary, any module, except the kernel, can be replaced with a custom version of the corresponding program element. Direct connection between the modules without the use of a local network allows the computing power to be saved in favor of solving the most important tasks.

When the system is started, the interpreter performs the assembly of the installed configuration of the modules and their command lists in the switch (figure 4), from which a particular command is directly called. Before executing, user commands pass the translation stage from the rScript language to the numerical values corresponding to the command numbers in the switch, while the kernel of the system issues commands directly to the switch.

Figure 5 the algorithm for the entire execution cycle of the high-level GoFromTo(A, B) command moving from point A to point B:

- The command goes to the interpreter, where it is translated to the corresponding number in the switch.
- The switch calls a method corresponding to the number in the walk simulation module, where the entire task is broken down into iterations based on the received parameters.
- At each iteration, the solutions of the direct and inverse kinematics problem are performed to find the positions of the joints at the current moment of the walking cycle.
- From the data obtained, a command for the robotic system is formed, which is then sent to the system core for safety checks.

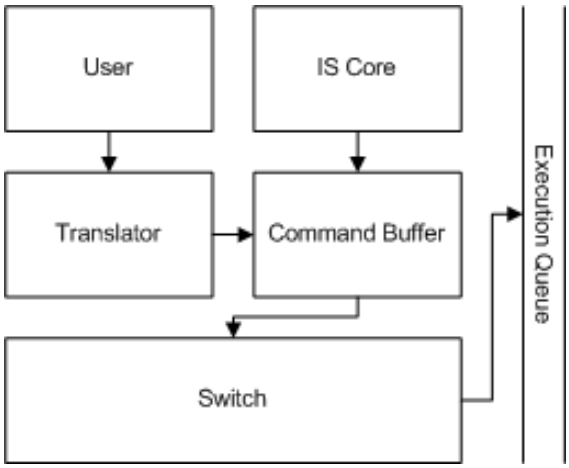

Figure 4: The scheme for executing user and kernel commands

- If the command is safe (it will not lead to fracture of the joints, system freeze, etc.), it is sent to the driver where it is translated into machine code and executed.

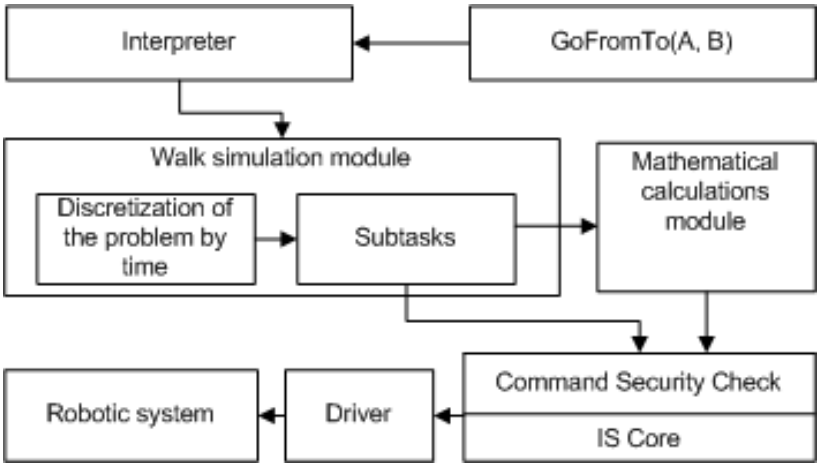

Figure 5: Scheme for executing the robot move function from A to B point

4.4 Machine code translation

The manipulation of the SPA "Android Technics" robots is performed using two arrays of 1472 bytes each. The first is used to send commands (an array of commands) to the main robot controller and the second contains the sensor data (an array of data). Data exchange takes place in a separate thread continuously and regardless of whether new commands are given or not. In the instruction array, 16 bytes of data are stored for each motor, which store the motor rotation angle, drive stiffness, and other servo parameters.

These parameters are unique and can vary depending on the motors, controllers and other components of the electronic circuit used by the robot. Therefore, the developed software system provides a special module a driver that translates the commands of the system into the required format, considering the hardware features of the connected equipment.

4.5 Operation of the software system

The RoboStudio system is the software development environment for any kind of robotic systems. It is possible to develop control programs of almost any complexity and direction without recompiling the main code of the software system by using the scripting language interpreter.

The emulator allows debugging newly developed scenarios on a virtual robot model, thereby avoiding the risks of executing erroneous commands.

Modularity of the system and connection of the equipment drivers allow to expand the functionality of the software system during its operation and organize the manipulation of virtually any robotic systems.

The graphical user interface of the software system (figure 6) consists of the emulator panel, the source code editor, the console command and the status bar, which displays information about the power supply and the readings of all the sensors of the robotic system.

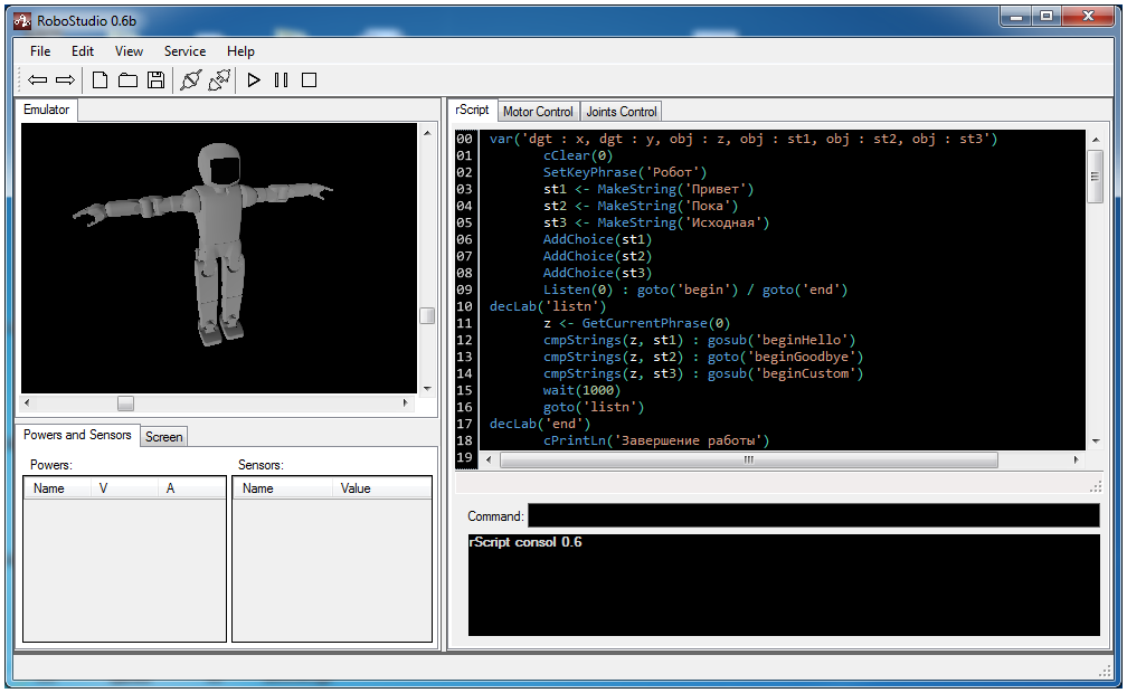

Figure 6: RoboStudio main window

Standard functionality allows to get rid of unnecessary software, saving the computing power for solving more important tasks, allows to flexibly configure the software system for almost any equipment without changes in the source code, to perform full and simultaneous monitoring of all electronic components of the robotic system, which increases the safety of operating expensive equipment.

The presence of software products like RoboStudio explains the great success of foreign developers. But, unfortunately, such systems are classified and are not widely available. Thus, the developed system will contribute to the import substitution in the field of robotics production.

5 Differences between RoboStudio and analogues

RoboStudio runs on Windows operating systems and ROS runs under OS based on the Linux kernel, which is why RoboStudio cannot be a direct competitor to ROS. These products are developed to solve different kinds of problems in different working conditions, but have common architectural features: modularity, hardware independence, parallelization of tasks, etc.

The most important difference of the developed software system from ROS is the greater automation of the installation, configuration and operation of the presented software system. This is expressed in the presence of a basic set of functions required when working with robotic systems. On the other hand, before starting to work with ROS, one needs to install and configure required software.

6 Further development of RoboStudio

At the moment, through the drivers, RoboStudio was able to scan the premises with Microsoft Kinect, record and recognize speech commands from an analog microphone, control the robot AR-601E and the "Mechatronics"

stand.

As mentioned above, robotics systems are controlled by exchanging instruction packets and sensor readings. The packet format can be configured in accordance with the architecture of the robotic system, and the communication channel can be implemented with USB, Ethernet and WLAN interfaces. Thus, RoboStudio does not have any strict limitations on the type of equipment connected to it, since the driver connected to the system performs low-level control, and the list of supported RoboStudio equipment is constantly expanding.

References

[Len18] Lentin Joseph. Robot Operating System (ROS) for Absolute Beginners: Robotics Programming Made Easy. *Apress, 2018.* – ISBN 9781484234051.

[Wil17] William D. Smart, Brian Gerkey, Morgan Quigley. Programming Robots with ROS. - *Chapman and Hall/CRC, 2017.* – ISBN 9781449323899.

[New15] Wyatt Newman. A Systematic Approach to Learning Robot Programming with ROS. – *O'Reilly Media, Inc., 2015.* – ISBN 9781498777872.

[Cot07] C. Cot, D. Ltourneau, C. Raevsky, Y. Brosseau, and F. Michaud. Using marie for mobile robot component development and integration. – *Software Engineering for Experimental Robotics Book Series, vol. 30 of Springer Tracts in Advanced Robotics*, Springer, Berlin, Germany, 2007.

[Bea05] . Beaudry, Y. Brosseau, C. Ct et al. Reactive planning in a motivated behavioral architecture. – *in Proceedings of the National Conference on Artificial Intelligence, vol. 3*, AAAI 05, 2005.

[Jac07] J. Jackson. Microsoft Robotics Studio: a technical introduction. IEEE Robotics and Automation Magazine, vol. 14, no. 4, pp. 8287, 2007.

[Gad05] K. Gadeyne, T. Lefebvre, and H. Bruyninckx. Bayesian hybrid model-state estimation applied to simultaneous contact formation recognition and geometrical parameter estimation. The International Journal of Robotics Research, vol. 24, no. 8, 2005.

[Joh08] K. Johns and T. Taylor. Professional Microsoft Robotics Developer Studio. Wrox Press, Birmingham, UK, 2008.

[Qui07] Morgan Quigley, Eric Berger, Andrew Y. STAIR: Hardware and Software Architecture. AAAI Robotics Workshop, 2007.

[Mor08] S. Morgan. Programming Microsoft Robotics Studio/ Microsoft Press, Redmond, Wash, USA, 2008.