SOFTWARE ANALYSIS FOR MOBILE ROBOTS CONTROL PROGRAMS

Abstract. The use of the software allows the mobile robot to control the working parameters: turn on and off the mechanisms and devices, monitor the indicators of the sensors, perform various technological operations (cutting, welding, painting, etc.), calculate the trajectory of movement depending on the working surface, etc. Research in the field of robotics testifies to the high activity of scientific works on the creation of high-precision and energy-efficient robotic systems in general for autonomous mobile robots by improving control programs. The work is devoted to the review and analysis of the software for creating control programs for mobile robots. The work presents a generalized structural diagram of a hierarchical mobile robot control system, in which decentralized software processing of information takes place, and separate software and hardware components are remote from each other. When building a mobile robot control system, various robot programming environments are considered, which represent a wide range of tools for creating various models and systems. Moreover, the issues of using graphic and text software environments with high-level programming languages are considered. Development environments are considered among the software complexes: LabView; NXT-G; Robolab; EV3-G; MRDS; Scratch; 12Blocks; Simulink; ROBO Pro; Arduino Studio and TRIK Studio. The following are considered the most common programming languages at work: C++, Python, Pascal, JAVA and Scratch. All software is analyzed according to the following criteria: mathematical expressions, computational model, interpretation, stand-alone use, code generation, modeling, debugging, tutorials, free, platforms, designers, license and development prospects. Among the software, EV3 and Arduino text tools stand out for their capabilities, and Simulink and LabView among graphic tools, as these software tools have proven themselves to be powerful development environments with fairly universal approaches to creating programs in mobile robotics.

Keywords: mobile robot; software; analysis; programming languages; control system
Problem statement

Compiling complex programs for robots includes writing the software according to which the robot operates. In robotics, software is considered as an area that extends from the software of the robots themselves, the software of the systems related to the robots, to the software related to robotics as a whole and covers both of the above mentioned types. Despite the existence of such classifications, it is not always clear to which group one or another type of software used in robotics should be assigned [1–3].

The central core of the robot's software includes programs written by the end user and interpreter software that translates the user's programs into a language understood by the controller. The software of the controllers allows various systems to work, providing a response to feedback signals. Robotic systems can output information using graphics systems, which requires special software, and controllers process signals in real time, which requires significant computing power. Moreover, the second-generation robots, based on the use of sensory information (visual and tactile), have a much larger volume of software for processing the information received from the sensors.

Sensor devices (video cameras, sets of sensors) are able to provide the feedback system with a large amount of information, and one central processor will not be able to process it. Therefore, in the processing process, there is a multi-level at which information is processed by processors with the appropriate software [4–7].

Therefore, robotics software is based on the extensive use of various disciplines. Combination of automated systems of design and management of production processes. The complexity of the software used in both systems is extremely high. Since robotics is built on three foundations: the application of electronics, mechanics, computer software. Programming and robotics are generally closely related. More and more scientists and manufacturers are investigating the processes of writing high-precision and energy-efficient software, as interest in process robotics grows, as does investment in projects related to programming and robotics.

The article aim

Existing programming environments are developing quite quickly, and analytical publications superficially highlight the important tasks of connecting the control program with robotics and do not provide generalized practical recommendations for using one or another software depending on the task of the robot. The purpose of this article is to analyze and formalize the distribution of commercial and open tools for programming mobile robots, taking into account the technical features of writing programs.

Basic material

One of the fastest growing areas of robotics is mobile robotics [8 – 13]. Mobile robot can be divided into two categories: one is a remote-controlled robot, and the other is a robot that can perform certain actions in an autonomous mode. In most cases, the robot is controlled by a human operator at the movement level, which requires a person to constantly observe the robot and quickly control its movements.

The mobile robot control system is presented in fig. 1. Software for the mobile robot control system should solve the following tasks:

- Processing of sensory data (including data from the interface with the operator) in order to collect information about the robot and the environment around it.
- Planning activities to understand the target task and planning the sequence of subtasks necessary to complete this task.
- Formation of such software trajectories of WRI movement that would lead to the execution of a local subtask by the robot (for example, arriving at a target point in an environment with obstacles).

Formation of such setting actions on the actuators of the robot, which would lead to the most accurate and fast execution of the program trajectory of motion.

![Figure 1 – Generalized block diagram of the mobile robot control system](image-url)

---

1. Управління технологічними процесами

112
One of the features of building a control system for an intelligent mobile robot is that it is built according to a hierarchical multi-level principle, according to which, with an increase in the hierarchical rank of a subsystem, its degree of intelligence increases. The topmost link in this hierarchy is the behavior control system, followed by the motion control system, and the actuator control system is the lowest link in this hierarchy. In addition to the listed subsystems, the structure has an information-measuring system, which must also have some intellectual capabilities, and an interface with the operator.

The behavior control system (strategic level) is designed to form the appropriate behavior of the robot to complete the task assigned to it. At the output, this system generates target designation for the motion control system: the target waypoint, the required state of the robot drives, the commands for controlling the operating modes of the information-measuring system.

The motion control system (tactical level) is designed to plan such software trajectories of the robot's movement that would bring the robot to the specified target state in an environment with obstacles, taking into account the dynamic characteristics of the robot. The target state for this system is formed by the behavior management system. At the output, this system generates the required command value of the speeds of linear movement, azimuthal rotation of the robot.

The actuator control system (drive level of the control system) solves the tasks of controlling the actuators of the robot. This system implements an interface with the robot's hardware.

The information-measuring system is designed to collect, process and convert sensory information into signals that are convenient for use in the robot control system. In this robot, the video image received from the camera is converted into a set of parameters, on the basis of which other subsystems make certain decisions.

The operator interface is an on-screen menu for conducting a natural language text dialogue with the user, as well as a manual control panel for the robot.

To create a robot control system, you can use different programming environments. These environments can be divided into two large groups - these are visual and text-based programming environments. Also, robot control environments are distinguished by whether they are specialized in controlling a particular robot or support a number of robots from different manufacturers (Fig. 2).

One of the important features of working with text languages is that you have to remember the syntax of the language, keywords, parentheses, commas, and so on. But with a visual programming language, it is easier to determine which block is responsible for what and how to build connections between them. Visual programming is used not only for simple tasks, but also in quite complex tasks. For example, in relationship editors in relational databases, dataflow programming, program designers, and so on [14 – 18].

During the development of any program, it is assumed that it will develop over time – to receive new functions and entities. Perhaps some parts may change with an increase in the number of robot sensors. In visual programming, the interface for manipulating graphic objects is currently limited, but development is actively underway to expand the work area, which will allow creating complex programs. To manage complexity in text programming, many concepts and architectural approaches have come up. For example, object-oriented programming, various architectural design patterns. If you follow them, it will save the developer time and it will be easier to scale the project.

![Diagram: Robot programming environments](image)

C and C++ are #1 among robotics languages. Although C++ is not so easy to work with because it requires software to be compiled, it is still one of the most reliable programming languages. It allows you to create complex programs that follow a clear structure. Today, C++ is arguably more useful in robotics than C. However, the latter remains one of the most energy-efficient programming languages. Python is a very flexible and fast open source programming language. It is probably one of the easiest, most popular and most versatile languages. It is object-oriented programming (OOP), completely connected with the development
of artificial intelligence and virtual reality. Additionally, there are a large number of free libraries for Python. Python is useful in robotics because it is one of the main programming languages in ROS (along with C++). But it may become even more popular as more robotic electronics support this language by default. Pascal is a BASIC language, and is literally based on the BASIC language. Most often, industrial robots are programmed in the Pascal language. It is simple because it uses structured programming and data structuring. Java is a general-purpose object-oriented programming language. It is designed to allow application developers to build code once and then reuse it anywhere. In other words, JAVA code can run on any JAVA-enabled platform without the need for recompilation. In addition, JAVA is a useful language in robotics and is used in the creation of artificial intelligence. Scratch is extremely popular among beginning roboticians. It is a visual programming language, in essence its principle of operation is to drag and connect blocks.

LabVIEW [19, 20] is a graphical G-language software development environment created by National Instruments in 1986. LabVIEW allows you to quickly create applications for control, testing, measurement, and more. This environment allows you to program in terms of data flows and allows you to use various design patterns to create applications, for example, it is possible to apply the architecture of a state machine. LabVIEW supports a large number of hardware platforms, provides a huge set of libraries that contain tools for working with complex mathematical structures, tools for creating virtual instruments, computer vision algorithms, etc. For the interaction of blocks, libraries provide a set of different links that differ in the type of data transmitted through them. The program created in the LabVIEW environment is a virtual instrument (virtual instrument), it is divided into two parts: a block diagram that describes the logic of a virtual instrument, and a front panel that describes the interface of the instrument. It is important to note that the language compiler automatically parallelizes code sections that have parallel blocks, creating separate threads for their execution. The possibilities of using this environment are great, there are components that allow you to use this environment to work with LEGO MINDSTORMS NXT/EV3 robotics educational kits. LabVIEW allows you to interpret programs and generate code from them to run programs autonomously on a device.

Microsoft Robotics Developer Studio (MRDS) [21]. MRDS Platform includes the visual programming language Visual Programming Language (VPL) and a simulated visual 3D environment. The visual programming language the implementation of the algorithm, but also the management of the complexity of the project. In Visual Programming Language (VPL) is offered as a means of describing robot behavior algorithms for novice programmers, the C # language is for professional ones. Writing a program in VPL consists in choosing the appropriate components for solving the problem and establishing a connection between them.

LEGO MINDSTORMS Education NXT-G [22] is a graphical programming environment for the LEGO Mindstorms NXT constructor. The environment is based on the LabVIEW industrial environment and uses the G data flow language. This software has an intuitive interface, the creation of robot control programs resembles the creation of flowcharts and is carried out using special blocks placed on LEGO beams along the axis of the sequence of actions. The order of program execution is determined by the order of the blocks. NXT-G automatic laces the blocks in the diagram physically: execution obeys the order of the blocks, the data needed by the following blocks must be explicitly connected by the flow. The environment provides a rather large set of blocks (one hundred and ninety-three). In NXT-G, there is practically no support for mathematical expressions: to specify complex expressions, you have to build a parse tree in blocks. The advantage of the environment is that it is distributed free of charge.

Robolab [23] is another robot programming environment that is a simplified version of the LabView industrial programming environment. The environment uses a visual language that has a total of about four hundred blocks. In order not to frighten a novice user with a cumbersome palette, the environment has the ability to select the level of use of the program. Levels limit the size of the palette used, the first level, for example, contains about twenty elements and only allows you to substitute a block in the empty space allotted for it. At the last level, the entire palette is available to the user (the placement of blocks is not limited in any way). The palette includes control blocks, blocks of various arithmetic metrical actions (mathematical expressions can be specified explicitly in C language), blocks of variables, subroutines, work with threads of execution (parallelization of execution), loops (implemented using labels and jumps). Blocks in Robolab are shrouded in a network of various "wires", various modifiers come through them, corresponding to different types of data. This makes it difficult to understand when working with a large program. Another disadvantage is that the blocks for interacting with different constructors are not separated in any way. They are mixed, but not all commands, for example, for the LEGO RCX can be used with the LEGO NXT robot.

TRIK Studio [24] Another example of a programming environment is the TRIK Studio robot programming environment (see Figure 6). It allows you to program several types of microcontrollers – LEGO and TRIK using a sequence of icons. In total, there are about a hundred different blocks in the language that are responsible for interaction with the robot and algorithmic
and mathematical support. The environment has a
modern user interface. For the convenience of
programming, the blocks in the palette are divided into
groups according to their functional value. The
programming language in TRIK Studio is completely
based on the control flow model, data flows are not used.

ROBO Pro [25] is the official programming
environment for the ROBO TX controller that controls
models assembled from the fishertechnik kit. Programming in it is carried out in the language of block
diagrams. The language supports all major algorithmic
constructs and data types. The environment does not
satisfy the rest, "advanced" criteria. It is worth noting that
ROBO Pro is practically the only environment that
programs real robots in terms of block diagrams,
however, the system does not support programming of
any other devices, except for the ROBO TX controller.

12Blocks [26] is another Scratch-like tool for
programming Lego Mindstorms NXT and Arduino
robots, and other less popular platforms are available
(like Scribbler). The environment is cross-platform,
available for Windows, Linux and Mac OS X. It is
possible to execute programs on the Cognition 3D
simulator, generate code from a diagram, and integrate
with ROS7. The language supports all the basic
algorithmic constructions and data types, it is possible to
extract code into a subroutine. There are opportunities for
autonomous execution of the program by the robot,
debugging the program on a computer with sending
commands to the robot, as well as plotting graphs from
sensors in real time. The negative aspects include the
following: 12Blocks has poor methodological support,
there are practically no communities around the
environment on the Internet (however, there is a set of
English-language video instructions for using the main
features of the environment). Also, the system does not
have any means of automatic checking of tasks. 12Blocks
is distributed under a commercial license.

Scratch [27] is an open source cross-platform
visual programming environment developed at the
Massachusetts Institute of Technology to teach students
the basics of computer science. Programming is carried
out by connecting blocks, resembling mosaic elements.
Scratch allows you to draw and program simple graphic
objects called sprites. In its “pure” form, Scratch does not
allow you to program robots, however, there are a large
number of extensions and environments based on Scratch
that allow you to program Lego WeDo, Lego NXT, Lego
EV3 and Arduino robots. Among such "independent"
projects created on the basis of Scratch, we mention S4A
and mBlock for Arduino programming and Enchanting
for NXT programming. Common advantages for
Scratch-like environments are ease of learning, an
attractive user interface, openness and freeness, the
ability to debug remote control of the robot from a
computer and download code for offline execution (the
latter is not available in all Scratch systems). There is also
the possibility of executing a program on a virtual sprite,
which, according to our criteria, can be considered as
debugging on a simulator (however, there is no question
of the proximity of such a simulation to reality). The
negative aspects include the lack of "advanced" means of
teaching programming. For example, there is no
possibility of generating readable code from a visual
model, which could greatly facilitate the transition of
students to textual languages. Algorithmic aspects are not
fully supported, for example, there is no support for
arrays of dimension greater than 1. There are also no tools
for automatically checking the correctness of task
solutions.

LEGO MINDSTORMS Education EV3 [28]
software16 for LEGO Mindstorms EV3 sets solves some of
the problems of the NXT-G environment, such as
setting math formulas. The environment supports LEGO
NXT programming (although there are known
compatibility issues). EV3 software provides the user
with a small set of blocks for programming, execution is
subject to an explicitly defined flow of control, which
partially uses the data transfer model (see Figure 4).
The language provided by the environment uses fifty-three
different blocks that are responsible for controlling
various sensors, sensors, actuators, controller buttons, for
implementing mathematical functions, as well as
algorithmic constructions: fork, loop, switch, etc. The
environment does not support all operating systems (for
every, there is no support for Linux).

Simulink [29] is a graphical programming and
simulation environment that uses block diagrams. The
environment was created by MathWorks. The principles
of its work are similar to LabVIEW. Simulink allows you
to simulate various dynamic models, conduct simulation
and automatic code generation, testing and verification.
Provides many libraries with various blocks, allows you
to interact with the MATLAB package, use algorithms in
models and export simulation results for further analysis.
Using the Robotics System Toolbox13, Simulink has the
ability to develop control programs for autonomous
robots. The environment provides an extensive set of
libraries containing various blocks (about two hundred)
for verification, interaction with sensors and other robot
devices, for working with mathematical operations, and
others. Like LabVIEW, Simulink is based on a data flow
model, which is better suited for robot programming due
to its reactive nature.

Arduino [30] is a robot programming environment
based on Arduino. The Arduino development
environment interface contains the following main
elements: a text editor for writing code, a message area,

115

a text console, a toolbar with traditional buttons, and a
main menu. This framework is written in Java and is
based on Processing and other open source software. Unlike the online version of the code editor (Arduino Web Editor), the desktop version can be used when there is no internet. This software allows the computer to communicate with the Arduino to both transfer data and upload code to the controller. The Arduino programming language is based on C / C++, linked to the AVR Libc library and allows you to use any of its functions.

All the listed software are evaluated according to the criteria and the results of the comparison of the environments are given in the table.

### Table – Comparison of visual programming environments for robots

<table>
<thead>
<tr>
<th></th>
<th>LabView</th>
<th>NXT-G</th>
<th>Robolab</th>
<th>EV3-G</th>
<th>MRDS</th>
<th>Scratch</th>
<th>12Blocks</th>
<th>Simulink</th>
<th>ROBO</th>
<th>Arduino Studio</th>
<th>TRIK Studio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematical</td>
<td>+</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>±</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>expressions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computation model†</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>D</td>
<td>D</td>
<td>C</td>
<td>D</td>
<td>C</td>
<td>D</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>Interpretation</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Standalone use</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>±</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Code generation</td>
<td>C</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>C#</td>
<td>–</td>
<td>C and</td>
<td>C</td>
<td>Arduino C</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>etc.</td>
<td></td>
<td>JavaScrip t, F# and etc.</td>
<td></td>
</tr>
<tr>
<td>Simulation</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Debugging</td>
<td>+</td>
<td>–</td>
<td>±</td>
<td>±</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Methodologic aids</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>+</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Free</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>±</td>
<td>±</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Platforms²</td>
<td>WLM</td>
<td>WM</td>
<td>WM</td>
<td>WM</td>
<td>W</td>
<td>WLMw</td>
<td>WLM</td>
<td>WLM</td>
<td>WLM</td>
<td>WLM</td>
<td>WLM</td>
</tr>
<tr>
<td>Constructors³</td>
<td>NE</td>
<td>N</td>
<td>N</td>
<td>NE</td>
<td>Nf</td>
<td>NEA</td>
<td>NAS</td>
<td>EA</td>
<td>f</td>
<td>A</td>
<td>NET</td>
</tr>
<tr>
<td>License⁴</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>O</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Development</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

¹C means for "control flow", D – for "data flow".
²For lack of space, abbreviations are used. Each letter corresponds to the operating system. W means “Windows”, M – “Mac OS X”, L – “Linux”, w – “web”.
⁴O means “open”, P – “proprietary”.

### Conclusions

The use of the software allows the mobile robot to control the working parameters: turn on and off the mechanisms and devices, monitor the indicators of the sensors, perform various technological operations (cutting, welding, painting, etc.), calculate the trajectory of movement depending on the working surface, etc. High-quality software allows you to increase the accuracy of work and the efficiency of the use of energy resources.

The article presents a comparative analysis of a large number of currently popular environments for programming mobile robots. After considering all the tools listed above, it becomes clear that programming environments for robots, as a rule, are a small set of text or graphic blocks. Based on these blocks, programs are created to solve typical robot tasks using an easy-to-understand execution model – the control flow model (perhaps with partial use of the model's data flow). That is, most programming environments are primarily based on the execution model in terms of data flows, where the useful work of a block is performed only when data is received.

Of course, the evaluation of the software environment depends on the field of use of the mobile

Theoretical Artificial Intelligence robotic systems. Materials of the all-Ukrainian scientific and practical conference of young scientists and students "Information technologies in education, technology and industry" on October 13, Ivano-Frankivsk, 117–119.

Learning Technology Review.


References
Управління технологічними процесами


The article was received by the editorial board 29.03.2023
Управління розвитком складних систем (53 – 2023) ISSN 2219-5300

высокоточных и энергоэффективных робототехнических систем для автономных мобильных роботов снизил уровень вводимого керуемых программ. Работа присвоена розгляду и анализу программного обеспечения создания керуемых программ мобильных роботов. У роботі представлено узагальнену структурну схему ієрархічної системи керування мобильным роботом, в якій відбувається децентрализоване програмне опрацювання інформації, а окремі програмно-апаратні компоненти віддалені один від одного. При побудові системи управління мобильного робота розглядаються різні середовища програмування роботів, які представляють широкий інструментарій для створення різних моделей і систем. Розглядаються питання застосування графічних та текстових програмних середовищ з модами програмування високого рівня. Серед програмних комплексів розглядаються середовища розробки: LabView; NXT-G; Robolab; EV3-G; MRDS; Scratch; 12Blocks; Simulink; ROBO Pro; Arduino Studio and TRIK Studio. Найпоширенішими ж мовами програмування в роботі вважаються такі: C++, Python, Pascal, JAVA та Scratch. Все програмне забезпечення аналізується за такими критеріями: математичні вирази, обчислювальна модель, інтерпретація, автономне використання, генерація коду, моделювання, налагодження, методичні посібники, безкоштовність, платформи, конструктори, ліцензії та перспективи розвитку. Серед програмного забезпечення своїми можливостями відзначаються текстові засоби EV3 та Arduino, а серед графічних – Simulink та LabView, оскільки ці програмні засоби зарекомендували себе потужними середовищами розробки з доволі універсальними підходами створення програм у мобільній робототехніці.

Ключові слова: мобільний робот; програмне забезпечення; аналіз; мови програмування; система керування

Link to the publication
