DOI: https://doi.org/10.30837/MMP.2022.151

MULTIFLOWED SOFTWARE MOTION CONTROL TECHNOLOGY FOR A TWO-LINK MANIPULATOR WITH FOUR DEGREES OF FREEDOM

Novoselov S., Sychova O., Tesliuk S.

This paper describes a multiflowed software motion control technology for a two-link manipulator with four degrees of freedom. The proposed technology uses a set of independent timers that allow to realize independent control flows of program execution. This technology uses the developed database structure to store the control program structure, which is a set of instructions, conditional and unconditional transition operators, waiting commands with the possibility of connecting to the PLC I/O ports via Modbus protocol. A characteristic feature of the proposed technology of executing commands in the program is the concept of folders. Folders in this sense are a grouping of commands that constitute a certain cyclic sequence of actions for the manipulator. Folders are not a visual component, but the essence, uniting several commands, executed one after another, in a database. An angular manipulator motion control program using visual components has been developed. Testing of the developed program was carried out, which showed its performance and reliability of the execution of commands given by the operator.

Introduction

Software control systems of manipulator movement are designed to create programs to control the movement of manipulator links, remotely control the device and visualize the current state of the moving mechanisms. The main task of the software tool is to facilitate the process of creating control programs and increase productivity by visualizing the motion of mechanical moving parts of the manipulator [1–3].

When creating control programs, the characteristics of the specific type of manipulator for which they are created are taken into account. Three basic functions of data transformations are aimed at solving three standard configuration problems of manipulator kinematics with protection of their solutions from dangerous manipulator movements:

– conversion of the angular configuration of manipulator links to Cartesian coordinates of a selected point on the gripper axis;

– transformation of the manipulator target coordinates with the gripper target parameters into the angular configuration of the manipulator links at the target point;

– linear interpolation of motion in Cartesian coordinates of the target vector according to the specified values of angular configurations at the current and target points of the planned motion of the manipulator gripper.

The aim of the work is to create a software tool for motion control and simulation of angular manipulators using visual components.

Multiflowed software manipulator motion control

The control object is a training model of a manipulator. The manipulator contains two movable joints and can rotate around the vertical axis. The manipulator also has a gripper for gripping and moving parts within its working area.

There are three stepper motors at the heart of the design. Each stepper motor realizes a certain degree of freedom. The control module, based on the Arduino Mega controller, controls the motors.

The manipulator has end sensors, one for each degree of freedom. At the beginning of operation, the control system is initialized. This starts a test run of each stepper motor and monitors the wear of the corresponding end sensor. If all sensors are triggered, the device enters the mode of waiting for commands from the user.

A simplified functional diagram of the industrial robot is shown in Fig. 1. An industrial robot has a mechanical part (containing one or more manipulators) and a control system for this mechanical part. In addition, the robot may have sensing means (forming together an information-sensory system), whose signals are fed into the control system [4, 5].

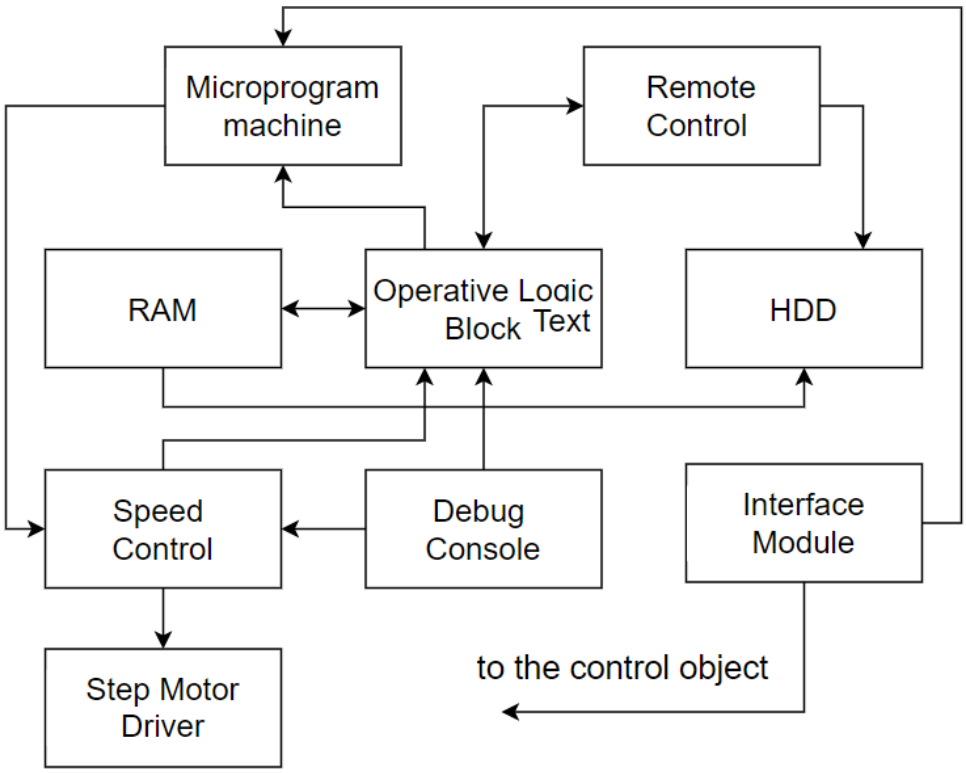

Fig. 1. Functional diagram of an industrial robot

As a rule, the executive mechanism of a manipulator is an open kinematic chain, the links of which are connected to each other in series by different types of joints. The combination and mutual arrangement of links and joints determine the number of degrees of freedom and the scope of the robot's manipulation system. It is usually assumed that the first three joints in the executive mechanism of the manipulator implement transport degrees of freedom (ensuring the presentation of the working body to a given location); the others implement orientation degrees of freedom (being responsible for the necessary orientation of the working device).

The structural diagram of the software is shown in Fig. 2.

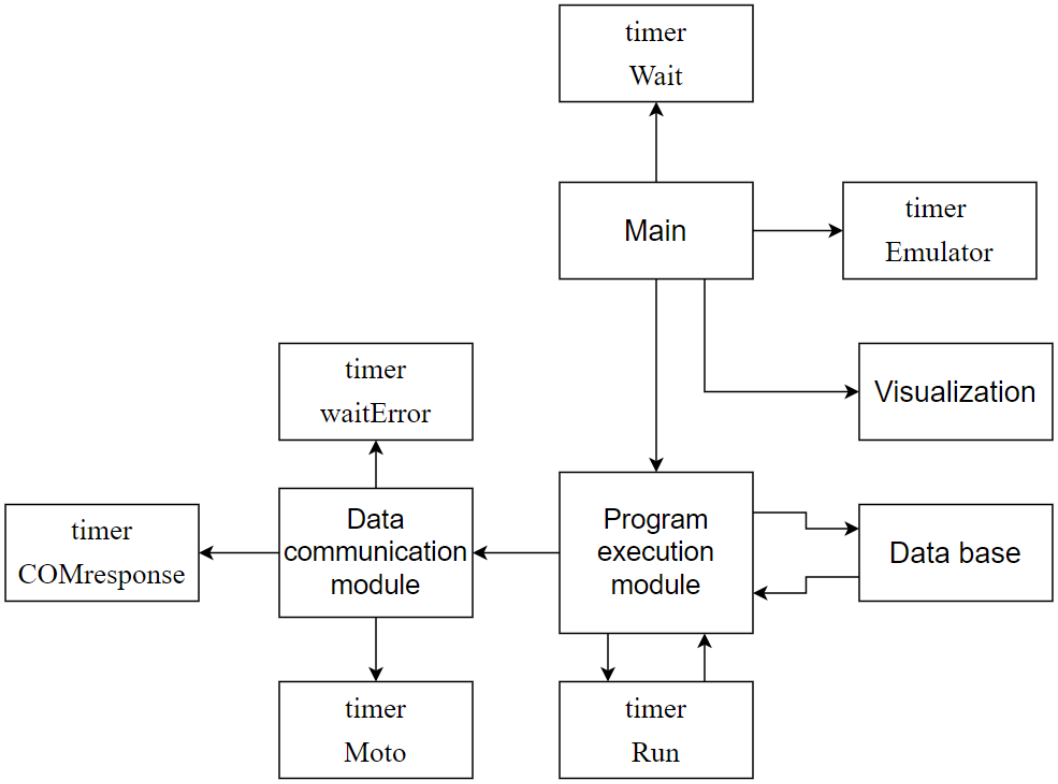

Fig. 2. The structural diagram of the software mean

The basic modules of the program are: the main module, the program execution module, the data communication module, the database module, and the visualization module.

There are also six timers in the program that allow to implement independent control flows of the program.

The program execution module organizes the operation of the instructions stored in the database. The appropriate module is engaged to retrieve the next command from the database. Each command is selected sequentially. The queue controls an independent flow, for which the timer_Run timer is responsible. When the next command is received, a command in G-Code format is prepared.

The following commands are involved in the program: positioning tool G0, turning on gripper M5, turning off gripper M3, turning on pump M1, turning off pump M2, turning off laser M6, turning off laser M7, turning on motor power M17, turning off motor power M18, manipulator calibration G28.

The visualization module is used in the program emulation mode to visually control the position of the manipulator links to facilitate setting the control program.

Each timer works independently. Synchronization is done with the corresponding states stored in the status_exec variable.

The operation of the program is organized using independent execution flows. Corresponding timers control each flow: timer_Moto, timer_Run, timer_Emulator, timer_Wait, timer_COMresponse, timer_waitError.

The timer_Moto is designed to cut off the power to the stepper motors if the manipulator is not used for a long time. The stepper motors used in the construction have no mechanical brakes. Therefore, they can lower the manipulator links to the bottom end position when the load turns off the power. In order to hold the cargo in the set position, the stepper motor drivers are provided with the function of switching on the holding current, which is 50% of the basic working current. This is done to prevent excessive heating of the motor windings.

However, some drivers do not have a hold current reduction function. In any case, to reduce the power consumption of the whole design it is appropriate to power down the motors if they are not used for a long time. By default, the timer_Moto interval in the program is set to 10s. This value is automatically updated with each command sent.

Fig. 3 shows a state conditions diagram explaining how to use the timer_Moto timer.

The main program flow calls the sendCommand function whenever an instruction needs to be sent to the manipulator control module. This function restarts the timer by setting a time interval of 10,000s.

When the wait time expires, it checks if there is communication with the serial data transmission port. If there is no communication, a message is displayed on the bar form status and the timer is turned off.

If there is communication, the corresponding G-Code is transmitted, which causes the power to the stepper motors to be turned off. The timer is then turned off and control is transferred to the main program flow.

The timer_Run timer is responsible for calling commands from the list that makes up the manipulator control program (Fig. 4).

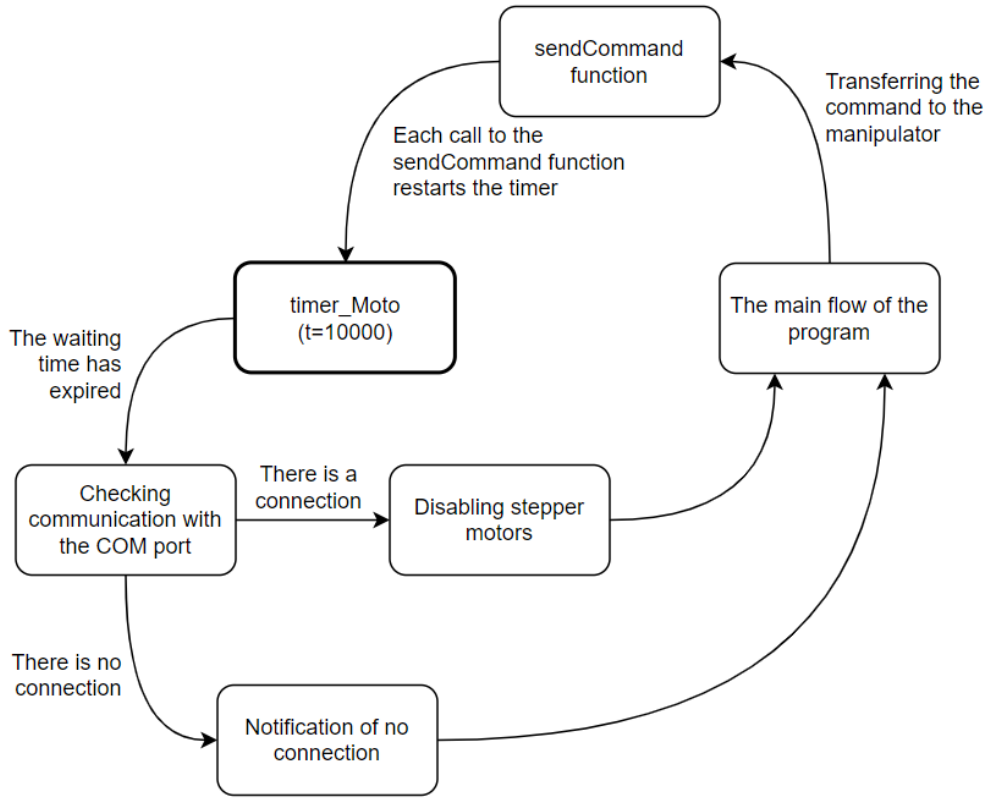

Fig. 3. Principle of timer_Moto usage

First, the current state of the status_exec variable is checked, which can have the following values:

– «OK» – the previous command is completed correctly, the application is in waiting state.

– «GET_OPERATION» – the program is in the state of searching for a new command.

– «EXEC_OPERATION» – the program is in the state of executing the current command.

– «STOP» – the program is in the state of completion.

A diagram of the state of the program execution process is shown in Figure 5.

A characteristic feature of the proposed method of executing commands in the program is the concept of folders. Folders in this case are a grouping of commands that make up a certain cyclic sequence of actions for the manipulator. Folders are not a visual component, but the essence, uniting several commands, executed one after another, in a database. Every program has at least one folder, which is the main folder. This structure is created at the beginning of the job and contains all the commands in the program.

Cycles can be used in the program. When a cycle is created, a new folder is also created in which all the commands that make up the body of the cycle will be

placed. Folders can contain subfolders, which in turn can also contain folders. Entering a folder means that the sequential course of commands is interrupted and a new one becomes current - the cycle of commands entering the new folder.

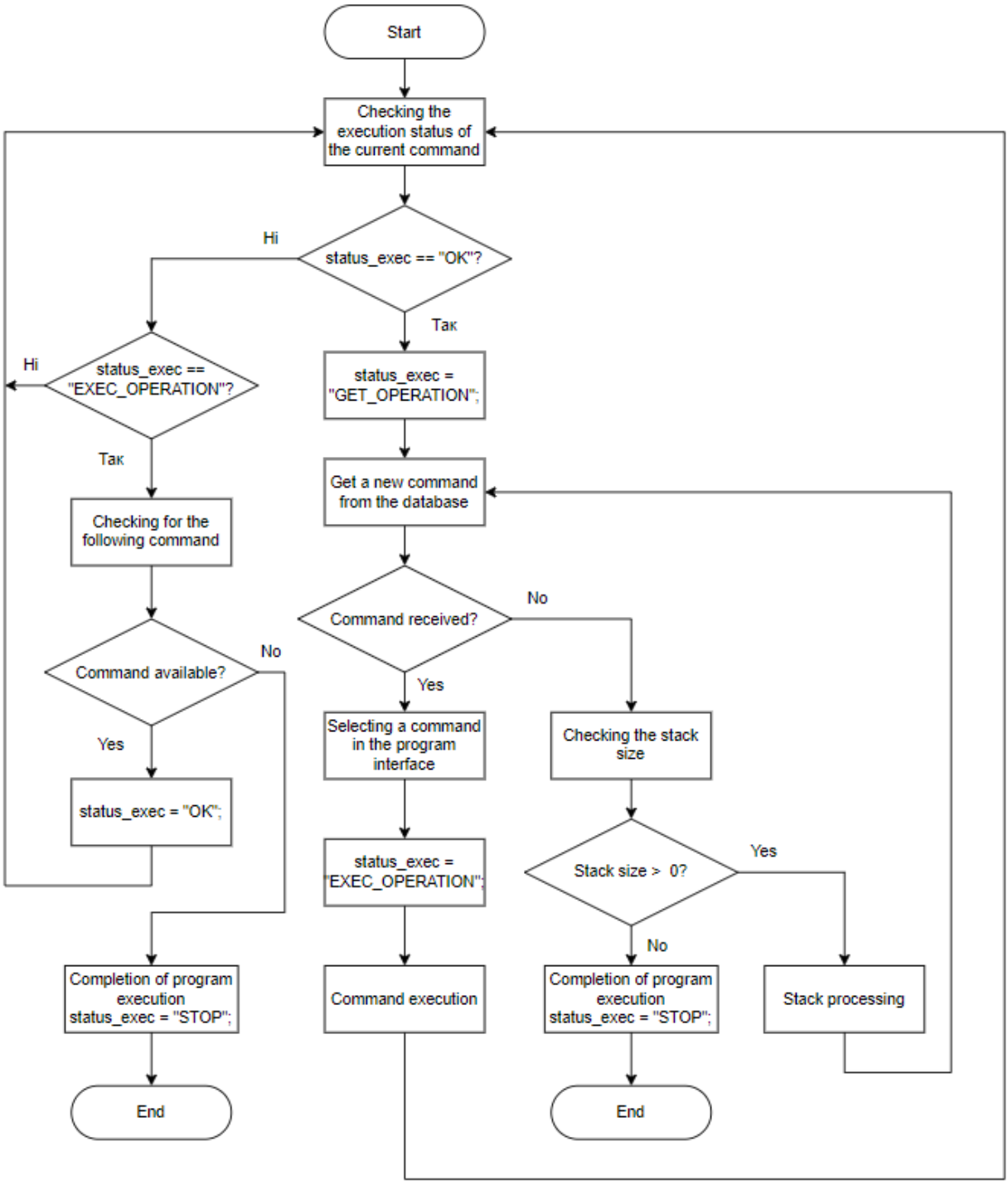

Fig. 4. Algorithm of the program pointer processing method

In order to be able to return to the previous command cycle, a stack is provided. The current program counter number is entered in the stack, and the ID of the command first in the new branch (the first command in the new folder) is passed to the counter. At the end of the cycle, the size of the stack is checked.

If it is greater than zero, the last entry, which is the return point to the previous cycle, is taken from it.

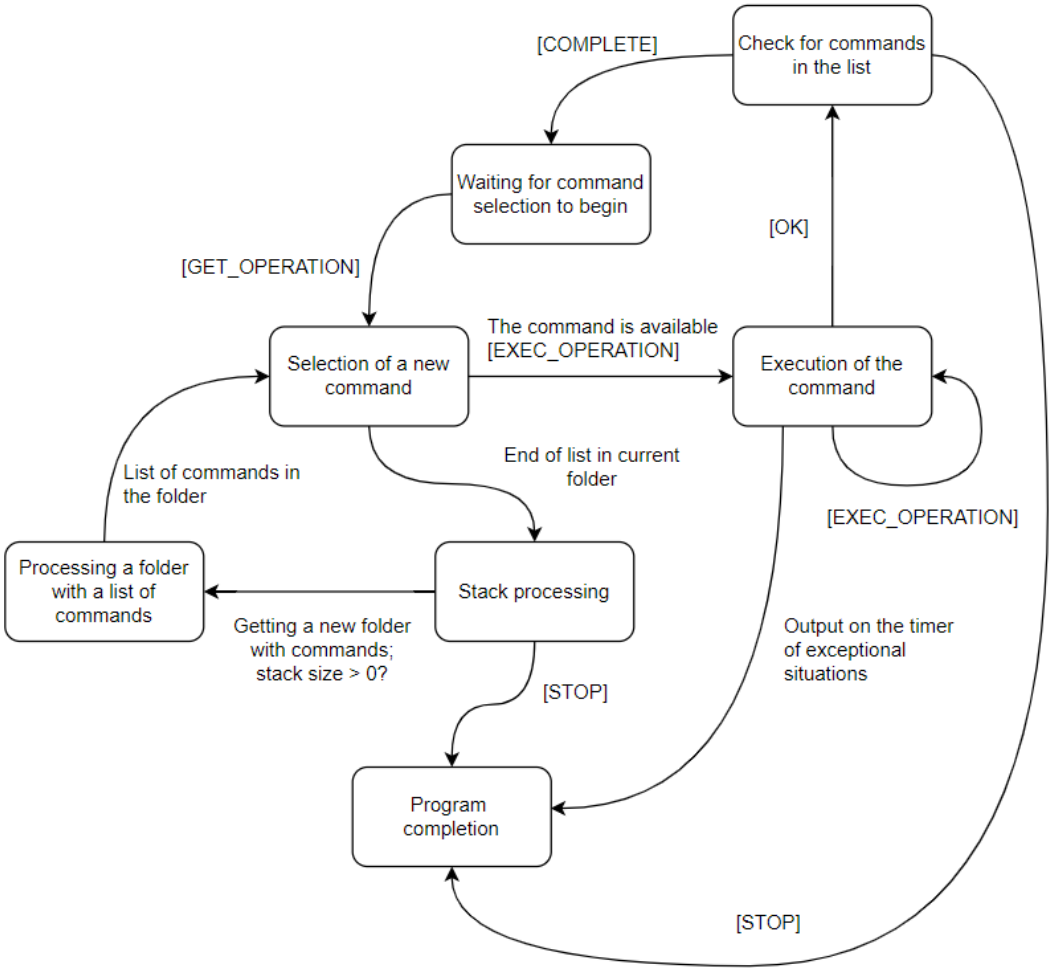

Fig. 5. State diagram for the program execution process

After getting the stack number, we call the search method in the database to find the identifier of the folder that contains the received command. After getting the folder number, a list of all commands in it is generated and a new value of the program counter is determined: cur_num_operation_exec++.

If the program counter number is out of the acceptable folder value range, the stack size is checked again. If it is equal to zero, the application is considered completed and the corresponding procedure with the value «status_exec = «STOP» is started.

If the stack size is not zero, the program execution continues according to the algorithm (Fig. 4) and the state diagram (Fig. 5).

The timer timer_Emulator runs only in emulation mode. Its purpose is to produce a delay of 500 ms on each command responsible for the movement of the manipulator links. This is done to visually assess the correctness of the commands and the sequence of the commands.

It should be noted that timer_Emulator does not run when the manipulator model is rendered on the computer screen. In this case, the sign of the completion of the manipulator link movement command is determined in real time if the virtual tool reaches a given point in space.

The timer timer_COMresponse is designed to organize the timing of the command execution by the manipulator control module. With the help of this timer after sending the Move command with the coordinates of movement of the working tool the current time of its execution is displayed. The timer stops after receiving a move confirmation from the control module in the form of one of the response options:

– «HOMING COMPLETE» – it is received when the calibration of the manipulator is completed.

– LINEAR MOVE» – accepted when the manipulator has finished moving to the set position.

– «OK» – accepted in case of completing the work of the manipulator's executive tool.

In all of the above cases, the current state of the curr_action variable changes to OK, which is an indication that the current state of the program is complete.

The timer timer_Wait is designed to delay the execution of instructions and the «Wait» command of the same name. When the timer_Wait command is executed, the argument of the current instruction is transmitted as a millisecond value. The PC screen displays the message «Program Execution Delay for XX ms». When the program delay is over, an acknowledgement curr_action=«OK» is generated.

The timer timer_waitError is used to generate an error message when the response from the manipulator control module is too long.

Development of a motion control program for an angular type manipulator using visual components

To implement the project of manipulator control and implementation of production equipment, it is necessary to present a functionally complete product that can work without connecting to the Internet. Therefore, to create a database, DBMS SQLite was chosen.

To store the project data, the table «Project» was developed. The project is created at the beginning of the work and to it are tied to all of the subsequent entities. One of the main entities is «Program». This entity is a description of the

manipulator control program, specifying all operators and arguments transferred to them during the work.

The «SpecialPoint» table is used to store operational data about special points that are frequently used in a project. Such points can be the initial reference coordinate or the final coordinates of the working tool movement during the manipulator operation.

The SetupDevice table is needed to store the settings of the manipulator motion control program. Fig. 6 shows the database structure.

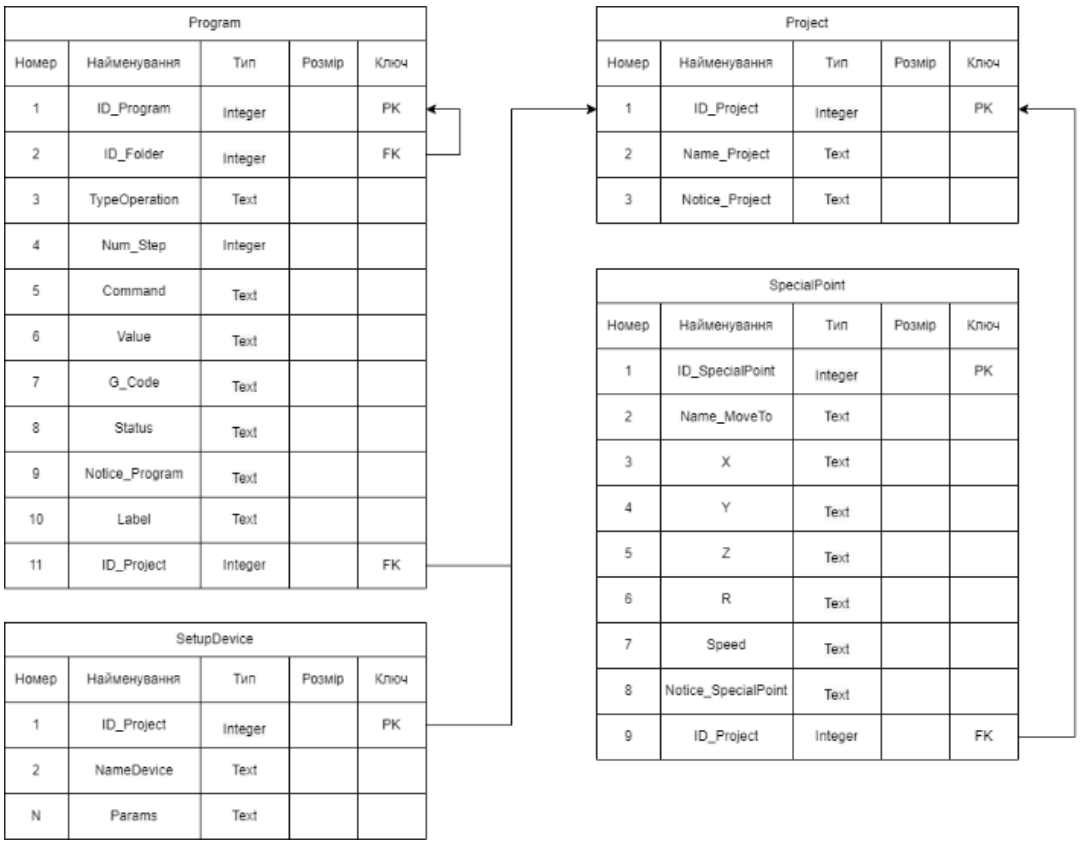

Fig. 6. Database structure

This figure shows the SetupDevice table in abbreviated form. The Params line shows that there are several fields in the structure to store the necessary parameters for the program to work. All tables are linked by the ID_Project field. When a new project is created, a new database file of the specified structure is created.

Fig. 7 shows the interface of the program. The left side of the interface is used to place commands to control the manipulator. A separate line that can be dragged and dropped within the program segment, thus changing the course of the program, represents each command. In the right side you can change parameters of each command. The contents of this area of the interface depend

on the specific instruction. There are a total of six different types of instructions: Move To, While, Repeat, IF, GoTo, WaitInput.

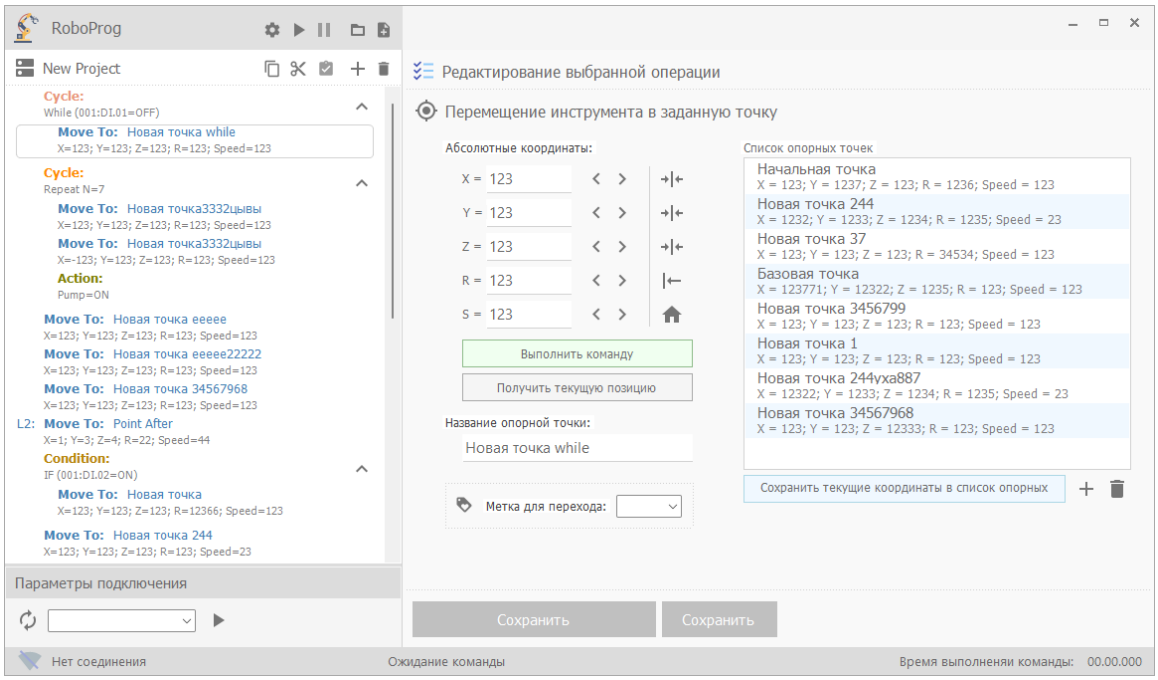

Fig. 7. The program interface and an example of entering the «Move To» command parameters

The Move To command is used to move the manipulator tool to a given point in space. This command requires the following parameters: move *X* coordinate, move *Y* coordinate, move *Z* coordinate, move *R* coordinate, move *S* speed. The *XYZ* coordinate specifies the position of the tool's end point in 3D space. The *R* coordinate is the parameter that defines the displacement of the manipulator along the rail to which it is attached (if any).

The emulation mode provides a list of reference points. This list allows you to remember the key positions of the manipulator, often repeated during its operation. This list can be updated during the program operation. Points can be both added to and removed from the list. All of them are stored in the database table. Each point can be given a unique name to make it easy to identify during work.

Also in this mode it is possible to move the manipulator to a given point without executing the whole program. Thus, the operator has the opportunity to debug each step of the program during its creation in real time.

In order to read from the memory of the manipulator data about the current state of its links and automatically fill data fields with corresponding real coordinates a special button «Get current position» is provided.

The program also has a GoTo label. This parameter is used to assign a unique identifier to a command, which is used to unconditionally jump to it from any line in the program when using the GoTo statement.

The GoTo command allows you to jump from one command to another using a unique identifier that begins with the letter L. There are nine possible identifiers: L1...L9 respectively.

Using the command is simple enough: you must select the appropriate letter from the drop-down list. After that the instruction will be written in the command, for example, GoTo: Label=>L1.

The While command is one of two kinds of cycles. This cycle pauses the execution of the program until the specified condition is met. As a condition, it is implemented to wait for the set signal level to be set on the specified port of the industrial controller.

You can select digital or analog input type. In the first case you can select the input state either ON or OFF. If the input type is analog, you can set any value from –35565 to 35565. Also for the analog signal you can set the comparison signs «=», «>» and «<». In all cases you must specify the device address in modbus protocol format and the number of the digital or analog contact in the PLC.

The Repeat cycle type allows you to specify the number of repetitions for all the commands contained in the cycle body. All commands included in the cycle body are included in the virtual folder that joins them. Each command attached to a cycle can be moved by means of visual editing only within the given cycle. The command can be moved out of the cycle only by executing the «Cut» and «Paste» commands. All commands within a cycle are stored in the database with a reference to this cycle in the ID_Folder field.

The IF command is a conditional statement and is another representative of containers. This command decides whether a sequence of other commands included in the container body will be executed or if they will all be skipped. The conditional While command works directly with the PLC inputs. The ports are accessed via RS-485 network using modbus protocol or via Ethernet network using modbus TCP/IP. The condition of the IF command execution is the specified state of the discrete or analog inputs of the PLC.

The Wait command allows you to implement one of two delays: a time delay or a delay until a specified value is set on the specified PLC input.

When debugging a program to control a manipulator, animating the movement of its links and working tool from one point to another is very useful. Usually the task of 3D visualization requires the use of additional libraries

and computing power. When developing the program it was decided not to use additional libraries, and to calculate the nodal points of the manipulator by means of C# in real time, using the laws of inverse kinematics.

To simulate the position of manipulator links and visualize their motion two views of the device are used: the top view and the side view (Fig. 8).

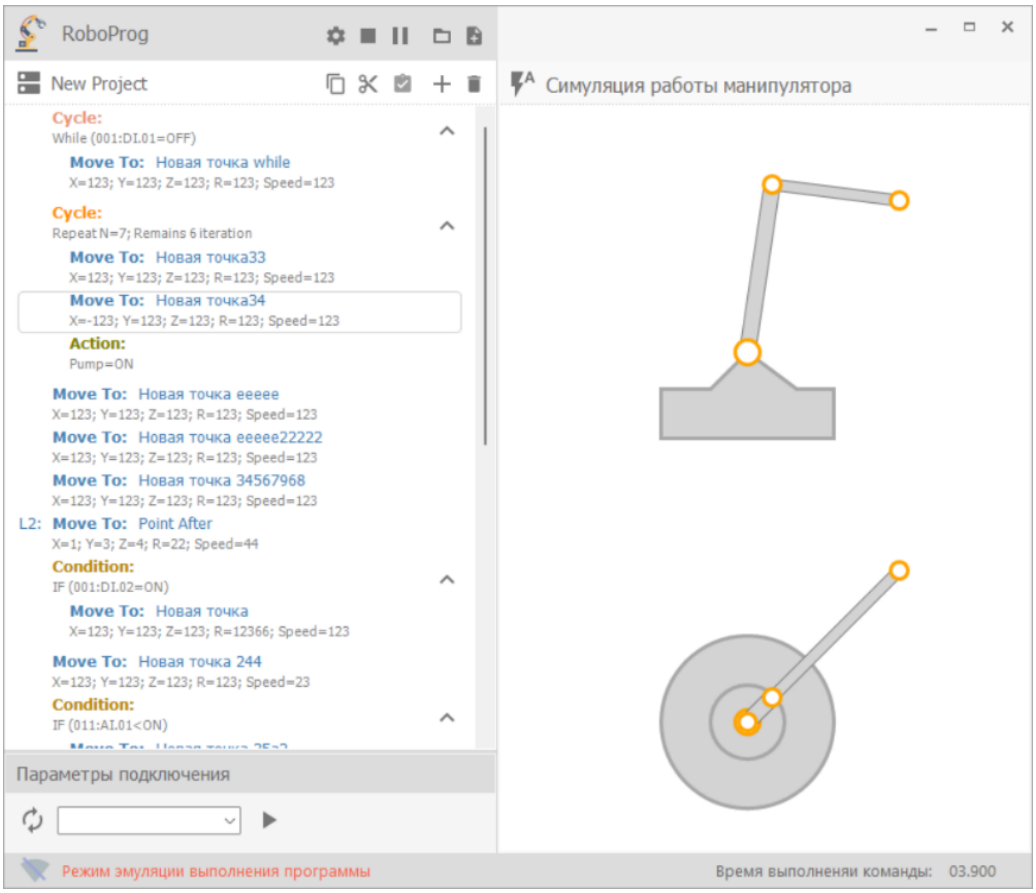

Fig. 8. Program interface in the manipulator simulation mode

These two views allow to estimate manipulator's motion without using isometric view of coordinate system. The difficulty in solving the problem is that the manipulator can rotate around the vertical axis, so the side view will be a transformed view of the vertical plane of vision.

Conclusion

This paper describes the developed technology of program control of robotic manipulator, describes the developed language of visual programming of industrial manipulator using multiflowed control of the processing of control commands. The proposed technology uses a set of independent timers, allowing to realize independent control flows of the program execution process. The process of task

distribution between the flows and synchronization algorithm are described. The control object is a model of a robotic manipulator. The manipulator has two movable joints and can rotate around the vertical axis. In addition, the manipulator has a gripper for gripping and moving parts within its working area. The peculiarity of the developed method is that no additional libraries are used, and the calculation of the nodal points of the manipulator is performed by means of the selected programming language in real time using the law of inverse kinematics.

References

1. S. Nair, A. Rajeswaran, V. Kumar, Ch. Finn, A. Gupta, «R3M: A Universal Visual Representation for Robot Manipulation», arXiv:2203.12601v2 [cs.RO], 18 Apr. 2022.
2. S. Novoselov and O. Sychova, «Using Wireless Technology for Managing Distributed Industrial Automation Objects within the Concept of Industry 4.0», 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 2019, pp. 580–584, **DOI**: https://doi.org/10.1109/PICST47496.2019.9061333
3. S. Novoselov, O. Sychova and S. Tesliuk, «Development of the Method Local Navigation of Mobile Robot a Based on the Tags with QR Code and Wireless Sensor Network», 2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), Polyana, Ukraine, 2019, pp. 46–51.
4. Nevludov, O. Sychova, A. Andrusevich, S. Novoselov, D. Mospan and V. Mospan, «Simulation of the Sensor Network of Base Stations in a Local Positioning System in Intelligent Industries,» 2020 IEEE Problems of Automated Electrodrive. Theory and Practice (PAEP), Kremenchuk, Ukraine, 2020, pp. 1–6.
5. Ostanin, Mikhail & Mikhel, Stanislav & Evlampiev, Alexey & Skvortsova, Valeria & Klimchik, Alexandr. (2020). Human-robot interaction for robotic manipulator programming in Mixed Reality. 2805–2811.
6. K. Inoue, Y. Nishihama, T. Arai and Y. Mae, «Mobile manipulation of humanoid robots-body and leg control for dual arm manipulation», Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), 2002, pp. 2259-2264 vol.3, **DOI**: https://doi.org/10.1109/ROBOT.2002.1013568
7. S.-J. Yi, «Software Framework for an Intelligent Mobile Manipulation Robot», 2018 International Conference on Information and Communication Technology Robotics (ICT-ROBOT), 2018, pp. 1–4. **DOI**: https://doi.org/10.1109/ICT-ROBOT.2018.8549895
8. B.-V. Mauricio, E. Belo, «Application of H∞ theory to a 6 DOF flight simulator motion base». Journal of the Brazilian Society of Mechanical Sciences and Engineering. 2012, Volume 34, pp. 193–204. **DOI**: https://doi.org/10.1590/S1678-58782012000200011
9. Wu, L.; Zhao, R.; Li, Y.; Chen, Y.-H. Optimal Design of Adaptive Robust Control for the Delta Robot with Uncertainty: Fuzzy Set-Based Approach. Appl. Sci. 2020, 10, 3472. **DOI**: https://doi.org/ 10.3390/app10103472