Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 100 (2015) 33 – 40

Procedia Engineering

www.elsevier.com/locate/procedia

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

Technology Supervisory Control for Mechatronic Devices via the Internet

V.P.Andreev[a,b,d]*, K.B.Kirsanov[b,d], P.F.Pletenev[a], Yu.V.Poduraev[a],V.E.Pryanichnikov[a,b,c,d,e], E.A.Prysev[b,d]

[a]*MSTU "STANKIN", Vadkovsky lane 1, Moscow, 127994, Russia*
[b]*International Institute of New Educational Technologies, RSUH, Miusskaya sq. 6, Moscow, 125993, Russia*
[c]*Keldysh Institute of Applied Mathematics, RAS, Miusskaya sq. 4, Moscow, 125047, Russia*
[d]*International Laboratory "Sensorika", Miusskaya sq. 4, Moscow, 125047, Russia*
[e]*University of Zadar, Mihovila Pavlinovica bb, Zadar, 23000 Croatia*

Abstract

This paper presents the results of research on the development of technology for supervisory control of mechatronic devices through the Internet. We propose a hierarchical structure for the management of a group of mobile robots (a group of mechatronic devices) by combining their information-measuring and control systems into a local area network. The main requirements to the functional structure of these systems are formulated. The development of a spatially distributed control system involves a number of problems associated with peculiarities of the use of digital radio and Internet channels. These problems can be treated by creating specialized software and hardware tools, such as the spatially distributed scientific and educational Internet-laboratory proposed in this paper. The laboratory can be used to organize full-fledged access to specific mechatronic systems of different models and different producers through the Internet. The main requirements for the implementation of this laboratory are formulated through the use of the "concept of drivers". The architecture and technical implementation of the pilot project of this laboratory are described. The driver concept designed for incorporating the mechatronic devices into the structure of the laboratory is demonstrated by the example of the AMUR and Festo Robotino robots.The work was supported by the Russian Academy of Engineering, project "Intelligent Robotronics" and partially by the Russian Foundation for Basic Research, project nos. 13-07-01032 and 13-07-00988.

Keywords: mechatronic device; robotic system; local area network; spatially-distributed information-measuring and control system

Corresponding author. Tel.: +7-965-210-79-51.
E-mail address: andreevvipa@yandex.ru

1. Introduction

Robotic systems are diverse due to their functionality (industrial, rescue, medical, military, entertainment, consumer, scientific, educational, etc.), specific application features (ground-based, airborne, above-water, underwater, space-based, etc.), and motion type (wheeled, track, floating, flying, walking, creeping, etc.). There are many companies producing such systems. The diversity of even single-purpose mechatronic systems produced by the industry of different countries leads to the problem of training the specialists working with these systems. In view of rather high costs of these technical systems, the problem of choice arises: one has to try many devices to make sure that a given device is suitable for solving the required problems within available financial resources.

In our opinion, these problems can be solved by creating special-purpose software and hardware tools. In this work, we propose a spatially distributed scientific and educational Internet-laboratory that can be used to organize full-fledged access to specific mechatronic systems of different models and different producers remotely through the Internet. The special network interaction mechanism implemented in this laboratory allows third-party engineering and scientific manpower to be engaged in new designs independent of their location.

2. Current status of the problem

At present, there have been many application and system software solving the formulated problem only partially. Among the most significant frameworks, we can mention the Robotics Operating System (ROS) [1] and Microsoft Robotics Developer Studio (MRDS) [2]. These software packages make it possible to efficiently integrate various mechatronic components into a unique control system but only within a single laboratory. These packages are not independent on programming language and hardware-software platform (MRDS) and ignore the problems of spatially distributed control and remote visual observation of the execution of control commands. The use of the Internet as a communication channel requires logging mechanisms with synchronization of events and tracking of time delays. Both ROS and MRDS have no such mechanisms. The use of radio channels in mobile robotics creates problems in providing the system resilience to communication failures; the software mentioned above does not address this problem. In the spatially distributed system, when the management of "alien" mechatronic devices is permitted, it becomes very important to have a mechanism for authorization rules, which cannot be found also in ROS and MRDS.

A project with similar activities was performed in 2009–2011 jointly with the Institute of Robotics and Mechatronics of the German Aerospace Center (Institut für Robotik und Mechatronik, Deutsches Zentrum für Luft- und Raumfahrt, DRL-RMC), the Russian State Scientific Center for Robotics and Technical Cybernetics (RTC), and the Rocket and Space Corporation *Energia* (Russia). This project examined the use of the Internet for managing ROBOTIC, a two-section robot mounted on the external surface of the Russian Orbital Segment of the International Space Station (ISS). Communication sessions with the ISS board were conducted from both DRL-RMC (Germany) and RTC (St. Petersburg, Russia) through DRL-RMC using the Internet for the communication between RTC and DRL-RMC. A characteristic of this project was the use of human–machine interface with torque/force feedback. This project considered merely the impact of delays in communication channels on management [3, 4].

3. Hierarchy principle of supervisory control of a group of mobile robots

In [5] we described the conception of building an information-measuring and control system (IMCS) of mobile robots (MRs) with supervisory control. Analyzing the information flows in IMCSs of mobile robotic systems and the functions for processing these flows, we formulated the main requirements to the structure of these systems:

- the electronic components of IMCSs must be elements of a local area network (LAN) with mobile knots;
- the mobile and stationary units of these systems must be linked through digital radio channels (Wi-Fi, Wi-Max, etc.);
- increased range and improved quality of radio communication should be achieved by creating an optimal, dynamically changing architecture of the distributed IMCS and combining the parameters of antennas and remotely configurable devices that form the communication channel;
- the computer vision (CV) of the robotic system should consist of at least two video cameras: one or more onboard the MR and one or more on remote satellite devices (stationary or mounted on mobile platforms);
- the CV of the robotic system should be controllable: one or more video cameras must have the PTZ (**P**an, **T**ilt, **Z**oom) functionality with control through digital radio channels;

- the video signals and signals from other sensors must be preprocessed onboard the MR (filtering of noises, correction of distortions, compression of images, etc.);
- the state of the power system and the radio-channel quality must be controlled remotely (from control panel);
- the operator control panel must ensure: a continuous mapping of all video streams as raster images; a mnemonic mapping of the state of radio channels, power systems of mobile units, and readings of other sensors on display screens; a remote control of camera functions and parameters of all LAN units.

Based on the above-listed provisions of this concept, we use a hierarchy principle of supervisory control of a group of mobile robots [6]. As an extension to this principle, Fig. 1 demonstrates the 4-level structure of the spatially distributed IMCS built on the basis of a unique heterogeneous local area network.

The group of robotic systems (for example, the group of MRs operating in the same area), each with an IMCS constructed as a local area network, is combined into a larger LAN through communication channels [7, 8]. The **first** level of hierarchy includes the LAN of the mobile robot itself with wired digital channels (onboard IMCSs). The **second** level includes distributed IMCSs of individual robotic systems organized as a combination of the onboard LAN and the LAN of the control station (one robot to one operator) through the use of digital radio channels. The **third** level includes the control system of the upper (command) level coordinating the operation of several operators. For this, LANs of individual robotic systems should be combined into a single LAN (a local area network with mobile knots) through an appropriate communication channel, thus providing a link between the commander and operators of robotic systems (digital radio or wired channels). As a result, we have a distributed IMCS of the group of MRs (a group of MRs includes working robots, escort robots – Satellites, control stations, etc.). This combination makes it possible to control simultaneously the operation of multiple robots equipped with different tools and ensure that the distributed IMCS is reconfigurable and scalable. The next, **fourth**, level of hierarchy creates a spatially distributed IMCS by combining the LANs of individual groupings of MRs and the LAN of the Situation Center into a single LAN through VPN-tunnels of the Internet. As a result, we have a spatially distributed robotic IMCS as a combined heterogeneous local area network.

As shown earlier [5], the second and third hierarchy levels can be most efficiently organized on the basis of Wi-Fi and TCP\IP. The networks at the third and fourth levels are combined through VPN networks that encapsulate TCP\IP protocols and unite spatially distributed knots. The technical implementation of this structure involves no great difficulties because the given technologies and protocols are established technologies and have many variants of the software and hardware design on the market.

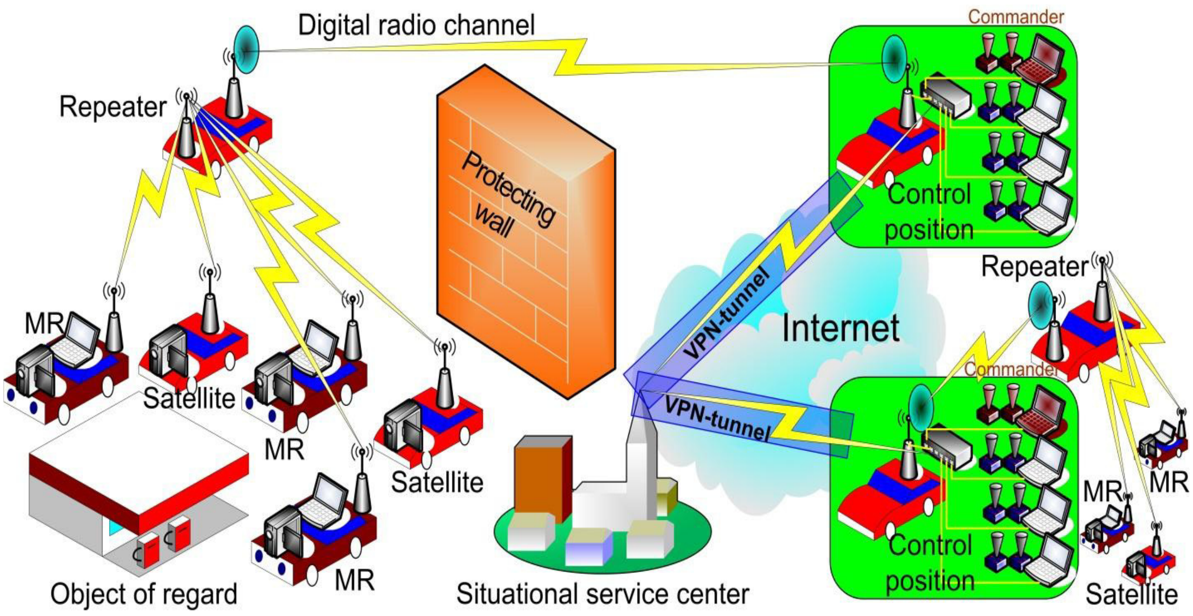


Fig. 1. Spatially distributed IMCS as a unique heterogeneous local area network.

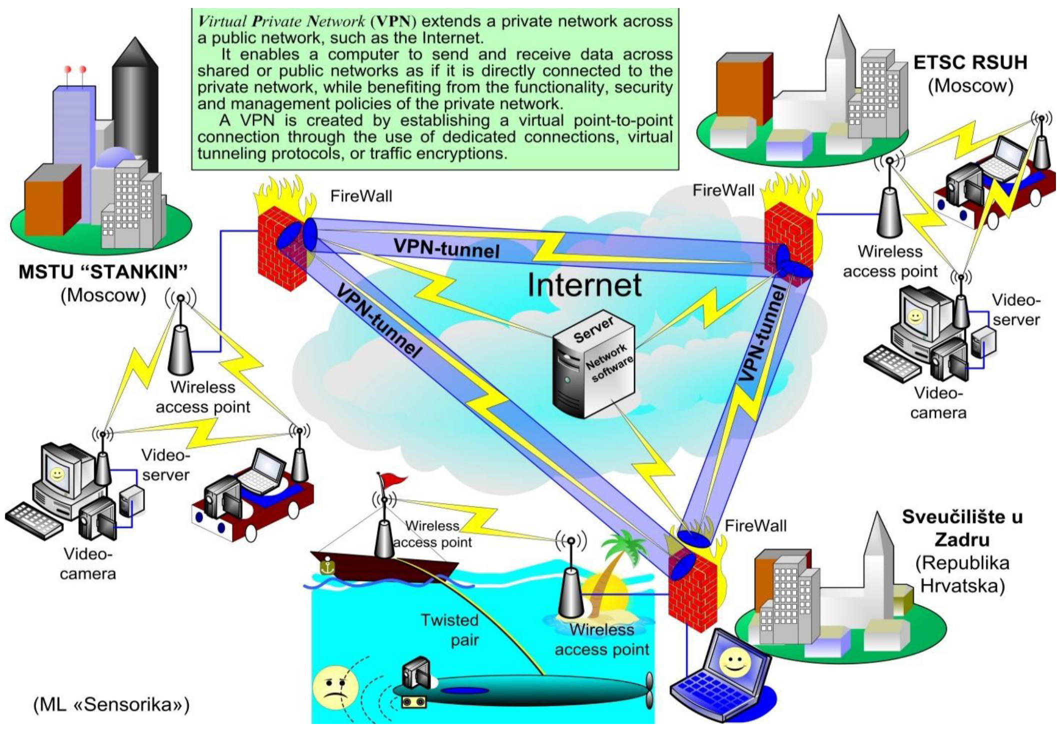
4. Spatially-Distributed Scientific and Educational Internet-Laboratory

The proposed approach to the robotic system as a LAN with mobile knots makes it possible to create a spatially distributed educational and scientific Internet-laboratory combining the software and hardware systems of robotic devices into a unique educational environment (Fig. 2).

This laboratory must meet a number of requirements. The key requirements are:

- versatility – robotic and other mechatronic systems of various models and manufacturers can be included into the laboratory;
- reliability – the packets between control interfaces of the robotic system can be delivered at all levels of the network interaction;
- integrity – there is a mechanism for managing the access rights and delimitation of authority between members of the educational process;
- flexibility and scalability – the network structure is has a decentralized mechanism of management and can be reconfigured online;
- security – protection from unauthorized access to control interfaces of the robotic system.

A minimal control object in the structure of the laboratory is a mechatronic device such as an onboard microcontroller, a manipulator, or even a sensor. The combination of several such mechatronic devices (components) is usually called a robot. One of the key problems is to develop a unified system of rules for creating rules of software for specific models of robotics devices, allowing their mechatronic components to be incorporated into the LAN of the laboratory without modifying the low-level program code of the robot. In other words, a finite set of control instructions and network protocols is required to build an application program interface (API). This interface must ensure a transparent inclusion of the elements of mechatronic devices into the information space of the laboratory.



Virtual ***P****rivate* ***N****etwork* (**VPN**) extends a private network across a public network, such as the Internet.

It enables a computer to send and receive data across shared or public networks as if it is directly connected to the private network, while benefiting from the functionality, security and management policies of the private network.

A VPN is created by establishing a virtual point-to-point connection through the use of dedicated connections, virtual tunneling protocols, or traffic encryptions.

Fig. 2. Logical scheme of a spatially-distributed scientific and educational Internet-laboratory with decentralized control.

The API is based on the “concept of drivers”: each mechatronic device has a unique network control protocol on the basis of existing low-level program interfaces [9]. Then, the manufacture of mechatronic devices can relatively easily integrate its device into the laboratory network by creating only an additional driver that implements deterministic program instructions of control and network interaction protocols. When a new mechatronic device enters into the network, the software of the laboratory management subsystem during the interaction with the driver automatically connects this device to the shared network and issues data on available mechatronic devices in the general registry of technical details.

This subsystem is based on well-proven network technologies and solutions with synchronous and asynchronous communication on the basis of TCP/IP protocols and has the following components:

- protocols of synchronous and asynchronous exchange of message;
- unified registry of services;
- operator program module.

According to the OSI model classification [10], the interaction occurs at the application level and can conditionally be divided into 2 sublevels: *transport* and *logical*.
The *transport* sublevel must meet the following requirements:

- be of high performance in transmitting both long messages (megabytes), for example, multistream video and short messages (bytes) – control commands;
- have immunity to the disconnection, for example, due to the unstable operation of the radio channel;
- have an opportunity of the organization of one-to-many interaction, which is necessary for group control tasks.

The *logical* sublevel must meet the following requirements:
- be cross-platform and multi-language – the ability to develop software for different operating systems (Windows, Linux x86\64\arm, Android, MCST Elbrus) and languages (C, Java, Python, Fortran);
- human-readability – the capability to explore transmitted data without using specific software tools. The data are transmitted as plain text, except when this is critical for productivity (for example, video);
- possibility of isolated operation – the operability of interacting robots or mechatronic devices involved in the LAN of a given hierarchy level should not be disrupted when disconnected from a higher level network that is external to them;
- decentralization and capacity for self-organization – no need for any "central" server regulating the operation of all components of the system, except one wants to limit access to devices and register the users.

The exchange of messages is based on the ZeroMQ library [11]. This software fully meets the stated requirements. An additional benefit is that this library is licensed under GNU LGPLV3, i.e. is a free software, which allows it to be used in our designs [12, 13].

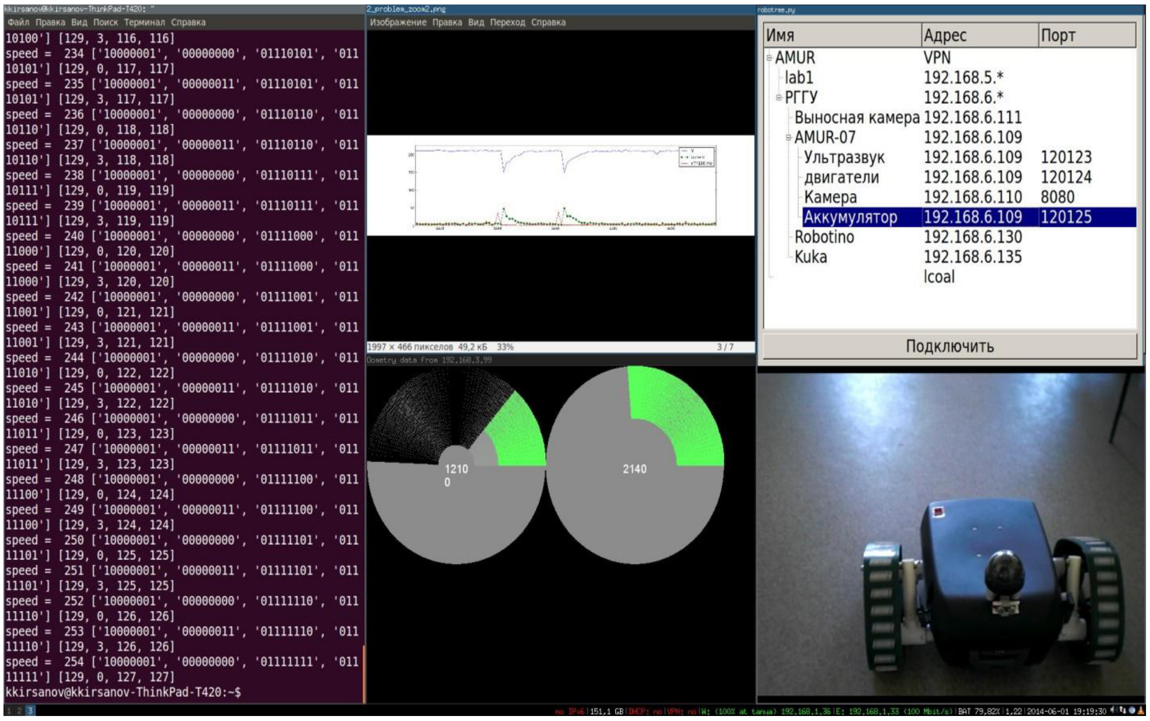
The architecture of the laboratory involves the so-called *unified catalog of services*. By a service, we mean any mechatronic device or its part that can be somehow dealt with through actions/operations and that has deterministic characteristics. For example, this can be a TV camera on the mobile root. In this case, an action means that it can be connected to the transmitted video stream and characteristics mean connection parameters and video stream parameters. The set of all possible actions/operations and characteristics are called the *profile* of the mechatronic device. Accordingly, the catalog contains (in the real-time mode) the profiles of all mechatronic devices that are currently connected to the laboratory. When several laboratories are combined into a single network, the local catalogs of services are synchronized with each other.

The technical implementation of the unified catalog of services is performed using well-known infrastructural solutions (DNS with dynamic update capabilities and LDAP) installed on the shared server (see “Server” in Fig. 2). The DNS server provides a transparent registration of a new mechatronic device in the network by using unique addressing within the given laboratory. The LDAP server issues and updates data on available mechatronic devices in the catalog, synchronizes the catalogs with each other, and provides users (operator program module) with information about mechatronic devices through LDAP. The LDAP server makes it possible to easily construct a hierarchic role model for access to the mechatronic device. Thus, the robot “owner” can deliver the right to manage the robot only to certain users, while the connection to the TV camera or other sensors can be delivered to all users who want to use that resource. Also, LDAP can provide a scheduled access to devices.

The access to mechatronic devices is provided through the *operator program module*. This module makes it possible to observe the available mechatronic devices, to connect to a given device to execute operations (for example, controlling), and map the data stub from available sensors. To adequately map the hierarchical network structure on the graphical user interface, a window-hierarchical interface is designed on the basis of approaches applied in frame (mosaic) window managers. The workspace of the screen is split into mutually non-overlapping rectangular areas (while these areas in traditional "window" systems may overlap and one window can cover another). Each area is used by a separate component of the system to display data and is automatically scaled to a desired size. Figure 3 shows an example of one of the levels of such a window-hierarchical interface.

The upper right corner of the interface displays a tree representing the hierarchy of mechatronic devices and robots found in some laboratory. By mouse-clicking on the laboratory, a group of windows appears with brief information about each component. If clicking on a given device, its frame is shown in full, and the remaining frames are automatically scaled to the minimum-possible size. This makes it possible to avoid the need to deal with dozens of windows to fix the required one. Each robot, laboratory, or group of laboratories is attached to a typical set of windows, which, nevertheless, can be changed during the system operation. In this example, the windows display the devices that are available in the AMUR laboratory and can be accessed with the help of an appropriate driver (a remote camera and KUKA, Robotino, and AMUR-07 robots). Some components of AMUR-07 can be accessed at the management and/or control level: the left window displays the PWM log of the robot engines, the top center displays the time sweep of current and voltage values, the bottom center displays the odometry readings, and the bottom right displays the AMUR-07 robot image received continuously from a remote controlled TV camera.

The subsystem of the laboratory provides each member with access to services both through graphical interface and using APIs. This solution can be easily scaled making it possible to combine several laboratories into a single informational and educational environment. This possibility can be accomplished only by creating VPN-tunnels between laboratories that are equipped with appropriate technical tools. These VPN-tunnels can be created on both bilateral and multilateral bases if relevant agreements are concluded between organizations. The inclusion of "own" mechatronic devices in the structure of the laboratory and the differentiation of access rights to them is one of the services of the graphical interface that is delivered only to users registered on the server.



kkirsanov@kkirsanov-ThinkPad-T420: ~
Файл Правка Вид Поиск Терминал Справка
2_problem_zoom2.png
Изображение Правка Вид Переход Справка
1997 × 466 пикселов 49,2 кБ 33%
3 / 7
robotree.py
1210
0
2140
Подключить

```
10100'] [129, 3, 116, 116]
speed =  234 ['10000001', '00000000', '01110101', '011
10101'] [129, 0, 117, 117]
speed =  235 ['10000001', '00000011', '01110101', '011
10101'] [129, 3, 117, 117]
speed =  236 ['10000001', '00000000', '01110110', '011
10110'] [129, 0, 118, 118]
speed =  237 ['10000001', '00000011', '01110110', '011
10110'] [129, 3, 118, 118]
speed =  238 ['10000001', '00000000', '01110111', '011
10111'] [129, 0, 119, 119]
speed =  239 ['10000001', '00000011', '01110111', '011
10111'] [129, 3, 119, 119]
speed =  240 ['10000001', '00000000', '01111000', '011
11000'] [129, 0, 120, 120]
speed =  241 ['10000001', '00000011', '01111000', '011
11000'] [129, 3, 120, 120]
speed =  242 ['10000001', '00000000', '01111001', '011
11001'] [129, 0, 121, 121]
speed =  243 ['10000001', '00000011', '01111001', '011
11001'] [129, 3, 121, 121]
speed =  244 ['10000001', '00000000', '01111010', '011
11010'] [129, 0, 122, 122]
speed =  245 ['10000001', '00000011', '01111010', '011
11010'] [129, 3, 122, 122]
speed =  246 ['10000001', '00000000', '01111011', '011
11011'] [129, 0, 123, 123]
speed =  247 ['10000001', '00000011', '01111011', '011
11011'] [129, 3, 123, 123]
speed =  248 ['10000001', '00000000', '01111100', '011
11100'] [129, 0, 124, 124]
speed =  249 ['10000001', '00000011', '01111100', '011
11100'] [129, 3, 124, 124]
speed =  250 ['10000001', '00000000', '01111101', '011
11101'] [129, 0, 125, 125]
speed =  251 ['10000001', '00000011', '01111101', '011
11101'] [129, 3, 125, 125]
speed =  252 ['10000001', '00000000', '01111110', '011
11110'] [129, 0, 126, 126]
speed =  253 ['10000001', '00000011', '01111110', '011
11110'] [129, 3, 126, 126]
speed =  254 ['10000001', '00000000', '01111111', '011
11111'] [129, 0, 127, 127]
kkirsanov@kkirsanov-ThinkPad-T420:~$
```

Имя	Адрес	Порт
AMUR	VPN	
lab1	192.168.5.*	
РГГУ	192.168.6.*	
Выносная камера	192.168.6.111	
AMUR-07	192.168.6.109	
Ультразвук	192.168.6.109	120123
двигатели	192.168.6.109	120124
Камера	192.168.6.110	8080
Аккумулятор	192.168.6.109	120125
Robotino	192.168.6.130	
Kuka	192.168.6.135	
	lcoal	

Fig. 3. View of one of the levels of the window-hierarchical graphical user interface of the network of Internet-laboratory.

5. Implementation of the «concept of drivers» on the example Festo Robotino robot

The “concept of drivers” was used to include the AMUR-10 scientific and educational robot (made by Sensorika Int. Lab.) into the structure of the laboratory. A driver was constructed that makes it possible to perform (apart from its direct functions) dynamic reprogramming of the onboard personal computer and a low-level management of the mobile robot components. This deep penetration into the mechatronic structure of the robot became possible due to the network approach to constructing its IMCS (see Section 3). The mechanism of the inclusion of third-party mechatronic devices into the laboratory network was tuned up with the help of the Robotino educational mobile robot (Festo Didactic company).

The computer onboard the Robotino robot works under the Ubuntu Jaunty operating system with a modified kernel. The onboard computer and mechatronic components of the robot interact through a set of system services (demons), which are managed through a special application program interface (API). There are two versions of the interface and the respective sets of demons. The first version depends on the set of libraries and has already been available in Robotino. For this version, a number of programs have been written by many authors (see the Robotino community portal [14]). In the second version, the developers reduced the number of dependencies to one but the backward compatibility (as stated by the developers) with programs that are based on the first version actually has not been confirmed. In view of this, the authors decided here to use the first version of the program interface.

Developers offer a set of APIs for use with C, C++, Java, .NET, MATLAB, Simulink, LabVIEW, and Microsoft Robotics Developer Studio [14, 15]. At present, the primary development language is Python. Since no such interface could be found for Python, the need for its development arises. This problem can be solved in two ways. The first is to replace the demons operating with the real-time kernel. The second is to translate into Python the classes and methods of the Robotino program interface, which was written in one of the above-mentioned languages. The second variant was chosen in view of its best adequacy to the concept proposed by the authors – to minimize the changes in program modules supplied with mechatronic device. The C++ interface was chosen as the basis.

There are two main options for combining the program code written C++ and Python: (a) running the Python interpreter from the C++ code and (b) writing an extension module in C++ for Python. Option (a) has the significant drawback of lacking dynamic reprogramming. The second option is devoid of this drawback, and the extension module designed to work with the C++ software interface can be used then for addressing other tasks (for example, SLAM). In addition, the threshold of entry to programming in Robotino decreases because the threshold for Python is lower than for C++ or other languages.

Among many existing methods for creating extension modules for Python, the following can be considered currently as dominant: (a) the direct use of Python API in C (Python C-API) and (b) the use of libraries Boost::Python, SWIG, and Cython, which are interfaces to Python C-API. The first method requires many details of type conversions from Python objects into C types, which highly complicates the problem. The second method is simpler. At the same time, the use of Boost::Python on the onboard computer of the robot proves to be difficult because of its large size. The SWIG and Cython libraries provide a similar methodology for writing Python interface modules on the basis of compiled C and C++ libraries; yet, Cython additionally provides an opportunity to speed up the execution of the Python code.

The computer onboard Robotino has already a Python interpreter version 2.6 installed. Since the library of the laboratory works with the PyZMQ, this library must also be installed on the onboard computer. That is enough for running the library of the laboratory; however, the activities with the Robotino API module require an additional Cython module with appropriate Python header files.

The Robotino driver was written in Python; however, because the performance of the Python interpreter is low, it is recommended to write the modules of management of the mechatronic device in C++ or C and use Python only for creating an interface to these modules and prototyping. The driver reports that Robotino is available in the laboratory network, connects to the Robotino API, and arranges the exchange of data (sensor data and control commands) between libraries.

Conclusions

Currently, the proposed structure of spatially-distributed scientific and educational Internet-laboratory with decentralized management has proved its efficiency and has been implemented for setting up of permanent VPN-tunnels for the interaction between robots of the type AMUR [16] and «Robotino», which operates between the MSTU "STANKIN" (Moscow) and the Center for Technological Support of Education (CTSE) of the

International Institute of New Educational Technologies (IINET) RSUH (Moscow). A similar communication exists between the Keldysh Institute of Applied Mathematics, Russian Academy of Sciences (Moscow), CTSE IINET RSUH, and the Far East Federal University (Vladivostok). Currently, work is underway to organize a communication between CTSE IINET RSUH and the University of Zadar (Croatia) as well as to test all units of the full-scale scientific and educational Internet-laboratory in the pilot exploration mode, including for related applications [17, 18]. To materialize the educational opportunities, methodological-training systems have been prepared for several universities.

References

[1] ROS. – URL: http://www.ros.org. Accessed: 22.09.2014.

[2] Microsoft Robotics Developer Studio. – URL: http://www.microsoft.com/en-us/download/details.aspx?id=29081. Accessed: 22.09.2014.

[3] Jordi Artigas, Ryu Jee-Hwan, C.Preusche and G.Hirzinger. Network representation and passivity of delayed teleoperation systems. Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on 25-30 Sept. 2011, pp.177-183.

[4] Günter Niemeyer and Jean-Jacques E. Slotine. Telemanipulation with time delays. Int. Journal of Robotics Research, 23(9):873-890, September 2004.

[5] Pryanichnikov V., Andreev V. The Application of Network Technologies to Constructing Group Controlled Systems with Machine Vision for Mobile Robots. Annals of DAAAM for 2012& Proceedings of the 23th international DAAAM Symposium "Intelligent Manufacturing & Automation" 24-27th October 2012 Zadar, Croatia, ISSN 2304-1382, 2012. – V.23, No.1. – P.1167 – 1174.

[6] Victor Andreev, Sergey Kuvshinov, Valentin Pryanichnikov, Yury Poduraev. Education on the basis of virtual learning robotics laboratory and group-controlled robots. 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013. Procedia Engineering, 2014. – V.69. – P.35 – 40.

[7] Andreev V., Pryanichnikov V., Prysev E. Multi-access control of distributed mobile robotic systems on the base of networking technologies. Annals of DAAAM for 2010&Proceedings of the 21th International DAAAM Symposium "Intelligent Manufacturing & Automation: Focus on Interdisciplinary Solutions" 20-23rd October 2010 Zadar, Croatia. ISSN 1726-9679. 2010. – P.15 – 16.

[8] Pryanichnikov V., Andreev V., Ivchenko V., Kuvshinov S., Prysev E. Adaptive environment for developing and programming of mobile robots. Annals of DAAAM for 2011&Proceedings of the 22th International DAAAM Symposium "Intelligent Manufacturing & Automation: Power of Knowledge and Creativity", 23–26 November 2011, ISSN 1726-9679, Vienna, Austria, 2011, vol. 22, No.1, pp. 609 – 610.

[9] Kirsanov K. Software architecture of control system for heterogeneous group of mobile robots. 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014. Procedia Engineering ISSN 1877-7058. IN PRESS.

[10] GOST R ISO/IEC 7498-1-99, Information technology. Open systems interconnection. Basic reference model. Part 1. The basic model. – URL: http://protect.gost.ru/v.aspx?control=7&id=132355. Accessed: 27.05.2014.

[11] Hintjens Pieter, ZeroMQ: Messaging for Many Applications, O'Reilly Media, Inc., 2013.

[12] ØMQ Licensing. – URL: http://zeromq.org/area:licensing. Accessed: 27.05.2014.

[13] GNU lesser general public license. Version 3, 29 June 2007. -http://www.gnu.org/licenses/lgpl.html. Accessed: 27.05.2014.

[14] Robotino Wiki. [2014—2014]. URL: http://wiki.openrobotino.org. Accessed: 28.06.2014.

[15] Robotino // Wikipedia. [2014—2014]. URL: http://ru.wikipedia.org/?oldid=63887402. Accessed: 28.06.2014.

[16] Andreev V., Pryanichnikov V. Operation environment of mobile robots with supervision control, Annals of DAAAM for 2011&Proceedings of the 22th International DAAAM Symposium "Intelligent Manufacturing & Automation: Power of Knowledge and Creativity", 23–26 November 2011, ISSN 1726-9679, Vienna, Austria, 2011, vol. 22, No.1, pp. 21 – 22.

[17] Katalinic B., Pryanichnikov V., Ueda K., Cesarec P., Kettler R., et al., Bionic Assembly System: hybrid control structure, working scenario and scheduling. Proceedings of 9th National Congress on Theoretical and Applied Mechanics, Brussels, 9–11 May 2012, pp.101 – 108.

[18] Frolov A.A., Biryukova E.V., Bobrov P.D., Mokienko O.A., Platonov A.K., Pryanichnikov V.E., and Chernikova L.A., Principles of Neurorehabilitation Based on the Brain–Computer Interface and Biologically Adequate Control of the Exoskeleton, Human Physiology, 2013, Vol.39, No.2, pp. 196 – 208.