Data Exchange Platform Dedicated to Distributed Control Systems

Przemysław Strzelczyk, Krzysztof Tomczewski
Opole University of Technology, Department of Electrical Engineering, Automatic Control and Informatics, Prószkowska 76, 45-758 Opole, Poland

Abstract: The paper presents the concept of generic exchange data platform dedicated to distributed control systems. The platform can be used to control industrial facilities, to synchronize the work of industrial and mobile robots. The solution can also be applied to active prostheses. The article presents an example of information exchange subsystem structure as well as data flow between platform applications.

Keywords: distributed control systems, communication platform, robotic middleware, exoskeleton

1. Introduction

The ability of control through providing proper methods of communication between the distributed automation system nodes gives a big opportunity to extend and to change robotic system configuration and also opens new opportunities up allowing expanding or limiting the functionality of the system. The basis of failure-free control distributed system is fast and reliable information exchange between the objects. The more complex system is, the greater amount of data is transferred. This is a big problem in complex automation systems. The protocol and the data transmission algorithm have a big impact on the data exchange efficiency.

Master-slave architecture is the most commonly used in automation systems. In such systems, the master plays a key role setting information exchange scenario with slave units. This architecture most often enforces to define system construction structure in advance or to use static predefined network addresses. An additional problem is the lack of direct communication possibility between slave units. To enable communication between devices, one of the available communication protocols can be used, e.g. Modbus. Implementation of any protocol imposes the creation of application dependent on the operating system and hardware platform. In addition, data exchange rules implementation in control and monitor applications is a big problem. It makes core implementation of control algorithm complicated.

This paper presents a communication platform proposal implementing selected communication mechanism and providing data exchange services between applications run in each nodes of distributed automation control system and separating control applications from the operating system and hardware platforms. Described communication platform allows also defining any architectural: master-slave, multi-slave/master with the possibility of direct communication between all devices.

Many companies and scientific centers around the world have been working on similar data exchange and modularity solutions for robotic systems. For example, NASA has been working on CLARAty platform, which has been designed for the development and maturation of various research technologies. CLARAty platform is currently used in the Mars Exploration Program and Rover Mission [9, 10]. The most popular open solutions are: Player, MIRO, ROS, OROCOS and RT-Middleware [9]. The main purpose of the above systems is to provide software genericness of higher-level applications and to provide communication primitives [12]. The architecture of currently available solutions requires major changes to existing software during the adaptation process.

The proposed solution allows using the functionality at every stage of software evolution with relatively small effort compared to existing big systems. Communication platform carries such a task out should be characterized by generic construction [12]. The delimitation should be so effective that the possible change of the operating system does not have a direct impact on operation of application based on platform assumptions [11]. Further sections of the article contain information about platform genericness and current information exchange and synchronization mechanisms.

2. Platform Genericness

Platform genericness is understood in such a way that the platform should differentiate the relationship between the operating system controlling the device and the application uses the API (*Application programming interface*) from the library provided by the platform [8]. The removal of relationships with operating system functions and specified operating system behaviors allows to keep a platform genericness. It means the possibility of executing application on different hardware platforms with different operating systems without applying changes [7]. The platform genericness was developed thanks to generic programming paradigm, which focuses on abstracting types to a narrow collection

Autor korespondujący:
Przemysław Strzelczyk, przemyslawstrzelczyk@gmail.com

Artykuł recenzowany
nadesłany 12.07.2017 r., przyjęty do druku 30.08.2017 r.

of functional requirements and on implementing algorithms in terms of these requirements. Most interfaces defined by the algorithms are strict and narrow to the types they operate on, the same algorithm can be used against a wide collection of types [8].

The platform is based on intermediary library and layers preparing the environment and executing queries coming from the library calls. However, the platform software responsible for this separation has to in some way be associated with selected operating system. The relationship level is dependent on communication mechanisms built into the operating system itself. The platform software will include eventually implementation of dependent snippets for the most popular operating systems. This will allow to build a software package based on properly designed shell script program automating compilation process (make file) on most commonly used operating systems without the need of additional platform functionality modification (cross compiling) [1]. A large relationship factor will be determined by the need of build mentioned mechanisms and direct links with the hardware. However, system selection without these elements would be inefficient in terms of time spent on platform adjustment as well as on complementary elements development and also in terms of performance.

A potential area of use of such a solution may be mobile robots. The platform can provide the communication between robots, between the robot and the operator console, but also between the modules within a single robot. Exemplary application of data exchange platform in multi-robot system is shown in Fig. 2. Planned application of communication platform is data

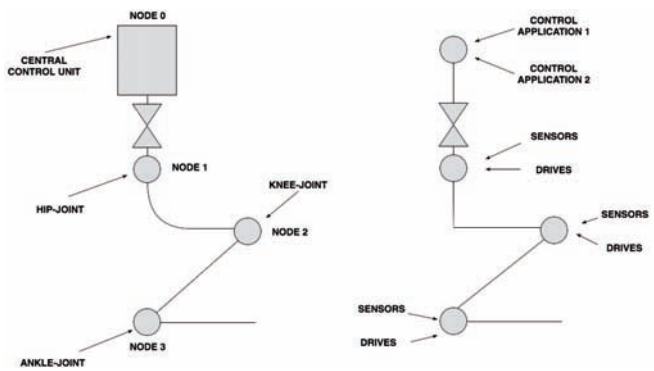

Fig. 1. Sample structure of active prostheses
Rys. 1. Przykładowa struktura aktywnej protezy

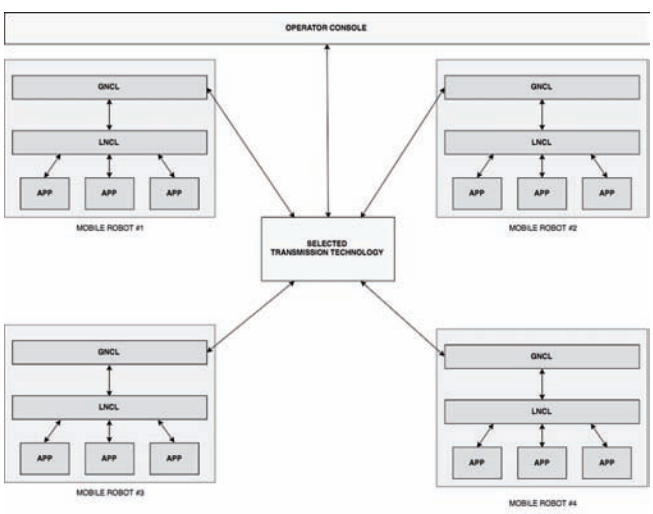

Fig. 2. Application of the platform in the mobile robot system
Rys. 2. Zastosowanie platformy w systemie robotów mobilnych

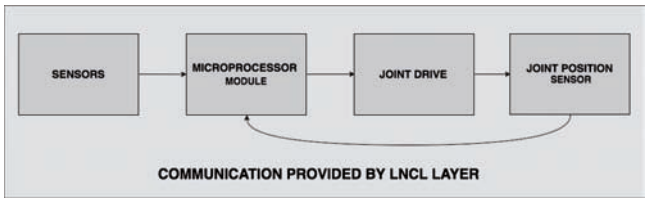

Fig. 3. An example of the structure of the control system of the kinematic coupling
Rys. 3. Przykład sprzężenia w układzie sterownia kinematycznego

exchange in scope of complex rehabilitation equipment such as: active artificial limbs, elements supporting the movement or exoskeletons for people partially paralyzed. In such complex systems, it is advisable to use distributed control systems, with nodes placed at critical points, such as hip, knee, ankle etc. The platform allows to drives control, sensors reading etc. These devices are complex robotic elements. There are close links between variables articulated values and behavior of the device in the global coordinate system. It is necessary to exchange information between the individual modules and the central control unit, e.g. in order to maintain equilibrium, a move and so on. An example structure of such a system is shown in Fig. 1. An exemplary structure of module located in the node is shown in Fig. 3.

Platform software adjusts all the functionalities to the new equipment. If one of communication platform assumption concerning the availability of broadcast or multicast data transferring to application is present, the software platform will provide the functionality regardless of hardware or operating system changes through all further revisions.

3. Structure of the Distributed Control System

The communication platform allowing to exchange of information is not necessarily limited to one device. The main task of the presented concept is the ability to work in distributed systems. In addition to providing genericness, platform provides the ability to exchange data between distributed systems components. The data exchange in such systems should be done in a fast and reliable way, but with a high security level. Exchange of information in distributed systems is an issue that could cause a lot of problems [5, 6].

The need of transfer confirmation is the basic problem. In general, the distributed control system may have dynamically changing number of objects, so it is not possible to adopt a fixed number of nodes that exist in the system. Node of such a system is a module on which communication platform was launched along with control applications that use its functionality. An example would be a system composed of several mobile robots where some of them have lost communication with each other. The number of nodes in the system decreases when layer responsible for the information management concerning the platform points, notify the lack of response to previously recorded unit. If new node is detected by the platform, it will be immediately included in the system and allow applications installed on other network nodes communicate with the newly discovered object. The algorithm implemented in the control application will condition kind of interaction between these objects. Exemplary of the distributed control system structure is shown in Fig. 4. The example presented in Fig 4. assumes that Wi-Fi/Ethernet standard has been selected as transmission technology. The topic of this paper was not the creation of transmission technology but a platform allowing communication of higher-level applications. The transmission technology can be freely selected as long

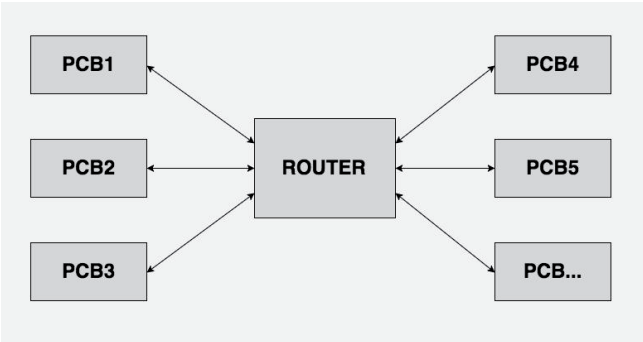

Fig. 4. An example of the distributed control system structure
Rys. 4. Przykładowa struktura rozproszonego systemu sterowania

as the conditions that allow for the correct implementation of this platform are met (TCP/UDP support). The Wi-Fi/Ethernet router solution was chosen due to easy access to devices that support this technology and low configuration complexity. The Wi-Fi/Ethernet technology was chosen due to Linux built-in support. The author recommends using a technology, which allows avoiding the centralized distribution of information (ad-hoc solutions).

4. Data Exchange in the Single Node of the Platform

Information exchange between applications within a single node is performed via an intermediate layer of the platform LNCL (*Local Node Communication Layer*). This layer is responsible for packet transport between applications within a single object. In the case of mobile robots, this kind of communication is responsible for data exchange between applications running within single robots. Multiprocessing and multithreading support are not common solutions in control systems mainly due to reliability requirements. This library provides the ability to use information exchange mechanisms also in multi-threaded and multi-processed control systems. Proper use of these mechanisms can lead to significant software performance improvements especially if hardware platform provides multi-core solutions.

Using the supplied API library NodeAPI (*RegisterTransportAddress* function) each application can register itself in the system, and then obtain the unique identifier to allow the functioning in the entire system. If the operating system allows the use of multithreading, each thread running in the process will also have the opportunity to register at least one transport address. This allows the modular software design and the appropriate segregation of responsibilities. Providing dynamic registration opportunity to LNCL layer significantly extends the capabilities of the software running on the platform. Applications can send data packets to each other knowing only their TID (*Transport ID*). Layers such as LNCL have static addresses. Using of fixed addressing allows for communication within a single instance of the platform without the use of additional mechanisms to detect dynamic address of the individual layers. It means that, each application run has knowledge about address using to communicate in advance. Adopted data transfer mechanism allows to send a package without forward imposing its size. It was accomplished using a specially prepared header preceding each message. If there would be a need to send a data structure, transport system has to prepared to handle a message with defined size properly by calling *CreateMessage* function with all relevant information such as the size of data transferred, the recipient identifier and a pointer to the data. In case of success function returns a pointer to the memory, indicating the place where specially prepared message supplemented with all necessarily data is located to properly forward by the platform. The returned pointer will be passed to *SendMessage* function responsible for sending pre-prepared package.

Communication within a single instance of the platform has been implemented using shared memory. (Linux domain sockets) Shared memory was selected because of the transfer performance during exchanging information between processes [2].

5. Data Exchange Between Distributed Objects

According to the assumptions the platform has also functionality for data exchange between distributed nodes. Communication with other objects in each instance is implemented by means of GNCL layer (*Global Node Communication Layer*). GNCL has static address registered in LNCL. Its task is to transfer information using TCP/IP protocol to other distributed system modules. The choice of TCP/IP has been dictated by the need of packet delivery confirmation. Another criterion of selection was the order of packet delivery. TCP/IP protocol ensures the order of transmitted packets and their potential retransmission in case of error. The protocol has also been chosen because the complexity of the platform-based system can change dynamically in the runtime. The architecture should allow the creation of simple network as well as a very complex one. In order to send message, it is necessary to determine the recipient. Address of the addressee within the software platform is defined using the 64-bit TID. More significant 32 bits contain the address of the node to which information is addressed. The remaining 32 bits specify the address of an internal application to which the message is sent. After sending, the message goes to LNCL, where the transport address analysis is determined whether this information is local or external. In case of external message, it will be forwarded directly to GNCL layer. Next, local GNCL layer forwards the packet to recipient GNCL, which in turn passes it to the local LNCL then it further forwards the packet to target application.

6. Synchronization of Control System Operation

One of the main problems during development of distributed systems is the synchronization of processes in local as well as global scope. Each of nodes has a number of layers required for correct requests handling connected with offered functionality. During platform software startup boot order of its individual components is extremely important because every booting layer is dependent on the functionality offered by other layers [3]. Wrong order or synchronization lack could lead to startup phase failure or race condition. The first launched and the most important layer in a scope of single node is LNCL.

Other layers use possibility of data exchange between registered recipients, so queuing within a single node platform is extremely important. An example showing such an event in the global scope can be three platforms nodes layout. The first acts as a central control unit. The other two are responsible for determining the position and drive control in two kinematic connectors of active prostheses. Starting calculations at the positioning unit in a global scope before properly started unit responsible for measurement could lead to the designation of an erroneous initial position. Similarly, transferring information

about calculated values of settings to control applications operating in kinematic joints before their starting would not lead to a move, which may also affect the further operation. It is therefore important to maintain sufficient start-up procedure.

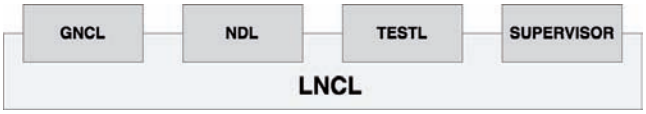

Fig. 5. Dependencies between communication platform layers
Rys. 5. Zależności między warstwami komunikacyjnymi platformy

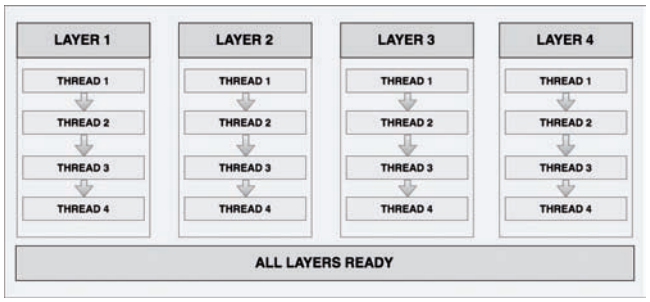

Fig. 6. Layer startup structure
Rys. 6. Struktura uruchomieniowa warstwy

Communication between all nodes and their applications has to be activated as the first. The next step is to run the measurement and control applications. During startup, it is necessary to establish starting work conditions of each application. The implementation of local and global synchronization manager provided by the communication platform solves the problem. Startup synchronization is not limited to the processes only. Described communication platform has the ability to synchronize threads within a process. Thread synchronization plays a significant role during startup of individual layers e.g. LNCL. Each of layers has a number of subprograms supporting functionality offered by the layer. LNCL layer owns implemented server supporting any message transmission attempt in local scope. Creating a new thread starts the server. Layer cannot mark itself as started properly if all of its threads will not achieve the intended point of execution (in server case, the point is reached when it is ready to handle incoming connections). An exemplary layer startup flow is shown in Fig. 5.

7. Automatic Node Detection Mechanism

Depending of the purpose, the structure of a distributed control system may vary. Sometimes there is a need to change the structure in already running system. This is important especially in systems supervising the mobile robots. In many cases, the structure of the platform may change dynamically.

Thanks to a NDL (*Node Discover Layer*) it is possible to detect new appearing objects in the system and monitor the existence of previously recorded. Standard time in which layer searches the resources to find new nodes and confirm the existence of previously recorded ones is 500 ms. The NDL mechanism is based on UDP protocol and cyclical sending queries and matching their answers. Information about system structure change can be very important, and delivered in a short period of time allows for the correct response of the entire system.

8. Results of Tests

Testing of described solution is complicated, mainly because of the distributed structure and the possibility to dynamically change of system structure. Dedicated environment consisting of several nodes equipped with communication platform software has been developed to test the platform assumptions.

The test stand shown in Fig. 7 has been prepared specifically for the Raspberry Pi hardware modules because thanks to its low price and wide capabilities it has been chosen as the main hardware used during platform testing. The test stand consists of three Raspberry Pi devices connected via MikroTik Could Router Switch CRS125-24G-1S-2HnD-IN using Ethernet. The modules are equipped with the Raspbian operating system, installed and configured data exchange platform software.

Two test scenarios were performed. The first scenario consisted of data exchange within local scope (using LNCL layer). Two applications dedicated for testing purposes had only been developed (using C++ language) and each of them was properly registered in the platform system. The first one performed as a sender and the second one as a receiver. The single package sent from application A to application B had a size of 1000 bytes of random data. The test has been carried out 1024 times (Table 1).

Table 1. The results of local exchange data test
Tabela 1. Wyniki testu lokalnej wymiany danych

Packet No.	Transmission time [ms]	Packet No.	Transmission time [ms]	Packet No.	Transmission time [ms]
0	0.06992	8	0.021792	16	0.000427
1	0.043983	9	0.021404	17	0.008648
2	0.023618	10	0.021388	18	0.023158
3	0.022417	11	0.021331	19	0.021553
4	0.02229	12	0.021425	20	0.021509
5	0.021878	13	0.021412	21	0.021364
6	0.021489	14	0.059337	22	0.021401
7	0.023894	15	0.057931	23	0.021306

The obtained data from first test were statistically analyzed with the following results: $x_{min} = 0.00042$ ms, $x_{max} = 0.00042$ ms, $\overline{x}_1 = 0.04621$ ms, $\sigma_1 = 0.07446$.

The second scenario consisted of data exchange using "everyone to everyone" principle. The test program that sends a 1000 byte packet to the indicated platform nodes and waits for acknowledge from the receiver side was run on each of the Raspberry Pi modules. The time measurement started when the data set was sent and finished when the acknowledgment was received. The acknowledgment had a form of a packet with the same content as the one sent. The confirmation from the recipient was necessary because the platform instances work on non time-synchronized operating systems. After the measurements are completed, the result is averaged (Table 2).

Table 2. The results of global exchange data test
Tabela 2. Wyniki testu globalnej wymiany danych

Packet No	Transmission time [ms]	Packet No.	Packet No.	Packet No.	Transmission time [ms]
0	0.5123	8	0.9277	16	0.7821
1	0.4453	9	0.4552	17	0.5242
2	0.7883	10	0.5611	18	0.3451
3	0.4763	11	0.6132	19	0.7481
4	0.4112	12	0.4252	20	0.8512
5	0.6211	13	0.5331	21	0.4625
6	0.8135	14	0.5932	22	0.5162
7	0.7422	15	0.5793	23	0.6821

The obtained data from second test were statistically analyzed with the following results: $x_{min} = 0.2344$ ms, $x_{max} = 0.9277$ ms, $\overline{x}_2 = 0.5526$ ms, $\sigma_2 = 0.1574$.

The results shown in Table 1. and Table 2. include delays caused by the operating system scheduler. The next communication platform releases the priority of the process will be changed. This procedure will reduce the probability of expropriation.

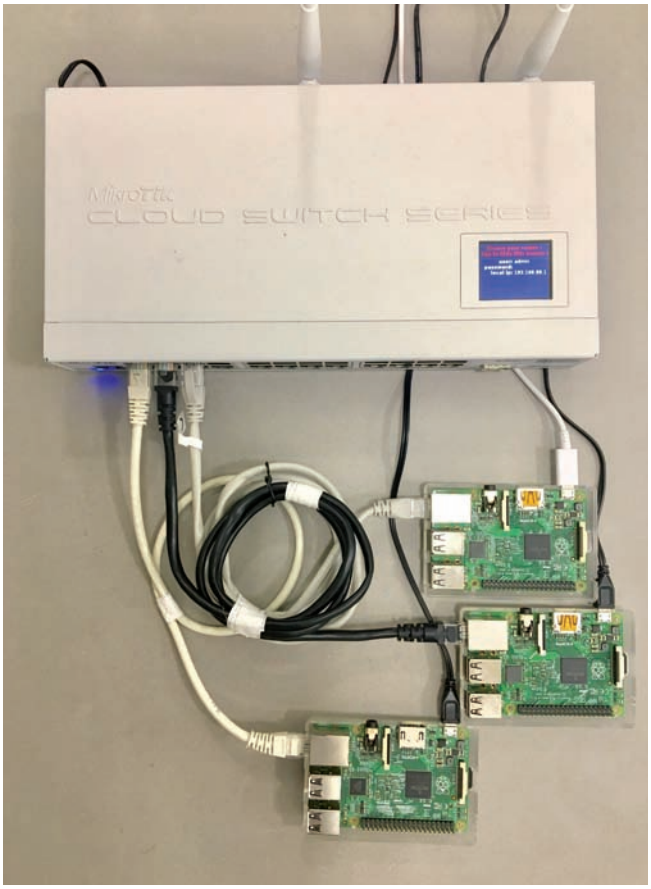

Fig. 7. Test stand
Rys. 7. Stanowisko testowe

9. Summary

The structure of a platform providing data exchange options in distributed systems generates huge opportunities for expansion of control systems. It removes system constraints arising from the location of its elements. The proposed platform can be successfully used in the automation and robotics while performing the tasks associated with the need to collect and exchange information in remote points. Thanks to its genericness, the platform is largely universal, and applications based on it are fully cross-platform programs.

References

1. Wojtczyk M., Knoll A., *A Cross Platform Development Workflow for C/C++ Applications*, The third International Conference on Software Engineering Advances (ICSEA-2008), 2008, DOI: 10.1109/ICSEA.2008.41.
2. Shapley Gray J., *Interprocess Communications in Linux*, ISBN: 0-13-046042-7, Prentice Hall Professional 2003.
3. Itami Y., Ishigooka T., Yokoyama T., *A Distributed Computing Environment for Embedded Control Systems with Time-Triggered and Event-Triggered Processing*, 14th IEEE International Conference on Embedded and Real--Time Computing Systems and Applications, 2008, DOI: 10.1109/RTCSA.2008.38.
4. Wittenmark B., Nilsson J., Torngren M., *Timing problems in real-time control systems*, [in:] Proceedings of American Control Conference 1995, DOI: 10.1109/ACC.1995.531240.
5. Noriaki A., Takashi S., Kosei K., Tetsuo K, Woo-Keun Y., *RT-Middleware: Distrbiuted Component Middleware for RT* (*Robot Technology*), IEEE/RSJ International Conference on Intelligent Robots and Systems 2005, DOI: 10.1109/IROS.2005.1545521.
6. Volpe R., Nesnas I., Estlin T., Mutz D., Petras R., Das H., *The CLARAty architecture for robotic autonomy*, Proceedings of IEEE Aerospace Conf., Montana, March 2001, DOI: 10.1109/AERO.2001.931701.
7. Ceriani S., Migliavacca M., *Middleware in robotics*, Internal Report For „Advanced Methods of Information Technology for Authonomous Robotics", Politecnico di Milano.
8. Alexandrescu A., *Modern C++ Design: Generic Programming and Design Patterns Applied*, ISBN: 978-0201704310, Addison-Wesley 2011.
9. Elkady A., Sobh T., *Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography*, Hindawi Publishing Corporation, "Journal of Robotics", Vol. 2012, DOI: 10.1155/2012/959013.
10. Nesnas I.A.D, Wright A., Bajracharya M., Simmons R., Estlin T., *CLARAty and Challanges od Developing Interoperable Robotic Software*, NASA Ames Research Center, Moffet Field, Sunnyvale, CA 95134 March 2003.
11. Schlegel Ch., Steck A., Brugali D., Knoll A., *Design Abstraction and Processes in Robotics: From Code-Driven to Model-Driven Engineering*, Technische Universtat Munchen, Munchen Germany.
12. Ch. Schlegel, *Communication Patterns as Key Towards Component-Based Robotics*, Software Engineering for Experimental Robotics, Springer Tracts in Advanced Robotics, DOI: 10.5772/5759.

Platforma wymiany danych dedykowana dla rozproszonych systemów sterowania

Streszczenie: Artykuł przedstawia koncepcje generycznej platformy wymiany danych dedykowanej dla rozproszonych systemów sterowania. Platforma może zostać wykorzystana do sterowania obiektami przemysłowymi a także do synchronizacji pracy przemysłowych oraz mobilnych robotów. Rozwiązanie może być również zastosowane w aktywnych protezach. Artykuł prezentuje strukturę podsystemu wymiany danych oraz opisuje przepływ informacji pomiędzy aplikacjami bazującymi na oprogramowaniu platformy.

Słowa kluczowe: rozproszone systemy sterowania, platforma komunikacyjna, oprogramowanie pośredniczące, egzoszkielet

Przemysław Strzelczyk, MSc Eng.
przemyslawstrzelczyk@gmail.com

PhD candidate at Opole University of Technology, Faculty of Electrical Engineering, Automatics and Computer Science field of study – Automatics and Robotics.

Krzysztof Tomczewski, PhD DSc
k.tomczewski@po.opole.pl

Opole University of Technology, Faculty of Electrical Engineering, Automatics and Computer Science Institute of Power Systems and Robotics, Head of the Department of Electrical Drives, Diagnostics and Industrial Electronics.