Abstract

The paper proposes an approach to intermodular communication method’s specification for heterogeneous modular mobile robot based on principles of ROS (Robotic Operation System). One of key features of this specification is accounting for structure of modular robot’s control system built on full functionality principle. The full functionality of a mechatronic device is the ability to perform its goal function using only its own facilities for executing instructions from an external control system. Consequently, every module is a fully-functional electronic or mechatronic device with its own control system, including all required components: sensors, computing modules and drives. As modules’ computing devices are usually microcontrollers or other embedded computers which are very restricted in computing power, using ROS in every module is complicated and is not efficient. Proposed specification use ZeroMQ library combined with UDP protocol, which made it possible to implement specification for embedded devices. Rationale behind the choice of type and topology of communication networks is presented. Requirements for APIs (application programming interfaces) of modules of different types are formulated.

The results of an experimental study of the developed specification of the method of intermodular information interaction on the operating model of a heterogeneous modular mobile robot are presented.

Keywords: modular robot; heterogeneous robot; control system; local area network; network protocol

1. Introduction

This paper is the part of the research work on creation of a heterogeneous modular mobile robot (MR) for a wide range of purposes. The heterogeneous modular MR is a system of individual functional devices that, when assembled in one of a finite number of configurations, constitute a synergetic system – a mobile robot with executive, tactical and strategic control levels. Each individual device (a module) in such a robot is a mechatronic or an electronic device. Such individual devices should have universal mechanical, electrical and programming interfaces. This ensures the ability of the whole structure to be reconfigurable on site of works directly and on any time of operation. Even better interchangeability and flexibility of modules can be achieved if they are designed to carry out one specific function and to carry it out well i.e. to be fully functional. The full functionality of a mechatronic device is the ability to perform its goal function using only
its own facilities for executing instructions from an external control system [1]. The full functionality of robot modules is achieved by designing each module as a full-featured electronic or mechatronic device with its own the information-measuring and control system (IMCS) containing all necessary components: sensors, computing devices and actuators. But here lies a contradiction. For example, the transport module carries out robot motion function. If such module will blindly make its way towards destination without considering environment’s obstacles, it will eventually bump in some obstacle which was not expected by an external control system. But this does not mean that such a module must be equipped with its own distance sensors. To avoid collision a modular MR should be equipped with one or more distance sensors modules. Sensors modules’ and transport modules’ control systems are separated, therefore a necessity of synchronization between them becomes apparent. Also further increase in number of functional modules leads to organization of the general distributed control system which unites all control systems of each individual module. In such distributed control system every module should be able to exchange any information (including control commands) with any other module directly. Thus the whole system needs a specific communication method. Such method exists already and it is open-sourced project called Robot Operating System. It is considered state-of-the-art for mobile robotic systems at the moment. But also at the very same moment it is unusable in purposes of modular robotics as it cannot be used on low-power embedded computers directly.

The purpose of the study is to develop a specification of the intermodular communication method, built on principles of ROS (Robot Operating System), but able to run on embedded computers. As a basis, such specification should use the full functionality principle.

In order to achieve this goal it was necessary to complete the following tasks:
1. Select the type and topology of the communication network.
2. Develop the method of the information interaction of modules.
3. Define the hardware and software requirements for the modules.
4. Develop modules’ systems of commands.
5. Develop new network protocols and modify well-known network protocols to implement the inter-module communication application programming interfaces (API).

2. State-of-the-art modular mobile robotics

Let's consider existing modular robots from the point of view of their intermodular communication system correspondence to the principle of full functionality and distributed control system. In 1987, Japanese specialists introduced the CEBOT modular robot [2]. This is the first of the known mobile robots, which is heterogeneous and fully modular, implying the unity of interfaces. Modular robots are divided into homogeneous and heterogeneous.

Homogeneous robots consist of many identical modules. Each module contains all the components necessary for it to function – motors, propellers, batteries, control systems, etc. In fact, a homogeneous modular robot is a superposition of a set of individual identical robots. An example of such robots is the modular robot M-TRAN [3]. Despite the simplicity of the movements carried out by one module, many modules, built into a certain structure, can perform complex coordinated movements. M-TRAN, as well as other homogeneous robots, for example, ATRON [4] and TRANSMOTE [5], have a common feature – all of them, even having united in the general structure, do not form a common full-fledged distributed control system. Each of the modules is independent, but all of the modules are controlled by one module-supervisor via execution level commands. The difficulty for a supervisor to control such modules directly is worth noting as it is necessary to develop additional software that translates the supervisor commands into a sequence of motion commands for individual modules, and, in general, this software runs on an external computer, rather than being embedded in the modules themselves. Also it’s worth noting, that modules of all presented homogeneous robots are not full-featured. Modules of one type should be communicating with each other to carry out their primary function to move the robot.

Heterogeneous robots consist of a finite set of modules of different functionality that are connected to each other through unified mechanical and electrical interfaces. Unlike homogeneous, the modules of heterogeneous robots, in general, cannot function normally without each other.

SMART [6] and Thor [7] are heterogeneous robots. Each of these robots is a set of different modules and submodules (separate mechanical components of modules). The Thor robot is the direct successor of the homogeneous ODIN modular robot [8], including intermodular communication – both robots use RS 485 bus. The SMART robot is slightly different – one of its modules, connected via a Bluetooth wireless channel to the supervisor, becomes the master on the CAN bus through which it manages other modules, relaying commands received from the supervisor and the data from the other modules back to the supervisor via Bluetooth. In other words, all modules are driven by one module-supervisor on executive level only. Like homogeneous robots, they do not implement a full-fledged distributed control system, which makes them use a central high-power computer. This also leads to a lack of full functionality of modules of such modular MRs.

The most advanced in the field of intermodular interaction are the previously mentioned CEBOT and a modular robot for space research [9]. For them it is stated that all modules communicate with every other module – the modules have additional horizontal connections, which, for example, allowed the automatic configuration in the CEBOT modules.

For each of the listed robots, developers have created special software that allows building the CS intended only for this design, which does not allow the use of third-party modules in the design. This, in turn, makes it very difficult to
implement such a modular robot property as operational reconfiguration, which is especially important in cases of maximum uncertainty of the conditions of forthcoming works, i.e. in case of extreme situations, accidents, disasters and active counteractions, when it is not possible to determine in advance the composition and configuration of the necessary robotics [10]. Therefore, the task of creating a full-fledged distributed control system for modular robots that is capable of dynamically changing its structure and incorporating third-party modules into the robot design is topical.

Development of general APIs for full functional modules will enable realization of operational reconfiguration property of modular mobile robot.

This property can used successfully to develop robots for other planets’ and Moon exploration. A review [11] of robotics needed to create a lunar base provides a large selection of robots for a variety of purposes: from robot bulldozers to robot builders and robot repairers. Such a wide range of robots is economically unprofitable to bring to the moon. It is sufficient to create a set of full functional modules and assemble needed robots on the Moon.

The intermodular communication method is being developed on top of functional-modular principle of modular mobile robot construction [12]:

1. Each module is a functionally completed electronic or mechatronic device with its own information measuring and control system (IMCS) the principle of distributed control.
2. IMCS of a robot is a local area network with modules as nodes.
3. Incorporation of any mechatronic device in the network structure of a robot is done using special software – drivers.

Functional completeness is a property of every module to be able to execute on its own the task that is given to it by a module supervisor (some general control system) [13]. Then a distributed control system is a system with mostly independent control processes, distributed between modules’ computers, interacting with each other via intermodular communication method.

3. Communication networks: choosing type and topology

Modern modular robots use different serial busses, both homogeneous and heterogeneous, to achieve better modules’ miniaturization and minimization of the number of mechanical interfaces’ contacts:

- CAN – robots M-TRAN and SMART;
- IrDA (in fact RS-232) – robot ATRAN;
- RS-485 – robots Thor and Odin;
- TRANSMOTE robot uses ZigBee interface – a serial bus in a wider meaning.

![Diagram of communication networks](image)

Fig. 1. “Bus” (a) and “Star” (b) topology for Ethernet

From the review above it is clear that communication in known modular robots is most often based on a topology of the "bus" type with local branches (Fig. 1a), rarely – on a topology of a "star" (Fig. 1b). Topology "bus" most fully reflects the structure of the modular robot - each module has several "neighbors" through which it interacts with the rest of the system. The disadvantage of such buses is their low throughput. But in the above constructions, the bandwidth is sufficient, since such robots have, in fact, two ways to function: (1) execute a set of simple rules that the authors placed in the microcontroller ROM in each of the modules, and (2) execute the commands from an external computer, supplied
through a module to the bus of which (and, hence, to the common bus of the all connected modules) this computer is connected. This means that the role of the intermodular interaction system is small and, in fact, reduces only to the most rapid and efficient transmission of executive or tactical level commands and the readings of the sensors. Then the information flow in the process of intermodular interaction is small, since each module have either none or just a few of sensors and, in fact, it is necessary to transmit only executive level commands.

Many companies are now using the Ethernet standard. Its expansion for real-time operation EtherCAT has become the new standard among industrial field busses due to the versatility of the Ethernet standard [14]. A comparison of the various existing networks is given in Table 1.

<table>
<thead>
<tr>
<th>Features</th>
<th>RS-485</th>
<th>CAN</th>
<th>Ethernet</th>
<th>EtherCAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real time</td>
<td>Hard</td>
<td>Hard</td>
<td>Soft</td>
<td>Hard</td>
</tr>
<tr>
<td>Main topologies</td>
<td>Bus</td>
<td>Bus</td>
<td>Point-to-pont, various(2)</td>
<td>Point-to-pont, bus, various (3)</td>
</tr>
<tr>
<td>Top bandwidth (on certain connection length)</td>
<td>10 Mbit/s at 10 m</td>
<td>1 Mbit/s at 40 m</td>
<td>100 Mbit/s, 1 Gbit/s at 100 m</td>
<td>100 Mbit/s at 100 m</td>
</tr>
<tr>
<td>Collisions handling</td>
<td>Not defined</td>
<td>CR(4)</td>
<td>CD(4)</td>
<td>Master-Slave(5)</td>
</tr>
<tr>
<td>Service data length, bit</td>
<td>80(6)</td>
<td>44 or 64(7)</td>
<td>464 or 336(8)</td>
<td>160</td>
</tr>
<tr>
<td>Max data length, bit</td>
<td>2088(9)</td>
<td>64</td>
<td>11536 or 11664(8)</td>
<td>11888</td>
</tr>
<tr>
<td>Prevalence of equipment</td>
<td>High</td>
<td>Average</td>
<td>Very high</td>
<td>Low</td>
</tr>
<tr>
<td>Equipment cost</td>
<td>From low to average</td>
<td>From average to high</td>
<td>From low to average</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 1. Different networks’ properties

Notes for table 1: (1) – implementation dependent, standard does not defines this variable; (2) – topology can be both bus and star, depending on devices used; (3) – can have any topology, but most devices support bus topology; (4) - CAN is a synchronous bus with Collision Resolving (CR), (not Collision Detect (CD) as in Ethernet), it transmits messages deterministically; (5) – use a special frame format made by master device, which are read from and written to by slave devices, this ensures synchronicity of bus control at all times; (6) – for MODBUS implementation; (7) – regular and extended frame format; (8) –TCP and UDP protocols respectively; (9) –MODBUS implementation strictly regulates length of a message.

The principle of functional completeness of the modules eliminates the need to send executive level commands – only upper (tactical and strategic) level commands are transmitted. In this case, the “softer” requirements to the delivery time of messages and the use of interfaces with "soft real time" are acceptable. In our case, we are going to use in the robot’s sensor system a large number of different sensors and a multi camera system of technical vision. Therefore, the intermodule communication channel must be broadband. According to the data in table 1, Ethernet is the most high-speed interface, and the wide prevalence of the corresponding equipment ensures its low cost. In the case of EtherCAT, all intermodule communication must occur via the “master” of the bus, which reduces the efficiency of direct inter-module communication. Although Ethernet cannot be considered as a hard real-time protocol, but, as noted above, there is no need.

Ethernet is not a bus by definition – it uses “one to one” connections. One can make it as one by installing switches between nodes enabling “one to many” connections. This design enables both bus and star topologies.

Fig. 1a and Fig. 1b show the topologies "bus" and "star" superimposed on the topology of a modular robot. The choice of topology must be made on the basis of the requirements for the desired frequency of messaging, requirements for assembly and economic indicators, as well as taking into account the limitations of the network functioning of a particular topology. A comparison of these topologies is presented in table 2.

<table>
<thead>
<tr>
<th>Features</th>
<th>Bus</th>
<th>Star</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralization</td>
<td>Decentralized</td>
<td>Centralized</td>
</tr>
<tr>
<td>Switch count</td>
<td>$n$ switches with $s + m$ interfaces</td>
<td>$n$ switches with $\sum_{i=1}^{n} m_i$ interfaces</td>
</tr>
<tr>
<td>Ethernet lines, passing through each module</td>
<td>$s + m_i$ lines from switch to other modules</td>
<td>In the worst case($\sum_{i=1}^{n} m_i - m_k$ lines, where $k$ – module with a switch)</td>
</tr>
<tr>
<td>Additional restrictions</td>
<td>Max 4 switches on the path of “point-to-point” connection to minimize collisions</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 2. Different topologies’ properties.
Variables, used in table 2: \( n \) – count of modules in a robot; \( m_i \) – count of Ethernet connections per module (in most cases \( m_i = 1 \)); \( s \) – count of potential module neighbors (in most cases either 2 or 6).

As can be seen from the table, both topologies have drawbacks – The “bus” requires either to use a large number of switches, or to integrate several Ethernet interfaces into each device (similar to EtherCAT), whereas the “star” causes a much larger number of wires to be laid. The latter, in our case, does not have a significant value, since the wired system of the local computer network is located within the small dimensions of the robot design. Therefore, it is proposed to use the topology “star”.

4. Communication networks: intermodular communication method

One of the most known intermodular communication method is ROS (Robotics Operation System) framework [15]. To use ROS, one needs a computer with a Linux-like operating system (OS), for which there is a port of ROS. Also, a large number of visualization, simulation and debugging tools were developed for it. At the moment ROS has several disadvantages - it was not designed to work directly on microcontrollers and it requires installation of a quite complex and voluminous software. Therefore, it is necessary to develop such a system so that it would be, on one hand, compatible with ROS and, on the other hand, convenient to implement on embedded systems.

Previous approach, proposed in [16], was based in Internet laboratory library (zmq_robot)[17]. It is built on top of ZMQ library [18] and allows creation of both modules’ control systems and supervisor client software. This library has the same disadvantages as ROS – it requires installation of complex software, which only works on full-fledged OS, but not on embedded systems.

Intermodular communication method specification was created to solve this problem. It takes into account full functionality property of modules. For communications it is proposed to use ZMQ library instead of zmq_robot. In addition to ZMQ it is proposed to use UDP protocol with broadcast messages. This extension enables the usage on most basic embedded devices. Unlike ROS, each module has one interface common for all modules, through which it provides general information about itself – the name of the module, its description, functional interfaces, etc.

5. Hardware and software requirements for control systems’ implementation

Hardware and software of module’s control system, in a minimum implementation, must have a computing device capable of working in the Ethernet through UDP protocol and with computing power sufficient to realize functional purpose of the module. It should have minimal possible cost, sizes and mass. As it was noted earlier, a distributed robot’s CS as a whole is formed from the hardware and software of all modules. Due to the distribution of calculations between the microprocessors of the CS modules, the robot has a higher performance compared to conventional control systems that are built on a single computer.

The analysis of microcomputers of various manufacturers was carried out, among which a multicriteria Pareto optimization was carried out. The optimal candidates for hardware are Arduino Uno with Ethernet Shield, Arduino Mega with Ethernet Shield, Taijiuno Due Pro R3, Seeeduino Cloud, 8duino ZERO, 8duino One and Raspberry Pi B Plus.

6. Modules’ command systems

Based on the principle of full functionality of the modules, it is possible to distinguish transport, energy, information, communication, technological, auxiliary functions and the general control function. Within the developed specification of the intermodular communication system, command systems have been developed for the following functional modules of the modular MR:

1. Transport module – provides movement of the robot (transport function).
2. Power module – provides power supply to electronic and electromechanical components of the robot, and their safe switching on and off (energy function).
3. Sensor system: short-range and long-range sensor modules - modules provide search for obstacles and objects of manipulation at different distances and provide sensory information to other modules (information function).
4. The intellectual control module – provides control of the robot as a whole, synchronizes information interaction of all modules, performs information processing and setting of tasks coming from the external supervisor of the robot, generates and distributes tasks between modules, being thus a supervisor for them, creates and controls The operation of the radio channel (communication function and the function of the general control system).

7. Experimentation

To study the efficiency of the developed method of intermodular communication, a working model of a modular MR was created, consisting of: a wheeled transport module (Arduino Yun running OpenWRT), a power module (Arduino Mega with an Ethernet Shield module) and a network management module (Raspberry Pi B+ running Arch Linux ARM).

Control programs for the IMCS of modules were created on top of the above specification. For the power module, as the least productive part of the system, it was possible to achieve a message exchange rate of up to 500 Hz. The experiments showed the effectiveness of the proposed solutions.
8. Conclusion

A specification of the intermodular communication method, built on principles of ROS, but able to run on embedded computers, was created. This specification is using the full functionality principle as a basis.

These results were achieved:
1. Ethernet with “star” topology was chosen to be communication network for heterogeneous modular robot due to its versatility and wide prevalence of devices implementing it.
2. The intermodular communication method specification was created. It is based on ideas of ROS. The specification is built on top of ZeroMQ TCP messaging library with additional UDP sockets as a mean to enable direct communication with embedded microcontrollers via the Ethernet. The latter cannot be achieved with ROS due to its design.
3. Embedded computers of different models and manufacturers were analyzed and Pareto optimization was carried out among them. Out of 10 selected, 7 computers are, by different criteria, efficient as processing units for modules of a modular robot.
4. In the first approximation, the main function and system of commands for transport, power, sensor and intellectual control modules have been defined.
5. The specification was implemented for 3 embedded devices (Arduino Yun, Arduino Mega, Raspberry Pi B+). Each device was used in a different module respectively – transport, power and supervisor control modules. Control systems for these modules were created using implementations mentioned above with respect to their systems of commands developed earlier.

In the future it is necessary to implement the specification for other architectures of embedded devices (such as ARM or Xtensa). It is necessary to develop systems of commands for other types of modules, such as manipulation module.

9. Acknowledgments

The authors are expressing their sincere gratitude to S. Kuvshinov, Director of International Institute of the New Educational Technologies of Russian State University for the Humanities, for assistance in construction of the experimental equipment.

This work was supported by the Russian Foundation for Basic Research, grants no. 16-07-00811a & 16-07-01264a.

10. References


