Smart-M3-Based Robot Interaction in Cyber-Physical Systems

Nikolay Teslya[*†], Sergey Savosin[*]
[*]SPIIRAS, St. Petersburg, Russia
[†]ITMO University, St. Petersburg, Russia
{teslya, svsavosin}@iias.spb.su

Abstract— **Nowadays cyber-physical systems are spreading wide all over the world. One of the examples of cyber-physical system is a "smart home" devices interaction for home cleaning. The paper presents the interaction process between several physical devices through the Smart-M3-based smart space. These devices are robots that are modelling the vacuum cleaner behavior. For this purpose, the scenario of robots interaction in smart space is described in the paper. For the scenario implementation, robots should be constructed and programmed. The paper provides short survey of the most powerful robotic kits and description of its programming possibilities. The Lego Mindstorms EV3 Education kit is chosen for scenario implementation. The LeJOS is installed on the control block to extend its functionality. This functionality includes connection to the Smart-M3-based smart space and Java support for programming robot actions.**

I. INTRODUCTION

Nowadays cyber-physical systems are spreading wide all over the world. They are based on the real time interaction between physical world and cyber world. Such systems rely on communication, computation and control infrastructures commonly consisting of several levels for the two worlds with various resources as sensors, actuators, computational resources, services, etc. The interaction between system components can be based on a smart space technology, which is a part of both cyber and physical world. Physical devices, which solve different tasks together, are presented by services in the smart space. Every device is described by an ontology, which provides device's abilities and restrictions.

One of the examples of cyber-physical system is a "smart home" system [1-3]. There are many definitions and interpretations of "smart home" term (see S.Solaimani et.al [4]). In this paper, the following definition is used: "Smart Home is a house or living environment that contains the technology to allow devices and systems to be controlled automatically." [5]. The example of automatically controlled devices and systems is a robots interacting for home cleaning [6]. The following devices are used for this purpose: (i) robot vacuum cleaner that can create room map using the light-sensitive sensors and clean room using the map (e.g. Yujin Robot iClebo Arte YCR-M05), (ii) manipulating robot (e.g. FESTO Robotino XT), allowing to move light furniture (e.g. chairs, or coffee table) or decorative elements (pots with flowers, vases, etc.) for efficient cleaning, (iii) adaptive illumination control system for illumination level changing, depending on the current situation. All of them should interoperate each other to solve different tasks. For example, vacuum cleaner needs light to create a room map. Therefore, it should "ask" illumination control system to increase illumination level in the room. Vacuum cleaner can find different hindrances while cleaning. In this case, it should to "ask" manipulating robot to move hindrances to another place until the cleaning would not be ended.

The simple scenario can describe this example. Two or more robots receive a task to execute actions, e.g. find an object and bring it to a storage. Only one robot should handle this task. For this purpose, robots should interact to find the one who will bring the object to the storage. Interaction process includes the object finding, distances measurement and information sharing between robots.

This work presents the interaction between several robots through the Smart-M3-based smart space. The interaction is based on the scenario presented above. Robots can move through the physical space, manipulate objects, read and interpret data from sensors, interoperate through the smart space using ontologies and make simple decisions about future actions.

For the robot constructing, it is suggested to use the robot kits applicable in the education process, which allow to design and built robots from modules. The kit should meet several requirements to provide functionality for the scenario implementation: it should include at least distance and brightness sensors, gyroscope, a set of motors, and powerful control block with programming possibilities and wireless network connection for smart space join.

The paper is structured as follows. Section 2 discusses the existing education kits for rapid robot constructing. Section 3 introduces alternatives for Lego Mindstorms EV3 integrated operating system. These alternatives allow using high-level programming languages for robot activity programming. Section 4 describes the Smart-M3 platform

ISSN 2305-7254

used for smart space organisation. Section 5 provides detailed description of the scenario. A case study based on the provided scenario is described in Section 6.

II. EDUCATION ROBOT KITS

The education robot kits are used for robotic education. Robotics is an area of study that incorporates elements from mechanical engineering, electrical engineering, and informatics. Using of these kits help students to improve their knowledge in many disciplines, such as mathematics, computer programming, electronics, physics, as well as obtain experience in the robot constructing. M. Ruzzenente et.al. divide five categories of robotic kits [7]:

- Building Body Kits that permit the creation of a mechanical skeleton of a robot.
- Electronic Components that is focused on microprocessors, electro-mechanical components and sensors.
- Software Kits: development environments, that often include simulation environments to allow testing and development of specific firmwares;
- Programmable Robots kits where the robot offers no flexibility in terms of hardware and electronics expansion but allows the user focus on the reprogramming of their firmware.
- Complete Starter Kits. This category includes all robots that allow flexibility in terms of body design, electronics, mechanics and software functionality.

The following survey of robotic kits will be concentrated on the complete starter kits, because in contrast to other kits they do not require particular mechanical and electronic knowledge and provide easy to use environment for programming. M. Ruzzenente et al. divide two classes of complete starter kits: versatile and non-versatile. Versatile kits are chosen for this work because they are consist of set of sensors, motors and actuators, and different structural elements for mechanical part constructing, that allow to construct robots with different morphology without difficult process of sensors, motors and chips developing. These kits also include controller board to control the inputs and outputs of the robot and provide environment for robot programming. These characteristics allow concentrating on the scenario developing without spending resources to robot development. The following versatile complete starter kits will be described in subsections: VEX Robotics Design System, Lynxmotion Servo Erector Set, and Lego® Mindstorms.

A. VEX Robotics Design System

VEX Robot Starter Kits[1] include everything needed to build a fully-functional pre-programmed. There are two available kits: fully-autonomous with programming and manually operated with dual control.

Programming kit consists of the following components (Fig. 1):

- Vex Clawbot Kit with 4 motors and about 300 structural parts.
- VEX ARM® Cortex®-based Microcontroller. (ARM® Cortex® M3 processor ARMv7, 72 MHz, 64 Kb RAM, 384 KB program space).
- Sensors: two limit switches, two bumper switches, ultrasonic range sensor (distances from 1.5 in. to 115 in.), three infrared light sensor and infrared LED.

A huge amount of other sensors and motors can be bought separately.

Fig. 1. VEX Programming Control Starter Kit

The dual control kit does not include sensors, but provides manual control by using wireless joystick. Joystick works with VEX ARM® Cortex®-based Microcontroller, so the robot still can be autonomous.

Programming of the microcontroller can be made by using ANSI C programming language. The analogue is an EasyC[2] language, which is a drag and drop programming language for beginners. It uses pre-made blocks to help users get started with programming and it allows users to program faster than using a standard coding language.

B. Servo Erector Set

The Servo Erector Set kit[3] provided by the Lynxmotion manufacture contains of over 500 components. They include (Fig. 2):

[1] http://www.vexrobotics.com

[2] http://www.intelitekdownloads.com/easyCV4/

[3] http://www.lynxmotion.com/c-73-servo-erector-set.aspx

Fig. 2. Lynxmotion Servo Erector Set (S.E.S.) V1.1

- BotBoarduino Shield-Compatible Robot Controller PCB (Arduino-based controller).
- IR Range sensor (distances from 10 to 80 cm).
- Bluetooth module.
- 16 different motors.
- Other parts.

Other sensors and motors can be bought separately. The third-party sensors and motors are also supported. Every part of the set is well documented with full physical characteristics such as pins, produced signals, amplitude-frequency characteristics and other technical information, so the developer can control every nuance of the system.

Programming is available via FlowBotic Studio[4] (FlowStone graphical programming language with supporting writing custom modules in Ruby) or Arduino programming language[5].

C. Lego ® Mindstorms

Lego® Mindstorms[6] is one of the most popular sets in robot constructing education. The latest system, called the Lego® Mindstorms EV3, was released on September 1, 2013 [8]. A standard Education EV3 Core set consists of the following components (Fig. 3 shows only control block with available sensors and motors):

- Control block called Brick (ARMv9 core CPU 300 MHz, 64 Mb RAM, 16 Mb flash memory and microSDHC port (supports microSD and microSDHC memory cards with capacity up to 32 Gb), USB host, WiFi through USB dongle, Bluetooth, speaker, LCD display and 6 hardware buttons).
- Motors (one medium motor and two large motors).
- Sensors: ultrasonic sensor (for distance measurement, from 3 to 150 cm), touch sensor (for handling touching), gyroscopic sensor (for measurement of turn angle and turn acceleration), light sensor (for brightness measurement and color detection).
- More than 550 Lego parts.

Additionally infrared sensor with remote control and third-party sensors developed by Dexter Industries can be bought. Up to four EV3 control blocks can be connected using a USB cable and thereby enabling robot to have sixteen output ports and sixteen input ports all controlled from the main EV3 Brick.

Lego® Mindstorms has own IDE for robot actions programming. It allows to program actions by graphical programming language using blocks with customizing parameters. These blocks cover all robot functionality, but for the advancing features, it is possible to provide environment for development with high-level program languages.

Robot can also be controlled manually by the iOS[7] and Android[8] applications via Bluetooth connection.

Most of the other kits either are built based on the described sets or provides less functionality of sensors, motor, or control block. Among the presented kits, only Lego® Mindstorms EV3 meets requirements defined in Section 1. It provides the needed sensors and motors set as well as wide possibilities to reconfigure the control block for using high-level program languages for robot activity programming.

III. Operation Systems for Lego® Mindstorms EV3 control block

There are many projects providing usage of the different program languages support for EV3[9]. One of them provides

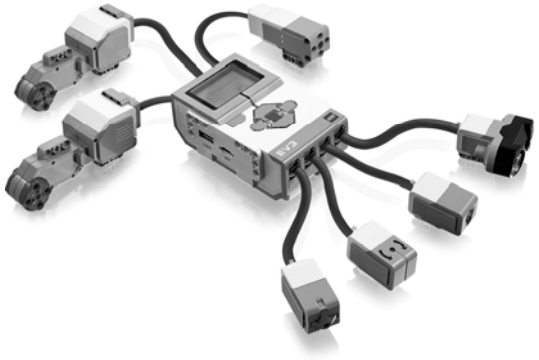

Fig. 3. Lego® Mindstorms EV3 Education kit

[4] http://www.flowbotics.com/
[5] http://arduino.cc/en/Reference/HomePage
[6] http://mindstorms.lego.com

[7] https://itunes.apple.com/app/lego-mindstorms-robot-commander/id681786521
[8] https://play.google.com/store/apps/details?id=com.lego.mindstorms.robotcommander
[9] http://en.wikipedia.org/wiki/Lego_Mindstorms#Programming_languages_2

environment for compiling programs under existing control brick's OS (e.g. NXTGCC, Lego.NET, different libraries for GCC, etc.) The second class of projects provides controlling the EV3-based robot using different languages through the Bluetooth and/or USB interfaces (NXT_Python, OCaml-mindstorm, LabVIEW, etc). The third class provides a replacement of the existing OS. Due to the fact, that Lego® Mindstorms EV3 control block is based on the ARMv9 processor and the main OS is Linux-based, it is possible to run another Linux-based OS, that is built for ARM architecture. Using Linux-based OS allows writing programs with any supported programming language. There a two ways of replacement: replacement of the kernel embedded into the control block (ROBOTC), and additional OS on SD-card without replacing the existing OS (brickOS, LeJOS, ev3dev).

The case when there is an additional OS without replacing the existing is more preferred in the view of the fact that it cannot corrupt the data on the brick. Existing OS still can be booted and used. Interaction with control block in the case of replacement is faster than through using the Bluetooth interface.

One of the Linux-based systems is a project called brickOS[10]. This is an open source embedded operating system providing a C and C++ programming environment for the Lego® Mindstorms Robotics Kits.

The alternative to brickOS is a project called LeJOS[11] (acronym from Lego Java Operating System). It provides the full featured embedded OS with GUI. LeJOS provides Linux environment with Java Runtime Environment. LeJOS Java bindings implement access to the robot's hardware. It allows Lego Mindstorms robots to be easily programmed with Java language and to interface the software with most Java resources in term of API. Some of LeJOS main features are [9]:

- object oriented language (Java);
- pre-emptive threads (determined context switching);
- (multi-dimensional) arrays;
- synchronization;
- exceptions;
- types of variable including float, long, and String;
- most of the standard Java classes are available;
- well-documented robotics APIs.

The ev3dev[12] is another interesting project that provides a Debian-based OS for EV3. It provides a console system available through SSH, which can be updated and extended by using extensive Debian repository. For hardware access, ev3dev provides drivers and language bindings for C++, Google Go, Node.js and Python programming languages. The other languages also can be used, but it is need to implement interface libraries by yourself.

Due to the Linux kernel all systems includes firmware for the wireless network USB-adapters. It makes possible to connect robots to the local area network and unite them to the smart space. For the robot realization, the LeJOS has been used because it provides the most powerful environment for the program development (threads with synchronization, standard classes and types of variable) as well as wide device type support (all native devises and 3rd party sensors and motors).

IV. SMART-M3

For the interaction between robots, a smart space technology has been used [10]. The smart space is based on the Smart-M3 platform[13] [11]. The main difference of this platform compared with other existing solutions described by Liuha et al.[12], Johanson et al.[13], Xie et al.[14], and Martin et al. [15] is that the Smart-M3 is an open source platform, it is accessible for downloading and testing, supported by development community (last accessible version has been uploaded on the 04.02.2013), and supports modern mobile platforms (Android, Symbian, Windows Phone).

The key ideas of the Smart-M3 platform are device, domain, and vendor independent. Another key idea is that devices and software entities can publish their embedded information for other devices and software entities through simple, shared information brokers . Information exchange in smart space is implemented via HTTP using Uniform Resource Identifier (URI) [16]. Semantic Web technologies have been applied for interoperation purposes. In particular, ontologies are used to provide semantic interoperability between robots and control devices (smartphones and/or personal computers). Characteristics described above allow combining robots, smartphones, and computers into one information space to provide cooperation between them, control robots, and gather statistics.

Smart-M3 platform consists of two main parts: information agents and kernel (Fig. 4). The kernel consists of two elements: Semantic Information Broker (SIB) and data storage. Information agents are software entities installed on the devices, which are interoperate through the smart space. These agents interact with SIB through the Smart Space Access Protocol (SSAP) [11]. The realization of the protocol is available by libraries for the most popular programming languages, like Java, C/C++, Python, C#. The SIB is the access point for receiving the information to be stored, or retrieving the stored information. All this information is stored in the data storage as a graph that conforms to the rules of the Resource Description

[10] http://brickos.sourceforge.net/
[11] http://www.lejos.org/
[12] https://github.com/ev3dev
[13] Smart-M3. URL: http://sourceforge.net/projects/smart-m3

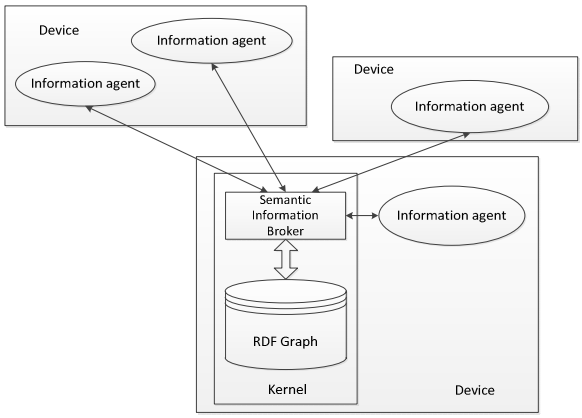

Fig. 4. Smart-M3 Platform

Framework (RDF)[14]. By these rules, all information is described by triples "Subject - Predicate - Object".

V. ROBOTS INTERACTION SCENARIO

The following scenario is proposed to implementation: two (or more) robots should find an object around each other and then the nearest robot move to the object. They do not know about the position of another robot and moving to another robot is restricted. In this way, the whole task can be split to the several subtasks:

- Task 1. Each robot should scan an area around. While scanning robot controls the incoming data from the gyroscopic & ultrasonic sensors and recording turn angle with measured distance.
- Task 2. Each robot should find the objects. The measurements from the previous step are processed. Processing determines objects that were captured by the ultrasonic field of vision. The result of parsing will be angle where the object was found and distance to this object. In Fig. 5. "Robot 1" finds two objects. First is on the angle 1.1, at distance 1.1; and second is on the angle 1.2 at distance 1.2. The same situation is for the "Robot 2".
- Task 3. Each robot should find another robot. Robots shares found objects through the smart space. Then they compare the distances to objects. The object will be find in the case of distance1.1 is not equal to distance 2.2 (Fig. 5). Only two distances should be equal for this purpose — distance 1.2 and distance 2.2. If there are more than two equal distances, robots should check that the object is not the other robot. For this purpose random robot moves to object and repeat all moves from the Task 1. These steps are repeated until only two distances become equal.
- Task 4. Each robot should interoperate with another robot and decide who will go to the object. When the object is found robots choose the shortest distance to the object and the robot that owns the distance moves to the object.
- Task 5. Selected robot should carry out defined task with the object.

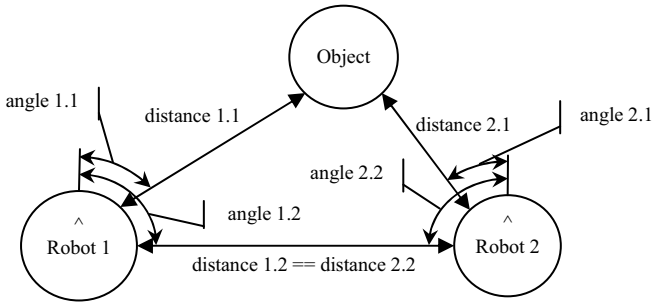

Fig. 5. Object finding scenario

The scenario has a real life application. For example, there are two robotic vacuum cleaner in the house. Before the cleaning they creating a map of dirtying and split the house to the two parts based on the nearest dirty points. The other example is a situation when vacuum cleaner finds a hindrance, for example big toy or chair. It shares information about the situation through the smart space and manipulating robots should decide which of them would go to remove the hindrance.

For the scenario the ontology, presented in Fig. 6 has been designed. There are two main entities in the ontology: robot and object. The object is corresponding to any thing around robots, including the robot itself. The object entity is described by two characteristics: angle and distance. These characteristics are unique for each robot; therefore, two objects in the smart space can present one object in physical space.

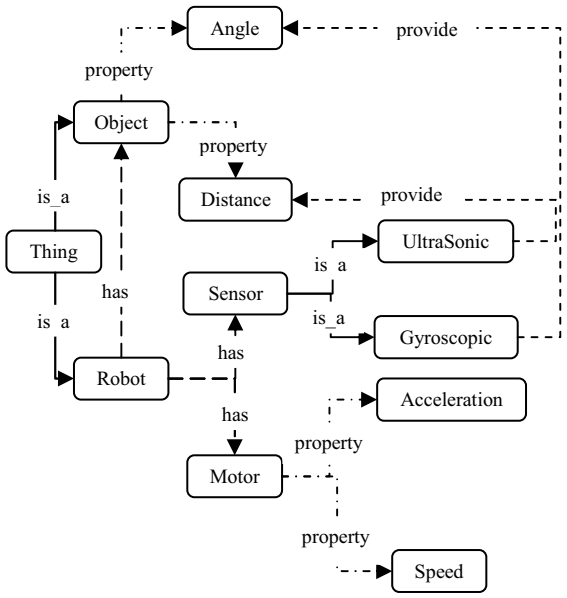

Fig. 6. Searching object ontology

[14] RDF. W3C standard. URL: http://www.w3.org/RDF/

The *robot* entity describes only robots. It has two types of characteristics: *motor* and *sensor*. The *motor* entity describes the current characteristics of the robot's motor, such as *speed* and *acceleration*. It also can be controlled by the commands from the smart space. The *sensor* entity describes the sensors available for robot. For the purposed scenario, only two types of sensors are needed: *gyroscopic* and *ultrasonic*. The *gyroscopic* sensor provides a turn *angle* of the robot as well as *angle* of a found *object*. The *ultrasonic* sensor provides a *distance* to the found *object*.

VI. SCENARIO IMPLEMENTATION

The scenario presented above has been implemented by using two autonomous robots (Fig. 7). These robots are Lego EV3 cars with ultrasonic and gyroscope sensors. Each car is driven by two independent large motors and controlled by control block with LeJOS installed on the SD-card and WiFi USB-adapter for local area network connection (Fig. 8 and Fig. 9). For the control block the LeJOS has been chosen because it is provide full functionality OS with JRE Environment as well as plugin for Eclipse IDE for easy developing and debugging of applications. Applications can be developed in other IDE/editors but it is need to compile a *.jar file and move it to the control block through SSH.

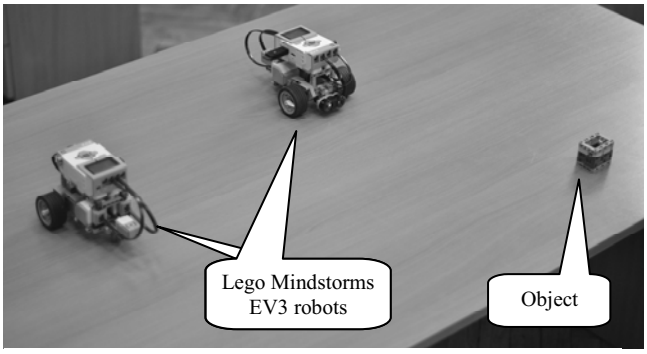

Fig.7. Scenario implementation

The architecture for the scenario implementation is presented in Fig. 8. Robots (Robot 1 and Robot 2) are connected to the Smart-M3-based space. The control device is a smartphone. It can send commands to smart space for start and stop robot actions as well as gather information from smart space about current state of each robot. Each robot is subscribed to commands patterns and information provided from the other robots in smart space. EV3 Control block provides four input and four output ports for connecting devices (motors and sensors). These ports support data feedback from devices using one of the following approaches: analog values, I2C communication, UART communication.[15]

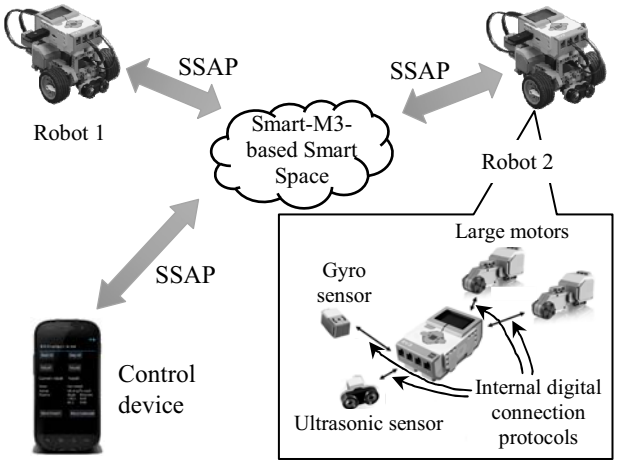

Fig. 8. System architecture for the scenario implementation

Connection to the Smart-M3-based smart space is provided by using the Java KPI library[16]. It provides function for all operations described by SSAP.

After the start command receiving from the smart space, each car rotate at 360 degrees and fetch information about turn angle and distance from the gyroscopic and ultrasonic sensors correspondingly. This information stores for the future processing to separate found objects. The separation algorithm discerns objects by analyzing difference between neighbor distances. Each object includes an angle when it was firstly found and angle when the last measurement has been discerned as well as corresponding distances. The real object angle and distance to the objects is an average value between these values.

All found objects are sorted by the average distance and shared through the smart space. Each robot has subscription to new objects that are appearing in the smart space. When the objects from the other robot appear, they are queried by

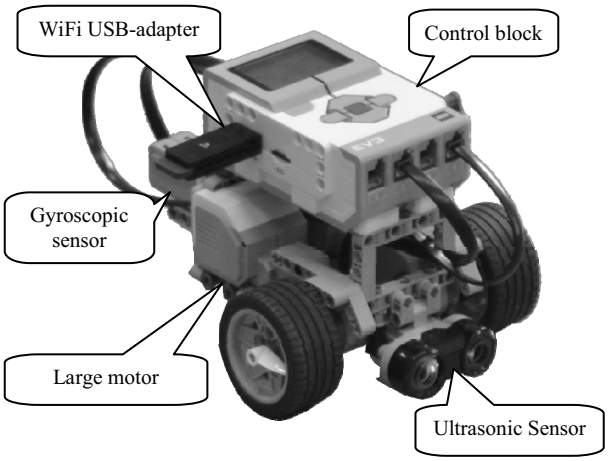

Fig. 9. Lego Mindstorms EV3 robot

[15] http://www.legomindstormsev3.com/ev3-hardware/lego-mindstorms-ev3-programmable-brick/ev3-technical-overview/

[16] http://sourceforge.net/projects/smartm3-javakpi/

listening robot and comparing with existing set of its objects. The values equality means that robots find each other and these values should be excluded from the set. If there are more than two equalities, robots should move toward one of the equal objects and repeat the area scanning. When robots find each other, they can detect the object and select the nearest robot. This robot turns to the angle the object is located in and moves to the object.

The following features have been noticed while the scenario implementation. Sensor handling has an unclear lag time as well as reaction on the commands to motors. Therefore, the robot can stop a little bit further than it should. In addition, the distance sensor has rather wide range of scanning. The one object can have different, non-equal measurements of distance by this reason. A lot of the working time is spending to the object detection. Hypothetically, this is caused by the defects of object searching algorithm and small power capacity of the control block.

VII. CONCLUSION

The paper describes the interaction of several robots through the Smart-M3-based smart space. For developing the interaction between robots, the scenario has been proposed. This scenario presents the case when two robots should interoperate to solve task together. The case study shows the scenario implementation.

Robots based on the Lego Mindstorms robot-constructing kit has been built for the scenario implementation. This kit provides high functionality for constructing different kind of robots. The control block is based on the standard ARMv9 architecture. That allows running environment for robot activity programming.

The Smart-M3-based smart space is used for the interaction between robots. This approach allows uniting different kind of devices in single environment to share information and interoperate in real time to solve assigned tasks.

For the future work the following tasks is defined:

- Decrease object searching time as well as accuracy of objects detection.
- Raw sensor data processing has to be improved.
- More complex scenario can be implemented based on the case presented in the paper.

ACKNOWLEDGMENT

The research was supported partly by projects funded by grants # 14-07-00363, # 13-07-00271, # 14-07-00378, # 12-07-00298, # 13-07-12095, and # 13-07-00336 of the Russian Foundation for Basic Research. The presented work is also a part of the research carried out within the projects funded by grant 074-U01 of the ITMO University.

REFERENCES

[1] A.V. Gavrilov, Iskusstvenniy Domovoy, Iskusstvenniy Intellekt i Prinatie Resheniy, vip. 2, 2012, c. 77–89.

[2] P. Carreira, S. Resendes, A.Santos, Towards automatic conflict detection in home and building automation systems, *Pervasive and Mobile Computing*, vol. 12, 2014, pp. 37–57.

[3] C.Belley, S. Gaboury, B. Bouchard, A. Bouzouane, An efficient and inexpensive method for activity recognition within a smart home based on load signatures of appliances, *Pervasive and Mobile Computing*, vol. 12, 2014, pp. 58–78.

[4] S. Solaimani, W. Keijzer-Broers, and H. Bouwman, "What we do and don't know about the Smart Home: an analysis of the Smart Home literature," Indoor Built Environ. 2013.

[5] Y.P. Cong, Z.Q. Wei and M.D. Hu, A Smart Home architecture based on concept ontology. *Appl Mech Mater* 2013; 303–306(February): 1559–1564.

[6] A.M. Kashevnik, A.V. Ponomarev, S.V. Savosin, Hybrid Systems Control Based on Smart Space Technology. *SPIIRAS Proceedings*, in press.

[7] M. Ruzzenente, M. Koo, K. Nielsen, L. Grespan, and P. Fiorini, A Review of Robotics Kits for Tertiary Education. *Proc. of 3rd Int. Workshop "Teaching Robotics, Teaching with Robotics"*, Integrating Robotics in School Curriculum Rivadel Garda(Trento, Italy), April 20, 2012, pp. 153–162.

[8] Coming Fall 2013: LEGO MINDSTORMS EV3, URL: http://www.lego.com/en-us/mindstorms/news/2013/january/announcing-lego-mindstorms-ev3/

[9] R. Grandi, R. Falconi and C. Melchiorri, Robotic Competitions: Teaching Robotics and Real-Time Programming with LEGO Mindstorms. In *Proceedings of The 19th World Congress of the International Federation of Automatic Control (IFAC 2014)*, Cape Town, South Africa | August 24-29, 2014.

[10] A. Smirnov, A. Kashevnik, N. Shilov, et. al., "Anonymous Agent Coordination in Smart Spaces: State-of-the-Art", in *2nd Conf. on Smart Spaces – ruSmart* 2009. LNCS Vol. 5764, Aug. 2009. pp. 42-51.

[11] J. Honkola, H. Laine, R. Brown, O. Tyrkko, "Smart-M3 Information Sharing Platform," *Proc. IEEE Symp. Computers and Communications (ISCC'10)*. IEEE Comp. Soc., pp. 1041-1046, June 2010.

[12] P. Liuha, A. Lappeteläine, J.-P. Soininen, Smart Objects for Intelligent Applications, *ARTEMIS mag.*, vol. 5, 2009: 27–29.

[13] B. Johanson, A. Fox, P. Hanrahan, T. Winograd, The Event Heap: An Enabling Infrastructure for Interactive Workspaces, *Proc. of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, 2001.

[14] W. Xie, Y. Shi, G. Xu, Y. Mao, Smart Platform — A Software Infrastructure for Smart Space (SISS), *Proc. Fourth IEEE Int. Conf. on Multimodal Interfaces*, 2002, 429–434.

[15] D. Martin, A. Cheyer, D. Moran, The Open Agent Architecture: A framework for building distributed software systems, *Applied Artificial Intelligence: An Int. J.*, vol.13 (1-2), 1999: 91–128.

[16] T. Berners-Lee, R. Fielding, L. Masinter, RFC 3986 – Uniform Resource Identifier (URI): Generic Syntax, URL: http://tools.ietf.org/html/rfc3986