# M3-Driven Smart Space Creation Using a DD-WRT-Based Device

Sergei Mikhailov[1,2], Alexey Kashevnik[1,2]
[1]ITMO University, Saint Petersburg, Russian Federation
[2]SPIIRAS, Saint Petersburg, Russian Federation
saboteurincave@gmail.com, alexey@iias.spb.su

*Abstract*—**The paper describes the process of smart space creation based on integration of Smart-M3 platform with a DD-WRT-based device. Smart-M3 is an open source platform which implements concept of smart space. Wi-Fi router is used as platform hosting which reduces the number of devices participating in smart space-based scenarios. The article covers a process of compilation and installation of Smart-M3 platform on DD-WRT-based Wi-Fi router. Evaluation shows that smart space organized this way can be used for scenarios with few participants. The authors developed "Smart-M3 Control Panel" web-service which allows users to control Smart-M3 platform by a graphical web interface. "Smart-M3 Control Panel" user can view the current status of platform; launch, stop, and reload it; view information storage content and change it; download log files; and change startup options. SocketIO interface was used for the user interaction with web service.**

## I. Introduction

Smart space [1] is a set of different devices with the opportunity of information and knowledge sharing between them. Smart-M3 platform is an open source product which implements this concept. The key idea of this platform is formation of device-, domain- and vendor independent smart space. Smart-M3 platform can be used in different scenarios: [2], [3], [4], [5].

The authors have made a number of scenarios for collaborative mobile robots' work based on robotic set Lego Mindstorm EV3 using Smart-M3 [6], [7], [8], [9]. The platform is installed on the personal computer with the Unix-like operating system. Data exchange opportunity between mobile robots, for example by the Wi-Fi, is required. For more convenient utilization of devices involved in scenario the authors have examined an opportunity to install the platform on a router and suggested the way of doing it. Platform installation on the router makes robotic scenarios more mobile by excluding the use of personal computers; this reduces the number of devices for scenario deployment.

The following platforms have been researched for smart space creation on a router. After exploration none of platforms are suitable for mentioned scenarios. The authors have decided to adapt Smart-M3 platform for operating on a DD-WRT router. By default, the router does not provide access to memory, so it is necessary to use alternative operating systems.

The authors have compiled Smart-M3 platform on the router but it is possible to use the cross-compilation technique to speed up a compilation process. Some platform packages required additional configuring or specific versions, or manipulating with source code. Performance evaluation of Smart-M3 platform on router and personal computer has shown that the router processing power is enough for scenarios with a small number of participants. The authors have developed the web-service "Smart-M3 Control Panel" for control Smart-M3. This service is installed on the router and allows end-users to interact with platform.

The rest of the paper is structured as follows. Related work is presented in section II. Section III describes Smart-M3 platform. Section IV gives restrictions on a router choice and presents router preparation before Smart-M3 platform installation, which is defined in section V. Section VI describes a web-service "Smart-M3 Control Panel" for Smart-M3 platform's operation. Performance comparison of Smart-M3 platform on router and personal computer in section VII. The results are summarized in Conclusion.

## II. Related work

Platform meSchup IoT [10], [11] brings a device interaction concept within smart space by pluggable applications, which work on a central server and control devices based on the received data. The applications can work in parallel on the central server and leave server at any moment. MeSchup IoT platform allows to integrate devices from different manufacturers (Android, Arduino, .Net Gadgeteer and nRF51822-based hardware, computers based on Windows and Linux OS) for co-operative joint task solving. The platform has server and client software. Client software has to be installed on devices for sending data to the central server and getting instructions how to act from pluggable applications. Server software deals with applications, provides communication between participants of the smart space and gives an access to all information from the devices. Interaction scripts are developed in JavaScript language.

CHROMOSOME solution [12], [13] is supposed to facilitate the integration of heterogeneous components in the automation of industrial and everyday tasks in a smart home. It receives data from devices in wireless networks and provides communication between autonomous automated systems via high-level management platform. The architecture of the platform was developed using hardware abstraction level upon a hardware. Key services of the platform present a data-centric way of communications, and high-level components handle the logic of applications. A topology of system can change during CHROMOSOME platform operation because of entering or leaving devices. For solving this problem the platform calculates a new lookup table without system work interruptions. The Datacentric way of communications is based on “publication/subscribe” and “request/response” principles. Interoperability between components of an application from different developers is provided by modeling knowledge domain with “dictionaries” containing terms and concepts of themes. Besides, the clarification of semantic items’ in theme, “Dictionary” specifies the attributes of objects that describe the subject more widely. CHROMOSOME supports Windows, Linux and FreeRTOS operating systems.

Article [14] describes the smart space creation using the upper level ontology, which allows to interact among various applications in different environments. Working with ontologies (storing of information, context data processing, making queries to the information storage) was implemented by Jena2 framework for semantic web applications creation . Locating of smart space uses event tracking and message sending based on Siemens UPnP SDK.

Papers [15], [16] represent a solution for creating applications simultaneous operating in multiple smart spaces on the basis of Smart-M3 platform. The solution suggests two types of application programming interfaces: semantic end-user application programming interfaces (S-API) and interface of program execution (Execution API). Both interfaces work together providing a dialogue between the smart space and executive program (Driver Component), which in turn works with a professional programming interface of a device. RDF-Script is used as a programming language and allows users to describe workflow action for applications using multiple smart spaces. An “application executor” component monitors the overall performance of the program and execute RDFScript code for entrance into smart space and selected actions implementation. Smart Modeler program allows ordinary users to make applications visually, creating a set of commands or a set of goals to be achieved by the application.

XANA solution [17] expands a product Team Computing [18] (TeC) by software product line (SPL) concept. This solution intends to simplify the process of creation, configuration and deployment of applications for smart homes. SPL involves the creation of applications with the help of smart homes experts, and the end users complete the resulting configuration by using parameters from specialists. TeC Platform is an event-oriented and provides developers with a language that allows to use charts for work in a team to achieve common goals. XANA Solution was tested using hardware X10.

Project PECES (PErvasive Computing in Embedded Systems) [19] aims to create a platform for cooperation of various devices that are located in different smart spaces. To ensure interoperability between devices interacting in various contexts, PECES used pre-designed ontologies. At the core level of the cooperative platform there are two technologies: Aura [20] (the creation of smart environments to provide services) and BASE [21] (the support for the adaptation of communication protocols and technologies). For adapting to the constant changes in the environment the platform automatically creates the device configuration and updates it using the concept of roles and rules. The roles can be assigned to any device, and the rules define the contextual constraints on the assignment of roles to participate in the program. To work with multiple smart spaces PECES project uses the following components: “Coordinator” (a device that is responsible for the identification of participants in smart spaces based on roles and rules), “Member” (dynamic incoming and outgoing from a smart space device, which can be used in work) and “Gateway” (the device that provides a connection between participants of different intellectual spaces).

The smart space organization for scenarios with no more than twenty devices involves the usage of router based on DD-WRT operating system. As some systems are narrow focused (designed to work with intelligent buildings, e.g. sensors — [10], [14], [17]) and others should be installed only a desktop computer [12], [15], [19], they are not suitable for installing on a router since they are too powerful.

## III. Smart space. Smart-M3 platform

Smart space aims to seamless integration of different devices by developing ubiquitous computing environments, where different participants can share information with each other, make various computations and interact for joint task solving.

The open source Smart-M3 platform [22] is based on Redsib software allows to perform access to ontology oriented information and knowledge. Smart-M3 contains kernel and knowledge processors (KP) [23]. Kernel consists of two parts: semantic information broker (SIB) and information storage. SIB receives incoming requests for cooperation manipulations with the information storage from knowledge processor and sends back operation results.

There are different implementations of semantic information broker, provided by different members of the community [24]. The authors have chosen Redsib based Smart-M3 implementation as a most stable version of the platform.

All information from the information storage is kept as a graph that is formed according of the rules of structured data representation RDF — Resource Description Framework. Information in the graph is described as a triplet — “subject — predicate — object” (a subject uses a predicate to affect on an object). The subject and the predicate can be URI (Uniform Resource Identifier), the object can be URI link as well as literal (some value with certain type). Knowledge processors function on the smart space device software. Knowledge processors interact with SIB using SSAP protocol — Smart Space Access Protocol. SSAP protocol operations are described in XML format.

KP can make following operations:

- Join the smart space — knowledge processor has to register before work in SIB, that cooperates with the smart space;
- Insert information into the smart space in the form of RDF-triplet;
- Remove data from the smart space;
- Update information — updating consists of two steps: data removing and then inserting new information;
- Request necessary information by pattern (Query);
- Subscribe to information — knowledge processor subscribes to specific information and at the time when

necessary information appears in the smart space, a notification is sent to KP;

- Unsubscribe from information;
- Leave the smart space.

Platform Smart-M3 [22] can be loaded from the project official repository — Sourceforge site. The Platform is represented in two options: redsib-0.9.2-src.tar.gz — archive with the project packages source code, and rebsib-0.9.2-amd64.tar.gz — compiled system for Ubuntu OS version greater than 10.04 with the architecture x86 or x64. For the platform installation on Wi-Fi router the source code should be compiled for DD-WRT operation system. Packages that should be compiled:

- Libxml — library, that is XML-analyzer.
- Redland — libraries' set, that is used for RDF work. Redland supports work with query language SPARQL, that is necessary for Redland operation.
- Whiteboard — package implements functionality for cooperation of knowledge processors and semantic information broker. It also provides package redsibd work.
- Redsidb — realization of semantic information broker.
- Sib-tcp — realization of sockets functionality. Sockets are required for data transfer between knowledge processors and semantic information broker.

## IV. DD-WRT/OpenWRT based device for smart space creating

A Wi-Fi router with DD-WRT (http://www.dd-wrt.com/site/index) or OpenWRT (https://openwrt.org/) support has been chosen as a base device. These projects use Linux-based operating system and extend router functionality. Setup of alternative operating system grants the access to filesystem of the router using SSH protocol and allows users to install additional software. Complimentary software installation can be done by compiling the source code of software on the router or using the cross compilation technique (compiling the source code for another platform which a compiler is running on) or using special package managers.

The following factors influence on the choice of the router: the possibility of installation DD-WRT/OpenWRT firmware and presence of a USB port (optional requirement). Users can check an opportunity of installation on DD-WRT/OpenWRT project site. If the router has a low amount of non-volatile random-access memory (less than 1 Gigabyte) then it is necessary to choose a router with USB-slots for plugging USB flash memory to store of Smart-M3 packages and software. An alternative operating system should have USB support on its own core, otherwise it is impossible to mount a flash drive.

In this article the authors used Asus RT-N16 router and DD-WRT firmware as an alternative operating system. Instructions for DD-WRT installation on Asus RT-N16 router can be found on the official DD-WRT site (http://www.dd-wrt.com/wiki/index.php/Asus_RTN16). After firmware installation, SSH access to the router can be provided by enabling "Enable SSHd" option in "Service" menu in the router web interface for Asus RT-N16 and then restarting the router.

USB flash drive has to be prepared before using it as the software storage by drive formatting and making disk partition. The following partition sections need to be created:

- 1–2 Gb partition with ext3 filesystem, which would be use as libraries' and applications' storage ("Optware" label or `/opt`);
- 64–256 Mb partition with linux-swap system, which would be use as swap-file ("Swapfile" label);
- Rest of disk's space partition with ext3 filesystem, which would be use as a main storage for packages ("Data" label or `/mnt`).

After the creation of the partitions, it is necessary to enable USB support on router. That can be done through using the router's web interface "Services" — "USB", and enabling the following options: "Core USB Support", "USB Storage Support", "Automatic Drive Mount", and choosing `/opt` partition as "Disk Mount Point". If USB flash drive is plugged and partitions are done correctly, the information about flash drive and mounted `/opt` partition is available on "USB" menu. After these steps, it is needed to create script `startup.bash` (view listing 1) at `/opt` partition which allows to mount "Data" partition and increase available amount of memory of the router `/jffs` partition. Moreover, it is necessary to set the script as "Run-on-mount Script Name" at Web GUI: "Services" — "USB". This action runs `startup.bash` script every time when the router starts working.

**Listing 1** Listing of startup.bash

```
#!/opt/bash
mount /dev/discs/disc0/part3 /mnt
```

## V. Smart space organization

Compiler GCC (https://gcc.gnu.org/) and related libraries should be installed on router for Smart-M3 platform source code's compilation. These components are not available on router by default, so they must be installed independently. Optware packages of project NSLU2-Linux (http://www.nslu2-linux.org/wiki/Optware/Packages?from=Unslung.Packages) installation can help with the components' mounting on the router. Size increase of the file structure JFFS (Journaling Flash File System) is necessary before the installation, that can be done by adding to a bash script `startup.bash` one single line: `mount /opt/jffs /jffs`. This solution provides expansion of the file structure JFFS's size due to the directory `/opt`. Optware additional software packages installation occurs with the script whose content are shown in the listing 2 below:

Command `ipkg-opt install` produces download of the additional software. Following packages are needed for Smart-M3 platform's compilation:

- `ipkg-opt install buildroot` (this package includes GCC compiler)

**Listing 2** Listing of Optware packages installation

```
wget http://www.3iii.dk/linux/
optware/optware-install-ddwrt.sh
-O - | tr -d '\r' >
/tmp/optware-install.sh
sh /tmp/optware-install.sh
```

- `ipkg-opt install optware-devel` (this package contains the number of libraries and header files for program compilation)
- `ipkg-opt install busybox` (this package contains a newer version of Bash interpreter)

All packages will be installed in `/opt` directory which will be used for further Smart-M3 platform compilation.

Router should be prepared before compilation's start by `before_compilation.bash` script's execution. Code of the script is given in listing 3. This script deletes all records about custom dynamic libraries' location and searches executable files in `/opt` directory on the flash driver, that contains compiler and accompanying libraries. These operations are essential because they allow uniquely identify the location of the compiler, libraries and software in `/opt` directory for further successful compilation as some of the necessary libraries are pre-installed in the file system, but they are outdated without updating possibility. Thereafter archive with the source code of Smart-M3 platform should be downloaded on the flash driver.

**Listing 3** Listing of before_compilation.bash script

```
unset LD_LIBRARY_PATH
export PATH=/opt/bin:/opt/sbin:/bin:
/sbin:/usr/sbin:/usr/bin
```

Compilation of Smart-M3 platform goes through the initial dependencies resolution. Each package must be configured for router by configure command which requires the future location for programs and libraries' installation in `/opt` directory of flash driver as a parameter. Compilation can be started by execution of `make` command after the configuration. In the case of the successful compilation the application can be installed with the command `make install`. Installation's common scenario is given in listing 4.

**Listing 4** Common scenario of packages' installation

```
./configure --prefix=/opt
make
make install
```

The authors will mark package that is already in the archive with the <u>underline</u>, otherwise package is additional and should be downloaded and moved to the router's file system.

There are packages for Smart-M3 platform's installation:

- <u>Libxml</u>, <u>Raptor</u> — these packages can be installed according to the listing 4;
- <u>Rasqal</u> — uClibc is a C library, that is used in DD-WRT firmware. It doesn't support of rounding function `round()`, so it is necessary to modify this package for successful compilation and installation. The best way to solve the problem with round function is adding your own realization of this function to the file `src/rasqal_literal.c`, then check for existence of function `round()` must be deleted in the configuration file. This operation can be done in two ways. The first is disable the abnormal termination in the absence of round function in system libraries (for that the line *"AC MSG ERROR([Could not find ceil, floor, round in default libs or with -lm])"* in the file `configure.ac` must be deleted, command `autoreconf i` in the package's root would be execute and then package has to be installed like in listing 4). The second is changing the script configure by deleting the line described above and then installing the package like in listing 4;
- bdb — this software can be installed with the help of package manager: `ipkg-opt install libdb`;
- <u>Redland</u> — it is necessary to configure the platform work by specifying the location of libreries bdb and enabling thread support, described in listing 5;
- libffi-3.2.1, gettext-0.19.2, glib 2.28.2 — this package should be installed as described in listing 4. Later versions are not permissible as they have redesigned thread work and some availabilities that library uClibc doesn't support;
- dbus-1.10.6, dbus-glib-0.100, libuuid-1.0.3, <u>Whiteboard</u>, <u>Sib-tcp</u>, libtool 1.14, <u>Redsibd</u> — these packages can be installed according to the listing 4.

**Listing 5** Installation scenario of package Redland

```
./configure --prefix=/opt
--with-bdb=/opt --with-threads
make
make install
```

Sessional bus `dbus` and programs `redsibd` and `sib-tcp` should be launched for Smart-M3 platform's work. During router's start sessional bus is not automatically functioning, so it is necessary to create it at each start of the platform and destroy it at the end of work.

Launch of `dbus`, that is used for platform's components' communication, can be done by the `dbus-launch` command execution. Programs `redsibd` and `sib-tcp` require location and identifier number of session in environment variables DBUS_SESSION_BUS_ADDRESS and DBUS_SESSION_BUS_PID for communication with each other. Listing 6 has script of Smart-M3 platform's launch.

It is necessary to store identifier numbers of processes in file for correct platform's shutdown. Listing 7 describes the script of Smart-M3 platform's turning off and disabling the `dbus` session.

## VI. SMART-M3 CONTROL PANEL

For controlling the Smart-M3 platform the web-service "Smart-M3 Control Panel" has been developed. It is possible

**Listing 6** Listing of Smart-M3 platform's launch script

```
#!/opt/bin/bash
eval $(dbus-launch --sh-syntax)
export DBUS_SESSION_BUS_ADDRESS
export DBUS_SESSION_BUS_PID

redsibd &
redsibdPid=$!
sib-tcp &
sibtcpPid=$!

echo $redsibdPid $sibtcpPid
$DBUS_SESSION_BUS_PID >
/tmp/smartM3Pid
```

**Listing 7** Listing of Smart-M3 platform's shutdown script

```
#!/opt/bin/bash
input=‘cat /tmp>smartM3Pid‘
IFS=’ ’ read -a pids <<< "$input"
kill $pids[0] kill $pids[1] kill
$pids[2]
```

to install "Smart-M3 Control Panel" on personal computer and router. Web-service provides the following possibilities:

- Launch, stop and relaunch Smart-M3 platform in case of errors and incorrect working;
- Watch current status of the platform: "running", "stopped", "breakdown" (platform is not working);
- Download Smart-M3 log files;
- View platform's information storage in real time with possibility of filtering and sorting data (view Fig. 1);
- Add, change or remove RDF triples in the information storage;
- Set up launch options of Smart-M3 platform. User can choose type of RDF storage, type of subscription storage, limit SPARQL processing threads; select port of communication between knowledge processes and platform; setup subscription checking interval.

Python 2.7. was chosen as development language for web-service "Smart-M3 Control Panel". The following technologies and software were used:

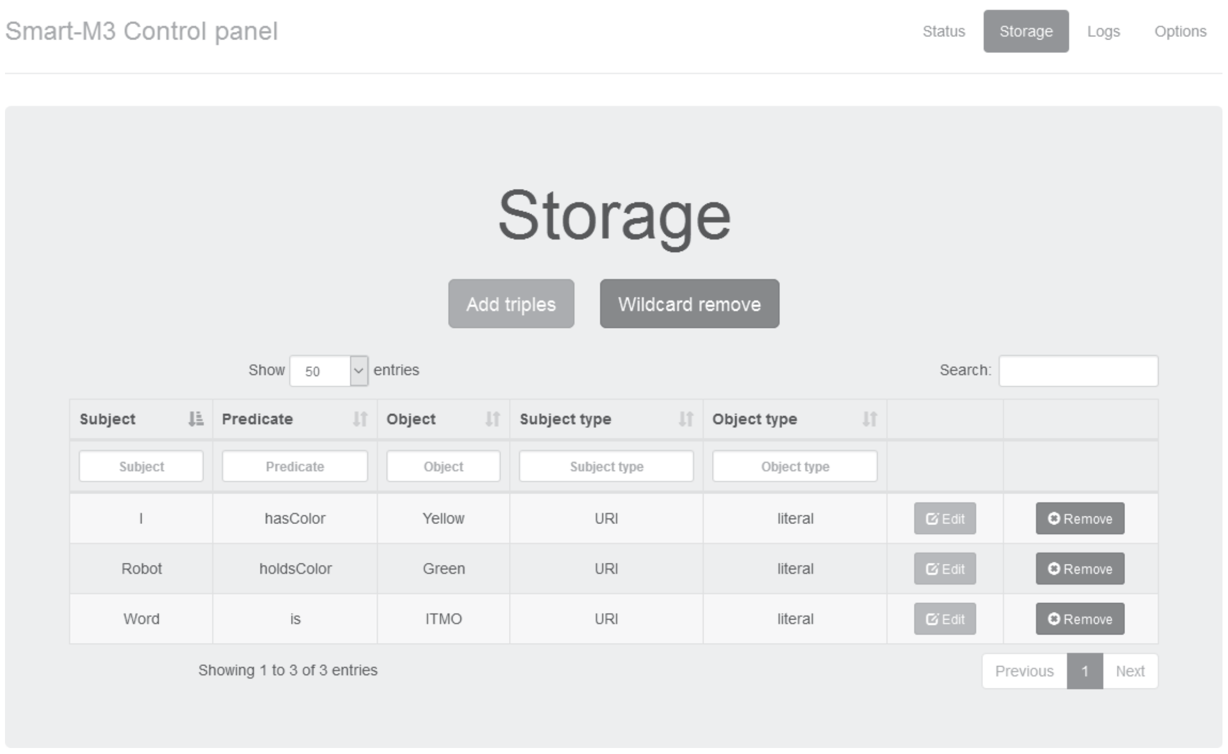
- Flask (http://flask.pocoo.org/) — microframework for creating web applications;
- Socket.IO (http://socket.io/) — JavaScript library, which allows realtime, bi-directional communication between clients and server based on websockets or AJAX messages. The authors have used this library for interaction realization between users and knowledge processes on router. Python package Flask-SocketIO brings server functionality, and package socketIO-client 0.7.2 brings client functionality;
- Gevent (http://www.gevent.org/), greenlet (https://greenlet.readthedocs.io) — these network libraries were used for creating a cooperative multitasking based on lightweight coroutines;
- Gevent-websocket 0.9.5 — plugin-extension for network library gevent, which was used by Flask-SocketIO for weboscket server's functionality;
- jQuery (https://jquery.com/) — JavaScript library was used for DOM manipulating;
- Python-KP (https://github.com/smart-m3/python_kp) — knowledge processor implementation on Python language.

System architecture of "Smart-M3 Control Panel" service is presented on Fig 2. There are two components of service, which is located on a router: FlaskApp and SmartM3Watcher. FlaskApp components have included web-server, which sends static page content as HTTP response to users and Socket.IO-server which gets requests by special protocol, interacts with Smart-M3 platform and sends response back. SmartM3Watcher component monitors changes of current status of Smart-M3 platform and information storage. If either of them were changed, SmartM3Watcher will send information by Socket.IO protocol to FlaskApp component and after that FlaskApp will broadcast this changes to all users.

The source code of "Smart-M3 Control Panel" service is located on [25]. It contains two components: smartM3ControlPanelFlask (FlaskApp in architecture of service) and smartM3ControlPanelWatcher (SmartM3Watcher).

Component smartM3ControlPanelFlask is structured as follows:

- `smartm3_control_panel_flask.py` — script which acts as start point of application;
- `app/files` — this folder contains configuration file with launch options of Smart-M3 platform and information storage representation options. Also, this folder serves as storage of log files;
- `app/modules` — this folder stores modules of the web site. "Main" module is responsible for main page of the site; "Storage" module takes care about interaction between users and information storage of Smart-M3 Platform; "Logs" module works with log-files; "Options" module brings service options such as launch options of Smart-M3 platform and data representation in "Storage" module. Every module has *routes.py* file which contains service routes and *events.py* which includes reactions on Socket.IO-events. Every module can represent itself as named SocketIo-namespace thereby reducing the amount of transmitted information by placing users in a given namespace. Also, this directory has "smartM3KP" module — realization knowledge processor;
- `app/scripts` — this folder has startup and shutdown scripts of Smart-M3 platform for personal computers and router Asus RT-N16;
- `app/static` — static .css and .js files are located in this folder;
- `app/templates` — templates of html files are located in this folder;



| Subject | Predicate | Object | Subject type | Object type | | |
|---|---|---|---|---|---|---|
| I | hasColor | Yellow | URI | literal | Edit | Remove |
| Robot | holdsColor | Green | URI | literal | Edit | Remove |
| Word | is | ITMO | URI | literal | Edit | Remove |

Showing 1 to 3 of 3 entries

Fig. 1. Smart-M3 storage page

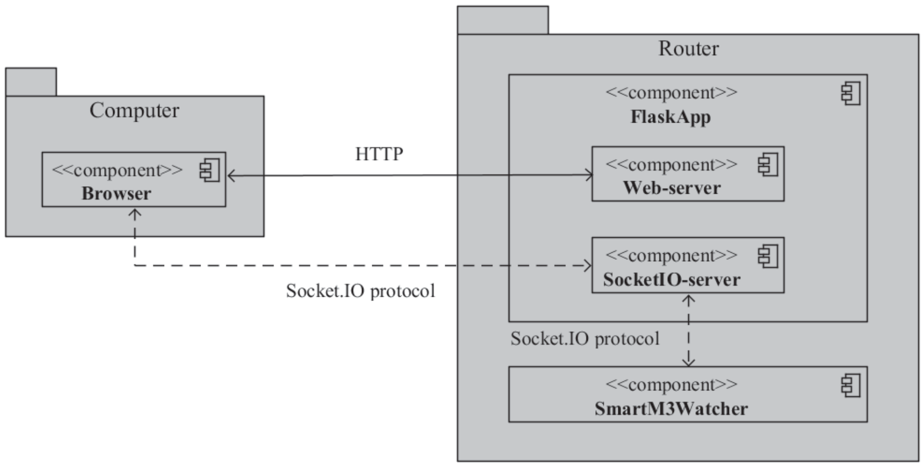


Fig. 2. Smart-M3 Control Panel system architecture

- `app/settings.py` — this directory has inner setting of service including choice of running platform.

Component smartM3ControlPanelWatcher contains next components and folders:

- `smartMWatcher.py` — script which acts as start point of application;
- `app/WatcherThread.py` — script, which contains abstract thread class, which must work in separate process;
- `app/WatcherStatus.py` — script with thread class, based on WatcherThread, which monitors a current status of Smart-M3 platform and gives to the Socket.IO server information about changes;
- `app/WatcherStorage.py` — script with thread class, based on WatcherThread, which monitors information storage and gives to the Socket.IO server information about changes;
- `app/SmartM3KP.py` — script which contains implementation of knowledge processor;

- `app/setting.py` — this directory has inner setting of service including choice of running platform.

## VII. Performance comparison of Smart-M3 platform on router and personal computer

Each robot in the presented scenarios operates with about 5 subscriptions and 50 RDF-triplets. In this case the information storage can contain simultaneously not more than 100 subscriptions and about 1000 RDF-triplets.

Smart-M3 platform's performance measurement has been done after installation of the platform to router Asus RT-N16. The authors have measured the speed of the RDF-triplets' insertion, requests by queries, and they have also computed the number of possible subscriptions. The same test has been made on computer Acer Aspire E5-774G with using Linux Mint as operating system, based on Oracle VirtualBox virtualization. Bdb (Berkley DB) was chosen as storage of RDF-triples. Every dataset was processed after running tests. Firstly, 10th and 90th percentile was calculated, which allows to reduce noise in the dataset. Each value that is less than the tenth percentile and more than the ninety percentile was removed. Secondly, the remaining data were grouped into 50 values and the mean value of the groups have been calculated. Finally, the list of mean values was used as graph source. These actions allow to get a clear view of the platforms' performance.

The first test contains measurement of triples insertion (triple `<'someone_i', 'is_a', 'something_i'>`, where $i$ is a number) and query execution speed (triple `<null, 'is_a', 'null'>`). This cycle was automatically repeated for 10000 times.

Insertion time on computer and router are presented at the Fig. 3 and Fig. 4. The difference between the minimum and maximum value (delta-value) on computer is 0.0005–0.001 seconds, and on router — 0.2 seconds. The fluctuations of the values occur in a small range, which can be explained by technical reasons.

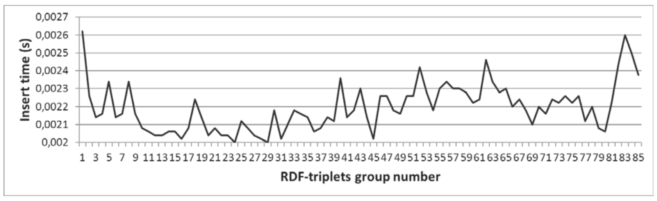


Fig. 3. Insertion time graph — personal computer

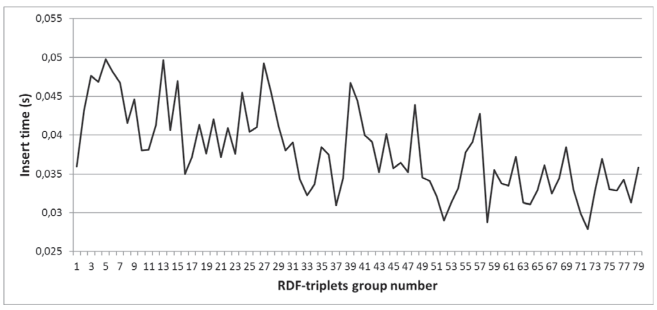


Fig. 4. Insertion time graph — router

Measurement of query execution on both platforms are presented on Fig. 5 (computer) and Fig. 6 (router). Both graphs indicate to a direct correlation between the number of triplets in the information storage and query processing time. Computer value range is 0.02–0.1 seconds, and router — 0.5–3.5.

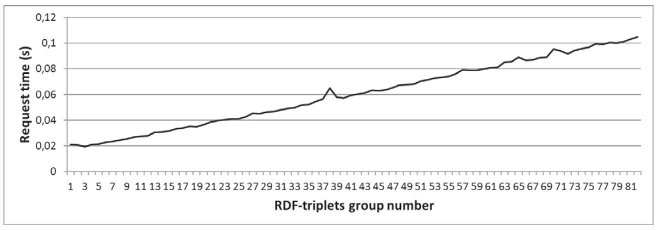


Fig. 5. Query execution time graph — personal computer

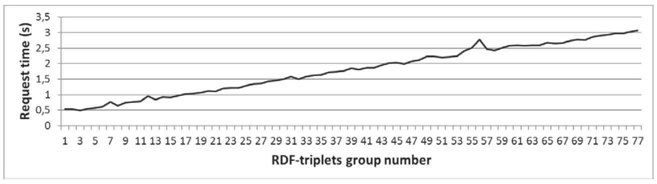


Fig. 6. Query execution time graph — router

The second test contains measurement of maximum number of subscriptions and processing time of subscription of Smart-M3 platform on computer and router (view Fig. 7 and Fig. 8). The authors have created $N$ subscription in a row, where $N$ — maximum number of subscriptions that was found experimentally for both platforms. After that, the authors have inserted a triple, which have triggered subscriptions, and have measured subscription execution time. Maximum number of subscriptions that is allowed for Smart-M3 platform on router is around 650, that is little less than on computer that supports around 1000. Computer's delta value is 0.002 seconds, and router's — 0.1–0.15 second. Subscription graphs on both platforms looks different. Router's graph shows direct correlation between subscription count and subscription execution time and computer's graph shows nothing special. Router's behaviour in this test can be explained by a low amount of RAM and slow bandwidth of USB.

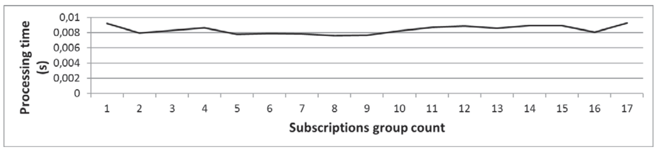


Fig. 7. Subscribe execution time graph — computer

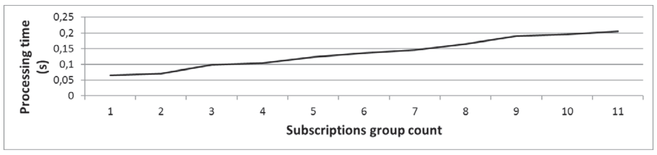


Fig. 8. Subscribe execution time graph — router

In all tests, router performance was less than performance of the computer: insert values differed in 10–15 times, query values differed in 25–30 times and subscriptions values differed in 4–6 times. A large gap in readings in the tests can be explained by the different access speeds to the memory of hard disk and USB flash drive. However, router performance is acceptable for the scenarios mentioned in the paper, that doesn't require real time response of Smart-M3 platform. But sometimes router has a significantly performance slowdown (view Fig. 9). For example, in this period, the insertion of 8000 triplets can reach 10–40 seconds at normal values in 3 seconds. The router performance slowdown can be affected by increased data transfer from router and swap data from RAM to flash memory. However, this period of time does not last very long, so it is doesn't affect hugely of general scenario's flow.

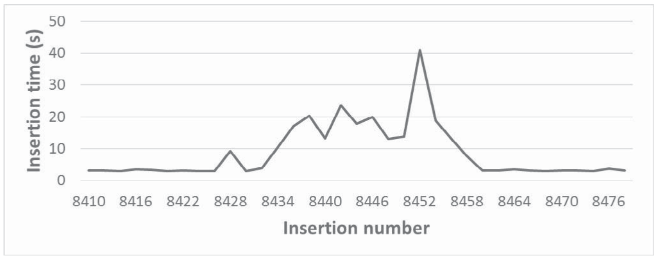


Fig. 9. Example of router slowdown during insertion test

## VIII. Conclusion

The study has solved the problem of the smart space organization on the basis of Smart-M3 platform based of DD-WRT operating system for mobile robots' collaborative work scenario. This paper describes Smart-M3 platform's integration for router Asus RT-N16. The platforms packages compilation runs directly on the router. The alternative way of packages compilation is the cross-compilation on a personal computer with the help of utilities that are provided by the firmware developer, for compilation of applications and libraries under the given processor architecture. Testing has revealed that Smart-M3 platform installed on a router provides sufficient operation speed for scenarios with a few mobile robots. Experiments have shown that the suggested way of smart space organization allows to create scenarios with twenty mobile robots participation. Web-service "Smart-M3 Control Panel" is described in the paper in details. It allow users to control the Smart-M3 platform. The service provides the ability to browse current status of the platform, turn on/off and restart platform depending on its state, view and interact with the content of information storage, download log files and change the platform settings. For future work the authors are planning to add the function to visualize the content of information storage.

## Acknowledgment

The presented results are part of the research carried out within the project funded by grants ## 16-07-00462 of the Russian Foundation for Basic Research. The work has been partially financially supported by Government of Russian Federation, Grant 074-U01.

## References

[1] S. Balandin and H. Waris, "Key Properties in the Development of Smart Spaces", *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments: 5th International Conference, UAHCI 2009*, San Diego, CA, USA, July 19–24, pp. 3–12, 2009.

[2] D. Korzun, S. Marchenkov, A. Vdovenko and O. Petrina, "A Semantic Approach to Designing Information Services for Smart Museums", *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, pp. 15–34, 2016.

[3] D. Korzun, A. Borodin, I. Paramonov, A. Vasilyev and S. Balandin, "Smart Spaces Enabled Mobile Healthcare Services in Internet of Things Environments", *International Journal of Embedded and Real-Time Communication Systems*, vol. 6, iss. 1, pp. 1–27, January, 2015.

[4] A. Kashevnik, N. Teslya, B. Padun, K. Kipriyanov and V. Arckhipov, "Industrial Cyber-Physical System for Lenses Assembly: Configuration Workstation Scenario", *Proceedings of the 17th Conference of the Open Innovations Association FRUCT*, Yaroslavl, Russia, 20–24 April, pp. 62–67, 2015.

[5] A. Smirnov, A. Kashevnik, N. Shilov, H. Paloheimo, H. Waris and S. Balandin, "Increasing Broker Performance in Smart-M3 Based Ridesharing System" *Smart Spaces and Next Generation Wired/Wireless Networking: 11th International Conference, NEW2AN 2011, and 4th Conference on Smart Spaces, ruSMART 2011*, St. Petersburg, Russia, August 22–25, 2011.

[6] A. Smirnov, A. Kashevnik, N. Teslya, S. Mikhailov and A. Shabaev, "Smart-M3-Based Robots Self-Organization in Pick-and-Place System", *Proceedings of the 17th Conference of the Open Innovations Association FRUCT*, Yaroslavl, Russia, 20–24 April, pp. 210–215, 2015.

[7] A. Smirnov, A. Kashevnik, S. Mikhailov, M. Mironov and M. Petrov, "Ontology-based collaboration in multi-robot system: Approach and case study", *11th Systems of Systems Engineering Conference, SoSE 2016*, Kongsberg, Norway, 2016.

[8] A. Smirnov, A. Kashevnik , S. Mikhailov, M. Mironov and O. Baranuic , "Multi-level Robots Self-organization in Smart Space: Approach and Case Study", *Internet of Things, Smart Spaces, and Next Generation Networks and Systems: 15th International Conference*, Springer International Publishing, pp. 68–79, 2015.

[9] A. Smirnov, A. Kashevnik, S. Mikhailov , M. Mironov and M. Petrov, "Smart M3-Based Robot Interaction Scenario for Coalition Work", *Interactive Collaborative Robotics: First International Conference, ICR 2016*, St.Petersburg, Russia, pp. 199–207, 2016.

[10] T. Kubitza, "Apps for Environments: Demonstrating Pluggable Apps for Multi-Device IoT-Setups", *Proceedings of the 6th International Conference on the Internet of Things, IoT'16*, Stuttgart, Germany, pp. 185–186, 2016.

[11] T. Kubitza and A. Schmidt, "Towards a Toolkit for the Rapid Creation and Programming of Smart Environments", *End-User Development: 5th International Symposium, IS-EUD*, Madrid, Spain, May 26–29, 2015.

[12] C. Buckl, M. Geisinger, D. Gulati, F. J. Ruiz-Bertol and A. Knoll, "CHROMOSOME: A Run-Time Environment for Plug & Play-Capable Embedded Real-Time Systems", *SIGBED Rev.*, vol. 11, no. 3, pp. 36–39, 2014.

[13] S. Sommer, M. Geisinger, C. Buckl, G. Bauer and A. Knoll, "Reconfigurable Industrial Process Monitoring using the CHROMOSOME Middleware", *SIGBED Rev.*, vol. 10, no. 4, pp. 61–64, 2013.

[14] X. Wang, J. S. Dong, C. Chin, S. Hettiarachchi and D. Zhang, "Semantic Space: An Infrastructure for Smart Spaces", *IEEE Pervasive Computing*, vol. 3, no. 3, pp. 32–39, 2004.

[15] M. Palviainen, J. Kuusijärvi and E. Ovaska, "A semi-automatic end-user programming approach for smart space application development", *Pervasive and Mobile Computing*, vol. 12, pp. 17–36, 2014.

[16] M. Palviainen, J. Kuusijärvi and E. Ovaska, "Framework for End-User Programming of Cross-Smart Space Applications", *Sensors*, vol. 12, no. 11, pp. 14442–14466, 2012.

[17] V. Tzeremes and H. Gomaa, "A Software Product Line Approach for End User Development of Smart Spaces", *Proceedings of the Fifth International Workshop on Product LinE Approaches in Software Engineering, PLEASE '15*, Florence, Italy, pp. 23–26, 2015.

[18] J.P Sousa, V. Tzeremes and A. El-Masri, "Space-aware TeC: End-user development of safety and control systems for smart spaces", *IEEE*

*International Conference of Systems, Man and Cybernetics*, Istanbul, Turkey, pp. 2914–2921, 2010.

[19] K. Selvarajah, R. Zhao and N. Speirs, ”Building Smart Space Applications with PErvasive Computing in Embedded Systems (PECES) Middleware”, *GSTF Journal on Computing (JoC)*, vol. 1, no. 4, 2012.

[20] D. Garlan, D. Siewiorek, A. Smailagic and P. Steenkiste, ”Project Aura: Towards Distraction-Free Pervasive Computing”, *IEEE Pervasive Computing*, vol. 1, no. 2, pp. 22–31, 2002.

[21] C. Becker, G. Schiele, H. Gubbels and K. Rothermel, ”BASE — A Micro-broker-based Middleware For Pervasive Computing”, *Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communication*, Fort Worth, USA, pp. 443–451, 2003.

[22] Smart-M3. Web: https://sourceforge.net/projects/smartm3/files/Smart-M3RedSIB_0.9.2

[23] J. Honkola, H. Laine, R. Brown and O. Tyrkko, ”Smart-M3 Information Sharing Platform”, *Computers and Communications (ISCC), 2010 IEEE Symposium*, pp. 1041-1046, 2010.

[24] F. Viola, A. D’Elia, D. Korzun, I. Galov, A. Kashevnik and S. Balandin, ”The M3 Architecture for Smart Spaces: Overview of Semantic Information Broker Implementations”. *Proceedings of the 19th Conference Open Innovations Association FRUCT*, Jyväskylä, Finland, November 7–11, pp. 264–272, 2016.

[25] Smart-M3 Control Panel at SourceForge.net. Web: https://sourceforge.net/projects/smartm3/files/SmartM3ControlPanel