Smart-M3 CuteSIB Demo for a Wireless Router with OpenWrt-Based Firmware

Sergey A. Marchenkov, Dmitry E. Baganov, Dmitry G. Korzun
Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
{marchenk, baganov, dkorzun}@cs.karelia.ru

Abstract—**The use of wireless technologies is now inevitable in smart spaces development for Internet of Things. A smart space is created by deploying a Semantic Information Broker (SIB) on a host device. This demo studies the opportunities of a wireless router as a SIB host device. CuteSIB is one of SIB implementations of the Smart-M3 platform and focus is on Qt-based devices. A cross-compiling method is used to build CuteSIB installation packages for the routers operating with such OpenWrt-Based firmware as DD-WRT. For the case study, we consider the SmartRoom system—a Smart-M3-based application that creates a virtual shared workspace in a multimedia equipped room to support collaborative activity participants.**

I. Introduction

Smart spaces are gaining relevance as promising deployment environments for novel classes of applications stemming from the dynamic discovery and interaction between smart objects and resources available in their physical localities. The Internet of Things (IoT) supports ubiquitous connectivity property for smart spaces [1], [2]. We consider smart spaces deployed in localized resource-restricted IoT environments [3]. Such a smart space is typically associated with a physical spatial-restricted place equipped with a variety of devices.

Smart-M3 platform provides software implementations for such a central element of an smart space as Semantic Information Broker (SIB) [4]. In this demo work, we study the opportunities of a wireless router for being a SIB host device, in contrast to more common case of server hosting. Such a class of embedded devices can be evolved towards packaged product composed of Smart-M3 software components (SIB and some software agents) to advance the smart spaces deployment in the wireless and resource-restricted settings of localized IoT environments [5].

CuteSIB [6] is an implementation variant of Smart-M3 SIB with the focus on a wide spectrum of Qt-based IoT devices. Our previous work confirms that a single-board computer (Raspberry Pi) capacity is enough to host both CuteSIB and some search service operating with the public DBpedia service from the Internet [7]. In this demo work, we consider a wireless router to host CuteSIB. A wireless router is a low-capacity device. Consequent compilation of Smart-M3 software components is a slow process [8]. We introduce more effective way for CuteSIB installation. The target class of wireless routers is restricted with OpenWrt-Based firmware such as DD-WRT using a cross-compiling method. Our solution is evaluated based on a case study of the SmartRoom system [9].

The rest of the paper is organized as follows. Section II describes our way for CuteSIB installation on wireless router using a cross-compiling method. Section III discusses the opportunities of a wireless router based on a case study the SmartRoom system. Section IV summarizes our recent results.

II. CuteSIB for DD-WRT

A. Cross-compiling method

Cross-compiling of CuteSIB is performed for the ASUS RT-N66U wireless router, which is the typical representative of an embedded devices. The hardware specification of the router is summarized in Table I. The router is well equipped to run such OpenWrt-Based Firmware as DD-WRT. Mounting a USB storage device via USB 2.0 port can solve the problem of lack of memory providing nearly-unlimited space to install many applications and libraries including the CuteSIB dependencies and the application itself.

DD-WRT is a Linux based alternative OpenSource firmware suitable for a great variety of WLAN routers and embedded systems [10]. There is a project's wiki page about the ASUS RT-N66U with the firmware installation instructions [11]. It is required to choose the latest recommended K3.X Broadcom build (at present, 26138 is a recommended build) as USB support does not work in K2.6 builds.

The floating-point unit (FPU) emulation has been disabled since revision 20047 of Kernel 3.x build [12], so Optware packages from NSLU2-Linux [13] is not fully supported officially. Optware packages is currently in development for Kernel 3.x builds [14]. In order to solve FRU emulation problems OpenWrt packages for the brcm47xx platform with a MIPSel architecture to be used [15]. Furthermore, OpenWrt packages pre-compiled for the appropriate platform can be used for different Openwrt-based firmwares providing portability.

The DD-WRT team works now to merge the code base of DD-WRT with the OpenWRT firmware. Consequently OpenWRT's package management system (ipkg) is also applicable

TABLE I. Asus RT-N66U wireless router specification

Item	Description
Platform	Broadcom BCM4706
Target	brcm47xx
Instruction Set	MIPS32 74K series
Bootloader	CFE
CPU	600 MHz
Flash memory	32 Mb
RAM	256 Mb
WLAN 2.4GHz	b/g/n
WLAN 5.0GHz	a/n
USB ports	2 x 2.0

for DD-WRT. Since ipkg is a command line program, SSH is needed into the router to run this utility. The Journaling Flash File System (JFFS) should be enabled. The following steps to enable JFFS through the router web page.

1) On the router web page click on *Administration*.
2) Scroll down to *JFFS2 Support* section.
3) Click *Enable JFFS2*, Click *Save*.
4) Wait couple seconds, then click *Apply*.
5) Wait again. Go back to the *Enable JFFS2* section, and enable *Clean JFFS2*.
6) Do not click *Save*. Click *Apply* instead.
7) Wait till the process is completed, then disable *Clean JFFS* again and click *Save*.
8) It may be wise to reboot the router, just to make sure.

JFFS is mounted on the USB drive, see DD-WRT project's wiki page [16]. After the preparation and partition of the drive, USB support needs to be enabled on the router. Then the following console commands are executed on the router to mount the USB drive (data partition) to JFFS:

```
mount /dev/sda3 /mnt
mkdir /mnt/jffs
mount /mnt/jffs/ /jffs
```

In order to use ipkg, it is needed to create the folder `/jffs/tmp/ipkg`. To assure that the install OpenWrt packages will run stable one must provide compatible OpenWrt uClibc library. It is recommended to install the latest release of uClibc library (version 0.9.33.2) manually.

The cross-compiling process consist of the following three steps: (1) cross-compiling of Qt, (2) cross-compiling of dependent libraries for CuteSIB, (3) cross-compiling of CuteSIB. Cross-compiling of these program components on x86 systems for a routers with OpenWrt-Based Firmware with building the ipkg packages can be performed using the OpenWrt build system, which is the part of the OpenWrt Software Development Kit providing a set of Makefiles, utilities, and patches that allows developers to easily generate both a cross-compiling toolchain and a root file system for a wireless routers.

A ipkg package is built using the source code located in the package subdirectory under the SDK directory. In order to tell the OpenWrt build system how to build a package from the source code, it is necessary to create a special BuildPackage Makefile in the package directory with the source code. The BuildPackage Makefile defines the meta information of the package, where to download the package, how to configure, how to compile, where to install the compiled libraries and binaries, etc. The following command is executed in the SDK directory to build a package *example*:

```
make package/example/compile
```

B. Required Qt modules

The latest release of Qt for embedded Linux can be downloaded from the official site of the Qt project [17]. Qt Core and Qt Network are necessary and sufficient Qt modules for CuteSIB operation. Each module must be built as a separate ipkg package using the OpenWrt build system. It is necessary to create the Qt package directory as described above with all the required files. In order to cross-compile Qt modules for a router with an MIPSel architecture the following way of running the configure script at the BuildPackage created Makefile is used:

```
./configure -arch mipsel -no-c++11
-xplatform linux-openwrt-g++ -no-strip
-make libs {other options}
```

Other options include disabling unnecessary parts of Qt and setting the install paths of the binaries and libraries for OpenWrt SDK. The qmake tool for Openwrt SDK requires a specific qmake.conf and qmake.mk files located in Qt package directory. This files describe the platform configuration variables, the qmake configure and compile options.

TABLE II. THE CUTESIB DEPENDENT LIBRARIES

Name	Description	Download URL
libxml2 (2.9.4)	Software C library for parsing XML documents	http://xmlsoft.org/downloads.html
expat (2.2.0)	Stream-oriented XML C parser library	https://sourceforge.net/projects/expat
libiconv (1.14)	Character encoding conversion library	https://gnu.org/software/libiconv
python (2.7.13)	Standard library that is distributed with Python	https://www.python.org/downloads
raptor2 (2.0.15)	Library that provides parsers and serializers that generate RDF triples	http://librdf.org/raptor
rasqal (0.9.33)	Library that supports RDF query languages	http://librdf.org/rasqal
redland (1.0.17)	Library that provides the RDF API and triple stores	http://librdf.org

C. Dependencies

CuteSIB requires special libraries which are needed to cross-compile it. The majority of dependent libraries are found in the respective packages in the OpenWrt download area for the appropriate release and platform, in this case, it is Attitude Adjustment release and brcm47xx platform respectively. There are some of the packages such as *libdb*, *unixodbc*, *zlib*, etc [18]. However, other dependent libraries are to be built from source code with the OpenWrt build system. A list of these libraries is summarized in Table II.

Each of these libraries relies on Autotools build system which may lead to a number of problems in a cross-compiling setting. OpenWrt SDK defines the following rules, that must be added to the top of the library BuildPackage Makefile, to help work around this problem:

```
PKG_FIXUP:=autoreconf
PKG_FIXUP:=patch-libtool
PKG_FIXUP:=gettext-version
```

The libraries should be compiled as a shared libraries. Code that is built into shared libraries should normally be position-independent code, so that the shared library can readily be loaded at any address in router memory. The OpenWrt build system can be instructed to generate position-independent code using the *-fPIC* option in BuildPackage Makefile.

TABLE III. THE CUTESIB SOFTWARE COMPONENTS

Component	Description
AccessPointInterface	It provides an interface for access points that is used by CuteSIB. This interface is a factory to create access point instances.
CuteSIB	The SIB core contains some helper submodules and implements commands processing.
DiscoveryAccessPoint	This access point is used to discovery CuteSIB with broadcast requests. UDP Discovery Protocol Handler prepares information about CuteSIB and sends it back for request.
HttpAccessPoint	This access point handles HTTP SPARQL request (GET/POST). With HTTP Access Point CuteSIB can be accessed as a SPARQL-endpoint (e.g. DBPedia) with query request support.
TcpAccessPoint	This access point provides TCP access and is used as main access point to communicate with CuteSIB.
TcpAsyncAccessPoint	This access point provides TCP asynchronous access and is used as additional access point to communicate with CuteSIB.

D. Implementation

The CuteSIB implementation is based on the Qt framework. The software components included in this Qt software project are listed in Table III. CuteSIB have plug-ins based architecture in order to achieve higher extensibility due to the modular approach. All the access points are used as plug-ins. The architecture allows inclusion/exclusion of certain plug-ins in compilation phase or in runtime.

For cross-compiling of the CuteSIB software components Cmake make system is used. CMake is a flexible alternative to qmake for automating the generation of build configurations. It controls the software compilation process by using simple configuration files, called *CMakeLists.txt* files. The OpenWRT build system provides abstraction for the suite of CMake tools using *include/cmake.mk* file in the SDK root directory.

The CuteSIB project should have the BuildPackage Makefile with cmake call in *Build/Configure* define section. Furthermore, each CuteSIB component is to located in a separate subcategory with the CMakeLists.txt file. The CuteSIB core is built as an executable program using *add_executable* cmake command in *CMakeLists.txt*. The access point are built as dynamic shared libraries using *add_library* cmake command with *SHARED* setting property.

After cross-compiling of the CuteSIB and all of its dependencies, the prebuilt packages are copied onto the router by using SSH. In order to install the packages on */jffs* partition, the following console command is used:

```
ipkg -d root -force-depends install <ipk>
```

To configure access points and RDF triplestore, the CuteSIB config file is used. The *Triplestore* section of the config file contains contains the settings for type of triplestore, name of database, and path where storage is located. Berkeley DB is an efficient and fast database engine, which is a good choice for the type of triplestore on embedded devices. In case of a router, the access points are loaded into the CuteSIB executable program as dynamic shared libraries. For instance, the following section of the config file sets the parameters for dynamic TCP access point:

```
[AccessPoint:tcp_ap10]
name=libTcpAccessPoint
path=/jffs/usr/lib/
parameters=port:10010
```

This way of CuteSIB installation is applied to a wide class of wireless routers with OpenWrt-Based firmware. The prebuilt packages can be used directly for a router with the same CPU architecture without additional preparation steps.

III. APPLICATION DEMO STUDY

The demo uses the SmartRoom system [9], which is a Smart-M3 service-oriented application for assisting such collaboration activity as conference or meeting. The system creates a shared SmartRoom space where services are constructed by software agents running on embedded, mobile, and remote devices. The agents are also called knowledge processors (KPs) to specify the Smart-M3 focus on information sharing. The SmartRoom space is deployed within the localized IoT environment of a multimedia equipped room. The environment is equipped with variety of devices connected to the local area network created by the wireless router.

In the SmartRoom case study, we deploy CuteSIB on the wireless router to create the SmartRoom space for the core services and their clients, as shown in Fig. 1. The core SmartRoom services provide the basic functionality associated with automated holding of collaborative activities. Conference-service dynamically maintains the activity program (i.e., conference section or agenda of talks). The result is visualized by Agenda-service. Presentation-service displays multimedia presentations and videos related to talks of the participants. Content-service keeps multimedia content which is used in activity and provides the shared content in the SmartRoom space. SmartRoom participants use their personal mobile devices to access services via SmartRoom clients running on the smartphone or tablet. Admin-client is used by the chairman and

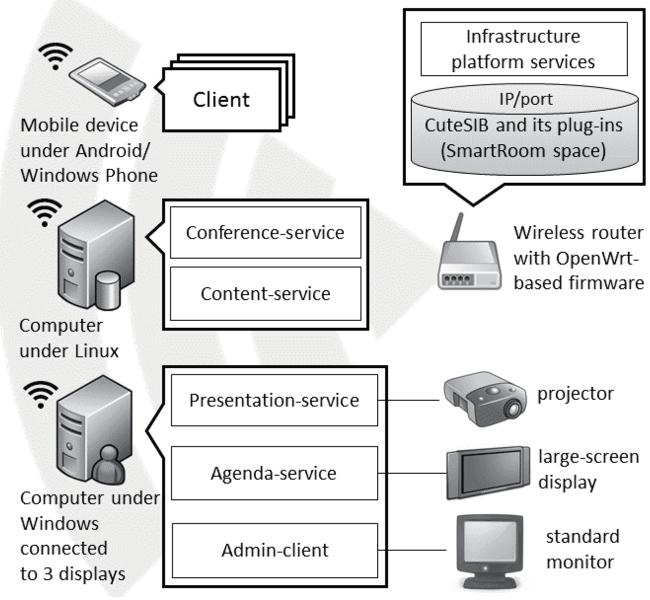
Infrastructure platform services
IP/port
CuteSIB and its plug-ins (SmartRoom space)
Client
Mobile device under Android/ Windows Phone
Wireless router with OpenWrt-based firmware
Conference-service
Content-service
Computer under Linux
Presentation-service
Agenda-service
Admin-client
projector
large-screen display
standard monitor
Computer under Windows connected to 3 displays

Fig. 1. SmartRoom deployment with a wireless router to host the SIB

provides activity control panel (e.g., the chairman controls the slide show and other visualization).

In the basic case, the following devices are needed: several local computers for data processing and service construction control, large-format displays for visual service delivery, personal mobile devices of participants, and a wireless router for the local network. The router is both CuteSIB host device and wireless access point for other participating devices. Conference-service is run along with Content-service on one local computer. This computer acts as a server performing data processing and providing web access to multimedia content. Presentation-service, Agenda-service, and Admin-client run on another local computer, which also serves the three displays.

The following equipment, application software components, and utilities are used.

1) Router ASUS RT-N66U: Wireless router (see Table I).
2) Two local desktops (CPU Intel Core i5, 4 GB RAM, 1 TB FlashRam, 100 MBit WAN): Hosting for the core SmartRoom services.
3) CuteSIB ver. 0.5.0 (sourceforge.net/projects/smart-m3/): Implementation of Smart-M3 SIB.
4) Core SmartRoom services (sourceforge.net/projects/smartroom/files/services/).
5) Mobile SmartRoom clients for Windows Phone or Android (sourceforge.net/projects/smartroom/files/clients/).
6) *pidstat*: Utility to record workload data on the SIB host machine.
7) *SibTest*: Utility for testing the SIB.

We evaluate the performance of the SmartRoom system. The simulation testbed is shown in Fig. 1). The workload is generated by $1 \leq n \leq 30$ virtual clients, each has update rate $1 \leq \lambda \leq 200$ operations per second.

On the computer with *Conference-service* and *Content-service* a SSH connection with the router is established. CuteSIB is launched on DD-WRT router. Utility *pidstat* is

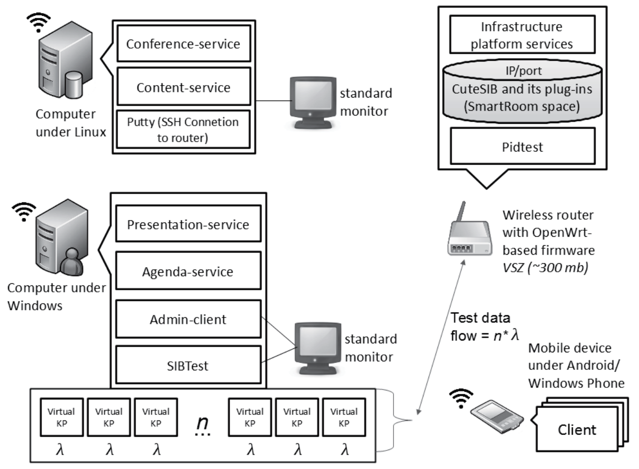

Fig. 2. Testbed for experiments

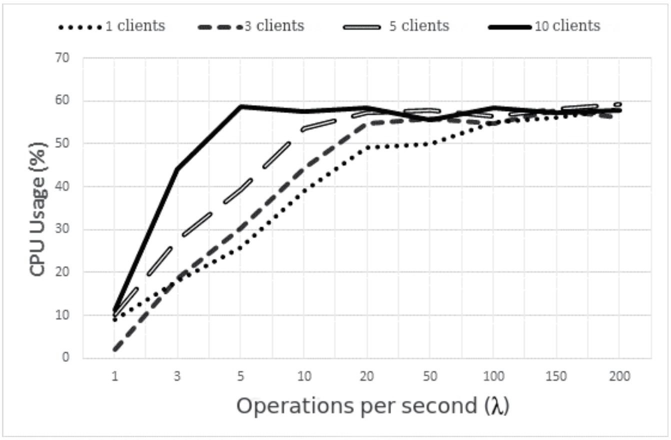

Fig. 3. CPU Usage with increasing update rate λ of a virtual client

started on the router and measures the CPU Usage (%) and Virtual Set Size (VSZ) memory (Mb).

Using utility *SIBTest*, n virtual clients are started to generate background read&write workload for the SIB with the sum rate is $n\lambda$. The utility allows setting n, λ, and the number of subscriptions.

One dedicated smartphone (Windows Phone or Android) runs a real SmartRoom client to accesses the core SmartRoom services. This activity goes in parallel with the background workload generated by n virtual clients. In every demo experiment, the SmartRoom client performs random user-driven slide changing during 30 s.

The measured CPU Usage is shown in Fig. 3. With increasing the number of clients, the router CPU quickly reaches the capacity limit of 60%. The rest 40% are used by other processes. In particular, the *dropbear* process is responsible for the SSH connection $(30 \dots 35\%)$ and the DD-WRT system processes $(5 \dots 10\%)$. This limit shows when the increasing workload saturates the CPU capacity.

When the router has reached the 60% limit of CPU usage, the SIB is overloaded with the stress workload. The number of incoming requests becomes increasing faster than the SIB can serve the requests, i.e., the SIB internal queue grows monotonically till the exhaustion of the available memory. Our experiments show that the CuteSIB process consumes the available VSZ memory (300 Mb) in $8 \dots 10$ minutes. The standard RAM in router ASUS RT-N66U is 256 Mb. The additional memory can be used from a flash card, similarly the swap file technique in UNIX/Windows systems.

The behavior of CPU Usage in dependence on the number of virtual clients is shown in Fig. 4. The router quickly reaches the 60% limit, the available capacity is saturated, and the SIB becomes overloaded. Nevertheless, the SIB still serves the requests until the VSZ memory exhaustion. In fact, this property confirms the high dependability level of CuteSIB, compared with other Smart-M3 SIB implementations.

The processing time observable on the user side is shown in Fig. 5. The real SmartRoom client is changing slides, and the operation delay (i.e., the response time—the time to complete a slide change) is measured (in seconds). Starting

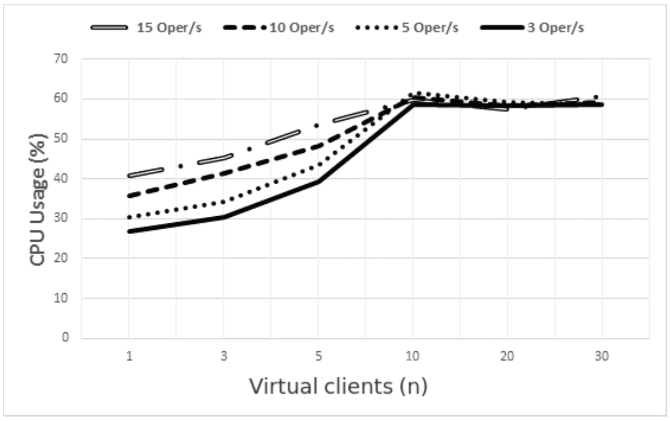

Fig. 4. CPU Usage with increasing number of virtual clients n

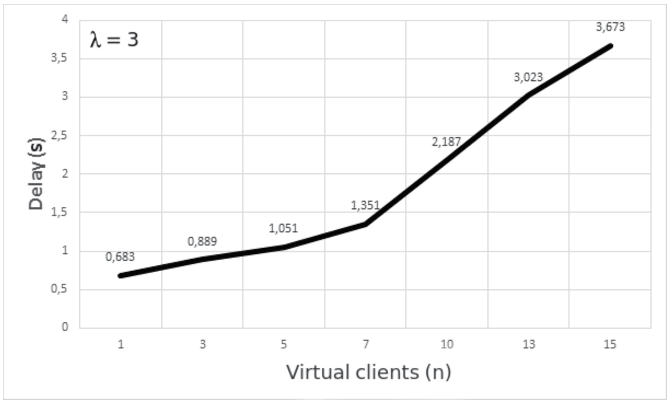

Fig. 5. Delay in changing slides from mobile SmartRoom client

from $n \approx 8 \dots 10$ the delay becomes growing fast due to the upcoming SIB overload. For $15 < n \leq 30$ (not shown in Fig. 5) the delay becomes inappropriate for practical use. Nevertheless, the case study shows that the router capacity is enough for small collaborative activity with a dozen of participants at most.

IV. Conclusion

This demo experimentally examined the opportunities of a wireless router for being a Smart-M3 SIB host device. First, we presented a technical solution to CuteSIB installation on a wireless router with OpenWrt-Based firmware. Similar solutions can be applied to install many other variants of Qt-based middleware. Second, we experimented with a particular case study of the SmartRoom system. Our study indicates that router capacity is satisfactory for deployment of small SmartRoom spaces.

Acknowledgment

This research is supported by the Ministry of Education and Science of Russia within project # 2.5124.2017 of the basic part of state research assignment for 2017–2019.

References

[1] S. Balandin and H. Waris, "Key properties in the development of smart spaces," in *Proc. 5th Int'l Conf. Universal Access in Human-Computer Interaction (UAHCI '09). Part II: Intelligent and Ubiquitous Interaction Environments, LNCS 5615*, C. Stephanidis, Ed. Springer-Verlag, Jul. 2009, pp. 3–12.

[2] L. Roffia, F. Morandi, J. Kiljander, A. D. Elia, F. Vergari, F. Viola, L. Bononi, and T. Cinotti, "A semantic publish-subscribe architecture for the Internet of Things," *IEEE Internet of Things Journal*, vol. PP, no. 99, 2016.

[3] D. Korzun, A. Kashevnik, S. Balandin, and A. Smirnov, "The Smart-M3 platform: Experience of smart space application development for Internet of Things," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems. Proc. 15th Int'l Conf. Next Generation Wired/Wireless Networking and 8th Conf. on Internet of Things and Smart Spaces (NEW2AN/ruSMART 2015), LNCS 9247*, S. Balandin, S. Andreev, and Y. Koucheryavy, Eds. Springer, Aug. 2015, pp. 56–67.

[4] 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," in *Proc. of the 19th Conference of Open Innovations Association FRUCT*, S. Balandin and T. Tyutina, Eds. FRUCT Oy, Helsinki, Finland, Nov. 2016, pp. 264–272.

[5] D. G. Korzun, I. V. Galov, and A. A. Lomov, "Smart space deployment in wireless and mobile settings of the Internet of Things," in *Proc. IEEE 3rd International Symposium on Wireless Systems IDAACS:SWS*, 2016, pp. 86–91.

[6] I. Galov, A. Lomov, and D. Korzun, "Design of semantic information broker for localized computing environments in the Internet of Things," in *Proc. 17th Conf. of Open Innovations Association FRUCT*. ITMO University, IEEE, Apr. 2015, pp. 36–43.

[7] S. Marchenkov, A. Borodulin, D. Baranov, and D. Korzun, "CuteSIB demo for Raspberry Pi," in *Proc. 18th Conf. Open Innovations Framework Program FRUCT*. ITMO Univeristy, Apr. 2016, pp. 545–545.

[8] S. Mikhailov, "Smart-M3 platform installation to DD-WRT-based Wi-Fi router," in *Proc. 18th Conf. of Open Innovations Association FRUCT*, S. Balandin, T. Tyutina, and A. Levina, Eds. ITMO University, Apr. 2016, pp. 564–566.

[9] D. Korzun, I. Galov, A. Kashevnik, and S. Balandin, "Virtual shared workspace for smart spaces and M3-based case study," in *Proc. 15th Conf. of Open Innovations Association FRUCT*, S. Balandin and U. Trifonova, Eds. ITMO University, Apr. 2014, pp. 60–68.

[10] About DD-WRT. [Online]. Available: https://www.dd-wrt.com/site/content/about

[11] (2016, Feb.) Asus RT-N66U. [Online]. Available: http://www.dd-wrt.com/wiki/index.php/Asus{_}RT-N66U

[12] (2012, Sep.) Trac: Changeset 20047. [Online]. Available: http://svn.dd-wrt.com/changeset/20047

[13] (2009, Oct.) NSLU2-Linux. [Online]. Available: http://www.nslu2-linux.org/

[14] (2015, Dec.) OTRW2 (Optware The Right Way Take 2). [Online]. Available: http://www.dd-wrt.com/wiki/index.php/OTRW2_(Optware_the_right_way_Take_2)

[15] (2013, Jun.) Trac: Changeset 21807. [Online]. Available: http://svn.dd-wrt.com/changeset/21807

[16] (2015, Feb.) USB Storage. [Online]. Available: http://www.dd-wrt.com/wiki/index.php/USB_storage

[17] Qt Downloads. [Online]. Available: https://download.qt.io

[18] (2014, Jul.) OpenWrt: Wireless Freedom. [Online]. Available: https://downloads.openwrt.org/attitude_adjustment/12.09/brcm47xx