УДК 006.72 **DOI** 10.15622/sp.52.9

С.А. Михайлов, А.М. Кашевник

ОРГАНИЗАЦИЯ ИНТЕЛЛЕКТУАЛЬНЫХ ПРОСТРАНСТВ НА ОСНОВЕ ПЛАТФОРМЫ SMART-M3 С ИСПОЛЬЗОВАНИЕМ УСТРОЙСТВ НА БАЗЕ ОПЕРАЦИОННОЙ СИСТЕМЫ DD-WRT

Михайлов С.А., Кашевник А.М. **Организация интеллектуальных пространств на основе платформы Smart-M3 с использованием устройств на базе операционной системы DD-WRT.**

Аннотация. Интеллектуальное пространство представляет собой сервис-ориентированную инфраструктуру для возможности обеспечения общего доступа к информации различными устройствами. Статья описывает процесс организации интеллектуальных пространств путем интеграции платформы Smart-M3 и устройства функционирующего на базе операционной системы DD-WRT. Smart-M3 представляет собой платформу с открытыми исходными кодами, реализующую концепцию интеллектуального пространства. В качестве устройства для интеграции был выбран Wi-Fi маршрутизатор, что позволяет использовать его одновременно и для организации интеллектуального пространства, и для обеспечения беспроводной связи между устройствами. Использование Wi-Fi маршрутизатора упрощает развертывание сценариев с небольшим количеством участников. Статья подробно описывает процесс компиляции и установки платформы Smart-M3 для операционной системы DD-WRT. Компиляция пакетов с исходным кодом платформы происходит на маршрутизаторе. Измерение быстродействия стандартных операций платформы Smart-M3 на маршрутизаторе показало, что интеллектуальное пространство организованное таким образом может успешно использоваться для сценариев с небольшим количеством участников. Авторами был разработан веб-сервис «Smart-M3 Control Panel», который позволяет пользователям управлять платформой Smart-M3 посредствам графического веб интерфейса. С использованием этого сервиса пользователь получает возможность просматривать текущий статус платформы; запускать, останавливать или перезапускать платформу; просматривать содержимое информационного хранилища; загружать лог-файлы и менять параметры запуска платформы Smart-M3. Интерактивное взаимодействие с пользователем было реализовано при помощи протокола SocketIO.

Ключевые слова: интеллектуальное пространство, Smart-M3, маршрутизатор, DD-WRT, Smart-M3 Control Panel.

1. Введение. Интеллектуальные пространства представляют собой совокупность различных устройств, для которых имеется возможность организации совместного общего доступа к их информации и знаниям. Для реализации такой концепции в работе используется платформа Smart-M3, являющейся свободно распространяемым программным продуктом.

Авторами был разработан ряд сценариев коллаборативной работы мобильных роботов [1, 2, 3], построенных на базе робототехнического комплекта Lego Mindstorms EV3 с использованием платформы Smart-M3, установленной на персональный компьютер под управлением операционной системы семейства Unix. Подразумевается существование возможности обмена

данными между мобильными роботами (например, с помощью Wi-Fi сети). Для повышения удобства использования оборудования, необходимого для работы сценариев, авторы исследовали возможность и предложили последовательность шагов для интеграции платформы Smart-M3 и маршрутизатора, предоставляющего мобильным роботам доступ к Wi-Fi сети. Такая интеграция позволяет сделать робототехнические сценарии мобильнее путем исключения использования персонального компьютера, что уменьшает количество устройств, необходимое для развертывания сценариев.

Измерение производительности платформы Smart-M3, установленной на маршрутизаторе, показало, что быстродействия платформы достаточно для организации сценариев с небольшим количеством мобильных роботов. Такой способ организации интеллектуальных пространств позволяет создавать сценарии с участием до 20 мобильных роботов.

2. Анализ предметной области. Платформа meSchup IoT [4, 5] предлагает способ взаимодействия устройств, находящихся в интеллектуальном пространстве, через приложения, которые работают на центральном сервере (устройстве meSchHub) и управляют устройствами на основе полученных от них данных. На центральном сервере несколько приложений могут работать параллельно и завершать работу в любой момент времени. Данная платформа позволяет объединять устройства от различных производителей (Android, Arduino, микроконтроллеры .NET Gadgeteer и nRF51822, Raspberry Pi и Intel Edison, компьютеры на базе операционных систем Windows и Linux) для решения задач. Платформа meSchup IoT состоит из клиентского и серверного программного обеспечения. Клиентское программное обеспечение необходимо установить на устройства для передачи показаний их датчиков в приложения на сервере и приема управленческих команд со стороны приложений. Серверное программное обеспечение гарантирует работу приложений, коммуникацию между участниками интеллектуального пространства и предоставляет доступ ко всей информации с используемых устройств. Серверное программное обеспечение платформы meSchup IoT может автоматически обнаруживать устройства и конфигурировать их для участия в сценариях. Сценарии взаимодействия разрабатываются на событийной скриптовой платформе, которая исполняет код на языке программирования Javascript.

Решение CHROMOSOME [6, 7] облегчает интеграцию гетерогенных компонентов для автоматизации индустриальных и

повседневных задач, получение данных с устройств в беспроводных сетях и коммуникацию между автономными автоматизированными системами посредством высокоуровневой платформы управления. Архитектура платформы разработана с использованием слоя аппаратных абстракций (hardware abstraction layer) над исполняющей платформой. Ключевые сервисы платформы обеспечивают датацентрический подход к коммуникациям, а высокоуровневые компоненты обеспечивают логику приложений. При этом во время работы платформы топология системы может измениться из-за подключения и отключения устройств. Для решения этой проблемы обеспечивается вычисление таблицы сетевой маршрутизации, не прерывая работы системы. Датацентрический подход к коммуникациям базируется на принципах «публикация/подписка» и «запрос/ответ» (клиент/сервер), а также возможности конфигурации коммуникационных отношений при помощи фильтрации потоков данных через атрибуты. Интероперабельность между компонентами приложения от разных разработчиков достигается при помощи моделирования областей знаний с использованием «словарей», в которых заранее смоделированы понятия и концепты. Помимо уточнения семантического значения предметов в теме, «словарь» уточняет атрибуты предметов, который более широко описывают тему. CHROMOSOME поддерживает работу на операционных системах Windows, Linux и FreeRTOS.

Статья [8] описывает создание интеллектуального пространства с использованием онтологии верхнего уровня для обеспечения базовых концепций среди различных приложений. Работа с онтологией (хранение и обработка информации и семантические запросы) реализована при помощи набора для создания приложений в семантическом вебе Jena2. Обнаружение участников интеллектуального пространства, отслеживание событий и отправка сообщений реализовано при помощи Siemens UPnP SDK.

Решение XANA [9] расширяет продукт Team Computing (TeC) [10] при помощи концепции «программного обеспечения как производственной линии» (Software Product Line — SPL). Данное решение задумано для упрощения создания, настройки и развертывания приложения для интеллектуальных домов при помощи экспертов, при этом конечный пользователь должен настроить полученную конфигурацию при помощи параметров, предоставленных экспертами. Платформа TeC является событийно-ориентированной и предоставляет разработчикам язык, который позволяет использовать диаграммы для работы в команде для

достижения общих целей. Решение XANA тестировалось с использованием аппаратных средств X10.

Проект PECES (PErvasive Computing in Embedded Systems) [11] нацелен на создание платформы для информационного взаимодействия различных устройств, которые находятся в разных интеллектуальных пространствах. Для обеспечения интероперабельности между этими устройствами используются заранее разработанные онтологии. Для обеспечения кооперации участников платформа использует следующие технологии: Aura [12] (создание интеллектуальных сред для предоставления сервисов) и BASE [13] (поддержка адаптации коммуникационных протоколов и технологий). Для своевременной реакции на изменения в окружающей среде платформа автоматически создает конфигурацию устройств и обновляет ее, используя концепцию ролей и правил. Роли могут быть присвоены любому устройству, а правила определяют контекстные ограничения на присвоение ролей для участия в работе приложения. Для работы с несколькими интеллектуальными пространствами проект PECES использует следующие компоненты: «Координатор» (устройство, которое отвечает за идентификацию участников интеллектуальных пространств основываясь на ролях и правилах), «Участник» (динамические входящее и уходящее из интеллектуального пространства устройство, которое может быть использовано в работе) и «Шлюз» (устройство, которое обеспечивает связь между участниками различных интеллектуальных пространств).

Авторы статьи [14] описывают платформу для поддержки семантической интероперабильности в интеллектуальных пространствах на основе механизма публикации-подписки. Данное решение разработано для концепции Интернета вещей (Internet of Things — IoT). Архитектура приложения основана на идеях платформы Smart-M3, но при этом поддерживает только две операции работы с информационным хранилищем: обновление информации и подписка на ее изменения. Из отличительных особенностей серверной части решения можно отметить наличие механизма отложенного обновления хранилища и возможность отмены обновления хранилища в случае, если связанное с ним событие не наступило.

Однако данные решения не подходят для сценариев с небольшим количеством участников (до 20), предполагающих использование устройства на базе операционной системы DD-WRT для организации интеллектуального пространства. Часть систем слишком узконаправленные (основное предназначение — работа с

датчиками в умных домах) [4, 8, 9, 14], а часть из них можно установить только на стационарный компьютер [6, 11].

3. Интеллектуальное пространство. Smart-M3. Платформа с открытыми исходными кодами Smart-M3 [15] на базе программного обеспечения RedSib [16] позволяет обеспечивать совместный онтологоориентированный доступ к информации и знаниям различных устройств, объединенных в интеллектуальное пространство. Smart-M3 состоит из двух частей [17]: ядра (Kernel) и интеллектуальных агентов (KP — Knowledge Processor). Ядро, в свою очередь, состоит из: семантического информационного брокера (Semantic Information Broker — SIB) и информационного хранилища (Information Storage). SIB принимает входящие запросы от информационных агентов на операции по взаимодействию с информационным хранилищем и отправляет результаты таких операций обратно. Существуют различные реализации семантического информационного брокера [18]. Так, например, работы [19, 20] описывают реализацию семантического информационного брокера CuteSIB для платформы Smart-M3. Данная реализация брокера использует язык программирования C++ вместе с программной платформой Qt и имеет модульную архитектуру, что позволяет сторонним разработчикам расширить при необходимости функционал платформы. Благодаря этим особенностям, пользователи могут скомпилировать и установить CuteSIB на разных операционных системах и платформах, в том числе и на маршрутизаторах. В статье [21] авторы описывают еще одну реализацию семантического информационного брокера для платформы Smart-M3 — pySIB. Основная особенность данного брокера — реализация функционала на языке программирования Python, использование модульной архитектуры и использование протокола JSSAP (операции протокола SSAP записываются в формате JSON, что уменьшает количество передаваемой информации на 10% в случае больших сообщений и на 40% для маленьких). Архитектура pySIB позволяет использовать модуль обеспечения приватности, который гарантирует конфиденциальность информации и авторизацию информационных агентов. Авторы выбрали за основу реализацию семантического информационного брокера на основе RedSib, которая зарекомендовала себя как самая стабильная версия платформы. Для других реализаций процесс интеграции Smart-M3 и маршрутизатора будет аналогичным.

Вся информация, расположенная в информационном хранилище, хранится в форме графа, который построен по правилам модели представления структурированных данных RDF — Resource Description Framework. Согласно этой модели информация

описывается тройками — «Субъект — Предикат — Объект» (субъект воздействует на объект предикатом). Субъект и предикат может являться URI ссылкой (Uniform Resource Identifier — унифицированный идентификатор ресурса), объект может быть либо URI ссылкой, либо литералом (некое значение с определенным типом). Интеллектуальные агенты — программное обеспечение, которое функционирует на устройствах, являющихся участниками интеллектуального пространства. Интеллектуальные агенты обмениваются информацией и знаниями друг с другом в интеллектуальном пространстве, а также управляют устройствами в физическом пространстве на основе полученной информации и знаниях. Взаимодействие интеллектуальных агентов с SIB происходит посредствам протокола SSAP — Smart Space Access Protocol, операции которого описываются в формате XML.

Интеллектуальные агенты могут совершать следующие операции в интеллектуальном пространстве:

– подключаться к интеллектуальному пространству (Join) — перед работой информационному агенту необходимо зарегистрироваться в семантическом информационном брокере, при помощи которого будет происходить взаимодействие с интеллектуальным пространством;

– вставлять информацию в интеллектуальное пространство (Insert) в виде RDF-тройки;

– удалять информацию (Remove) из интеллектуального пространства;

– обновлять информацию (Update) путем удаления старой информации и вставкой новой;

– запрашивать необходимую информацию по шаблону (Query);

– подписываться на определенную информацию (Subscribe) — интеллектуальный агент подписывается на определенного вида информацию, и когда в интеллектуальном пространстве появляется информация, которая удовлетворяет подписке, то информация об этом посылается интеллектуальному агенту;

– отписываться от определенной информации (Unsubscribe);

– покидать интеллектуальное пространство (Leave).

– Платформу Smart-M3 можно загрузить с сайта Sourceforge [15] — официального репозитория проекта. Платформа представлена в двух вариантах: *redsib_0.9.2-src.tar.gz*, представляющий собой архив с исходными кодами пакетов, составляющих платформу и *redsib_0.9.2_amd64.tar.gz*, являющийся скомпилированной версией для операционной системы Ubuntu версии

не менее 10.04 (архитектуры x86 и x64). Для того чтобы установить платформу на Wi-Fi маршрутизатор, необходимо скомпилировать исходные коды для операционной системы DD-WRT, которая управляет маршрутизатором. Для этого надо скомпилировать следующие пакеты, входящие в состав Smart-M3.

– Libxml — библиотека, являющаяся XML-анализатором.

– Redland — набор библиотек, направленных на работу с RDF. Redland поддерживает работу с языком запросов SPARQL и предоставляет свое собственное API по работе с хранением троек.

– Raptor — библиотека, предоставляющая возможность анализа и сериализации RDF-троек, необходимая для работы набора библиотек Redland.

– Rasqal — библиотека, обеспечивающая возможность работы с языком запросов SPARQL, необходимая для работы набора библиотек Redland.

– Whiteboard — пакет, реализующий функционал классной доски, необходимой как для работы интеллектуальных агентов, так и для семантического информационного брокера. Данный пакет необходим для работы пакета redsibd.

– Redsibd — реализация семантического информационного брокера.

– Sib-tcp — реализация работы с сокетами для передачи данных от информационных агентов к семантическому информационному брокеру и обратно.

4. Устройство на базе ОС DD-WRT для организации интеллектуального пространства. В качестве базового устройства для организации интеллектуального пространства был выбран Wi-Fi маршрутизатор, поддерживающий альтернативное стороннее программное обеспечение: DD-WRT [22] или OpenWRT [23]. Данные проекты основаны на ядре операционной системы Linux и расширяют стандартные возможности маршрутизатора. После установки такого программного обеспечения появляется возможность обратиться к файловой системе маршрутизатора при помощи протокола SSH и устанавливать дополнительное программное обеспечение. Установка производится либо путем компиляции исходного кода на маршрутизаторе, либо за счет использования кросс-компиляции (компилирование исходных кодов программного обеспечения на персональном компьютере для другой архитектуры процессора, отличной от архитектуры процессора персонального компьютера), либо, используя пакетные менеджеры, которые предоставляют проекты DD-WRT и OpenWRT.

На выбор маршрутизатора влияют следующие факторы: возможность установки альтернативного программного обеспечения DD-WRT\OpenWRT и наличие USB-порта (опционально). Возможность установки альтернативного программного обеспечения на ту или иную модель маршрутизатора можно посмотреть на сайте проектов: DD-WRT / OpenWRT. Если в выбранном маршрутизаторе небольшое количество энергонезависимой памяти (менее 1 гигабайт), то необходимо также выбрать маршрутизатор с USB-портом, который необходим для подключения флэш-накопителя для хранения пакетов и программного обеспечения платформы Smart-M3. При выборе встроенного программного обеспечения для маршрутизатора необходимо обратить внимание на поддержку в используемом ядре операционной системы USB, в противном случае смонтировать флэш-накопитель не получится.

В рамках данной работы использовался маршрутизатор Asus RT-N16 и программное обеспечение DD-WRT в качестве альтернативной прошивки. Инструкцию по установке данного внутреннего программного обеспечения для маршрутизатора Asus RT-N16 можно получить на официальном сайте DD-WRT. Общий принцип установки системы: сброс заводских настроек маршрутизатора (очистка NVRAM — энергонезависимая память, в которой хранятся настройки маршрутизатора), установка начальной версии DD-WRT, сброс заводских настроек, установка полноценной версии DD-WRT, сброс заводских настроек.

После установки прошивки для доступа к файловой системе маршрутизатора необходимо включить SSH-доступ к маршрутизатору. Для этого необходимо в меню «Services» активировать опцию «Enable SSHd» и перезагрузить маршрутизатор.

Перед использованием флэш-накопителя необходимо произвести операцию форматирования и разбития памяти на разделы. Необходимо создать следующие разделы.

– Раздел «Optware» размером 1-2 гигабайт с файловой системой *ext3*. Данный раздел используется как хранилище библиотек для установленных на флэш-накопитель приложений.

– Раздел «Swapfile» размером 64-256 мегабайт с файловой системой *linux-swap*. Данный раздел используется как файл подкачки в том случае, если операционная система будет работать с данными, которые целиком не могут поместиться в оперативную память.

– Раздел «Data» на все оставшееся место на флэш-накопителя с файловой системой *ext3*. Данный раздел будет использоваться как хранилище данных.

После подготовки накопителя необходимо включить поддержку USB флэш-накопителя на самом маршрутизаторе. Для маршрутизатора ASUS RT-N16 для выполнения этой процедуры необходимо открыть панель управления маршрутизатора, перейти в меню «Services», выбрать пункт подменю «USB», и включить следующие опции: «Core USB Support», «USB Storage Support». Далее, необходимо включить опцию «Automatic Drive Mount» и выбрать раздел */opt* как «Disk Mount Point» — раздел диска, который будет автоматически монтироваться при включении маршрутизатора и загрузки операционной системы.

Однако таким образом будет подсоединен только один раздел диска, поэтому необходимо выполнить скрипт, который при старте системы будет монтировать раздел «Data» флэш-накопителя (создать в разделе */opt* скрипт *startup.bash*, содержимое которого представлено в листинге 1).

```
mount /dev/discs/disc0/part3 /mnt
mount /opt/jffs/ /jffs
```

Листинг 1. Код скрипта startup.bash

После создания скрипта *startup.bash*, необходимо обеспечить его запуск каждый раз при включении маршрутизатора и загрузки внутреннего программного обеспечения. Для этого необходимо перейти в меню «Services», выбрать пункт подменю «USB» и указать в опции «Run-on-mount Script Name» путь до скрипта *startup.bash*.

5. Организация интеллектуального пространства. Для компиляции исходного кода платформы Smart-M3 на маршрутизаторе необходимо поставить компилятор GCC (https://gcc.gnu.org/) и сопутствующие библиотеки. Первоначально в маршрутизаторе эти компоненты отсутствуют, поэтому их необходимо поставить самостоятельно. Это можно сделать при помощи установки пакетов Optware [24] проекта NSLU2-Linux на маршрутизатор. Перед установкой необходимо увеличить размер файловой структуры JFFS (Journaling Flash File System). Для этого необходимо добавить в скрипт startup.bash следующую строчку: *mount /opt/jffs /jffs*. Таким образом, размер файловой системы JFFS будет увеличен за счет директории на разделе */opt*. Установка пакетов дополнительного программного обеспечения Optware происходит при помощи скрипта, содержимое которого представлено в листинге 2:

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

Листинг 2. Код скрипта установки пакетов Optware

Загрузку дополнительного программного обеспечения можно выполнить при помощи команды *ipkg-opt install*. Для компиляции платформы Smart-M3 необходимо поставить следующие пакеты с помощью утилиты ipkg:

– ipkg-opt install buildroot (пакет builtroot включает в себя GCC компилятор);

– ipkg-opt install optware-devel (пакет optware-devel включает в себя набор библиотек и заголовочных файлов, необходимых для компиляций программ);

– ipkg-opt install busybox (пакет busybox включает в себя обновленную версию интерпретатора bash).

Необходимо учесть, что все пакеты будут установлены в раздел */opt* флэш-накопителя в котором и будет происходить дальнейшая компиляция платформы Smart-M3.

Перед началом компиляции необходимо подготовить маршрутизатор к компиляции платформы, путем исполнения скрипта *before_compilation.bash*, код которого приведен в листинге 3. Данный скрипт удаляет все записи о местонахождении пользовательских динамических библиотек и определяет поиск исполняемых файлов в разделе */opt* (расположенный на флэш-накопителе), на котором находится компилятор и сопутствующие библиотеки. Данные операции необходимо провести для того, чтобы однозначно определить местонахождение компилятора, библиотек и программного обеспечения в разделе */opt*, так как часть необходимых библиотек уже предустановлена в файловой системе маршрутизатора, однако они являются устаревшими без возможности их обновления.

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

Листинг 3. Код скрипта before_compilation.bash

После выполнения скрипта необходимо загрузить архив с исходным кодом платформы Smart-M3 [15] на флэш-накопитель. Компиляция платформы Smart-M3 происходит путем первоначальной установки зависимостей. Каждый пакет необходимо сконфигурировать для маршрутизатора при помощи команды *configure*, причем необходимо указать будущее место установки программ и библиотек в разделе */opt* флэш-накопителя. После конфигурации необходимо произвести компиляцию при помощи команды *make* и в случае успешной компиляции установить приложение при помощи команды *make install*. Общий сценарий установки представлен в листинге 4.

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

Листинг 4. Общий сценарий установки пакетов

Исходный код включает в себя пакеты, необходимые для работы платформы. При помощи подчеркивания авторы будут отмечать те пакеты, которые уже находятся в архиве, в противном случае, необходимо будет скачать дополнительные пакеты и перенести их в файловую систему маршрутизатора.

Установка платформы Smart-M3 имеет следующий порядок.

– Пакеты Libxml, Raptor устанавливаются согласно листингу 4.

– Установка пакета Rasqal имеет следующий нюанс. Библиотека uClibc, используемая в прошивке DD-WRT, не поддерживает функцию округления *round()*, поэтому необходимо модифицировать данный пакет для успешной компиляции и установки. В связи с этим необходимо добавить свою реализацию функции *round()*, в файл *src/rasqal_literal.c*. Также необходимо удалить проверку существования функции *round()* в конфигурационном файле. Данную операцию можно осуществить двумя способами: 1) отключить аварийное завершение конфигурации при отсутствии функции округления в системных библиотеках (удалить строчку «*AC MSG ERROR([Could not find ceil, floor, round in default libs or with -lm])*» в файле configure.ac и выполнить команду autoreconf – i в корневой папке пакета); 2) изменить скрипт *configure* путем удаления выше описанной строчки. После этого необходимо установить пакет, как описано в листинге 4.

– Пакет bdb устанавливается при помощи пакетного менеджера ipkg: *ipkg-opt install libdb*.

– Для установки пакета Redland необходимо сконфигурировать работу платформы, указав местоположение библиотек bdb и включив поддержку потоков в соответствии с листингом 5.

– Пакеты libffi-3.2.1, gettext-0.19.2, glib 2.28.2, dbus-1.10.6, dbus-glib-0.100, libuuid-1.0.3, Whiteboard, Sib-tcp, libtool 1.14 и Redsibd устанавливаются в соответствии с листингом 4. При этом версии пакета glib 2.28.2, которые вышли позднее указанной версии не допустимы, так как они не совместимы с библиотекой uClibc.

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

Листинг 5. Сценарий установки пакета Redland

Для работы платформы Smart-M3 необходимо запустить сессионную шину dbus, а также программы *redsibd* и *sib-tcp*. Автоматически сессионная шина при запуске маршрутизатора не создается, поэтому необходимо создавать ее вручную при каждом запуске платформы Smart-M3 и уничтожать при завершении работы.

Запуск сессионной шины, посредством которой происходит взаимодействие компонентов платформы, можно осуществить при помощи команды *dbus-launch*. Для взаимодействия с друг другом программам *redsibd* и *sib-tcp* необходимо, чтобы в переменных окружения DBUS_SESSION_BUS_ADDRESS и DBUS_SESSION_BUS_PID были занесены адрес и идентификационный номер процесса сессии. Таким образом, в листинге 6 отображен скрипт запуска сессионной шины и платформы Smart-M3.

```
#!/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
```

Листинг 6. Скрипт запуска платформы Smart-M3

При запуске платформы Smart-M3 необходимо сохранять идентификационные номера процессов в файл, которые используются для корректного завершения работы платформы. На листинге 7 показан скрипт завершения работы платформы Smart-M3 и закрытия сессии dbus:

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

Листинг 7. Скрипт выключения платформы Smart-M3

6. Веб-сервис управления платформой Smart-M3. Авторами был разработан веб-сервис «Smart-M3 Control Panel» для управления платформой Smart-M3, установленной на маршрутизатор. Веб-сервис может быть легко адаптирован для работы на персональном компьютере. Разработанный веб-сервис обладает следующим функционалом:

– запуск, остановка и перезапуск платформы Smart-M3 в случае ошибок;

– отображение текущего статуса платформы: «запущено» («running»), «остановлено» («stopped»), «критическая ошибка системы» («breakdown»);

– просмотр лог-файлов платформы Smart-M3;

– отображение содержимого хранилища RDF-троек в реальном времени с возможностью фильтрации и сортировки содержимого (рисунок 1);

– добавление, изменение и удаление RDF-троек из хранилища;

– установка опций запуска платформы Smart-M3, включающая в себя: выбор хранилища для RDF-троек; выбор хранилища для подписок; ограничение количество потоков обработки SPARQL-запросов; установка порта, по которому будет осуществляться общение между интеллектуальными агентами и платформой; установка интервала опроса подписок;

– установка опций отображения хранилища RDF-троек: выбор видимости составляющих RDF-тройки и отображение служебных RDF-троек платформы.

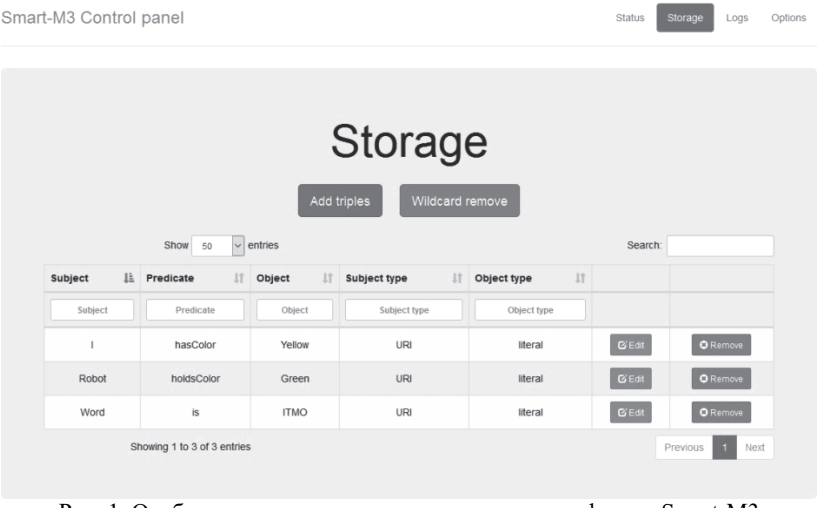

Рис. 1. Отображение содержимого хранилища платформы Smart-M3

Сервис «Smart-M3 Control Panel» был реализован на языке Python версии 2.7. Во время разработки сервиса были использованы следующие технологии и программное обеспечение.

– Легковесная программная платформа для созданий web-приложений Flask (http://flask.pocoo.org/).

– Технология Socket.IO (http://socket.io/), предоставляющая двустороннюю событийную коммуникацию в режиме реального времени на основе вебсокетов (websockets) и AJAX-сообщений. Авторы использовали данную технологию для реализации взаимодействия пользователей с интеллектуальными агентами на маршрутизаторе. Расширение программной платформы Flask-SocketIO предоставляет необходимый функционал на стороне сервера, а socketIO-client 0.7.2 — на стороне клиента.

– Сетевые библиотеки Gevent (http://www.gevent.org/) и greenlet (https://greenlet.readthedocs.io) используются для обеспечения кооперативной многозадачности на основе микропотоков.

– Gevent-websocket 0.9.5 является расширением сетевой библиотеки gevent, которая используется Flask-SocketIO для обеспечения работоспособности websocket-сервера;

– Библиотека jQuery (https://jquery.com/) используется для интерактивного изменения структуры страниц на клиентской стороне.

– Библиотека Python-KP (https://github.com/smart-m3/python_kp) представляет собой реализацию интеллектуального агента на языке Python.

Архитектура сервиса «Smart-M3 Control Panel» представлена на рисунке 2. Компоненты FlaskApp и SmartM3Watcher устанавливаются на маршрутизатор. Компонент FlaskApp включает в себя веб-сервер, который по HTTP-запросу отображает пользователю содержимое статических страниц и Socket.IO-сервер, который выполняет действия с хранилищем платформой Smart-M3 и возвращает ответ отправителю. Компонент SmartM3Watcher следит за текущим состоянием платформы, а также за изменениями в информационном хранилище. При каждом изменении данный компонент отправляет информацию компоненту FlaskApp по Socket.IO протоколу, после чего происходит широковещательная трансляция данных изменений всем пользователям.

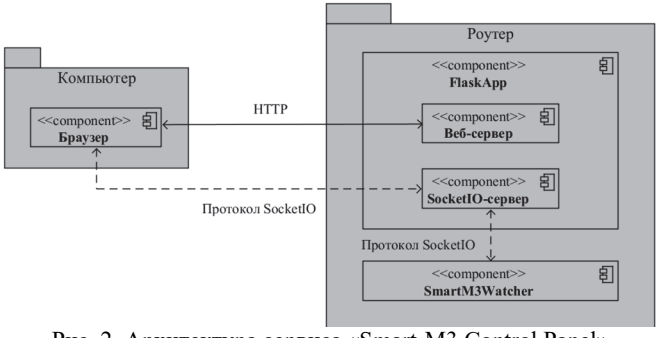

Рис. 2. Архитектура сервиса «Smart-M3 Control Panel»

Исходный код сервиса «Smart-M3 Control Panel» доступен по адресу [25]. Проект содержит два пакета: smartM3ControlPanelFlask (компонент FlaskApp, скомпилированный для архитектуры процессора маршрутизатора) и smartM3ControlPanelWatcher (компонент SmartM3Watcher).

Компонент smartM3ControlPanelFlask содержит следующую структуру.

– Файл smartm3_control_panel_flask.py является исполняемым скриптом, который является точкой входа в приложение.

– Директория app/files хранит в себе файлы конфигурации параметров запуска платформы Smart-M3 и отображения информации. Во время функционирования сервиса в данную директорию будут сохраняться лог-файлы платформы.

– Директория app/modules содержит модули веб-сервиса, которые используются в работе приложения. Модуль «main» отвечает за заглавную страницу сайта, модуль «storage» за взаимодействие с информационным хранилищем платформы Smart-M3, модуль «logs» за взаимодействие с лог-файлами, модуль «options» за настройку опций запуска платформы Smart-M3 и отображения информации в модуле «storage».

Каждый из этих модулей содержит файл *routes.py*, в котором прописаны маршруты обращений и файл *events.py*, который определяет реакцию сервиса на события Socket.IO. Также в данной директории содержится модуль smartM3KP, который является реализацией информационного агента.

– В директории app/scripts содержатся скрипты для запуска и остановки платформы Smart-M3. Представлены версии скриптов как для маршрутизатора Asus RT-N16, так и для персональных компьютеров.

– В директории app/static расположены статические .css и .js файлы, которые необходимы для работы веб-сайта;

– В директории app/templates находятся шаблоны html страниц;

– Файл app/settings.py содержит в себе настройки сервиса smartM3ControlPanelFlask, в котором можно указать платформу на которой предполагается работа, а также пути для модулей сайта.

Компонент smartM3ControlPanelWatcher содержит следующую структуру.

– Файл smartM3Watcher.py является исполняемым файлом, который является точкой входа в приложение.

– Файл app/WatcherThread.py является абстрактным классом потока, который должен работать в отдельном процессе.

– Файл app/WatcherStatus.py унаследован от класса WatcherThread, который работает в отдельном процессе и следит за текущим состоянием платформы Smart-M3, а также сообщает Socket.IO-серверу об изменении состояния платформы.

– Файл app/WatcherStorage.py унаследованный от класса WatcherThread, который работает в отдельном процессе и следит за содержимым информационного хранилища платформы Smart-M3, а также сообщает SocketIO-серверу об изменении в хранилище.

– Файл app/SmartM3KP.py представляет собой реализацию информационного агента;

– Файл app/settings.py включает в себя настройки сервиса smartM3ControlPanelWatcher, в котором можно указать платформу, на которой будет работать сервис.

7. Сравнение производительности платформы Smart-M3, функционирующей на маршрутизаторе и на персональном компьютере. Использование платформы Smart-M3 для функционирования на маршрутизаторе предполагается для небольших сценариев (до 20 мобильных роботов). В таких сценариях в среднем каждый робот оперирует с не более чем пятью подписками и порядка 50 RDF-троек. Таким образом, в среднем, в сценариях такого типа подразумевается использование порядка 100 подписок и около 1000 RDF-троек в информационном хранилище. После установки платформы Smart-M3 на маршрутизатор Asus RT-N16 было произведено измерение ее производительности. Были осуществлены замеры скорости вставки RDF-троек и обращения к ним при помощи запросов, а также вычисление максимального количества возможных подписок и скорости их обработки. Аналогичные измерения были произведены на компьютере Acer Aspire E5-774G с использованием операционной системы Linux Mint, функционирующую с использованием виртуальной машины Oracle VirtualBox.

Для каждого набора данных, полученных при измерении производительности платформы был вычислен 10-ый и 90-ый перцентиль, и все данные, которые меньше по значению, чем 10-ый перцентиль, и больше по значению, чем 90-ый, были исключены из рассмотрения для уменьшения «шума» в выборках. Оставшиеся данные в каждой из выборок были сгруппированы в наборы по 50 значений и у каждого из этого набора было вычислено среднее значение. Усредненные значения групп были использованы как исходные данные для графиков производительности платформы Smart-M3.

Измерение скорости вставки RDF-троек и запросов к информационному хранилищу включало в себя вставку тройки вида

<’someone_i’, ‘is_a’, ‘something_i’>, где *i* — номер в цикле и запрос вида <null, ‘is_a’, null>. Данная операция повторялась в цикле 10000 раз.

На рисунках 3 и 4 показаны измерения скорости вставки RDF-троек на персональном компьютере и маршрутизаторе. Тесты с использованием bdb (Berkeley DB) в качестве хранилища RDF-троек показали, что платформа Smart-M3 на маршрутизаторе стабильно оперирует с не более чем 9500 RDF-тройками, что практически сопоставимо с максимальным количеством RDF-троек на персональном компьютере — 10000. Разница между максимальным и минимальным значениями времени вставки на компьютере составляет 0,0005-0,001 секунд, а на маршрутизаторе — 0,2 секунды. Данные значения могут быть объяснены разницей в скорости записи памяти на накопителе на жестких дисках и флэш-накопителе.

На рисунках 5 и 6 отображено время обработки запросов к информационному хранилищу платформы Smart-M3 на персональном компьютере и маршрутизаторе. Оба графика показывают прямую зависимость между количеством RDF-троек и временем обработки запроса. Диапазон значений времени запросов на персональном компьютере составляет 0,02-0,1 секунды, у маршрутизатора — 0,5-3,5 секунд.

На рисунках 7 и 8 изображено максимальное количество подписок и время их обработки в платформе Smart-M3 на маршрутизаторе и компьютере. Измерения включали в себя создание N подписок подряд, где N — максимально возможное количество подписок, найденное экспериментально для обеих платформ, и измерение скорости реакции подписки на вставленную в информационное хранилище RDF-тройку. Максимальное количество подписок, которые позволяет использовать платформа Smart-M3 на маршрутизаторе около 650, а на персональном компьютере около 1000.

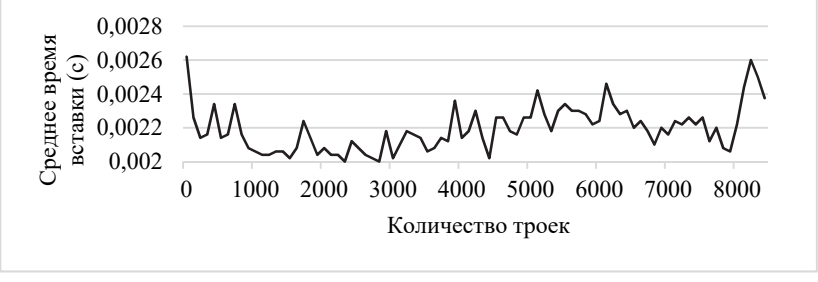

Рис 3. Измерение скорости вставки RDF-троек в информационное хранилище на персональном компьютере

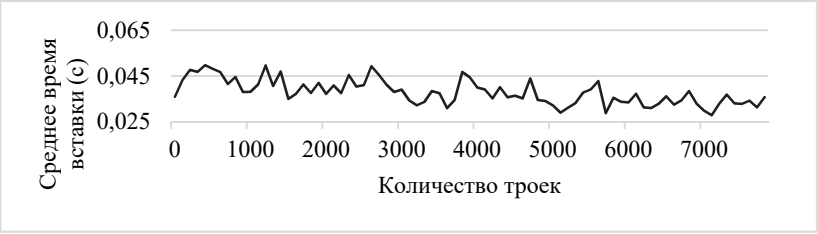

Рис 4. Измерение скорости вставки RDF-троек в информационное хранилище на маршрутизаторе

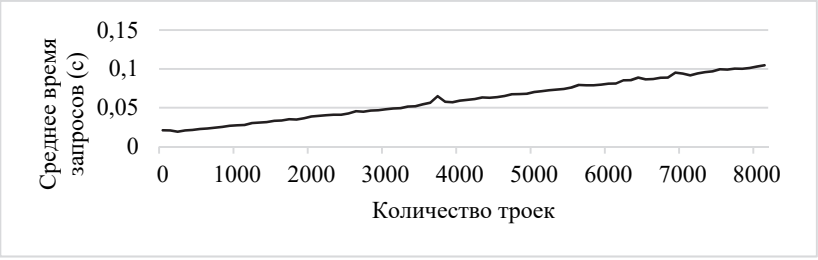

Рис 5. Время обработки запросов к информационному хранилищу на персональном компьютере

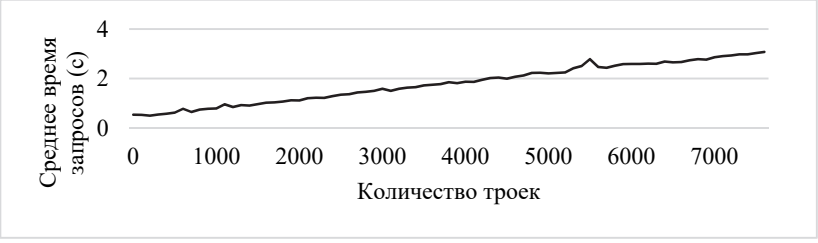

Рис 6. Время обработки запросов к информационному хранилищу на маршрутизаторе

Разница между минимальным и максимальным значениями на персональном компьютере равна 0,002 секунды, а на маршрутизаторе — 0,2 секунд. Графики зависимости времени обработки подписок от количества подписок на обеих платформах отличаются друг от друга. График производительности маршрутизатора указывает на прямую зависимость, а график производительности персонального компьютера не отображает зависимости между величинами. Этот факт можно объяснить небольшим количеством оперативной памяти маршрутизатора и недостаточной пропускной способностью протокола USB.

Представленные измерения показали, что производительность платформы Smart-M3 на маршрутизаторе уступает использованию персонального компьютера. Разница по времени вставки RDF-троек отличается в 10-15 раз; по запросам на информацию из информационного хранилища — в 25-30 раз; а по обработке подписок — в 4-6 раз. Данный разброс значений можно объяснить разной скорость доступа и записи к энергонезависимой памяти на персональном компьютере и маршрутизаторе. Однако производительность платформы Smart-M3 на маршрутизаторе достаточна для вышеуказанных сценариев коллаборативной работы мобильных роботов, так как они не требуют обработки запросов в реальном времени.

Использование маршрутизатора для таких сценариев показало, что время от времени случаются задержки в его работе (рисунок 9). К примеру, в данном промежутке времени, время вставки 8000 RDF-троек могло достигать 10-40 секунд, когда нормальным значением является промежуток в 2-3 секунды. Данные задержки могут быть обусловлены процессом перемещения данных из оперативной памяти в память флэш-накопителя. Однако такие задержки встречаются нечасто и связаны главным образом с перегрузкой маршрутизатора.

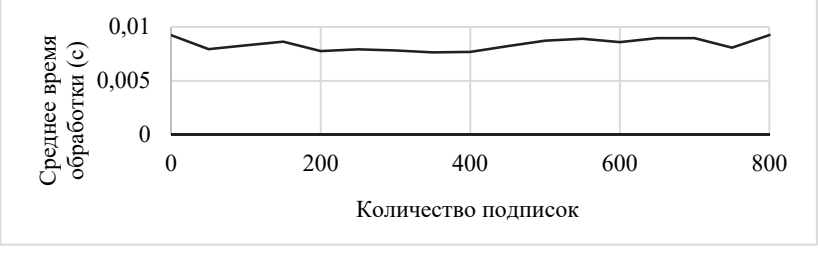

Рис. 7. Время обработки операции подписки на персональном компьютере

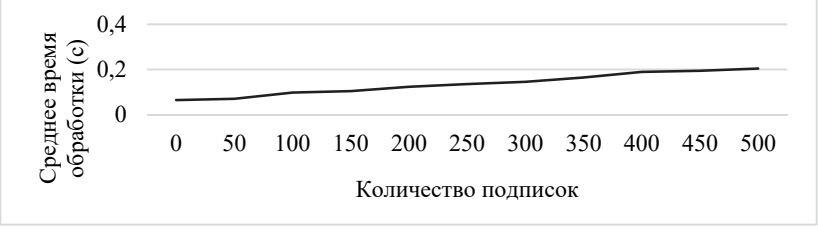

Рис. 8. Время обработки операции подписки на маршрутизаторе

Рис. 9. Пример медленной работы маршрутизатора при измерении скорости вставки RDF-троек

8. Заключение. В статье решена задача организации интеллектуального пространства на базе платформы Smart-M3 с использованием устройства на основе операционной системы DD-WRT для сценариев коллаборативной работы мобильных роботов. В статье подробно описан процесс интеграции на примере устройства Asus RT-N16. Компилирование пакетов платформы выполняется непосредственно в самом маршрутизаторе. В качестве альтернативного варианта компиляции пакетов может быть использована кросс-компиляция на персональном компьютере с использованием набора утилит и библиотек для выбранной архитектуры процессора.

В статье подробно описан веб-сервис «Smart-M3 Control Panel», который позволяет пользователю управлять платформой Smart-M3. Сервис предоставляет возможность просматривать текущий статус платформы, включать/выключать/перезапускать в зависимости от ее состояния, просматривать содержимое информационного хранилища, загружать лог-файлы и изменять настройки платформы. В дальнейшем авторы планируют добавить возможность визуализации содержимого информационного хранилища.

Литература

1. *Smirnov A. et al.* Ontology-based collaboration in multi-robot system: Approach and case study // 11th Systems of Systems Engineering Conference (SoSE 2016). 2016. pp. 1–6.
2. *Smirnov A. et al.* Multi-level Robots Self-organization in Smart Space: Approach and Case Study // 15th International Conference on Internet of Things, Smart Spaces, and Next Generation Networks and Systems (NEW2AN 2015). 8th Conference on ruSMART 2015. St. Petersburg. 2015. pp. 68–79.

3. *Smirnov A. et al.* Smart M3-Based Robot Interaction Scenario for Coalition Work // Proceedings of the First International Conference on Interactive Collaborative Robotics (ICR 2016). 2016. pp. 199–207.
4. *Kubitza T.* Apps for Environments: Demonstrating Pluggable Apps for Multi-Device IoT-Setups // Proceedings of the 6th International Conference on the Internet of Things (IoT'16). 2016. pp. 185–186.
5. *Kubitza T., Schmidt A.* Towards a Toolkit for the Rapid Creation and Programming of Smart Environments // End-User Development: Proceedings of the 5th International Symposium. 2015. LNCS 9083. pp. 230.
6. *Buckl C. et al.* CHROMOSOME: A Run-Time Environment for Plug & Play-Capable Embedded Real-Time Systems // SIGBED Rev. 2014. vol. 11 no. 3 pp. 36–39.
7. *Sommer S. et al.* Reconfigurable Industrial Process Monitoring using the CHROMOSOME Middleware // SIGBED Rev. 2013. vol. 10 no. 4. pp. 61–64.
8. *Wang X. et al.* Semantic Space: An Infrastructure for Smart Spaces // IEEE Pervasive Computing. 2004. vol. 3 no. 3 pp. 32–39.
9. *Tzeremes V., Gomaa H.* 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. 2015. pp. 23–26.
10. *Sousa J.P., Tzeremes V., El-Masri A.* Space-aware TeC: End-user development of safety and control systems for smart spaces // IEEE International Conference of Systems, Man and Cybernetics. 2010. pp. 2914–2921.
11. *Selvarajah K., Zhao R., Speirs N.* Building Smart Space Applications with PErvasive Computing in Embedded Systems (PECES) Middleware // GSTF Journal on Computing (JoC). 2012. vol. 1. Issue 4. pp. 57–62.
12. *Garlan D., Siewiorek D., Smailagic A., Steenkiste P.* Project Aura: Towards Distraction-Free Pervasive Computing // IEEE Pervasive Computing. 2002. vol. 1. no. 2. pp. 22–31.
13. *Becker C., Schiele G., Gubbels H., Rothermel K.* BASE - A Micro-broker-based Middleware For Pervasive Computing // 1st IEEE International Conference on Pervasive Computing and Communication. 2003. pp. 443–451.
14. *Roffia L. et al.* A semantic publish-subscribe architecture for the Internet of Things // IEEE Internet of Things Journal. 2016. vol. 3. Issue 6. pp. 1274–1296.
15. Официальный репозиторий платформы Smart-M3. URL: https://sourceforge.net/projects/smart-m3/files/Smart-M3-RedSIB_0.9.2 (дата обращения: 13.04.2017).
16. *Morandi F. et al.* RedSib: A Smart-M3 semantic information broker implementation // Proceedings of the 12th Conference of Open Innovations Association FRUCT and Seminar on e-Tourism. 2012. pp. 86–98.
17. *Honkola J., Laine H., Brown R., Tyrkko O.* Smart-M3 Information Sharing Platform // Proceedings of the The IEEE symposium on Computers and Communications. 2010. pp. 1041–1046.
18. *Viola F. et al.* The M3 Architecture for Smart Spaces: Overview of Semantic Information Broker Implementations // Proceedings of the 19th Conference Open Innovations Association FRUCT. 2016. pp. 264–272.
19. *Galov I., Lomov A., Korzun D.* Design of Semantic Information Broker for Localized Computing Environments in the Internet of Things // The 17th Conference of Open Innovations Association FRUCT. 2015. pp. 36–43.
20. *Korzun D., Galov I., Lomov A.* Smart Space Deployment in Wireless and Mobile Settings of the Internet of Things // The 3rd IEEE Int'l Symp. on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems. 2016. pp. 86–91.

21. *Viola F., D'Elia A., Roffia L., Cinotti T.S.* A modular lightweight implementation of the Smart-M3 semantic information broker // 18th Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT). 2016. pp. 370–377.
22. Официальный сайт проекта DD-WRT. URL: http://www.dd-wrt.com/site/index (дата обращения: 13.04.2017).
23. Официальный сайт проекта OpenWrt. URL: https://openwrt.org/ (дата обращения: 13.04.2017).
24. Официальный сайт проекта NSLU2-Linux. URL: http://www.nslu2-linux.org/wiki/Optware/Packages?from=Unslung.Packages (дата обращения: 13.04.2017).
25. Официальный репозиторий проекта Smart-M3 Control Panel. URL: https://sourceforge.net/projects/smart-m3/files/SmartM3ControlPanel/ (дата обращение: 13.04.2017).

Михайлов Сергей Андреевич — студент, Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики» (Университет ИТМО), программист лаборатории интегрированных систем автоматизации, Федеральное государственное бюджетное учреждение науки Санкт-Петербургского института информатики и автоматизации Российской академии наук (СПИИРАН). Область научных интересов: интеллектуальные пространства, робототехника, онтологии. Число научных публикаций — 5. saboteurincave@gmail.com; Биржевая линия, 14-16, Санкт-Петербург, 199034; р.т.: +7(911)7187033.

Кашевник Алексей Михайлович — к-т техн. наук, старший научный сотрудник лаборатории интегрированных систем автоматизации, Федеральное государственное бюджетное учреждение науки Санкт-Петербургского института информатики и автоматизации Российской академии наук (СПИИРАН). Область научных интересов: управление знаниями, облачные среды, человеко-машинное взаимодействие, робототехника, профилирование, онтологии, интеллектуальные пространства. Число научных публикаций — 150. alexey@iias.spb.su; 14-я линия, 39, Санкт-Петербург, 199178; р.т.: +7(812)328-8071, Факс: +7(812)328-0685.

Поддержка исследований. Работа выполнена при финансовой поддержке РФФИ (проект №16-29-04349), бюджетной темы №0073-2014-0005, субсидии 074-U01.

DOI 10.15622/sp.52.9

S.A. MIKHAILOV, A.M. KASHEVNIK

SMART-M3-BASED SMART SPACE CREATIONUSING A DD-WRT-BASED DEVICE

Mikhailov S.A., Kashevnik A.M. **Smart-M3-based Smart Space Creation using a DD-WRT-based Device.**

Annotation: Smart space is a service orientated infrastructure for knowledge sharing between devices. This paper describes a smart space creation process based on integration of Smart-M3 platform with DD-WRT-based device. Smart-M3 is an open source platform which implements the concept of smart space. Wi-Fi router was chosen as a DD-WRT-based device, which allows using it for smart space organization and provides wireless connection between devices at the same time. This method simplifies deployment of scenarios with several participants. The paper describes a process of compilation and installation of Smart-M3 platform to DD-WRT operating system. Testing of the main Smart-M3 operations showed that a smart space, organized in this way, can be used for considered scenarios. The authors have developed "Smart-M3 Control Panel" web-service which allows users to control the Smart-M3 platform by a graphical web interface. User of "Smart-M3 Control Panel" can view a current status of the platform; launch, stop or reload the platform; view information storage; download log files and change startup options. SocketIO protocol was used for user interaction with a web service.

Keywords: smart space, Smart-M3, router, DD-WRT, Smart-M3 Control Panel.

Mikhailov Sergei Andreevich — student, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics (ITMO University), programmer at computer aided integrated systems laboratory, St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS). Research interests: smart-spaces, robotics, ontologies. The number of publications — 5. saboteurincave@gmail.com; 14-16, Birgevaya line, St. Petersburg, 199034, Russia; office phone: +7(911)7187033.

Kashevnik Alexey Mihajlovich — Ph.D., senior researcher at computer aided integrated systems laboratory, St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS). Research interests: knowledge management, cloud computing, human-computer interaction, robotics, user profiling, ontologies, smart spaces. The number of publications — 150. alexey@iias.spb.su; 39, 14-th Line V.O., St. Petersburg, 199178, Russia; office phone: +7(812)328-8071, Fax: +7(812)328-0685.

Acknowledgements. This research is supported by RFBR (grant 16-29-04349), budgetary theme №0073-2014-0005, subsidy 074-U01.

References

1. Smirnov A. et al. Ontology-based collaboration in multi-robot system: Approach and case study. 11th Systems of Systems Engineering Conference (SoSE 2016). 2016. pp. 1–6.
2. Smirnov A. et al. Multi-level Robots Self-organization in Smart Space: Approach and Case Study // 15th International Conference on Internet of Things, Smart Spaces, and Next Generation Networks and Systems (NEW2AN 2015). 8th Conference on ruSMART 2015. St. Petersburg. 2015. pp. 68–79.
3. Smirnov A. et al. Smart M3-Based Robot Interaction Scenario for Coalition Work. Proceedings of the First International Conference on Interactive Collaborative Robotics (ICR 2016). 2016. pp. 199–207.
4. Kubitza T. Apps for Environments: Demonstrating Pluggable Apps for Multi-Device IoT-Setups. Proceedings of the 6th International Conference on the Internet of Things (IoT'16). 2016. pp. 185–186.
5. Kubitza T., Schmidt A. Towards a Toolkit for the Rapid Creation and Programming of Smart Environments. End-User Development: Proceedings of the 5th International Symposium. 2015. LNCS 9083. pp. 230.

6. Buckl C. et al. CHROMOSOME: A Run-Time Environment for Plug & Play-Capable Embedded Real-Time Systems. SIGBED Rev. 2014. vol. 11 no. 3 pp. 36–39.
7. Sommer S. et al. Reconfigurable Industrial Process Monitoring using the CHROMOSOME Middleware. SIGBED Rev. 2013. vol. 10 no. 4. pp. 61–64.
8. Wang X. et al. Semantic Space: An Infrastructure for Smart Spaces. *IEEE Pervasive Computing*. 2004. vol. 3 no. 3 pp. 32–39.
9. Tzeremes V., Gomaa H. 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. 2015. pp. 23–26.
10. Sousa J.P., Tzeremes V., El-Masri A. Space-aware TeC: End-user development of safety and control systems for smart spaces. IEEE International Conference of Systems, Man and Cybernetics. 2010. pp. 2914–2921.
11. Selvarajah K., Zhao R., Speirs N. Building Smart Space Applications with PErvasive Computing in Embedded Systems (PECES) Middleware. *GSTF Journal on Computing (JoC)*. 2012. vol. 1. Issue 4. pp. 57–62.
12. Garlan D., Siewiorek D., Smailagic A., Steenkiste P. Project Aura: Towards Distraction-Free Pervasive Computing. *IEEE Pervasive Computing*. 2002. vol. 1. no. 2. pp. 22–31.
13. Becker C., Schiele G., Gubbels H., Rothermel K. BASE - A Micro-broker-based Middleware For Pervasive Computing. 1st IEEE International Conference on Pervasive Computing and Communication. 2003. pp. 443–451.
14. Roffia L. et al. A semantic publish-subscribe architecture for the Internet of Things. *IEEE Internet of Things Journal*. 2016. vol. 3. Issue 6. pp. 1274–1296.
15. Oficial'nyj repozitorij platformy Smart-M3 [Official repository of Smart-M3 platform]. Avaiable at: https://sourceforge.net/projects/smart-m3/files/Smart-M3-RedSIB_0.9.2 (accessed: 13.04.2017).
16. Morandi F. et al. RedSib: A Smart-M3 semantic information broker implementation. Proceedings of the 12th Conference of Open Innovations Association FRUCT and Seminar on e-Tourism. 2012. pp. 86–98.
17. Honkola J., Laine H., Brown R., Tyrkko O. Smart-M3 Information Sharing Platform. Proceedings of the The IEEE symposium on Computers and Communications. 2010. pp. 1041–1046.
18. Viola F. et al. The M3 Architecture for Smart Spaces: Overview of Semantic Information Broker Implementations. Proceedings of the 19th Conference Open Innovations Association FRUCT. 2016. pp. 264–272.
19. Galov I., Lomov A., Korzun D. Design of Semantic Information Broker for Localized Computing Environments in the Internet of Things. The 17th Conference of Open Innovations Association FRUCT. 2015. pp. 36–43.
20. Korzun D., Galov I., Lomov A. Smart Space Deployment in Wireless and Mobile Settings of the Internet of Things. The 3rd IEEE Int'l Symp. on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems. 2016, pp. 86–91.
21. Viola F., D'Elia A., Roffia L., Cinotti T.S. A modular lightweight implementation of the Smart-M3 semantic information broker. 18th Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT). 2016. pp. 370–377.
22. Oficial'nyj sajt proekta DD-WRT [Official site of DD-WRT firmware]. Avaiable at:: http://www.dd-wrt.com/site/index (accessed: 13.04.2017).
23. Oficial'nyj sajt proekta OpenWr [Official site of OpenWrt firmware]. Avaiable at: https://openwrt.org/ (accessed: 13.04.2017).
24. Oficial'nyj sajt proekta NSLU2-Linux [Official site of NSLU2-Linux project]. Avaiable at: http://www.nslu2-linux.org/wiki/Optware/Packages?from=Unslung.Packages (accessed: 13.04.2017).
25. Oficial'nyj repozitorij proekta Smart-M3 Control Panel [Official site of Smart-M3 Control Panel project]. Avaiable at: https://sourceforge.net/projects/smart-m3/files/SmartM3ControlPanel/ (accessed: 13.04.2017).