

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение высшего
образования
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

УДК 004.04, 004.82, 004.7
№ госрегистрации 114121750065



ОТЧЁТ
О ПРИКЛАДНЫХ НАУЧНЫХ ИССЛЕДОВАНИЯХ

Разработка прототипа масштабируемой сервис-ориентированной программно-аппаратной платформы на основе беспроводных сенсорных и агентных сетей, технологий семантического веба и облачных вычислений в целях агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных в распределенной сети электронных потребительских устройств (Internet of Things)

по теме:

Экспериментальные исследования поставленных перед ПНИ задач
(промежуточный)

Этап 4

ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технического комплекса России на 2014-2020 годы»

Соглашение о предоставлении субсидии от 24.11.2014 г. №14.575.21.0101

Руководитель проекта
к.т.н., доцент

30.06.2016 Муромцев
(подпись, дата)

Санкт-Петербург 2016

СПИСОК ИСПОЛНИТЕЛЕЙ

Руководитель проекта:

доцент,
канд. техн. наук.

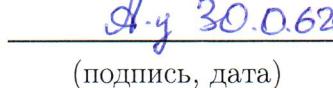

30.06.2016 Д.И.Муромцев
(подпись, дата) (введение, заключение)

Исполнители темы:

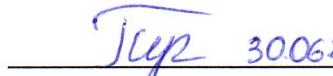
ассистент


30.06.2016 М.А. Колчин
(подпись, дата) (раздел 1, раздел 4)

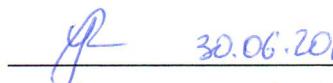
инженер


30.06.2016 А.А. Андреев
(подпись, дата) (раздел 1, раздел 2)

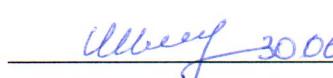
инженер


30.06.2016 Д.С. Гарайзуев
(подпись, дата) (раздел 3)

инженер


30.06.2016 Н.В. Климов
(подпись, дата) (раздел 4, раздел 8)

инженер


30.06.2016 И.А. Шилин
(подпись, дата) (раздел 6, раздел 7)

ассистент


30.06.2016 В.А. Заколдаев
(подпись, дата) (раздел 3)

заведующий лабораторией


30.06.2016 О.А. Кураш
(подпись, дата) (раздел 7, раздел 8)

главный специалист


30.06.2016 Д.О. Захаров
(подпись, дата) (раздел 4, раздел 5)

инженер

Юг 30.06.2016
(подпись, дата)

А.Н. Югансон
(раздел 4, раздел 5)

Индустриальный партнер:

ЗАО "МОРСКИЕ КОМПЬЮТЕРНЫЕ СИСТЕМЫ"
(раздел 9, раздел 10)

Руководитель работ


(подпись)

В.Г. Яковлев

Нормоконтролер:

Беззубик
(подпись, дата)

В.В. Беззубик

РЕФЕРАТ

Отчет 225 с., 13 рис., 5 табл., 18 прил., 1 ч.

СЕМАНТИЧЕСКИЙ ИНТЕРНЕТ ВЕЩЕЙ, СЕМАНТИЧЕСКИЕ ТЕХНОЛОГИИ, ОБЛАЧНЫЕ ВЫЧИСЛЕНИЯ, ОНТОЛОГИИ, ОБРАБОТКА СЛОЖНЫХ СОБЫТИЙ, СЕРВИС-ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА, БЕСПРОВОДНЫЕ СЕНСОРНЫЕ СЕТИ, ОБРАБОТКА ПОТОКОВЫХ RDF ДАННЫХ, ИНТЕРНЕТ ВЕЩЕЙ, ВИЗУАЛИЗАЦИЯ ГЕТЕРОГЕННЫХ ДАННЫХ, ИНТЕГРАЦИЯ ГЕТЕРОГЕННЫХ ДАННЫХ, АНАЛИЗ ПОТОКОВЫХ ДАННЫХ, АНАЛИЗ ГЕТЕРОГЕННЫХ ДАННЫХ, ПУБЛИКАЦИЯ ПОТОКОВЫХ ДАННЫХ.

В отчете представлены результаты исследований, выполненных по этапу 4 ПНИ «Разработка прототипа масштабируемой сервис-ориентированной программно-аппаратной платформы на основе беспроводных сенсорных и агентных сетей, технологий семантического веба и облачных вычислений в целях агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных в распределенной сети электронных потребительских устройств (Internet of Things)».

В данном отчете описываются результаты разработки и изготовление экспериментальных образцов систем сбора, нормализации, анализа и визуализации гетерогенных данных распределенной сети электронных потребительских устройств, а также их экспериментальные исследования. А именно следующих систем:

- а) макет шлюза сбора и первичной обработки данных (МШСПОИ),
- б) экспериментальный образец масштабируемой сервис-ориентированной программно-аппаратной платформы для сбора, нормализации, обработки и визуализации больших массивов гетерогенных данных распределенной сети электронных потребительских устройств (ЭО ПАП).

Для решения поставленных на данном этапе ПНИ задач применяются методы математического моделирования, методы онтологического инжиниринга, методы логического вывода на знаниях, технологии построения веб-

сервисов, технологии семантического веба, а также методы объектно-ориентированного программирования.

Решения разрабатываемые в рамках данного этапа в отличии от существующих решений позволяют реализовать механизм работы с существующими электронными потребительскими устройствами (Internet of Things) вне зависимости от модели устройств и поддерживаемых технологий передачи данных. Кроме того семантические веб технологии и методы онтологического инжиниринга позволяют реализовать эффективный обмен данными между несколькими информационными системами за счет использования верхнеуровневых онтологий, обеспечивающих унифицированную модель данных распределенной сети ЭПУ.

В приложениях А–В представлена эскизная конструкторская документация на МШСПОИ в составе схемы электрической функциональной, схемы комбинированной и чертежа общего вида. А в приложениях Г–Д представлена программная документация на МШСПОИ в составе описания и текста программы.

В приложениях Е–П представлена программная документация на ЭО ПО ПАП в составе спецификации, описаний и текстов программ подсистем ЭО ПО ПАП, описания применения ЭО ПО ПАП, а так же логическая и физическая структуры базы данных ЭО ПО ПАП.

В приложениях Р–Т представлена эскизная конструкторская документация на ЭО ПАП в составе схемы структурной, инструкции по эксплуатации и формуляра.

Характеристики разрабатываемых экспериментальных образцов оцениваются проведением лабораторных испытаний и экспериментальных исследований, программа и методики которых прилагаются к данному отчету о ПНИ. Для проведения лабораторных испытаний и экспериментальных исследований применяются следующие системы:

- а) Программно-аппаратный стенд (ПАС), изготовленный на 1-м этапе ПНИ. Он использовался для тестирования масштабируемости модулей сбора, нормализации и хранения данных ЭО ПАП.
- б) Экспериментальный образец внешней информационной системы, который использовался для тестирования масштабируемости ин-

терфейса прикладного программирования ЭО ПАП, через который внешние информационные системы получают доступ к данным электронных потребительских устройств, подключенных к ЭО ПАП.

Результаты полученные на данном этапе прикладного научного исследования соответствуют Плану-графику работ и требованиям технического задания. В частности, разрабатываемый ЭО ПАП позволяет обрабатывать информацию о не менее 1000 одновременно подключенных электронных устройств. А так же МШСПОИ может предоставлять доступ к данным ЭПУ не менее 50 клиентам.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	14
1 Разработка МШСПОИ	16
1.1 Назначение МШСПОИ	16
1.2 Состав МШСПОИ	16
1.3 Разработка аппаратного обеспечения	17
1.4 Разработка программного обеспечения	18
1.5 Выводы	19
2 Разработка программы и методики лабораторных испытаний МШСПОИ	20
2.1 Программа лабораторных испытаний МШСПОИ	20
2.2 Методики лабораторных испытаний МШСПОИ	20
2.2.1 Методика проверки эскизной конструкторской и программной документации на соответствие установленной комплектности и оценка её качества	20
2.2.2 Методика проверки соответствия объекта испытаний его эскизной конструкторской и программной документации	21
2.2.3 Методика проверки соответствия показателей значениям заданным техническим заданием	21
2.3 Выводы	21
3 Лабораторные испытания МШСПОИ	22
3.1 Проверка технической документации на соответствие установленной комплектности и её качества	22
3.2 Проверка соответствия объекта испытаний его технической документации	22
3.3 Проверка соответствия показателей значениям заданным техническим заданием	22
3.4 Выводы	26
4 Разработка ЭО ПО ПАП	27
4.1 Назначение программной части ЭО ПАП	27
4.2 Состав программной части ЭО ПАП	27

4.3	Разработка подсистемы СОС	27
4.4	Разработка подсистемы ЦОД	29
4.5	Разработка подсистемы ПЭ МШСПОИ	32
4.6	Выводы	33
5	Разработка ЭО ПАП	34
5.1	Назначение программно-аппаратной платформы	34
5.2	Состав программно-аппаратной платформы	34
5.3	Аппаратное обеспечение	34
5.3.1	Сервер для работы подсистемы ЦОД	34
5.3.2	Компьютер для работы подсистемы СОС	37
5.3.3	Макет шлюза сбора и первичной обработки данных . .	37
5.4	Программное обеспечение	37
5.5	Выводы	38
6	Разработка программы и методики экспериментальных исследований ЭО ПАП	41
6.1	Программа экспериментальных исследований ЭО ПАП . . .	41
6.2	Методика проверки технической документации на соответствие установленной комплектности и оценки её качества . .	42
6.3	Методика проверки соответствия объекта исследований его эскизной конструкторской и программной документации . .	42
6.4	Методика проверки соответствия показателей значениям заданным техническим заданием	42
6.4.1	Состав методик выполнения проверок	42
6.4.2	Проверка показателей работы интерактивной визуализации ЭО ПАП	43
6.4.3	Проверка работоспособности ЭО ПАП при хранении большого объема данных	44
6.5	Выводы	45
7	Экспериментальные исследования ЭО ПАП с использованием ПАС .	47
7.1	Проверка показателей работы интерактивной визуализации ЭО ПАП	47
7.2	Проверка работоспособности ЭО ПАП при хранении большого объема данных	47

7.3	Выводы	49
8	Экспериментальные исследования ЭО ПАП с использованием экспериментального образца внешней информационной системы, получающей доступ к гетерогенным данным распределенной сети электронных потребительских устройств через интерфейс прикладного программирования ЭО ПО ПАП	50
8.1	Состав проверок с использованием экспериментального образца внешней информационной системы	50
8.2	Проверка поддержки синхронных и асинхронных режимов передачи данных	50
8.3	Проверка времени доступа к показаниям	50
8.4	Проверка времени исполнения команд для процессов	52
8.5	Выводы	53
9	Изготовление макета шлюза сбора и первичной обработки данных (МШСПОИ)	54
9.1	Состав работ по изготовлению МШСПОИ	54
9.2	Подгтовка и сборка аппаратной части МШСПОИ	54
9.3	Разработка исполняемого файла и развертывание на МШСПОИ	56
9.4	Выводы	57
10	Изготовление экспериментального образца масштабируемой сервис-ориентированной программно-аппаратной платформы для сбора, нормализации, обработки и визуализации больших массивов гетерогенных (структурированных, полуструктурированных и неструктурированных) первичных исследовательских данных в распределенной сети электронных потребительских устройств (Internet of Things)	58
10.1	Состав работ по изготовлению ЭО ПАП	58
10.2	Разработка исполняемых файлов программного обеспечения ЭО ПАП	58
10.3	Установка и конфигурирование операционной системы и необходимого программного обеспечения	61
10.4	Установка и конфигурирование программного обеспечения ЭО ПАП	61

10.5	Выводы	62
	ЗАКЛЮЧЕНИЕ	64
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	66
	Приложение А Схема электрическая функциональная МШСПОИ . . .	67
	Приложение Б Схема комбинированная соединений и подключения МШСПОИ	68
	Приложение В Чертеж общего вида МШСПОИ	69
	Приложение Г Описание программного обеспечения МШСПОИ	70
	Приложение Д Текст программы программного обеспечения МШСПОИ	77
	Приложение Е Спецификация программного обеспечения ЭО ПАП . .	102
	Приложение Ж Описание подсистемы ЦОД ЭО ПАП	109
	Приложение З Текст программы подсистемы ЦОД ЭО ПАП	119
	Приложение И Описание подсистемы СОС ЭО ПАП	150
	Приложение К Текст программы подсистемы СОС ЭО ПАП	159
	Приложение Л Описание программного эмулятора МШСПОИ	169
	Приложение М Текст программы программного эмулятора МШСПОИ	175
	Приложение Н Описание применения ЭО ПО ПАП	183
	Приложение О Описание логической структуры базы данных	191
	Приложение П Описание физической структуры базы данных	192
	Приложение Р Схема структурная ЭО ПАП	193
	Приложение С Инструкция по эксплуатации ЭО ПАП	194
	Приложение Т Формуляр ЭО ПАП	203

ОПРЕДЕЛЕНИЯ

В настоящем отчете о ПНИ применяют следующие термины с соответствующими определениями:

Интернет вещей (Internet of Things) — концепция сети однозначно идентифицируемых встраиваемых вычислительных устройств в рамках существующей инфраструктуры Интернета.

Технологии семантического веба (Semantic Web technologies) — технологии, которые предоставляют общий инструмент, позволяющий осуществлять обмен данными и их многократное использование между и за пределами программ, предприятий и сообществ.

Связанные данные (Linked Data) — методы публикации взаимосвязанных между собой наборов структурированных данных в рамках существующей инфраструктуры Интернета.

Онтология (Ontology) — попытка всеобъемлющей и подробной формализации некоторой области знаний с помощью концептуальной схемы. Такая схема состоит из структур данных, содержащей все релевантные классы объектов, их связи и правила, принятые в этой области.

Логический вывод (Logical reasoning) — процесс рассуждения, в ходе которого осуществляется переход от некоторых исходных суждений (предпосылок) к новым суждениям - заключениям.

Гетерогенность — разнородность, наличие неодинаковых частей в структуре, в составе чего-либо.

Интерактивная визуализация данных — способ графического представления информации, позволяющий пользователю взаимодействовать с системой отображения информации и наблюдать ответную реакцию системы.

REST (Representational State Transfer) — метод взаимодействия компонентов распределённого приложения в сети Интернет, при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос (обычно GET или POST; такой запрос называют REST-запрос), а необходимые данные передаются в качестве параметров запроса. Этот способ является альтернативой более сложным методам, таким как SOAP, CORBA и RPC.

Событие предметной области — какое-либо событие рассматриваемое в контексте конкретной предметной области. Например, превышение температуры теплоносителя при теплоснабжении жилого дома и т.п.

Система управления потоковыми данными — это программа для управления непрерывными потоками данных. Она аналогична системе управления базами данных (СУБД), которые, однако, предназначены для статических данных в обычных базах данных. СУПД также предлагает гибкую обработку запросов, так что информация может быть выражена с помощью запросов. Однако, в отличие от СУБД, в СУПД выполняет непрерывный запрос, который не только исполняется один раз, а исполняется постоянно. Таким образом, запрос непрерывно выполняется до тех пор, пока не будет явно удален.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

IoT — Internet of Things

WSN — Wireless Sensor Networks

ПО — программное обеспечение

URI — Unified Resource Identifier

UDP — User Datagram Protocol

HTTP — Hypertext Transfer Protocol

CoAP — Constrained Application Protocol

API — Application Programming Interface

ЭО ПАП — экспериментальный образец программно-аппаратной платформы

ПО ЭО ПАП — программное обеспечение экспериментального образца программно-аппаратной платформы

ПАС — программно-аппаратный стенд

ЭПУОВР — электронные потребительские устройства с ограниченными вычислительными ресурсами

SSN — Semantic Sensor Network Ontology

RDF — Resource Description Framework

OWL — Web Ontology Language

REST — Representational State Transfer

СУПД — Система управления потоковыми данными

Fuseki — Apache Jena Fuseki

OSGI — Open Services Gateway Initiative

ВВЕДЕНИЕ

В данном промежуточном отчете по 4-му этапу ПНИ «Разработка прототипа масштабируемой сервис-ориентированной программно-аппаратной платформы на основе беспроводных сенсорных и агентных сетей, технологий семантического веба и облачных вычислений в целях агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных в распределенной сети электронных потребительских устройств (Internet of Things)» приведены результаты теоретических исследований на данном этапе ПНИ.

Общими целями выполнения данного прикладного научного исследования являются:

- а) создание комплекса научных/научно-технических решений в области разработки методов и алгоритмов, обеспечивающих повышение эффективности научных исследований посредствам агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных распределенной сети электронных потребительских устройств (Internet of Things).
- б) получение значимых научных результатов в области разработки масштабируемой сервис-ориентированной программно-аппаратной платформы на основе беспроводных сенсорных и агентных сетей, технологий семантического веба и облачных вычислений в целях агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных в распределенной сети электронных потребительских устройств (Internet of Things).

Технологии Интернета вещей хоть и получили широкую известность, но все еще находятся на ранней стадии развития. Существует большое количество исследовательских и технических проблем, например такие как программная и аппаратная архитектура, стандартизация, безопасность и конфиденциальность. Научно-технические задачи, решаемые на данном этапе ПНИ, направлены на обеспечение синтаксической и семантической интеропе-

перабельности компонентов распределенной сети электронных потребительских устройств. А применение новых методов и инструментов технологий семантического веба определяют актуальность данной работы.

Целью работы на данном этапе ПНИ являются изготовление и экспериментальные исследования решений, разработанных на основе результатов полученных на предыдущих этапах ПНИ. На данном этапе были спроектированы, работаны и изготовлены следующие решения:

- а) макет шлюза сбора и первичной обработки данных (МШСПОИ),
- б) экспериментальный образец масштабируемой сервис-ориентированной программно-аппаратной платформы для сбора, нормализации, обработки и визуализации больших массивов гетерогенных данных распределенной сети электронных потребительских устройств (ЭО ПАП).

Эскизная конструкторская и программная документация на МШСПОИ представлена в Приложениях А–Д. В приложениях Е–П представлена программная документация на ЭО ПО ПАП. А в приложениях Р–Т представлена эскизная конструкторская документация на ЭО ПАП.

1 Разработка МШСПОИ

1.1 Назначение МШСПОИ

Макет шлюза сбора и первичной обработки данных предназначен для сбора данных распределенной сети электронных потребительских устройств по беспроводным и проводным каналам связи, их первичную обработку и отправку по каналам связи в подсистемы ЦОД, входящей в состав ЭО ПАП.

1.2 Состав МШСПОИ

На основе установленного назначения в МШСПОИ были сформированы следующие составные элементы:

- а) Основной вычислительный блок с модулем беспроводной связи на основе чипа Espressif ESP8266¹
- б) интерфейсы приема и передачи цифровых и аналоговых данных
- в) компоненты ввода и вывода информации
- г) вспомогательный блок для единовременного программирования МШСПОИ
- д) блок для начальной конфигурации МШСПОИ

Основный вычислительный блок на основе чипа Espressif ESP8266 уже содержит в себе модуль беспроводной передачи данных и интерфейсы приема и передачи цифровых данных, а также один аналогово-цифровой преобразователь, расширение возможностей которого происходит за счет подключения дополнительных компонентов ввода и вывода информации.

В качестве компонентов ввода и вывода информации были подготовлены тестовые блоки на основе цифровых и импульсных счетчиков потребления электрической энергии и расхода воды, систем реле и модулей индикации.

Блок для начальной конфигурации позволяет установить параметры инициализации каждого из компонентов ввода и вывода информации, а также основного вычислительного блока: таких, например, как параметры внешней сети, к которой необходимо осуществлять подключение. В разработан-

¹Официальный сайт Espressif: <https://espressif.com/>

ном решении аппаратный переключатель позволяет перейти вычислительному блоку в режим конфигурирования, при котором вычислительный блок автоматически переконфигурирует модуль беспроводной связи в режим программной точки доступа с временными фиксированными параметрами подключения, зависящими от серийного номера устройства.

Изменения состояний ресурсов и параметров конфигурации фиксируются в энергонезависимой памяти EEPROM, доступной как внутренний ресурс системы.

Вспомогательный блок единовременного конфигурирования позволяет программировать основной чип, записывая на него исполняемый файл при развертывании МШСПОИ. Он построен на основе универсального внутрисхемного программатора, переводящего сигналы ПК по шине USB в стандарты последовательного порта.

1.3 Разработка аппаратного обеспечения

В качестве основного модуля для разработанного МШСПОИ была выбрана система на чипе ESP8266 от Espressif, имеющая беспроводной модуль связи стандартов WiFi b/g/n и являющая одной из самых распространенных и доступных для программирования на момент разработки модуля¹.

Дополнительно было разработан вспомогательный модуль для начальной загрузки исполняемого бинарного файла для устройства согласно официальной документации о назначении портов ввода-вывода и логике работе программы² на основе универсального внутрисхемного программатора, преобразующего сигнал с ПК по шине USB в стандарты последовательного порта.

Дополнительно был выстроен внешний интерфейс для начального конфигурирования устройства с переключателем и двумя светодиодами для индикации текущего состояния ("подключено к внешней сети" / "находится в режиме конфигурирования"). В режиме подключения к внешней публичной сети недоступны ресурсы для конфигурирования. После конфигурирования параметры системы сохраняются в энергонезависимой EEPROM-памяти, доступной в системе, как внутренний ресурс.

¹Официальный сайт Espressif: <https://espressif.com>

²Портал разработчиков Espressif: <http://bbs.espressif.com/>

Все доступные выводы основной системы представлены в качестве открытых в зависимости от предлагаемого режима использования. В разработанном ПО выполнялась проверка подключения к световым индикаторам, а также к квартирным счетчикам расхода воды и энергопотребления.

Было выбран и сформирован корпус устройства, защищающий внутренние компоненты системы и предоставляющий интерфейсы ввода-вывода для подключения к необходимым актуаторам и сенсорам.

1.4 Разработка программного обеспечения

Для выработки методического подхода по развертыванию разрабатываемого решения были поставлены цели по достижению:

- а) поддержки автоматизированного развертывания на множестве устройств
- б) поддержки контролируемого и автоматического обновления
- в) минимализации необходимых задач по конфигурированию для конечного пользователя
- г) гибкости и богатства возможностей для расширения предлагаемой системы сторонними разработчиками

Благодаря выбранным целям, разрабатываемая программно-аппаратная платформа по своей архитектуре позволяет режиме получать параметры конфигурации локальной сети, в которой она развертывается. Программно реализована поддержка обновления прошивки модуля при конфигурации с сохранением предыдущих пользовательских параметров в энергонезависимой памяти EEPROM устройства.

Благодаря выбранной платформе Arduino и открытости программного обеспечения, разработка дополнительной логики и функционала приложения возможна, как на linux-совместимых платформах с набором библиотек WiringPi для Raspberry Pi¹, так и на более распространенных и доступных Arduino-совместимых устройствах, например ESP8266.

¹Библиотека WiringPi: <http://wiringpi.com/>

Разработанный модуль позволяет принимать данные с цифровых и аналоговых сенсоров и управлять подключенными актуаторами: не менее 8 устройств без дополнительных самостоятельных и практически неограниченное число в качестве внешнего модема беспроводной связи.

Каждый актуатор и сенсор представлен в качестве ресурса, доступного для чтения по протоколу СоАР для систем с ограниченными ресурсами¹. Сохранение в энергонезависимую EEPROM-память текущих данных об устройствах происходит автоматически при их изменении.

1.5 Выводы

В данном разделе отчета описано аппаратное и программное обеспечения макета шлюза сбора и первичной обработки данных. В результате данной работы были разработаны эскизная конструкторская документация и программная документация представленные в Приложениях А–Д в составе следующих документов:

- а) схемы электрической функциональной в соответствии с ГОСТ 2.701-84,
- б) схемы комбинированной в соответствии с ГОСТ 2.701-84,
- в) чертежа общего вида в соответствии с ГОСТ 2.102-68,
- г) описание программы в соответствии с ГОСТ 19.402-78,
- д) и текста программы в соответствии с ГОСТ 19.401-78.

По данной технической документацией Индустриальным партнером был разработан экспериментальный образец макета шлюза сбора и первичной обработки данных. Данный экспериментальный образец который использовался для проведения лабораторных испытаний, описанных в главе 3, в соответствии с программой и методиками, описанными в главе 2.

¹Описание технологии СоАР: <https://coap.technology>

2 Разработка программы и методики лабораторных испытаний МШСПОИ

2.1 Программа лабораторных испытаний МШСПОИ

Для проведения лабораторных испытаний МШСПОИ была разработана программа лабораторных испытаний, учитываяшая предъявляемые требования технического задания. Программа испытаний включается в себя 8 проверок:

- а) проверка технической документации на соответствие установленной комплектности и оценка её качества,
- б) проверка соответствия объекта испытаний его эскизной конструкторской и программной документациям,
- в) проверка МШСПОИ на осуществление первичного преобразования данных в RDF-модель,
- г) проверка МШСПОИ на возможность передачи данных на ЦОД по протоколу TCP/UDP/IP,
- д) поддержка платформы Arduino

2.2 Методики лабораторных испытаний МШСПОИ

2.2.1 Методика проверки эскизной конструкторской и программной документации на соответствие установленной комплектности и оценка её качества

Проверка выполняется следующим образом. Проверяется соответствие эскизной конструкторской и программной документации на объект испытаний комплектности и её качества - требованиям приведенным в Техническом задании для каждого пункта документа ГОСТа.

Комплект документации считается выдержаншим исследование, если его комплектность и качество соответствует требованиям приведенным в Техническом задании для каждого пункта ГОСТа.

2.2.2 Методика проверки соответствия объекта испытаний его эскизной конструкторской и программной документации

Проверка выполняется следующим образом. Проверяется соответствие объекта испытаний его эскизной конструкторской и программной документации. Объект испытаний считается выдержавшим проверку, если он соответствует каждому документу комплекта документации.

2.2.3 Методика проверки соответствия показателей значением заданным техническим заданием

Проверка выполняется следующим образом.

Если в каком-либо из случаев испытаний в допустимых изготавителем изделия внешних условиях от объекта получено сообщение RDF, не соответствующее нотации, то объект испытаний считается не выдержавшим проверку, во всех остальных случаях выдержавшим.

2.3 Выводы

В данном разделе отчета описана программа и методики лабораторных испытаний макета шлюза сбора и первичной обработки данных. Результате данной работы был разработан документ «Программа и методики лабораторных испытаний МШСПОИ 1000.000000.000ПМ», который идет в дополнении к данному отчету в виде отдельного документа с приложениями.

В разработанную программу лабораторных испытаний включены проверки разработанной эскизной конструкторской и программной документации в соответствии с пунктом 6.1.4 ТЗ и проверки соответствия характеристик МШСПОИ пункту 4.3.3 ТЗ.

3 Лабораторные испытания МШСПОИ

3.1 Проверка технической документации на соответствие установленной комплектности и её качества

Проверка комплектности и качества технической документации осуществлялась в соответствии с пунктом 6 Технического задания. В результате проверки установленно, что техническая документация соответствует требуемой комплектности и качеству.

3.2 Проверка соответствия объекта испытаний его технической документации

Проверка соответствия объекта испытаний осуществлялась в соответствии с технической документацией, представленной в Приложениях А–Д. В результате проверки установлено, что объект испытаний соответствует его технической документации.

3.3 Проверка соответствия показателей значениям заданным техническим заданием

При проведении лабораторных испытаний МШСПОИ выполнялась проверка соответствия техническим характеристикам:

- а) осуществление первичного преобразования данных в RDF-модель,
- б) возможность передачи данных на ЦОД по протоколу TCP или UDP,
- в) реализация на архитектуре контроллера Arduino MEGA или аналоге.

Реализация МШСПОИ произведена на чипе Espressif ESP8266 12e¹ с использование окружения Arduino. Реализация окружения Arduino для выбранной платформы позволяет запускать подготовленные приложения напря-

¹См. http://www.esp8266.com/wiki/doku.php?id=esp8266-module-family#esp-12-e_q

мую на модуле ESP8266, без необходимости использовать дополнительный микроконтроллер.

Ядро окружения Arduino¹ для системы ESP8266 включает библиотеки коммуникации по беспроводной сети Wi-Fi, поддерживаемой данным модулем, используя протоколы TCP и UDP на основе стандарта IP версии 4.

Для получения текущей версии окружения Arduino используются следующие команды:

```
cd ~  
git clone https://github.com/esp8266/Arduino.git  
mv Arduino esp8266  
cd esp8266/tools  
python get.py
```

Далее для загрузки версии 2.3.0 ядра окружения, необходимо выполнить следующую команду:

```
git checkout tags/2.3.0
```

Для выбора репозитория, и после осуществить загрузку на чип ESP8266.

```
cd ~  
git clone https://github.com/semitoproject/minicoap.git  
cd makeEspArduino  
make -f makeEspArduino.mk upload
```

После выполнения указанных команд осуществляется сборка и загрузка разработанной версии МШСПОИ для модуля ESP8266.

Подключение модуля ESP8266 к персональному компьютеру для загрузки ПО МШСПОИ осуществляется единожды при изготовлении согласно разработанной схеме электрической с увеличенной стабильностью за счет добавления подтягивающих сопротивлений и емкости. Схема электрическая представлена в Приложении А.

При испытаниях схемы с улучшенной стабильностью успешные запуски модуля при загрузке ПО и перезагрузке увеличились с 20 процентов до 100, что отражено на рисунке 8.3.

Для проверки успешности реализации поддерживаемого протокола была реализована симуляция пользовательских запросов на модуль, поддерживаемый согласно ТЗ до 4 клиентов для одновременного обслуживания.

¹См. <https://github.com/esp8266/Arduino>



Рисунок 3.1 — Число успешных запусков при использовании выработанной схемы с улучшенной стабильностью

Серия тестов была выполнена для оценки поддержки ответа на запросы получения текущей информации, отправка запросов на управления и обнаружение модуля. Успешность результатов зафиксирована на рисунке 3.2, каждая серия запросов содержала 10 тыс запросов, вычислялось среднее время задержки на ответы, оно не привысило 400 мс.

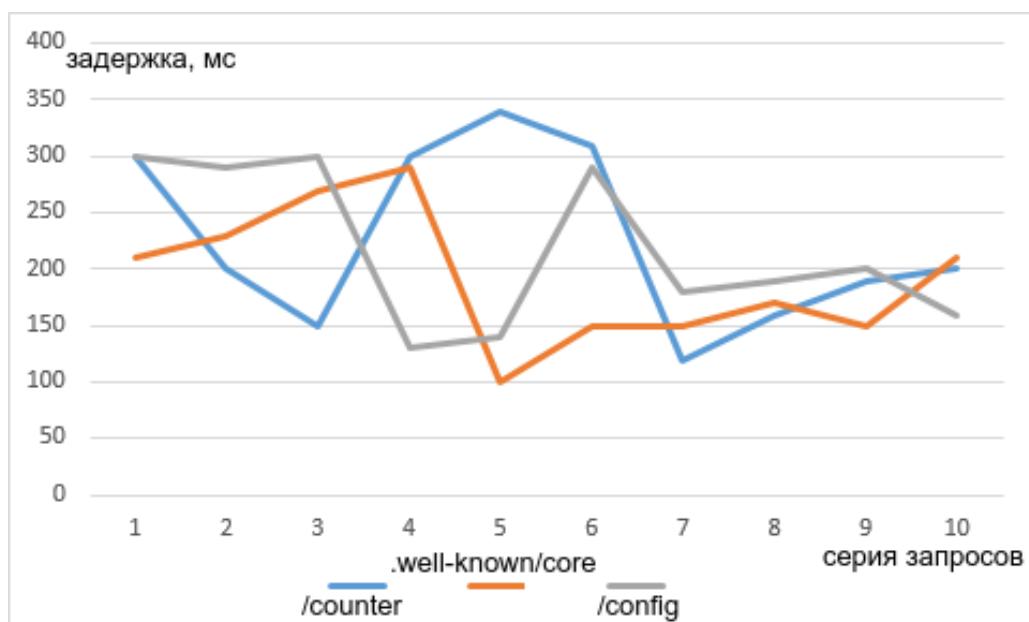


Рисунок 3.2 — Результаты тестов по оценке поддержки ответов на запросы

По получению результатов ответа на выполняемые запросы каждый ответ был проанализирован в автоматическом режиме на предмет соответствия представления данных выбранной RDF модели. Каждый из запросов успешно прошел проверку.

Проверка методов доступа по конфигурированию модуля и обнаружению ресурсов для конфигурирования, используя RDF-модель для представления данных осуществлялась следующим образом:

Широковещательный GET-запрос на стандартный порт СоАР-протокола 5683 и стандартный ресурс для обнаружения данных: *.well-known/core*, результат – список ресурсов модуля:

```
</counter>;  
</config>;  
</config/schema>;  
</config/context>
```

В случае попыток доступа к config-ресурсов из внешней сети возвращается стандартная ошибка доступа согласно выбранной политике безопасности.

GET-запрос на ресурс */config* возвращает текущую конфигурацию в JSON-LD формате, либо стандартную ошибку 404 в случае ее отсутствия. PUT запрос принимает JSON-LD сообщения с новой конфигурацией.

Для ресурсов:

- a) */config/schema* - GET запрос возвращает схему конфигурации в JSON-LD формате.
- б) */config/context* - GET запрос возвращает JSON-LD контекст конфигурации.

Результат возвращаемого контекста, доступного для изменения:

```
{  
    "@context": {  
        "xsd": "http://www.w3.org/2001/XMLSchema#",  
        "rdfs": "http://www.w3.org/2000/01/rdf-schema#",  
        "@vocab": "/config/schema"  
    }  
}
```

Результат возвращаемой конфигурационной схемы. Имя каждого поля конфигурации содержит URI-идентификатор, при этом поле @vocab содержит словарь используемых префиксов для упрощения работы со схемой:

```
{  
    "@context": "/config/context",  
    "wifi-name": {  
        "@type": "xsd:string",  
        "rdfs:label": "Wi-Fi Network Name"  
    },  
    "wifi-password": {  
        "@type": "xsd:string",  
        "rdfs:label": "Wi-Fi Password"  
    }  
}
```

Результат возвращаемой конфигурации:

```
{  
    "@context": "/config/context",  
    "wifi-name": "ISST",  
    "wifi-password": "changed_password"  
}
```

Таким образом, лабораторные испытания отражают успешность выполнения поставленных требований к техническим характеристикам МШСПОИ.

Программа и методики лабораторных испытаний, а также протоколы прилагаются к данному отчету о ПНИ в виде отдельного самостоятельного документа.

3.4 Выводы

В данном разделе отчета описаны лабораторные испытания макета шлюза сбора и первичной обработки данных в соответствии с разработанной программой и методиками лабораторных испытаний. В программу испытаний входило 8 проверок с использованием экспериментального образца МШСПОИ разработанного Индустриальным партнером в соответствии с эскизной конструкторской и программной документациями.

По результатам проведенных лабораторных испытаний сделан вывод о соответствии МШСПОИ всем требованиям Технического задания, а именно требованиям изложенным в пунктах 4.3.3 и 6.1.4 ТЗ.

4 Разработка ЭО ПО ПАП

4.1 Назначение программной части ЭО ПАП

ЭО ПО ПАП предназначен для агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, слабоструктурированных и неструктурных данных распределенной сети электронных потребительских устройств.

4.2 Состав программной части ЭО ПАП

В состав ЭО ПО ПАП входят следующие подсистемы:

- а) Симулятор облака сенсоров (далее - подсистема СОС),
- б) Центр обработки данных (далее - ЦОД),
- в) Макет шлюза сбора и первичной обработки данных (далее - МШСПОИ).

Подсистема ЦОД состоит из следующих модулей:

- а) модуль сбора данных (далее - МСД),
- б) модуль агрегации данных (далее - МАД),
- в) модуль нормализации и анализа данных (далее - МНАД),
- г) модуль интерактивной визуализации и трансляции данных (далее - МИВТД),
- д) модуль работы с хранилищем данных (далее - МБД),
- е) модуль работы с хранилищем семантических метаданных и онтологий (далее - МБЗ)

Состав модулей соответствует пункту 4.3.5.3 Технического задания.

4.3 Разработка подсистемы СОС

СОС предназначен для имитации работы распределенной сети электронных потребительских устройств (Internet of Things).

В качестве виртуальных электронных устройств были выбраны:

- а) комнатный датчик температуры,
- б) уличный датчик температуры,
- в) и устройство регулирования теплоподачи.

Используя данные виртуальные электронные устройства, подсистемой СОС реализуется имитация мониторинга и управления теплоснабжением жилого дома. То есть комнатные датчики температуры сообщают о текущей температуре в квартирах жилого дома, уличный датчик сообщает о текущей температуре на улице в районе жилого дома, а устройство регулирования теплоподачи позволяет увеличивать или уменьшать теплоснабжение, тем самым увеличивая или уменьшая температуру в квартирах.

В СОС раз в заданный период времени выполняется вычисление температуры уличного датчика случайной в заданном диапазоне. Диапазон температуры на улице и период времени изменения задается пользователем перед запуском СОС.

Значение устройства регулирования передачи измеряется в процентах и имеет диапазон значений от 0 до 100. Данное значение регулируется управляющими командами присыпаемыми по протоколу СоАР.

Для комнатного датчика температуры задана погрешность, т.к. в разных комнатах дома имеется различная потеря тепла. Температура каждой комнаты дома вычисляется на основании текущей температуры комнаты, оптимальной температуры для комнаты, заданной пользователем в конфигурационном файле перед запуском, температуры на улице и значения устройства регулирования теплопередачи:

$$t = T - S + P$$

Где t - вычисляемая температура в комнате, T - температура в комнате до вычисления, S - сдвиг температуры, P - погрешность температуры для комнаты.

Сдвиг температуры вычисляется по следующей формуле:

$$S = T - O - (50 * p/100 - 24 + Ts)$$

Где S - сдвиг температуры, T - температура в комнате до вычисления, O - оптимальная температура в комнате, р - значение устройства регулирования передачи, Ts - температура на улице.

Подсистема СОС реализована на языке Java для работы в среде Java Virtual Machine не ниже 8-й версии. И оптимизирована для работы на компьютере с характеристиками соответствующими требованиям пункта 4.3.4 Технического задания.

4.4 Разработка подсистемы ЦОД

Программа предназначена для сбора данных с узлов сенсорной сети, нормализации собранных данных на основе высокогорневых OWL- онтологий и анализа потоковых и исторических данных для выявления различных ситуаций предметной области.

Модульная структура подсистемы ЦОД представлена на рисунке 4.1.

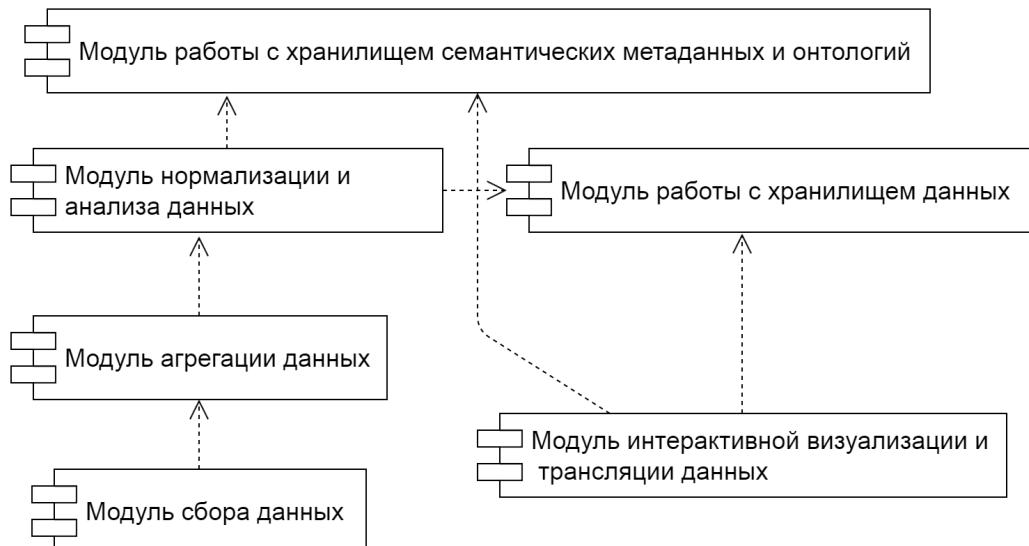


Рисунок 4.1 – Модульная структура подсистемы ЦОД

МСД обеспечивает прием и сохранение больших массивов данных от распределенной сети электронных устройств. МАД обеспечивает динамическую классификацию и идентификацию больших массивов данных, генерирующихся в распределенной сети электронных потребительских устройств. МНАД обеспечивает приведение к нормальному виду и препроцессинг больших массивов структурированных, полуструктурных и неструктурных данных. МИВДТ обеспечивает представление результатов агрега-

ции и анализа. МБД обеспечивает хранение данных. В качестве базы данных выбрана Cassandra, позволяющая работать с большими объемами данных. Описание физической структуры базы данных представлено в приложении П. Пример работы МИВДТ представлен на рисунках 4.2 и 4.3.

Systems	
1	Temperature Simulator Device / 100
2	Temperature Simulator Device / 106
3	Temperature Simulator Device / 10
4	Temperature Simulator Device / 1000
5	Temperature Simulator Device / 105
6	Temperature Simulator Device / 103
7	Temperature Simulator Device / 104
8	Temperature Simulator Device / 102
9	Temperature Simulator Device / 1
10	Temperature Simulator Device / 101

Рисунок 4.2 – Постраничный список добавленных электронных устройств

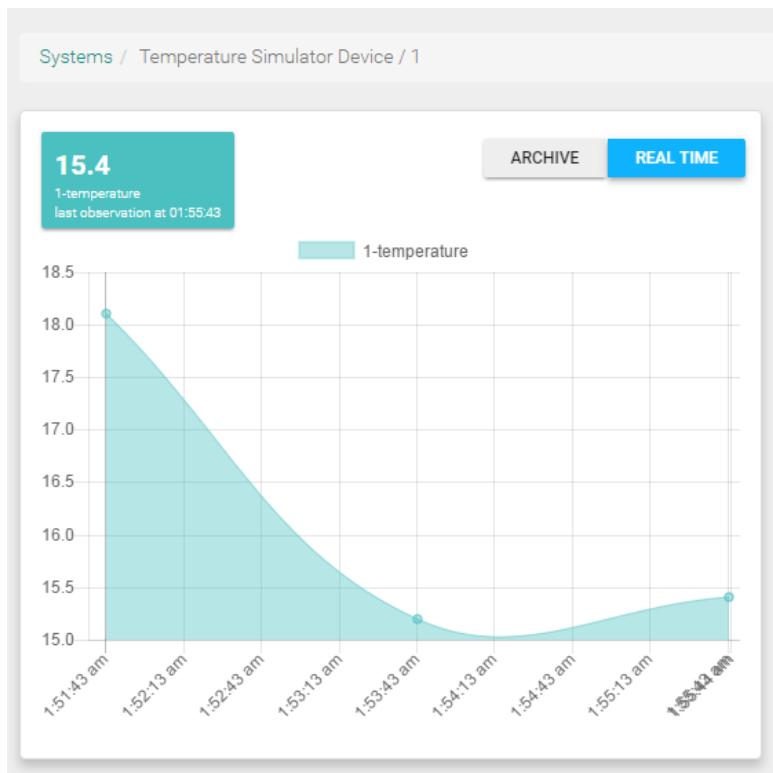


Рисунок 4.3 – Показания электронного устройства

Для подключения ЭПУОВР к ЭО ПАП используются драйвера в виде динамический загружаемых модулей (далее - драйверов), описанных в отчете за 2-й этап данного ПНИ [1]. В подсистеме ЦОД имеются уже разработанные

драйвера, которые можно переиспользовать для подключения ЭВУОПР. На рисунке 4.4 приведен интерфейс со списком доступных драйверов.

The screenshot shows a web-based interface for the SemIoT Platform. At the top, there is a navigation bar with tabs: 'SemIoT Platform' (selected), 'Explorer', and 'Configuration'. On the right side of the bar, it says 'root' with a dropdown arrow. Below the navigation bar, the main content area has a title 'Available drivers'. A table lists nine items, each with a 'Name' and an 'Action' column containing a blue 'INSTALL' button. The items are:

No	Name	Action
1	Weather Station (based on DHT-22)	INSTALL
2	Winghouse Machine Tool	INSTALL
3	Narodmon.ru Weather Station (Temperature & Humidity)	INSTALL
4	Netatmo Weather Station (Outdoor module)	INSTALL
5	Plain Lamp (LED)	INSTALL
6	Mercury 201.6 (over SemIoT Device)	INSTALL
7	Simulator	INSTALL
8	Temperature Simulator	INSTALL
9	Regulator Simulator	INSTALL

Рисунок 4.4 — Интерфейс со списком доступных драйверов

На рисунке 4.5 приведен интерфейс с конфигурацией одного из доступных драйверов.

The screenshot shows the 'New driver configuration' page. At the top, it says 'New driver configuration' and has a 'SAVE' button. Below that is a table with configuration parameters:

Username	garayzuev@gmail.com
Password	Ny7lZe#o
The initial delay	1
The polling interval	20
Client application ID	566ffb00e8ede139a1529208
Client application secret	C5hws3Fo0fG0wiZpdd1AKi64OTZK7G3lc0eu5R
Get from driver only new observations	true

Below the table is a section titled 'Repeatable configuration' with a 'ADD +' button. At the bottom left is a 'REMOVE -' button. At the very bottom, there is a row of numerical values: 60.11, 30.56, 59.81, 29.98, and a small icon.

Рисунок 4.5 — Интерфейс конфигурации драйвера

Подсистема ЦОД реализована на языке Java для работы в среде Java Virtual Machine не ниже 8-й версии. И оптимизирована для работы на ком-

компьютере с характеристиками соответствующими требованиям пункта 4.3.2 Технического задания.

4.5 Разработка подсистемы ПЭ МШСПОИ

Программный модуль МШСПОИ предназначен для предоставления доступа к данным электронных устройств и управлением через беспроводную сеть Wi-Fi. Программный модуль поддерживает основные открытые платформы и позволяет реализовать узел получения данных с устройств, не имеющих прямого доступа в проводные и беспроводные сети. Программный модуль реализует следующие функции:

- а) получение данные с электронных устройств любым доступным операционной системе или прошивке устройства образом (импульсные входы, цифровые и аналоговые входы, по прерываниям и в цикле),
- б) преобразование данных в RDF,
- в) формирование интерфейс доступа по выбранному бинарному протоколу RFC 7252 (Constrained Application Protocol (CoAP)) с возможностью чтения и записи данных.

Работа ПЭ МШСПОИ состоит из следующих этапов:

- а) ожидание данных из СОС,
- б) преобразование данных в RDF,
- в) запись данных в памяти,
- г) ожидание запроса на получение данных (от ЦОД),
- д) отправка данных по протоколу СоAP.

Подсистема ПЭ МШСПОИ реализована на языке Java для работы в среде Java Virtual Machine не ниже 8-й версии. И оптимизирована для работы на компьютере с характеристиками соответствующими требованиям пункта 4.3.3 Технического задания.

4.6 Выводы

В данном разделе отчета описано программное обеспечение масштабируемой сервис-ориентированной программно-аппаратной платформы для сбора, нормализации, обработки и визуализации больших массивов гетерогенных данных распределенной сети электронных потребительских устройств. В соответствии с пунктом 4.3.1 Технического задания, ЭО ПО ПАП состоит из следующих подсистем:

- а) подсистема СОС, имитирующая работы распределенной сети электронных потребительских устройств;
- б) подсистема ЦОД, выполняющая функции агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных;
- в) подсистема ПЭ МШСПОИ, имитирующая работу макета шлюза сбора и первичной обработки данных.

В результаты данной работы была разработана программная документация в соответствии с пунктом 6.1.3 ТЗ, представленная в Приложениях Е–П. Программная документация представлена в следующем составе:

- а) спецификации,
- б) описание программ подсистем в соответствии с ГОСТ 19.402-78,
- в) текстов программ подсистем в соответствии с ГОСТ 19.401-78,
- г) описания применения в соответствии с ГОСТ 19.502-78,
- д) описания логической и физической структуры базы данных.

По данной программной документации Индустриальным партнером был разработан экспериментальный образец ПАП. Данный экспериментальный образец был использован для проведения экспериментальных исследований, описанных в главах 7 и 8, в соответствии с программой и методиками, описанными в главе 6.

5 Разработка ЭО ПАП

5.1 Назначение программно-аппаратной платформы

Масштабируемая сервис-ориентированная программно-аппаратная платформа предназначена для сбора, нормализации, обработки и визуализации больших массивов гетерогенных данных распределенной сети электронных потребительских устройств (Internet of Things).

5.2 Состав программно-аппаратной платформы

ЭО ПАП состоит из нескольких компонентов, включающих как программное, так и аппаратное обеспечение:

а) программного обеспечения в составе:

- 1) подсистема СОС, предназначенная для имитации работы распределенной сети электронных потребительских устройств,
- 2) подсистема ЦОД, выполняющая функции агрегации, нормализации, анализа и визуализации гетерогенных данных,
- 3) программный эмулятор МШСПОИ для экспериментальных исследований ЭО ПАП.

б) аппаратного обеспечения в составе:

- 1) сервера, предназначенного для работы подсистемы ЦОД;
- 2) персонального компьютера, предназначенного для работы подсистемы СОС;
- 3) макета шлюза сбора и первичной обработки данных.

5.3 Аппаратное обеспечение

5.3.1 Сервер для работы подсистемы ЦОД

Согласно техническому заданию процессор для подсистемы ЦОД должен быть не хуже Intel Xeon 5600/5500. Поэтому, в соответствии с таблицей 5.1, были выбраны три процессора того же семейства, характеристики,

которых не хуже требуемого. Далее на основе тестов компании PassMark Software мы видим, что процессор Intel Xeon E3-1280 является подходящим вариантом для подсистемы ЦОД.

Таблица 5.1 — Таблица сравнения процессоров Intel Xeon

Процессор	Тест PassMark (больше - лучше)	Рейтинг (меньше - лучше)	Соотношение цена/качество (больше - лучше)
Intel Xeon E3-1245 v3, 3.40GHz	9151	50	NA
Intel Xeon X5690, 3.47GHz	9258	47	5.36
Intel Xeon E3-1280 V2, 3.60GHz	9843	31	14.91

Процессор Intel Xeon E3-1280V2 Quad-Core, 3.60ГГц, LGA1155, 5.0 GT/s, 8М, OEM - четырехъядерный процессор, для специализированного компьютера. Данный процессор обеспечивает требуемую производительность, необходимую для работы подсистемы ЦОД. Процессор выполнен с соблюдением норм 22 нанометрового технического процесса.

Основные характеристики процессора, в соответствии с таблицей 5.2: тактовая частота 3.6 ГГц, кэш-память общим объемом 8 МБ, ориентация под платформу LGA1155 и требование к подсистеме питания процессора PCG 05A. Системная шина, на которую рассчитан процессор и её частота довольно велика – 5,00 GT/s. Процессор изготовлен по технологии Intel Viiv и Intel vPro и поддерживает 64-разрядную архитектуру ОС.

Выбор материнской платы зависит от различных характеристик выбранного процессора, такие как поддерживаемые чипсеты, сокет и т.д. На основе этих характеристик была выбрана материнская плата Intel S1200BTSR Beartooth, Socket-LGA1155, iC202, 4xDDR3, 6xSATA, Video, 2xGbLAN, mATX – серверная системная плата Intel, в соответствии с таблицей 5.3.

Таблица 5.2 — Спецификация процессора

Наименование	Описание
Характеристики	Процессор Ivy Bridge-H2 Quad Core 3.6GHz (LGA1155, 8MB, DMI, 69Вт, 22nm) Tray Xeon E3-1280 V2
Основные характеристики	
Частота процессора	3.6 ГГц
Модельный ряд	Intel Xeon
Модель	E3-1280V2
Ядро	Ivy Bridge-H2
Количество ядер	4
Характеристики поддерживаемой памяти	
Макс. объем памяти	32 ГБ
Тип памяти	DDR3

На основе поддерживаемых характеристик процессора и материнской платы для подсистемы ЦОД была выбрана оперативная память Kingston, PC3-10667 1333MHz DDR3 ECC CL9 1Rx8 DIMM, KVR13LE9S8/4 - модуль небуферизированной памяти типа DDR3, объёмом 4 ГБ, с частотой 1333 МГц и пропускной способностью 10600Мб/сек. Данная оперативная память является высококачественным вариантом для подсистемы ЦОД.

Так же на основе поддерживаемых характеристик материнской платы и требований ТЗ для подсистемы ЦОД был выбран жесткий диск Samsung Barracuda SpinPoint F3 ST1000DM005 HD103SJ. Этот диск из линейки SpinPoint F3 емкостью 1 терабайт с интерфейсом SATA 3.0 и скоростью вращения шпинделя 7200об./мин. Данный диск является оптимальным вариантом с его комбинацией надёжности и высокой производительности. Данный жесткий диск подходит для мощной производительной системы, а также может использоваться для сетевых хранилищ данных и для файловых серверов.

В соответствии с таблицей 5.4, все требования технического задания, на аппаратную часть подсистем ЦОД были соблюдены.

Таблица 5.3 — Спецификация материнской платы

Наименование	Описание
Тип разъема процессора	Socket 1155
Кол-во процессоров	1
Поддерживаемые процессоры	Intel Xeon E3-1200 series, Intel Core i3-2100 series
Форм фактор	Micro ATX
Чипсет	Intel C202
Тип модулей памяти	DDR3 ECC UDIMM
Макс. объем памяти	32 ГБ

5.3.2 Компьютер для работы подсистемы СОС

Требования к техническим характеристикам компьютера для работы подсистемы СОС приводятся в пункте 4.3.4 ТЗ. В соответствии с данными требованиями для сравнения были выбраны 3 компьютера с характеристиками не хуже требуемых: Dell XPS 8700 8700-7320, HP ProDesk490 G3 и DX-D-6434006. Их сравнение представлено в таблице 5.5.

На основе характеристик процессора и соотношения цена/характеристики для подсистемы ЦОД был выбран компьютер DX-D-6434006, как наиболее оптимальный вариант. Данный компьютер соответствует всем требованиям ТЗ, изложенным в пункте 4.3.4.

5.3.3 Макет шлюза сбора и первичной обработки данных

Разработка макета шлюза сбора и первичной обработки данных описана в главе 1 данного отчета.

5.4 Программное обеспечение

Разработка программного обеспечения программно-аппаратной платформы описана в главе 4 данного отчета.

Таблица 5.4 — Сравнение характеристик аппаратного обеспечения с требованиями ТЗ

Наименование	Требования к серверу ЭО ПАП по ТЗ	Реализация в ЭО ПАП	
		кол-во	название
ОЗУ ГБ/СРУ	не менее 2	4	Kingston, PC3-10667 1333MHz DDR3 ECC CL9 1Rx8 DIMM, KVR13LE9S8/4
дисковое пространство ГБ/СРУ	не менее 20	1	HDD 1 Tb Samsung Barracuda SpinPoint F3 ST1000DM005 HD103SJ 3.5"7200rpm 32Mb
модель процессора	не хуже Intel Xeon 5600/5500	1	Intel Xeon E3-1280V2 Quad-Core, 3.60ГГц, LGA1155, 5.0 GT/s, 8M, OEM
материнская плата	нет требований	1	Intel S1200BTSR Beartooth, Socket-LGA1155, iC202, 4xDDR3, 6xSATA, Video, 2xGbLAN, mATX

5.5 Выводы

В данном разделе отчета описана масштабируемая сервис-ориентированная программно-аппаратной платформа для сбора, нормализации, обработки и визуализации больших массивов гетерогенных данных распределенной сети электронных потребительских устройств. В соответствии с пунктом 4.3.1 Технического задания, ЭО ПАП состоит из:

- а) программного обеспечения;
- б) аппаратного обеспечения в составе:
 - 1) сервера, предназначенного для работы подсистемы ЦОД;

Таблица 5.5 — Сравнение характеристик выбранных компьютеров

Наименование характеристики	Dell XPS 8700 8700-7320	HP ProDesk490 G3	DX-D-6434006
Процессор	Intel Core i7 4790	Intel Core i7 6700	Intel Core i7 4790K
Тест процессора PassMark (больше - лучше)	10003	9994	11192
Тип ОЗУ	DDR3	DDR4	DDR3
Размер ОЗУ	16ГБ	16ГБ	16ГБ
Объем дискового пространства	1ГБ	1ГБ	1ГБ
Тип жесткого диска	HDD	HDD	HDD
Кол-во USB портов	10	8	6
Тип и модель видеокарты	дискретная Nvidia GeForce GTX745	встроенная Intel HD Graphics 530	встроенная Intel HD Graphics
Соотношение цены/характеристик (больше - лучше)	0.8	0.9	0.95

- 2) персонального компьютера, предназначенного для работы подсистемы СОС;
- 3) макета шлюза сбора и первичной обработки данных.

В результаты данной работы была разработана эскизная конструкторская документация в соответствии с пунктами 6.1.4 и 6.2 ТЗ, представленная в Приложениях Р–Т. Эскизная конструкторская документация представлена в следующем составе:

- a) схемы структурной в соответствии с ГОСТ 2.701-84,
- б) инструкции по эксплуатации,

в) формуляра в соответствии с ГОСТ 2.601-2006 и ГОСТ 2.610-2006.

По данной технической документации Индустриальным партнером был разработан экспериментальный образец ПАП. Данный экспериментальный образец был использован для проведения экспериментальных исследований, описанных в главах 7 и 8, в соответствии с программой и методиками, описанными в главе 6.

6 Разработка программы и методики экспериментальных исследований ЭО ПАП

6.1 Программа экспериментальных исследований ЭО ПАП

Для проведения экспериментальных исследований ЭО ПАП была разработана программа, учитывающая предъявляемые требования технического задания. Программа экспериментальных исследований в том числе включают в себя проверки с использованием программно-аппаратного стенда для симуляции распределенной сети электронных потребительских устройств и экспериментального образца внешней информационной системы, предназначеннай для тестирования интерфейса прикладного программирования для доступа к данным электронных устройств. Целью данных проверок является определить соответствие разработанных систем требованиям технического задания.

Всего программа включает 11 проверок:

- а) проверка технической документации на соответствие установленной комплектности и оценка её качества (пункты требований ТЗ 6.1.3.1, 6.1.3.2, 6.1.3.4, 6.1.5, 6.1.9),
- б) проверка соответствия Объекта исследований его эскизной конструкторской и программной документациям,
- в) исследование поддержки протокола TCP/IP (пункт требований ТЗ 4.2.1),
- г) исследование времени построения графических форм визуализации конфигурации сети устройств (пункт требований ТЗ 4.2.4.1),
- д) исследование времени отклика операций навигации интерактивных графических форм (пункт требований ТЗ 4.2.4.2),
- е) исследование времени отклика интерактивных графических форм представления данных устройств (пункт требований ТЗ 4.2.4.3),
- ж) исследование поддержки синхронных и асинхронных режимов передачи данных (пункт требований ТЗ 4.2.2),
- з) исследование хранения данных (пункт требований ТЗ 4.2.3),

- и) исследование передачи данных в реальном времени (пункт требований ТЗ 4.2.5.1),
- к) исследование предоставления доступа к данным (пункт требований ТЗ 4.2.5.2),
- л) исследование симуляции распределенной сети потребительских устройств (пункт требований ТЗ 4.3.7.3).

6.2 Методика проверки технической документации на соответствие установленной комплектности и оценки её качества

Проверка выполняется следующим образом. Проверяется соответствие эскизной конструкторской и программной документации на объект исследований комплектности и её качества - требованиям приведенным в Техническом задании для каждого пункта документа ГОСТа.

Комплект документации считается выдержавшим исследование, если его комплектность и качество соответствует требованиям приведенным в Техническом задании для каждого пункта ГОСТа

6.3 Методика проверки соответствия объекта исследований его эскизной конструкторской и программной документации

Проверка выполняется следующим образом. Проверяется соответствие объекта исследований его эскизной конструкторской и программной документации. Объект исследований считается выдержавшим проверку, если он соответствует каждому документу комплекта документации

6.4 Методика проверки соответствия показателей значением заданным техническим заданием

6.4.1 Состав методик выполнения проверок

Методики выполнения проверок:

- а) исследование поддержки протокола TCP/IP,

- б) исследование времени построения графических форм визуализации конфигурации сети устройств,
- в) исследование времени отклика операций навигации интерактивных графических форм,
- г) исследование времени отклика интерактивных графических форм представления данных устройств

рассмотрены в разделе 6.4.2

Методики выполнения проверок:

- а) исследование хранения данных,
- б) исследование передачи данных в реальном времени,
- в) исследование симуляции распределенной сети потребительских устройств

рассмотрены в разделе 6.4.3

Проверка на исследование поддержки синхронных и асинхронных режимов передачи данных рассмотрена в разделе 8.2, а проверка на исследование предоставления доступа к данным рассмотрена в разделе 8.3

6.4.2 Проверка показателей работы интерактивной визуализации ЭО ПАП

Для выполнения проверок данного раздела необходимо запустить ЦОД и ПАС путем выполнения следующих команд

```
cd ~/semiot-platform/  
docker-compose pull  
docker-compose up -d
```

После запуска ЦОД и ПАС нужно открыть на клиентском ПК браузер и включить в нем режим разработчика. Перейти внутри появившегося окна во вкладку сеть. Внутри этой вкладки будет отображаться время отклика интерактивных графических форм. Время отклика должно соответствовать соответствующим значениям указанных под соответствующими пунктами в ТЗ. Для замера времени отклика необходимо перейти на любую страницу внутри домена ЦОД, используя операции навигации по интерактивным гра-

фическим формам, и посмотреть занятое время для этой страницы внутри данной вкладки. Соответственно, если

- а) какая-либо страница интерактивных графических форм не будет отображена во время переходов внутри домена ЦОД путем использования операций навигации по интерактивным графическим формам, то проверка на исследование поддержки протокола TCP/IP считается не пройденной, во всех остальных случаях пройденной,
- б) время отклика какой-либо страницы интерактивных графических форм за исключением страницы визуализации конфигурации сети устройств и страницы представления данных электронных потребительских устройств составляет более 5 секунд, то проверка на исследование времени отклика операций навигации интерактивных графических форм считается не пройденной, во всех остальных случаях пройденной,
- в) время построения визуальных форм для графического представления данных электронных потребительских устройств при открытии страницы выбранного устройства не превышает 3 секунд, то проверка на исследование времени отклика интерактивных графических форм представления данных устройств считается пройденной, иначе не пройденной,
- г) время отклика страницы конфигурации сети пользовательских устройств менее 15 секунд, то проверка на исследование времени построения графических форм визуализации конфигурации сети устройств считается пройденной, иначе не пройденной.

Результаты для следующих проверок приведены в разделе 7.

6.4.3 Проверка работоспособности ЭО ПАП при хранении большого объема данных

Для выполнения проверок данного раздела необходимо запустить ЦОД и ПАС путем выполнения следующих команд

```
cd ~/semiot-platform/  
docker-compose pull  
docker-compose up -d
```

Также необходимо изменить конфигурационный файл для ЦОД путем замены количества симулируемых устройств с 1000 до 3000. Дождавшись запуска ЦОД необходимо выполнить следующие действия для анализа успешного или неуспешного прохождения проверок:

Перейти на страницу запущенного устройства. В появившейся визуальной форме для графического представления данных электронных потребительских устройств, должны отобразится текущие показания, которые были получены для этого устройства ЦОДом от ПАС. Если новое показание не придет (не будет добавлено на визуальную форму) в течение 15 минут, то проверки на исследование симуляции распределенной сети потребительских устройств и на исследование передачи данных в реальном времени считаются не пройденными, иначе пройденными.

Так как в пункте 4.2.3 ТЗ описано, что ЦОД должен обеспечивать возможность хранения данных от 1000 устройств, передающих данные раз в 15 минут на протяжении года, то количество таких показаний в ЦОД к концу года должно составить $1000 * 4 * 24 * 365 = 35,040,000$. Так как ПАС запущен с параметрами симуляции 3000 электронных потребительских устройств и каждое устройство передает данные раз в 30 секунд, то необходимое количество дней для замера составит $35,040,000 \div (3000 * 2 * 60 * 24) = 4.055$ дней. Следовательно, если через 4.055 дней не останавливая ЦОД, возможно зайти на страницу температурного устройства в ЦОД и будет представлена визуальная форма для графического представления данных электронных потребительских устройств в которой можно будет просмотреть показания устройства за этот промежуток времени, то проверка на исследование хранения данных считается пройденной, иначе пройденной.

6.5 Выводы

В данном разделе отчета описана программа и методики экспериментальных исследований экспериментальноо образца масштабируемой сервис-ориентированной программно-аппаратной платформы для сбора, нормализации, обработки и визуализации больших массивов гетерогенных данных распределенной сети электронных потребительских устройств. Результате данной

работы был разработан документ «Программа и методики экспериментальных исследований ЭО ПАП 1000.000000.000ПМ», который идет в дополнении к данному отчету в виде отдельного документа с приложениями.

В разработанную программу экспериментальных исследований включены проверки разработанной эскизной конструкторской и программной документации в соответствии с пунктом 6.1.3 ТЗ и проверки соответствия характеристик ЭО ПАП пунктам 4.3.1, 4.3.2 и 4.3.5 ТЗ.

7 Экспериментальные исследования ЭО ПАП с использованием ПАС

7.1 Проверка показателей работы интерактивной визуализации ЭО ПАП

При проведении экспериментального исследования ЭО ПАП с использованием ПАС были использованы следующие сценарии:

- проверка времени отклика интерактивных графических форм;
- проверка работоспособности ЭО ПАП с 1000 одновременно подключенных электронных пользовательских устройств.

Для проведения экспериментальных исследований был развернут ЭО ПАП и запущен ПАС с симуляцией 1000 устройств. В качестве электронных потребительских устройств используются общедомовые счетчики тепла. Проведены десять экспериментов для каждого из сценариев, при этом в первом случае оценивалось среднее время сбора данных для построения графических форм, среднее время отклика системы для операций навигации и среднее время построения визуальных форм для представления данных. В результате исследований выбран результат эксперимента наиболее близкий к среднему значению.

Результаты десяти перечисленных экспериментов для 1000 одновременно подключенных счетчиков представлены на рисунках 7.1, 7.2 и 7.3. Соответственно среднее время сбора данных для построения графических форм равно 816 мс, среднее время отклика системы для операций навигации равно 51 мс, а среднее время построения визуальных форм для представления данных равно 178 мс.

7.2 Проверка работоспособности ЭО ПАП при хранении большого объема данных

Для данного исследования осуществлялась проверка на работоспособность после записи в ЦОД такого количества данных, которое эквивалентно передачи от одного устройства показаний раз в 15 минут и хранения этих

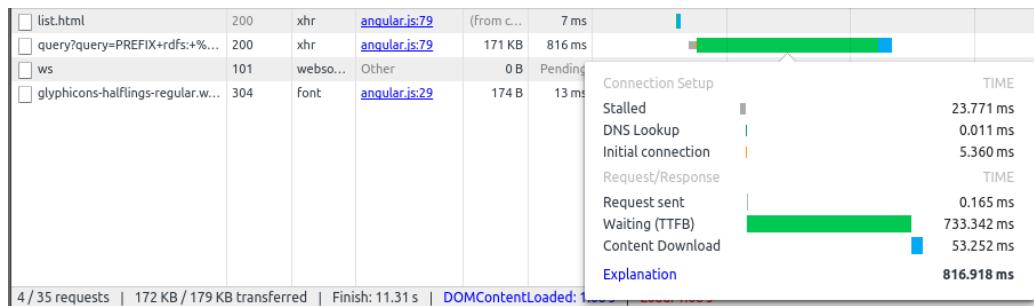


Рисунок 7.1 – Среднее время сбора данных для построения графических форм (1000 счетчиков)

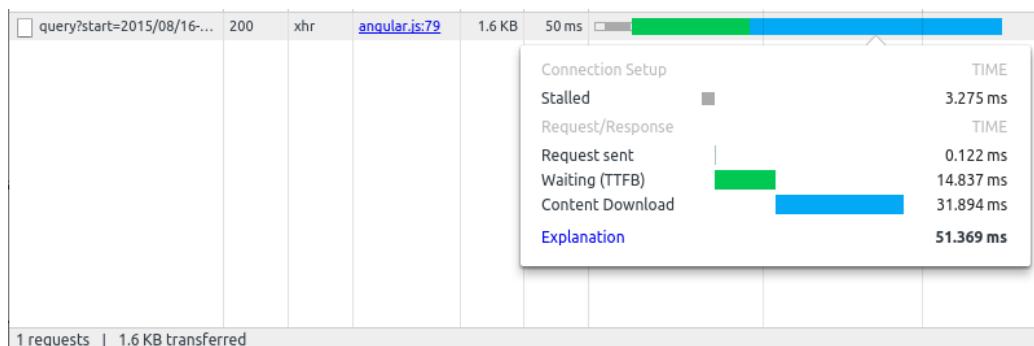


Рисунок 7.2 – Среднее время отклика системы для операций навигации (1000 счетчиков)

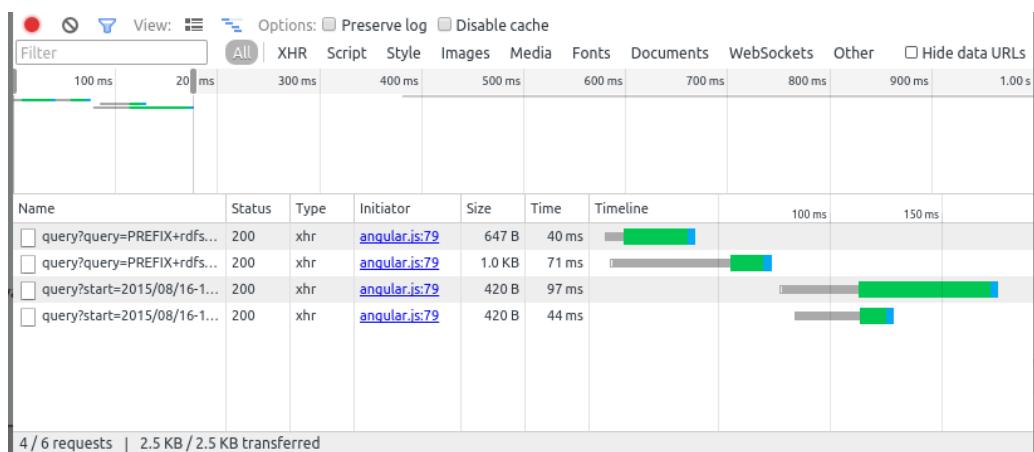


Рисунок 7.3 – Среднее время построения визуальных форм для представления данных (1000 счетчиков)

показаний для 1000 устройств в течении года. Таким образом количество записей, хранимых в ЦОД, должно составить порядка 35 миллионов. Поэтому для этого случая был развернут ЭО ПАП и запущен ПАС в режиме симуляции 3000 устройств, которые передавали данные раз в 30 секунд. Соответственно, для такого режима необходимое количество данных, хранящихся в

ЦОД, накапливалось за 4 дня работы ЭО ПАП. После получения такого количества данных, была проверена работоспособность подсистемы ЦОД ЭО ПАП путём открытия страницы построения визуальных форм для представления данных устройства и наблюдения, что в системе есть данные за данный промежуток времени.

Также была осуществлена проверка, что данные поступают в режиме реального времени путём выполнения действий, описанных в разделе 6.4.3. Проверка симуляции сети электронных потребительских устройств также прошла успешно.

7.3 Выводы

В данном разделе отчета описаны экспериментальные исследования ЭО ПАП с использованием программно-аппаратного стенда, разработанного на 1-м этапе данного ПНИ. Данные исследования включали следующие проверки:

- а) проверка времени отклика интерактивных графических формы,
- б) проверка работоспособности ЭО ПАП с 1000 одновременно подключенных электронных потребительских устройств.

В рамках первой проверки было установлено, что среднее время сбора данных для построения графических форм составляет 816 мс, среднее время отклика системы для операций навигации равно 51 мс, а среднее время построения визуальных форм для представления данных равно 178 мс. Данные показатели соответствуют требованиям пункта 4.2.4 Технического задания.

В рамках второй проверки было установлено, что ЭО ПАП работает корректно при одновременно подключенных 1000 электронных потребительских устройств, а также поддерживает синхронный и асинхронный режимы работы. Данные показатели соответствуют требованиям пункта 4.2.3 Технического задания.

Таким образом экспериментальными исследованиями ЭО ПАП с использованием ПАС было установлено соответствие ЭО ПАП требованиям Технического задания.

8 Экспериментальные исследования ЭО ПАП с использованием экспериментального образца внешней информационной системы, получающей доступ к гетерогенным данным распределенной сети электронных потребительских устройств через интерфейс прикладного программирования ЭО ПО ПАП

8.1 Состав проверок с использованием экспериментального образца внешней информационной системы

При проведении экспериментального исследования ЭО ПАП с использованием экспериментального образца внешней информационной системы (ЭО ВИС) были разработаны следующие сценарии:

- а) проверка поддержки синхронных и асинхронных режимов передачи данных;
- б) проверка времени доступа к показаниям;
- в) проверка времени исполнения команд для процессов.

8.2 Проверка поддержки синхронных и асинхронных режимов передачи данных

Для данного сценария исследования достаточно запустить ЭО ВИС. При запуске данная система создает пул из 50 клиентов асинхронно пытающихся получить данные с разных устройств. Таким образом, если все запросы к устройствам, а следовательно, и запуск ЭО ВИС, пройдут успешно, то данный сценарий можно признать успешно протестированным.

8.3 Проверка времени доступа к показаниям

Для этого сценария ЭО ВИС в процессе своей работы создает сравнительные записи между временем появления показания и временем его получения в системе. Данная разница отображает время, затраченное каждым показанием для записи в ЦОД и транспортировке на дальнейшее ПО, обрабатывающее эти показания. Измерения проводились на 3000 устройств генерирующих данные каждые 30 секунд. При этом был выбран временной про-

межуток в 5 минут для подсчета максимального, среднего и минимального времени обработки показаний. В среднем в каждом таком промежутке содержится по 25-30 тысяч показаний. Минимальное значение обработки показаний было 12 мс для всех случаев, поэтому эти показания не были вынесены на отдельный график. На рисунке 8.1 отображено среднее время обработки показаний. Как из него можно заметить, максимальное время для средних значений не превышает 300 мс, а большая часть показаний обрабатывается меньше чем за 20 мс. Соответственно, если такая работа проведена успешна, то проверка на исследование предоставления доступа к данным пройдена успешно, иначе провалена.

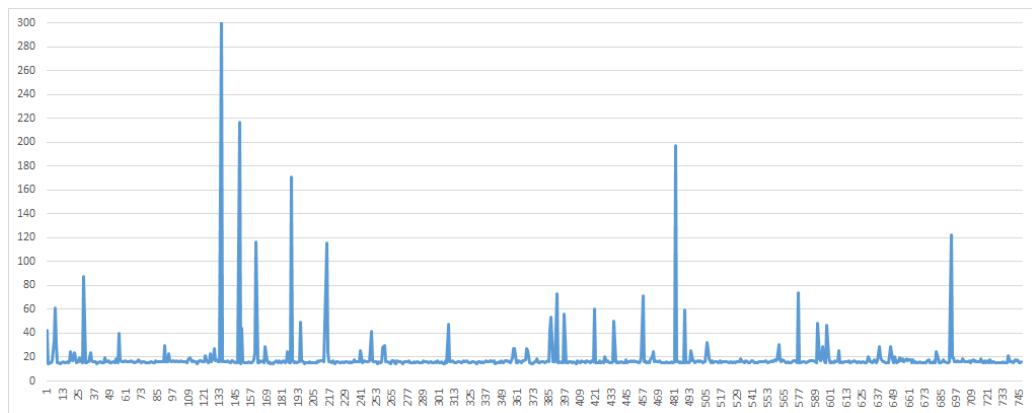


Рисунок 8.1 — Среднее время обработки показаний (в миллисекундах)

На рисунке 8.2 отображено максимальное время обработки показаний. Как можно судить по этим данным, максимальное время не превышает 10 секунд для одного показания среди всех, при этом максимальное время большей части показаний не превышает 1 секунды.

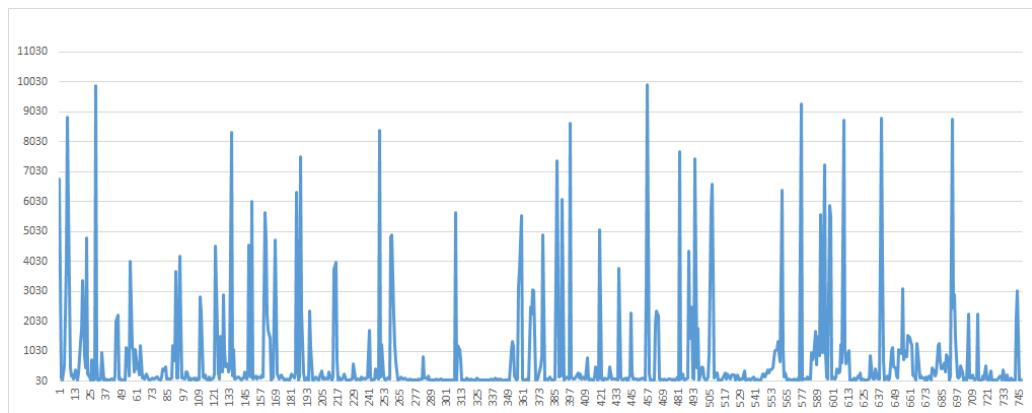


Рисунок 8.2 — Максимальное время обработки показаний (в миллисекундах)

8.4 Проверка времени исполнения команд для процессов

Для данного сценария исследования была просимулирована ситуация, когда показания с устройств зависят от какого регулятора, меняя значение которого соответственно изменялись показания с устройств. Было опробовано симулирование ситуации, где в качестве устройств выступали температурные датчики, и ЭО ВИС необходимо было поддерживать для каждой группы датчиков температуры значения не превышающие 24 градусов по Цельсию и не менее 18 градусов. При таких условиях значение для каждого регулятора менялись на 2 единицы в пределах диапазона от 0 до 100 в зависимости от того была ли средняя температура для каждой из больше или меньше пороговых значений. Команды отправлялись каждые 30 секунд до достижения каждой из групп датчиков оптимальных температур. Для статистики брались данные о времени выполнении команды (максимальное, среднее и минимальное) за каждые 5 минут. В среднем в каждом таком промежутке находится по 20 команд. На рисунке 8.3 красным отмечено максимальное время выполнения, синим - среднее, а черным - минимальное. Как можно увидеть из графика, максимальное время не превышает 14 секунд, при этом в среднем максимальное время не превышает 6 секунд. Максимальное значение среди средних показаний времени исполнения команд составляет 6 секунд и в среднем не превышает 2 секунд, а минимальное время колеблется около 0,5-1 секунд.

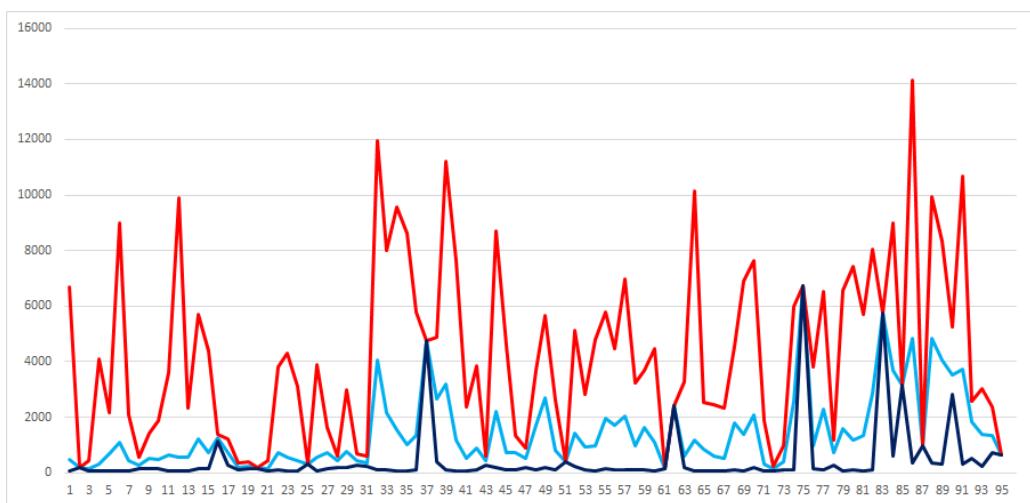


Рисунок 8.3 — Время обработки команд (в миллисекундах)

8.5 Выводы

В данном разделе отчета описаны экспериментальные исследования ЭО ПАП с использованием экспериментального образца внешней информационной системы, получающей доступ к гетерогенным данным распределенной сети электронных потребительских устройств через интерфейс прикладного программирования ЭО ПО ПАП. Данные исследования включали следующие проверки:

- а) проверка поддержки синхронных и асинхронных режимов передачи данных,
- б) проверка времени доступа к показаниям,
- в) проверка времени исполнения команд для процессов.

В рамках первой проверки было установлено, что ЭО ПАП поддерживает оба режима передачи данных. Данное факт подтверждает соответствие требованию пункта 4.2.2 Технического задания.

В рамках второй проверки было установлено, что максимальное время доступа к показаниям равно 300 мс для 3000 одновременно подключенных электронных устройств, и среднее время доступа равно 20мс.

В рамках третьей проверки было установлено, что среднее время исполнения команд с 3000 одновременно подключенных электронных устройств соответствует 2 секундам.

Таким образом экспериментальными исследованиями ЭО ПАП с использованием ЭО ВИС было установлено соответствие ЭО ПАП требованиям Технического задания.

9 Изготовление макета шлюза сбора и первичной обработки данных (МШСПОИ)

9.1 Состав работ по изготовлению МШСПОИ

В рамках данной работы выполняется изготовление МШСПОИ на основе результатов работы, описанной в главе 1. Данный этап включает в себя следующие работы:

- а) подготовка и сборка аппаратной части МШСПОИ,
- б) разработка исполняемого файла и развертывание на МШСПОИ.

Выполнение перечисленных выше работ необходимо для изготовления МШСПОИ для проведения его лабораторных испытаний, описанных в главе 3.

9.2 Подготовка и сборка аппаратной части МШСПОИ

Сборка аппаратной части МШСПОИ подразумевает подготовку составных компонентов МШСПОИ, а также их компоновку в устройство в едином корпусе.

Данная работа предполагает выделение следующих составных элементов МШСПОИ, описанных на этапе описания разработки:

- а) Основной вычислительный блок с модулем беспроводной связи на основе чипа Espressif ESP8266¹
- б) интерфейсы приема и передачи цифровых и аналоговых данных
- в) компоненты ввода и вывода информации
- г) вспомогательный блок для единовременного программирования МШСПОИ
- д) блок для начальной конфигурации МШСПОИ

Совместно с блоком программирования система на чипе ESP8266 от Espressif, которая представляет основной вычислительный блок с модулем

¹Официальный сайт Espressif: <https://espressif.com/>

беспроводной связи, позволяет использовать следующую функциональность для сторонних сенсоров и актуаторов:

- 10 цифровых входов-выходов, каждый из которых поддерживает аппаратные прерывания, широтно-импульсную модуляцию, протоколы передачи данных I2C и OneWire.
- 1 аналогово-цифровой преобразователь, допускающий максимальное напряжение 3.2В
- поддержку развертывания исполняемых файлов, созданных на основе набора библиотек Arduino
- автоматическая загрузка обновлений прошивки, в том числе пошине USB

В качестве компонентов ввода и вывода информации были подготовлены тестовые блоки на основе цифровых и импульсных счетчиков потребления электрической энергии и расхода воды, система реле и модули индикации.

Для получения информации и управления необходимо выполнить следующие примеры СоАР-запросов, обрабатывающие данные в формате JSON-LD и проверить результаты с помощью аналогичных GET-запросов:

```
PUT "coap://device_ip/led1": {"@context":"/led1/context","pwm-value":"100"}  
PUT "coap://device_ip/tick1": {"@context":"/tick1/context","tick-value":"3"}
```

Для блока начальной конфигурации задействовано 3 входа общего назначения. Блок для начальной конфигурации позволяет установить параметры инициализации каждого из компонентов ввода и вывода информации, а также основного вычислительного блока: таких, например, как параметры внешней сети, к которой необходимо осуществлять подключение. 2 светодиода используются для отображения состояния подключения к внешней публичной сети и состояния режима конфигурирования.

В разработанном решении аппаратный переключатель позволяет перейти вычислительному блоку в режим конфигурирования, при котором вычислительный блок автоматически переконфигурирует модуль беспроводной связи в режим программной точки доступа с временными фиксированными параметрами подключения, зависящими от серийного номера устройства.

Изменения состояний ресурсов и параметров конфигурации фиксируются в энергонезависимой памяти EEPROM, доступной как внутренний ресурс системы.

Для формирования параметров конфигурации необходимо отправить следующий пример COAP-запроса, обрабатывающий данные в формате JSON-LD, находясь в режиме конфигурирования и проверить результаты с помощью аналогичных GET-запросов:

```
PUT "coap://device_ip/config":  
{@context ":"/config/context","wifi-name":"ISST","wifi-password":  
"SSIDpassword"}
```

Вспомогательный блок единовременного конфигурирования позволяет программировать основной чип, записывая на него исполняемый файл при развертывании МШСПОИ. Он построен на основе универсального внутрисхемного программатора, переводящего сигналы ПК по шине USB в стандарты последовательного порта.

9.3 Разработка исполняемого файла и развертывание на МШСПОИ

Загрузка исполняемого файла для развертывания МШСПОИ производится на основе вспомогательного программатора. Для работы с ним используется инструментарий ESP8266 от разработчиков чипа Espressif¹. С помощью построенного на основе инструментария и набора библиотек Arduino, подготовленные исходные коды логики МШСПОИ компилируются в бинарный исполняемый файл для архитектуры чипа основного вычислительного блока МШСПОИ по инструкциям, доступным в репозитории проекта²

```
git clone --recursive  
https://github.com/semitoproject/semitot-device-prototype.git  
cd semiot-device-prototype  
cd esp8266/tools  
python get.py  
cd ../..  
sh semiot-device.sh
```

¹Портал разработчиков Espressif: <http://bbs.espressif.com/>

²Репозиторий проекта: <https://github.com/semitoproject/semitot-device-prototype/>

Посредством драйвера для программатора и Python-скрипта для загрузки обновления прошивки из набора библиотек Arduino для ESP8266 espota выполняется отправка данных на устройство¹. Для автоматизации процедуры развертывания подготовлены bash-скрипты и makefile-инструкции для утилиты make, позволяющие существенно ускорить и упростить процедуру настройки среды кросс-компиляции исполняемого файла для чипа ESP8266.

9.4 Выводы

В данном разделе отчета описано изготовление МШСПОИ проведенного Индустриальным партнером. Данная работа заключается в подготовке и сборке аппаратной части МШСПОИ, и разработке исполняемого файла и его развертывании на аппаратной части.

Все работы проведены в соответствии с Техническим заданием, а именно пунктом 4.3. А так же в соответствии с эскизной конструкторской и программной документацией МШСПОИ, представленной в данном отчете о ПНИ.

Для подтверждения факта изготовления МШСПОИ была сформирована комиссия и составлен акт изготовления от 27-го июня 2016 года, который прилагается к данному отчету о ПНИ.

¹Репозитории ESP8266 Arduino на портале Github: <https://github.com/esp8266/Arduino>

10 Изготовление экспериментального образца масштабируемой сервис-ориентированной программно-аппаратной платформы для сбора, нормализации, обработки и визуализации больших массивов гетерогенных (структурных, полуструктурных и неструктурных) первичных исследовательских данных в распределенной сети электронных потребительских устройств (Internet of Things)

10.1 Состав работ по изготовлению ЭО ПАП

В рамках данной работы выполняется изготовление ЭО ПАП на основе результатов работ, описанных в главах 4 и 5. Данный этап включает в себя следующие работы:

- а) разработка исполняемых файлов программного обеспечения ЭО ПАП,
- б) установка и конфигурирование операционной системы,
- в) установка и конфигурирование необходимого программного обеспечения,
- г) установка и конфигурирование программного обеспечения ЭО ПАП.

Выполнение перечисленных выше работ необходимо для изготовления и развертывания ЭО ПАП для проведения его экспериментальных исследований, описанных в главах 7 и 8.

10.2 Разработка исполняемых файлов программного обеспечения ЭО ПАП

В рамках данной работы необходимо разработать исполняемые файлы для каждой из подсистем ЭО ПО ПАП.

Подсистемы СОС и ПЭ МШСПОИ разработаны на языке программирования Java 8-й версии, соответственно необходимо обеспечить их исполнение в среде Java Virtual Machine. Для этих целей была использована система

сборки Maven¹, которая может выполнять компилирование, сборку и развертывание программного обеспечения.

Исходный код подсистем был организован в Java-пакеты, в соответствии с логической структурой классов подсистемы. Имена пакетов начинаются на `ru.semiot`, в соответствии с доменом проекта.

На следующем шагу были подобраны внешние библиотеки необходимые для сборки исходного кода, в соответствии с их последними версиями и типами лицензий. Так как в рамках данного ПНИ непредусмотрено использование программных библиотек с закрытым исходным кодом, то отбирались только те библиотеки, лицензии которых разрешают их свободное использование в любых целях. На основе списка библиотек был создана конфигурация Maven-проекта.

Так как подсистемы являются конфигурируемыми, то был разработан модуль конфигурирования, позволяющих считывать конфигурацию из файла и обновлять её в случае изменения. Файл конфигурации имеет формат `.properties`.

В результате были получены файлы формата `.jar`, которые являются исполняемыми и могут быть запущены с помощью следующей команды: `java -jar <имя файла>`.

Исполняемые файлы были получены для подсистем СОС, ПЭ МШС-ПОИ, а так же для модулей подсистемы ЦОД. Подсистема ЦОД была разбита на 3 исполняемых файла:

- а) 1-й файл включает в себя модули МСД, МАД, МНАД и МБЗ,
- б) 2-й файл включает в себя только модуль МБД,
- в) 3-й файл включает в себя модуль МИВДТ.

В отличии от других подсистем ЭО ПАП подсистема ЦОД состоит из нескольких самостоятельных модулей, поэтому для её развертывания была применена контейнерная система Docker². Система Docker позволяет создавать обособленные окружения для каждого из исполняемых файлов, что упрощает их последующее развертывание.

¹См. <http://maven.apache.org/>

²См. <http://docker.com/>

На базе системы Docker для подсистемы ЦОД были созданы следующие контейнеры:

- а) контейнер, включающий программное обеспечение Crossbar.io¹, которое служит брокером, реализующим протокол Web Application Messaging Protocol²;
- б) контейнер, включающий исполняемый файл с модулями МСД, МАД, МНАД и МБЗ;
- в) контейнер, включающий базу данных Apache Cassandra³, которая используется модулем МБД для хранения данных электронных устройств;
- г) контейнер, включающий исполняемый файл с модулем МБД;
- д) контейнер, включающий базу данных Blazegraph⁴, которая используется модулем МБЗ для хранения метаданных электронных устройств;
- е) контейнер, включающий сервер приложений Wildfly и модуль МИВДТ;
- ж) контейнер, включающий веб-сервер NGINX, который ограничивает доступ к модулям подсистемы ЦОД;
- з) контейнер, включающий базу данных MySQL для хранения учетных записей пользователей и других служебных данных.

Подсистема ЦОД может быть развернута как на одной машине, так и на нескольких. В первом случае используется система Docker Compose⁵, а во втором случае система Docker Swarm⁶.

¹ См. <http://crossbar.io/>

² См. <http://wamp-proto.org/>

³ См. <http://cassandra.apache.org/>

⁴ См. <https://www.blazegraph.com>

⁵ См. <https://docs.docker.com/compose/overview/>

⁶ См. <https://docs.docker.com/swarm/overview/>

10.3 Установка и конфигурирование операционной системы и необходимого программного обеспечения

Как описано в главе 5 для работы ЭО ПАП необходимо использовать компьютер и сервер для работы всех его подсистем.

Сервер используется для развертывания нескольких виртуальных машин, когда как компьютер используется только для работы подсистемы СОС. Но для того и другого установка и конфигурирование ОС было выполнено одинаково.

В качестве операционной системы виртуальных машин сервера использовалась Ubuntu Server 16.04 LTS, а для компьютера операционная система того же семейства, но с графической оболочкой, Ubuntu Desktop 16.04.

Кроме стандартной процедуры установки операционной системы и её первоначальной настройки, так же были произведены следующие настройки:

- а) увеличение лимита файловых дескрипторов на процесс с 1024 до 10000,
- б) увеличение лимита потоков на процесс с 1024 до 4096.

Увеличение данных лимитов необходимо для стабильной работы ЭО ПАП с большим количеством одновременно подключенных электронных потребительских устройств.

На компьютер подсистемы СОС была установлена среда Java Virtual Machine версии 8, необходима для запуска исполняемого файла.

Для работы ПЭ МШСПОИ была создана виртуальная машина и так же установлена среда Java Virtual Machine версии 8.

Для работы подсистемы ЦОД было создано 4 виртуальные машины, на каждой из которых была установлена система Docker и на одной из них система Docker Swarm для создания кластера.

10.4 Установка и конфигурирование программного обеспечения ЭО ПАП

В рамках данной работы была выполнена установка и конфигурирование всех подсистем программного обеспечения ЭО ПАП.

Для установки подсистемы СОС и ПЭ МШСПОИ были загружены исполняемые файлы в формате .jar, которые при запуске считывают конфигурацию из файла `/semiot-platform/config.properties`. Данный конфигурационный файл позволяет, в случае с подсистемой СОС, установить параметры симуляции, а в случае ПЭ МШСПОИ, параметры подключения к подсистеме СОС.

Установка подсистемы ЦОД отличается от других подсистем и заключается в загрузке Docker-контейнеров на виртуальные машины, записи конфигурации в файл `/semiot-platform/config.properties`, запуска Docker-контейнеров, загрузке и конфигурировании драйверов устройств. Конфигурационный файл должен содержать следующие настройки:

- а) домен, на котором будет доступен пользовательский интерфейс и прикладной программный интерфейс,
- б) логин и пароль пользователя с ролью администратор.

Загрузка Docker-контейнеров осуществляется командой `docker pull <имя контейнера>`, а запуск контейнера командой `docker run <имя контейнера>`. Кроме того при запуске контейнера необходимо передать следующие аргументы:

- а) `-it` запуск с доступом к терминалу,
- б) `-d` запуск в режиме демона,
- в) `-p` запуск с перенаправлением портов.

После того как все контейнеры запущены ЭО ПАП готов к работе. Последним шагом является создание пользователей для доступа к пользовательскому интерфейсу и установка драйверов электронных устройств из предложенного списка.

10.5 Выводы

В данном разделе отчета описано изготовление ЭО ПАП проведенным Индустриальным партнером. Данная работа заключалась в разработке исполняемых файлов ЭО ПО ПАП, установке и конфигурировании опера-

ционных системы, а так же в установке и конфигурировании программного обеспечения ЭО ПАП.

Все работы проведены в соответствии с Техническим заданием, а именно пунктом 4.3. А так же в соответствии с эскизной конструкторской и программной документацией ЭО ПАП, представленной в данном отчете о ПНИ.

Для подтверждения факта изготовления ЭО ПАП была сформирована комиссия и составлен акт изготовления от 20-го июня 2016 года, который прилагается к данному отчету о ПНИ.

ЗАКЛЮЧЕНИЕ

В данном отчете представлены результаты работ по экспериментальными исследованиям на 4 этапе ПНИ «Разработка прототипа масштабируемой сервис-ориентированной программно-аппаратной платформы на основе беспроводных сенсорных и агентных сетей, технологий семантического веба и облачных вычислений в целях агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных в распределенной сети электронных потребительских устройств (Internet of Things)».

В данном отчете представлены результаты следующих работ, в соответствии с Планом-графиком работ (далее, ПГ):

- а) разработка МШСПОИ,
- б) разработка программы и методик лабораторных испытаний МШСПОИ,
- в) лабораторные испытания МШСПОИ,
- г) разработка ЭО ПО ПАП,
- д) разработка ЭО ПАП,
- е) разработка программы и методик экспериментальных исследований ЭО ПАП,
- ж) экспериментальные исследования ЭО ПАП с использованием ПАС,
- з) экспериментальные исследования ЭО ПАП с использованием экспериментального образца внешней информационной системы, получающей доступ к гетерогенным данным распределенной сети электронных потребительских устройств через интерфейс прикладного программирования (API).

Далее перечисляются основные результаты, полученные за отчетный период и описанные в данном отчете о ПНИ.

В рамках работы 4.1 ПГ была разработана эскизная конструкторская и программная документация представленная в Приложениях А–Д.

В рамках работ 4.2 и 4.3 была разработана программа и методики лабораторных испытаний МШСПОИ и проведены испытания. Программа и ме-

тодики лабораторных испытаний и соответствующие протоколы испытаний прилагаются к данному отчету о ПНИ в виде самостоятельного документа.

В рамках работы 4.4 была разработана программная документация представленная в Приложениях Е–П.

В рамках работы 4.5 была разработана эскизная конструкторская документация представленная в Приложениях Р–Т.

В рамках работ 4.6–4.8 была разработана программа и методики экспериментальных исследований и проведены исследования. Программа и методики экспериментальных исследований и соответствующие протоколы исследований прилагаются к данному отчету о ПНИ в виде самостоятельного документа.

На отчетном этапе были достигнуты все необходимые результаты в полном объеме, предусмотренные Планом-графиком работ, которые удовлетворяют требования, сформулированным в техническом задании.

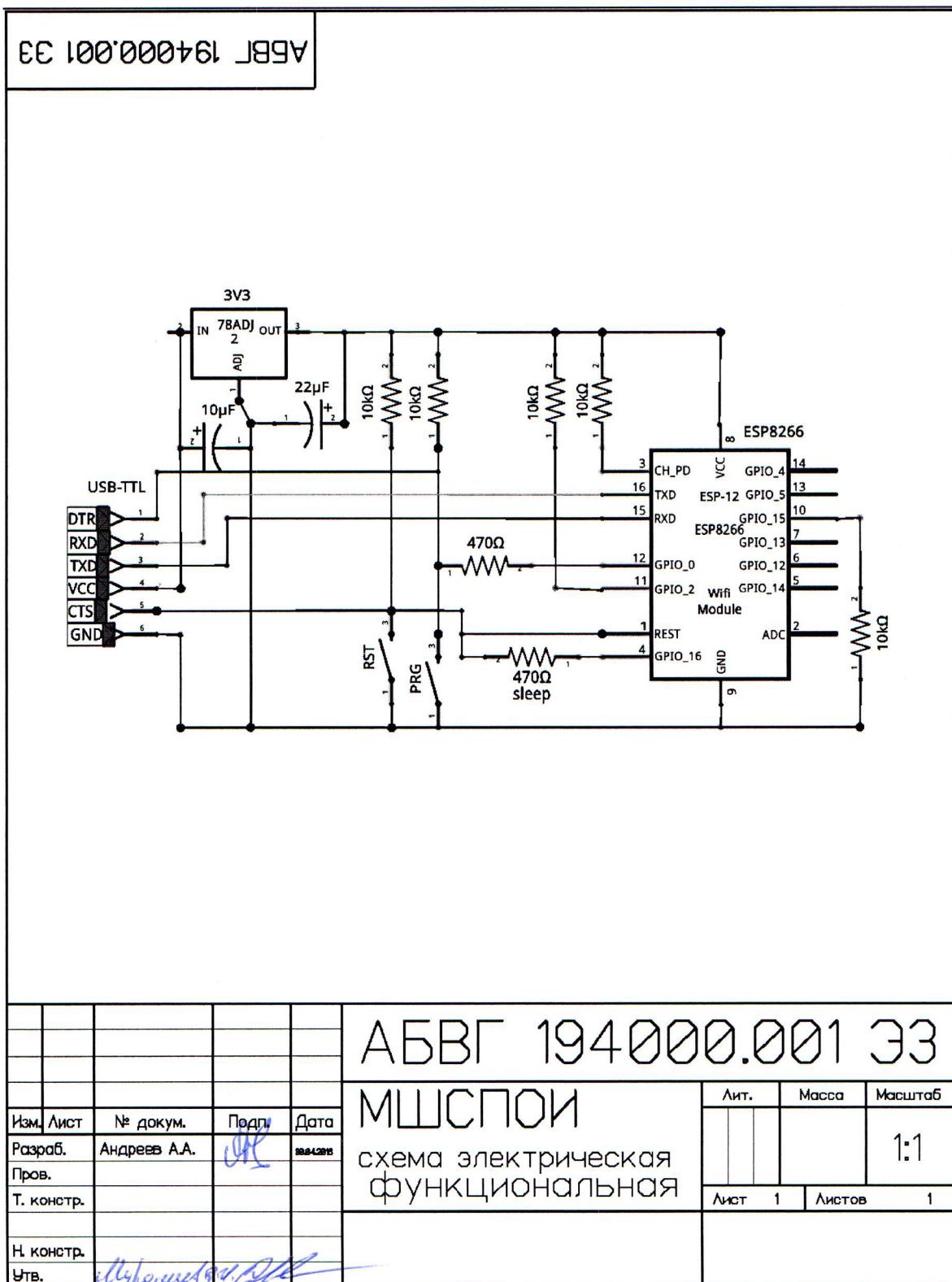
На специализированном сайте в сети Интернет, доступном по адресу <http://semiot.ru>, размещена информацию о ходе выполнения и результатах промежуточных работ по данному ПНИ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Муромцев, Д. И. Разработка прототипа масштабируемой сервис-ориентированной программно-аппаратной платформы на основе беспроводных сенсорных и агентных сетей, технологий семантического веба и облачных вычислений в целях агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полуструктурных и неструктурных данных в распределенной сети электронных потребительских устройств (Internet of Things). Этап 1 «Выбор направления исследований» / Д. И. Муромцев и др.; ВИНИТИ. — М.: Университет ИТМО, 2014. — Июль.

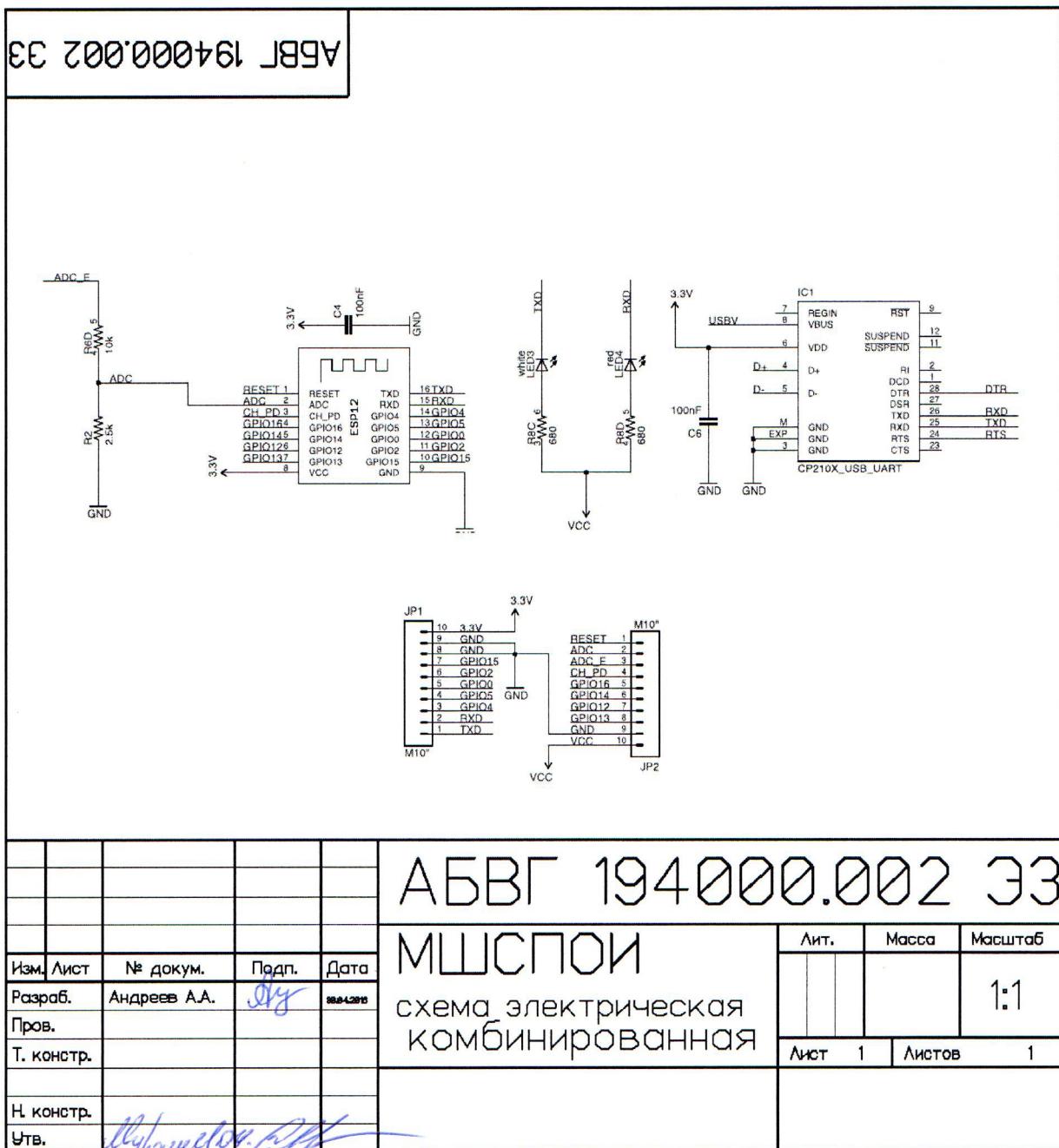
Приложение А

Схема электрическая функциональная МШСПОИ



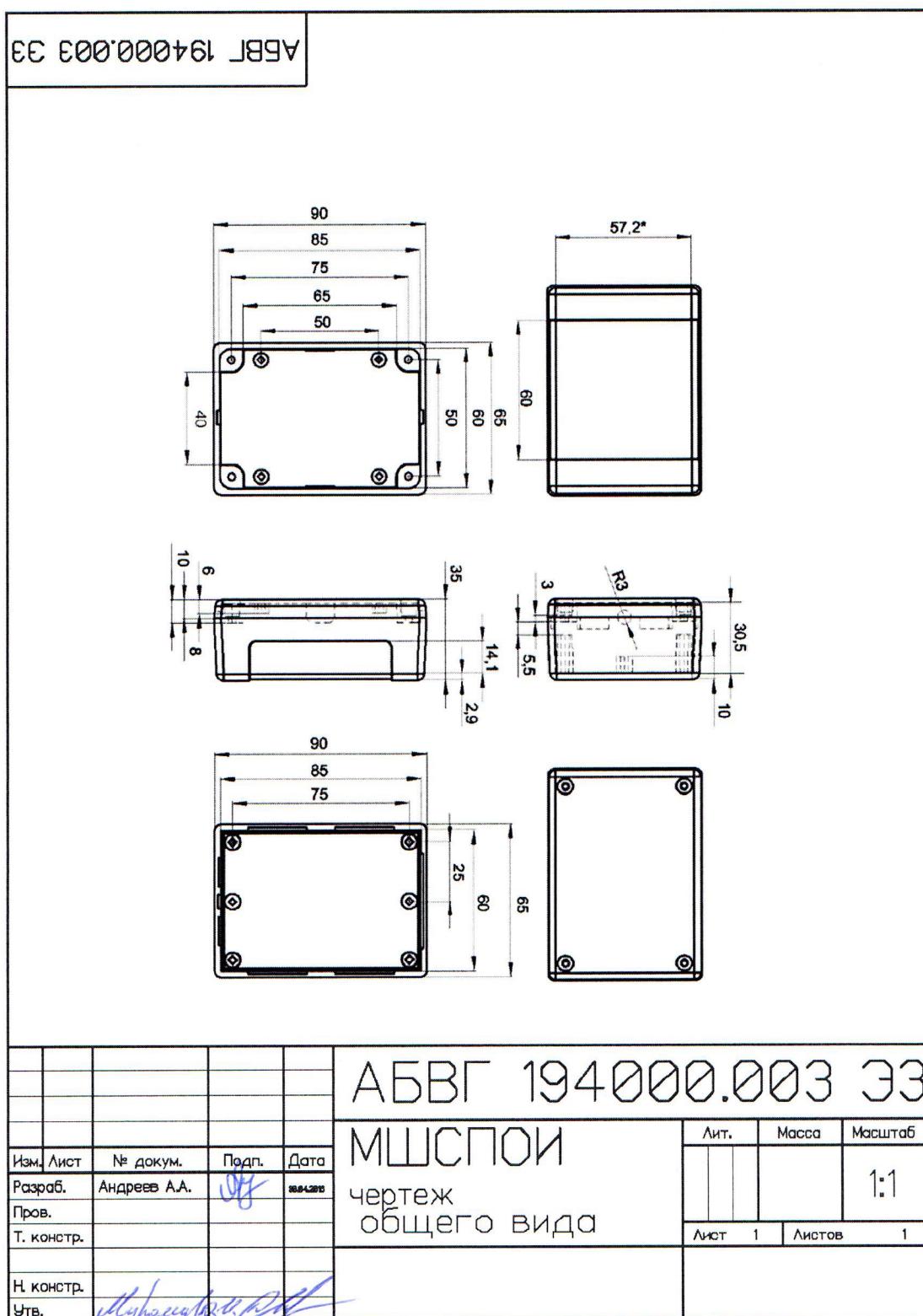
Приложение Б

Схема комбинированная соединений и подключения МШСПОИ



Приложение В

Чертеж общего вида МШСПОИ



Приложение Г

Описание программного обеспечения МШСПОИ

УТВЕРЖДАЮ

Руководитель проекта

к.т.н., доцент

 Д.И. Муромцев
«30» июня 2016 г.

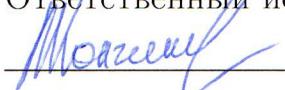
Программное обеспечение МШСПОИ

Описание программы

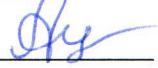
ЛИСТ УТВЕРЖДЕНИЯ

МШСПОИ.00001-01 13 01-ЛУ

Ответственный исполнитель

 М.А. Колчин
«30» июня 2016 г.

Исполнитель

 А.А. Андреев
«30» июня 2016 г.

2016

УТВЕРЖДЕНО

МШСПОИ.00001-01 13 01-ЛУ

Программное обеспечение МШСПОИ

Описание программы

МШСПОИ.00001-01 13 01

Инв. № подл.	Подпись и дата	Взам инв. №	Инв. № дубл.	Подпись и дата

Аннотация

Программное обеспечение МШСПОИ предназначено для предоставления доступа к данным электронных устройств и управлением через беспроводную сеть Wi-Fi. Программный модуль поддерживает основные открытые платформы и позволяет реализовать узел получения данных с устройств, не имеющих прямого доступа в проводные и беспроводные сети. Программный модуль реализует следующие функции: позволяет получать данные с электронных устройств любым доступным операционной системе или прошивке устройства образом (импульсные входы, цифровые и аналоговые входы, по прерываниям и в цикле), формировать интерфейс доступа по выбранному бинарному протоколу RFC 7252 (Constrained Application Protocol (CoAP)) с возможностью чтения и записи данных. Дополнительно реализована поддержка информирования всех заинтересованных клиентов об обновлениях выбранного ресурса согласно протоколу CoAP. Кроме того, на основе полученного интерфейса сформирован защищенный механизм конфигурирования подключения устройства к пользовательской локальной сети Wi-Fi посредством дополнительной программной точки доступа Wi-Fi, защищенной по стандарту WPA2/PSK. Предлагаемый механизм конфигурирования также является самодокументируемым с использованием рекомендации консорциума W3C RDF Schema, а программный модуль поддерживает автоматическое обнаружение новых устройств и их ресурсов по стандарту RFC 6690 (CoRE Link Format).

Оформление программного документа «Описание программы» произведено по требованиям ЕСПД (ГОСТ 19.402-78).

Г.1 Общие сведения

Программный модуль МШСПОИ работает в операционных системах семейства Linux, а также в системах на чипах семейства Espressif ESP8266, используя двоичный сетевой протокол СоАР на основе стандарта UDP.

Г.2 Функциональное назначение

Программный модуль МШСПОИ предназначен для обнаружения и предоставления доступа к требуемым ресурсам системы электронных устройств: для получения своевременного уведомления об изменениях, а также для ручного получения текущего статуса о состоянии системы и управления ресурсами, доступными для изменения.

Г.3 Описание логической структуры

Система производит опрос сетевого модуля на начиание новых входящих сетевых пакетов по стандартизированному СоАР порту 5683 протокола UDP. В случае получения нового сообщения сохраняется адрес отправителя и производится проверка пакета на формальное соответствие стандарту СоАР и отсутствие ошибок.

После получения информации о запрашиваемом методе и имени ресурса из входящего пакета производится проверка на наличие такового в системе согласно стандарту RFC 6690 (CoRE Link Format).

Частным случаем является документированный ресурс *.well-known/core*, который предоставляет информацию о доступных ресурсах в системе.

Каждый ресурс при этом является самодокументируемым, реализующим принципы Linked Data.

В случае обнаружения имеющегося ресурса выполняется указанный для него обработчик и отправляется сформированное обработчиком ответное сообщение. В противном случае отправляется стандартизированное сообщение об ошибке.

Г.4 Используемые технические средства

Исполнение программного модуля МШСПОИ предназначено для систем на чипах семейства Espressif ESP8266, а также процессоров архитектуры Intel x86, x86_64, ARMv7. Реализация выполнена на языке C++ стандарта 2011 года ISOIEC 14882:2011.

Кроссплатформенное программное обеспечение предполагает использование библиотек Arduino для модулей ESP8266, а также библиотеку WiringPi для ARMv7-процессоров, совместимых с архитектурой с миникомпьютером Raspberry Pi. Для архитектур Intel x86 и x86_64 доступна поддержка операционных систем семейства Linux и компиляторов GCC версии не менее 4.8.

Г.5 Вызов и загрузка

Для сборки программного обеспечения на архитектурах, поддерживающих операционные системы семейства Linux, подготовлены инструкции для инструментария CMake версии не менее 2.8. Для работы библиотеки WiringPi требуется дополнительное разрешение на обращение к системным ресурсам и портам ввода-вывода.

Для выполнения сборки необходимо последовательно выполнить следующие команды:

- а) cmake
- б) make
- в) ./semiot-minicoap

Для сборки программного обеспечения на архитектурах чипов семейства Espressif ESP8266 требуется воспользоваться инструментарием Espressif SDK и набором библиотек Arduino для него, после выполнив команды для сборки и загрузки на модуль:

- а) make
- б) make upload

Система поддерживает вывозы по протоколу СоAP в формате пакетов без разбивки на блоки.

Г.6 Входные данные

Программный модуль получает на вход UDP-пакеты формата СоAP RFC 7252 и идентифицирует отправителя по IP-адресу версии 4 в случае использования систем на чипах семейства Espressif ESP8266 и IP версии 4 и 6 в остальных случаях.

Г.7 Выходные данные

Программный подууль отправляет UDP-пакеты отправителю в формате СоAP RFC 7252 и выводит служебную информацию о состоянии работы в потоки системного вывода в случае отладочного режима.

Приложение Д

Текст программы программного обеспечения МШСПОИ

УТВЕРЖДАЮ

Руководитель проекта

к.т.н., доцент

 Д.И. Муромцев
«30» июня 2016 г.

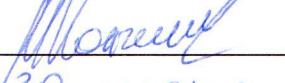
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МШСПОИ

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

МШСПОИ.00001-01 12 01-ЛУ

Ответственный исполнитель

 М.А. Колчин
«30» июня 2016 г.

Исполнитель

 А.А. Андреев
«30» июня 2016 г.

2016

УТВЕРЖДЕНО

МШСПОИ.00001-01 12 01-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МШСПОИ

Текст программы

МШСПОИ.00001-01 12 01

Инв. № подл.	Подпись и дата	Взам инв. №	Инв. № дубл.	Подпись и дата

Аннотация

Программный модуль МШСПОИ предназначен для предоставления доступа к данным электронных устройств и управлением через беспроводную сеть Wi-Fi. Программный модуль поддерживает основные открытые платформы и позволяет реализовать узел получения данных с устройств, не имеющих прямого доступа в проводные и беспроводные сети. Программный модуль реализует следующие функции: позволяет получать данные с электронных устройств любым доступным операционной системе или прошивке устройства образом (импульсные входы, цифровые и аналоговые входы, по прерываниям и в цикле), формировать интерфейс доступа по выбранному бинарному протоколу RFC 7252 (Constrained Application Protocol (CoAP)) с возможностью чтения и записи данных. Дополнительно реализована поддержка информирования всех заинтересованных клиентов об обновлениях выбранного ресурса согласно протоколу CoAP. Кроме того, на основе полученного интерфейса сформирован защищенный механизм конфигурирования подключения устройства к пользовательской локальной сети Wi-Fi посредством дополнительной программной точки доступа Wi-Fi, защищенной по стандарту WPA2/PSK. Предлагаемый механизм конфигурирования также является самодокументируемым с использованием рекомендации консорциума W3C RDF Schema, а программный модуль поддерживает автоматическое обнаружение новых устройств и их ресурсов по стандарту RFC 6690 (CoRE Link Format).

Исходным языком программирования является C++.

Оформление программного документа «Текст программы» произведено по требованиям ЕСПД (ГОСТ 19.401-78).

Д.1 Текст программы программного обеспечения МШСПОИ

Д.1.1 Заголовочный файл с описанием основного класса реализации протокола СоАР

```
#ifndef MINICOAP_H
#define MINICOAP_H

#include "minicoapdefinitions.h"

class MiniCoAP
{
public:
    MiniCoAP();
    int begin();
    int coap_make_response(const coap_packet_t *inpkt, coap_packet_t *outpkt,
                           uint8_t *content, size_t content_len, coap_responsecode_t rspcode,
                           coap_content_type_t content_type);
    int addEndpoint(coap_method_t method, coap_endpoint_func handler, const
                    coap_endpoint_path_t *path, const char *core_attr = NULL, bool
                    *obs_changed = NULL);
    void answerForIncomingRequest();
    void buildWellKnownCoreString(char *dst, ssize_t len);
    coap_client_socket_t getCurrentSocket();

#ifdef OBS_SUPPORT
    void answerForObservations();
#endif // OBS_SUPPORT

private:
#ifdef OBS_SUPPORT
    void answerForObservation(unsigned int index);
    int addObserver(const coap_packet_t *inpkt);
    int removeObserver();
#endif // OBS_SUPPORT
    int receiveUDP();
    int sendUDP();
    int answer(coap_packet_t *pkt);
    unsigned int endpointsCount = 0;
    coap_endpoint_t endpoints[MAX_ENDPOINTS_COUNT];

    unsigned int port;
    int fd;

#ifdef ARDUINO
    WiFiUDP udp;
    coap_client_socket_t servaddr, cliaddr;

```

```

#else
    #ifdef IPV6
        struct sockaddr_in6 servaddr , cliaddr ;
    #else /* IPV6 */
        struct sockaddr_in servaddr , cliaddr ;
    #endif /* IPV6 */
#endif // ARDUINO

    uint8_t buf[4096];
    uint8_t scratch_raw[4096];
    coap_rw_buffer_t scratch_buf;
    uint16_t rsplen = MAXRESPLEN; // .well-known/core answer length
    int coap_parse(coap_packet_t *pkt, const uint8_t *buf, size_t buflen);
    int coap_parseHeader(coap_header_t *hdr, const uint8_t *buf, size_t buflen);
    int coap_parseToken(coap_buffer_t *tokbuf, const coap_header_t *hdr, const
                        uint8_t *buf, size_t buflen);
    // http://tools.ietf.org/html/rfc7252#section-3.1
    int coap_parseOptionsAndPayload(coap_option_t *options, uint8_t
                                   *numOptions, coap_buffer_t *payload, const coap_header_t *hdr, const
                                   uint8_t *buf, size_t buflen);
    // advances p
    int coap_parseOption(coap_option_t *option, uint16_t *running_delta, const
                         uint8_t **buf, size_t buflen);
    int coap_compare_uri_path_opt(const coap_packet_t* inpkt, const
                                 coap_endpoint_path_t* path);
    int coap_handle_req(const coap_packet_t *inpkt, coap_packet_t *outpkt);
    int coap_build(uint8_t *buf, size_t *buflen, const coap_packet_t *pkt);
    void coap_option_nibble(uint32_t value, uint8_t *nibble);
    // options are always stored consecutively, so can return a block with same
    option num
    const coap_option_t *coap_findOptions(const coap_packet_t *pkt, uint8_t
                                         num, uint8_t *count = NULL);
    void endpoint_setup(void);
    char rsp[MAXRESPLEN];
    void build_rsp(); // build .well-known/core answer
    static int handle_get_well_known_core(const coap_packet_t *inpkt,
                                         coap_packet_t *outpkt);

#endif // DEBUG

    void coap_dumpHeader(coap_header_t *hdr);
    void coap_dump(const uint8_t *buf, size_t buflen, bool bare);
    void coap_dumpOptions(coap_option_t *opts, size_t numopt);
    void coap_dumpPacket(coap_packet_t *pkt);

#endif // OBS_SUPPORT

```

```
};

#endif // MINICOAP_H
```

Д.1.2 Исходный файл с описанием основного класса реализации протокола СоАР

```
#include "minicoap.h"

MiniCoAP :: MiniCoAP()
{
    port = PORT;
    scratch_buf = {scratch_raw, sizeof(scratch_raw)};
#ifdef ARDUINO

#else // ARDUINO
#ifdef IPV6
    fd = socket(AF_INET6, SOCK_DGRAM, 0);
#else /* IPV6 */
    fd = socket(AF_INET, SOCK_DGRAM, 0);
#endif /* IPV6 */
    fcntl(fd, F_SETFL, O_NONBLOCK);
    bzero(&servaddr, sizeof(servaddr));
#endif /* IPV6 */
#ifdef IPV6
    servaddr.sin6_family = AF_INET6;
    servaddr.sin6_addr = in6addr_any;
    servaddr.sin6_port = htons(port);
#else /* IPV6 */
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(port);
#endif /* IPV6 */
    int bind_result = ::bind(fd, (struct sockaddr *)&servaddr, sizeof(servaddr));
#endif /* DEBUG */
    printf("MiniCOAP: bind socket result: %d\n", bind_result);
#endif // ARDUINO
    endpoint_setup();
}

int MiniCoAP :: begin()
{
#ifdef ARDUINO
    return udp.begin(port);
#endif
    return 0;
}
```

```

}

int MiniCoAP::addEndpoint(coap_method_t method, coap_endpoint_func handler,
    const coap_endpoint_path_t *path, const char *core_attr, bool* obs_changed)
{
    if (endpointsCount+1<=MAX_ENDPOINTS_COUNT) {
        endpoints[endpointsCount].method=method;
        endpoints[endpointsCount].handler=handler;
        endpoints[endpointsCount].path=path;
        endpoints[endpointsCount].core_attr=core_attr;
        endpoints[endpointsCount].obs_changed=obs_changed;
        endpointsCount++;
        return 1;
    }
    return 0;
}

void MiniCoAP::answerForIncomingRequest()
{
    int x = receiveUDP();
    if (x>0) { // filling buf
        rsplen=x;
#ifndef DEBUG
        printf("Received %d bytes:",rsplen);
        coap_dump(buf, rsplen, true);
        printf("\n");
#endif
    }
    int rc;
    coap_packet_t pkt;
    if (0 == (rc = coap_parse(&pkt, buf, (size_t)rsplen))) {
        answer(&pkt);
    }
#ifndef DEBUG
    else {
        printf("Bad packet rc=%d\n", rc);
    }
#endif // DEBUG
}
}

void MiniCoAP::buildWellKnownCoreString(char *dst, ssize_t len)
{
    memset(dst, 0, len);
    const coap_endpoint_t *ep = endpoints;
    int i;
    len--; // Null-terminated string
}

```

```

ep++; // well-known core

while (NULL != ep->handler)
{
    if (NULL == ep->core_attr) {
        ep++;
        continue;
    }

    if (0 < strlen(dst)) {
        strncat(dst, ",", len);
        len--;
    }

    strncat(dst, "<", len);
    len--;

    for (i = 0; i < ep->path->count; i++) {
        strncat(dst, "/", len);
        len--;

        strncat(dst, ep->path->elems[i], len);
        len -= strlen(ep->path->elems[i]);
    }

    strncat(dst, ">;", len);
    len -= 2;

    strncat(dst, ep->core_attr, len);
    len -= strlen(ep->core_attr);

    ep++;
}

#endif DEBUG
printf("Well-known core: %s\n", dst);
#endif // DEBUG
}

coap_client_socket_t MiniCoAP::getCurrentSocket()
{
    return cliaddr;
}

void MiniCoAP::answerForObservation(unsigned int index)
{
    if (index<MAX_OBSERVATIONS_COUNT) {

```

```

        if (!observers[index].cliaddr.available) {
            int i = observers[index].endpoint_path_index;
            if (endpoints[i].obs_changed) {
                if (*endpoints[i].obs_changed) {
                    coap_client_socket_t *clisock = &observers[index].cliaddr;
#define ARDUINO
                    cliaddr.host = clisock->host;
                    cliaddr.port = clisock->port;
#else // ARDUINO
                    cliaddr.sin_family = clisock->socket.sin_family;
                    cliaddr.sin_addr.s_addr = clisock->socket.sin_addr.s_addr;
                    cliaddr.sin_port = clisock->socket.sin_port;
#endif // ARDUINO
                    observers[index].obs_tick++;
                    answer(&observers[index].inpkt);
                }
            }
        }
    }

void MiniCoAP::answerForObservations()
{
    for (int i = 0; i<MAX_OBSERVATIONS_COUNT; i++) {
        answerForObservation(i);
    }
    for (int i = 0; i<MAX_ENDPOINTS_COUNT; i++) {
        if (endpoints[i].obs_changed) {
            *endpoints[i].obs_changed = false;
        }
    }
}

int MiniCoAP::addObserver(const coap_packet_t *inpkt)
{
    for (int i=0;i<MAX_OBSERVATIONS_COUNT; i++) {
        if (!observers[i].cliaddr.available) {
            bool addrIsEq = false;
#define ARDUINO
            addrIsEq = (observers[i].cliaddr.host==cliaddr.host) &&
                (observers[i].cliaddr.port==cliaddr.port);
#else // ARDUINO
            addrIsEq =
                ((observers[i].cliaddr.socket.sin_family==cliaddr.sin_family) &&

```

```

        (observers[i].cliaddr.socket.sin_addr.s_addr==cliaddr.sin_addr.s_addr)&&
        (observers[i].cliaddr.socket.sin_port==cliaddr.sin_port));
#endif // ARDUINO
    if (addrIsEq) {
        // updating token:
        memcpy(observers[i].scratch_raw,inpkt->tok.p,inpkt->tok.len);
        uint8_t opt_count;
        if (coap_findOptions(inpkt,COAP_OPTION_OBSERVE,&opt_count)) {
            observers[i].inpkt.opts[opt_count].buf.len=0;
            memcpy(observers[i].inpkt.hdr.id,inpkt->tok.p,2); // so ugly
        }
        return i;
    }
}
int x = -1;
for (int i=0;i<MAX_OBSERVATIONS_COUNT; i++) {
    if (observers[i].cliaddr.available) {
        x = i;
        break;
    }
}
if (x>-1) {
    if (inpkt) {
        observers[x].obs_tick=1;
        observers[x].inpkt=*inpkt;
        observers[x].inpkt.tok.p = observers[x].scratch_raw;
        memcpy(observers[x].scratch_raw,inpkt->tok.p,inpkt->tok.len);
#endif // ARDUINO
        observers[x].cliaddr=cliaddr;
#else // ARDUINO
        observers[x].cliaddr.socket=cliaddr;
#endif // ARDUINO
        observers[x].cliaddr.available=false;
        // saving endpoint_path_index:
        for (int i=0;i<MAX_ENDPOINTS_COUNT; i++) {
            if
                (coap_compare_uri_path_opt(&observers[x].inpkt,endpoints[i].path))
            {
                observers[x].endpoint_path_index=i;
                break;
            }
        }
    }
}
}

```

```

        return x;
    }

int MiniCoAP::removeObserver()
{
    int x = addObserver(NULL);
    if (x>-1) {
        observers[x].cliaddr.available=true;
    }
    return x;
}

int MiniCoAP::receiveUDP()
{
    ssize_t x;
#ifdef ARDUINO
    x = udp.parsePacket();
    if(x>0) {
        udp.read(buf,sizeof(buf));
        cliaddr.host = udp.remoteIP();
        cliaddr.port = udp.remotePort();
    }
#else
    socklen_t len = sizeof(cliaddr);
    x = recvfrom(fd, buf, sizeof(buf), 0, (struct sockaddr *)&cliaddr, &len);
#endif // ARDUINO
    return x;
}

int MiniCoAP::sendUDP()
{
#ifdef ARDUINO
    udp.beginPacket(cliaddr.host, cliaddr.port);
    udp.write(buf,rsplen);
    udp.endPacket();
    return rsplen;
#else
    return sendto(fd, buf, rsplen, 0, (struct sockaddr *)&cliaddr,
                  sizeof(cliaddr));
#endif // ARDUINO
    return -1;
}

int MiniCoAP::answer(coap_packet_t* pkt)
{

```

```

int rc;
rsplen = sizeof(buf);
coap_packet_t rsppkt;
#ifndef DEBUG
    coap_dumpPacket(pkt);
#endif
    coap_handle_req(pkt, &rsppkt);

    if (0 != (rc = coap_build(buf, (size_t*)&rsplen, &rsppkt))) {
#ifndef DEBUG
        printf("coap_build failed rc=%d\n", rc);
#endif // DEBUG
        return 0;
    }
    else
    {
#ifndef DEBUG
        printf("Sending: ");
        coap_dump(buf, rsplen, true);
        printf("\n");
        coap_dumpPacket(&rsppkt);
#endif // DEBUG
        int sendedCount = sendUDP();
        if (sendedCount== -1) {
        }
    }
    return 0;
}

int MiniCoAP::coap_parse(coap_packet_t *pkt, const uint8_t *buf, size_t buflen)
{
    int rc;
    // coap_dump(buf, buflen, false);
    if (0 != (rc = coap_parseHeader(&pkt->hdr, buf, buflen))) {
        return rc;
    }
    // coap_dumpHeader(&hdr);
    if (0 != (rc = coap_parseToken(&pkt->tok, &pkt->hdr, buf, buflen))) {
        return rc;
    }
    pkt->numopts = MAXOPT;
    if (0 != (rc = coap_parseOptionsAndPayload(pkt->opts, &(pkt->numopts),
                                                &(pkt->payload), &pkt->hdr, buf, buflen))) {
        return rc;
    }
    // coap_dumpOptions(opts, numopt);
}

```

```

        return 0;
    }

int MiniCoAP::coap_make_response( const coap_packet_t *inpkt , coap_packet_t
    *outpkt , uint8_t *content , size_t content_len , coap_responsecode_t rspcode ,
    coap_content_type_t content_type)
{
    uint8_t msgid_hi = inpkt->hdr.id[0];
    uint8_t msgid_lo = inpkt->hdr.id[1];
    outpkt->hdr.ver = 0x01;
    outpkt->hdr.t = COAP_TYPE_ACK;
    outpkt->hdr.tkl = 0;
    outpkt->hdr.code = rspcode;
    outpkt->hdr.id[0] = msgid_hi;
    outpkt->hdr.id[1] = msgid_lo;
    outpkt->numopts = 1;

    // need token in response
    if (inpkt->hdr.tkl) {
        outpkt->hdr.tkl = inpkt->hdr.tkl;
        outpkt->tok = inpkt->tok;
    }

    // cheking for observe option:
    bool obs = false;
    const coap_option_t *observeOption =
        coap_findOptions(inpkt,COAP_OPTION_OBSERVE);
    if (observeOption != NULL) {
        obs = true;
        // https://tools.ietf.org/html/rfc7641#section-2
        if (observeOption->buf.len == 0) { // register
            int observer_count = addObserver(inpkt);
            #ifdef DEBUG
                printf("add observer result: %d\n",observer_count);
            #endif // DEBUG
            if (observer_count>-1) {
                outpkt->numopts = 2;
                // safe because 1 < MAXOPT
                outpkt->opts[0].num = COAP_OPTION_OBSERVE;
                outpkt->opts[0].buf.p = &observers[observer_count].obs_tick;
                outpkt->opts[0].buf.len = 1;
            }
        }
        else if ((observeOption->buf.len == 1) && (*observeOption->buf.p==1)) {
            removeObserver();
        }
    }
}

```

```

        }

    }

outpkt->opts[obs].num = COAP_OPTION_CONTENT_FORMAT;
outpkt->opts[obs].buf.p = scratch_buf.p;
if (scratch_buf.len < 2)
    return COAP_ERR_BUFFER_TOO_SMALL;
scratch_buf.p[0] = ((uint16_t)content_type & 0xFF00) >> 8;
scratch_buf.p[1] = ((uint16_t)content_type & 0x00FF);
outpkt->opts[obs].buf.len = 2;
outpkt->payload.p = content;
outpkt->payload.len = content_len;

return 0;
}

int MiniCoAP::coap_parseHeader(coap_header_t *hdr, const uint8_t *buf, size_t buflen)
{
    if (buflen < 4)
        return COAP_ERR_HEADER_TOO_SHORT;
    hdr->ver = (buf[0] & 0xC0) >> 6;
    if (hdr->ver != 1)
        return COAP_ERR_VERSION_NOT_1;
    hdr->t = (buf[0] & 0x30) >> 4;
    hdr->tkl = buf[0] & 0x0F;
    hdr->code = buf[1];
    hdr->id[0] = buf[2];
    hdr->id[1] = buf[3];
    return 0;
}

int MiniCoAP::coap_parseToken(coap_buffer_t *tokbuf, const coap_header_t *hdr,
    const uint8_t *buf, size_t buflen)
{
    if (hdr->tkl == 0)
    {
        tokbuf->p = NULL;
        tokbuf->len = 0;
        return 0;
    }
    else
    if (hdr->tkl <= 8)
    {
        if (4U + hdr->tkl > buflen)
            return COAP_ERR_TOKEN_TOO_SHORT; // tok bigger than packet
    }
}

```

```

        tokbuf->p = buf+4; // past header
        tokbuf->len = hdr->tkl;
        return 0;
    }
    else
    {
        // invalid size
        return COAP_ERR_TOKEN_TOO_SHORT;
    }
}

int MiniCoAP::coap_parseOptionsAndPayload(coap_option_t *options, uint8_t
    *numOptions, coap_buffer_t *payload, const coap_header_t *hdr, const uint8_t
    *buf, size_t buflen)
{
    size_t optionIndex = 0;
    uint16_t delta = 0;
    const uint8_t *p = buf + 4 + hdr->tkl;
    const uint8_t *end = buf + buflen;
    int rc;
    if (p > end)
        return COAP_ERR_OPTION_OVERRUNS_PACKET; // out of bounds

    //coap_dump(p, end - p);

    // 0xFF is payload marker
    while ((optionIndex < *numOptions) && (p < end) && (*p != 0xFF))
    {
        if (0 != (rc = coap_parseOption(&options[optionIndex], &delta, &p,
            end-p)))
            return rc;
        optionIndex++;
    }
    *numOptions = optionIndex;

    if (p+1 < end && *p == 0xFF) // payload marker
    {
        payload->p = p+1;
        payload->len = end-(p+1);
    }
    else
    {
        payload->p = NULL;
        payload->len = 0;
    }
}

```

```

        return 0;
    }

int MiniCoAP::coap_parseOption(coap_option_t *option, uint16_t *running_delta,
                               const uint8_t **buf, size_t buflen)
{
    const uint8_t *p = *buf;
    uint8_t headlen = 1;
    uint16_t len, delta;

    if (buflen < headlen) // too small
        return COAP_ERR_OPTION_TOO_SHORT_FOR_HEADER;

    delta = (p[0] & 0xF0) >> 4;
    len = p[0] & 0x0F;

    // These are untested and may be buggy
    if (delta == 13)
    {
        headlen++;
        if (buflen < headlen)
            return COAP_ERR_OPTION_TOO_SHORT_FOR_HEADER;
        delta = p[1] + 13;
        p++;
    }
    else
        if (delta == 14)
    {
        headlen += 2;
        if (buflen < headlen)
            return COAP_ERR_OPTION_TOO_SHORT_FOR_HEADER;
        delta = ((p[1] << 8) | p[2]) + 269;
        p+=2;
    }
    else
        if (delta == 15)
            return COAP_ERR_OPTION_DELTA_INVALID;

    if (len == 13)
    {
        headlen++;
        if (buflen < headlen)
            return COAP_ERR_OPTION_TOO_SHORT_FOR_HEADER;
        len = p[1] + 13;
        p++;
    }
}

```

```

else
if (len == 14)
{
    headlen += 2;
    if (buflen < headlen)
        return COAP_ERR_OPTION_TOO_SHORT_FOR_HEADER;
    len = ((p[1] << 8) | p[2]) + 269;
    p+=2;
}
else
if (len == 15)
    return COAP_ERR_OPTION_LEN_INVALID;

if ((p + 1 + len) > (*buf + buflen))
    return COAP_ERR_OPTION_TOO_BIG;

// printf("option num=%d\n", delta + *running_delta);
option->num = delta + *running_delta;
option->buf.p = p+1;
option->buf.len = len;
//coap_dump(p+1, len, false);

// advance buf
*buf = p + 1 + len;
*running_delta += delta;

return 0;
}

int MiniCoAP::coap_compare_uri_path_opt(const coap_packet_t *inpkt, const
                                         coap_endpoint_path_t *path)
{
    if (path) {
        uint8_t count;
        const coap_option_t *opt;
        if (NULL != (opt = coap_findOptions(inpkt, COAP_OPTION_URI_PATH,
                                             &count)))
        {
            if (count != path->count)
                return 0;
            for (int i=0;i<count;i++)
            {
                if (opt[i].buf.len != strlen(path->elems[i]))
                    return 0;
                if (0 != memcmp(path->elems[i], opt[i].buf.p, opt[i].buf.len))
                    return 0;
            }
        }
    }
}

```

```

        }
        // match!
        return 1;
    }
}

int MiniCoAP::coap_handle_req(const coap_packet_t *inpkt, coap_packet_t *outpkt)
{
    coap_endpoint_t *ep = &endpoints[0];

    while (NULL != ep->handler)
    {
        if (ep->method != inpkt->hdr.code)
            goto next;
        if (coap_compare_uri_path_opt(inpkt, ep->path)) {
            // match!
            return ep->handler(inpkt, outpkt);
        }
    next:
        ep++;
    }

    coap_make_response(inpkt, outpkt, NULL, 0, COAP_RSPCODE_NOT_FOUND,
                      COAP_CONTENTTYPE_NONE);

    return 0;
}

int MiniCoAP::coap_build(uint8_t *buf, size_t *buflen, const coap_packet_t *pkt)
{
    size_t opts_len = 0;
    size_t i;
    uint8_t *p;
    uint16_t running_delta = 0;

    // build header
    if (*buflen < (4U + pkt->hdr.tkl))
        return COAP_ERR_BUFFER_TOO_SMALL;

    buf[0] = (pkt->hdr.ver & 0x03) << 6;
    buf[0] |= (pkt->hdr.t & 0x03) << 4;
    buf[0] |= (pkt->hdr.tkl & 0x0F);
    buf[1] = pkt->hdr.code;
    buf[2] = pkt->hdr.id[0];
}

```

```

buf[3] = pkt->hdr.id[1];

// inject token
p = buf + 4;
if ((pkt->hdr.tkl > 0) && (pkt->hdr.tkl != pkt->tok.len))
    return COAP_ERR_UNSUPPORTED;

if (pkt->hdr.tkl > 0)
    memcpy(p, pkt->tok.p, pkt->hdr.tkl);

// // http://tools.ietf.org/html/rfc7252#section-3.1
// inject options
p += pkt->hdr.tkl;

for (i=0;i<pkt->numopts;i++)
{
    uint32_t optDelta;
    uint8_t len, delta = 0;

    if (((size_t)(p-buf)) > *buflen)
        return COAP_ERR_BUFFER_TOO_SMALL;
    optDelta = pkt->opts[i].num - running_delta;
    coap_option_nibble(optDelta, &delta);
    coap_option_nibble((uint32_t)pkt->opts[i].buf.len, &len);

    *p++ = (0xFF & (delta << 4 | len));
    if (delta == 13)
    {
        *p++ = (optDelta - 13);
    }
    else
    if (delta == 14)
    {
        *p++ = ((optDelta-269) >> 8);
        *p++ = (0xFF & (optDelta-269));
    }
    if (len == 13)
    {
        *p++ = (pkt->opts[i].buf.len - 13);
    }
    else
    if (len == 14)
    {
        *p++ = (pkt->opts[i].buf.len >> 8);
        *p++ = (0xFF & (pkt->opts[i].buf.len-269));
    }
}

```

```

        memcpy(p, pkt->opts[i].buf.p, pkt->opts[i].buf.len);
        p += pkt->opts[i].buf.len;
        running_delta = pkt->opts[i].num;
    }

    opts_len = (p - buf) - 4; // number of bytes used by options

}

if (pkt->payload.len > 0)
{
    if (*buflen < 4 + 1 + pkt->payload.len + opts_len)
        return COAP_ERR_BUFFER_TOO_SMALL;
    buf[4 + opts_len] = 0xFF; // payload marker
    memcpy(buf+5 + opts_len, pkt->payload.p, pkt->payload.len);
    *buflen = opts_len + 5 + pkt->payload.len;
}
else
    *buflen = opts_len + 4;
return 0;
}

void MiniCoAP::coap_option_nibble(uint32_t value, uint8_t *nibble)
{
    if (value<13)
    {
        *nibble = (0xFF & value);
    }
    else
    if (value<=0xFF+13)
    {
        *nibble = 13;
    } else if (value<=0xFFFF+269)
    {
        *nibble = 14;
    }
}

const coap_option_t *MiniCoAP::coap_findOptions(const coap_packet_t *pkt,
    uint8_t num, uint8_t *count)
{
    size_t i;
    const coap_option_t *first = NULL;
    if (count) {
        *count = 0;

```

```

    }

    for ( i=0;i<pkt->numopts ; i++)
    {
        if ( pkt->opts [ i ].num == num)
        {
            if (NULL == first )
                first = &pkt->opts [ i ];
            if ( count) {
                (*count)++;
            }
        }
        else
        {
            if (NULL != first )
                break;
        }
    }
    return first ;
}

void MiniCoAP :: endpoint _ setup ()
{
/*
endpoints [ 0 ]. method=COAP_METHOD_GET;
endpoints [ 0 ]. handler=(coap _ endpoint _ func )&MiniCoAP :: handle _ get _ well _ known _ core ;
endpoints [ 0 ]. path=&path _ well _ known _ core ;
endpoints [ 0 ]. core _ attr="ct=40";
*/
}

void MiniCoAP :: build _ rsp ()
{

int MiniCoAP :: handle _ get _ well _ known _ core ( const coap _ packet _ t *inpkt ,
    coap _ packet _ t *outpkt )
{
    return 0;
}

#ifndef DEBUG

void MiniCoAP :: coap _ dumpHeader ( coap _ header _ t *hdr )
{
    printf ("Header:\n");

```

```

    printf("  ver  0x%02X\n", hdr->ver);
    printf("  t     0x%02X\n", hdr->t);
    printf("  tkl   0x%02X\n", hdr->tkl);
    printf("  code  0x%02X\n", hdr->code);
    printf("  id    0x%02X%02X\n", hdr->id[0], hdr->id[1]);
}

void MiniCoAP::coap_dump(const uint8_t *buf, size_t buflen, bool bare)
{
    if (bare)
    {
        while (buflen--)
            printf("%02X%s", *buf++, (buflen > 0) ? " " : "");
    }
    else
    {
        printf("Dump: ");
        while (buflen--)
            printf("%02X%s", *buf++, (buflen > 0) ? " " : "");
        printf("\n");
    }
}

void MiniCoAP::coap_dumpOptions(coap_option_t *opts, size_t numopt)
{
    size_t i;
    printf(" Options:\n");
    for (i=0;i<numopt; i++)
    {
        printf("  0x%02X [ ", opts[i].num);
        coap_dump(opts[i].buf.p, opts[i].buf.len, true);
        printf(" ]\n");
    }
}

void MiniCoAP::coap_dumpPacket(coap_packet_t *pkt)
{
    coap_dumpHeader(&pkt->hdr);
    coap_dumpOptions(pkt->opts, pkt->numopts);
    printf(" Payload: ");
    coap_dump(pkt->payload.p, pkt->payload.len, true);
    printf("\n");
}

#endif // DEBUG

```

Д.1.3 Заголовочный файл с описанием класса реализации ресурса устройства

```
#ifndef COAPRESOURCE_H
#define COAPRESOURCE_H

#include "minicoapdefinitions.h"
#include "minicoap.h"

class CoAPResource
{
public:
    CoAPResource(MiniCoAP* coapServer);
    virtual int getMethod(const coap_packet_t *inpkt, coap_packet_t *outpkt) {}
    virtual int putMethod(const coap_packet_t *inpkt, coap_packet_t *outpkt) {}
    virtual int postMethod(const coap_packet_t *inpkt, coap_packet_t *outpkt) {}
    virtual int deleteMethod(const coap_packet_t *inpkt, coap_packet_t *outpkt)
    {
        coap_endpoint_path_t resourcePath;
        bool resourceChanged = true;
        char coreAttr[10];
        MiniCoAP *getServer();
    }
private:
    MiniCoAP *server;
};

#endif // COAPRESOURCE_H
```

Д.1.4 Исходные коды реализации класса ресурса устройства

```
#include "coapresource.h"

CoAPResource::CoAPResource(MiniCoAP *coapServer)
{
    server = coapServer;
    coap_endpoint_func getMP = std::function<int(const coap_packet_t *inpkt,
                                                   coap_packet_t *outpkt)>(std::bind(&CoAPResource::getMethod, this,
                                            std::placeholders::_1, std::placeholders::_2));
    coap_endpoint_func putMP = std::function<int(const coap_packet_t *inpkt,
                                                   coap_packet_t *outpkt)>(std::bind(&CoAPResource::putMethod, this,
                                            std::placeholders::_1, std::placeholders::_2));
    server->addEndpoint(COAP_METHOD_GET, getMP, &resourcePath, coreAttr, &resourceChanged);
    server->addEndpoint(COAP_METHOD_PUT, putMP, &resourcePath);
}

MiniCoAP *CoAPResource::getServer()
```

```
{  
    return server;  
}
```


Приложение Е

Спецификация программного обеспечения ЭО ПАП

УТВЕРЖДАЮ

Руководитель проекта

к.т.н., доцент

 Д.И. Муромцев
«30» июня 2016 г.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭО ПАП

Спецификация

ЛИСТ УТВЕРЖДЕНИЯ

ПАП.00001-01 13 01-ЛУ

Ответственный исполнитель

 М.А. Колчин
«30» июня 2016 г.

Исполнитель

 И.А. Шилин
«30» июня 2016 г.

2016

102

УТВЕРЖДЕНО

ПАП.00001-01 13 01-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭО ПАП

Спецификация

ПАП.00001-01 13 01

Инв. № подл.	Подпись и дата	Взам инв. №	Инв. № дубл.	Подпись и дата

ПАП.00001-01 13 01

Обозначение	Наименование	Примечание
	Документация	
ПАП.00001-01 13 01-ЛУ	Экспериментальный образец	
	программного обеспечения	
	масштабируемой	
	сервис-ориентированной	
	программно-аппаратной	
	платформы для сбора,	
	нормализации , обработки и	
	визуализации больших	
	массивов гетерогенных	
	данных в распределенной	
	сети электронных	
	потребительских устройств	
	Спецификация	
	Лист утверждения	
ПАП.00001-01 13 01	Экспериментальный образец	
	программного обеспечения	
	масштабируемой	
	сервис-ориентированной	
	программно-аппаратной	
	платформы для сбора,	
	нормализации , обработки и	
	визуализации больших	
	массивов гетерогенных	
	данных в распределенной	
	сети электронных	
	потребительских устройств	
	Спецификация	

ПАП.00001-01 01

Обозначение	Наименование	Примечание
ПАП.00001-01 31-ЛУ	Экспериментальный образец	
	программного обеспечения	
	масштабируемой	
	сервис-ориентированной	
	программно-аппаратной	
	платформы для сбора,	
	нормализации , обработки и	
	визуализации больших	
	массивов гетерогенных	
	данных в распределенной	
	сети электронных	
	потребительских устройств	
	Описание применения	
	Лист утверждения	
ПАП.00001-01 31	Экспериментальный образец	
	программного обеспечения	
	масштабируемой	
	сервис-ориентированной	
	программно-аппаратной	
	платформы для сбора,	
	нормализации , обработки и	
	визуализации больших	
	массивов гетерогенных	
	данных в распределенной	
	сети электронных	
	потребительских устройств	
	Описание применения	

ПАП.00001-01 01

Обозначение	Наименование	Примечание
	Компоненты	
ЦОД.00001-01 12 01-ЛУ	Подсистема Центр обработки данных ЭО ПАП	
	Текст программы	
	Лист утверждения	
ЦОД.00001-01 12 01	Подсистема Центр обработки данных ЭО ПАП	
	Текст программы	
ЦОД.00001-01 13 01-ЛУ	Подсистема Центр обработки данных ЭО ПАП	
	Описание программы	
	Лист утверждения	
ЦОД.00001-01 13 01	Подсистема Центр обработки данных ЭО ПАП	
	Описание программы	
СОС.00001-01 12 01-ЛУ	Подсистема Симулятор облака сенсоров ЭО ПАП	
	Текст программы	
	Лист утверждения	
СОС.00001-01 12 01	Подсистема Симулятор облака сенсоров ЭО ПАП	
	Текст программы	

ПАП.00001-01 01

ПАП.00001-01 13 01

Приложение Ж

Описание подсистемы ЦОД ЭО ПАП

УТВЕРЖДАЮ

Руководитель проекта

к.т.н., доцент

 Д.И. Муромцев
«30» июня 2016 г.

ПОДСИСТЕМА ЦОД ЭО ПО ПАП

Описание программы

ЛИСТ УТВЕРЖДЕНИЯ

ЦОД.00001-01 13 01-ЛУ

Ответственный исполнитель

 М.А. Колчин
«30» июня 2016 г.

Исполнитель

 И.А. Шилин
«30» июня 2016 г.

УТВЕРЖДЕНО

ЦОД.00001-01 13 01-ЛУ

ПОДСИСТЕМА ЦОД ЭО ПАП

Описание программы

ЦОД.00001-01 13 01

Инв. № подл.	Подпись и дата	Взам инв. №	Инв. № дубл.	Подпись и дата

Аннотация

В данном программном документе приведено описание программы «Центр обработки данных» (далее - ЦОД), предназначеннай для агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, слабоструктурированных и неструктурных данных.

В документе приводятся общие сведения о ЦОД, реализуемые функции, описание логической структуры программного обеспечения, входные и выходные данные, используемые программным обеспечением. Оформление программного документа «Описание программы» произведено по требованиям ЕСПД (ГОСТ 19.402-78).

Ж.1 Общие сведения

Ж.1.1 Обозначение и наименование программы

Наименование программы: Центр обработки данных (ЦОД).

Ж.1.2 Программное обеспечение, необходимое для функционирования программы

Системные программные средства, используемые программой ЦОД, должны быть представлены версией операционной системы Ubuntu Server 14.10 или выше.

Также для функционирования программы ЦОД на сервере необходимо предустановленное программное обеспечение Java Virtual Machine 1.8, MySQL 5.6 или выше, Docker 1.9.1, Docker-compose 1.7.1.

Ж.1.3 Языки программирования, на которых написана программа

Программа ЦОД реализована на языках программирования Java. Основные использованные программные библиотеки:

- а) Owner – библиотека для работы с файлами конфигурации;
- б) Apache Felix – библиотеки реализующая спецификацию OSGi;
- в) Apache Jena – библиотека поддерживающая позволяющая выполнять SPARQL запросы;
- г) Jawampa – библиотека позволяющая работать с WAMP сервером.

Ж.2 Функциональное назначение

Ж.2.1 Назначение программы

Программа предназначена для сбора данных с узлов сенсорной сети, нормализации собранных данных на основе высокогорневых OWL- онтологий и анализа потоковых и исторических данных для выявления различных

ситуаций предметной области. Программа применяется для предоставления доступа внешним системам к гетерогенным данным сенсорной сети по средствам интерфейса прикладного программирования (API) скрывающего различия моделей данных и технологий, используемых узлами сенсорной сети.

Программа реализует следующие функции:

- а) Собирает информацию об узлах сенсорной сети и агрегирует их показания;
- б) Записывает новые показания с узлов в специализированную базу данных для временных данных (Time series database);
- в) Записывает статическую информацию об узлах в RDF-хранилище, которое предоставляет SPARQL-точку доступа;
- г) Выполняет логический вывод (reasoning) над RDF-данными узлов для приведения модели данных к унифицированному формату, используемом программой;
- д) Выполняет анализ текущих и исторических показаний для обнаружения ситуаций предметной области;
- е) Реализует интерактивный графический пользовательский интерфейс в форме веб-приложения.

Программа ЦОД предоставляет доступ внешним системам к текущим показаниям через программный интерфейс. Так же предоставляется возможность добавить свои показания в систему при добавлении драйвера в виде динамических загружаемых модулей (далее - драйверов). Данные драйвера реализуют один интерфейс. Данный интерфейс описывает основные функции необходимые системе для работы с устройством с ограниченными ресурсами. Использование драйверов позволяет агрегировать данных различных типов устройств.

ЦОД хранит данные об устройствах, драйверах и показаниях.

Ж.2.2 Сведения о функциональных ограничениях на применение

Функциональные ограничения на применение отсутствуют.

Ж.3 Описание логической структуры

Ж.3.1 Алгоритм программы

Ж.3.1.1 Этапы алгоритма работы

Работа ЦОД состоит из следующих этапов:

- а) Запуск модулей
 - 1) ожидание запуска модулей, в случае если есть зависимость
 - 2) чтение файла конфигураций
- б) Ожидание конфигурирования системы
- в) Подключение к базам данных
- г) Ожидание запуска драйверов пользователем
- д) Запись описаний устройств и показаний в БД
- е) Ожидание и обработка запросов на получение текущих описаний и показаний устройств

Ж.3.1.2 Запуск модулей

ЦОД состоит из нескольких модулей, при запуске ЦОД, модули стартуют, используя docker-compose.yml файл. Некоторые из модулей зависят от других модулей, поэтому при запуске модуль ожидает завершения запуска модуля от которого он зависит.

Файл конфигурации модуля располагается в корневой директории модуля. Чтение файла производиться с помощью библиотеки Owner, а формат файла соответствует формату Java Properties.

Ж.3.1.3 Ожидание конфигурирования системы

После запуска всех модулей, системе необходимо получить данные об используемом домене в системе, поэтому система переходит в режим ожидания до тех пор, пока пользователь не введет домен.

Ж.3.1.4 Подключение к базам данных

Осуществляется подключение к базе знаний Fuseki. В которой осуществляется хранение описания устройств. Подключение к базе данных происходит с использованием библиотеки Apache Jena.

Ж.3.1.5 Ожидание запуска драйверов

Для начала добавления показаний и устройств в систему необходимо добавить драйвер. Драйвер добавляется пользователем через программный интерфейс ЦОД.

Ж.3.1.6 Запись описаний устройств и показаний в БД

Описания устройств записывается базу знаний Fuseki. Показания при помощи Wamp сервера передаются в МБД, где далее записываются в базу данных Cassandra. Для каждого устройства показания передаются по отдельному топику.

Ж.3.1.7 Ожидание и обработка запросов на получение текущих описаний и показаний устройств

Ожидание и обработка запросов на получение текущих показаний производится с использованием REST сервиса, являющегося одним из модулей ЦОД.

Ж.3.2 Используемые методы

Методы, используемые программным комплексом, указаны при описании алгоритмов в разделе 3.1.

Ж.3.3 Структура программы с описанием функций составных частей и связи между ними

Подсистема ЦОД состоит из следующих модулей:

- а) модуль сбора данных (далее - МСД),
- б) модуль агрегации данных (далее - МАД),
- в) модуль нормализации и анализа данных (далее - МНАД),
- г) модуль интерактивной визуализации и трансляции данных (далее - МИВТД),
- д) модуль работы с хранилищем данных (далее - МБД),
- е) модуль работы с хранилищем семантических метаданных и онтологий (далее - МБЗ)

Состав модулей соответствует пункту 4.3.5.3 Технического задания.

МСД обеспечивает прием и сохранение больших массивов данных от распределенной сети электронных устройств. МАД обеспечивает динамическую классификацию и идентификацию больших массивов данных, генерирующихся в распределенной сети электронных потребительских устройств. МНАД обеспечивает приведение к нормальному виду и препроцессинг больших массивов структурированных, полуструктурных и неструктурированных данных. МИВДТ обеспечивает представление результатов агрегации и анализа.

Ж.3.4 Связи программы с другими программами

Связи программы с другими программами отсутствуют.

Ж.4 Используемые технические средства

В состав используемых технических средств входит:

- а) оперативная память не менее 2GByte/CPU;
- б) дисковое пространство не менее 15-20 GByte/CPU;
- в) модель процессора не хуже Intel Xeon – 2606, 4x2 GHz CPU Cores.

Ж.5 Вызов и загрузка

Загрузка и запуск осуществляется при помощи утилиты docker-compose. Для загрузки необходимо загрузить из директории проекта файл

`docker-compose.yml`. Далее выполнить из этой директории команду `sudo docker-compose pull`. Данная команда скачает все необходимые образы для работы программы. Команда `sudo docker-compose up -d` выполняет запуск программы.

Ж.6 Входные данные

Входными данными программы ЦОД являются файлы конфигурации модулей. Настройка модулей осуществляется через редактирование файла конфигурации расположенного в директории модуля. Файл представляет собой файл в формате Java Properties, где параметры задаются через строки вида [название параметра]=[значение параметра].

Ж.7 Выходные данные

Выходными данными программы ЦОД являются описание и показания электронных потребительских устройств. Выходные данные записываются в базу данных и отправляются в ответ на запросы клиентов к REST API и визуальному интерфейсу.

Приложение З

Текст программы подсистемы ЦОД ЭО ПАП

УТВЕРЖДАЮ

Руководитель проекта

к.т.н., доцент



Д.И. Муромцев

«30» июня 2016 г.

ПОДСИСТЕМА ЦОД ЭО ПО ПАП

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

ЦОД.00001-01 12 01-ЛУ

Ответственный исполнитель



М.А. Колчин

«30» июня 2016 г.

Исполнитель



И.А. Шилин

«30» июня 2016 г.

2016

УТВЕРЖДЕНО

ЦОД.00001-01 12 01-ЛУ

ПОДСИСТЕМА ЦОД ЭО ПАП

Текст программы

ЦОД.00001-01 12 01

Инв. № подл.	Подпись и дата	Взам инв. №	Инв. № дубл.	Подпись и дата

Аннотация

В данном программном документе частично приведен текст программы «Центр обработки данных» (далее - ЦОД), предназначеннай для агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, слабоструктурированных и неструктурных данных.

Исходным языком программирования является Java. А средой выполнения является Java Virtual Machine 1.8 или выше.

Оформление программного документа «Текст программы» произведено по требованиям ЕСПД (ГОСТ 19.401-78).

3.1 Текст программы подсистемы ЦОД на исходном языке

3.1.1 Текст МСД на исходном языке

3.1.1.1 Реализация механизма запуска

```
package ru.semiot.platform.drivers.netatmo.weatherstation;

import java.util.Properties;
import org.apache.felix.dm.DependencyActivatorBase;
import org.apache.felix.dm.DependencyManager;
import org.osgi.framework.BundleContext;
import org.osgi.framework.Constants;
import org.osgi.service.cm.ManagedService;
import ru.semiot.platform.deviceproxyservice.api.drivers.DeviceDriver;
import ru.semiot.platform.deviceproxyservice.api.drivers.DeviceDriverManager;

public class Activator extends DependencyActivatorBase {

    @Override
    public void init(BundleContext bc, DependencyManager manager) throws
        Exception {
        Properties properties = new Properties();
        properties.setProperty(Constants.SERVICE_PID, Keys.DRIVER_PID);

        manager.add(createComponent()
            .setInterface(new String[]{DeviceDriver.class.getName(),
                ManagedService.class.getName()}, properties)
            .setImplementation(DeviceDriverImpl.class)
            .add(createServiceDependency()
                .setService(DeviceDriverManager.class)
                .setRequired(true)))
            .add(createConfigurationDependency().setPid(Keys.DRIVER_PID)));
    }
}
```

3.1.1.2 Реализация работы с менеджером драйверов

```
package ru.semiot.platform.drivers.netatmo.weatherstation;

import org.osgi.service.cm.ConfigurationException;
import org.osgi.service.cm.ManagedService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import ru.semiot.platform.deviceproxyservice.api.drivers.Configuration;
```

```

import ru.semiot.platform.deviceproxyservice.api.drivers.Device;
import ru.semiot.platform.deviceproxyservice.api.drivers.DeviceDriver;
import ru.semiot.platform.deviceproxyservice.api.drivers.DeviceDriverManager;
import ru.semiot.platform.deviceproxyservice.api.drivers.DriverInformation;

import java.net.URI;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Dictionary;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.ScheduledFuture;
import java.util.concurrent.TimeUnit;

public class DeviceDriverImpl implements DeviceDriver, ManagedService {

    private static final Logger logger =
        LoggerFactory.getLogger(DeviceDriverImpl.class);

    private final Map<String, Device> devicesMap
        = Collections.synchronizedMap(new HashMap<>());
    private final Map<String, WeatherStationObservation> observationsMap
        = Collections.synchronizedMap(new HashMap<>());
    private final String driverName = "Netatmo.com (only temperature)";
    private final Configuration configuration = new Configuration();
    private final DriverInformation info = new DriverInformation(
        Keys.DRIVER_PID,
        URI.create("https://raw.githubusercontent.com/semitoproject/semito-drivers/"
            +
            "master/netatmo-weatherstation/" +
            "src/main/resources/ru/semito/platform/drivers/netatmo/weatherstation/protot
            +
            "NetatmoWeatherStationOutdoorModule"));
}

private volatile DeviceDriverManager deviceManager;

private ScheduledExecutorService scheduler;
private List<ScheduledFuture> handles = null;
private List<Configuration> configurations;
private NetatmoAPI netAtmoAPI;
private List<Integer> countsRepeatableProperties;

```

```

public void start() {
    logger.info("{} started!", driverName);
    deviceManager.registerDriver(info);

    handles = new ArrayList<>();
    this.scheduler =
        Executors.newScheduledThreadPool(countsRepeatableProperties.size());
    logger.debug("Try to start {} pullers", countsRepeatableProperties.size());
    for (Configuration cfg : configurations) {
        handles.add(startPuller(cfg));
    }
    logger.debug("All pullers started");
}

public void stop() {
    logger.debug("Try to stop {} pullers", handles.size());
    for (ScheduledFuture handle : handles) {
        stopPuller(handle);
    }
    logger.debug("All pullers stoped");
    handles = null;
    scheduler.shutdown();
    try {
        scheduler.awaitTermination(1, TimeUnit.MINUTES);
    } catch (InterruptedException ex) {
        logger.warn(ex.getMessage(), ex);
    }
    scheduler.shutdownNow();
    logger.debug("Scheduler stoped");
    logger.info("{} stopped!", driverName);
}
}

@Override
public void updated(Dictionary dictionary) throws ConfigurationException {
    synchronized (this) {
        if (dictionary != null) {
            if (!configuration.isConfigured()) {
                logger.debug("Configuration got");
                try {
                    configuration.putAll(dictionary);
                    Configuration commonConfiguration = getCommonConfiguration();
                    netAtmoAPI = new NetatmoAPI(
                        commonConfiguration.getAsString(Keys.CLIENT_APP_ID),
                        commonConfiguration.getAsString(Keys.CLIENT_SECRET));
                    checkConnection(
                        commonConfiguration.getAsString(Keys.USERNAME),

```

```

        commonConfiguration .getAsString (Keys .PASSWORD) ) ;
countsRepeatableProperties =
        getCountsRepeatableProperties (Keys .AREA) ;
configurations = getConfigurations (countsRepeatableProperties) ;
configuration .setConfigured () ;
logger .info ("Received configuration is correct !") ;
} catch ( ConfigurationException ex) {
    configuration .clear () ;
    throw ex;
}
} else {
    logger .warn ("Driver is already configured ! Ignoring it ") ;
}
} else {
    logger .debug ("Configuration is empty . Skipping it ") ;
}
}
}

public void checkConnection (String user , String pass) throws
ConfigurationException {
logger .debug ("Try to authenticate with server ") ;
if (netAtmoAPI .authenticate (user , pass)) {
    logger .info ("Successfully authenticated !") ;
} else {
    logger .error ("Couldn 't authenticate !") ;
    throw new ConfigurationException (user + ":" + pass , "Login or password is
incorrect ");
}
}

private List < Configuration > getConfigurations (List < Integer > counts) throws
ConfigurationException {
logger .debug ("Try to get repeatable configuration for each puller ") ;
List < Configuration > conf = new ArrayList <> () ;
for ( int i : counts) {
    Configuration cfg = getAreaConfiguration (i) ;
    cfg .put (Keys .ONLY_NEW_OBS , configuration .get (Keys .ONLY_NEW_OBS)) ;
    conf .add (cfg) ;
}
return conf ;
}

public Configuration getConfiguration () {
    return configuration ;
}

```

```

public boolean isRegistered(String id) {
    return devicesMap.containsKey(id);
}

public Device getDeviceById(String id) {
    return devicesMap.get(id);
}

public int getNumberOfRegisteredDevices() {
    return devicesMap.size();
}

public Set<String> getIDsOfRegisteredDevices() {
    return devicesMap.keySet();
}

public WeatherStationObservation getObservation(String deviceId, String type)
{
    return observationsMap.get(toObsKey(deviceId, type));
}

@Override
public String getDriverName() {
    return driverName;
}

public void updateDevice(Device newDevice) {
    devicesMap.put(newDevice.getId(), newDevice);

    //TODO: Trigger update in the database
}

public void registerDevice(Device device) {
    devicesMap.put(device.getId(), device);

    deviceManager.registerDevice(info, device);
}

public void publishNewObservation(WeatherStationObservation observation) {
    //Replace previous observation
    String deviceId =
        observation.getProperty(NetatmoDeviceProperties.DEVICE_ID);
    String type =
        observation.getProperty(NetatmoDeviceProperties.OBSERVATION_TYPE);
    observationsMap.put(toObsKey(deviceId, type), observation);
}

```

```

        deviceManager.registerObservation(devicesMap.get(deviceId), observation);
    }

public ScheduledFuture startPuller(Configuration config) {
    logger.debug("Try to start puller!");
    logger.debug("Config is " + config.toString());
    ScheduledPuller puller = new ScheduledPuller(this, config, netAtmoAPI);

    logger.debug("Try to schedule polling. Starts in {}min with interval {}min
                 with configuration [{}]",
                configuration.get(Keys.POLLING_START_PAUSE),
                configuration.get(Keys.POLLING_INTERVAL),
                config.toString());
}

ScheduledFuture handle = this.scheduler.scheduleAtFixedRate(
    puller,
    configuration.getAsLong(Keys.POLLING_START_PAUSE),
    configuration.getAsLong(Keys.POLLING_INTERVAL),
    TimeUnit.MINUTES);

logger.debug("Puller started!");
return handle;
}

public void stopPuller(ScheduledFuture handle) {
    logger.debug("Try to stop puller!");
    if (handle == null) {
        return;
    }
    handle.cancel(true);
    logger.debug("Puller stoped!");
}

private String toObsKey(String deviceId, String type) {
    return deviceId + "-" + type;
}

private List<Integer> getCountsRepeatableProperties(String propPrefix) throws
    ConfigurationException {
    logger.debug("Try to get count of repeatable property \\"{}\\\"", propPrefix);
    List<Integer> counts = new ArrayList<>();
    int index;

    for (String key : configuration.keySet()) {
        if (key.contains(propPrefix) && !counts.contains(

```

```

        index = Integer.parseInt(key.substring(0, key.indexOf("." +
                propPrefix)))) {
            counts.add(index);
        }
    }
    if (counts.isEmpty()) {
        logger.error("Bad repeatable configuration! Did not find a repeatable
                property");
        throw new ConfigurationException(propPrefix, "Did not find a repeatable
                property");
    }
    return counts;
}

private Configuration getCommonConfiguration() throws ConfigurationException {
    logger.debug("Try to get common configuration");
    Configuration config = new Configuration();
    try {
        //Put only needed properties
        config.put(Keys.CLIENT_APP_ID, configuration.get(Keys.CLIENT_APP_ID));
        config.put(Keys.CLIENT_SECRET, configuration.get(Keys.CLIENT_SECRET));
        config.put(Keys.USERNAME, configuration.get(Keys.USERNAME));
        config.put(Keys.PASSWORD, configuration.get(Keys.PASSWORD));
        config.put(Keys.POLLING_START_PAUSE,
                configuration.get(Keys.POLLING_START_PAUSE));
        config.put(Keys.POLLING_INTERVAL,
                configuration.get(Keys.POLLING_INTERVAL));
        if (configuration.getAsString(Keys.ONLY_NEW_OBS).equalsIgnoreCase("true")
                ||
                configuration.getAsString(Keys.ONLY_NEW_OBS).equalsIgnoreCase("false"))
        {
            config.put(Keys.ONLY_NEW_OBS, configuration.get(Keys.ONLY_NEW_OBS));
        } else {
            throw new java.lang.NullPointerException();
        }
    } catch (java.lang.NullPointerException ex) {
        logger.error("Bad common configuration! Can not extract fields");
        throw new ConfigurationException("Common property", "Can not extract
                fields", ex);
    }
    return config;
}

private Configuration getAreaConfiguration(int area) throws
        ConfigurationException {
    logger.debug("Try to get configuration for {} area", area);
}

```

```

Configuration config = new Configuration();
double lon_ne, lat_ne, lon_sw, lat_sw;
try {

    lon_ne = Double.parseDouble(configuration.getAsString(area + "." +
        Keys.AREA + ".1.longitude"));
    lat_ne = Double.parseDouble(configuration.getAsString(area + "." +
        Keys.AREA + ".1.latitude"));
    lon_sw = Double.parseDouble(configuration.getAsString(area + "." +
        Keys.AREA + ".2.longitude"));
    lat_sw = Double.parseDouble(configuration.getAsString(area + "." +
        Keys.AREA + ".2.latitude"));
    if (lon_ne > 180 || lon_ne < -180 || lon_sw > 180 || lon_sw < -180
        || lat_ne > 90 || lat_ne < -90 || lat_sw > 90 || lat_sw < -90
        || lon_ne < lon_sw && lat_ne > lat_sw || lon_ne > lon_sw && lat_ne <
        lat_sw) {
        throw new java.lang.NullPointerException();
    }
    if (lon_ne < lon_sw && lat_ne < lat_sw) {
        logger.debug("Swap values");
        config.put(Keys.LONGITUDE_NORTH_EAST, String.valueOf(lon_sw));
        config.put(Keys.LATITUDE_NORTH_EAST, String.valueOf(lat_sw));
        config.put(Keys.LONGITUDE_SOUTH_WEST, String.valueOf(lon_ne));
        config.put(Keys.LATITUDE_SOUTH_WEST, String.valueOf(lat_ne));
    } else {
        config.put(Keys.LONGITUDE_NORTH_EAST, String.valueOf(lon_ne));
        config.put(Keys.LATITUDE_NORTH_EAST, String.valueOf(lat_ne));
        config.put(Keys.LONGITUDE_SOUTH_WEST, String.valueOf(lon_sw));
        config.put(Keys.LATITUDE_SOUTH_WEST, String.valueOf(lat_sw));
    }
} catch (java.lang.NullPointerException ex) {
    logger.error("Bad repeatable configuration! Can not extract field of
        property {}.{}, Keys.AREA, area);
    throw new ConfigurationException(Keys.AREA + "." + area, "Can not extract
        field of repeatable property", ex);
}
return config;
}
}

```

3.1.2 Текст МАД на исходном языке

3.1.2.1 Реализация механизма запуска

```
package ru.semiot.platform.deviceproxyservice.manager;
```

```

import org.apache.felix.dm.DependencyActivatorBase;
import org.apache.felix.dm.DependencyManager;
import org.osgi.framework.BundleContext;
import org.osgi.framework.Constants;
import org.osgi.service.cm.ManagedService;
import ru.semiot.platform.deviceproxyservice.api.drivers.DeviceDriverManager;
import ru.semiot.platform.deviceproxyservice.api.manager.DirectoryService;

import java.util.Properties;

public class Activator extends DependencyActivatorBase {

    private static final String PID =
        "ru.semiot.platform.deviceproxyservice.manager";

    @Override
    public void init(BundleContext bc, DependencyManager manager) throws
        Exception {
        Properties properties = new Properties();
        properties.setProperty(Constants.SERVICE_PID, PID);

        manager.add(createComponent()
            .setInterface(new String[]{(
                DeviceDriverManager.class.getName(),
                ManagedService.class.getName())},
            properties)
            .setImplementation(DriverManagerImpl.class)
            .add(createServiceDependency()
                .setService(DirectoryService.class)
                .setRequired(true))
            .add(createConfigurationDependency().setPid(PID)));
    }

}

```

3.1.2.2 Реализация менеджера драйверов

```

package ru.semiot.platform.deviceproxyservice.manager;

import com.github.jsonldjava.utils.JsonUtils;
import org.apache.jena.rdf.model.Model;
import org.osgi.framework.BundleContext;
import org.osgi.framework.InvalidSyntaxException;
import org.osgi.framework.ServiceReference;
import org.osgi.service.cm.ConfigurationException;
import org.osgi.service.cm.ManagedService;
import org.slf4j.Logger;

```

```

import org.slf4j.LoggerFactory;
import ru.semiot.common.rdf.ModelJsonLdUtils;
import ru.semiot.platform.deviceproxyservice.api.drivers.Command;
import
    ru.semiot.platform.deviceproxyservice.api.drivers.CommandExecutionException;
import ru.semiot.platform.deviceproxyservice.api.drivers.CommandResult;
import ru.semiot.platform.deviceproxyservice.api.drivers.Configuration;
import
    ru.semiot.platform.deviceproxyservice.api.drivers.ControllableDeviceDriver;
import ru.semiot.platform.deviceproxyservice.api.drivers.Device;
import ru.semiot.platform.deviceproxyservice.api.drivers.DeviceDriverManager;
import ru.semiot.platform.deviceproxyservice.api.drivers.DeviceProperties;
import ru.semiot.platform.deviceproxyservice.api.drivers.DriverInformation;
import ru.semiot.platform.deviceproxyservice.api.drivers.Observation;
import ru.semiot.platform.deviceproxyservice.api.drivers.RDFTemplate;
import ru.semiot.platform.deviceproxyservice.api.manager.CommandFactory;
import ru.semiot.platform.deviceproxyservice.api.manager.DirectoryService;
import ws.wamp.jawampa.WampClient;

import java.io.IOException;
import java.util.Collection;
import java.util.Dictionary;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.SynchronousQueue;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;

public class DriverManagerImpl implements DeviceDriverManager, ManagedService {

    private static final Logger logger =
        LoggerFactory.getLogger(DriverManagerImpl.class);
    private static final String FRAME_PATH_PREFIX =
        "/ru/semito/platform/deviceproxyservice/manager/";
    private static final String OBSERVATION_FRAME_PATH =
        FRAME_PATH_PREFIX + "Observation-frame.jsonld";
    private static final String COMMANDRESULT_FRAME_PATH =
        FRAME_PATH_PREFIX + "CommandResult-frame.jsonld";
    private static final String SYSTEM_FRAME_PATH = FRAME_PATH_PREFIX +
        "System-frame.jsonld";
    private static final String VAR_SYSTEM_ID = "${SYSTEM_ID}";
    private static final String VAR_PROCESS_ID = "${PROCESS_ID}";
    private static final String TOPIC_OBSERVATIONS =
        "${SYSTEM_ID}.observations.${SENSOR_ID}";
    private static final String TOPIC_COMMANDRESULT =
        "${SYSTEM_ID}.commandresults.${PROCESS_ID}";
    private final Configuration configuration = new Configuration();
}

```

```

private final Object observationFrame;
private final Object systemFrame;
private final Object commandResultFrame;

//Injected by Dependency Manager
private BundleContext bundleContext;

//Injected by Dependency Manager
private DirectoryService directoryService;

private ExecutorService executor = new ThreadPoolExecutor(10, 10, 0L,
    TimeUnit.MILLISECONDS,
    new SynchronousQueue(), new ThreadPoolExecutor.CallerRunsPolicy());

public DriverManagerImpl() {
    //Loading JSONLD frame for observations
    Object frame = null;
    try {
        frame = JsonUtils.fromInputStream(
            this.getClass().getResourceAsStream(OBSERVATION_FRAME_PATH));
    } catch (Throwable e) {
        logger.error(e.getMessage(), e);
    }
    this.observationFrame = frame;

    //Loading JSONLD frame for systems
    try {
        frame = JsonUtils.fromInputStream(
            this.getClass().getResourceAsStream(SYSTEM_FRAME_PATH));
    } catch (Throwable e) {
        logger.error(e.getMessage(), e);
    }
    this.systemFrame = frame;

    try {
        frame = JsonUtils.fromInputStream(
            this.getClass().getResourceAsStream(COMMANDRESULT_FRAME_PATH));
    } catch (Throwable e) {
        logger.error(e.getMessage(), e);
    }
    this.commandResultFrame = frame;
}

public void start() {
    logger.info("Device Manager is starting...");
    try {

```

```

WAMPClient
    .getInstance()
    .init(configuration.getAsString(Keys.WAMP_URI),
          configuration.getAsString(Keys.WAMP_REALM),
          configuration.getAsInteger(Keys.WAMP_RECONNECT),
          configuration.getAsString(Keys.WAMP_LOGIN),
          configuration.getAsString(Keys.WAMP_PASSWORD))
    .subscribe(
        (WampClient.State newState) -> {
            if (newState instanceof WampClient.ConnectedState) {
                logger.info("Connected to {}", configuration.get(Keys.WAMP_URI));
            } else if (newState instanceof WampClient.DisconnectedState) {
                logger.info("Disconnected from {}. Reason: {}", configuration.get(Keys.WAMP_URI),
                            ((WampClient.DisconnectedState) newState).disconnectReason());
            } else if (newState instanceof WampClient.ConnectingState) {
                logger.info("Connecting to {}", configuration.get(Keys.WAMP_URI));
            }
        });
    logger.info("Device Proxy Service Manager started!");
} catch (Throwable ex) {
    logger.error(ex.getMessage(), ex);
    try {
        WAMPClient.getInstance().close();
    } catch (IOException ex1) {
        logger.error(ex1.getMessage(), ex1);
    }
}
}

public void stop() {
try {
    WAMPClient.getInstance().close();
    executor.shutdown();
    executor.awaitTermination(30, TimeUnit.SECONDS);
    executor.shutdownNow();
} catch (Throwable ex) {
    logger.error(ex.getMessage(), ex);
}
logger.info("Device Proxy Service Manager stopped!");
}

```

@Override

```

public void updated(Dictionary<String, Object> dictionary) throws ConfigurationException {
    synchronized (this) {
        if (dictionary != null) {
            if (!configuration.isConfigured()) {
                // default values
                configuration.put(Keys.WAMP_URI, "ws://wamprouter:8080/ws");
                configuration.put(Keys.WAMP_REALM, "realm1");
                configuration.put(Keys.WAMP_RECONNECT, "15");
                configuration.put(Keys.WAMP_LOGIN, "internal");
                configuration.put(Keys.WAMP_PASSWORD, "internal");
                configuration.put(Keys.TOPIC_NEWANDOBSERVING,
                        "ru.semiot.devices.newandobserving");
                configuration.put(Keys.TOPIC_INACTIVE, "ru.semiot.devices.turnoff");
                configuration.put(Keys.FUSEKI_PASSWORD, "pw");
                configuration.put(Keys.FUSEKI_USERNAME, "admin");
                configuration.put(Keys.FUSEKI_UPDATE_URL,
                        "http://triplestore:3030/blazegraph/sparql");
                configuration.put(Keys.FUSEKI_QUERY_URL,
                        "http://triplestore:3030/blazegraph/sparql");
                configuration.put(Keys.FUSEKI_STORE_URL,
                        "http://triplestore:3030/blazegraph/sparql");
                configuration.put(Keys.PLATFORM_DOMAIN, "http://localhost");
                configuration.put(Keys.PLATFORM_SYSTEMS_PATH, "systems");
                configuration.put(Keys.PLATFORM_SUBSYSTEM_PATH, "subsystems");
                configuration.put(Keys.PLATFORM_PROCESS_PATH, "processes");

                configuration.putAll(dictionary);

                //Composite properties
                configuration.put(Keys.PLATFORM_SYSTEMS_URI_PREFIX,
                        configuration.get(Keys.PLATFORM_DOMAIN) + "/"
                        + configuration.get(Keys.PLATFORM_SYSTEMS_PATH));
                configuration.put(Keys.PLATFORM_PROCESS_URI_PREFIX,
                        configuration.get(Keys.PLATFORM_SYSTEMS_URI_PREFIX) + "/{{"
                        + DeviceProperties.DEVICE_ID + "}}/"
                        + configuration.get(Keys.PLATFORM_PROCESS_PATH));

                configuration.setConfigured();
            }
            logger.debug("Manager was configured");
        } else {
            logger.warn("Manager is already configured! Ignoring it");
        }
    } else {
        logger.debug("Configuration is empty. Skipping it");
    }
}

```

```

        }

    }

@Override
public void registerDriver(DriverInformation info) {
    directoryService.loadDevicePrototype(info.getPrototypeUri());
}

@Override
public void updateDevice(Device device) {
    //TODO: Here we should update device's states.
}

@Override
public void registerDevice(DriverInformation info, Device device) {
    executor.execute(() -> {
        logger.debug("Device [{}] is being registered", device.getId());

        try {
            /**
             * Resolve common variables, e.g. platform's domain name.
             */
            final Model description = device.toDescriptionAsModel(configuration);

            boolean isAdded = directoryService.addNewDevice(info, device,
                description);

            if (isAdded) {
                logger.debug("Device [{}] added to database!", device.getId());
                long start = System.currentTimeMillis();
                String message = JsonUtils.toString(
                    ModelJsonLdUtils.toJsonLdCompact(description, systemFrame));
                WAMPClient.getInstance().publish(
                    getConfiguration().getAsString(Keys.TOPIC_NEWANDOBSERVING),
                    message)
                    .subscribe(WAMPClient.onError());
                long end = System.currentTimeMillis();
                logger.info("Device [{}] was registered in {} ms!", device.getId(),
                    end - start);
            } else {
                logger.warn("Device [{}] was not added in database!", device.getId());
            }
        } catch (Throwable ex) {
            logger.error(ex.getMessage(), ex);
        }
    });
}

```

```

}

@Override
public void registerObservation(Device device, Observation observation) {
    executor.execute(() -> {
        try {
            logger.debug("Observation [Device ID={}] is being registered",
                    device.getId());
            // TODO: There's no guarantee that WAMPClient is connected.
            String message = observation.getRDFTemplate().resolveToString(
                    observation.getProperties(), device.getProperties(),
                    configuration);
            WAMPClient.getInstance().publish(
                TOPIC_OBSERVATIONS
                    .replace("${SYSTEM_ID}", device.getId())
                    .replace("${SENSOR_ID}",
                            observation.getProperty(DeviceProperties.SENSOR_ID)),
                    message)
                    .subscribe(WAMPClient.onError());
            logger.info("Observation [Device ID={}] was registered",
                    device.getId());
        } catch (Throwable ex) {
            logger.error(ex.getMessage(), ex);
        }
    });
}

@Override
public void removeDataOfDriverFromFuseki(String pid) {
    // directoryService.removeDataOfDriver(pid);
}

@Override
public void registerCommand(Device device, CommandResult result) {
    executor.execute(() -> {
        try {
            Model model = result.toRDFAsModel(configuration);
            //TODO: There's no guarantee that WAMPClient is connected?
            WAMPClient.getInstance().publish(
                TOPIC_COMMANDRESULT
                    .replace(VAR_SYSTEM_ID, device.getId())
                    .replace(VAR_PROCESS_ID,
                            result.get(DeviceProperties.PROCESS_ID)),
                    JsonUtils.toString(ModelJsonLdUtils.toJsonLdCompact(model,
                            commandResultFrame)))
                    .subscribe(WAMPClient.onError());
        } catch (Throwable e) {

```

```

        logger . error ( e . getMessage ( ) , e );
    }
}

@Override
public Model executeCommand ( String deviceId , Model commandModel )
    throws CommandExecutionException {
try {
    String pid = directoryService . findDriverPidByDeviceId ( deviceId );

    if ( pid != null ) {
        try {
            Collection services = bundleContext . getServiceReferences (
                ControllableDeviceDriver . class , "( service . pid=" + pid + " )");
            if ( ! services . isEmpty ( ) ) {
                logger . debug ( " Driver [{} ] found ! " , pid );
                ServiceReference < ControllableDeviceDriver > reference =
                    ( ServiceReference ) services . iterator ( ) . next ( );
                ControllableDeviceDriver driver =
                    bundleContext . getService ( reference );
                String commandId = CommandFactory . extractCommandId ( commandModel );
                RDFTemplate template = driver . getRDFTemplate ( commandId );
                Command command = CommandFactory . buildCommand ( commandModel ,
                    template );
                CommandResult result = driver . executeCommand ( command );
                return result . toRDFAsModel ( configuration );
            } else {
                throw CommandExecutionException . driverNotFound ( );
            }
        } catch ( InvalidSyntaxException e ) {
            throw CommandExecutionException . driverNotFound ( e );
        }
    } else {
        throw CommandExecutionException . driverNotFound ( );
    }
} catch ( Throwable e ) {
    if ( e instanceof CommandExecutionException ) {
        throw e;
    } else {
        throw CommandExecutionException . badCommand ( e );
    }
}
}
}

```

```

    public Configuration getConfiguration() {
        return configuration;
    }

}

```

3.1.3 Текст МНАД на исходном языке

3.1.3.1 Реализация представления показаний

```

package ru.semiot.platform.deviceproxyservice.api.drivers;

import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.riot.Lang;
import org.apache.jena.riot.RDFLanguages;

import java.io.StringReader;
import java.io.StringWriter;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Objects;

public abstract class Observation {

    private final Map<String, Object> properties = new HashMap<>();

    /**
     * @param timestamp number of seconds or milliseconds since the Unix Epoch
     */
    public Observation(String deviceId, String sensorId, String timestamp) {
        properties.put(DeviceProperties.DEVICE_ID, deviceId);
        properties.put(DeviceProperties.SENSOR_ID, sensorId);
        properties.put(DeviceProperties.OBSERVATION_TIMESTAMP, timestamp);

        //Timestamp to ISO 8601 date and time
        String dateTime = null;
        if (timestamp.length() > 10) {
            dateTime = DateTimeFormatter.ISO_OFFSET_DATE_TIME
                .withZone(ZoneOffset.UTC)
                .format(Instant.ofEpochMilli(Long.valueOf(timestamp)));
        }
    }
}

```

```

    } else {
        dateTime = DateTimeFormatter.ISO_OFFSET_DATE_TIME
            .withZone(ZoneOffset.UTC)
            .format(Instant.ofEpochSecond(Long.valueOf(timestamp)));
    }
    properties.put(DeviceProperties.OBSERVATION_DATETIME, dateTime);
}

public Map<String, Object> getProperties() {
    return properties;
}

public String getProperty(String name) {
    return properties.get(name).toString();
}

public abstract RDFTemplate getRDFTemplate();

public Model toObservationAsModel(Map<String, Object> properties,
    Configuration configuration) {
    StringReader descr = getRDFTemplate().resolveToReader(
        this.properties, properties, configuration);

    Model model = ModelFactory.createDefaultModel().read(descr, null,
        RDFLanguages.strLangTurtle);

    return model;
}

public String toObservationAsString(Map<String, Object> properties,
    Configuration configuration,
    Lang lang) {
    StringWriter writer = new StringWriter();

    toObservationAsModel(properties, configuration).write(writer,
        lang.getName());

    return writer.toString();
}

public boolean equalsIgnoreTimestamp(Object o) {
    if (o == this) {
        return true;
    }

    if (!(o instanceof Observation)) {

```

```

        return false;
    }
    Observation m = (Observation) o;
    if (m.properties.size() != this.properties.size()) {
        return false;
    }

    try {
        Iterator<Map.Entry<String, Object>> i =
            this.properties.entrySet().iterator();
        while (i.hasNext()) {
            Map.Entry<String, Object> e = i.next();
            String key = e.getKey();
            Object value = e.getValue();
            if (value == null) {
                if (!(m.properties.get(key) == null &&
                    m.properties.containsKey(key))) {
                    return false;
                }
            } else {
                if (!value.equals(m.properties.get(key))) {
                    if (!key.equals(DeviceProperties.OBSERVATION_TIMESTAMP)
                        && !key.equals(DeviceProperties.OBSERVATION_DATETIME)) {
                        return false;
                    }
                }
            }
        }
    } catch (ClassCastException | NullPointerException unused) {
        return false;
    }

    return true;
}

@Override
public boolean equals(Object obj) {
    if (obj == null) {
        return false;
    }

    if (obj instanceof Observation) {
        Observation that = (Observation) obj;

        return this.properties.equals(that.properties);
    }
}

```

```

        return false;
    }

    @Override
    public int hashCode() {
        int hash = 3;
        hash = 59 * hash + Objects.hashCode(this.properties);
        return hash;
    }

}

```

3.1.4 Текст МИВТД на исходном языке

3.1.4.1 Реализация страницы авторизации

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>SemIoT Platform | Login</title>
    <link rel="stylesheet"
        href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700"
        type="text/css">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
        rel="stylesheet">
    <link
        href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.6/css/bootstrap.css"
        rel="stylesheet">
    <link rel="stylesheet"
        href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-material-design/0.5.9/css/material.css"
        rel="stylesheet">
<style>
    html {
        height: 100%;
        width: 100%;
        position: relative;
    }
    body {
        margin-bottom: 60px;
        width: 100%;
    }
    a {
        cursor: pointer;
    }
    h3 {

```

```

        text-align: center;
    }
    .well {
        background-color: #fff;
        padding: 19px;
        margin-bottom: 20px;
        -webkit-box-shadow: 0 8px 17px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0
            rgba(0, 0, 0, 0.19);
        box-shadow: 0 8px 17px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0,
            0, 0, 0.19);
        border-radius: 2px;
        border: 0;
    }
    body .container .well {
        padding: 30px;
    }
    .login-container {
        margin-left: 25%;
        margin-top: 5%;
    }
    form {
        padding: 0 15px;
    }
    form button.btn.btn-primary {
        display: block;
        margin: 10px auto 0 auto;
    }
    .form-group.label-floating label.control-label,
    .form-group.placeholder label.control-label {
        top: -12px;
    }
</style>
</head>
<body>
<div class="container">
    <div class="col-md-6 login-container">
        <div class="well bs-component">
            <h2>SemIoT Platform</h2>
            <h4>Authorization</h4>
            <form action="j_security_check" method="post"
                  class="form-horizontal">
                <div class="form-group label-floating">
                    <label for="username"
                          class="control-label">Username</label>
                    <input type="text" class="form-control" id="username"
                           name="j_username" autofocus>

```

```

        <span class="help-block">Enter your
            username</code></span>
        </div>
        <div class="form-group label-floating">
            <label for="password"
                class="control-label">Password</label>
            <input type="password" class="form-control"
                id="password" name="j_password">
            <span class="help-block">Enter your
                password</code></span>
        </div>
        <button type="submit" class="btn
            btn-primary">Submit</button>
    </form>
</div>
</div>
<script
    src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>

<script
    src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
<script
    src="https://fezvrasta.github.io/bootstrap-material-design/dist/js/material.min.js"></script>
<script>$ .material .init ();</script>
</body>
</html>

```

3.1.5 Текст МБД на исходном языке

3.1.5.1 Реализация механизма запуска

```

package ru.semiot.services.tsdbservice;

import static ru.semiot.services.tsdbservice.ServiceConfig.CONFIG;

import org.aeonbits.owner.event.RollbackBatchException;
import org.aeonbits.owner.event.RollbackOperationException;
import org.aeonbits.owner.event.TransactionalPropertyChangeListener;
import org.eclipse.jetty.server.Server;
import org.glassfish.jersey.jetty.JettyHttpContainerFactory;
import org.glassfish.jersey.server.ResourceConfig;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import ru.semiot.services.tsdbservice.rest.ApplicationConfig;
import ru.semiot.services.tsdbservice.wamp.NewDeviceListener;

```

```

import ru.semiot.services.tsdbservice.wamp.WAMPClient;
import ws.wamp.jawampa.WampClient;

import java.beans.PropertyChangeEvent;
import java.io.IOException;
import java.net.URI;

import javax.ws.rs.core.UriBuilder;

public class Launcher {

    private static final Logger logger = LoggerFactory.getLogger(Launcher.class);

    private static final long TIMEOUT = 5000;

    public static void main(String[] args) {
        Launcher launcher = new Launcher();
        launcher.run();
    }

    private void run() {
        boolean isConnected = false;
        while (!isConnected) {
            try {
                TSDBClient.getInstance().init();
                logger.info("Connected to the tsdb");
                isConnected = true;
            } catch (Exception e) {
                logger.warn(e.getMessage());
                logger.warn("Can't connect to the tsdb! Retry in {}ms", TIMEOUT);
                try {
                    Thread.sleep(TIMEOUT);
                } catch (InterruptedException ex) {
                    logger.error(ex.getMessage());
                }
            }
        }
        try {

            CONFIG.addPropertyChangeListener("ru.semiot.platform.wamp_password",
                new TransactionalPropertyChangeListenerImpl());

            URI uri = UriBuilder.fromUri("http://0.0.0.0/").port(8787).build();
            ResourceConfig resourceConfig = new ApplicationConfig();

```

```

Server jettyServer = JettyHttpContainerFactory.createServer(uri,
    resourceConfig);

try {
    jettyServer.start();
    jettyServer.join();
} finally {
    jettyServer.destroy();
}
} catch (Exception e) {
    logger.error(e.getMessage(), e);
} finally {
    try {
        TSDBClient.getInstance().stop();
        WAMPClient.getInstance().close();
    } catch (IOException ex) {
        logger.error(ex.getMessage(), ex);
    }
}
}

private class TransactionalPropertyChangeListenerImpl
    implements TransactionalPropertyChangeListener {

    @Override
    public void beforePropertyChange(PropertyChangeEvent event)
        throws RollbackOperationException, RollbackBatchException {}

    @Override
    public void propertyChange(PropertyChangeEvent evt) {
        try {
            WAMPClient.getInstance().init().subscribe((WampClient.State newState)
                -> {
                if (newState instanceof WampClient.ConnectedState) {
                    logger.info("Connected to {}", CONFIG.wampUri());

                    try {
                        NewDeviceListener newDeviceListener = new NewDeviceListener();
                        WAMPClient.getInstance()
                            .subscribe(CONFIG.topicsSubscriber())
                            .subscribe(newDeviceListener);

                        // newDeviceListener.loadDeviceTopicsAndSubscribe();
                    } catch (Throwable ex) {
                        logger.debug(ex.getMessage(), ex);
                    }
                }
            });
        }
    }
}

```

```
        } else if (newState instanceof WampClient.DisconnectedState) {
            logger.info("Disconnected from {}", CONFIG.wampUri());
        } else if (newState instanceof WampClient.ConnectingState) {
            logger.debug("Connecting to {}", CONFIG.wampUri());
        }
    });
} catch (Throwable ex) {
    logger.error(ex.getMessage(), ex);
}
}
```

3.1.6 Текст МБЗ на исходном языке

3.1.6.1 Реализация работы с базой знаний

```
package ru.semiot.platform.deviceproxyservice.directory;

import org.apache.jena.atlas.web.auth.HttpAuthenticator;
import org.apache.jena.atlas.web.auth.SimpleAuthenticator;
import org.apache.jena.query.DatasetAccessorFactory;
import org.apache.jena.query.Query;
import org.apache.jena.query.QueryExecutionFactory;
import org.apache.jena.query.QueryFactory;
import org.apache.jena.query.ResultSet;
import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.update.UpdateExecutionFactory;
import org.apache.jena.update.UpdateFactory;
import org.apache.jena.update.UpdateRequest;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import ru.semiot.platform.deviceproxyservice.api.drivers.Configuration;
import rx.Observer;
import rx.subjects.PublishSubject;

import java.util.List;
import java.util.concurrent.TimeUnit;

public class RDFStore {

    private static final Logger logger = LoggerFactory.getLogger(RDFStore.class);
    private final HttpAuthenticator httpAuthenticator;
    private final Configuration configuration;
```

```

private final PublishSubject<Model> ps = PublishSubject.create();

public RDFStore(Configuration configuration) {
    this.configuration = configuration;
    httpAuthenticator = new SimpleAuthenticator(
        configuration.getString(Keys.TRIPLESTORE_USERNAME),
        configuration.getString(Keys.TRIPLESTORE_PASSWORD).toCharArray());
}

ps.buffer(
    configuration.getInteger(Keys.STORE_OPERATION_BUFFERIDLE),
    TimeUnit.SECONDS,
    configuration.getInteger(Keys.STORE_OPERATION_BUFFERSIZE))
    .subscribe(new BatchUploader());
}

public void save(Model model) {
    ps.onNext(model);
}

public void save(String graphUri, Model model) {
    DatasetAccessorFactory
        .createHTTP(configuration.getString(Keys.TRIPLESTORE_STORE_URL),
            httpAuthenticator)
        .add(graphUri, model);
}

public ResultSet select(String query) {
    return select(QueryFactory.create(query));
}

public ResultSet select(Query query) {
    Query select = QueryFactory.create(query);
    ResultSet rs = QueryExecutionFactory
        .createServiceRequest(
            configuration.getString(Keys.TRIPLESTORE_QUERY_URL),
            select,
            httpAuthenticator)
        .execSelect();
    return rs;
}

public void update(String update) {
    UpdateRequest updateRequest = UpdateFactory.create(update);

    UpdateExecutionFactory.createRemote(
        updateRequest,

```

```

        configuration .getAsString ( Keys .TRIPLESTORE_UPDATE_URL) ,
            httpAuthenticator )
        .execute () ;
    }

private class BatchUploader implements Observer<List<Model>> {

    @Override
    public void onCompleted() {}

    @Override
    public void onError ( Throwable e ) {
        logger .error ( e .getMessage () , e );
    }

    @Override
    public void onNext ( List<Model> models ) {
        if ( !models .isEmpty ()) {
            long start = System .currentTimeMillis ();
            Model buffer = ModelFactory .createDefaultModel ();
            models .forEach ( buffer ::add );
            long end = System .currentTimeMillis ();
            logger .info (" Buffered {} models in {} ms" , models .size () , end - start );

            start = System .currentTimeMillis ();
            DatasetAccessorFactory
                .createHTTP ( configuration .getAsString ( Keys .TRIPLESTORE_STORE_URL) ,
                    httpAuthenticator )
                .add ( buffer );
            end = System .currentTimeMillis ();

            logger .info (" Uploaded {} models in {} ms" , models .size () , end - start );
        }
    }
}

```


Приложение И

Описание подсистемы СОС ЭО ПАП

УТВЕРЖДАЮ

Руководитель проекта

к.т.н., доцент

 Д.И. Муромцев
«30» июня 2016 г.

ПОДСИСТЕМА СОС ЭО ПО ПАП

Описание программы
ЛИСТ УТВЕРЖДЕНИЯ
СОС.00001-01 13 01-ЛУ

Ответственный исполнитель

 М.А. Колчин
«30» июня 2016 г.

Исполнитель

 И.А. Шилин
«30» июня 2016 г.

УТВЕРЖДЕНО

СОС.00001-01 13 01-ЛУ

ПОДСИСТЕМА СОС ЭО ПАП

Описание программы

СОС.00001-01 13 01

Инв. № подп.	Подпись и дата	Взам инв. №	Инв. № дубл.	Подпись и дата

Аннотация

В данном программном документе приведено описание программы «Симулятор облака сенсоров» (далее - СОС), являющейся первичным источником данных распределенной сети электронных потребительских устройств, предназначенной для имитации работы распределенной сети электронных потребительских устройств.

В документе приводятся общие сведения о СОС, реализуемые функции, описание логической структуры программного обеспечения, входные и выходные данные, используемые программным обеспечением.

Оформление программного документа «Описание программы» произведено по требованиям ЕСПД (ГОСТ 19.402-78).

Ж.1 Общие сведения

Ж.1.1 Обозначение и наименование программы

Наименование программы: Симулятор облака сенсоров (СОС).

Ж.1.2 Программное обеспечение, необходимое для функционирования программы

Системные программные средства, используемые программой СОС, должны быть представлены версией операционной системы Ubuntu Server 14.10 или выше и/или Windows 7 или выше.

Также для функционирования программы СОС на персональном компьютере необходимо предустановленное программное обеспечение Java Virtual Machine 1.8.

Ж.1.3 Языки программирования, на которых написана программа

Программа СОС реализована на языках программирования Java. Основные использованные программные библиотеки:

- а) Owner – библиотека для работы с файлами конфигурации;
- б) Californium – библиотека для разработки серверов и клиентов поддерживающих протокол передачи данных СоАР.

Ж.2 Функциональное назначение

Ж.2.1 Назначение программы

Программа СОС являющейся первичным источником данных распределенной сети электронных потребительских устройств, предназначеннай для имитации работы распределенной сети электронных потребительских устройств.

Программа СОС имитирует работу заданного числа электронных устройств для заданного числа домов и квартир в них. Так же в СОС про-

изводится имитация температуры на улице. В программе СОС имитируются электронные устройства измеряющие температуру, данная температура зависит от температуры на улице, заданной погрешности и величины отопления в доме (от 0 до 100

Ж.2.2 Сведения о функциональных ограничениях на применение

Производится имитация только домов, электронных устройств, измеряющих температуру и рычага, задающего процент мощности отопления дома.

Ж.3 Описание логической структуры

Ж.3.1 Алгоритм программы

Ж.3.1.1 Этапы алгоритма работы

Работа СОС состоит из следующих этапов:

- а) чтение файла конфигурации
- б) симуляция описаний и показаний устройств в соответствии с настройками, установленными в файле конфигурации
- в) передача симулируемых описаний и показаний устройств по СоАР протоколу
- г) ожидание и обработка запросов на изменение/получение процента мощности отопления дома

Ж.3.1.2 Чтение файла конфигурации

При запуске программа СОС считывает файл конфигурации. В данном файле хранятся количество домов, количество квартир в доме, количество сенсоров в квартире, начальное значение процента мощности отопления дома и т.д.

Файл конфигурации модуля располагается в корневой директории модуля. Чтение файла производится с помощью библиотеки Owner, а формат файла соответствует формату Java Properties.

Ж.3.1.3 Симуляция описаний и показаний устройств в соответствии с настройками, установленными в файле конфигурации

После чтения файла конфигурации в соответствии с заданными параметрами создается список домов и устройств. Имитируется температура на улице и далее имитируется в соответствии с уличной температурой и, заданным процентом мощности отопления, показания устройств.

Ж.3.1.4 Передача симулируемых описаний и показаний устройств по СОАР протоколу

Далее симулируемые описания и показания устройств передаются по протоколу СОАР. Описание можно получить при помощи get запроса. Для получения показаний необходимо подписать на топик с показаниям, куда они будут приходить один раз в течение заданного интервала.

Ж.3.2 Ожидание и обработка запросов на изменение/получение процента мощности отопления дома

После запуска СОС, можно изменить или получить процент мощности отопления для каждого из домов. Данный процент непосредственно влияет на имитируемую температуру датчиков в доме.

Ж.3.3 Используемые методы

Методы, используемые программным комплексом, указаны при описании алгоритмов в разделе 3.1.

Ж.3.4 Структура программы с описанием функций составных частей и связи между ними

Программа СОС состоит из следующих основных классов:

- а) Launcher – осуществляет запуск программы и считывание конфигурационного файла.
- б) Пакет классов соар – осуществляет передачу данных по протоколу СоАР.
- в) Пакет классов model – содержит классы описывающие модель симулируемых домов, т.е. дома, квартиры и их датчики.
- г) Пакет классов scheduler – выполняет действия производимые раз в некоторый промежуток времени.

Ж.3.5 Связи программы с другими программами

Связи программы с другими программами отсутствуют.

Ж.4 Используемые технические средства

В состав используемых технических средств входит:

- а) оперативная память не менее 16 Гб;
- б) дисковое пространство не менее 1 Тб;
- в) модель процессора не хуже Intel Core i7-4790K 4.0ГГц;
- г) средства ввода-вывода в составе: клавиатура и манипулятор «мышь», монитор, сетевой интерфейс Ethernet, порты USB не менее 4-х.

Ж.5 Вызов и загрузка

Пользователю, под именем которого осуществляется работа с ПК, должен быть открыт доступ к диску файлового хранилища на чтение и запись. Необходимо создать файл конфигурации «config.properties» в котором указать все необходимые параметры. Для запуска необходимо загрузить файл

«building-simulator.jar». Далее запустить через командный терминал по средству команды: `java -jar building-simulator.jar &> simulator.log &`.

Ж.6 Входные данные

Входными данными программы СОС является файл конфигурации. Настройка модулей осуществляется через редактирование файла конфигурации расположенного в директории «/semiot-platform/building-simulator». Файл представляет собой файл в формате Java Properties, где параметры задаются через строки вида [название параметра]=[значение параметра]. В конфигурационном файле можно задать следующие параметры:

- а) количество домов;
- б) количество квартир;
- в) количество устройств в каждой квартире;
- г) промежуток времени между соседними показаниями;
- д) процент мощности отопления;
- е) минимальную температуру на улице;
- ж) максимальную температуру на улице;
- з) оптимальная температура в квартире;
- и) количество показаний между экстремумами.

Ж.7 Выходные данные

Выходными данными программы СОС являются описание и показания электронных потребительских устройств. Выходные данные передаются по протоколу СоАР.

Приложение К

Текст программы подсистемы СОС ЭО ПАП

УТВЕРЖДАЮ

Руководитель проекта

к.т.н., доцент



Д.И. Муромцев

«30» июня 2016 г.

ПОДСИСТЕМА СОС ЭО ПО ПАП

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

СОС.00001-01 12 01-ЛУ

Ответственный исполнитель



М.А. Колчин

«30» июня 2016 г.

Исполнитель



И.А. Шилин

«30» июня 2016 г.

2016

УТВЕРЖДЕНО

СОС.00001-01 12 01-ЛУ

ПОДСИСТЕМА СОС ЭО ПАП

Текст программы

СОС.00001-01 12 01

Инв. № подп.	Подпись и дата	Взам инв. №	Инв. № дубл.	Подпись и дата

Аннотация

В данном программном документе приведено описание программы «Симулятор облака сенсоров» (далее - СОС), являющейся первичным источником данных распределенной сети электронных потребительских устройств, предназначеннной для имитации работы распределенной сети электронных потребительских устройств.

Исходным языком программирования является Java. А средой выполнения является Java Virtual Machine 1.8 или выше.

Оформление программного документа «Текст программы» произведено по требованиям ЕСПД (ГОСТ 19.401-78).

3.1 Текст программы подсистемы СОС на исходном языке

3.1.1 Реализация механизма запуска

```
package ru.semiot.platform;

import static java.util.concurrent.TimeUnit.MINUTES;
import static java.util.concurrent.TimeUnit.SECONDS;

import org.aeonbits.owner.ConfigFactory;
import org.json.JSONArray;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import ru.semiot.platform.coap.BuildingObserver;
import ru.semiot.platform.model.Building;
import ru.semiot.platform.model.Device;
import ru.semiot.platform.scheduler.ScheduledObserve;
import ru.semiot.platform.scheduler.ScheduledTemperature;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.ScheduledFuture;

public class Launcher {

    private static final Logger logger = LoggerFactory.getLogger(Launcher.class);

    private static final ServiceConfig config =
        ConfigFactory.create(ServiceConfig.class);

    private static ScheduledExecutorService schedulerObserve;
    private static ScheduledObserve scheduledObserve;
    private static ScheduledFuture handleObserve = null;

    private static ScheduledExecutorService schedulerTemperature;
    private static ScheduledTemperature scheduledTemperature;
    private static ScheduledFuture handleTemperature = null;

    private static List<Building> buildings = new ArrayList<Building>();

    private static BuildingObserver server;

    public static void main(String[] args) {
        init();
    }
}
```

```

        start();
    }

private static void start() {
    server.start();
    startScheduledObserve();
    startScheduledTemperature();
}

private static void init() {
    StreetTemperature.setMinTemperature(config.temperatureMin());
    StreetTemperature.setMaxTemperature(config.temperatureMax());
    StreetTemperature
        .setCountObservationInTransitionExtremum(config.countObservationInTransitionEx
// StreetTemperature.setTemperature(t);
    Device.setOptimumTemperature(config.temperatureOptimum());

    for (int i = 0; i < config.countBuildings(); i++) {
        buildings
            .add(new Building(config.countFlats(), config.countDevices(),
                config.pressureValue()));
    }
    server = new BuildingObserver(5683, getDescription(), buildings);

    schedulerObserve = Executors.newScheduledThreadPool(1);
    scheduledObserve = new ScheduledObserve(server, buildings);

    schedulerTemperature = Executors.newScheduledThreadPool(1);
    scheduledTemperature = new ScheduledTemperature();
}

private static String getDescription() {
    if (!buildings.isEmpty()) {
        JSONArray jsonArray = new JSONArray();
        for (Building building : buildings) {
            jsonArray.put(building.getDescription());
        }

        return jsonArray.toString();
    }
    return "";
}

public static void startScheduledObserve() {
    if (handleObserve != null)
        stopScheduledObserve();
}

```

```

        handleObserve = schedulerObserve.scheduleAtFixedRate(scheduledObserve, 5,
            config.scheduledDelayObserve(), SECONDS);
        logger.info("Scheduled observe started. Repeat will do every {} seconds",
            String.valueOf(config.scheduledDelayObserve()));
    }

    public static void stopScheduledObserve() {
        if (handleObserve == null)
            return;

        handleObserve.cancel(true);
        handleObserve = null;
        logger.info("Scheduled observe stoped");
    }

    public static void startScheduledTemperature() {
        if (handleTemperature != null)
            stopScheduledTemperature();
        handleTemperature =
            schedulerTemperature.scheduleAtFixedRate(scheduledTemperature, 0,
                config.scheduledDelayTemperature(), MINUTES);
        logger.info("Scheduled temperature started. Repeat will do every {}"
            " minutes",
            String.valueOf(config.scheduledDelayTemperature()));
    }

    public static void stopScheduledTemperature() {
        if (handleTemperature == null)
            return;

        handleTemperature.cancel(true);
        handleTemperature = null;
        logger.info("Scheduled temperature stoped");
    }

}

```

3.1.2 Реализация ресурса СоАР с описанием устройства

```

package ru.semiot.platform.coap;

import org.eclipse.californium.core.CoapResource;
import org.eclipse.californium.core.coap.CoAP;
import org.eclipse.californium.core.server.resources.CoapExchange;

public class DescriptionResource extends CoapResource {

```

```

private static final String name = "desc";
private String description;

public DescriptionResource(String description) {
    super(name);
    this.description = description;
}

@Override
public void handleGET(CoapExchange exchange) {
    exchange.respond(CoAP.ResponseCode.CONTENT, description);
}

}

```

3.1.3 Реализация класса здание (дом)

```

package ru.semiot.platform.model;

import org.json.JSONArray;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

public class Building {

    private static int countBuildings = 0;

    private final int numberBuilding;
    private List<Flat> flats = new ArrayList<Flat>();
    private double presure;

    public Building(int countFlats, int countDevices, double presure) {
        for (int i = 0; i < countFlats; i++) {
            flats.add(new Flat(countDevices));
        }
        numberBuilding = ++countBuildings;
        this.presure = presure;
    }

    public double getPresure() {
        return presure;
    }

    public void setPresure(double presure) {
        this.presure = presure;
    }
}

```

```

    }

    public int getBuildingNumber() {
        return numberBuilding;
    }

    public JSONObject getDescription() {
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("building_number", numberBuilding); // building_number

        JSONObject jsonGeo = new JSONObject();
        jsonGeo.put("latitude", 63.8383);
        jsonGeo.put("longitude", 35.923913);
        jsonObject.put("building_location", jsonGeo); // building_location

        if (!flats.isEmpty()) {
            JSONArray jsonFlats = new JSONArray();
            for (Flat flat : flats) {
                jsonFlats.put(flat.getDescription());
            }
            jsonObject.put("building_flats", jsonFlats);
        }
        return jsonObject;
    }

    public List<JSONObject> getObservations(long timestamp) {
        if (!flats.isEmpty()) {
            List<JSONObject> list = new ArrayList<JSONObject>();
            for (Flat flat : flats) {
                list.addAll(flat.getObservations(timestamp, pressure));
            }
            return list;
        }
        return null;
    }

}

```

3.1.4 Реализация класса посылающего показания устройств через заданный промежуток времени

```

package ru.semiot.platform.scheduler;

import org.json.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

```

```

import ru.semiot.platform.coap.BuildingObserver;
import ru.semiot.platform.model.Building;

import java.util.ArrayList;
import java.util.List;

public class ScheduledObserve implements Runnable {

    private static final Logger logger =
        LoggerFactory.getLogger(ScheduledObserve.class);

    private BuildingObserver observer;
    private List<Building> buildings;
    double i = 0;

    public ScheduledObserve(BuildingObserver observer, List<Building> buildings) {
        this.observer = observer;
        this.buildings = buildings;
    }

    public void run() {
        List<JSONObject> list = new ArrayList<JSONObject>();
        for (Building building : buildings) {
            list.addAll(building.getObservations(System.currentTimeMillis()));
        }

        logger.debug("Observer update.");
        observer.update(null, list.toString());
    }
}

```

Лист регистрации изменений

Приложение Л

Описание программного эмулятора МШСПОИ

УТВЕРЖДАЮ

Руководитель проекта

к.т.н., доцент

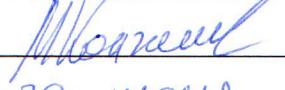
 Д.И. Муромцев
«30» июня 2016 г.

Программный эмулятор МШСПОИ

Описание программы
ЛИСТ УТВЕРЖДЕНИЯ

ПЭ.00001-01 13 01-ЛУ

Ответственный исполнитель

 М.А. Колчин
«30» июня 2016 г.

Исполнитель

 А.А. Андреев
«30» июня 2016 г.

УТВЕРЖДЕНО

ПЭ.00001-01 13 01-ЛУ

Программный эмулятор МШСПОИ

Описание программы

ПЭ.00001-01 13 01

Инв. № подп.	Подпись и дата	Взам инв. №	Инв. № дубл.	Подпись и дата

Аннотация

Программный эмулятор МШСПОИ предназначен для эмуляции предоставления доступа к данным электронных устройств и управлением. Программный эмулятор поддерживает основные открытые платформы и позволяет реализовать узел получения данных с устройств, не имеющих прямого доступа в проводные и беспроводные сети. Программный модуль реализует следующие функции: позволяет получать данные с электронных устройств любым доступным операционной системе или прошивке устройства образом (импульсные входы, цифровые и аналоговые входы, по прерываниям и в цикле), формировать интерфейс доступа по выбранному бинарному протоколу RFC 7252 (Constrained Application Protocol (CoAP)) с возможностью чтения и записи данных. Дополнительно реализована поддержка информирования всех заинтересованных клиентов об обновлениях выбранного ресурса согласно протоколу CoAP. Кроме того, на основе полученного интерфейса сформирован защищенный механизм конфигурирования подключения устройства к пользовательской локальной сети Wi-Fi посредством дополнительной программной точки доступа Wi-Fi, защищенной по стандарту WPA2/PSK. Предлагаемый механизм конфигурирования также является самодокументируемым с использованием рекомендации консорциума W3C RDF Schema, а программный модуль поддерживает автоматическое обнаружение новых устройств и их ресурсов по стандарту RFC 6690 (CoRE Link Format).

Оформление программного документа «Описание программы» произведено по требованиям ЕСПД (ГОСТ 19.402-78).

Л.1 Общие сведения

Программный эмулятор МШСПОИ является кросплатформенной программой на языке программирования Java и использует бинарный сетевой протокол СоАР на основе протокола передачи данных UDP.

Л.2 Функциональное назначение

Программный эмулятор МШСПОИ предназначен эмуляции для обнаружения и предоставления доступа к требуемым ресурсам системы электронных устройств: для получения своевременного уведомления об изменениях, а также для ручного получения текущего статуса о состоянии системы и управления ресурсами, доступными для изменения.

Л.3 Описание логической структуры

Система производит опрос сетевого модуля на начиание новых входящих сетевых пакетов по стандартизированному СоАР порту 5683 протокола UDP. В случае получения нового сообщения сохраняется адрес отправителя и производится проверка пакета на формальное соответствие стандарту СоАР и отсутствие ошибок.

После получения информации о запрашиваемом методе и имени ресурса из входящего пакета производится проверка на наличие такового в системе согласно стандарту RFC 6690 (CoRE Link Format).

Частным случаем является документированный ресурс *.well-known/core*, который предоставляет информацию о доступных ресурсах в системе.

Каждый ресурс при этом является самодокументируемым, реализующим принципы Linked Data.

В случае обнаружения имеющегося ресурса выполняется указанный для него обработчик и отправляется сформированное обработчиком ответное сообщение. В противном случае отправляется стандартизованное сообщение об ошибке.

Л.4 Используемые технические средства

Исполнение программного эмулятора МШСПОИ предназначено для любых систем реализующих среду Java SE версии 8 и выше.

Л.5 Вызов и загрузка

Для сборки программного обеспечения используется система сборки Maven, которая реализует механизмы компиляции и сборки исполняемого файла.

Для выполнения сборки необходимо выполнить следующую команду в директории, содержащей файл pom.xml: mvn clean install.

Система поддерживает вызовы по протоколу СоAP в формате пакетов без разбивки на блоки.

Л.6 Входные данные

Программный эмулятор получает на вход UDP-пакеты формата СоAP RFC 7252 и идентифицирует отправителя по IP-адресу версии 4 и 6.

Л.7 Выходные данные

Программный эмулятор отправляет UDP-пакеты отправителю в формате СоAP RFC 7252 и выводит служебную информацию о состоянии работы в потоки системного вывода в случае отладочного режима.

Лист регистрации изменений

Приложение М

Текст программы программного эмулятора МШСПОИ

УТВЕРЖДАЮ

Руководитель проекта

к.т.н., доцент



Д.И. Муромцев

«30» июня 2016 г.

ПРОГРАММНЫЙ ЭМУЛЯТОР МШСПОИ

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

ПЭ.00001-01 12 01-ЛУ

Ответственный исполнитель



М.А. Колчин

«30» июня 2016 г.

Исполнитель



А.А. Андреев

«30» июня 2016 г.

УТВЕРЖДЕНО

ПЭ.00001-01 12 01-ЛУ

ПОДСИСТЕМА ПЭ ЭО ПАП

Текст программы

ПЭ.00001-01 12 01

Инв. № подп.	Подпись и дата	Взам инв. №	Инв. № дубл.	Подпись и дата

Аннотация

Программный модуль МШСПОИ предназначен для предоставления доступа к данным электронных устройств и управлением через беспроводную сеть Wi-Fi. Программный модуль поддерживает основные открытые платформы и позволяет реализовать узел получения данных с устройств, не имеющих прямого доступа в проводные и беспроводные сети. Программный модуль реализует следующие функции: позволяет получать данные с электронных устройств любым доступным операционной системе или прошивке устройства образом (импульсные входы, цифровые и аналоговые входы, по прерываниям и в цикле), формировать интерфейс доступа по выбранному бинарному протоколу RFC 7252 (Constrained Application Protocol (CoAP)) с возможностью чтения и записи данных. Дополнительно реализована поддержка информирования всех заинтересованных клиентов об обновлениях выбранного ресурса согласно протоколу CoAP. Кроме того, на основе полученного интерфейса сформирован защищенный механизм конфигурирования подключения устройства к пользовательской локальной сети Wi-Fi посредством дополнительной программной точки доступа Wi-Fi, защищенной по стандарту WPA2/PSK. Предлагаемый механизм конфигурирования также является самодокументируемым с использованием рекомендации консорциума W3C RDF Schema, а программный модуль поддерживает автоматическое обнаружение новых устройств и их ресурсов по стандарту RFC 6690 (CoRE Link Format).

Исходным языком программирования является Java.

Оформление программного документа «Текст программы» произведено по требованиям ЕСПД (ГОСТ 19.401-78).

М.1 Текст программы программного эмулятора МШСПОИ

М.1.1 Исходный код реализации СоАР сервера

```
package ru.semiot.platform.drivers.dht22;

import java.io.Closeable;
import java.io.IOException;
import org.eclipse.californium.core.CoapServer;
import org.eclipse.californium.core.network.CoAPEndpoint;
import org.eclipse.californium.core.network.Endpoint;

public class CoAPInterface extends CoapServer
    implements Closeable, AutoCloseable {

    private static Endpoint endpoint;

    public CoAPInterface() {
        super();
        endpoint = new CoAPEndpoint();
        addEndpoint(endpoint);
    }

    public static Endpoint getEndpoint() {
        return endpoint;
    }

    @Override
    public void start() {
        super.start();
    }

    @Override
    public void close() throws IOException {
        super.destroy();
    }
}
```

М.1.2 Исходный код принятия показаний по СоАР

```
package ru.semiot.platform.drivers.dht22;

import java.text.SimpleDateFormat;
import java.util.Date;
```

```

import org.eclipse.californium.core.CoapClient;
import org.eclipse.californium.core.CoapResponse;

import ru.semiot.platform.deviceproxyservice.api.drivers.Device;

public class ScheduledDevice implements Runnable {

    private DeviceDriverImpl ddi;

    private static final int FNV_32_INIT = 0x811c9dc5;
    private static final int FNV_32_PRIME = 0x01000193;

    private CoapClient coapClientTemperature;
    private CoapClient coapClientHumidity;
    private String hash;

    private static final String templateTopic = "${DEVICE_HASH}";
    private static final String templateOnState = "prefix saref:
        <http://ontology.tno.nl/saref#> "
        + "<http://${DOMAIN}/${SYSTEM_PATH}/${DEVICE_HASH}> saref:hasState
        saref:OnState .";

    public ScheduledDevice(DeviceDriverImpl ddi) {
        this.ddi = ddi;
        hash = getHash(ddi.getDriverName() + ddi.getIp() +
                String.valueOf(ddi.getPort()));
        ddi.addDevice(new Device(hash, ddi.getTemplateDescription()
                .replace("${DEVICE_HASH}", hash).replace("${SYSTEM_PATH}",
                ddi.getPathSystemUri())
                .replace("${SENSOR_PATH}", ddi.getPathSensorUri())
                .replace("${DOMAIN}", ddi.getDomain())
                .replace("${LATITUDE}", String.valueOf(ddi.getLat()))
                .replace("${LONGITUDE}", String.valueOf(ddi.getLng()))));
    }

    public void createCoapClients() {
        coapClientHumidity = new CoapClient("coap://" + ddi.getIp() + ":" +
                String.valueOf(ddi.getPort()) + "/dht22/humidity");
        coapClientTemperature = new CoapClient("coap://" + ddi.getIp() + ":" +
                String.valueOf(ddi.getPort()) + "/dht22/temperature");
        coapClientTemperature.setEndpoint(CoAPInterface.getEndpoint());
        coapClientHumidity.setEndpoint(CoAPInterface.getEndpoint());
    }

    public void run() {
        if(ddi.listDevices().size() == 1 && ddi.listDevices().get(0).getTurnOn()) {

```

```

long timestamp = System.currentTimeMillis();
CoapResponse crHumidity = coapClientHumidity.get();
System.out.println(crHumidity.getResponseText());
try {
    Thread.sleep(1000);
} catch (InterruptedException e) {
    System.err.println(e.getMessage());
}
CoapResponse crTemperature = coapClientTemperature.get();
System.out.println(crTemperature.getResponseText());

if (crTemperature != null && crHumidity != null) {
    Device device = ddi.listDevices().get(0);
    if (!device.getTurnOn()) {
        device.setTurnOn(true);
        ddi.inactiveDevice(templateOnState
            .replace("${DEVICE_HASH}", device.getID())
            .replace("${SYSTEM_PATH}", ddi.getPathSystemUri())
            .replace("${DOMAIN}", ddi.getDomain())));
    }
    sendMessage(crTemperature.getResponseText().trim(),
        crHumidity.getResponseText().trim(),
        timestamp, hash);
} else { // TODO проверятьсялиэтотфакторомотключениядля
    Device device = ddi.listDevices().get(0);
    device.setTurnOn(false);
    ddi.inactiveDevice(DeviceDriverImpl.templateOffState
        .replace("${DEVICE_HASH}", device.getID())
        .replace("${SYSTEM_PATH}", ddi.getPathSystemUri())
        .replace("${DOMAIN}", ddi.getDomain()));

    coapClientTemperature.shutdown();
    coapClientHumidity.shutdown();
    ddi.restartSheduller();
}
}

private void sendMessage(String valueTemperature, String valueHumidity,
    long timestamp, String hash) {
    if (valueTemperature != null && valueHumidity != null) {
        String topic = templateTopic.replace("${DEVICE_HASH}", hash);

        final String date = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ssXXX")
            .format(new Date(timestamp));
    }
}

```

```

String messageTemperature = ddi
    .getTemplateObservationTemperature()
    .replace("${DOMAIN}", ddi.getDomain())
    .replace("${SYSTEM_PATH}", ddi.getPathSystemUri())
    .replace("${SENSOR_PATH}", ddi.getPathSensorUri())
    .replace("${DEVICE_HASH}", hash)
    .replace("${TIMESTAMP}", String.valueOf(timestamp))
    .replace("${DATETIME}", date)
    .replace("${VALUE}", valueTemperature);

String messageHumidity = ddi
    .getTemplateObservationHumidity()
    .replace("${DOMAIN}", ddi.getDomain())
    .replace("${SYSTEM_PATH}", ddi.getPathSystemUri())
    .replace("${SENSOR_PATH}", ddi.getPathSensorUri())
    .replace("${DEVICE_HASH}", hash)
    .replace("${TIMESTAMP}", String.valueOf(timestamp))
    .replace("${DATETIME}", date)
    .replace("${VALUE}", valueHumidity);

ddi.publish(topic, messageTemperature);
ddi.publish(topic, messageHumidity);
} else {
    System.out.println(hash + " has unknown value (null)");
}
}

private String getHash(String id) {
    String name = id + ddi.getDriverName();
    int h = FNV_32_INIT;
    final int len = name.length();
    for(int i = 0; i < len; i++) {
        h ^= name.charAt(i);
        h *= FNV_32_PRIME;
    }
    long longHash = h & 0xffffffffl;
    return String.valueOf(longHash);
}
}

```


Приложение Н

Описание применения ЭО ПО ПАП

УТВЕРЖДАЮ

Руководитель проекта

к.т.н., доцент

 Д.И. Муромцев
«30» июня 2016 г.

ЭО ПО ПАП
Описание применения
ЛИСТ УТВЕРЖДЕНИЯ
ПАП.00001-01 31 01-ЛУ

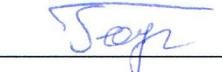
Ответственный исполнитель

 М.А. Колчин
«30» июня 2016 г.

Исполнитель

 И.А. Шилин
«30» июня 2016 г.

Исполнитель

 Д.С. Гарайзуев
«30» июня 2016 г.

УТВЕРЖДЕНО

ПАП.00001-01 31 01-ЛУ

ЭО ПО ПАП

Описание применения

ПАП.00001-01 31 01

Инв. № подп.	Подпись и дата	Взам инв. №	Инв. № дубл.	Подпись и дата

Аннотация

В данном программном документе приведено описание применения ЭО ПО ПАП, предназначенного для агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, слабоструктурированных и неструктурных данных распределенной сети электронных потребительских устройств.

В данном программном документе, в разделе «Назначение программы» приведено описание назначения программы, возможности данной программы, а также ее ограничения, накладываемые на область применения программы.

В разделе «Условия применения» указаны условия, необходимые для выполнения программы (требования к необходимым для данной программы техническим средствам, а также требования и условия организационного, технического и технологического характера).

В данном программном документе, в разделе «Описание задачи» указано определение задач.

В разделе «Входные и выходные данные» указаны сведения о входных и выходных данных.

H.1 Назначение и возможности программы

H.1.1 Назначение программы

ЭО ПО ПАП предназначен для агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, слабоструктурированных и неструктурированных данных распределенной сети электронных потребительских устройств.

ЭО ПО ПАП выполняет следующие функции:

- а) Собирает информацию об узлах сенсорной сети и агрегирует их показания;
- б) Записывает новые показания с узлов в специализированную базу данных для временных данных (Time series database);
- в) Записывает статическую информацию об узлах в RDF-хранилище, которое предоставляет SPARQL-точку доступа;
- г) Выполняет логический вывод (reasoning) над RDF-данными узлов для приведения модели данных к унифицированному формату, используемом программой;
- д) Выполняет анализ текущих и исторических показаний для обнаружения ситуаций предметной области;
- е) Реализует интерактивный графический пользовательский интерфейс в форме веб-приложения;
- ж) Симуляция показаний электронных потребительских устройств.

H.1.2 Возможности программы

ЭО ПО ПАП позволяет производить сбор разнородных данных, анализировать и визуализировать собранные данные сети электронных потребительских устройств. Так же позволяет производить симуляцию показаний электронных потребительских устройств.

H.1.3 Ограничения, накладываемые на область применения программы

Ограничения, накладываемые на область применения программы отсутствуют.

H.2 Условия применения

H.2.1 Требования к техническим (аппаратным) средствам

В состав используемых технических средств входит:

- а) оперативная память не менее 2GByte/CPU;
- б) дисковое пространство не менее 15-20 GByte/CPU;
- в) модель процессора не хуже Intel Xeon – 2606, 4x2 GHz CPU Cores.

H.2.2 Требования к программным средствам (другим программам)

Программное обеспечение ЭО ПО ПАП выполняется в среде операционной среды Ubuntu Server. ЭО ПО ПАП требует наличия следующего общесистемного программного обеспечения:

- Ubuntu Server 14.10 x64 или выше
- Java Virtual Machine 1.8 или выше
- MySQL 5.6 или выше
- docker-compose 1.7.1 и выше
- docker 1.9.1 и выше

H.2.3 Требования и условия организационного характера

Для обеспечения работоспособности ЭО ПО ПАП необходимо периодически проводить проверку правильности выполнения и загрузки ЭО ПО ПАП.

H.2.4 Требования и условия технического характера

Требования и условия технического характера отсутствуют.

H.2.5 Требования и условия технологического характера

Для работы ЭО ПО ПАП не требуется обеспечения каких либо особых требований и условий технологического характера.

H.3 Описание задачи

H.3.1 Определение задачи

В настоящее время активно растет количество электронных устройств, используемых как в повседневной жизни, так и в промышленных интересах. Согласно отчету Gartner “6.4 миллиарда подключенных устройств будут использоваться по всему миру в 2016 году, на 30 процентов больше, чем в 2015 году, и достигнет 20.8 млрд к 2020 году”. Данные устройства имеют разные форматы выходных данных, типы протоколов по которым работают. Данные устройства полезны, но огромное количество данных с них с каждым днем все тяжелее анализировать. Поэтому возникает задача повышения эффективности научных исследований посредством агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, полу-структурных и неструктурных данных в распределенной сети электронных потребительских устройств.

H.4 Входные и выходные данные

H.4.1 Сведения о входных данных

Входными данными ЭО ПО ПАП является файл конфигурации config.properties установленный в директории "/semiot-platform/building-simulator" и домен, используемый в системе, который указывается через визуальный интерфейс. Файл представляет собой файл в формате Java Properties, где параметры задаются через строки вида [название параметра]=[значение]

параметра]. В конфигурационном файле можно задать следующие параметры:

- а) количество домов;
- б) количество квартир;
- в) количество устройств в каждой квартире;
- г) промежуток времени между соседними показаниями;
- д) процент мощности отопления;
- е) минимальную температуру на улице;
- ж) максимальную температуру на улице;
- з) оптимальная температура в квартире;
- и) количество показаний между экстремумами.

Н.4.2 Сведения о выходных данных

Выходными данными ЭО ПО ПАП являются описание и показания электронных потребительских устройств. Выходные данные записываются в базу данных и отправляются в ответ на запросы клиентов к REST API и визуальному интерфейсу.

ПАП.00001-01 31 01

Приложение О

Описание логической структуры базы данных

observation	commandresult
system_id sensor_id } (PK) event_time (PK) feature_of_interest property value	system_id process_id } (PK) event_time (PK) command_parameters command_properties command_type

Приложение П

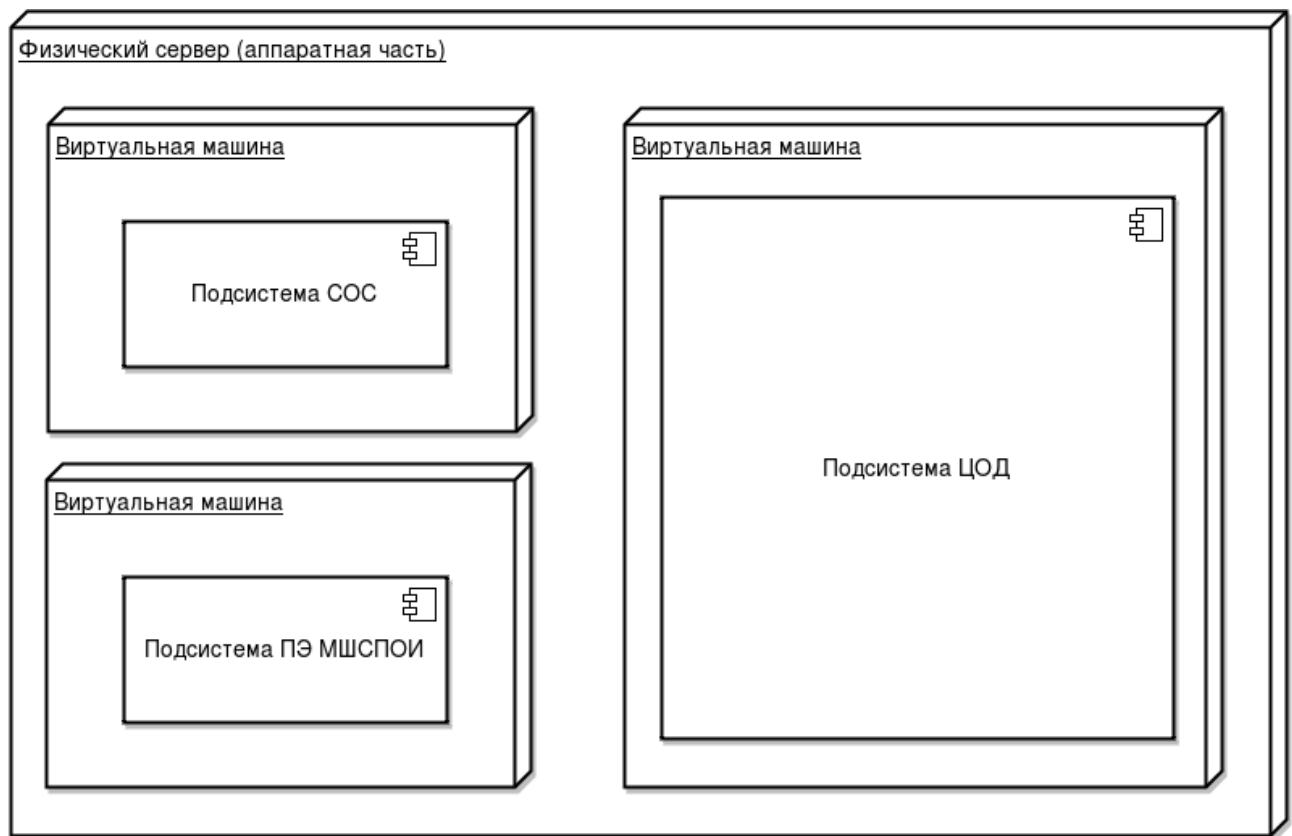
Описание физической структуры базы данных

Keyspace semiot

observation	commandresult
system_id: text sensor_id: text event_time: timestamp (PK) feature_of_interest: timestamp property: text value: text	system_id: text process_id: text event_time: timestamp (PK) command_parameters: list<frozen<command_parameter>> command_properties: list<frozen<command_property>> command_type: text

Приложение Р

Схема структурная ЭО ПАП



Приложение С

Инструкция по эксплуатации ЭО ПАП

ЭКСПЕРИМЕНТАЛЬНЫЙ ОБРАЗЕЦ
ПРОГРАММНО-АППАРАТНОЙ ПЛАТФОРМЫ (ЭО ПАП)
ПАП.000000.001 00 01
Инструкция по эксплуатации

Аннотация

Настоящий документ предназначен для операторов экспериментального образца программно-аппаратной платформы для сбора, нормализации, обработки и визуализации гетерогенных (структурированных, полуструктурных и неструктурных) данных распределенной сети электронных потребительских устройств.

В документе содержатся сведения о назначении ЭО ПАП и условиях, необходимых для его выполнения. Приведены последовательность действий оператора, обеспечивающие запуск и решение задач, и тексты сообщений выдаваемых в ходе работы.

C.1 Назначение ЭО ПАП

ЭО ПАП предназначен для агрегации, нормализации, анализа и визуализации больших массивов гетерогенных структурированных, слабоструктурированных и неструктурных данных распределенной сети электронных потребительских устройств.

ЭО ПАП выполняет следующие функции:

- а) Собирает информацию об узлах сенсорной сети и агрегирует их показания;
- б) Записывает новые показания с узлов в специализированную базу данных для временных данных (Time series database);
- в) Записывает статическую информацию об узлах в RDF-хранилище, которое предоставляет SPARQL-точку доступа;
- г) Выполняет логический вывод (reasoning) над RDF-данными узлов для приведения модели данных к унифицированному формату, используемом программой;
- д) Выполняет анализ текущих и исторических показаний для обнаружения ситуаций предметной области;
- е) Реализует интерактивный графический пользовательский интерфейс в форме веб-приложения;
- ж) Симуляция показаний электронных потребительских устройств.

C.2 Условия выполнения программного обеспечения ЭО ПАП

Программное обеспечение ЭО ПАП выполняется в среде операционной среды Ubuntu Server. ЭО ПО ПАП требует наличия следующего общесистемного программного обеспечения:

- а) Ubuntu Server 14.10 x64 или выше
- б) Java Virtual Machine 1.8 или выше
- в) MySQL 5.6 или выше
- г) docker-compose 1.7.1 и выше
- д) docker 1.9.1 и выше

Пользователю, под именем которого осуществляется работа с ПК, должен быть открыт доступ к диску файлового хранилища на чтение и запись.

C.3 Инсталляция программного обеспечения ЭО ПАП

C.3.1 Состав дистрибутива программного обеспечения ЭО ПАП

В состав ЭО ПАП входят следующие компоненты:

- а) docker-compose.yml
- б) docker.semiot.ru/wamp-router image
- в) docker.semiot.ru/data-archiving-service image
- г) cassandra:3.3 image
- д) docker.semiot.ru/device-proxy-service image
- е) docker.semiot.ru/fuseki image
- ж) docker.semiot.ru/api-gateway image
- з) docker.semiot.ru/frontend image
- и) docker.semiot.ru/mysql image
- к) building-simulator.jar

Перед инсталляцией программной части ЭО ПАП необходимо с помощью подсистемы виртуализации серверов создать виртуальную машину и выполнить установку операционной системы Ubuntu Server 14.10 или выше.

А также выполнить установку необходимого программного обеспечения перечисленного в разделе П.2.

C.3.2 Установка файлов программного обеспечения ЭО ПАП

В домашней директории пользователя, под именем которого будет выполняться ЭО ПАП, создать директорию, в которой разместить файлы программных компонентов docker-compose.yml и building-simulator.jar. Далее выполнить из этой директории команду: sudo docker-compose pull. Данная команда скачает все необходимые image файлы для работы программы (остальные компоненты ЭО ПАП). Далее необходимо создать config.properties файл с конфигурацией для каждого из модулей в соответствующей директории для модуля, в которых указать необходимые настройки, перечисленные в П.3.3.

Для проверки наличия установленной Java Virtual Machine необходимо выполнить в командном терминале следующую команду: java –version. Аналогично docker необходимо выполнить в командном терминале следующую команду: docker –version. Аналогично docker-compose необходимо выполнить в командном терминале следующую команду: docker-compose –version . В случае успешного выполнения в командном терминале должна быть выведен текст как на рисунке П.1.

```
java version "1.8.0_91"
Java(TM) SE Runtime Environment (build 1.8.0_91-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.91-b14, mixed mode)
user@ubuntu:~$ docker --version
Docker version 1.9.1, build a34a1d5
user@ubuntu:~$ docker-compose --version
docker-compose version 1.7.1, build 6c29830
```

Рисунок С.1 — Результат выполнения проверки версий java, docker и docker-compose.

C.3.3 Настройка программного обеспечения ЭО ПАП

Для настройки работы ЭО ПАП использует файл конфигурации config.properties установленный в директории "/semiot-platform/building-simulator". Файл представляет собой файл в формате Java Properties, где

параметры задаются через строки вида [название параметра]=[значение параметра]. В конфигурационном файле можно задать следующие параметры:

- а) количество домов;
- б) количество квартир;
- в) количество устройств в каждой квартире;
- г) промежуток времени между соседними показаниями;
- д) процент мощности отопления;
- е) минимальную температуру на улице;
- ж) максимальную температуру на улице;
- з) оптимальная температура в квартире;
- и) количество показаний между экстремумами.

Так же необходимо задать домен используемый в системе. Это осуществляется через визуальный интерфейс ЭО ПАП. Необходимо выполнить следующие действия

- а) зайти на адрес <https://localhost>
- б) авторизоваться под логином: root, паролем: root
- в) выполнить переход на страницу, выбрав из выпадающего списка "Configuration> "System"
- г) задать домен и нажать кнопку "Save"

После рекомендуется выполнить изменения текущего пароля для пользователя root. Для изменения пароля необходимо перейти на вкладку "Configuration> "User".

C.4 Выполнение программного обеспечения ЭО ПАП

C.4.1 Запуск программного обеспечения ЭО ПАП

Запуск программной части ЭО ПАП выполняется в следующей последовательности:

- а) запуск подсистемы ЦОД через командный терминал по средствам команды: sudo docker-compose up -d

- б) запуск подсистемы СОС через командный терминал по средсвам команды: `java -jar building-simulator.jar &> simulator.log &`.

Лог выполнения для подсистемы СОС доступен в файле `simulator.log`, для подсистемы ЦОД лог отображается при выполнении команды `docker-compose logs`.

C.4.2 Остановка программного обеспечения ЭО ПАП

Для остановки ЭО ПАП необходимо убить процессы запущенных программ, а для этого необходимо выполнить следующие действия:

- а) выполнить команду: `ps | grep <имя .jar пакета>`, где `<имя .jar пакета>` должно быть указано `building-simulator.jar`
- б) найти в первом столбце вывода последней команды номер процесса
- в) выполнить команду: `kill <номер процесса>`

Так же необходимо выполнить следующие команды из директории системы, для остановки подсистемы ЦОД:

- а) `docker-compose stop`
- б) `docker-compose rm` (в случае есть необходимо удалить процессы)

C.5 Сообщения оператору

Во время выполнения приведенных в предыдущих разделах команд оператор может столкнуться с некоторым сообщениями, в соответствии с таблицей C.1.

Таблица С.1 — Описание сообщений оператору

Сообщение	Ситуация
GenericJDBCException: Cannot open connection	В данном случае необходимо: проверить наличие необходимых настроек подключения к БД; проверить запущена ли БД
Exception: Port 5683 already in use	В данном случае необходимо удостовериться, что ни один из запущенных процессов в системе не использует порт 5683, после чего попробовать запустить программу снова
FileNotFoundException: config.properties (No such file or directory)	В данном случае надо удостовериться о наличии файла конфигурации config.properties в папке /semiot-platform/building-simulator.
UnsupportedClassVersionError	В данном случае необходимо удостовериться, что версия Java Virtual Machine соответствует требуемой

Лист регистрации изменений

Приложение Т

Формуляр ЭО ПАП

503200
(код продукции)

Программно-аппаратная платформа
наименование и индекс изделия

ФОРМУЛЯР
ПАП.000000.001 ФО 01
обозначение документа

Т.1 Основные сведения об изделии

Наименование: Экспериментальный образец масштабируемой сервис-ориентированной программно-аппаратной платформы для сбора, нормализации, обработки и визуализации больших массивов гетерогенных (структурированных, полуструктурных и неструктурных) данных в распределенной сети электронных потребительских устройств (Internet of Things).

Обозначение: ПАС.000000.001

Дата изготовления: 01.06.2016

Наименование и/или почтовый адрес изготовителя: федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики», 197101, г. Санкт-Петербург, Кронверкский пр., д.49

Т.2 Основные технические данные

Наименование параметра	Значение
Мощность, Вт	200
Наработка с начала эксплуатации	24 ч.
ОЗУ	12 Гб
Процессор	Intel Xeon 5600/5500
Дисковое пространство	1 Тб

Т.3 Ресурсы, сроки службы и хранения, гарантии производителя

Ресурс изделия до первого _____
ремонта _____

в течение срока службы ____ лет, в том числе срок хранения ____ лет (года) в
консервации (упаковке) изготовителя, в складских помещениях, на открытых
площадках и т.п.

Межремонтный ресурс _____
при _____ ремонте(ах) в течение срока службы _____ лет.

Указанные ресурсы, сроки службы и хранения действительны при со-
блюдении потребителем требований действующей эксплуатационной докумен-
тации.

Линия отреза при поставке на экспорт

Гарантии изготовителя: Предприятие-изготовитель гарантирует ста-
бильную работу ЭО ПАП. А так же гарантируется работоспособность элемен-
тов ЭО ПАП в соответствии с гарантиями производителей данных элементов.

Предприятие-изготовитель обязуется принимать меры по устранению
ошибок в ЭО ПАП при обнаружении в процессе работы и своевременном
сообщении о них со стороны пользователей с предоставлением полной ин-
формации об условиях возникновения ошибки.

Т.4 Консервация

Т.5 Свидетельство об упаковывании

Экспериментальный образец ПАП ПАП.000000.001 № _____

Упакован(а)_____

Согласно требованиям, предусмотренным в действующей технической документации.

должность

личная подпись

расшифровка подписи

Т.6 Свидетельство о приемке

Экспериментальный образец ЭО ПАП ПАП.000000.001

изготовлен(а) и принят(а) в соответствии с обязательными требованиями государственных (национальных) стандартов, действующей технической документацией и признан(а) годным(ой) для эксплуатации.

Т.7 Движение изделия при эксплуатации

Т.8.1 Прием и передача изделия

Т.8.2 Сведения о закреплении изделия в эксплуатации

Т.9 Учет работы изделия

Т.10 Учет технического обслуживания

Т.11 Учет работы по бюллетеням и указаниям

Т.12 Работы по эксплуатации

Т.12.1 Учет выполнения работы

Т.12.2 Периодический контроль основных эксплуатационных и технических характеристик

Т.12.3 Проверка средств измерений

Т.12.4 Техническое освидетельствование контрольными органами

Т.13 Хранение

T.14 Ремонт

Т.14.1 Краткие записи о произведенном ремонте

№ _____

Наработка с начала эксплуатации _____

Наработка после последнего ремонта _____

Причина поступления в ремонт _____

Сведения о произведенном ремонте _____

Т.14.2 Данные приемосдаточных испытаний

ЭО ПАП соответствует заданным в техническом задании характеристикам.

Т.14.3 Свидетельство о приемке и гарантии

наименование изделия	№ _____	обозначение	заводской номер
ремонта	наименование предприятия	вид документа	

Принят(а) в соответствии с обязательными требованиями государственных (национальных) стандартов и действующей технической документацией и признан(а) годным(ой) для эксплуатации.

Ресурс до очередного ремонта _____
в течение срока службы _____ лет (года),
в том числе срок хранения _____

Исполнитель ремонта гарантирует соответствие изделия требованиям действующей технической документации при соблюдении потребителем требований действующей эксплуатационной документации.

Т.15 Особые отметки

Т.16 Контроль состояния изделия и ведения формуляра