Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Сетевые операционные системы»

Студент **ИУ7-53Б** _____ **Сапожков А. М.**

Руководитель _____ **Строганов Ю. В.**

2022 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ-7
(Индекс)
_____ И. В. Рудаков
(И.О.Фамилия)
«16» сентября 2022 г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме

«Сетевые операционные системы»

Студент группы **ИУ7-53Б**

Сапожков Андрей Максимович

Направленность НИР

учебная

Источник тематики

НИР кафедры

График выполнения НИР: 25% к 6 нед., 50% к 9 нед., 75% к 12 нед., 100% к 15 нед.

Техническое задание
Провести обзор существующих операционных систем для устройств интернета вещей. Провести анализ предметной области интернета вещей, сформулировать критерии сравнения и оценки рассмотренных операционных систем. Классифицировать существующие решения по выделенным критериям.

Оформление научно-исследовательской работы:
Расчетно-пояснительная записка на **12-25** листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т. п.)
Презентация на **6-10** слайдах.

Дата выдачи задания «16» сентября 2022 г.

Руководитель НИР _____ **Ю. В. Строганов**
(Подпись, дата) (И.О.Фамилия)

Студент _____ **А. М. Сапожков**
(Подпись, дата) (И.О.Фамилия)

Реферат

Расчетно-пояснительная записка 49 с., 3 рис., 1 табл., 71 ист., 1 прил.

ОПЕРАЦИОННЫЕ СИСТЕМЫ, СЕТЕВЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ, ИНТЕРНЕТ ВЕЩЕЙ, IOT

Цель работы: изучение существующих операционных систем для устройств интернета вещей.

В данной работе проводится изучение и сравнение операционных систем для интернета вещей (IoT).

Результаты: среди рассмотренных операционных систем были выделены:

- Azure Sphere, Windows 10 IoT и Amazon FreeRTOS как наиболее функциональные и масштабируемые;
- ОСРВ МАКС и KasperskyOS как наиболее доступные с точки зрения использования прикладных служб;
- Ubuntu Core и Raspbian как наиболее адаптированные для бытового применения.

СОДЕРЖАНИЕ

Введение

В настоящий момент все наблюдаемые формы коммуникаций сводятся к схемам человек-человек и человек-устройство. Но интернет вещей (Internet of Things, IoT) предлагает нам колоссальное интернет-будущее, в котором появятся коммуникации типа машина-машина (Machine-to-Machine, M2M). Это дает возможность для объединения всех коммуникаций в общую инфраструктуру, позволяя не только управлять всем, что находится вокруг нас, но и предоставлять информацию о состоянии этих вещей [1].

Стоит отметить, что M2M — это подмножество IoT, которое явно имеет дело с соединениями между устройствами. IoT и M2M обеспечивают удаленный доступ для обмена информацией между устройствами без участия человека. Ключевое различие между IoT и M2M заключается в том, что M2M — это подключение двух или более устройств к Интернету для обмена данными, а IoT подключает любое устройство к интернету для изменения процессов в окружающем мире при помощи аналитики. M2M — это то, что обеспечивает интернет вещей связью, без которой IoT был бы невозможен. Идеология интернета вещей направлена на повышение эффективности экономики за счет автоматизации процессов в различных сферах деятельности и исключения из них человека.

Главная задача интернета вещей [2] — создание среды, в которой вещами можно управлять, а данные о них могут быть получены и обработаны для выполнения желаемой задачи посредством обучения устройств. Эта концепция позволяет не только объединять предметы материального мира посредством интернета для обмена информацией между ними, но и развивать возможности по накоплению, структурированию и анализу различной информации о поведении людей в городском пространстве, дома и на работе.

В основе большинства решений в области интернета вещей лежит устройство, которое подключается к облачным сервисам для обмена данными. Обра-

ботка данных может выполняться либо на самом устройстве, либо на удалённом сервере. Возможности для разработки таких решений обеспечивает операционная система. При выборе подходящей операционной системы для устройства интернета вещей разработчик должен убедиться, что она совместима с необходимым оборудованием и приложениями. Большое разнообразие операционных систем позволяет выбрать подходящий вариант в зависимости от особенностей конкретной задачи.

Целью данной работы является изучение существующих операционных систем для устройств интернета вещей. Для достижения поставленной цели необходимо решить следующие задачи.

1. Провести анализ предметной области интернета вещей.
2. Рассмотреть существующие операционные системы для устройств интернета вещей.
3. Сформулировать критерии сравнения и оценки рассмотренных операционных систем.
4. Провести сравнительный анализ существующих решений по выделенным критериям.

1 Анализ предметной области

В данном разделе будут представлены основные определения, а также будет описана архитектура интернета вещей и поставлена проблема выбора операционной системы для устройств интернета вещей.

1.1 Основные определения

Интернет вещей [3] — это система взаимосвязанных вычислительных устройств, которые могут собирать и передавать данные по беспроводной сети без участия человека. В более широком смысле интернет вещей можно определить как концепцию, описывающую сеть физических объектов («вещей»), в которые встроены датчики, программное обеспечение и другие технологии для обмена данными с другими устройствами и системами через интернет [4]. Под вещью в данном контексте понимается предмет физического мира (физическая вещь) или информационного мира (виртуальная вещь), который может быть идентифицирован и интегрирован в сети связи [5].

Операционная система компьютера [6] — комплекс взаимосвязанных программ, который действует как интерфейс между приложениями и пользователями, с одной стороны, и аппаратурой компьютера, с другой стороны.

Сетевая операционная система [6] позволяет пользователю работать со своим компьютером как с автономным и добавляет к этому возможнось доступа к информационным и аппаратным ресурсам других компьютеров сети. В идеальном случае сетевая ОС должна представить пользователю сетевые ресурсы не в виде сети, а в виде ресурсов единой централизованной виртуальной машины. Для такой операционной системы используют специальное название — **распределённая ОС**, или **истинно распределённая ОС**.

Многозадачные ОС подразделяются на три типа в соответствии с использованными при их разработке критериями эффективности:

— системы пакетной обработки (ОС ЕС);

— системы разделения времени (ОС семейств Linux, Windows, MacOS);

— системы реального времени (QNX, FreeRTOS).

Современные операционные системы, проектируемые для устройств интернета вещей, являются либо системами реального времени, либо системами разделения времени. **Операционная система реального времени** [6] — это система, предназначенная для управления физическими объектами (процессами), которая способна обеспечить предсказуемое время реакции в ответ на изменение состояния управляемого объекта (процесса). **Система разделения времени** [6] — такая форма организации вычислительного процесса, при которой сразу несколько пользователей одновременно работают на компьютере, причём каждому из них кажется, что он получил компьютер в полное своё распоряжение. Главной целью и критерием эффективности систем разделения времени является обеспечение удобства и эффективности работы пользователей.

Ввиду большого многообразия устройств, используемых в распределённых вычислительных системах, при выборе ОС для интернета вещей большую роль играет переносимость. ОС называют **переносимой** (portable) [6], или **мобильной**, если её код может быть сравнительно легко перенесён с процессора одного типа на процессор другого типа и с аппаратной платформы одного типа на аппаратную платформу другого типа. Мобильность — не бинарное состояние, а понятие степени.

Наиболее общим подходом к структуризации операционной системы является разделение всех её модулей на две группы: модули ядра, выполняющие основные функции ОС, и вспомогательные модули ОС. Ядро ОС [6] представляет собой сложный многофункциональный комплекс, имеющий многослойную структуру. В развитии современных операционных систем наблюдается тенденция в сторону переноса кода в верхние слои и удалении при этом всего, что только возможно, из режима ядра, оставляя минимальное **микроядро** [7]. Обычно это осуществляется перекладыванием выполнения большинства задач операционной системы на средства пользовательских процессов. Такой подход способствует переносимости, расширяемости, повышению надёжности систе-

мы и создаёт хорошие предпосылки для поддержки распределённых приложений. Модель с микроядром хорошо подходит для поддержки **распределённых вычислений** [6], так как использует механизмы, аналогичные сетевым: взаимодействие клиентов и серверов путём обмена сообщениями. Серверы микроядерной ОС могут работать как на одном, так и на разных компьютерах. В этом случае при получении сообщения от приложения микроядро может обработать его самостоятельно и передать локальному серверу или же переслать по сети микроядру, работающему на другом компьютере. В контексте интернета вещей переход к распределённой обработке становится актуальным ввиду минимальных изменений в работе операционной системы — просто локальный транспорт заменяется сетевым.

Разработчики ОС для встраиваемых систем продолжили идею минимизации и пришли к концепции наноядра. **Наноядро** [8] — архитектура ядра операционной системы компьютеров, в рамках которой крайне упрощённое и минималистичное ядро выполняет лишь одну задачу — обработку аппаратных прерываний, генерируемых устройствами компьютера. После обработки прерываний от аппаратуры наноядро, в свою очередь, посылает информацию о результатах обработки (например, полученные с клавиатуры символы) вышележащему программному обеспечению при помощи того же механизма прерываний. Также часто реализуют минимальную поддержку потоков: создание и переключение. В некотором смысле концепция наноядра близка к концепции HAL (Hardware Abstraction Layer), предоставляя вышележащему ПО удобные механизмы абстракции от конкретных устройств и способов обработки их прерываний. Наноядро предлагает аппаратную абстракцию, но без системных служб. В современных микроядрах также отсутствуют системные службы, поэтому термины микроядро и наноядро стали аналогичными.

1.2 Архитектура интернета вещей

Определение архитектуры интернета вещей является предметом серьезных дискуссий. Одной из наиболее подробных архитектур является семиуров-

невая модель, предложенная компанией Cisco [9]. Ниже, в соответствии с рисунком 1 [10], описаны её уровни [11] [12].

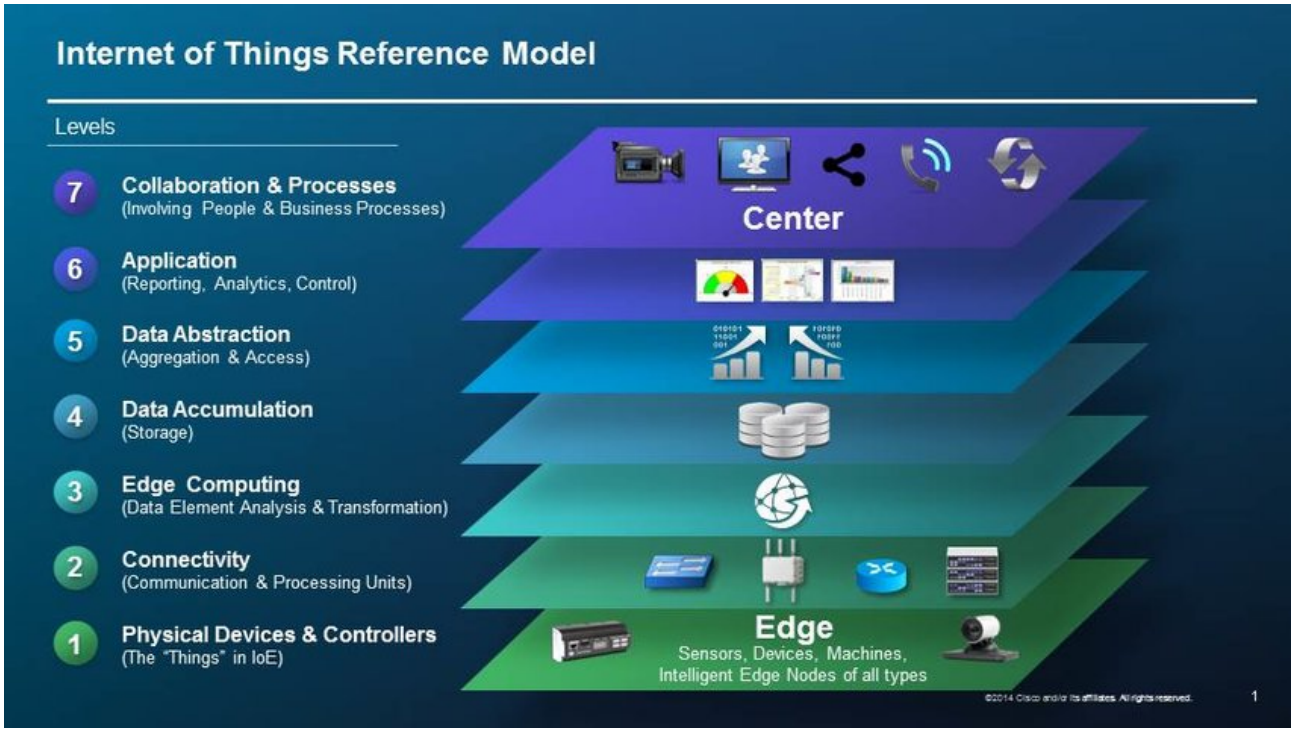


Рисунок 1 – Архитектура интернета вещей, предложенная Cisco

1. **Физические устройства и контроллеры** — это уровень, содержащий вещи в IoT. Сюда входит широкий спектр конечных устройств, которые могут отправлять или получать информацию (например, датчики и считыватели радиочастотной идентификации (RFID)).
2. **Соединение** — это уровень, содержащий все компоненты, способные передавать информацию. Передача может осуществляться между устройствами на первом уровне, между компонентами на этом уровне или между первым и третьим уровнем.
3. **Граничные (туманные) вычисления** (с английского Edge (Fog) computing) — это первый уровень, на котором происходит обработка данных. Здесь могут собираться и предварительно обрабатываться значительные объёмы информации до того, как они будут переданы в верхние уровни. Данный уровень также позволяет форматировать и декодировать данные до того, как они будут обработаны.

4. **Накопление данных** — это уровень, на котором данные сохраняются, чтобы приложения могли получить к ним доступ в случае необходимости. Как правило, необходимая обработка информации не может быть выполнена на сетевых скоростях, поэтому вычислительная система нуждается в промежуточном хранилище данных. Сохранённые данные также могут быть преобразованы и рекомбинированы, чтобы быть готовыми к использованию на более высоких уровнях. В результате на этом уровне данные в движении преобразуются в данные в состоянии покоя.
5. **Абстракция данных** — это уровень, позволяющий хранить данные более эффективным образом для повышения производительности более высоких уровней. На данном уровне над данными могут выполняться операции нормализации, индексирования, форматирования, проверки, консолидации, а также обеспечивается доступ к нескольким хранилищам данных.
6. **Приложения** — это уровень, на котором информация, накопленная ранее, интерпретируется приложениями. Именно на этом уровне располагается бизнес-логика приложений.
7. **Взаимодействие и процессы** — это уровень, объединяющий всё вместе. Система бесполезна, если информация, предоставляемая на этом уровне, не является полезной. Данные из интернета вещей должны использоваться для принятия обоснованных решений.

Операционные системы для интернета вещей принципиально меняют процесс разработки программного обеспечения для систем с многоуровневой архитектурой, описанной выше, так как позволяют разработчикам абстрагироваться от особенностей аппаратуры конкретных устройств и предоставляют шаблоны для создания приложений с определённой архитектурой.

1.3 Сферы применения интернета вещей

Чтобы поставить проблему выбора операционной системы для интернета вещей, необходимо обозначить круг задач, решаемых устройствами. Концепция интернета вещей находит своё применение во множестве различных сфер

жизнедеятельности человека: как в быту, так и в промышленности. Далее будут рассмотрены сферы, в которых применяются IoT-системы, и обозначены задачи, которые ставятся перед умными устройствами.

1.3.1 Интернет вещей в быту

В быту интернет вещей применяется при проектировании умных домов. Системы интернета вещей способны автоматизировать бытовые процессы и исключить непосредственное участие человека. К таким процессам можно отнести дистанционное управление бытовой техникой, музыкальными системами и системами освещения. Автоматизация может выполняться при помощи сбора и анализа статистики о бытовых процессах. Принципы функционирования IoT, описанные на примере бытовых процессов, можно перенести на бесконечное множество любых других, начиная от уличного освещения и управления светофорами, и до управления огромными предприятиями и городами [3].

1.3.2 Промышленный интернет вещей

Промышленный Интернет вещей (Industrial IoT, IIoT) [4] относится к применению технологии Интернета вещей в промышленных условиях. В последнее время в промышленности используется межмашинное взаимодействие (M2M) для обеспечения беспроводной автоматизации и управления. Но с появлением облачных и смежных технологий (таких как аналитика и машинное обучение) отрасли могут достичь нового уровня автоматизации и тем самым создать новые модели доходов и бизнеса. Организации, которые лучше всего подходят для IoT, — это те, которые могут выиграть от использования умных устройств в своих бизнес-процессах. Ниже приведены некоторые распространенные варианты использования IIoT.

1.3.2.1. Умные города

В умных городах используются такие устройства интернета вещей, как датчики и счетчики для сбора и анализа данных. Полученные данные могут использоваться для улучшения инфраструктуры, коммунального обслуживания и других городских сервисов.

1.3.2.2. Производства

Производители могут получить конкурентное преимущество, используя мониторинг производственных линий, чтобы обеспечить упреждающее обслуживание оборудования с помощью датчиков, обнаруживающих надвигающийся сбой. С помощью оповещений от датчиков производители могут быстро проверять оборудование и ремонтировать его в случае необходимости. Это позволяет компаниям сократить эксплуатационные расходы, а также увеличить время безотказной работы и повысить эффективность оборудования.

1.3.2.3. Транспорт и логистика

Транспортные и логистические системы могут извлекать выгоду из приложений IoT, отслеживая параметры перевозимых грузов. Например, можно отслеживать температуру перевозимых продуктов питания и напитков, цветочной и фармацевтической продукции, чтобы отправлять предупреждения, когда температура поднимается или падает до уровня, угрожающего качеству товаров.

1.3.2.4. Розничная торговля

Приложения IoT позволяют розничным компаниям управлять запасами, улучшать качество обслуживания клиентов, оптимизировать цепочку поставок и сокращать эксплуатационные расходы [4].

1.3.2.5. Государственный сектор

Преимущества IoT в государственном секторе и других средах, связанных с услугами, достаточно обширны [4]. Например, государственные коммунальные службы могут использовать приложения на базе интернета вещей для уведомления своих пользователей об отключениях и небольших перебоях в подаче воды и электроэнергии. Приложения IoT могут собирать данные о масштабах сбоев и управлять ресурсами, чтобы помогать коммунальным службам быстрее восстанавливать работу после сбоев.

1.3.2.6. Здравоохранение

Интернет вещей является важным аспектом **телемедицины** [3] (для обо-

значения интернета медицинских вещей иногда используется аббревиатура IoMT). Примеры его применения включают удаленную медицинскую диагностику, цифровую передачу медицинских изображений, видеоконсультации со специалистами и прочее. Приложения IoT также используются в носимых устройствах, которые могут контролировать здоровье человека и условия окружающей среды. Такие приложения не только помогают людям следить за состоянием своего здоровья, но и позволяют врачам удаленно наблюдать за пациентами.

1.3.2.7. Общая безопасность во всех отраслях

Помимо отслеживания физических показателей, IoT можно использовать для повышения безопасности труда [4]. Сотрудники опасных предприятий, таких как шахты, месторождения и электростанции, должны знать о возможном наступлении опасной ситуации. Когда они подключены к приложениям на основе датчиков IoT, они могут быть уведомлены об угрозах аварий, чтобы предпринять необходимые действия.

1.4 Проблема выбора операционной системы для интернета вещей

Операционные системы, предназначенные для интернета вещей, обладают различной функциональностью, которая определяет преимущества и недостатки каждого конкретного решения. Выбранная операционная система должна полностью соответствовать требованиям и ограничениям, предъявляемым к устройствам. Можно выделить четыре основных аспекта, которые разработчики устройств учитывают при выборе ОС для периферийных устройств IoT [13].

1. **Какой уровень надежности и долгосрочной поддержки необходим?**
   Под надёжностью в данном случае понимается соответствие операционной системы определенным стандартам и сертификациям. Необходимо, чтобы выбранная операционная система обеспечивала варианты, подходящие для конкретных отраслевых стандартов и требований.
2. **Какие требования предъявляются к производительности?**
   Выбранная операционная система должна соответствовать требованиям устрой-

ства к вычислительной мощности и производительности в реальном времени. Также при ответе на данный вопрос стоит учитывать другие аспекты, связанные с производительностью и оказывающие влияние на выбор ОС. К таким можно отнести энергопотребление и объём памяти устройства, а также требования к времени отклика системы на внешние события.

3. **Обеспечивает ли операционная система безопасность устройства?**

   Безопасность — один из основных факторов, учитываемых при разработке устройств интернета вещей. Это касается и используемой ОС, так как от неё зависит целостность данных, собираемых устройствами. Если система не обеспечивает необходимый уровень безопасности, то это может привести к порче или краже данных, нарушению запланированных процессов.

4. **Является ли выбранная операционная система масштабируемой?**

   В связи с тем, что операционные системы, как и любое программное обеспечение, меняются вместе с требованиями к функциям, которые они должны предоставлять. Разработка IoT-устройства с масштабируемой ОС позволяет в будущем адаптироваться к другим задачам без внесения значительных изменений. Масштабируемые ОС могут обрабатывать дополнительные ресурсы без изменения скорости вывода и охватывать несколько устройств.

2 Описание существующих решений

В данном разделе будет проведён анализ существующих операционных систем для устройств интернета вещей. Рассматриваемые операционные системы будут относиться к одному из двух типов: ОС реального времени и ОС разделения времени.

2.1 Операционные системы реального времени

2.1.1 Azure RTOS

Microsoft Azure RTOS ThreadX [14] — это операционная система реального времени (Real-Time Operating System, RTOS) для интернета вещей и пограничных устройств, работающих на микроконтроллерах (Micro Controller Unit, MCU). Azure RTOS разработана для поддержки устройств с жесткими ограничениями по ресурсам (как правило, работающих от аккумуляторов и имеющих менее 64 КБ флэш-памяти).

Azure RTOS обеспечивает сертифицированную по Common Criteria среду безопасности EAL4+, включая полную безопасность на уровне IP через IPsec и безопасность на уровне сокетов через TLS и DTLS. Криптобиблиотека прошла сертификацию FIPS 140-2. Также реализованы аппаратные криптографические возможности, защита памяти с помощью ThreadX MODULES и поддержка функций безопасности TrustZone ARM ARMv8-M.

Одной из отличительных особенностей данной ОС является архитектура ядра, организованного по структуре пикоядра (picokernel) [15]. При таком подходе службы ОС размещаются на одном уровне, что устраняет ненужные временные затраты при вызове функций.

Azure RTOS ThreadX разработана для глубоко встраиваемых приложений. Название ThreadX происходит от потоков, которые используются в качестве исполняемых элементов, а «X» обозначает переключение потоков. ThreadX поддерживает среды с многоядерными процессорами посредством асимметричной многопроцессорной обработки или симметричной многопроцессорной обра-

ботки. ThreadX распространяется с использованием маркетинговой модели, в которой исходный код бесплатен, а лицензии предоставляются бесплатно.

Azure RTOS используется в специализированном оборудовании, таком, как устройства беспроводной связи, принтеры, модемы, устройства хранения данных, медицинские устройства, интеллектуальные датчики.

2.1.2 Azure Sphere

Microsoft Azure Sphere [16] — это специализированная операционная система для микроконтроллеров на базе Linux, созданная Microsoft для работы на чипе, сертифицированном для Azure Sphere, и для подключения к службе безопасности Azure Sphere. Операционная система Azure Sphere предоставляет платформу для разработки приложений для интернета вещей , включая как приложения высокого уровня, так и приложения, поддерживающие работу в реальном времени. Это первая операционная система с ядром Linux [17] , которую Microsoft публично выпустила, и вторая Unix-подобная операционная система, разработанная компанией для публичного пользования.

Служба безопасности Azure Sphere [18] [19], представляет собой облачную службу, которая обеспечивает обслуживание, обновление и контроль чипов, сертифицированных Azure Sphere. Служба безопасности Azure Sphere устанавливает безопасное соединение между устройствами и интернетом и/или облачными службами и обеспечивает безопасную загрузку. Основная цель контакта между устройством Azure Sphere и службой безопасности Azure Sphere — проверка подлинности удостоверения устройства, обеспечение целостности и доверия к системному программному обеспечению, а также подтверждение того, что на устройстве работает доверенная кодовая база. Служба также предоставляет безопасный канал, используемый корпорацией Майкрософт для автоматической загрузки и установки обновлений ОС Azure Sphere и обновлений клиентских приложений на развернутые устройства.

Согласно рисунку 2 [20], система, основанная на Azure Sphere, состоит из трех компонентов, главный из которых — микроконтроллер (MCU), поддержи-

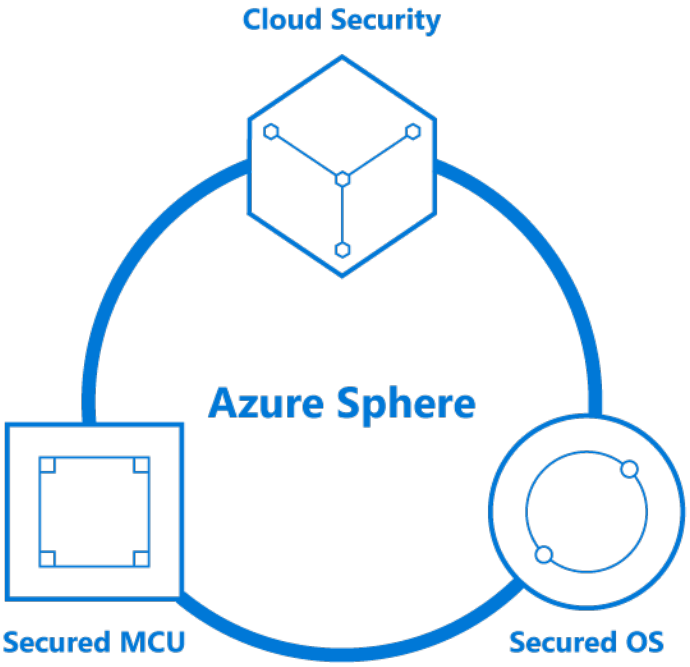


Рисунок 2 – Компоненты системы на базе Azure Sphere

вающий семь важнейших аппаратных функций, которые, по мнению Microsoft, являются необходимой основой для создания безопасных систем. К ним относятся поддержка невоспроизводимых ключей шифрования, защищённых аппаратными средствами, возможность обновления системного программного обеспечения и аппаратное разделение программных компонентов. У Microsoft есть определенный опыт создания таких систем, в частности, Xbox, который разработан для защищенного от взлома оборудования с возможностью безопасного обновления.

2.1.3 Amazon FreeRTOS

FreeRTOS [21] — это облачная операционная система реального времени для устройств интернета вещей с открытым исходным кодом. FreeRTOS можно бесплатно использовать по лицензии MIT на программное обеспечение с открытым исходным кодом. Для этой системы разработано больше 40 вариантов архитектуры, что предоставляет разработчикам широкий выбор аппаратного обеспечения наряду с набором готовых программных библиотек.

Ниже представлены ключевые особенности Amazon FreeRTOS [22].

1. **Локальное подключение**.

   Локальное подключение к периферийному устройству, работающему с AWS IoT Greengrass, позволяет устройствам под управлением FreeRTOS продолжать обмениваться информацией, собирать данные и выполнять необходимые действия без подключения к облаку. Устройства под управлением FreeRTOS могут подключаться к локальной сети через Wi-Fi или Ethernet, используя библиотеки для локальных подключений.

2. **Подключение к облаку**.

   Подключение к облаку позволяет собирать данные и выполнять различные задачи на устройствах на основе микроконтроллеров, а также использовать на устройствах приложения Интернета вещей и другие сервисы AWS Cloud. Устройства под управлением FreeRTOS можно подключать к AWS IoT Core [23] с использованием HTTP или возможностей обмена сообщениями на основе MQTT. MQTT — это нетребовательный к ресурсам протокол, который обеспечивает связь с ограниченными в ресурсах устройствами на основе микроконтроллеров.

3. **Поддержка теней устройств AWS IoT Core** [23].

   FreeRTOS также поддерживает API теней устройств и библиотеку теней устройств AWS IoT Core. С помощью функции теней устройств можно создать постоянную виртуальную версию каждого устройства (так называемую «тень»), содержащую его последнее состояние и позволяющую приложениям или другим устройствам считывать сообщения от данного устройства и взаимодействовать с ним.

4. **Поддержка AWS IoT Device Defender** [24].

   В FreeRTOS имеется библиотека для работы с AWS IoT Device Defender. Интеграция с сервисом AWS IoT Device Defender позволяет просто отслеживать метрики на стороне устройства для обнаружения аномального поведения (когда эти метрики отклоняются от ожидаемых значений). AWS IoT Device Defender непрерывно проверяет конфигурации Интернета ве-

щей, связанные с устройствами с FreeRTOS, чтобы обеспечить безопасность устройств.

5. **Беспроводные обновления**.

   AWS IoT Device Management [25] можно использовать с устройствами с FreeRTOS в качестве интегрированного решения для беспроводных обновлений. FreeRTOS уменьшает требования к памяти при развертывании беспроводных обновлений на устройствах на основе микроконтроллеров за счет передачи этих обновлений через единое TLS-соединение, совместно используемое с другими сеансами связи AWS IoT Core. Требуется лишь предоставить образ встроенного ПО, выбрать устройства для обновления, задать метод подписания кода и запланировать время обновления – все эти действия выполняются в консоли AWS IoT Device Management.

6. **Долговременная поддержка FreeRTOS Long Term Support (LTS)**.

   После выпуска долговременной поддержки FreeRTOS Long Term Support (LTS) FreeRTOS будет обеспечивать стабильность функций, а также обновления безопасности и исправления критических ошибок в течение двух лет.

2.1.4 Zephyr

Zephyr OS [26] — это масштабируемая операционная система реального времени (RTOS) для встраиваемых устройств с открытым исходным кодом. Кроссплатформенная архитектура ориентирована на разработку ПО как для микроконтроллеров, так и для систем на кристалле (System on Chip, SoC).

Архитектуры процессоров, поддерживаемые Zephyr OS:

— ARCv2 (EM и HS) и ARCv3 (HS6X);
— ARMv6-M, ARMv7-M и ARMv8-M (Cortex-M);
— ARMv7-A и ARMv8-A (Cortex-A, 32- и 64-разрядные версии);
— ARMv7-R, ARMv8-R (Cortex-R, 32- и 64-разрядные версии);
— Intel x86 (32- и 64-разрядная версии);
— MIPS (спецификация MIPS32 Release 1);
— NIOS II поколения 2;

— RISC-V (32- и 64-разрядные версии);

— SPARC V8;

— Tensilica Xtensa.

Zephyr [27] основана на малогабаритном ядре, предназначенном для использования во встроенных системах с ограниченными ресурсами: от простых встроенных датчиков окружающей среды и светодиодных носимых устройств до сложных встроенных контроллеров, смарт-часов и беспроводных приложений IoT.

Ниже представлены ключевые особенности Zephyr [26] [27].

1. **Несколько алгоритмов планирования**.

   Zephyr предоставляет полный набор вариантов планирования потоков, среди которых совместное и упреждающее планирование.

2. **Широкие возможности настройки**.

   Zephyr позволяет приложению включать только те возможности, которые ему нужны, по мере необходимости, а также указывать их количество и размер.

3. **Защита памяти**.

   Zephyr реализует настраиваемую защиту от переполнения стека для конкретной архитектуры, отслеживание разрешений объектов ядра и драйверов устройств, а также изоляцию потоков с защитой памяти на уровне потоков в архитектурах x86, ARC и ARM, пользовательском пространстве и доменах памяти.

   Для платформ без MMU/MPU и устройств с ограниченным объемом памяти поддерживается объединение кода конкретного приложения с пользовательским ядром для создания монолитного образа, который загружается и выполняется на оборудовании системы. И код приложения, и код ядра выполняются в одном общем адресном пространстве.

4. **Определение ресурса времени компиляции**.

   Zephyr позволяет определять системные ресурсы во время компиляции, что

уменьшает размер кода и повышает производительность для систем с ограниченными ресурсами.

5. **Поддержка дерева устройств**.

   Использование дерева устройств [28] для описания оборудования. Информация из дерева устройств используется для создания образа приложения.

6. **Собственный сетевой стек, поддерживающий несколько протоколов**.

   Полнофункциональная и оптимизированная сетевая поддержка, включая поддержку совместимых сокетов LwM2M и BSD. Также предоставляется поддержка OpenThread (на чипсетах Nordic) — ячеистая сеть (mesh network), предназначенная для безопасного и надежного подключения сотен продуктов по всему дому.

7. **Поддержка Bluetooth 5.0**.

   Совместимость с Bluetooth 5.0 (ESR10) и поддержка Bluetooth Low Energy Controller (LE Link Layer).

8. **Энергонезависимое хранилище** (Non-Volatile Storage, NVS).

   NVS позволяет хранить двоичные BLOB-объекты, строки, целые числа, длинные числа и любую их комбинацию.

2.1.5 ОСРВ МАКС

ОСРВ МАКС [29] — это операционная система реального времени для встраиваемых систем интернета вещей: умных устройств, шлюзов и автономных компонентов. Полное название: «Встраиваемая операционная система для Мультиагентных Когерентных Систем с повышенными требованиями к надежности».

Данная операционная система является полностью оригинальной российской разработкой [30]: в проекте не используются фрагменты других ОСРВ, что позволило воплотить самые современные архитектурные решения. ОСРВ МАКС входит в реестр российского ПО [31]. Целевыми платформами рассматриваемой операционной системы являются микроконтроллеры производства АО «ПКК Миландр» [32] и STMicroelectronics [33], основанные на чипах ARM

Cortex M1/M3/M4 и Analog Devices TigerSHARC. API ОСРВ МАКС соответствует POSIX Threads-стандарту [34].

Хорошей предпосылкой для уникальности ОСРВ МАКС [35] стал тот факт, что в своем продукте можно было реализовать то, что в других операционных системах сделать уже может быть слишком сложно или даже поздно в связи с их долгим существованием на рынке и устоявшейся архитектурой решений. В итоге, ОСРВ МАКС не только реализует весь классический функционал операционных систем данного типа, но и обладает рядом уникальных возможностей. Например, данная операционная система ориентируется не только на обеспечение работы одного устройства (микропроцессора, микроконтроллера), но и на взаимодействие устройств (отсюда и «мультиагентность» в названии ОС). Это позволяет упростить создание необходимых во встраиваемых системах механизмов. В основе этих возможностей лежит концепция распределенной общей памяти. Несколько независимых устройств могут обмениваться данными и синхронизировать их так, будто все они имеют физический доступ к общей памяти.

Применения ОСРВ МАКС [30]:

— робототехника и БПЛА[1]: системы управления, телеметрии и позиционирования;
— поддержка технологий интернета вещей (оптимальная конфигурация распределённой системы, автономное функционирование системы, поддержка масштабируемости;
— системы «умного дома» (управление электропитанием и освещением, управление климатом, системы мониторинга и безопасности;
— потребительская электроника и бытовая техника;
— mesh-сети.

2.1.6 Huawei LiteOS

Huawei LiteOS [36] — это операционная система реального времени с открытым исходным кодом, разработанная для сферы IoT и являющаяся частью

[1] БПЛА — беспилотные летательные аппараты.

операционной системы Huawei IoT operating system kernel.

Помимо базового ядра Huawei LiteOS предлагает расширения, которые включают поддержку C++, динамическую загрузку программ, низкое энергопотребление. Данная операционная система поддерживает спящий режим, который позволяет значительно снизить энергопотребление системы. Huawei LiteOS также предоставляет полный набор протоколов межсетевого взаимодействия поверх lwm2m, lwip и lwip для взаимодействия приложений. Рассматриваемая операционная система совместима с микроконтроллерами, оснащёнными процессорами ARM Cortex-M0, Cortex-M3, Cortex-M4, Cortex-M7, Cortex-A.

Ниже представлены ключевые особенности Huawei LiteOS [37].

1. **Микроядерная архитектура**.
2. **Поддержка сенсорных платформ**.
3. **Виртуальная машина на базе JavaScript**.

   Малогабаритное ПЗУ с низким использованием памяти обеспечивает независимое разделение пространства пользователя и приложений для обеспечения их безопасности.
4. **Наличие IoT-ориентированной платформы для разработки приложений**.

   Платформа предоставляет средства разработки ПО на JS-фреймворках для виртуальной машины на базе JavaScript.

Применения Huawei LiteOS [38]:

— мобильные камеры;
— умные парковки (Smart Parking);
— умный дом;
— умные бытовые счётчики;
— умное освещение.

2.2 Операционные системы разделения времени

2.2.1 Windows 10 IoT

Windows 10 IoT [39] [13] — это семейство операционных систем, которое включает три выпуска: Core, Enterprise, Server. Они различаются доступными функциями и поддерживаемыми драйверами. Все выпуски Windows IoT обеспечивают 10-летнюю долгосрочную поддержку и взаимодействие с другими службами и платформами Azure [40].

Устройства фиксированного назначения, использующие встраиваемые ОС Windows 10 IoT, широко используются в следующих сферах:

— торговля и банковская сфера:
  платежные терминалы;
  банкоматы;
  постаматы;
  кассы самообслуживания;
  автоматические депозитарные машины (АДМ);
  POS-системы;
— системы безопасности и видеонаблюдения (CCTV);
— логистика (управление транспортными потоками);
— системы пожаротушения и оповещения;
— DigitalSignage (цифровые вывески);
— системы электронных очередей.

Ниже представлено описание версий Windows 10 IoT.

1. **Windows 10 IoT Core** [41] — это версия Windows 10, оптимизированная для небольших устройств с дисплеем или без него, которые работают как на устройствах ARM, так и на устройствах x86/x64. Документация Windows IoT Core содержит информацию о подключении, управлении, обновлении, защите устройств и т. д.
2. **Windows IoT Enterprise** [42] — это полная версия Windows Enterprise [43],

которая обеспечивает корпоративную управляемость и безопасность для решений IoT. Windows IoT Enterprise использует все преимущества всемирной экосистемы Windows. В отличие от Windows 10 IoT Core, совместима только с архитектурами процессоров x86 и x64.

3. **Windows Server IoT** [44] — это полная версия Windows Server [45], которая обеспечивает корпоративную управляемость и безопасность для решений IoT. Windows Server IoT использует все преимущества экосистемы Windows. Данная операционная система является двоично совместимой [6] с Windows Server, поэтому возможно использование тех же инструментов разработки и управления, которые используются на серверах общего назначения. Однако когда дело доходит до лицензирования и распространения, версия общего назначения и версия IoT различаются. Windows Server IoT лицензируется только через OEM-канал с особыми выделенными правами на использование.

   Windows Server хорошо известна как серверная операционная система, используемая малыми предприятиями по всему миру. Что менее известно, так это то, что в течение многих лет Windows Server также использовался во многих специализированных решениях для розничной торговли, производства, здравоохранения и т. д. Windows Server IoT позволяет создавать решения фиксированного назначения с определенными допущениями и ограничениями в лицензионном соглашении.

2.2.2 Contiki-NG

Contiki-NG [46] — это кроссплатформенная операционная система с открытым исходным кодом для устройств с ограниченными ресурсами в интернете вещей. Она ориентирована на надежную связь с низким энергопотреблением и стандартные протоколы, такие как IPv6/6LoWPAN, 6TiSCH, RPL и CoAP. Contiki-NG поставляется с обширной документацией, учебными пособиями, дорожной картой, циклом выпуска и четко определенным потоком разработки для интеграции вкладов сообщества.

Contiki спроектирована для встраиваемых систем с ограниченным объёмом памяти. Объем кода составляет порядка 100 кБ, а использование памяти может быть настроено так, чтобы не превышать 10 кБ. При конфигурации по умолчанию Contiki использует 2 килобайта ОЗУ и 40 килобайт ПЗУ. ОС состоит из ядра, которое управляется событиями, программы во время исполнения загружаются и выгружаются динамически. Процессы используют облегчённую потоковую модель — **протопотоки** (protothread), которые обеспечивают линейный потоковый стиль инициализации ядра.

Система работает на различных платформах на базе энергоэффективных архитектур, таких как ARM Cortex-M3/M4 и Texas Instruments MSP430.

Существующие области применения Contiki включают системы уличного освещения, звукового мониторинга для умных городов, радиационного контроля и сигнализации.

Ниже представлены ключевые особенности Contiki-NG [47].

1. **Ядро, основанное на событиях**.

   Выполнение или реализация кода осуществляется обработчиком событий. Это означает, что процесс полностью зависит от события и никогда не прерывается другим процессом. Модель многопоточности сравнивается с моделью стека, в которой для параллельных процессов требуется меньше вычислений и памяти.

2. **Протопотоки** (Protothread).

   В ОС Contiki поддерживаются протопотоки, где для написания сложных для понимания и сопровождения кодов или программ используется событийно-ориентированный и явный подход. Это сохраняет высокоуровневую реализацию функций с абстракцией языка программирования и без накладных потоков выполняет условную блокировку. Для одного протопотока в Contiki OS требуется 2 байта оперативной памяти.

3. **Микро IP** (Micro IP, uIP).

   Данная технология ориентирована на протоколы TCP, ICMP и IP. Имеет

минимальные отличия от полного стека TCP/IP и предназначена для датчиков с ограниченными ресурсами.

2.2.3 Mbed OS

Arm Mbed OS [48] — это бесплатная операционная система IoT с открытым исходным кодом, которая включает все необходимые функции для разработки продуктов IoT на базе аппаратного обеспечения Arm Cortex-M, включая возможности машинного обучения, безопасность, стеки подключения, ядро RTOS и драйверы для датчиков и устройств ввода-вывода. Данная операционная система разработана для интернета вещей. Она интегрирована со стеками подключения, машинного обучения, сетевых технологий и безопасности, а также поддерживается программными библиотеками, аппаратными средствами разработки, учебными пособиями и примерами [49].

Mbed OS ориентирована на микроконтроллеры с процессорами ARM серии Cortex M и интегрирована со следующими облачными сервисами:

— Google Cloud Platform [50];
— Amazon Web Services (AWS) [51];
— Microsoft Azure [52];

Применения Mbed OS [53]:

— умное уличное освещение (Aaeon Node);
— умные городские велосипедные фонари (See.Sense);
— монитор промышленных активов (Agora EPM2M-AG-CELL).

2.2.4 KasperskyOS

KasperskyOS [54] — проприетарная частично POSIX-совместимая микроядерная операционная система. Она предназначена для разработки IT-продуктов для отраслей с повышенными требованиями к кибербезопасности, надежности и предсказуемости работы. Цель KasperskyOS — обеспечить защиту IT-систем от вредоносного кода и эксплуатации уязвимостей, а также снизить риски, связанные с ошибками в коде, случайными или намеренными повреждающими действиями.

Ниже представлены ключевые особенности KasperskyOS [55] [56].

1. **Микроядерность**.

   KasperskyOS является микроядерной операционной системой. Ядро предоставляет минимальную функциональность, включая планирование исполнения программ, управление памятью и вводом-выводом. Код драйверов устройств, файловых систем, сетевых протоколов и другого системного ПО исполняется в пользовательском режиме (вне контекста ядра).

2. **Безопасная микроядерная ОС**.

   KasperskyOS создана на базе специально разработанного с нуля микроядра и не является модификацией какой-либо из существующих ОС. Разработана на базе принципов MILS и FLASK и имеет в своем составе гибкую систему контроля доступа (KSS). Процессы могут предоставлять службы другим процессам и/или использовать службы других процессов (в том числе службы ядра) через механизм IPC [57], следуя правилам безопасности, которые регулируют взаимодействия процесса с другими процессами и ядром. На рисунке 3 [56] представлена схема взаимодействия процессов между собой и с ядром в KasperskyOS.

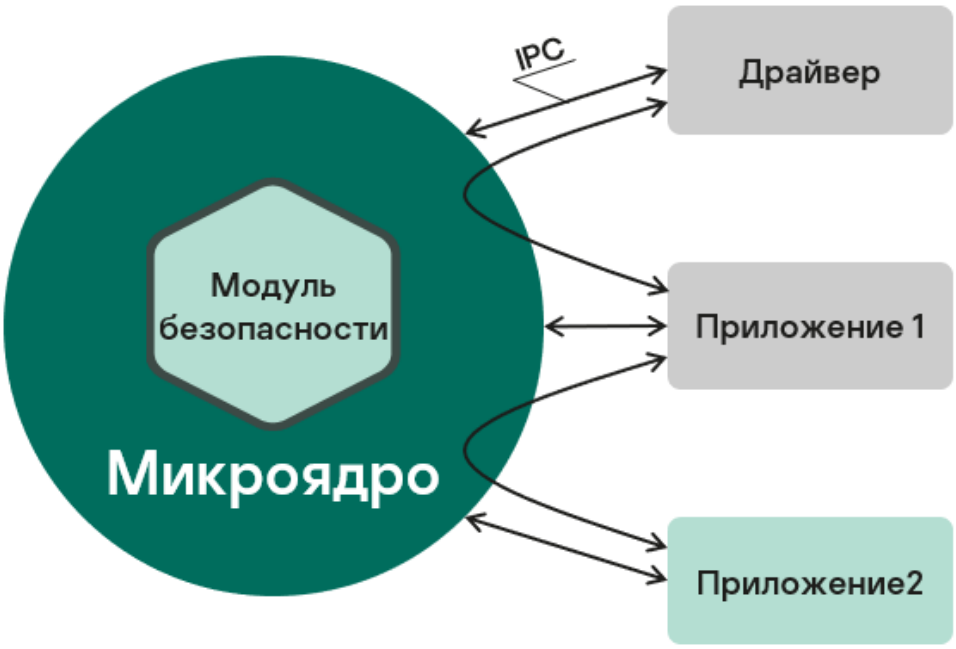


Рисунок 3 – Взаимодействие процессов между собой и с ядром в KasperskyOS

3. **Поддержка POSIX**.

   Поддержка около 98% POSIX API [55].

4. **Портирование**.

   KasperskyOS работает на платформах Intel x86/x86-64, ARMv5, ARMv7, ARMv8 и MIPS32. Поддержка других аппаратных платформ с MMU может быть организована по запросу.

5. **Поддержка аппаратной виртуализации**.

   KasperskyOS может использоваться как основа для безопасного гипервизора[2] с поддержкой технологий Intel VT-x и VT-d.

6. **Сокращение поверхности потенциальной атаки**.

   Разделение приложений на домены безопасности и полный контроль междоменных взаимодействий позволяет безопасно использовать потенциально уязвимые и/или недоверенные приложения.

2.2.5 TinyOS

TinyOS [58] — это операционная система с открытым исходным кодом под лицензией BSD, предназначенная для беспроводных устройств с низким энергопотреблением, таких как те, которые используются в сенсорных сетях (Wireless sensor networks), повсеместных вычислениях (ubiquitous computing), персональных сетях, умных зданиях и счетчиках.

Архитектура TinyOS включает две главные функциональные составляющие: планировщик задач и компонент [59]. Понятие «компонент» в TinyOS несколько отличается от общепринятого. Так, интерфейс компонента TinyOS состоит из двух частей: верхней (upper), предоставляемой этим компонентом как провайдером, и нижней (lower), требуемой для его функционирования. Обе части содержат описания команд и событий.

TinyOS имеет компонентную модель программирования [60], кодифицированную языком NesC, являющегося диалектом языка C. TinyOS не является

[2]Гипервизор — программа или аппаратная схема, обеспечивающая одновременную работу нескольких ОС на одном компьютере.

ОС в традиционном понимании. Это программная платформа для встраиваемых систем и набор компонентов, которые позволяют встраивать ОС, специфичную для каждого приложения. Типичное приложение имеет размер около 15K, из которых базовая ОС занимает около 400 байт. Размер самого большого приложения, системы запросов, подобной базе данных, составляет около 64 Кбайт.

Приложение на TinyOS представляет собой граф компонентов, каждый из которых является независимым объектом, который предоставляет один или несколько интерфейсов. Компоненты имеют три вычислительные абстракции: команды, события и задачи. Команды и события — это механизмы для межкомпонентного взаимодействия, в то время как задачи используются для выражения внутрикомпонентного параллелизма.

2.2.6 Ubuntu Core

Ubuntu Core [61] — это версия операционной системы Ubuntu [62], разработанная и спроектированная для IoT и встраиваемых систем. Данная операционная система обновляет себя и свои приложения автоматически. Пакеты Snap используются исключительно для создания замкнутой и основанной на транзакциях системы.

Ubuntu Core можно запускать как виртуальную машину или на следующих платформах:

— Raspberry Pi 2 and 3;
— Compute Module 3;
— Qualcomm DragonBoard 410c;
— Intel NUC;
— Intel Joule;
— Samsung Artik;
— KVM;
— Amazon Web Services (AWS);
— Microsoft Azure;

— Google Cloud Platform.

Ubuntu Core [63] — это транзакционная версия ОС Ubuntu Linux, созданная специально для устройств интернета вещей и развертывания больших контейнеров. Эта ОС использует то же ядро, библиотеки и системное программное обеспечение, что и стандартная Ubuntu, но в гораздо меньших масштабах.

Транзакционные операционные системы делят работу на полные, неделимые операции. Ubuntu Core работает за счет использования моментальных пакетов. Snaps — это zip-файлы, которые содержат контейнерное приложение и его зависимости, а также инструкции по безопасному запуску и взаимодействию с другим программным обеспечением. Snap запускаются на любом Linux (для ПК, сервера или облачного устройства), изолированном от базовой ОС для безопасной установки приложения.

Snap доступны только для чтения и являются неизменяемыми, что предотвращает любые изменения при установке в системе. Наряду с приложением и его зависимостями снимки содержат два отдельных пространства для хранения с возможностью записи, одно из которых является версионным и сохраняет копии любых обновлений данных, а другое хранит большие объемы статических данных, не требующих повторного дублирования.

2.2.7 Raspbian

Raspbian [64] — это операционная система с открытым исходным кодом, основанная на Debian и оптимизированная для аппаратной платформы Raspberry Pi.

Raspbian — это неофициальный перенос Debian Wheezy armhf [65] с настройками компиляции, скорректированными для получения оптимизированного кода «hard float», который будет работать на Raspberry Pi. Это обеспечивает значительно более высокую производительность для приложений, интенсивно использующих арифметические операции с плавающей запятой. Все остальные приложения также получают определенный прирост производительности за счет использования расширенных инструкций процессора ARMv6 в Raspberry

Pi.

Ниже представлены ключевые особенности Raspbian.

1. **Пользовательский интерфейс**.

   Raspbian имеет среду рабочего стола PIXEL, основанную на LXDE [66], которая похожа на многие распространенные рабочие столы, такие как macOS и Microsoft Windows. Рабочий стол имеет фоновое изображение. Строка меню расположена вверху и содержит меню приложений и ярлыки для веб-браузера (Chromium), файлового менеджера и терминала. На другом конце строки меню отображаются меню Bluetooth , меню Wi-Fi, регулятор громкости и часы. Рабочий стол также можно изменить по сравнению с его внешним видом по умолчанию, например, изменить положение строки меню [67].

2. **Управление пакетами**.

   Пакеты можно устанавливать через пакетный менеджер APT [68], используемый во всех операционных системах, основанных на Debian. Взаимодействие с APT возможно как через консольный, так и через графический интерфейс.

3. **Компоненты**.

   PCManFM — это файловый браузер, обеспечивающий быстрый доступ ко всем областям компьютера.

   Raspberry Pi OS распространяется с веб-браузером Chromium. Встроенный браузер поставляется с предустановленными uBlock Origin и h264ify.

   Raspberry Pi OS поставляется с IDE, такими как Thonny Python IDE, Mu Editor и Greenfoot. Он также поставляется с образовательным программным обеспечением, таким как Scratch и Bookshelf.

3 Классификация существующих решений

В данном разделе будут описаны критерии сравнения операционных систем для устройств интернета вещей.

3.1 Критерии сравнения операционных систем

Анализ операционных систем для интернета вещей является задачей классификации. Ниже приведены критерии для их сравнения.

1. **Тип ядра**.

   Тип ядра ОС определяет её архитектуру и является ключевым фактором, влияющим на производительность и масштабируемость ОС. Рассматриваются следующие типы ядра: монолитное ядро, микроядро, наноядро и гибридное ядро[3].

2. **Тип лицензии**.

   Тип лицензии определяет использование и распространение программного обеспечения, защищённого авторским правом. Лицензия является гарантией того, что издатель ПО, которому принадлежат исключительные права на программу, не подаст в суд на того, кто её использует.

3. **Поддержка POSIX**.

   Программные интерфейсы приложений, распространённые в традиционных моделях операционных систем, ограничивают переносимость программного кода и могут значительно снизить экономические показатели эффективности [69]. Степень POSIX-соответствия операционных систем характеризуют стандартизованные интерфейсы операционных систем и приложений и, следовательно, степень их мобильности на уровне исходного языка. Многие операционные системы, претендующие на POSIX-совместимость, зачастую поддерживают только небольшие подмножества этого стандарта, поэтому рассматриваемый критерий будет представлен как понятие степе-

[3]Не может быть классифицировано ни как монолитное ядро, ни как микроядро. Гибридное ядро объединяет аспекты этих двух типов.

ни: поддержка POSIX может быть полной, частичной или полностью отсутствовать. В таблице сравнения будут использоваться обозначения «+», «+/-» и «-» соответственно.

4. **Тип многозадачности**.

   Многозадачность ОС обеспечивает параллельную (или псевдопараллельную) обработку нескольких задач. По типу многозадачности рассматриваемые операционные системы можно разделить на три категории:

   — ОС, реализующие вытесняющую многозадачность;

   — ОС, реализующие кооперативную многозадачность;

   — ОС, реализующие одновременно оба типа многозадачности.

   Кооперативная многозадачность снижает накладные расходы работы системы за счет исключения лишних переключений задач и использования объектов синхронизации. Режим позволяет заранее предопределить последовательность выполнения задач, возлагая, однако, на разработчика ответственность за установку точек передачи управления между задачами.

   Вытесняющая многозадачность избавляет разработчика от необходимости планирования задач вручную (планирование с учетом точек и последовательности передачи управления между задачами) и обеспечивает оптимальную загрузку процессора. При вытесняющей многозадачности планирование задач осуществляется на основании их приоритетов, что позволяет разработчику определить критичность каждой задачи в приложении — аппаратные ресурсы будут распределяться между задачами в зависимости от их критичности.

5. **Кроссплатформенность**.

   Кроссплатформенность определяет способность операционной системы работать с несколькими аппаратными платформами. Для сравнения ОС по данному критерию будет рассматриваться бинарный признак, определяющий совместимость операционной системы с одной или несколькими архитектурами процессоров. В таблице сравнения будут использоваться обо-

значения «-» и «+» соответственно.

6. **Применение**.

   Операционные системы разрабатываются для конкретных целей, определяющих её специфику. По этой причине при выборе ОС для устройств интернета вещей важно понимать, на какие сферы применения она нацелена. Для сравнения операционные системы будут разделены на две категории: для промышленного интернета вещей (Industrial IoT, IIoT) и для интернета вещей в быту (Home IoT, HIoT). Решение о том, к какой категории IoT отнести ту или иную ОС, принималось на основе текущих сфер их применения. Если об этом не имеется достоверных сведений, то на основе характеристик оценивалась возможность масштабирования решений на базе рассматриваемой ОС до промышленных масштабов.

Описанные выше характеристики операционных систем, как правило, декларируются в документациях к ним. Также стоит отметить, что при сравнении операционных систем не рассматривается наличие сетевых функций, так как все ОС, рассматриваемые в данной работе, являются сетевыми.

3.2 Сравнительный анализ операционных систем

Результаты сравнения операционных систем для устройств интернета вещей приведены в таблице 1. Для краткости записи в данной таблице используются следующие обозначения описанных критериев:

— К1 — тип ядра (монолитное, гибридное, микроядро или наноядро);
— К2 — тип лицензии;
— К3 — поддержка POSIX (полная, частичная или полностью отсутствует);
— К4 — тип многозадачности (вытесняющая или кооперативная);
— К5 — кроссплатформенность (поддерживается одна или несколько аппаратных платформ);
— К6 — применение (промышленный интернет вещей или интернет вещей в быту).

Таблица 1 – Сравнение операционных систем для устройств интернета вещей

ОС	К1	К2	К3	К4	К5	К6
Azure RTOS	Наноядро	Microsoft Software License	-	Вытесняющая	+	IIoT
Azure Sphere	Монолитное	GPL-2.0	+/-	Вытесняющая	+	IIoT
Amazon FreeRTOS	Микроядро	MIT	+	Вытесняющая	+	IIoT
Zephyr	Наноядро	Apache Licence 2.0	+/-	Вытесняющая и кооперативная	+	IIoT
ОСРВ МАКС	Монолитное	BSD 3-Clause	+/-	Вытесняющая и кооперативная	+	IIoT
Huawei LiteOS	Микроядро	BSD 3-Clause	+	Вытесняющая	-	IIoT
Windows 10 IoT	Гибридное	Microsoft Software License	-	Вытесняющая	+	IIoT
Contiki-NG	Монолитное	BSD 3-Clause	+/-	Кооперативная	+	IIoT
Mbed OS	Монолитное	Apache Licence 2.0	+	Кооперативная	-	IIoT
KasperskyOS	Микроядро	Проприе-тарная	+/-	Не деклариро-ван	+	IIoT
TinyOS	Монолитное	BSD	+	Кооперативная	+	HIoT
Ubuntu Core	Монолитное	CC-BY-SA version 3.0 UK licence	+	Вытесняющая	+	HIoT
Raspbian	Монолитное	GNU GPL	+	Вытесняющая	-	HIoT

3.3 Вывод

Для промышленного интернета вещей наиболее универсальными, функциональными и масштабируемыми решениями будут Azure Sphere, Windows 10 IoT и Amazon FreeRTOS, которые не уступают конкурентам по рассматриваемым критериям, а также предоставляют интеграцию с многофункциональными облачными сервисами. Стоит отметить, что сервисы Microsoft и Amazon в момент написания данной работы[4] доступны не во всех странах [70] [71]. По этой причине можно рассмотреть более доступные ОСРВ МАКС и KasperskyOS. Выбор также будет зависеть от того, какой тип ОС больше подходит для решения конкретной задачи: ОС реального времени или ОС разделения времени.

Для бытового применения наиболее предпочтительными можно считать Ubuntu Core и Raspbian, так как они являются дистрибутивами Linux, основанными на Debian, и предоставляют графический пользовательский интерфейс, подобный тем, которые имеют операционные системы для персональных компьютеров.

[4]Ноябрь 2022 года.

Заключение

В рамках научно-исследовательской работы была проведена классификация сетевых операционных систем для устройств интернета вещей.

В результате сравнения были выделены:

- Azure Sphere, Windows 10 IoT и Amazon FreeRTOS как наиболее функциональные и масштабируемые;
- ОСРВ МАКС и KasperskyOS как наиболее доступные с точки зрения использования прикладных служб;
- Ubuntu Core и Raspbian как наиболее адаптированные для бытового применения.

В ходе выполнения данной работы были решены следующие задачи.

1. Проведён анализ предметной области интернета вещей;
2. Рассмотрены существующие операционные системы для устройств интернета вещей;
3. Сформулированы критерии сравнения и оценки рассмотренных операционных систем;
4. Проведён сравнительный анализ существующих решений по выделенным критериям.

Таким образом, все поставленные задачи были выполнены, поставленная цель достигнута.

Список использованных источников

1. Довгаль В. А. Довгаль Д. В. Интернет Вещей: концепция, приложения и задачи // Вестник Адыгейского государственного университета. Серия 4: Естественно-математические и технические науки. — 2018. — С. 129–135.

2. В. Маркеева А. Интернет вещей (IoT): возможности и угрозы для современных организаций // Общество: социология, психология, педагогика. — 2016.

3. Что такое интернет вещей? Определение и описание [Электронный ресурс]. — Режим доступа: https://www.kaspersky.ru/resource-center/definitions/what-is-iot (дата обращения: 09.11.2022).

4. What is IoT? [Электронный ресурс]. — Режим доступа: https://www.oracle.com/internet-of-things/what-is-iot/ (дата обращения: 09.11.2022).

5. Рекомендация МСЭ-Т Y.2060. Обзор интернета вещей. [Электронный ресурс]. — Режим доступа: https://iotas.ru/files/documents/wg/T-REC-Y.2060-201206-I!!PDF-R.pdf (дата обращения: 10.11.2022).

6. Олифер В. Г. Олифер Н. А. Сетевые операционные системы. — Питер, 2009. — ISBN: 9785272001207.

7. С. Таненбаум Э. Операционные системы: разработка и реализация. — Питер, 2006. — ISBN: 5-469-00148-2.

8. Techopedia Explains Nano Kernel [Электронный ресурс]. — Режим доступа: https://www.techopedia.com/definition/27005/nano-kernel (дата обращения: 26.11.2022).

9. Cisco - Networking, Cloud, and Cybersecurity Solutions [Электронный ресурс]. — Режим доступа: https://www.cisco.com/ (дата обращения: 09.11.2022).

10. Lamtzidis, Odysseas. (2019). An IoT Edge-as-a-service (Eaas) Distributed Architecture & Reference Implementation. 10.13140/RG.2.2.32690.76481.

11. Battery draining attacks against edge computing nodes in IoT net¨ works / Smith Ryan, Palin Daniel, Ioulianou Philokypros, Vassilakis Vassil¨ ios, and Shahandashti Siamak // Cyber-Physical Systems. — 2020. — 01. — Vol. 6. — P. 1–21.

12. Masoodhu Banu N. M. Sujatha C. IoT Architecture a Comparative Study // International Journal of Pure and Applied Mathematics. — 2017. — Vol. 117. — P. 45–49.

13. Руководство по выбору операционной системы для пограничного устройства Интернета вещей [Электронный ресурс]. — Режим доступа: https://www.quarta-embedded.ru/about/statiyi/rukovodstvo-po-vyboru-operacionnoj-sistemy-dlya-pogranichnogo-ustrojstva-interneta-veshchej/ (дата обращения: 10.11.2022).

14. What is Microsoft Azure RTOS? [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/en-us/azure/rtos/overview-rtos (дата обращения: 23.11.2022).

15. Обзор ОСРВ Azure ThreadX [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/ru-ru/azure/rtos/threadx/overview-threadx (дата обращения: 23.11.2022).

16. Azure Sphere [Электронный ресурс]. — Режим доступа: https://azure.microsoft.com/en-us/products/azure-sphere/ (дата обращения: 24.11.2022).

17. Linux [Электронный ресурс]. — Режим доступа: https://www.linux.org/ (дата обращения: 24.11.2022).

18. Azure Sphere Terminology [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/en-us/azure-sphere/product-overview/terminology (дата обращения: 24.11.2022).

19. What is Azure Sphere? [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/ru-ru/azure-sphere/product-overview/what-is-azure-sphere (дата обращения: 24.11.2022).

20. The three parts of Azure Sphere [Электронный ресурс]. — Режим доступа: https://arstechnica.com/gadgets/2018/04/microsofts-bid-to-secure-the-internet-of-things-custom-linux-custom-chips-azure/ (дата обращения: 24.11.2022).

21. FreeRTOS [Электронный ресурс]. — Режим доступа: https://aws.amazon.com/ru/freertos/ (дата обращения: 24.11.2022).

22. FreeRTOS features [Электронный ресурс]. — Режим доступа: https://aws.amazon.com/ru/freertos/features/ (дата обращения: 24.11.2022).

23. AWS IoT Core [Электронный ресурс]. — Режим доступа: https://aws.amazon.com/ru/iot-core/ (дата обращения: 24.11.2022).

24. AWS IoT Device Defender [Электронный ресурс]. — Режим доступа: https://aws.amazon.com/ru/iot-device-defender/ (дата обращения: 24.11.2022).

25. AWS IoT Device Management [Электронный ресурс]. — Режим доступа: https://aws.amazon.com/ru/iot-device-management/ (дата обращения: 24.11.2022).

26. Internet of Things. Embedded Operating Systems [Электронный ресурс]. — Режим доступа: http://ichatz.me/uniroma1/iot-2020/unir

oma1-internet_of_things-2020-ichatz-talk3.pdf (дата обращения: 24.11.2022).

27. Zephyr introduction [Электронный ресурс]. — Режим доступа: https://docs.zephyrproject.org/latest/introduction/index.html (дата обращения: 24.11.2022).

28. Zephyr devicetree [Электронный ресурс]. — Режим доступа: https://docs.zephyrproject.org/latest/build/dts/index.html (дата обращения: 24.11.2022).

29. МАКС ОСРВ [Электронный ресурс]. — Режим доступа: https://soware.ru/products/macs-rtos (дата обращения: 24.11.2022).

30. Российская операционная система реального времени для IT-оборудования и Интернета вещей [Электронный ресурс]. — Режим доступа: https://www.astrosoft.ru/products/development/rtos-macs/ (дата обращения: 24.11.2022).

31. Заявление о включении сведений о программном обеспечении в реестр российского программного обеспечения - «Операционная система реального времени для мультиагентных когерентных систем» [Электронный ресурс]. — Режим доступа: https://reestr.digital.gov.ru/request/186244 (дата обращения: 24.11.2022).

32. Milandr [Электронный ресурс]. — Режим доступа: https://www.milandr.com/ (дата обращения: 24.11.2022).

33. STMicroelectronics: Home [Электронный ресурс]. — Режим доступа: https://www.st.com/content/st_com/en.html (дата обращения: 24.11.2022).

34. «АстроСофт» представил новую версию ОСРВ МАКС 1.5 [Электронный ресурс]. — Режим доступа: https://www.astrosoft.ru/about/press_room/news/36002/ (дата обращения: 24.11.2022).

35. Что такое операционные системы реального времени на примере ОСРВ МАКС [Электронный ресурс]. — Режим доступа: https://internetofthings.ru/78-blog/207-chto-takoe-operatsionnye-sistemy-realnogo-vremeni-na-primere-osrv-maks (дата обращения: 24.11.2022).

36. Huawei LiteOS source code [Электронный ресурс]. — Режим доступа: https://github.com/LiteOS/LiteOS (дата обращения: 24.11.2022).

37. Huawei LiteOS Overview [Электронный ресурс]. — Режим доступа: https://www.huawei.com/minisite/liteos/en/about.html (дата обращения: 24.11.2022).

38. Huawei LiteOS [Электронный ресурс]. — Режим доступа: https://www.huawei.com/minisite/liteos/en/index.html (дата обращения: 24.11.2022).

39. An overview of Windows 10 IoT [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/en-us/windows/iot-core/windows-iot (дата обращения: 27.11.2022).

40. Продукты Azure [Электронный ресурс]. — Режим доступа: https://azure.microsoft.com/ru-ru/products/ (дата обращения: 27.11.2022).

41. An overview of Windows 10 IoT Core [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/en-us/windows/iot-core/windows-iot-core (дата обращения: 27.11.2022).

42. Getting Started with Windows IoT Enterprise [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/en-us/windows/iot/iot-enterprise/getting_started (дата обращения: 27.11.2022).

43. Windows 10 Enterprise [Электронный ресурс]. — Режим доступа: https://www.microsoft.com/en-us/evalcenter/evaluate-windows-10-enterprise (дата обращения: 27.11.2022).

44. An overview of Windows Server IoT 2019 [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/en-us/windows/iot-core/windows-server (дата обращения: 27.11.2022).

45. Windows Server 2022 | Microsoft [Электронный ресурс]. — Режим доступа: https://www.microsoft.com/en-us/windows-server (дата обращения: 27.11.2022).

46. Contiki-NG source code [Электронный ресурс]. — Режим доступа: https://github.com/contiki-ng/contiki-ng (дата обращения: 25.11.2022).

47. Payal Malik Malvika Gupta. An Overview Of Iot Operating System: Contiki Os And Its Communication Models // INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH. — 2020. — Т. 9.

48. Mbed OS [Электронный ресурс]. — Режим доступа: https://www.arm.com/products/development-tools/embedded-and-software/mbed-os (дата обращения: 24.11.2022).

49. Homepage - Handbook | Mbed [Электронный ресурс]. — Режим доступа: https://os.mbed.com/handbook/ (дата обращения: 24.11.2022).

50. Google Cloud: Cloud Computing Services [Электронный ресурс]. — Режим доступа: https://cloud.google.com/ (дата обращения: 24.11.2022).

51. Cloud Computing Services - Amazon Web Services (AWS) [Электронный ресурс]. — Режим доступа: https://aws.amazon.com/ (дата обращения: 24.11.2022).

52. Microsoft Azure: Службы облачных вычислений [Электронный ресурс]. — Режим доступа: https://azure.microsoft.com/ (дата обращения: 24.11.2022).

53. Free open source IoT OS and development tools for Arm | Mbed [Электронный ресурс]. — Режим доступа: https://os.mbed.com (дата обращения: 25.11.2022).

54. KasperskyOS [Электронный ресурс]. — Режим доступа: https://os.kaspersky.ru/ (дата обращения: 29.11.2022).

55. KasperskyOS. Техническая информация. [Электронный ресурс]. — Режим доступа: https://os.kaspersky.ru/wp-content/uploads/sites/14/2018/03/KasperskyOS-TechData-ru.pdf (дата обращения: 29.11.2022).

56. Обзор KasperskyOS Community Edition 1.1. Общие сведения. [Электронный ресурс]. — Режим доступа: https://support.kaspersky.com/help/KCE/1.1/ru-RU/overview_general_inf.htm (дата обращения: 29.11.2022).

57. Gray John Shapley. Interprocess Communications in Linux: The Nooks & Crannies. — Pearson, 2003. — ISBN: 0130460427.

58. TinyOS Home Page [Электронный ресурс]. — Режим доступа: http://www.tinyos.net/ (дата обращения: 26.11.2022).

59. TinyOS - Национальная библиотека им. Н. Э. Баумана [Электронный ресурс]. — Режим доступа: https://ru.bmstu.wiki/TinyOS (дата обращения: 26.11.2022).

60. Werner Weber Jan M. Rabaey Emile Aarts. Ambient Intelligence. — Springer Berlin, Heidelberg, 2005. — ISBN: 978-3-540-27139-0. — Режим доступа: https://link.springer.com/book/10.1007/b138670.

61. Ubuntu Core documentation [Электронный ресурс]. — Режим доступа: https://ubuntu.com/core/docs (дата обращения: 24.11.2022).

62. Ubuntu: Enterprise Open Source and Linux [Электронный ресурс]. — Режим доступа: https://ubuntu.com/ (дата обращения: 24.11.2022).

63. Ubuntu Core definition [Электронный ресурс]. — Режим доступа: https://www.techtarget.com/searchitoperations/definition/Ubuntu-Core (дата обращения: 24.11.2022).

64. Welcome to Raspbian [Электронный ресурс]. — Режим доступа: https://www.raspbian.org/ (дата обращения: 24.11.2022).

65. Информация о выпуске Debian 7.0 (wheezy) [Электронный ресурс]. — Режим доступа: https://www.debian.org/releases/wheezy/armhf/release-notes.ru.pdf (дата обращения: 24.11.2022).

66. LXDE [Электронный ресурс]. — Режим доступа: http://www.lxde.org/ (дата обращения: 24.11.2022).

67. Customise your Raspberry Pi desktop [Электронный ресурс]. — Режим доступа: https://projects.raspberrypi.org/en/projects/custom-pi-desktop/3 (дата обращения: 24.11.2022).

68. Пакетный менеджер APT [Электронный ресурс]. — Режим доступа: https://help.ubuntu.ru/wiki/apt (дата обращения: 24.11.2022).

69. Совместимость с POSIX [Электронный ресурс]. — Режим доступа: https://www.swd.ru/print.php3?pid=386 (дата обращения: 26.11.2022).

70. Microsoft suspends new sales in Russia [Электронный ресурс]. — Режим доступа: https://blogs.microsoft.com/on-the-issues/2022/03/04/microsoft-suspends-russia-sales-ukraine-conflict/?icid=mscom_marcom_TS1_Sales_update (дата обращения: 27.11.2022).

71. How Amazon is assisting in Ukraine [Электронный ресурс]. — Режим доступа: https://www.aboutamazon.com/news/community/amazons-assistance-in-ukraine#March4 (дата обращения: 27.11.2022).

Приложение А

Презентация.

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»

Сетевые операционные системы

Студент: Сапожков Андрей Максимович ИУ7-53Б

Научный руководитель: Строганов Юрий Владимирович

Цель и задачи

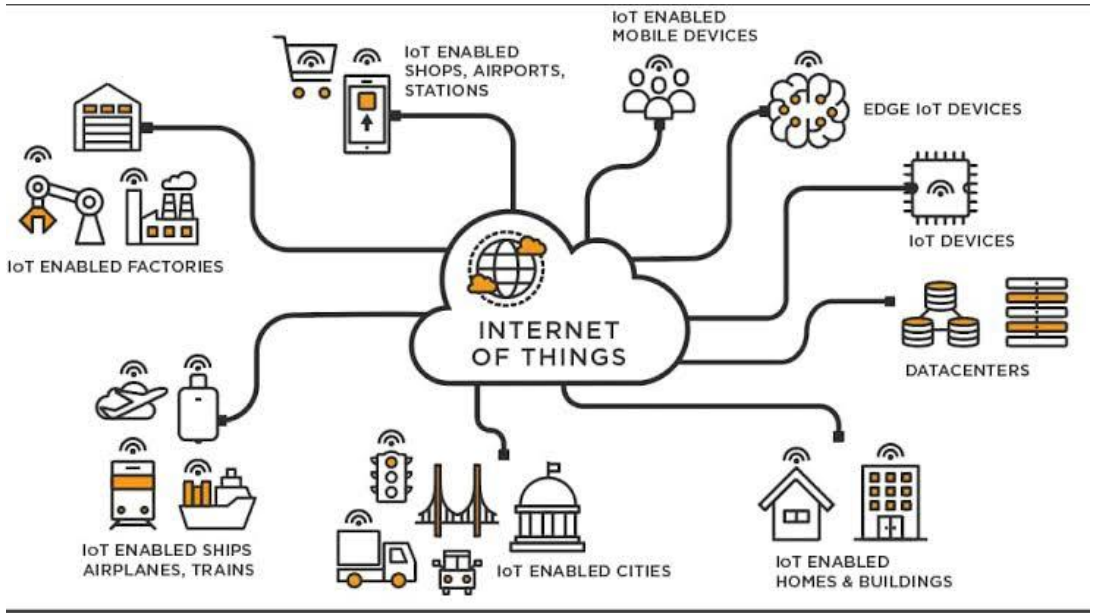
Цель – классификация существующих операционных систем для устройств интернета вещей.

Задачи:

1) проанализировать предметную область интернета вещей;
2) рассмотреть существующие операционные системы для интернета вещей;
3) сформулировать критерии сравнения и оценки рассмотренных операционных систем;
4) сравнить существующие решения по выделенным критериям.

Интернет вещей

Интернет вещей[1] — это концепция, описывающая сеть физических объектов, оснащённых технологиями для подключения и обмена данными с другими устройствами через интернет.
Интернет вещей[2] — это система взаимосвязанных вычислительных устройств, которые могут собирать и передавать данные по беспроводной сети без участия человека.



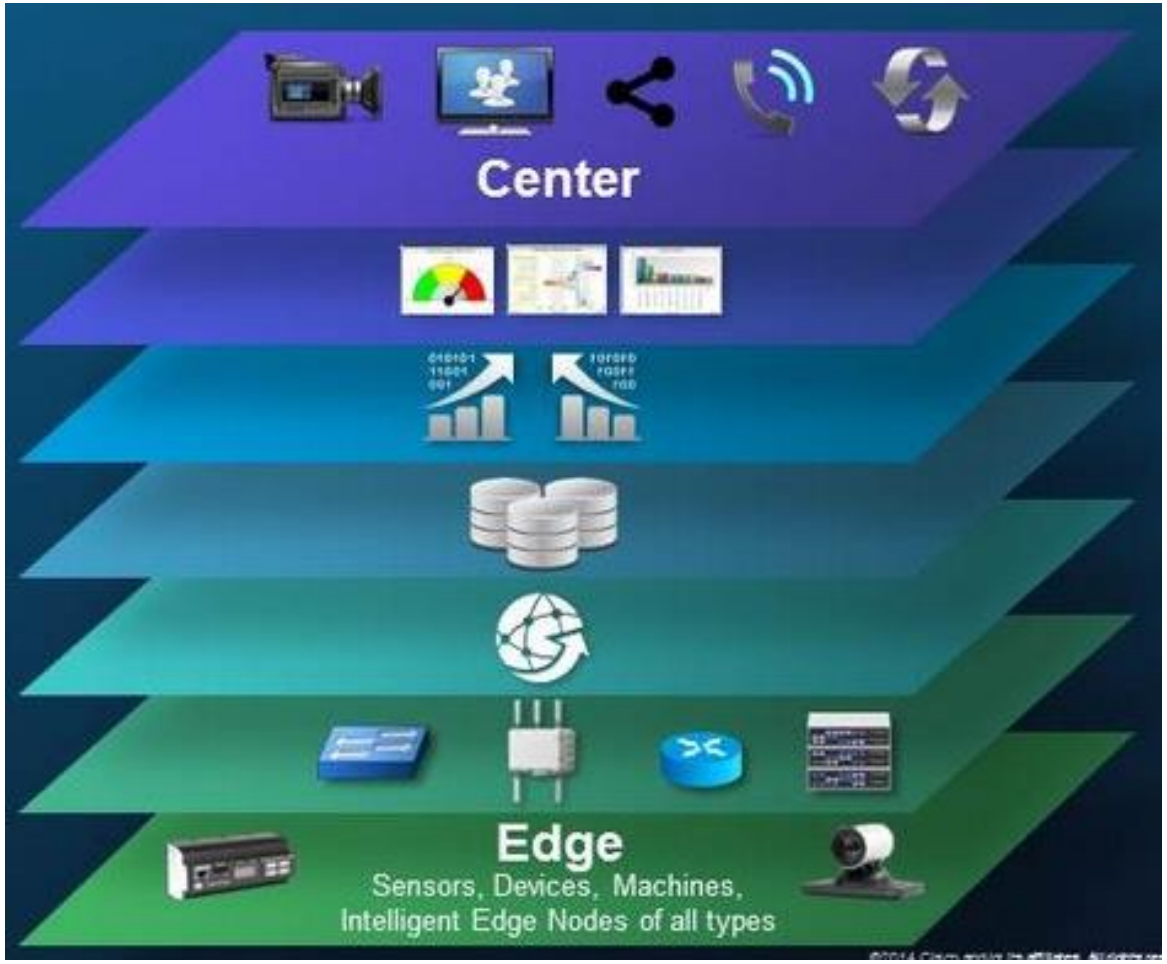
1. https://www.kaspersky.ru/resource-center/definitions/what-is-iot
2. https://www.oracle.com/internet-of-things/what-is-iot/

Приложения интернета вещей

1. Интернет вещей в быту;
2. Промышленный интернет вещей (IIoT):
   - Умные города;
   - Производства;
   - Транспорт и логистика;
   - Розничная торговля;
   - Государственный сектор;
   - Здравоохранение;
   - Общая безопасность во всех отраслях.

Архитектура интернета вещей

7. Взаимодействие и процессы.
6. Приложения.
5. Абстракция данных.
4. Накопление данных.
3. Граничные вычисления.
2. Соединение.
1. Физические устройства и контроллеры.



ОС реального времени

ОС	Тип ядра	Тип лицензии	POSIX	Многозадачность	Кроссплатформенность	Применение
Azure RTOS	Наноядро	Microsoft Software License	Отсутствует	Вытесняющая	+	IIoT
Azure Sphere	Монолитное	GPL-2.0 license	Частичная	Вытесняющая	+	IIoT
Amazon FreeRTOS	Микроядро	MIT	Полная	Вытесняющая	+	IIoT
Zephyr	Наноядро	Apache Licence 2.0	Частичная	Вытесняющая и кооперативная	+	IIoT
ОСРВ МАКС	Монолитное	BSD-3-Clause license	Частичная	Вытесняющая и кооперативная	+	IIoT
Huawei LiteOS	Микроядро	BSD-3-Clause license	Полная	Вытесняющая	-	IIoT

ОС разделения времени

ОС	Тип ядра	Тип лицензии	POSIX	Многозадачность	Кроссплатформенность	Применение
Windows 10 IoT	Гибридное	Microsoft Software License	Отсутствует	Вытесняющая	+	IIoT
Contiki-NG	Монолитное	BSD-3-Clause license	Частичная	Кооперативная	+	IIoT
Mbed OS	Монолитное	Apache Licence 2.0	Полная	Кооперативная	-	IIoT
Kaspersky OS	Микроядро	Проприетарная	Частичная	Не декларировано	+	IIoT
TinyOS	Монолитное	BSD	Полная	Кооперативная	+	HIoT
Ubuntu Core	Монолитное	CC-BY-SA version 3.0 UK licence	Полная	Вытесняющая	+	HIoT
Raspbian	Монолитное	GNU GPL	Полная	Вытесняющая	-	HIoT

Рекомендации по применению

В результате сравнения были выделены:

- наиболее функциональные и масштабируемые:
  Azure Sphere, **Windows 10 IoT** и **Amazon FreeRTOS**;
- наиболее доступные с точки зрения использования прикладных служб:
  ОСРВ МАКС и **KasperskyOS**;
- наиболее адаптированные для бытового применения:
  Ubuntu Core и **Raspbian**.

Заключение

В рамках научно-исследовательской работы была проведена классификация существующих операционных систем для интернета вещей. Для достижения этой цели были решены следующие задачи:

1) проанализирована предметная область интернета вещей;
2) рассмотрены существующие операционные системы для интернета вещей;
3) сформулированы критерии сравнения и оценки рассмотренных операционных систем;
4) проведено сравнение существующих решений по выделенным критериям.