**Интернет вещей в быту;**

В быту интернет вещей применяется при проектировании умных домов. Системы интернета вещей способны автоматизировать бытовые процессы и исключить 9 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.

**Промышленный интернет вещей (IIoT):**

Промышленный Интернет вещей (Industrial IoT, IIoT) 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.

**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.

**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.

**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.

**Розничная торговля;**

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

**Государственный сектор;**

Преимущества IoT в государственном секторе и других средах, связанных с услугами, достаточно обширны. 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.

**Здравоохранение;**

Интернет вещей является важным аспектом телемедицины [3] (для обозначения 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.

**Общая безопасность во всех отраслях.**

Помимо отслеживания физических показателей, IoT можно использовать для повышения безопасности труда. 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.

Определение архитектуры интернета вещей является предметом серьезных дискуссий. Одной из наиболее подробных архитектур является семиуровневая модель, предложенная компанией Cisco. На слайде описаны её уровни.

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 8 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.

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

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.

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
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. Степень POSIX-соответствия операционных систем характеризуют стандартизованные интерфейсы операционных систем и приложений и, следовательно, степень их мобильности на уровне исходного языка.
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.

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.

1. **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
2. **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.

**\*Выбирать ключевые фишки и комментировать таблицу\***

**Операционная система реального времени** — это система, предназначенная для управления физическими объектами (процессами), которая способна обеспечить предсказуемое время реакции в ответ на изменение состояния управляемого объекта (процесса).

**Azure RTOS**

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

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

Поддерживает среды с многоядерными процессорами посредством асимметричной многопроцессорной обработки или симметричной многопроцессорной обработки.

**Azure Sphere**

Microsoft Azure Sphere — 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 13 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, которую Microsoft публично выпустила, и вторая Unix-подобная операционная система, разработанная компанией для публичного пользования

Служба безопасности Azure Sphere, 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.

**Amazon FreeRTOS**

FreeRTOS — 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.

AWS IoT Core

**Zephyr**

Zephyr OS - это масштабируемая операционная система реального времени (RTOS) для встраиваемых устройств с открытым исходным кодом. Кросс-платформенная архитектурна разработку ПО как для микроконтроллеров, так и для систем на кристалле (System on Chip, SoC).

Несколько алгоритмов планирования.

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

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.

**ОСРВ МАКС**

ОСРВ МАКС — 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: в проекте не используются фрагменты других ОСРВ, что позволило воплотить самые современные архитектурные решения. ОСРВ МАКС входит в реестр российского ПО.

Хорошей предпосылкой для уникальности ОСРВ МАКС 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.

**Huawei LiteOS**

Huawei LiteOS — это операционная система реального времени с открытым исходным кодом, разработанная для сферы IoT и являющаяся частью операционной системы Huawei IoT operating system kernel.

Акцент на низкое энергопотребление

Спящий режим

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.

**\*Выбирать ключевые фишки и комментировать таблицу\***

**Система разделения времени** — 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.

**Windows 10 IoT**

Windows 10 IoT — 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.

1. **Windows 10 IoT Core** — 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** — это полная версия Windows Enterprise, 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** — это полная версия Windows Server [44], которая обеспечивает корпоративную управляемость и безопасность для решений IoT. Windows Server IoT использует все преимущества экосистемы Windows. Данная операционная система является двоично совместимой 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 позволяет создавать решения фиксированного назначения с определенными допущениями и ограничениями в лицензионном соглашении.

**Contiki-NG**

Contiki-NG — 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 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), 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.

**Mbed OS**

Arm Mbed OS — 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.

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

— Google Cloud Platform;

— Amazon Web Services (AWS);

— Microsoft Azure.

**KasperskyOS**

KasperskyOS — 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.

Всё построено на изолированных компонентах. Организация их взаимодействия – первостепенная задача KasperskyOS.

**TinyOS**

TinyOS — 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. 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, кодифицированную языком NesC, являющегося диалектом языка C. TinyOS не является 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.

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.

**Ubuntu Core**

Ubuntu Core — это версия операционной системы Ubuntu, 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 — 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.

Debian, GUI

**Raspbian**

Raspbian — это операционная система с открытым исходным кодом, основанная на Debian и оптимизированная для аппаратной платформы Raspberry Pi. Raspbian — это неофициальный перенос Debian Wheezy armhf 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.

Debian, GUI

**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.