Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина
Институт радиоэлектроники и информационных технологий — РТФ

Н. В. ПАПУЛОВСКАЯ

ОСНОВЫ ИНТЕРНЕТА ВЕЩЕЙ

Учебно-методическое пособие

Министерство науки и высшего образования
Российской Федерации

Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина

Н. В. Папуловская

ОСНОВЫ ИНТЕРНЕТА ВЕЩЕЙ

Учебно-методическое пособие

Рекомендовано методическим советом
Уральского федерального университета
для студентов инженерно-технических
специальностей ИРИТ–РтФ

Екатеринбург
Издательство Уральского университета
2022

УДК 004.89(075.8)
ББК 32.971.353я73
П17

Рецензенты:

д-р физ.-мат. наук, доц., завкафедрой теории функций факультета математики и компьютерных наук Кубанского государственного университета *М. В. Голуб*;

руководитель направлений учебной и промышленной робототехники ООО «МГБот» *И. В. Комаров*

Научный редактор — канд. техн. наук, доц. *А. В. Присяжный*

На обложке использовано изображение с сайта https://techcrunch.com/wp-content/uploads/2017/06/gettyimages-637700890.jpg

Папуловская, Наталья Владимировна.

П17 Основы интернета вещей : учебно-методическое пособие / Н. В. Папуловская ; М-во науки и высшего образования РФ. — Екатеринбург : Изд-во Урал. ун-та, 2022. — 104 с.

ISBN 978-5-7996-3537-4

Учебно-методическое пособие посвящено понятиям и технологиям, составляющим основу проектирования систем интернета вещей. Читатели получат представление о том, как устроены системы интернета вещей, каковы тренды в этой сфере в России и в мире. Кроме того, рассматриваются вопросы архитектуры систем интернета вещей, подключения периферийных устройств, программирования микроконтроллеров, технологий беспроводной связи, программных средств и приложений. Описана авторская методика изучения дисциплины, игровые методы обучения, приводятся практические задания, дидактические материалы.

Пособие предназначено для студентов УрФУ, обучающихся по направлениям подготовки 11.03.02 — Инфокоммуникационные технологии и системы связи; 09.03.01 — Информатика и вычислительная техника; 09.03.03 — Прикладная информатика; 09.03.04 — Программная инженерия; 27.03.04 — Управление в технических системах.

Библиогр.: 12 назв. Рис. 25. Прил. 2.

УДК 004.89(075.8)
ББК 32.971.353я73

ISBN 978-5-7996-3537-4

Оглавление

Предисловие

Интернет вещей — одна из актуальных, развивающихся областей инженерии, это прежде всего информационно-управляющая система, работающая на основе данных от физических предметов, снабженных датчиками. Устройства интернета вещей, с одной стороны, имеют интерфейс с коммуникационной сетью, а с другой — интерфейс, обеспечивающий физическое взаимодействие датчиков и исполнительных механизмов. Чтобы понять, как функционируют системы интернета вещей, необходимо комплексно изучать построение микроконтроллерных систем, схемотехнику, беспроводные сети, сервисы сбора и обработки данных.

Пособие содержит четыре раздела. Каждый из них включает теоретический учебный материал, методический и дидактический материал для проведения игровых занятий, тренингов и лабораторных работ и контрольные задания по теме раздела. Методические и дидактические материалы разработаны автором курса и предназначены для работы на аудиторных занятиях со студентами в рамках изучения дисциплины «Основы интернета вещей». Кроме того, в конце каждой главы для углубленного изучения темы предложены дополнительные образовательные материалы (цикл вебинаров академии Samsung Innovation Campus: открытые лекции российских IT-специалистов различных компаний и ведущих вузов по каждой теме).

Учебно-методическое пособие было разработано с использованием образовательных материалов социального проекта компании Samsung.

Автор благодарит своего мужа Андрея Кобелева за помощь в подготовке пособия.

Список сокращений

IoT — Internet of Things
LoRa — Long Range Wide Area Network
M2M — Machine-to-Machine
MQTT — Message Queuing Telemetry Transport
NB-IoT — Narrow Band Internet of Things
TCP/IP — Transmission Control Protocol/Internet Protocol
UART — Universal Asynchronous Receiver-Transmitter
UML — Unified Modeling Language
USB — Universal Serial Bus
АСУ ТП — автоматизированные системы управления технологическими процессами
АЦП — аналого-цифровое преобразование
БД — база данных
Индустрия 4.0 — прогнозируемая четвертая промышленная революция с массовым внедрением киберфизических систем в производство и обслуживание человеческих потребностей
МК — микроконтроллер
ОС — операционная система
ОС РВ — операционная система реального времени
ПК — персональный компьютер
ПО — программное обеспечение
САПР — система автоматизированного проектирования
ЦОД — центр обработки данных
ШИМ — широтно-импульсная модуляция

Введение

Системы интернета вещей стали появляться в конце XX в. Устройство, которое сегодня мы можем назвать первой системой интернета вещей, разработали студенты в 1982 г.: они установили камеру напротив автомата с кока-колой и подключили ее к локальной сети, чтобы проверять, не закончился ли в автомате прохладительный напиток. В 1990 г. выпускник Массачусетского технологического института Джон Ромки (John Romkey) подключил к интернету свой тостер, ставший первым официально зарегистрированным объектом из мира интернета вещей. Потом были умный пылесос, умная кофеварка и другие устройства, управление которыми осуществляется в автоматическом или автоматизированном режиме с использованием интернета для передачи данных.

Когда-то словосочетание «интернет вещей» казалось нам абсурдным и воспринималось как продажа вещей через интернет. Этот термин ввел британский инженер Кевин Эштон в 1999 г. как описание системы, в которой интернет связан с физическим миром через датчики. В настоящее время мы подразумеваем под этим термином целый комплекс цифровых технологий.

Интернет вещей (Internet of Things – IoT) — это концепция сети предметов («вещей»), оснащенных технологиями для взаимодействия друг с другом или с внешней средой.

Предпосылками для возникновения систем интернета вещей стали: уменьшение размеров вычислительных устройств; стандартизация протоколов; снижение стоимости электроники, связи и вычислительных мощностей; появление беспроводных технологий связи с низким потреблением энергии.

Со временем системы интернета вещей стали находить применение в современном мегаполисе. В крупных городах устанавливают умные остановочные комплексы, которые знают, когда подойдет тот или иной транспорт. По требованию федерального закона № 522 с 1 января 2021 г. застройщики обязаны закладывать в новостройки только умные приборы учета, т. е. все счетчики расхода воды и энергоснабжения в домах и учреждениях должны уметь самостоятельно передавать показания расхода энергоресурсов. В скором времени нас ждут умные перекрестки и умные транспортные системы, умные бытовые приборы, умные растения и животные и даже умные объекты искусства. Все эти автоматизированные системы интернета вещей должны сделать нашу жизнь лучше, проще и комфортнее.

Интеллектуальные транспортные системы — это оборудование, которым оснащаются городские и межмуниципальные дороги (светофоры, камеры наблюдения, информационные табло, умные остановки, метеостанции), и программное обеспечение, которое объединяет это оборудование в единую систему и позволяет ею управлять.

Интернет вещей (IoT) — это не просто система, сегодня так называют концепцию, объединяющую множество технологий для разработки систем автоматизации на основе данных, взаимодействующих с использованием сетевых решений.

Современный индустриальный мир, в свою очередь, погружается в промышленный интернет вещей (IIoT) с возможностью удаленного контроля ресурсов предприятия и управления ими в автоматизированном режиме. С помощью систем интернета вещей можно получать информацию о доступности оборудования, его техническом состоянии, загрузке, технологических нарушениях, графике технического обслуживания и т. п. Промышленный интернет вещей позволяет оперативно, в режиме реального времени получить информацию по всему оборудованию на предприятии, за секунды рассчитать его коэффициент полезного использования, а с применением предиктивной аналитики и нейронных сетей — спрогнозировать график планово-предупредительных ремонтов и загрузки.

Применение интернета вещей в промышленности создает новые возможности для развития производства и решает ряд важнейших задач: повышение производительности оборудования; снижение мате-

риальных и энергетических затрат; повышение качества, оптимизация и улучшение условий труда сотрудников компании; рост рентабельности производства и конкурентоспособности на мировом рынке.

Интернет вещей становится неотъемлемой частью четвертой промышленной революции (Индустрия 4.0), он определен как одно из ключевых научно-технических направлений национальной программы «Цифровая экономика Российской Федерации».

Глава 1. Основы работы с микроконтроллерными системами сбора данных

1.1. Особенности построения систем интернета вещей

В структуре любой системы интернета вещей можно условно выделить четыре части: микроконтроллерная система сбора данных, система сетевого взаимодействия, система хранения данных, система управления и мониторинга данных (рис. 1).

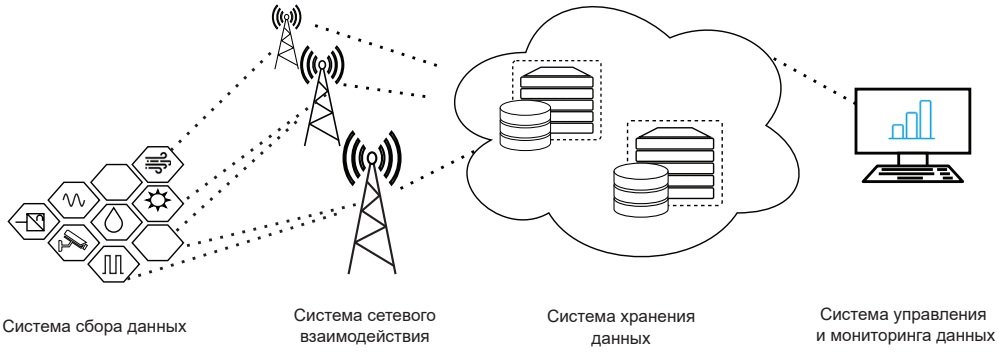


Рис. 1. Структура системы интернета вещей

Каждая из этих систем достойна отдельного курса и своего учебного пособия. В настоящем пособии мы только поверхностно рассмотрим особенностей разработки и функционирования частей системы интернета вещей, глобальной инфраструктуры для информационного общества, которая обеспечивает возможность предоставления более сложных услуг путем соединения друг с другом вещей (физических

и виртуальных) на основе существующих и развивающихся функционально совместимых информационно-коммуникационных технологий[1].

Рис. 2. Система сбора данных

В систему сбора данных (рис. 2) входят следующие устройства: сенсоры (датчики), контроллер или микрокомпьютер, актуатор и модуль передачи данных.

Датчики реагируют на изменения параметров окружающей среды (температура, скорость ветра, электропроводимость, уровень напряжения и т. п.), они действуют на физическом уровне и являются аналоговыми или аналого-цифровыми устройствами. Существует большое количество самых разных датчиков: ультразвуковой сенсор, инфракрасный дальномер, датчик влажности и температуры, датчик освещенности, гироскоп, акселерометр, датчик пламени и датчик дождя. В настоящее время возможности электронных сенсоров не только достигли уровня человеческого восприятия, но и во многом превзошли его. Так, например, человек не способен без дополнительного

[1] Обзор интернета вещей / Сектор стандартизации электросвязи МСЭ-Т. Женева, 2014. 22 с. (Серия Y: Глобальная информационная инфраструктура, аспекты протокола Интернет и сети последующих поколений).

оборудования определить уровень радиации, атмосферного давления или наличие в воздухе опасных примесей. Кроме того, в отличие возможностей человека электронное устройство может функционировать непрерывно и в труднодоступных местах.

Контроллер (микроконтроллер) или микрокомпьютер могут зафиксировать изменение состояния датчика и по заданному алгоритму запустить механизмы (актуаторы).

Актуатор (actuator) — исполнительное устройство или его активный элемент, преобразующий один вид энергии (электрическую, магнитную, тепловую, химическую) в другой (чаще всего — в механическую), что приводит к выполнению определенного действия, заданного управляющим сигналом.

К актуаторам можно отнести: шаговый двигатель, пневматический насос, вибромотор, любое электромеханическое устройство. Чтобы актуатор начал работать, его необходимо включить (запустить) — это может выполнить реле (электрическое устройство, предназначенное для замыкания и размыкания различных участков электрических цепей) по команде с микроконтроллера.

Совместимость контроллера с другими компонентами информационной системы определяется протоколами передачи данных и не накладывает ограничений на производителя и архитектуру. Система сетевого взаимодействия может быть построена на разных каналах и протоколах передачи данных. Кроме протоколов проводного интерфейса (RS-232, I2C, SPI, Ethernet, RS-485, USB), большое развитие получили беспроводные стандарты: Bluetooth, Wi-Fi, GSM, LTE, LoRa. Для систем интернета вещей особенно актуальны беспроводные сети уровня LPWAN (Low-Power Wide-Area Network) и мобильная связь.

Беспроводные сети — это сети, в которых для передачи сигналов через различные участки сети вместо проводов используются электромагнитные волны.

Система хранения данных предполагает разработку базы данных или использование облачных сервисов. Умные устройства генерируют данные и передают их по каналам связи в ЦОД, а затем они попадают в приложение пользователя. После предварительной обработки

данные часто отправляются в облако для более долгосрочного хранения, резервного копирования или глубокого анализа.

Система управления и мониторинга — это комплекс приложений, позволяющих работать с данными, иногда в режиме реального времени. Система должна уметь подключаться к умному устройству через каналы связи, задавать параметры работы, собирать информацию из хранилища данных, обрабатывать и визуализировать ее. Следовательно, речь идет о базе данных и приложений для пользователей системы.

1.2. Архитектура микроконтроллерных систем

Многие системы IoT реализуются в архитектуре «клиент — сервер». Клиент-серверное взаимодействие — это модель обмена данными между клиентским и серверным приложением, которая позволяет разделить функционал и вычислительную нагрузку на систему. В этой модели клиент выступает в роли заказчика услуг, а сервер — в роли поставщика. Модель взаимодействия между клиентом и сервером изображена на рис. 3.

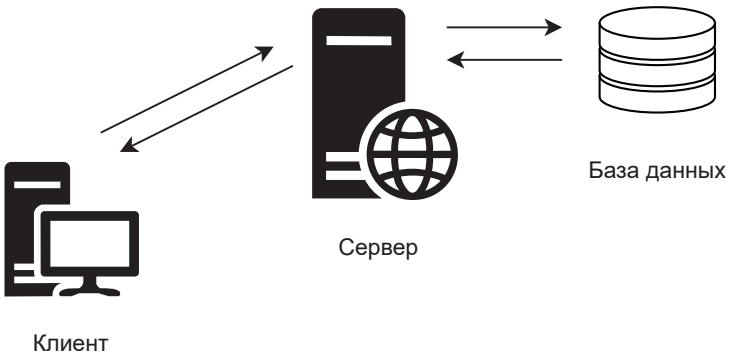


Рис. 3. Клиент-серверное взаимодействие

В основном взаимодействие между клиентским и серверным приложениями происходит с помощью сетевых протоколов: IP, TCP, HTTP, WebSocket и т. д. Инициатором этого взаимодействия всегда выступает клиент. Клиент запрашивает информацию у сервера. Сервер

обрабатывает запрос клиента, получает необходимые данные из базы и возвращает результат клиенту.

Сетевой протокол — согласованный и утвержденный стандарт, содержащий описание правил приема и передачи данных между двумя и более включенными в сеть устройствами или приложениями, обеспечивающими коммуникации на программном уровне.

Конечным элементом системы интернета вещей является микроконтроллер (МК), к которому подключены датчики (Д) и исполнительные устройства (ИУ). Через проводной или беспроводной интерфейс микроконтроллер взаимодействует с сервером, как правило, по протоколу HTTP (рис. 4). В клиент-серверном варианте на клиенте требуется реализовать все сетевые уровни и алгоритмы обработки сбоев, что неминуемо усложняет программное обеспечение микроконтроллера.

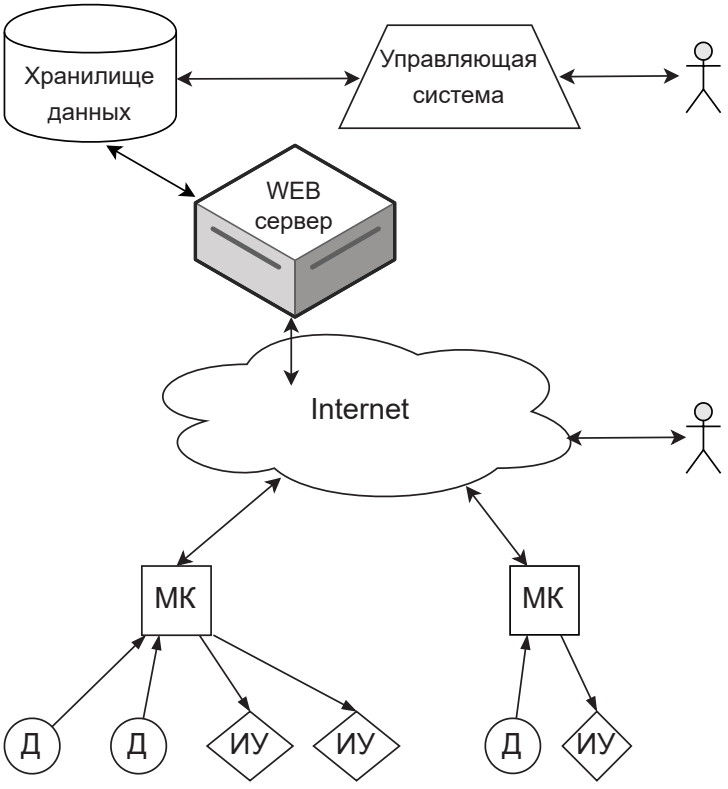


Рис. 4. Система IoT в архитектуре «клиент — сервер»

Системы, реализуемые в архитектуре «клиент — сервер» имеют следующие недостатки:

— клиент контролирует доставку данных, так как отсутствует асинхронный интерфейс;
— необходимо буферизировать данные на клиенте в случае отсутствия связи или ошибок сервера;
— маршрутизация — статическая;
— совместимость версий протокола обмена возлагается на клиента, что увеличивает затраты при модернизации;
— есть ограничения по выбору коммуникационных технологий;
— существуют сложности в реализации программного обеспечения на стороне клиента.

Предлагается пересмотреть подход к построению систем IoT таким образом, чтобы можно было реализовать произвольное решение на модульном принципе, позволяющем обеспечивать совместимость, отказоустойчивость и масштабируемость. При использовании сервис-ориентированной архитектуры (SOA) ключевым технологическим элементом является сервисная шина предприятия ESB. Структурная схема системы IoT, построенной в архитектуре ESB, представлена на рис. 5.

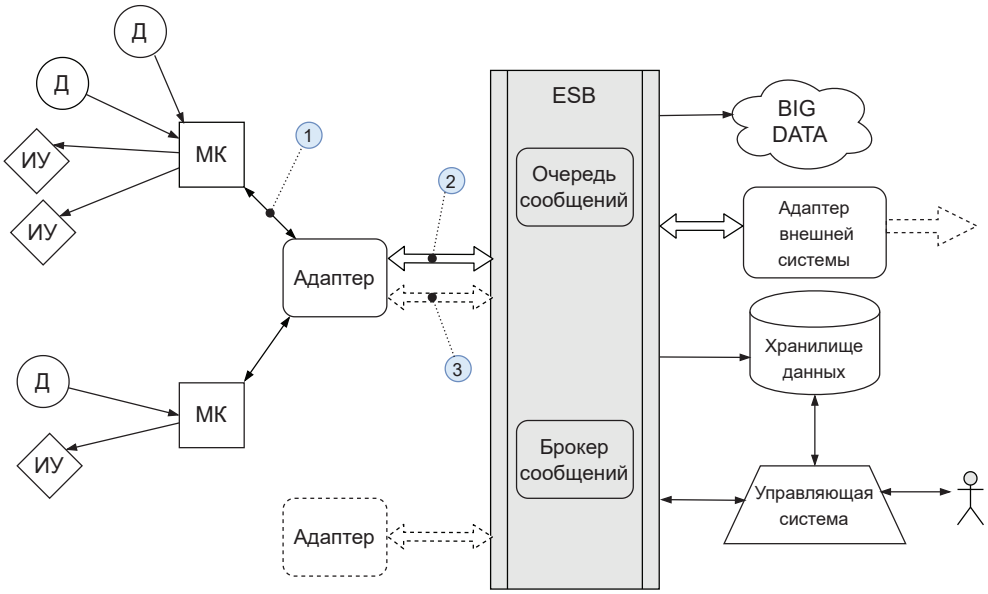


Рис. 5. Система IoT в сервис-ориентированной архитектуре

Конечным элементом системы является микроконтроллер (МК) с набором датчиков (Д) и исполнительных устройств (ИУ). Программное обеспечение микроконтроллера реализует:

— чтение показаний датчиков;
— выполнение команд управления;
— обмен данными с адаптером в синхронном режиме.

Синхронный способ передачи данных — способ передачи цифровых данных по последовательному интерфейсу, при котором приемнику и передатчику известно время передачи данных, то есть передатчик и приемник работают синхронно, в такт.

Адаптер является промежуточным звеном, которое осуществляет взаимодействие с микроконтроллерами по протоколу низкого уровня, исходя из географической удаленности и технических возможностей связи.

Адаптер может быть развернут на более совершенном микроконтроллере, микрокомпьютере с полноценной операционной системой или стационарном ПК. Обновление программного обеспечения адаптера можно осуществлять без физического доступа к устройству. В задачи адаптера входит буферизация сообщений, трансформация в формат шины данных и отправка на сервер очереди сообщений. Адаптер имеет возможность переключения на альтернативный канал передачи данных. В его функции входит мониторинг состояния подключенных к нему узлов и сигнализация о состоянии системы. Программное обеспечение адаптера настраивается в зависимости от функционального назначения вверенной ему подсистемы и может быть перенастроено без внесения изменений в другие части системы.

Сервис-ориентированная архитектура избавляет конечные узлы от задач коммуникационного взаимодействия, так как эта функция возложена на адаптер. На рис. 5 цифрой (1) обозначен канал передачи данных низкоуровневого протокола, (2) — основной канал связи с системой шифрования данных, (3) — резервный канал связи.

Шина данных обеспечивает хранение, диспетчеризацию и трансформацию сообщений. Таким образом обеспечивается интеграция с различными системами, в том числе внешними. Без модификации компонентов системы IoT возможна динамическая настройка маршрутов и подключение дополнительных узлов.

Существуют готовые открытые программные средства и библиотеки, реализующие функционал шины данных. На промышленном уровне целесообразно использовать коммерческие решения, такие как IBM WebSphere ESB, Oracle Enterprise Service Bus, SAP Process Integration и др. Для гражданского и бытового применения систем IoT вполне достаточно возможностей таких продуктов, как Apache ActiveMQ, RabbitMQ, Eclipse Mosquitto и т. п.

На рис. 6 представлена схема взаимодействия компонентов IoT-системы при использовании сети LoRA в сервис-ориентированной архитектуре. В такую структуру можно бесконечно добавлять новые устройства. Обмен данными с удаленными устройствами возлагается на базовую станцию, к которой подключен адаптер, взаимодействующий с остальными компонентами системы IoT.

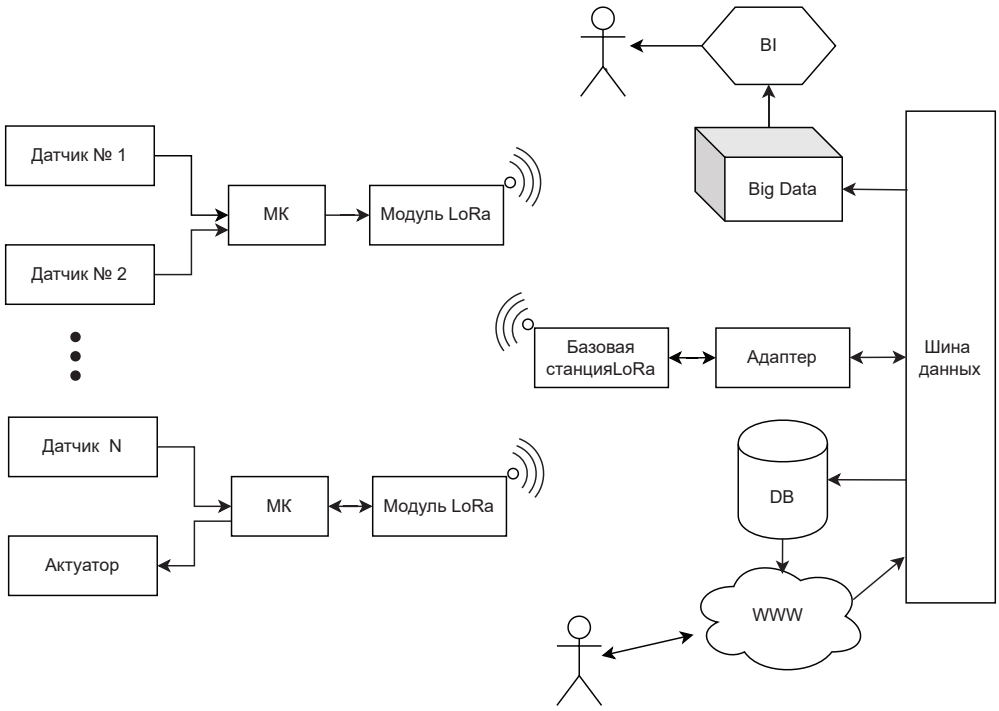


Рис. 6. Схема взаимодействия компонентов системы в сервис-ориентированной архитектуре

Далее рассмотрим ту часть системы, которая непосредственно собирает данные и заставляет актуатор работать: микроконтроллерную систему сбора данных. Сенсоры и актуаторы подключены к микро-

контроллеру, который функционирует в соответствии с загруженной в него программой.

Для выполнения функционала системы в первую очередь необходимо разработать и собрать схему устройства (рис. 7), составить алгоритм работы устройства и запрограммировать микроконтроллер, который располагается на микроконтроллерной плате.

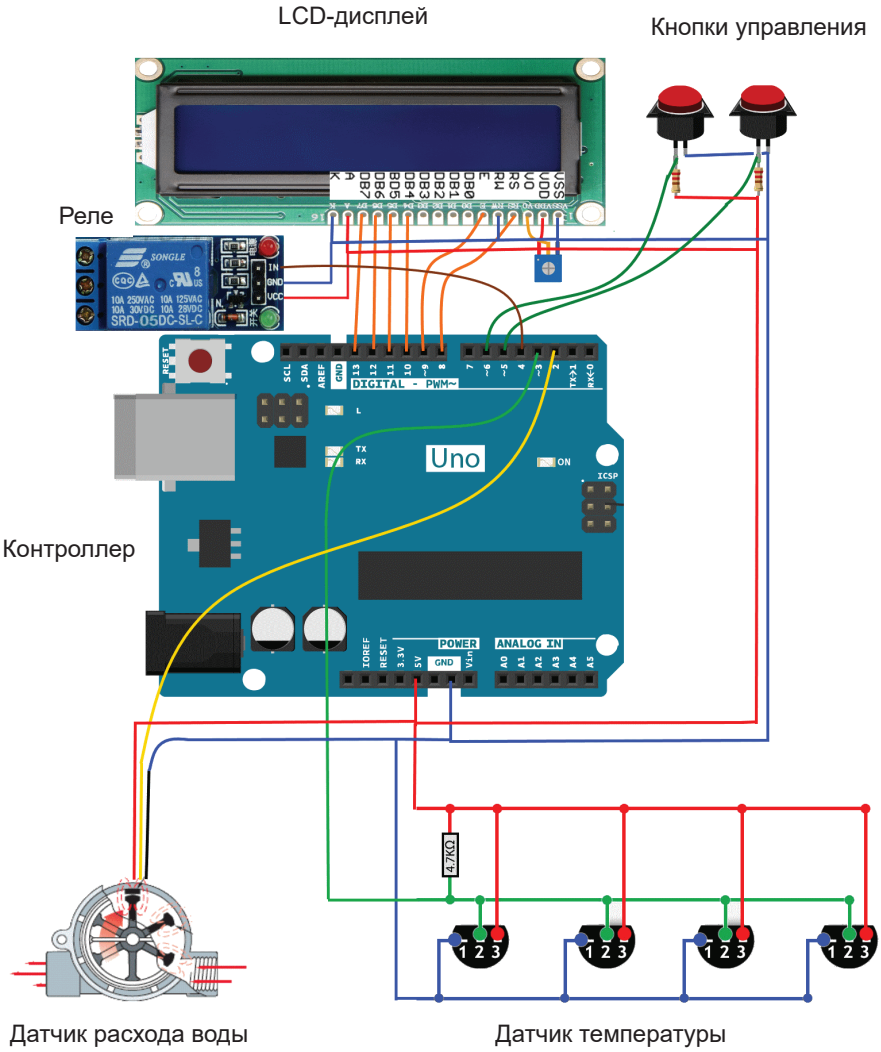


Рис. 7. Схема устройства, выполненная в сервисе Tinkercad

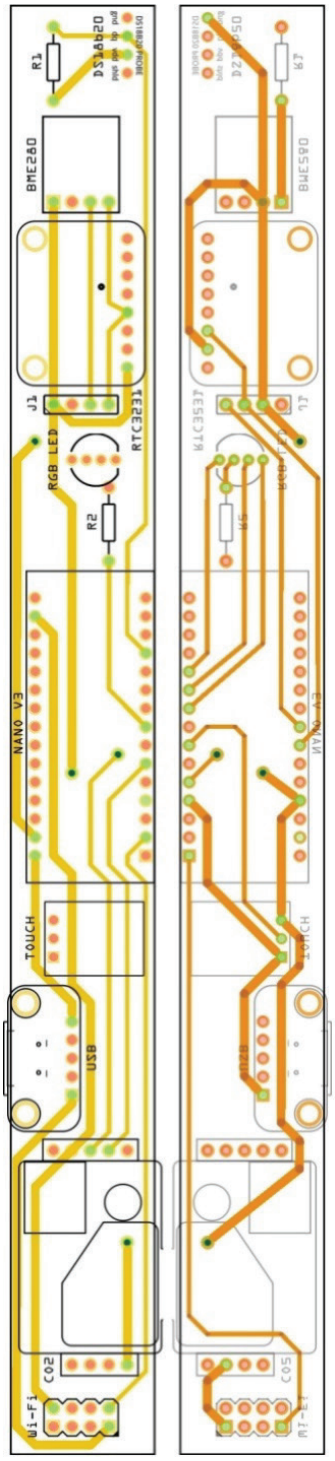


Рис. 8. Две стороны печатной платы IoT-устройства

Схема устройства сначала собирается на макетной плате, а затем переводится на печатную плату с помощью специализированного САПР.

Печатная плата (printing circuit board, PCB) — изделие, предназначенное для размещения и электрического соединения между собой электронных компонентов и функциональных узлов.

На рис. 8 изображена печатная плата, разработанная в САПР Fritzing.

1.3. Программирование микроконтроллеров

Микроконтроллер (МК) — это микросхема, предназначенная для программного управления электронными устройствами и осуществления взаимодействия между ними в соответствии с заложенной программой. В отличие от микропроцессоров, используемых в персональных компьютерах, в микроконтроллеры встроены дополнительные устройства: порты ввода-вывода, оперативная и постоянная память, АЦП, ШИМ, интерфейсы передачи данных и пр. (рис. 9).

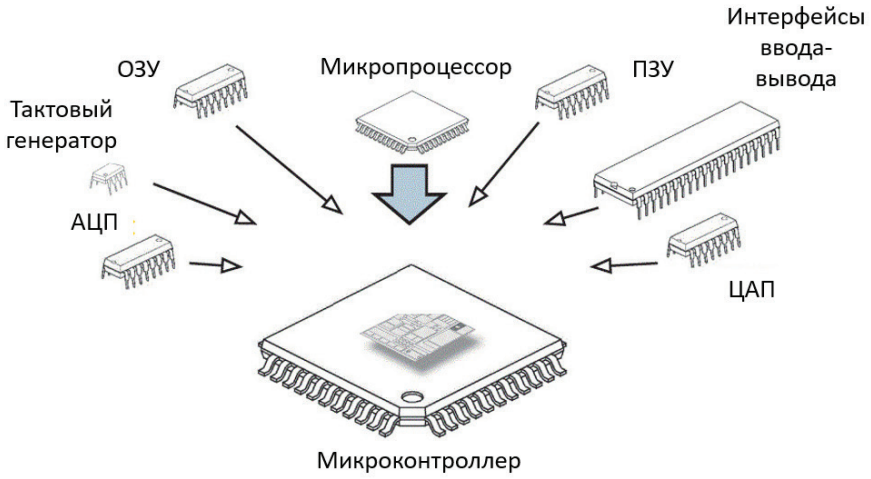


Рис. 9. Состав микроконтроллера

Микроконтроллер состоит из транзисторов и транзисторных цепочек, его сердцем является ядро. Многие компании выпускают МК

с лицензируемой архитектурой ARM (производитель ARM Limited). Микроконтроллеры могут в один момент времени выполнять только одну задачу. Между собой они отличаются по следующим характеристикам: количеству постоянной и оперативной памяти, напряжению питания, количеству интерфейсов ввода-вывода, по энергопотреблению, разрядности.

Микроконтроллеры являются самодостаточными устройствами с микропроцессором и памятью, у них есть практически все необходимое для простых вычислений, не требующих много памяти (оперативная память составляет всего несколько килобайт). Микроконтроллеры по сравнению с микрокомпьютерами имеют невысокую производительность, потребляют всего несколько миллиампер, поэтому могут долго работать от автономного источника питания, что для систем интернета вещей весьма актуально. Программы для них разрабатываются на *C*-подобных языках.

Микроконтроллеры STM32 — наиболее популярное и широко используемое семейство микроконтроллеров, применяемых в реальных системах, а не только в обучении. Индекс 32 обозначает разрядность ядра, 32-битное ядро позволяет иметь $2^{32} = 4$ Гб адресуемого пространства. Адресное пространство не тождественно размеру шины, в реальности МК имеет десятки килобайт ОЗУ.

Подробно о программировании МК можно прочитать в книге «Си для встраиваемых систем»[2]. Программирование микроконтроллера обычно выполняется с использованием некоторой платформы.

Компьютерная платформа — среда, в которой должен выполняться фрагмент программного обеспечения или объектный модуль с учетом накладываемых этой средой ограничений и предоставляемых возможностей. Термин **платформа** может применяться к разным уровням абстракции, включая определенную аппаратную архитектуру, операционную систему или библиотеку времени выполнения.

Возможно, многие уже знакомы с платформой Ардуино (Arduino). В ней можно быстро и легко решить базовые задачи управления различными устройствами, начиная со светодиода и заканчивая сложным

[2] chrns [псевд.]. Си для встраиваемых систем. URL: https://robotclass.ru/articles/c_for_embedded_systems/ (дата обращения: 30.09.2021).

объектом. Однако у Arduino есть ряд существенных недостатков из-за абстрагирования и упрощения технологии. Например, алгоритм программы часто пишется в бесконечном цикле и снабжается кодом задержек, тогда как в профессиональной разработке это делается через прерывание и многопоточность. Прерывание необходимо, чтобы переключиться с основной программы на некоторое событие, на которое должна отреагировать система.

Существует большое множество программных средств для разработки ПО микроконтроллерных устройств (IDE). Перечислим некоторые популярные и свободно распространяемые.

Atmel Studio — бесплатная интегрированная среда разработки, она позволяет создавать программный код для микроконтроллеров линейки ATmega, ATtiny. Содержит конфигуратор проектов, внутрисистемный отладчик, редактор кода, средства математической симуляции; доступные языки программирования — ассемблер или *C* (Си).

Fritzing — бесплатное ПО со свободным исходным кодом, предназначенное для моделирования и проектирования электрических цепей и схем. Позволяет с легкостью экспортировать прототип Arduino-совместимой макетной платы в хорошего качества печатную плату для фабричного изготовления.

Arduino IDE — кросс-платформенное приложение, разработанное на *C* и *C++*. Предназначено для написания и загрузки микропрограмм в Arduino-совместимые макетные платы и в платы других производителей. Исходный код для среды Arduino IDE выпущен под общедоступной лицензией GNU. Arduino IDE поддерживает языки *C* и *C++* с использованием специальных правил структурирования кода.

CodeVisionAVR представляет собой бесплатную полноценную среду разработки программного кода, она доступна для изучения самостоятельно: доступны почти все линейки микроконтроллеров AVR, код ПО МК в проекте, создаваемый компилятором среды, компактный и читаемый. Поддерживает все популярные программаторы. Редактор может одновременно работать с несколькими проектами и автоматически сохранять резервные копии проектов. Языки программирования — *C* или ассемблер.

MPLAB IDE — бесплатная полноценная среда разработки, состоящая из набора приложений и компонентов, позволяющих упростить написание и отладку программного кода PIC-совместимых микроконтроллеров. Как указано в книге Э. Таненбаума, Д. Уэзеролла «Ком-

пьютерные сети»[3], оболочка среды собрана из самостоятельных приложений, определенным образом скомпонованных и настроенных на взаимодействие друг с другом, таких как компилятор, текстовый редактор, программный симулятор и органайзер; языки программирования — *C* или ассемблер.

CooCox CoIDE является одним из самых простых и быстрых в плане установки, освоения и настройки решений, позволяет добиваться существенных результатов даже начинающим пользователям. Качественно сделанная среда CooCox CoIDE позволяет загружать исходный код программы, редактировать его, проводить компиляцию (сторонними средствами), прошивать контроллер и проводить отладку.

Keil MDK-ARM — это среда разработки для микроконтроллеров с ядром ARM Cortex-M. Полнофункциональная версия платная, но существует бесплатная ограниченная версия, которую можно скачать с официального сайта — она имеет большую популярность и достаточно проста.

В данном учебном пособии рассмотрим платформу ARM Mbed — официальную платформу производителя (ARM), по синтаксису приближенную к Arduino, но с гораздо более широкими возможностями. Платформа позволяет написать код прошивки на языке *C* и затем автоматически его компилирует для выбранной платы микроконтроллера. Затем этот файл можно использовать для программирования непосредственно самой платы.

Для работы с Mbed можно воспользоваться средой разработки онлайн, www.mbed.com. По ссылке Compiler на главной странице сервиса запускается среда разработки (рис. 10).

Для работы с IDE онлайн необходимо создать и подтвердить аккаунт на сайте Mbed, тогда персональные программы будут сохраняться в аккаунте и не потеряются. Затем необходимо выбрать тип платы, для которой выполняется разработка. Вверху окна, справа (рис. 11) откроется диалоговое окно с выбором платы из числа часто используемых, либо можно добавить новую плату из списка поддерживаемых. Например, выберем плату Nucleo F103RB.

[3] Таненбаум Э., Уэзеролл Д. Компьютерные сети. СПб. : Питер, 2012. 5-е изд. 960 с. : ил.

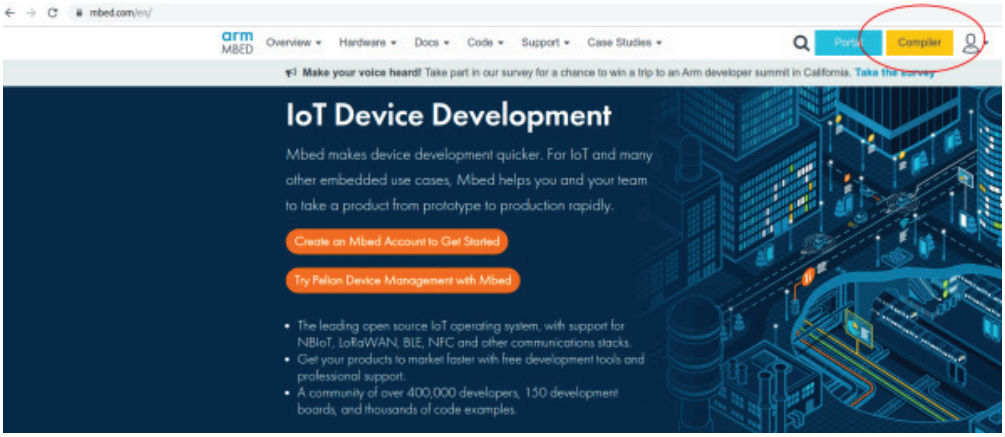


Рис. 10. Сервис среды разработки ARM Mbed

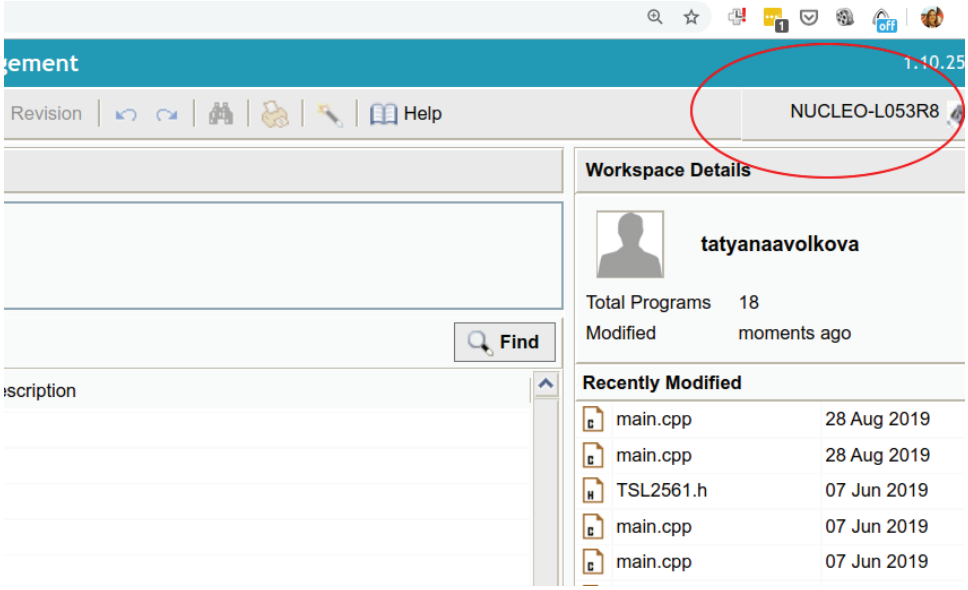


Рис. 11. Выбор микроконтроллерной платы

С первого раза эта программа не скомпилируется, поскольку в проекте не хватает библиотек. Поэтому нужно добавить библиотеки, выбрав пункт Import Library, далее — From Import Wizard. В модуль подключения библиотеки в поисковой строке нужно набрать mbed и выбрать библиотеку с названием mbed, после чего выполнить Import. В результате библиотека будет добавлена в проект и отобразится в виде шестеренки. Затем можно скомпилировать и скачать программу.

Далее необходимо подключить устройство к компьютеру, используя интерфейсный USB-кабель. Входящая в состав платы микросхема CP2102 — очень распространенный преобразователь USB-UART.

Устройство должно появиться в системе как флэшка. Получившуюся прошивку необходимо переместить на появившуюся в системе флэшку, и она загрузится в микроконтроллер. Плата будет мигать красно-зеленым светодиодом, а потом перестанет — это означает, что прошивка загрузилась.

1.4. Операционные системы реального времени

Современные микроконтроллеры как вычислительные устройства работают под операционной системой (ОС), она позволяет распределять системные ресурсы (память и вычислительные мощности), управлять синхронизацией, выполнять несколько программ одновременно, периодически переключаясь между ними. ОС микроконтроллера отличается от ОС на компьютере или мобильном устройстве, которые обычно называют операционными системами общего назначения. В ОС общего назначения важно, чтобы все работало плавно для пользователя, время на реакцию событий не всегда является критичным. В микроконтроллерных системах важно обеспечивать гарантированное время реакции на то или иное событие в системе. Если от момента возгорания до включения противопожарной системы пройдет много времени, то и тушить, возможно, будет нечего, поэтому операционные системы для МК-систем называют операционными системами реального времени (ОС РВ). Работа в реальном времени не означает скорость или высокую производительность работы, так как временной масштаб события на управляемом объекте может быть разным, все зависит от объекта автоматизации: микросекунды для летательного аппарата и десятки секунд для управления видеоинформацией. Например, система должна обработать аудиоинформацию длительностью 10 с заданного звукового спектра. Если для анализа и обработки ей потребуется 10,01 с, то эту обработку уже нельзя назвать процессом реального времени. Если же ей для обработки потребуется 9,99 с, то мы получим процесс реального времени. Существуют международные и национальные стандарты, определяющие реальное время. Стан-

дарт POSIX 1003.1 определяет реальное время в операционных системах как «способность операционной системы обеспечить требуемый уровень сервиса в заданный промежуток времени».

Для работы системы в реальном времени приходится решать проблемы привязки внутрисистемных событий к реальным моментам времени. Микроконтроллер как устройство ничего не знает о реальном физическом времени, он работает с тактами сигнала, но если отсчитать некоторое количество тактов, то можно отмерить время — при условии, что частота сигнала известна и стабильна. Например, пусть частота составляет 8 МГц, тогда за 1 мс должно пройти 8000 тактов. В состав любого МК с ядром Convex входит 24-битный таймер.

Работа в реальном масштабе времени оценивается тремя параметрами: дедлайн (deadline), латентность (latency), джиттер (jitter). Под дедлайном подразумевается критический временной интервал обслуживания или предельный срок завершения какого-то процесса. Этот временной интервал определяется характерным временем развития события в объекте/процессе, которое может привести к разрушению объекта/процесса. Латентность указывает время отклика системы на событие в целом. Джиттер характеризует разброс времен отклика системы, который может возникнуть, например, под влиянием других событий, происходящих в объекте или процессе. Данные в системе могут иметь разный уровень критичности, и если обработать их не вовремя, то это может привести к сбою или ошибкам в системе.

Работа ОС РВ определяется внешними процессами, происходящими в управляемом объекте. ОС РВ обязана реагировать на события, которые могут возникать в самом управляемом объекте или быть результатом внешнего воздействия. Время реакции на событие строго регламентировано и определяется динамикой управляемого объекта. Программный код, ответственный за обработку событий, называется задачей. Задача может находиться в состоянии выполняющейся в текущее время или невыполняющейся. Невыполняющаяся задача может быть готовой к выполнению (ждет своей очереди), заблокированной (в ожидании временного или внешнего события, например нажатия кнопки) или подвешенной (без временного ограничения, в ожидании вызова функции). На рис. 12 изображен граф переходов состояний задачи. Переход из одного состояния в другое определяется программой, таким образом система становится многозадачной.

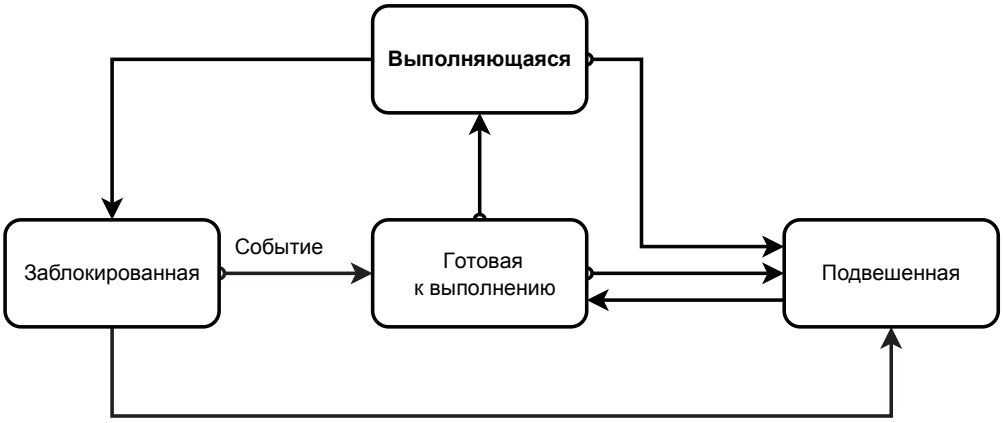


Рис. 12. Граф переходов состояний задачи

По заданному правилу ядро ОС разрешает выполняться одной из задач в один момент времени.

С точки зрения ОС внешнее событие представляется прерыванием. Процесс прерывания включает в себя следующий поток событий:

1) генерируется прерывание;
2) через некоторое время (T_{il}) получает управление обработчик прерывания;
3) обработчик в течение интервала времени T_{int} выполняет критическую по времени задачу;
4) ставится в очередь процесс, который будет осуществлять окончательную обработку прерывания;
5) по завершении задачи обработчика через T_{iret} выполняется выход из прерывания;
6) если обрабатывающий прерывание процесс — самый приоритетный, то ОС передает ему управление через T_{sl};
7) по завершении этого процесса обработка прерывания заканчивается.

В этой схеме только время обработки прерывания T_{int} зависит от содержания прерывания. Все остальные временные интервалы определяются прежде всего архитектурой операционной системы.

Для характеристики потока событий, вызываемых прерыванием, определяют еще один временной интервал — время переключения контекста T_{cont}, другими словами, время передачи управления между процессами. Таким образом, мы получаем следующие характер-

ные временные интервалы, которые определяют «реактивность» операционной системы:

- T_{il} — время задержки прерывания / ожидания обработки прерывания (Interrupt Latency);
- T_{int} — время обработки прерывания (Interrupt Time);
- T_{iret} — время возврата из прерывания (Interrupt Termination Time);
- T_{sl} — время задержки диспетчеризации (Sheduler Latency);
- T_{cont} — время переключения контекста.

Обычно соотношение временных интервалов выглядит следующим образом: $T_{sl} > T_{il} > T_{iret}$. Время переключения контекста $T_{cont} \approx 1{,}2\ T_{sl}$.

Ядро ОС РВ устроено сложнее, чем ядро ОС общего назначения, и алгоритмы его работы являются более трудоемкими. Для обеспечения предсказуемого поведения системы вводятся приоритеты. Каждая задача имеет свой приоритет, и ОС формирует список, который сортируется по приоритету: чем выше приоритет, тем важнее задача, т. е. ее необходимо выполнить в первую очередь. Если каждая задача будет выполняться со своим значением приоритета, то максимальная временная задержка есть результат суммы временной задержки, вносимой обработчиком рассматриваемого прерывания, и временных задержек, вносимых всеми более приоритетными процессами. Если же какие-либо события будут иметь одинаковый приоритет, ситуация станет непредсказуемой и заранее нельзя сказать, какой из них получит управление.

Список задач с приоритетами не может быть пустым, поэтому ОС создают задачу простоя (idle) в виде вечного цикла, имеющую наименьший приоритет. Создать задачу с вечным циклом и более высоким приоритетом нельзя, так как она всегда будет переходить из режима выполнения в режим ожидания и обратно и не даст выполниться задачам с более низким приоритетом.

За переключение задач отвечает планировщик, он входит в состав ядра ОС. В операционную систему FreeRTOS входят три алгоритма планировщика: вытесняющий алгоритм с квантованием времени, вытесняющий алгоритм без квантования времени и кооперативный алгоритм. От выбора алгоритма зависит работоспособность всей микропроцессорной системы: если был выбран неверный алгоритм, то возможна ситуация, когда задача не будет исполняться или две задачи будут пытаться работать с одним участком памяти, а следовательно, устройство будет работать неправильно.

ОС РВ не позволяет полностью абстрагироваться от аппаратной платформы, она предоставляет разработчику интерфейс распределения ресурсов МК в виде набора функций в библиотеке, которая подключается и компилируется в проекте вместе с прошивкой.

1.5. Интерфейсы передачи данных

За последние 50 лет было придумано множество разновидностей последовательных интерфейсов передачи данных, например: I^2C, I^2S, RS-232/422/485, USB, FireWire, UART, SDI-12, Ethernet и др. Под последовательной передачей данных понимают процесс передачи данных по одному биту за один промежуток времени (такт) по одному каналу связи или шине. Главной отличительной особенностью последовательной передачи является возможность контролируемой отправки данных на значительные расстояния. Последовательная передача в системах интернета вещей используется для передачи данных от датчика до микроконтроллера.

Интерфейс — это комплекс средств, предназначенных для взаимодействия двух систем друг с другом. В качестве таких систем может выступать что угодно, включая людей и искусственный интеллект.

Аналоговые датчики преобразуют значение физической величины (давление, температура, освещенность, расход, напряжение, ток) в электрический сигнал и передают его через интерфейс в контроллер. Аналоговые датчики располагаются, как правило, достаточно далеко от контроллера, поэтому их называют полевыми устройствами. Выходной сигнал датчика (полевого устройства) представляет собой аналоговый токовый сигнал диапазона 4...20 мА, на который накладывается частотно модулированный сигнал цифровой связи. Цифровые датчики, устройства с более сложной схемотехникой, передают сигналы в виде бинарного кода. Рассмотрим самые популярные последовательные интерфейсы.

Интерфейс UART (Universal Asynchronous Receiver/Transmitter) применяют как способ связи микроконтроллера и окружения. Для пе-

редачи данных микросхема, выполняющая функции передатчика, формирует код, который потом передается по символам последовательно в линию передачи данных, при этом добавляются биты, символизирующие начало передачи, окончание передачи и контроль четности. В связи с тем, что в линии нет синхронизирующих импульсов, приемник и передатчик должны быть настроены на одну частоту. Связь по UART осуществляется с помощью трех проводов: Rx (прием), Tx (передача) и общего.

FireWire — специализированный интерфейс, разработанный компанией Apple в начале 1992 г. Интерфейс FireWire позволяет на одной шине располагать несколько ведомых устройств и одно ведущее, при этом каждое имеет неповторяющийся адрес. Ценность интерфейса заключается в методе питания ведомого устройства: ему требуются всего два провода, причем по этим же двум проводам передаются данные. В более сложных ведомых устройствах используется еще один провод питания. Из-за низкой стоимости и необходимости всего двух проводов этот интерфейс распространен в самых разных решениях современной электроники. Например, на одну шину можно подключать сотни цифровых термометров с уникальными адресами, а при использовании распределителей 1-Wire — тысячи приборов на одну шину.

TWI и I^2C — последовательные двунаправленные интерфейсы, которые используют два провода для передачи данных: один провод для тактового сигнала, второй для данных. На одной шине из двух проводов одновременно могут работать до 128 устройств, каждое устройство может быть как ведущим, так и подчиненным (для этого ему нужны два адреса в сети). Для нормального функционирования шины нужно оба провода подключить через подтягивающие резисторы к источнику питания, также к источнику питания должно быть подключено каждое устройство вне зависимости от того, ведущее оно или подчиненное. Тактовый сигнал генерирует ведущее устройство, задавая скорость передачи данных в шине, но при этом есть возможность у каждого подчиненного корректировать этот тактовый сигнал и замедлять передачу во всей шине, если это необходимо для контроля качества передачи.

1-Wire — протокол передачи данных в обе стороны по одному проводу. На 1-Wire работают многие популярные датчики (домофонные чипы, датчики температуры, транзисторные ключи, программируемые порты ввода-вывода, АЦП и ЦАП, часы реального времени и др).

В 2007 г. был предложен стандарт беспроводной передачи данных WirelessHART — данные передаются на частоте 2,4 ГГц. Основу протокола составляет технология самоорганизующихся ячеистых сетей (Mesh Network). Они образуются на основе множества соединений типа «точка — точка», находящихся в области радиопокрытия друг друга.

Ячеистая сеть — это распределенная система передачи данных. Все узлы в этой сети (устройства, передающие данные) соединяются, могут обмениваться сигналами и ретранслировать их дальше.

Такая сеть обеспечивает максимальную надежность передачи данных за счет наличия резервных маршрутов: при выводе одного датчика из эксплуатации данные будут передаваться по другому пути и сеть останется работоспособной.

Технология WirelessHART позволяет беспроводным полевым приборам самостоятельно взаимодействовать друг с другом: автоматически соединяться и выполнять функции транзитной передачи данных для других участников сети.

1.6. Практические задания

Занятие № 1

Первое занятие по курсу предназначено для входного контроля аудитории, знакомства с терминологией курса и его основными темами. Для его проведения лучше всего подходят игровые методы обучения.

Задание 1. Игра «Найди пару и расскажи о себе»

Цель игры: протестировать уровень знания студентов.

Содержание игры: все студенты разбирают перевернутые карточки со стола преподавателя. На каждой карточке написан какой-либо термин. Задача студентов: разбиться на пары по смысловому содержанию терминов. Каждой паре студентов нужно найти объяснение терминов и придумать третий термин, связанный с этими двумя. Далее пары представляются аудитории, называют термины, объясняют их смысл и называют придуманный ими третий термин, подходящий к озвученной паре.

Примерный список пар терминов:
Bluetooth — Wi-Fi
Raspberry Pi — Arduino
Сайт — Контент
Микроконтроллер — Программный код
Облако — Большие данные
Сервер — База данных
Сервис — Облачные вычисления
Нейронная сеть — Большие данные
Маршрутизатор — Компьютерная сеть

Примечание: если студенты разделились не так, как задумал преподаватель, важно, чтобы они смогли объяснить логическую связь между терминами или, если это невозможно, осознали свою ошибку.

Задание 2. Командная игра-соревнование на знание терминов

Деление на три команды происходит следующим образом: после того как студенты вытянули каждый по листочку с термином, им необходимо разобраться, какие термины можно объединить в группы, и по этому принципу объединиться в три команды. Например, могут образоваться команда «IoT» (микроконтроллер, актуатор, реле, сенсор, печатная плата), команда «Сети» (коммутатор, маршрутизатор, шлюз, протокол, IP-адрес, оптоволокно), команда «Облачные сервисы» (серверное приложение, сетевой диск, конфигурация, виртуальная машина, докер).

Каждая команда получает набор карточек: на одних написаны термины, на других — определения. Задача — каждому термину сопоставить определение. Команда-победитель получает поощрительные баллы и право начать следующую игру.

Задание 3. Викторина «Кто хочет стать IoT-инженером»

В игре по мотивам телевизионной викторины «Кто хочет стать миллионером» участвуют три команды. Соответственно, на вопрос викторины дается командный ответ. За правильный ответ начисляются баллы в зависимости от сложности вопроса. Если одна команда ответила неверно, право отвечать переходит к следующей.

Занятие № 2

Цель занятия: знакомство с микропроцессорными устройствами IoT.

Занятие проходит в форме командной игры. Каждая команда получает конверт, в котором содержится описание системы интернета вещей.

Умный автополив

Устройство интернета вещей предназначено для автоматического полива комнатных растений из накопительного резервуара. На основании данных о влажности почвы и установленных настроек производится полив в течение заданного времени с помощью погружного насоса.

Данные о влажности почвы периодически передаются на интернет-сайт через Wi-Fi домашней сети.

При отсутствии воды в резервуаре полив не производится, а на электронную почту хозяина отправляется сообщение.

Умное освещение

Подъезд многоквартирного дома оснащен устройствами умного освещения на каждом этаже. Устройства включены в локальную сеть дома по технологии Ethernet. В темное время суток (по показаниям датчика освещенности) при появлении человека в зоне видимости PIR-датчика включается светодиодный светильник, а по локальной сети передаются команды включения света на соседних этажах. Данные о передвижениях и времени работы освещения сохраняются на внутреннем сайте многоквартирного дома.

Умное отопление

Загородный дом оснащен системой умного отопления, которая управляется через беспроводную сеть LoRa. Заданные параметры температурного режима поддерживаются при помощи датчиков температуры и влажности, которые расположены в нескольких помещениях дома. Система управляет электрическим отопительным котлом и может направлять теплоноситель в различные зоны при помощи электромагнитных клапанов.

Задание 1

По описанию системы команде необходимо выбрать необходимые устройства — компоненты системы (датчики, микросхемы, актуаторы) из разложенных на столе. Чтобы было интереснее, среди предложенных устройств могут быть лишние.

Задание 2

Команды должны из карточек с изображением компонентов системы собрать необходимый набор и составить схему умного устройства, как пазл. Дидактический материал с изображением схем и описанием компонентов представлен в прил. 1. Каждая схема разрезается на шесть частей. Чтобы усложнить задание, к предложенным карточкам будут добавлены лишние.

Задание 3

На составленной в задании 2 схеме следует разместить устройства, которые были собраны в задании 1. Побеждает та команда, которая максимально быстро соберет схему из карточек и возьмет наибольшее число необходимых устройств-компонентов.

Занятие № 3

Лабораторная работа «Программирование микроконтроллера STM32 на эмуляторе simulator.mbed.com»

Цель работы: реализация базового функционала устройства интернета вещей с использованием микроконтроллера STM32.

Задачи: изучить среду программирования микроконтроллеров; выполнить программирование микроконтроллера учебного робота с ультразвуковыми датчиками и оптическим датчиком обнаружения препятствия.

Необходимое оборудование: учебный робот, датчик расстояния ультразвуковой (2 шт.), датчик препятствия оптический (1 шт.), набор светодиодов (см. список в прил. 2).

Порядок выполнения работы:

1) зарегистрируйтесь на портале mbed.com;
2) зайдите на страницу эмулятора simulator.mbed.com;
3) изучите документацию;
4) создайте простой проект управления кнопкой светодиода;
5) добавьте датчик температуры;
6) напишите код, реагирующий на нажатие кнопок и считывающий данные с датчика;
7) добейтесь стабильной работы управления светодиодами в зависимости от показателя датчика;
8) в приложении mbed.com выберите микроконтроллер, установленный на роботе;
9) напишите управляющую программу микроконтроллера, позволяющую роботу двигаться по линии и объезжать препятствия;
10) изучите работу ультразвукового датчика препятствия при помощи учебного робота; рассмотрите реакцию датчика на материал объектов, расположенных перед ним;
11) изучите работу оптического датчика препятствия при помощи учебного робота; рассмотрите реакцию датчика на цвет линии и освещенность помещения;

12) постройте диаграммы направленности для датчиков в зависимости от разных материалов.

1.7. Дополнительные материалы

Посмотрите видеолекцию «Архитектура и типология систем IoT», читают Ксения Сизова (RedBees), Антон Куропятник (WoodenShark).

IT Академия Samsung. Архитектура и типология систем IoT : [видеолекция] // YouTube.com : [сайт]. URL: https://youtu.be/jwINTuBXQYA (дата обращения: 30.09.2021).

Ознакомьтесь с видеолекцией «Основы программирования микроконтроллеров». Читает Олег Артамонов — генеральный директор компании Unwired Devices («Беспроводные устройства»).

IT Академия Samsung. Основы программирования микроконтроллеров : [видеолекция] // YouTube.com : [сайт]. URL: https://youtu.be/finP05FFTv8 (дата обращения: 30.09.2021).

Контрольные вопросы и задания

1. Опишите цели разработки IoT-системы и основные ее преимущества для бизнеса.
2. Кто может выступать в качестве получателя и отправителя в системе IoT?
3. Назовите примеры и характеристики конечного узла в сети.
4. Опишите основную функцию базовой станции.
5. Какие задачи должен выполнять веб-интерфейс IoT-системы?

6. Какие характеристики выделяют у радиотехнологий?
7. Что такое модуляция сигнала и зачем она нужна?
8. Какие проблемы связаны с емкостью сети?
9. Нарисуйте цепочку передачи информации:
   — при телефонном разговоре;
   — использовании беспроводных наушников;
   — использовании фитнес-браслета;
   — локации по Bluetooth-меткам;
   — использовании умных счетчиков воды.
10. Найдите компоненты на изображении платы (см. рис. 8): микроконтроллер, систему питания, GPIO-выводы, отладочный разъем, светодиод, кнопки управления и перезагрузки.
11. Опишите роль и особенность каждого вывода на плате МК.
12. В чем отличия микроконтроллера от микропроцессора?
13. Нарисуйте типовую архитектуру IoT-устройства.
14. Составьте блок-схему алгоритма работы одной из умных систем, описанных в разделе 1.6, занятие № 2.
15. Зачем микроконтроллеру операционная система?
16. В каких состояниях может находиться задача в программе МК?
17. Опишите преимущества микроконтроллеров с ядром Cortex-M.
18. Какую роль выполняют уровни абстракций?
19. Опишите принцип работы таймера в МК.
20. Что означает межпроцессорная коммуникация?

Глава 2. Протоколы и передача данных в системах интернета вещей

2.1. Модели сетевого взаимодействия

В 1982 г. Международная организация по стандартизации (ISO) разработала и внедрила проект в области сетевых технологий, названный Open Systems Interconnection (OSI/ISO) — взаимодействием открытых систем. До OSI сетевые технологии были полностью проприетарными, основанными на корпоративных стандартах. OSI стала попыткой создания сетевых стандартов для обеспечения совместимости решений разных поставщиков. В то время многие большие сети были вынуждены поддерживать несколько протоколов взаимодействия и включали большое количество устройств, не имеющих возможности общаться с другими устройствами из-за отсутствия общих протоколов. Чтобы сетевые устройства могли «прозрачно» взаимодействовать друг с другом, разработали модель OSI. Модель определяет семь уровней взаимодействия систем (рис. 13).

Каждому уровню соответствует свой логический элемент данных, которыми можно оперировать на уровне в модели и используемых протоколах. Приведем описание уровней.

Физический уровень (1) определяет стандарты для сред передачи данных по типу передающей среды (медный кабель, витая пара, оптоволокно, радиоэфир, луч и др.), по типу модуляции сигнала и по сигнальному уровню логических дискретных состояний (нули и единицы). Логический элемент данных — бит.

Модуляция сигнала — это процесс изменения одного или нескольких параметров высокочастотного несущего колебания по закону низкочастотного информационного сигнала. Различают импульсно-амплитудную модуляцию, широтно-импульсную модуляцию и фазовую модуляцию.

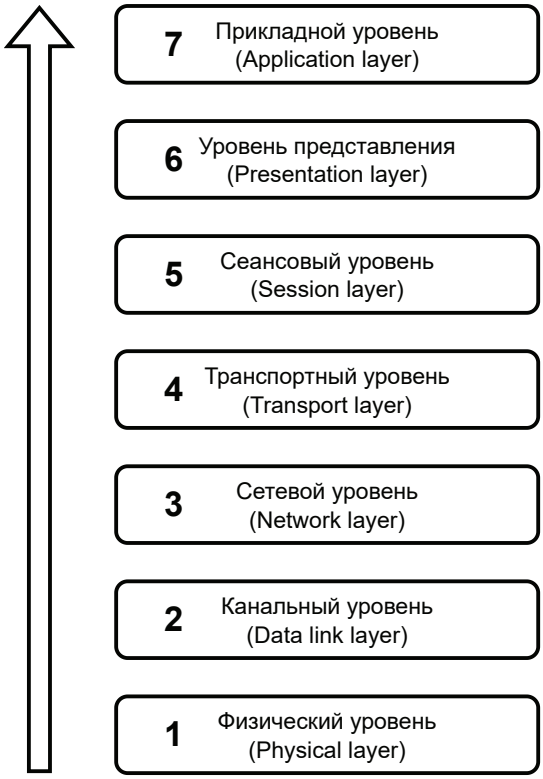


Рис. 13. Сетевая модель OSI

Канальный уровень (2) организует взаимодействие сетей на физическом уровне и контролирует возникающие при этом ошибки. Биты физического уровня он упаковывает в кадры. Протоколы канального уровня обеспечивают целостность данных и отправляют их на сетевой уровень.

Сетевой уровень (3) модели предназначен для определения пути передачи данных, он отвечает за трансляцию логических адресов и имен в физические, определение кратчайших маршрутов, коммутацию и маршрутизацию, отслеживание неполадок и «заторов» в сети. На сетевом уровне кадры упаковываются в пакеты.

Транспортный уровень (4) обеспечивает надежную передачу данных от отправителя к получателю. Здесь пакеты данных преобразуются в сегменты.

Сеансовый уровень (5) предназначен для поддерживания сеанса связи, приложения взаимодействуют между собой длительное время.

Логический фрагмент данных на этом и других высоких уровнях называется сообщением.

Уровень представления (6) преобразует данные: кодирует или декодирует. На этом уровне может осуществляться сжатие/распаковка, обеспечивается организация данных при их пересылке, защита данных и межплатформенные преобразования.

Прикладной уровень (7) позволяет работать с данными в пользовательских приложениях.

В модели OSI существуют ограничения: протокол может выполнять функции только своего уровня и не может выполнять функций другого уровня; любой протокол должен взаимодействовать либо с протоколами своего уровня, либо с протоколами на единицу выше и/или ниже своего уровня.

Эталонная модель OSI была большим шагом в создании концепций современных сетей передачи данных. Она популяризовала идею общей модели протоколов, расположенных на различных уровнях и определяющих взаимодействие между сетевыми устройствами и программным обеспечением, но не стала общепринятым стандартом. На практике модель сетевого взаимодействия постепенно стала изменяться в сторону упрощения. Например, стек протоколов TCP/IP (рис. 14) содержит всего четыре уровня: уровень доступа к сети, уровень сети Интернет, транспортный уровень и уровень приложений. Модели, положенные в основу стеков протоколов промышленных сетей, также существенно отличаются вследствие узкой специализации этих сетей.

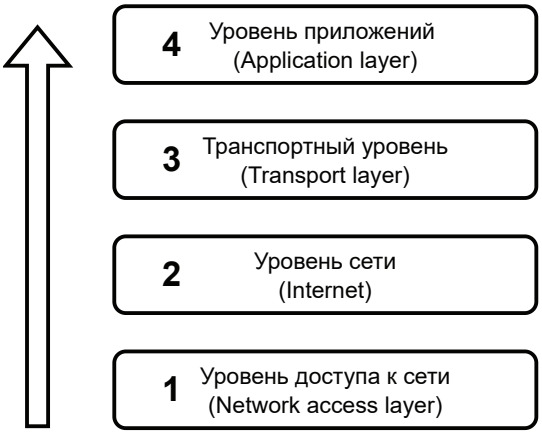


Рис. 14. Сетевая модель TCP/IP

Сетевая модель построения систем IoT также отличается от OSI. Можно выделить логическую 12-уровневую модель для IoT-систем:

1) физический уровень: представляет собой сбор информации (датчики) и осуществление механической работы (актуаторы);
2) уровень периферийного вычисления: обеспечивает преобразование аналоговой информации в цифровую и/или наоборот; уровень отвечает за сбор информации с датчика, но также и за приведение ее к стандартному виду, фильтрацию помех и предварительный анализ;
3) уровень периферийной коммуникации: обеспечивается передача данных от уровня периферийного вычисления до шлюза (адаптера);
4) уровень шлюза: этот уровень маршрутизации осуществляет отправку данных с периферийных устройств и получает данные конфигурации для них, сохраняет информацию о статусе периферийных устройств и собранные ими данные;
5) уровень внешней связи: передача данных по каналам сети Интернет; логический уровень внешней связи имеет стандартизированный протокол для решений IoT — LvM2M;
6) уровень безопасности: обеспечивает функции AAA (Authentication, Authorization, Accounting — аутентификация, авторизация и учет) и шифрование/дешифрование;
7) уровень внутрисерверной связи: обеспечивает передачу потоковой информации;
8) уровень сбора, обработки и хранения данных: сбор данных, приведение информации к стандартному виду, сохранение данных, управление жизненным циклом информации, уведомление других сервисов о поступлении новых данных;
9) уровень аналитики: данные анализируются и предсказываются возможные проблемы;
10) уровень уведомлений: конечные пользователи уведомляются о событиях;
11) уровень представления: отвечает за обслуживание, конфигурацию и изменения состояния системы, включая периферийные устройства и шлюзы;
12) уровень конфигурации: работает как хранилище для трех типов статусов периферийных устройств:
    — актуального состояния периферийного устройства;

— нового состояния периферийного устройства, которое будет загружено;
— промежуточного статуса периферийного устройства — указывает на процесс обновления от старых состояний к новым.

Перечислим протоколы, которые могут обеспечивать функциональность уровней в модели TCP/IP (рис. 15). Физический уровень модели может обеспечить протокол Ethernet — передача данных по проводу (витая пара) или SONET (Synchronous Optical Network) — данные передаются по оптоволоконному кабелю, протокол 802.11 — передача данных по радиоволнам, протокол DSL (Digital Subscriber Line) — данные передаются по проводам телефонной сети. На сетевом уровне протокол ICMP (Internet Control Message Protocol) используется для диагностики проблем со связностью в сети, протокол IP (Internet Protocol) обеспечивает адресацию пакетов данных. Транспортный уровень может представлять протокол TCP (Transmission Control Protocol): он устанавливает соединение и гарантирует, что никакие сообщения не потеряются или UDP (User Datagram Protocol) — для UDP-датаграмм ценность не установлена, сообщение может потеряться, но передача происходит быстро. На прикладном уровне HTTP (HyperText Transfer Protocol) или RTP (Real-Time Transport Protocol) передают полезные данные для сети IoT-устройств.

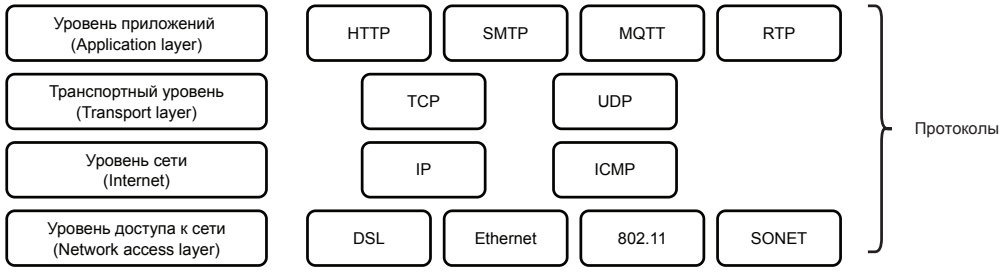


Рис. 15. Примеры протоколов в TCP/IP

2.2. Адресация в сети

Стек протоколов обеспечивает соединение отдельных подсетей, технологий передачи данных на уровне сигнала, имеет свою структуру адресации, и поэтому для идентификации узла в сети, передающего или принимающего данные, необходим единый способ адресации. Таким способом в TCP/IP-сетях является IP-адресация.

В стеке TCP/IP используются три типа адресов:

— физический, или аппаратный;
— IP-адрес (адрес сетевого адаптера);
— символьный, или доменный, адрес (имя).

Физический адрес — это уникальный номер, присвоенный сетевому устройству производителем в соответствии с выделенным диапазоном. Если подсетью является локальная сеть Ethernet, Token Ring или FDDI, то физический адрес — это MAC-адрес (Media Access Control).

MAC-адрес может быть у сетевого адаптера или порта маршрутизатора. Он является уникальными, имеет размер 6 байт и записывается в шестнадцатеричном виде, например A4-17-31-19-8B-17.

IP-адрес является основным типом адреса, на основании которого сетевой уровень передает/принимает данные в виде IP-пакетов. Существуют два вида IP адресов: IPv4 (Internet Protocol version 4) и IPv6 (Internet Protocol version 6).

Адрес IPv4 состоит из 4 байт. Устройство может одновременно входить в несколько сегментов сетей. В этом случае оно имеет несколько IP-адресов, по числу подключенных сетевых портов. Номер узла (сетевого соединения) в протоколе IP назначается независимо от его локального адреса. Следовательно, IP-адрес характеризует не отдельное устройство, подключенное в сеть, а одно сетевое соединение.

Порт — это сложная розетка, способная принимать-передавать не только информационные, но и управляющие сигналы. Стоит порт обычно на материнской плате или плате расширения (звуковой карте, видеокарте), он доступен на задней или передней стенке системного блока.

С помощью протокола IPv4 можно создать 2^{32} адреса. Со временем число устройств, требующих IP-адреса, стало больше, чем 4 294 967 296,

и был разработан протокол IPv6 (стандарт протокола IPv6 — документ RFC 2460 — был принят в 1998 г.). Длина IP адресов в протоколе IPv6 составляет 16 байт, следовательно, можно создать 2^{128} IP-адресов. Такого количества хватит, для того чтобы подключить в сеть очень много устройств.

Символьные (доменные) имена несут вспомогательную функцию комфортной работы с адресами. Людям удобнее использовать текстовые адреса, чем цифровые. Специальная служба DNS (Domain Name System) устанавливает соответствие между доменными именами и цифровыми IP-адресами.

Далее рассмотрим формат протоколов IPv4 и IPv6.

IPv4 адрес представляет собой 32-разрядное двоичное число, разделенное на группы по 8 бит, называемые октетами, например:

01000001 11111111 00101100 11110000

Удобно записывать IP-адрес в десятичной системе счисления: в виде четырех чисел, разделенных точками. Приведенный выше IP-адрес можно записать в следующем виде: 65.255.44.240.

Максимальное значение октета в двоичной системе счисления равно 11111111_2, что соответствует в десятичной системе 255_{10}. Поэтому IP-адрес, в котором хотя бы один октет выходит за пределы допустимого диапазона, является недействительным или несуществующим.

Хост (host) — это любое устройство, подключенное к локальной или глобальной сети. Им может быть компьютер, смартфон, телевизор, планшет, маршрутизатор, принтер и т. д.

Рассмотрим информационную составляющую IP-адреса. Он состоит из двух логических частей: первая часть определяет номер подсети (ID подсети), вторая часть — номер узла (ID хоста) в этой подсети. При передаче пакета из одной подсети в другую используется ID подсети. Когда пакет попал в подсеть назначения, ID хоста указывает на конкретный узел в рамках этой подсети. Если известно число разрядов, отводимых для представления номера узла (номера подсети), можно определить общее количество узлов (или подсетей).

Определение: если число разрядов для представления номера узла равно N, то общее количество узлов равно $(2^N - 2)$. Два узла вычитают-

ся вследствие того, что адреса со всеми разрядами, равными нулям или единицам, являются особыми и используются в специальных целях.

Например, если под номер узла в некоторой подсети отводится десять бит, то общее количество узлов в такой подсети равно $2^{10}-2 = = 1022$ узла.

Чтобы однозначно определить, какая часть IP-адреса отвечает за ID подсети, а какая — за ID хоста, применяются маски. В маске биты, равные 1, обозначают ID подсети, а биты, равные 0, — ID хоста. Например, маска 255.255.0.0. обозначает, что в адресе по 16 бит отводится под адреса подсети и хоста.

При разработке IPv6 был упрощен протокол, для того чтобы маршрутизаторы могли обрабатывать пакеты IPv6 быстрее, и обеспечена возможность защиты данных с помощью шифрования. Масок в IPv6 нет, но основное изменение — это длинные адреса отправителя и получателя. IPv6 состоит из 128 бит, разделенных на восемь 16-битных блоков. Каждый блок затем преобразуется в 4-значные шестнадцатеричные числа, разделенные двоеточиями: fe80::2003:d3a:a6fe:4bc9.

В IPv6 поле «Время жизни пакета» заменили на максимальное число транзитных участков, потому что на практике вместо времени жизни, даже в протоколе IPv4, указывается максимальное количество маршрутизаторов, через которое может пройти пакет, перед тем как он будет отброшен.

Существует проблема с использованием протокола IPv6 в системах интернета вещей, поскольку адрес занимает больше памяти, чем сами данные, которые нужно передать. Эта проблема решается с помощью сетевой технологии 6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks), рис. 16.

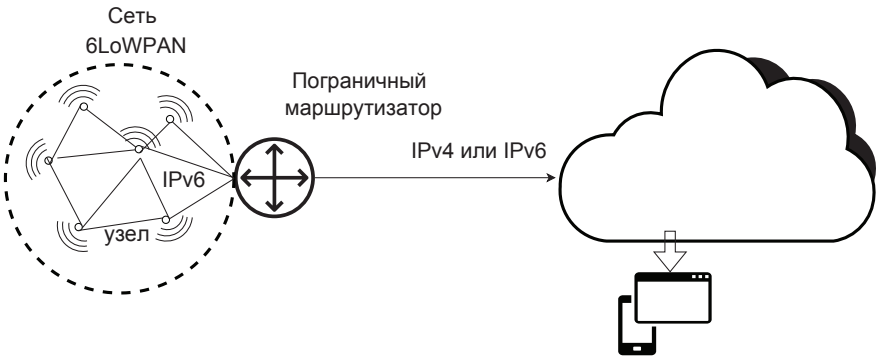


Рис. 16. Структура сетевой технологии 6LoWPAN

Условно выделяют «маленький» интернет, в котором находятся небольшие IoT-устройства, и «большой» (глобальную сеть), а посередине — шлюз (или роутер), который преобразует длинный адрес IPv6 в короткий 16-битный уникальный адрес «маленькой» сети.

Подключение множества устройств к «большому» интернету реализуется путем эффективной передачи пакетов IPv6 в небольших фреймах канального уровня, определенных в беспроводном стандарте IEEE 802.15.4.

2.3. Протоколы и технологии обмена данными

Рассмотрим технологии, позволяющие передавать данные в глобальную сеть Интернет. Микроконтроллер или шлюз в системах IoT должен передать данные серверу для их дальнейшей обработки и хранения. Для передачи данных можно использовать разные сетевые технологии: как проводные, так и беспроводные.

Ethernet (802.3) — технология передачи данных, с которой сталкивается каждый пользователь ПК, включенного в локальную сеть по медному или оптическому кабелю. Его назначение — передавать данные по витой паре (100 Мбит/с) или оптоволокну (1 Гбит/с.) Распознавание узла, передающего данные, происходит с помощью MAC-адреса, который «вшит» в аппаратную часть оборудования. Регистрация MAC-адреса происходит один раз при инициализации оборудования. Подключить микроконтроллер к локальной сети можно с помощью встроенного или внешнего модуля, предназначенного для соединения оборудования по протоколу 802.3.

Если система IoT включена в локальную сеть через Ethernet, то для передачи данных можно использовать протокол Modbus — это распространенный промышленный протокол для M2M-взаимодействия в системах АСУ ТП. Выпускается огромное количество датчиков, вычислителей, шлюзов и исполнительных устройств, работающих по протоколу Modbus, поэтому он позволяет легко собрать в единую сеть оборудование разных производителей. В Modbus-сети существует единственное ведущее устройство, у которого отсутствует адрес, и оно с фиксированным интервалом отправляет запросы ведомым устройствам — каждое из них имеет адрес от 1 до 254, двух одинаковых адресов быть в одной сети не должно.

Основное достоинство стандарта — в простоте использования, но есть и существенные недостатки. Перечислим основные:

— есть только два типа данных: флаг и шестнадцатибитное слово;
— адресация в системе задается вручную;
— без запроса ведущего подчиненный не может передать данные;
— невозможно в одном запросе получить данные из двух регистров, расположенных не рядом;
— подчиненный не может обнаружить разрыв связи с ведущим.

В сетях TCP/IP, где физическим каналом передачи данных является Ethernet-интерфейс, используется протокол Modbus TCP — данные кодируются в двоичный формат и упаковываются в пакеты[4].

При отсутствии технической возможности проводного подключения устройства IoT к локальной сети используются беспроводные технологии передачи данных.

Беспроводные технологии Wi-Fi и Bluetooth целесообразно использовать на небольших расстояниях (рис. 17). Преимущества этих технологий заключаются в высокой скорости передачи данных, низкой стоимости приобретения и владения, отсутствии необходимости шифрования. Для больших расстояний используются другие технологии. Если данные системы IoT занимают порядка десятка килобайт, скорость их передачи не является критичной и система IoT локализована на территории нескольких километров, то можно использовать технологии LPWAN, которые работают с маленькими данными, но для задач мониторинга вполне приемлемы. Не нужно беспокоиться о расстоянии, и устройство будет долго работать от автономного источника питания. Однако такая технология требует построения собственной сети передачи, которая не сможет конкурировать с покрытием сетей мобильных операторов. Мобильная связь позволяет передавать большие объемы данных с высокой скоростью и не требует первоначальных затрат. Использование общедоступных сетей требует дополнительных затрат на обеспечение безопасности данных.

[4] Подробнее см.: Как общаются машины: протокол Modbus // Habr.com : [сайт]. URL: https://habr.com/ru/company/advantech/blog/450234/(дата обращения: 30.09.2021).

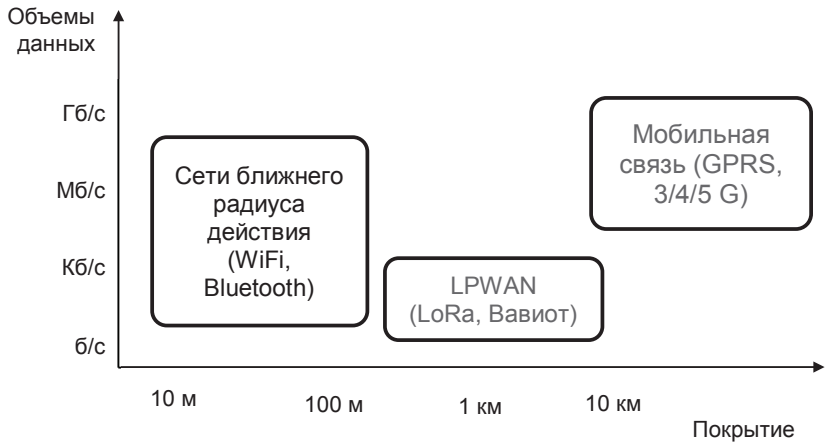


Рис. 17. Сравнение беспроводных сетей

Самый популярный протокол обмена данными в беспроводных системах интернета вещей — MQTT (Message Queuing Telemetry Transport). Это открытый сетевой протокол, работающий поверх TCP/IP, разработанный специально для малых вычислительных возможностей устройств интернета вещей. Отличительная особенность протокола MQTT в том, что он может работать в условиях частой потери связи, а в структуре передаваемых данных содержится минимальное количество служебной информации, чего нельзя сказать, например, об HTTP.

Основная идея MQTT заключается в следующем: специальное устройство, на котором расположен MQTT-брокер, служит посредником во взаимодействии MQTT-агентов — издателей (publishers) и подписчиков (subscribers). Агенты (издатели и подписчики) могут подключаться к брокеру и публиковать темы (топики) и сообщения или подписаться на темы и получать сообщения, опубликованные в этих темах. После завершения своей работы агент отключается от брокера.

На рис. 18 изображена упрощенная схема взаимодействия между подписчиком и издателем с использованием MQTT-брокера.

Имя топика «/home/alarms/1/status» обладает иерархической структурой и похоже на путь в файловой системе. Протокол MQTT позволяет использовать в имени топика подстановочные символы, что упрощает процесс подписки. С помощью подстановочного символа # можно подписаться и отслеживать состояния всех датчиков в топике.

Например, подписка на топик «/home/#» позволит получать сообщения от всех устройств в доме.

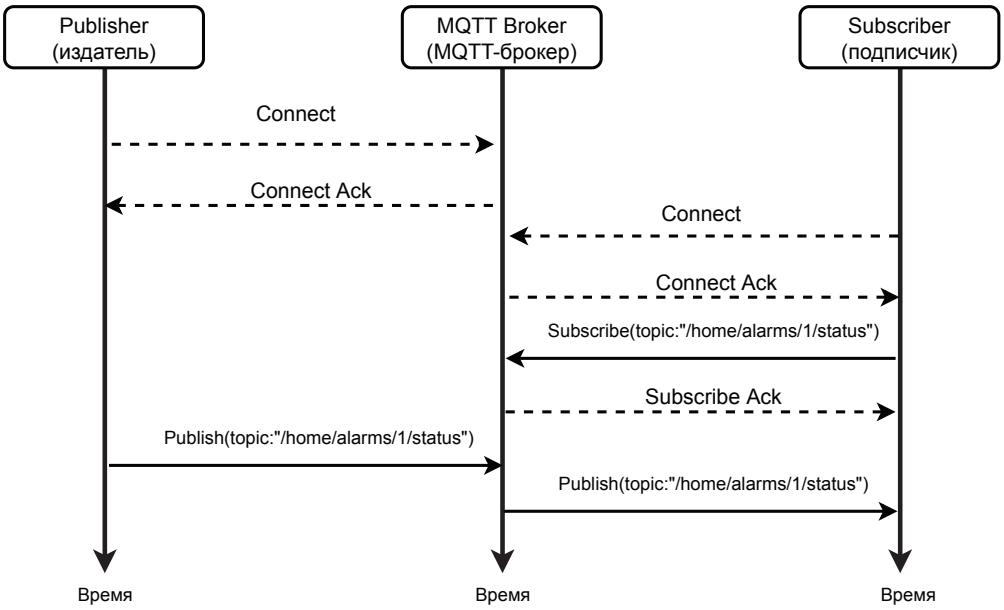


Рис. 18. Взаимодействие между брокером, издателем и подписчиком

MQTT поддерживает три уровня качества обслуживания (Quality of Service — QoS): QoS 0, QoS 1, QoS 2.

На уровне QoS 0 используется подход «максимум однократная доставка». Подписчик не подтверждает получение сообщений, издатель передает сообщение лишь один раз, не повторяя передачу в будущем.

Уровень QoS 1 определяет подход «минимум однократная доставка». Гарантируется, что подписчик получит сообщение хотя бы раз. Издатель будет отправлять сообщение повторно до тех пор, пока не получит подтверждение об успешной доставке.

Уровню QoS 2 соответствует самая медленная процедура доставки сообщений, но при этом обеспечивается наибольшая надежность. В нем реализуется подход «однократная доставка», но применяется четырехступенчатая процедура подтверждения.

Выбор уровня качества обслуживания зависит от специфики передаваемых данных и от того, насколько важно обеспечить их целостность.

2.4. Защита данных в беспроводных системах

Как быстроразвивающаяся технология, IoT испытывает ряд проблем, среди которых наиболее серьезной является проблема безопасности. С каждой новой умной вещью, подключенной к сети, выше риски, связанные с несанкционированным доступом в IoT-систему и использованием ее возможностей злоумышленниками. Понятно, что особого вреда не принесут сбои чьих-либо умных часов или фитнес-трекеров. Но вот взлом IoT-устройств, которые входят в систему M2M, может повлечь непредсказуемые последствия. В этом случае степень их безопасности должна соответствовать важности той или иной инфраструктуры: транспортной, энергетической и др., от которых зависит жизнедеятельность людей и работа экономики.

Одной из главных проблем интернета вещей является отсутствие стандартов и низкая безопасность передачи данных в силу малых вычислительных мощностей конечных устройств.

В мире бизнеса постепенно начинают признавать потребность в единых и доступных стандартах для IoT. Институт инженеров по электронике и электротехнике (IEEE) в лице своей Ассоциации по стандартизации разработал ряд стандартов и протоколов, призванных помочь в развитии подключенных систем.

Существует много стандартов для протоколов безопасности интернета, разработанных в различных органах по стандартизации. Эти протоколы безопасности предлагают различную защиту на разных уровнях и эффективно дополняют друг друга, но при этом слишком тяжелы для вычислительных возможностей простых микроконтроллеров, которые применяются в устройствах IoT.

Данные с конечных устройств передаются через множество сетей разного типа к облачной IoT-платформе. На этом пути данные несколько раз кодируются шлюзами безопасности, и это усложняет обеспечение комплексной безопасности системы Internet of Things. Общая безопасность всей IoT-системы будет зависеть от операторов связи и производителей шлюзов. Важно определить угрозы и риски, которые могут быть в IoT, и впоследствии подобрать необходимую защиту, которая должна быть развернута на всех уровнях сетевого взаимодействия.

Для масштабного решения проблем безопасности с помощью сертификации необходимо объединяющее решение, общая мотивация

для всех производителей выпускать защищенные устройства, а потребителям — отказаться от покупки тех изделий, безопасность которых не подтверждена. Прежде всего высочайшая безопасность устройств интернета вещей должна быть главной задачей именно производителей. В итоге результатом должен стать набор стандартов для безопасности всемирной системы интернета вещей.

2.5. Практические задания

Занятие № 4

Цель занятия: закрепить знания о 7-уровневой модели ISO, модели TCP/IP и передаче данных на уровне протоколов.

Задание: игра в командах «Распредели протоколы по уровням ISO»

Каждой команде выдаются карточки с названием некоторых протоколов, уровней модели ISO и функций уровней. Задачи команды — правильно распределить уровни, функции и протоколы для конкретной сети передачи данных; объяснить, как работают протоколы на разных логических уровнях сети.

Занятие № 5

Цель занятия: сформировать понятие об IP-адресации в компьютерных сетях.

Задание 1

Если маска подсети 255.255.255.224 и IP-адрес компьютера в сети 162.198.0.157, то чему равен порядковый номер компьютера в сети?

Решение

1. Так как первые три октета (октет — число маски, содержит 8 бит) равны 255, то в двоичном виде они записываются как 24 единицы, а значит, первые три октета определяют адрес сети.
2. Запишем число 224 в двоичном виде: $224_{10} = 11100000_2$.
3. Запишем последний октет IP-адреса компьютера в сети: 15710 = 10011110110.
4. Сопоставим последний октет маски и адреса компьютера в сети:
   111**00000**
   10011101

Жирным выделена нужная нам часть, отвечающая (по условию) за адрес компьютера в подсети. Переведем ее в десятичную систему счисления: $11101_2 = 29_{10}$.

Ответ: 29.

Задание 2

Для некоторой подсети используется маска 255.255.254.0. Сколько различных адресов компьютеров теоретически допускает эта маска, если два адреса (адрес сети и широковещательный) не используют?

Решение

1. Первые два октета равны 255, а значит, в двоичном виде они записываются как 16 единиц, следовательно, первые два октета определяют адрес сети.
2. Запишем число 254 в двоичном виде: $254_{10} = 11111110_2$. В конце этого числа стоит один ноль, еще 8 нолей мы получаем из последнего октета маски. Итого у нас есть 9 двоичных разрядов для того, чтобы записать адрес компьютера.
3. Поскольку $2^9 = 512$, но два адреса не используются, получаем $512 - 2 = 510$.

Ответ: 510.

Самостоятельная работа

1. Укажите MAC-адрес вашего рабочего компьютера:

   _____

2. Укажите IP-адреса рабочего компьютера в сети:

   _____

3. Проиллюстрируйте, как вы узнали эти данные.
4. Укажите маску подсети рабочего компьютера:

   _____

5. Определите, какое максимальное число активных подключений допускает эта маска:

   _____

6. Найдите порядковый номер компьютера в сети:

   _____

7. По IP-адресу сети и маске определите адрес сети:

   _____

8. Определите класс сети:

   _____

9. Проделайте аналогичную работу (п. 5–8), если маска подсети 255.255.252.0 и IP-адрес компьютера в сети 226.185.90.162.

Занятие № 6

Цели занятия: научиться настраивать клиент-серверное взаимодействие устройств интернета вещей с сервером сбора данных; изучить настройку мониторинга данных по протоколу MQTT.

Порядок выполнения работы:

1) установите и настройте серверное программное обеспечение MOSQUITTO;
2) установите и настройте клиентское программное обеспечение MOSQUITTO;
3) проверьте клиент-серверное взаимодействие MOSQUITTO отправкой и получением сообщений;
4) напишите серверный обработчик, подписанный на канал сервера и выполняющий действия в зависимости от приходящих от клиента данных;
5) протестируйте работоспособность взаимодействия.

Методические указания

Для выполнения работы необходимо изучить протокол MQTT.

Сервер MOSQUITTO — лишь один из возможных вариантов сервера. Можно использовать и другие: HiveMQTT, HBMQTT.

MOSQUITTO — это сервер, который можно запустить локально на ПК. Для выполнения работы установите себе на компьютер MOSQUITTO. Рекомендуется работать с сервером в ОС Linux. Установка на Ubuntu: sudo apt-get install mosquitto.

Можно установить MOSQUITTO и в Windows, он появится в Службах Windows, и его можно будет запускать и останавливать из оснастки управления службами (рис. 19).

MOSQUITTO можно использовать на шлюзе между локальной сетью системы интернета вещей и «большим» интернетом.

Для работы с MOSQUITTO необходимо освоить базовый функционал локального MQTT-сервера.

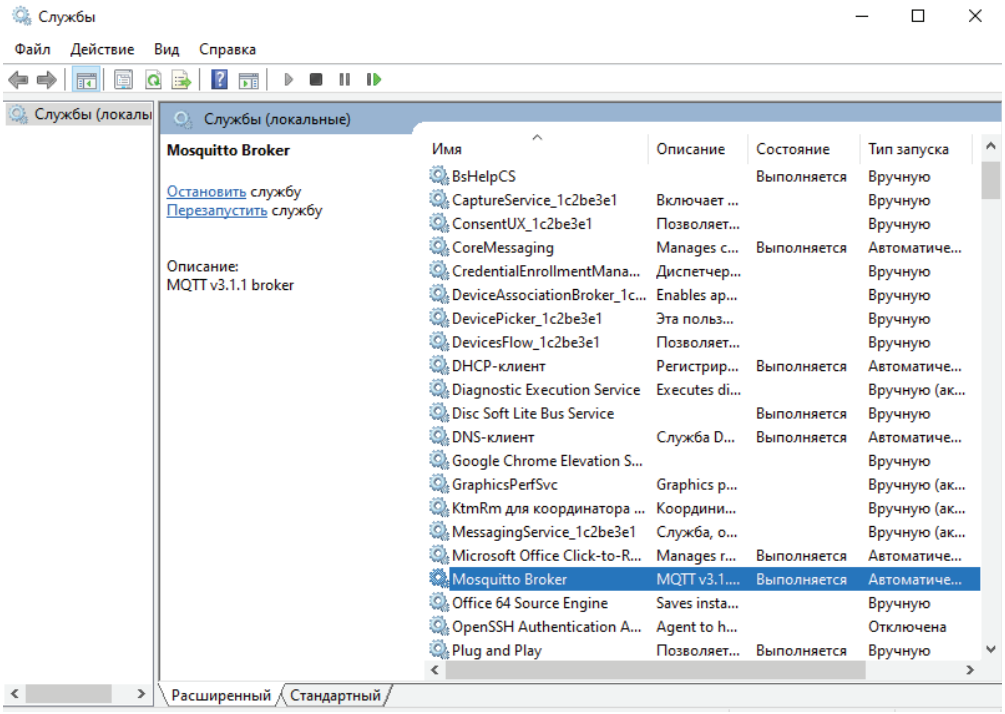


Рис. 19. Службы Windows: сервер MOSQUITTO

Задание 1. Вывод сообщения «Hello World»

1. Запустите сервер через терминал:
   mosquitto

Результат:

```
belvedersky — docker run -it --rm --name mosquitto -p 11883:1883 -v eclipse-mosquitto — docker — ssh home.local — 89×30
belvedersky@home:~ $ docker run  -it --rm \
--name mosquitto -p 11883:1883 -v \
`pwd`/config/mosquitto.conf:/mosquitto/config/mosquitto.conf \
eclipse-mosquitto
1645368999: mosquitto version 2.0.14 starting
1645368999: Config loaded from /mosquitto/config/mosquitto.conf.
1645368999: Opening ipv4 listen socket on port 1883.
1645368999: Opening ipv6 listen socket on port 1883.
1645368999: mosquitto version 2.0.14 running
```

Если запустить не получается из-за того, что порт уже используется, выберите другой порт через ключ -p.

Обычно по умолчанию сервер стартует на 1883 порту и запускается в режиме демона (daemon), то есть без участия пользователя. Порт

может быть занят просто потому, что MOSQUITTO уже стартовал в вашей системе. Можно это проверить, выведя список всех процессов.

В другом окне терминала введите команду подписки на топик:

```
mosquitto_sub -h "localhost" -t "mytopic" -q 1
```

В качестве хоста указан localhost (то есть компьютер), а в качестве топика — вымышленный mytopic.

Результат: программа будет ждать сообщений из этого топика.

Обязательно посмотрите в предыдущее окно, где стартовал сервер: вы увидите, что там добавилась новая информация.

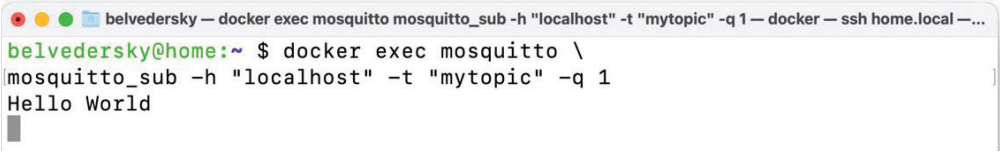
В третьем окне терминала введите команду, отправляющую сообщение — mosquitto_pub:

```
mosquitto_pub -h "localhost" -t "mytopic" -m "Hello World" -q 1
```

Параметры:

- h — адрес сервера;
- t — название топика, в котором публикуется сообщение (в данном примере это mytopic);
- m — текст сообщения;
- q — качество обслуживания.

В результате в окне с подпиской появится текст «Hello World»:



Задание 2. Подключение множества клиентов к одному серверу

Попробуйте передать данные по протоколу MQTT от одного компьютера в учебном классе к другому.

Выберите компьютер, который будет выполнять фукции сервера. Узнайте его IP-адрес. Запустите на нем сервер MQTT командой mosquitto.

Все остальные участники могут публиковать свои сообщения командой mosquitto_sub. Закончите работу, когда сообщения каждого из участников группы достигнут назначения.

При желании можно посмотреть, какие сообщения приходят на сервер, командой mosquitto_sub -h «192.168.1.15» -t «#»

Вы будете получать сообщения из всех топиков (символ # означает «все нижестоящие»).

Проверить работу сервера и увидеть вывод прямо в браузере можно с помощью расширения MQTTLens для браузера Chrome. Установите расширение, войдите и добавьте новое соединение. Для подключения к серверу достаточно знать IP-адрес, порт, логин и пароль сервера.

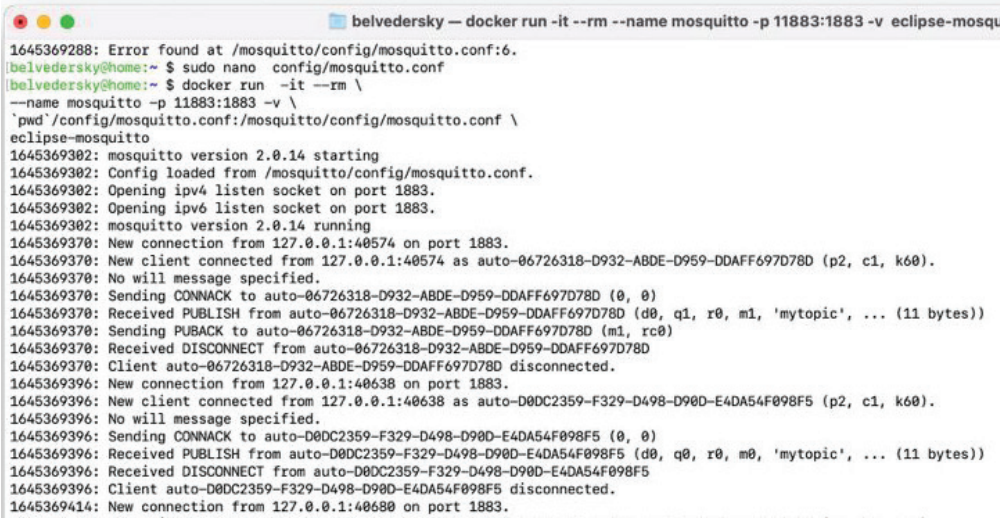
Hostname — введите IP-адрес компьютера, где запущен сервер, порт — 1883, введите логин с паролем. Хостом может быть как локальный, так и удаленный сервер. Попробуйте посмотреть в этом расширении вывод сообщений MQTT.

Задание 3. Выбор качества обслуживания (QoS)

Чтобы успешно работать с MQTT-сервером, нужно знать, по какой схеме происходит коммуникация. Проведите самостоятельные эксперименты. Запустите сервер MOSQUITTO в отладочном режиме:

mosquitto -v

Теперь, проделывая все те же действия, что и в задании 3, вы увидите большое количество промежуточных шагов:



```
1645369414: New client connected from 127.0.0.1:40680 as auto-130A3DED-D90D-4E17-38B8-B4AA82462E89 (p2, c1, k60).
1645369414: No will message specified.
1645369414: Sending CONNACK to auto-130A3DED-D90D-4E17-38B8-B4AA82462E89 (0, 0)
1645369414: Received PUBLISH from auto-130A3DED-D90D-4E17-38B8-B4AA82462E89 (d0, q1, r0, m1, 'mytopic', ... (11 bytes))
1645369414: Sending PUBACK to auto-130A3DED-D90D-4E17-38B8-B4AA82462E89 (m1, rc0)
1645369414: Received DISCONNECT from auto-130A3DED-D90D-4E17-38B8-B4AA82462E89
1645369414: Client auto-130A3DED-D90D-4E17-38B8-B4AA82462E89 disconnected.
1645369419: New connection from 127.0.0.1:40694 on port 1883.
1645369419: New client connected from 127.0.0.1:40694 as auto-55D91EB4-B37A-1FD6-70B3-D9E3CFB01055 (p2, c1, k60).
1645369419: No will message specified.
1645369419: Sending CONNACK to auto-55D91EB4-B37A-1FD6-70B3-D9E3CFB01055 (0, 0)
1645369419: Received PUBLISH from auto-55D91EB4-B37A-1FD6-70B3-D9E3CFB01055 (d0, q1, r0, m1, 'mytopic', ... (11 bytes))
1645369419: Sending PUBACK to auto-55D91EB4-B37A-1FD6-70B3-D9E3CFB01055 (m1, rc0)
1645369419: Received DISCONNECT from auto-55D91EB4-B37A-1FD6-70B3-D9E3CFB01055
1645369419: Client auto-55D91EB4-B37A-1FD6-70B3-D9E3CFB01055 disconnected.
1645369457: New connection from 127.0.0.1:40774 on port 1883.
1645369457: New client connected from 127.0.0.1:40774 as auto-86AFF572-FDE0-1BA5-E98F-1C8D98D1FC35 (p2, c1, k60).
1645369457: No will message specified.
1645369457: Sending CONNACK to auto-86AFF572-FDE0-1BA5-E98F-1C8D98D1FC35 (0, 0)
1645369457: Received PUBLISH from auto-86AFF572-FDE0-1BA5-E98F-1C8D98D1FC35 (d0, q1, r0, m1, 'mytopic', ... (11 bytes))
1645369457: Sending PUBACK to auto-86AFF572-FDE0-1BA5-E98F-1C8D98D1FC35 (m1, rc0)
1645369457: Received DISCONNECT from auto-86AFF572-FDE0-1BA5-E98F-1C8D98D1FC35
1645369457: Client auto-86AFF572-FDE0-1BA5-E98F-1C8D98D1FC35 disconnected.
1645369541: New connection from 127.0.0.1:40978 on port 1883.
1645369541: New client connected from 127.0.0.1:40978 as auto-47BA3ACA-CE19-6B97-A660-83107D5428D0 (p2, c1, k60).
1645369541: No will message specified.
1645369541: Sending CONNACK to auto-47BA3ACA-CE19-6B97-A660-83107D5428D0 (0, 0)
1645369541: Received SUBSCRIBE from auto-47BA3ACA-CE19-6B97-A660-83107D5428D0
1645369541:     mytopic (QoS 1)
```

Кроме того, вы можете увидеть промежуточные сообщения команд mosquitto_sub и mosquitto_pub, запустив их с ключом -d (debug).

Попробуйте отправить сообщения с разным параметром качества обслуживания (Quality of Service). Рассмотрите все случаи: когда QoS равняется 0, 1 или 2.

Задание 4. Взаимодействие с сервером на Python

В предыдущих заданиях работа с MQTT-сервером производилась в консоли при помощи команд mosquitto_sub и mosquitto_pub. Это удобно, когда задачу нужно выполнить один раз, например для считывания данных или отправки тестовых команд.

Для взаимодействия с MQTT-сервером существуют различные библиотеки. Рассмотрим библиотеку клиента MQTT Python Eclipse Paho. Подробную информацию можно получить в работе Г. К. Хайляра «Практическое программирование MQTT на Python»[5].

Библиотеку можно установить с помощью команды:

```
pip install paho-mqtt
```

Попробуйте запустить тестовую программу. Ее функционал — вывод на экран сообщений из определенного топика.

[5] Хайляр Г. К. Практическое программирование MQTT на Python. URL: http://onreader.mdl.ru/MQTTProgrammingWithPython/content/index.html#Preface (дата обращения: 30.09.2021).

```
import paho.mqtt.client as mqtt
# The callback for when the client receives a CONNACK response from
the server.
def on_connect(client, userdata, flags, rc):
    print(«Connected with result code «+str(rc))
  # Subscribing in on_connect() means that if we lose the connection and
   # reconnect then subscriptions will be renewed.
   client.subscribe(«$SYS/#»)

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
   print(msg.topic+» «+str(msg.payload))

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect(«iot.eclipse.org», 1883, 60)

# Blocking call that processes network traffic, dispatches callbacks and
# handles reconnecting.
# Other loop*() functions are available that give a threaded interface and a
# manual interface.
client.loop_forever()
```

Изучите текст примера и выделите для себя ключевые моменты:

1. Где и в какой момент происходит подписка на топик?
2. Где происходит подключение к серверу?
3. Когда обрабатывается входящее сообщение?

Модифицируйте программу так, чтобы она работала с вашим локальным сервером и подписывалась на тот топик, который вас интересует. Функция для отправки сообщения в топик имеет синтаксис

```
publish (topic, payload=None, qos=0, retain=False)
```

где topic — топик, в который идет сообщение; payload — текст сообщения; qos — качество обслуживания (Quality of Service); retain — новый параметр, если он имеет значение True, то сообщение будет «храниться до востребования», если клиент сейчас отключен, то после включения он все равно это сообщение получит, поскольку оно будет храниться

на сервере. В нашем случае информация может устареть, поэтому разумно установить его в False. Данный параметр хорош для передачи тех сообщений, которые не устаревают и должны храниться.

Задание 5. Публикация сообщений в конкретный топик и исследование работы с топиками MQTT

Добавьте в вашу программу на Python публикацию сообщений в некий топик (придумайте его сами). Можно сделать программный генератор сообщений: раз в несколько секунд отправляются данные о влажности и температуре, подобно тому как это делает реальное устройство. Бесконечный цикл loop_forever () в начале программы можно заменить на loop_start () и loop_stop () (прочитайте об этом в документации).

Используя локальный MQTT-сервер, выполните следующие действия:

1. Проверьте, чувствительны ли названия топиков к регистру.
2. Предположим, что у вас есть топик light, в нем — топики lamp1 и lamp2, и вы хотите послать сигнал всем лампочкам включиться. Можно ли послать сообщение сразу всем субтопикам определенного топика?
3. Проверьте, будет ли работать подписка на топик, если в конце названия топика добавить символ «/». А если в начале?
4. Выясните, что будет, если вызвать mosquitto_pub с ключом -l.
5. Изучите, что такое retain. Отправьте сообщение с ключом -r — важно, чтобы в этот момент еще не был запущен подписчик. Затем запустите подписку. Что произойдет? Что будет, если подписчик закрыт и затем запущен заново? Что, если оставить второе сообщение с тем же ключом? Чтобы вернуть все к прежнему состоянию, отправьте пустое сообщение с ключом -r.
6. Изучите, что такое «Последняя воля и завещание» (Last Will and Testament). Для этого при публикации не задавайте тело сообщения, но укажите параметры -- will-topic и --will-payload — название топика и собственно текст «завещания». В результате если вы в другом окне оформите подписку на этот топик и издатель (mosquitto_pub) незапланированно отключится, то подписчик (mosquitto_sub) получит от сервера «завещание». Обратите внимание, что команда mosquitto_pub в чистом виде отправляет сообщение и сразу отключается без аварийного за-

вершения, поэтому у вас не получится сделать это задание сразу. Подумайте, как смоделировать случай, когда издатель подключается на какое-то время и затем внезапно отключается.

2.6. Дополнительные материалы

Посмотрите видеолекцию «Инфраструктура транспортных сетей», читает Роман Андреев (СПбГУТ). Лекция включает обсуждение топологий 2G/3G/4G-сетей, основных производителей оборудования, функциональных блоков сети, а также прохождение аутентификации и вызовов в сетях связи.

IT Академия Samsung. Инфраструктура транспортных сетей : [видеолекция] // YouTube.com : [сайт]. URL: https://youtu.be/Xqn4nOjbzIo (дата обращения: 30.09.2021).

Ознакомьтесь с видеолекцией «Технологии беспроводной связи», читают Юрий и Ксения Сизовы (RedBees), Антон Куропятник (WoodenShark).

В этой лекции Юрий, Антон и Ксения остановятся на первых двух компонентах сети IoT — оконечных устройствах и каналах передачи данных до базовой станции / граничного роутера; кратко произведут обзор технологий и протоколов, применяемых в IoT; расскажут подробнее об их характеристиках, плюсах и минусах, а также основных трендах использования.

IT Академия Samsung. Как выбрать технологию IoT : [видеолекция] // YouTube.com : [сайт]. URL: https://youtu.be/Oo-45qVxX2I (дата обращения: 30.09.2021).

Контрольные вопросы и задания

1. Опишите сетевое взаимодействие систем интернета вещей с точки зрения модели OSI.
2. Какие протоколы в модели TCP/IP обеспечивают функциональность уровней для систем IoT?
3. Опишите функции 12-уровневой логической модели для систем IoT.
4. Сравните 7-уровневую модель OSI с 12-уровневой моделью построения систем IoT.
5. Зачем в стеке TCP/IP используется три типа адресов устройства? Охарактеризуйте всех типы адресов.
6. Опишите информационную составляющую IP-адреса.
7. Какие проблемы возникают при использовании протокола IPv6 в системах интернета вещей?
8. Какие топологии сетей используются в IoT и какие сети работают на каждой топологии?
9. Перечислите критерии, по которым можно сравнить сети как технологии.
10. В чем преимущества использования протокола Modbus? Какие можно выделить недостатки?
11. Сколько устройств можно подключить к одному приемнику при использовании узкополосной модуляции на одном канале в LPWAN?
12. На какой технологии можно создать полностью закрытую сеть передачи данных?
13. Опишите принцип работы Bluetooth. Нарисуйте топологию сети Bluetooth Mesh.
14. Какие диапазоны частот для радиосвязи доступны в России? Каковы преимущества диапазона Wi-Fi 5 ГГц?
15. Какую функцию выполняет координатор сети в ZigBee?
16. Как организована адресация в сети 6LoWPAN?
17. Нарисуйте структуру сетевой технологии 6LoWPAN.
18. От каких параметров зависит выбор беспроводной технологии?
19. Опишите основную идею протокола MQTT.
20. С какими проблемами безопасности сталкиваются разработчики систем интернета вещей?

Глава 3. Энергосберегающие технологии

3.1. Энергосбережение на уровне микроконтроллера

Энергопотребление IoT-устройства зависит от многих факторов. На потребление энергии может влиять выбранная сетевая технология, микроконтроллерная плата, интерфейс передачи данных и, конечно, программное обеспечение. Разработчики всегда стоят перед выбором между качеством (надежностью оборудования) и эффективностью. Эффективность может быть выражена либо стоимостью, либо временем бесперебойного функционирования устройства. В оптимальном варианте программное обеспечение должно использовать меньше памяти, использовать энергоэффективные режимы и выполняться на более дешевых контроллерах.

Существует несколько программных методов, позволяющих снизить энергопотребление устройств. Подробно об этом можно прочитать в работе П. Бобкова «Программные методы управления энергопотреблением устройств»[6].

Большинство современных микропроцессоров и микроконтроллеров имеют специальные режимы работы, предназначенные для снижения энергопотребления. Самым распространенным из них является режим ожидания (idle mode). В этом режиме ядро ОС микропроцессора останавливается, а все периферийные устройства продолжают работать. Каждый раз, когда операционная система обнаруживает, что все процессы заблокированы в ожидании прерывания, она долж-

[6] Бобков П. Программные методы управления энергопотреблением устройств // Chipenable.ru : [сайт]. URL: https://chipenable.ru/index.php/how-connection/84-programmnye-metody-upravleniya-energopotrebleniem-ustroystv.html (дата обращения: 30.09.2021).

на перевести микропроцессор в режим ожидания, чтобы уменьшить энергопотребление. Переключение в режим ожидания и выход из него требуют небольших временных затрат и могут происходить несколько раз в миллисекунду. Поскольку любое прерывание может вывести микропроцессор из режима ожидания, использование этого режима позволяет программному обеспечению осуществлять интеллектуальное ожидание событий в системе. Для максимальной производительности требуется качественная разработка прошивки МК. Если для снятия показаний внешних устройств использовать циклический опрос, то микропроцессор будет выполнять много «холостого» кода и тратить впустую энергию автономного источника питания. Более удачное решение состоит в том, чтобы использовать внешнее прерывание, сигнализирующее об изменениях. В однопоточной программе это позволит перевести микропроцессор в режим ожидания. В этом режиме микропроцессор потребляет минимум энергии до возникновения прерывания, при котором он «проснется» и продолжит выполнение кода. Если событие не может быть непосредственно привязано к внешнему прерыванию, переключение микропроцессора в режим ожидания и выход из него можно выполнять по сигналу системного таймера. Этот вариант предпочтительнее циклического опроса.

Большинство современных смартфонов работает на микропроцессорах и операционных системах, использующих режим ожидания. Фактически большая часть этих устройств переключается в режим ожидания и выходит из него многократно в течение секунды; они выходят из этого режима всякий раз, когда кто-то дотрагивается до сенсорного экрана, нажимает клавишу или когда возникает какое-либо событие.

Динамическая тактовая частота и регулировка напряжения питания являются еще одним способом уменьшения энергопотребления микропроцессора. Этот подход к управлению энергопотреблением основан на следующем правиле: энергия, расходуемая микропроцессором, прямо пропорциональна его тактовой частоте и квадрату напряжения питания. Микропроцессоры с функцией динамического сокращения тактовой частоты являются первым шагом в экономии электроэнергии: уменьшите тактовую частоту на половину, и потребляемая мощность сократится пропорционально. Однако при использовании только этого метода реализовать эффективные стратегии энергопотребления довольно сложно, поскольку исполняемый код при пониженной тактовой частоте потребует для своего завершения большего времени. Динамиче-

ское уменьшение напряжения — совсем другое дело. Все большее число микропроцессоров позволяет с понижением тактовой частоты уменьшать и напряжение питания, а это приводит к снижению энергопотребления даже в тех случаях, когда одно только сокращение тактовой частоты ничего не дает. Фактически, пока микропроцессор не загружен, частота и напряжение могут непрерывно уменьшаться. Работа все еще происходит, но в целом расход энергии ниже. Этот подход можно усовершенствовать, если учесть, что не все процессы (потоки) одинаково требовательны к пропускной способности микропроцессора. Процессы, которые эффективно используют пропускную способность, при пониженной тактовой частоте будут выполняться дольше; эти процессы используют каждый цикл, выделенный им. С другой стороны, процессы с ограничениями ввода-вывода тоже используют все циклы процессора, выделенные им, но при понижении тактовой частоты требуют для своего завершения то же самое время.

Разработчик должен понимать, в каких случаях возможно уменьшить тактовую частоту и напряжение питания, не оказывая особого влияния на производительность системы.

Уменьшение потребляемой мощности в режиме ожидания требует анализа аппаратных средств системы. Например, большинство систем с батарейным питанием в режиме ожидания продолжает питать порты ввода-вывода общего назначения. В состоянии входа выводы портов могут использоваться для пробуждения устройства; в состоянии выхода они могут использоваться для конфигурирования внешних периферийных устройств. Внимательное рассмотрение того, как эти выводы сконфигурированы, может оказать значительное влияние на потребляемую мощность в режиме ожидания. Например, если сконфигурировать вывод микропроцессора на выход и «подтянуть» его к плюсу питания, то установка на этом выводе логического нуля в режиме ожидания вызовет протекание тока через нагрузочный резистор. Если запрограммировать вывод на вход и он не будет ни к чему подтянут, потенциал на нем будет «плавать», что может привести к неопределенности логического уровня и, соответственно, повышению потребляемой мощности. Важно проанализировать эти ситуации и конфигурировать порты ввода-вывода правильным образом.

Все микроконтроллеры семейства STM выполнены по CMOS-технологии. В устройствах такого типа полное энергопотребление является суммой статического и динамического потреблений.

Статическое потребление определяется энергией, затрачиваемой на поляризацию транзисторов, а также токами утечки транзисторов. Именно оно определяет потребление микросхемы в режиме, когда тактовый генератор выключен.

Динамическое потребление определяется энергией, затрачиваемой на перезарядку паразитной емкости подзатворного слоя при переключении транзисторов. Оно определяется величиной емкости, напряжением питания транзисторов, а также частотой их переключения. Динамическое потребление является определяющим в режимах работы микроконтроллера, когда тактовый генератор запущен. В этом случае статическое потребление пренебрежимо мало.

Микроконтроллеры STM8L выполнены по особой технологии, обеспечивающей ультранизкие значения токов утечки. Таким образом, статическое потребление микроконтроллеров очень мало, а динамическое потребление может быть снижено следующими методами:

1) снижение напряжения питания микроконтроллера — влияет на динамические потери; для микроконтроллеров серии STM8L данный метод имеет малое значение, т. к. все линейки STM8L снабжены встроенными регуляторами напряжения, и выигрыш получается незначительным;
2) снижение частоты тактирования ЦПУ для задач, не требующих высокой производительности;
3) отключение питания и тактирования неиспользуемых модулей периферии — один из самых эффективных способов снижения энергопотребления;
4) использование режимов пониженного энергопотребления;
5) использование встроенных резисторов подтяжки портов и отключение их в начале процедуры обработки прерывания при работе с внешними сигналами, требующими подвязки к питанию для задания неактивного уровня.

3.2. Энергоэффективные сети

Рассмотрим несколько энергоэффективных технологий связи. Выбор технологии зависит от конкретной системы IoT и ее функционала.

GPRS — технология пакетной передачи данных в рамках сети сотовой связи GSM. При использовании системы GPRS информация собирается в пакеты и передается в эфир, они заполняют «пустоты» в сети, которые всегда есть в промежутках между разговорами абонентов. Этап установления соединения занимает несколько секунд, и у абонента появляется возможность передавать данные целиком, при этом более эффективно используются ресурсы сети. Недостатки технологии заключаются в первоначальной цели разработки указанного стандарта: сотовая связь была создана с расчетом на большое количество функций — отправку СМС, голосовую и факсимильную почту. Данные функции для передачи данных от IoT-устройств остаются избыточными. Кроме того, для GPRS необходимо постоянное подключение к сети Интернет, что не всегда возможно для IoT-устройств.

NB-IoT — стандарт, который был разработан на базе существующих стандартов мобильной связи как более адаптированная версия упомянутого GSM. Для работы NB-IoT в рамках сети сотовой связи выделяются узкие полосы частот на существующих базовых станциях. Устройства NB-IoT, подключающиеся к сети, проходят процедуру регистрации в сети, аналогичную обычным мобильным телефонам, после чего могут осуществлять обмен данными с сетью. NB-IoT предназначен для передачи коротких сообщений, и от него не требуется передача аудиовидеоконтента, больших файлов данных. Устройства в стандарте NB-IoT могут работать до 10 лет от одной обычной батарейки.

Самое важное в NB-IoT — возможность работы при более низких уровнях сигнала и при высоком уровне шумов. На физическом уровне у стандарта есть особенности:

— общая полоса для NB-IoT ограничена шириной в 180 кГц;
— радиотракт пользовательского устройства имеет всего одну антенну, приемник и передатчик;
— передача и прием разнесены по времени.

Bluetooth Low Energy (BLE) — это стандарт, специально разработанный для устройств с ограниченным источником автономной энергии, нуждающихся в отправке данных на протяжении нескольких дней или недель без подзарядки. Технология актуальна для эпизодической передачи небольших объемов данных на низких скоростях. BLE отличается низким энергопотреблением за счет того, что периферийное устройство может долгое время находиться в спящем режиме и отправляет данные в объеме полезной нагрузки. Во многих устройствах

интернета вещей необходимо передавать небольшие данные, поэтому BLE стал наиболее часто используемым протоколом связи.

Некоторые характеристики BLE:

— используемый частотный диапазон 2,400–2,4835 ГГц, диапазон поделен на 40 каналов по 2 МГц каждый;
— максимальная скорость передачи данных до 2 Мбит/с;
— дальность передачи от 10 до 30 метров — зависит от физического окружения, а также используемого режима передачи: например, в режиме большой дальности скорость передачи будет ниже, чем в высокоскоростном режиме;
— потребление BLE-трансивера во время передачи данных не превышает 15 мА;
— существует несколько возможных для реализации уровней обеспечения безопасности.

Устройства, использующие BLE, могут быть как двухрежимными, совместимыми с классическими Bluetooth-устройствами, так и однорежимными. Однорежимные модули BLE используются в мелкой электронике и аксессуарах типа измерителей пульса или бесконтактных ключей. Двухрежимными модулями BLE оснащаются смартфоны, планшетные ПК, ноутбуки.

Множество задач может решаться, когда в радиусе двухуровневого модуля определяются одноуровневые BLE-приборы. К этим приборам относятся приборы-сигнализаторы, которые уведомляют владельца об удалении от его персональных вещей, оснащенных BLE-модулем. Брелок с BLE можно использовать в качестве маячка для ребенка, чтобы не потерять его в людных местах. Устойчивая работа и низкое энергопотребление протокола BLE позволяют рассматривать его в качестве замены RFID-меток, а также в системах умных домов. Работа приборов через BLE позволяет открывать удаленно двери, ворота и приводить в действие прочие механизмы, подолгу не меняя аккумулятор в беспроводном и компактном пульте управления. Внедрение в смартфон BLE-модуля позволяет на приличном удалении через сопряженные каналы управлять любыми приборами умного дома.

Технология BLE предоставляет функциональные возможности, которые позволяют создавать приложения для удаленного управления, медицинского наблюдения, спортивных датчиков и других интеллектуальных устройств.

Рассмотрим сети LPWAN (Low-Power Wide-Area Network). Переходя к сетям, следует понимать, что под ними подразумевается несколь-

ко технологий, благодаря которым можно организовать передачу данных. С точки зрения развертывания одного из видов LPWAN-сети, кроме технических особенностей, также имеет значение организация бизнес-модели. Владельцы сети хотят контролировать не только физический уровень, но и производство (реализацию) конечных устройств и базовых станций. Например, «Стриж» — IoT-платформа на базе беспроводных LPWAN-сетей — является полностью проприетарной системой для онлайн-учета ресурсов ЖКХ, сбора телеметрии в промышленности и других отраслях.

LoRaWAN — это сетевой протокол с низким потреблением энергии, предназначенный для беспроводного подключения «вещей» к интернету в региональных, национальных или глобальных сетях.

Типовая беспроводная сеть LoRaWAN представляет собой совокупность шлюзов (gateways), пересылающих сообщения между оконечными устройствами (end-devices) и центральным сервером (network server), и характеризуется топологией «звезда» (рис. 20).

Топология «звезда» — это топология, в которой каждый узел в сети подключен к одному центральному коммутатору. Каждое устройство в сети напрямую связано с коммутатором и косвенно связано с любым другим узлом.

Шлюзы называют также концентраторами (concentrators) или базовыми станциями (base stations).

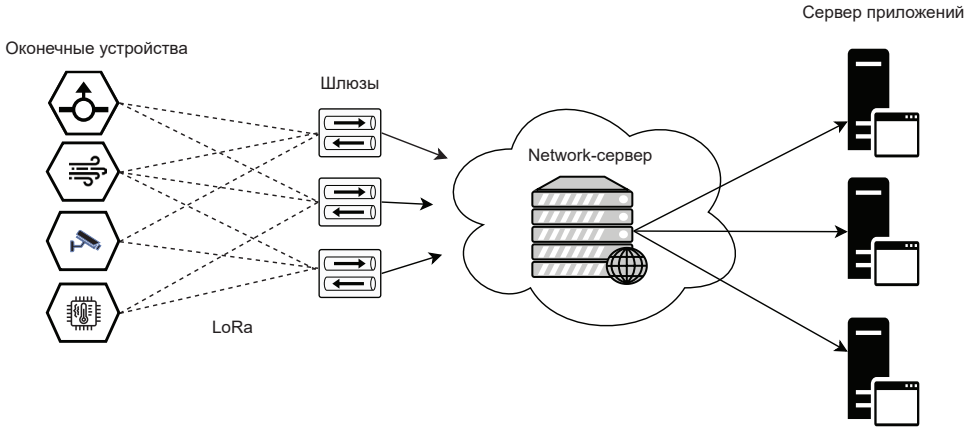


Рис. 20. Архитектура LoRaWAN

Связь между шлюзами и центральным сервером осуществляется через стандартные IP-соединения, а между шлюзами и оконечными устройствами — через беспроводные соединения, использующие широкополосную модуляцию LoRa или FSK.

Модуляция LoRa была разработана компанией Semtech и предназначена для низкоскоростной беспроводной передачи данных на расстояние до нескольких километров в безлицензионных диапазонах частот (в России 864,0–865,0 МГц и 868,7–869,2 МГц).

Связь между шлюзами и оконечными устройствами является двусторонней, но чаще всего основной объем данных передается от оконечных устройств к шлюзам. Технология LoRa обеспечивает скорость передачи в беспроводном канале от 0,3 до 50 кбит/с. Для разделения каналов используется как набор частотных каналов, так и скоростей передачи (Data Rates).

Для оптимизации работы системы используется адаптивное изменение скорости передачи — ADR (Adaptive Data Rate). Сетевой сервер оценивает качество сигнала, принимаемого от оконечного устройства, и может управлять как скоростью передачи, так и мощностью передатчика этого устройства.

Оконечное устройство может передавать данные на любом доступном канале и любой скорости передачи с учетом следующих факторов:

1. При каждой передаче частотный канал выбирается оконечным устройством из списка доступных каналов случайным образом.
2. Перед началом передачи оконечное устройство должно убедиться в том, что канал свободен (Listen Before Talk, LBT). Если канал занят, то устройство переходит на другой канал и повторяет процедуру LBT.
3. Оконечное устройство должно принимать во внимание ограничения местных регулирующих органов относительно процента времени, в течение которого устройство может занимать частотный канал.

Основные преимущества беспроводных сетей LoRaWAN, обусловленные использованием широкополосной модуляции LoRa и безлицензионных диапазонов частот:

— совместимость с существующими сетями и технологиями беспроводной передачи данных;
— высокая помехоустойчивость;
— способность обслуживать десятки тысяч устройств;

— большая зона охвата и малое энергопотребление оконечных устройств.

Беспроводные сети LoRaWAN применяются для следующих задач:

— считывание показаний счетчиков;
— мониторинг автотранспорта и грузов на определенной территории;
— контроль состояния контейнеров или емкостей на производстве;
— мониторинг производственного оборудования;
— мониторинг доступности парковочных мест;
— управление освещением;
— мониторинг погоды;
— контроль наличия вредных веществ в атмосфере;
— пожарная или охранная сигнализация;
— автоматизация зданий.

Спецификация сети LoRaWAN определяет три класса оконечных устройств: *A*, *B* и *C*, отличающихся друг от друга режимами приема.

Класс A (поддерживается всеми устройствами)

Устройства класса *A* после каждой передачи открывают два коротких временных окна на прием — RX1 и RX2 (рис. 21).

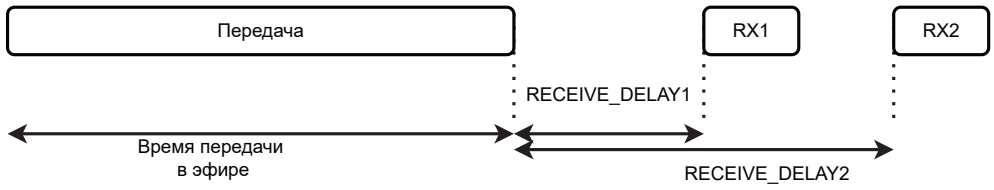


Рис. 21. Временная диаграмма процесса передачи устройств класса *A*

Интервалы от конца передачи до открытия первого и второго временных окон могут конфигурироваться, но должны быть одинаковыми для всех устройств в данной сети (RECEIVE_DELAY1, RECEIVE_DELAY2). Для европейского диапазона 868 МГц рекомендованное значение RECEIVE_DELAY1 составляет 1 секунду. Значение RECEIVE_DELAY2 должно быть на 1 секунду больше.

Устройства класса *A* являются самыми низкопотребляющими, но для передачи сообщения от сервера к оконечному устройству необходимо дождаться следующего исходящего сообщения от этого устройства.

Класс B (Beacon)

Кроме окон приема, определенных для устройств класса *A*, устройства класса *B* открывают дополнительные окна приема по расписанию. Для синхронизации времени открытия дополнительных окон приема шлюзы излучают маячки (beacons). Все шлюзы, входящие в состав одной сети, должны излучать маячки одновременно. Маячок содержит идентификатор сети и метку времени (UTC).

Использование класса *B* гарантирует, что при опросе оконечных устройств задержка отклика не будет превышать определенную величину, определяемую периодом маячков.

Класс C (Continuous)

Устройства класса *C* находятся в режиме приема практически все время за исключением промежутков, когда они передают сообщения (рис. 22).

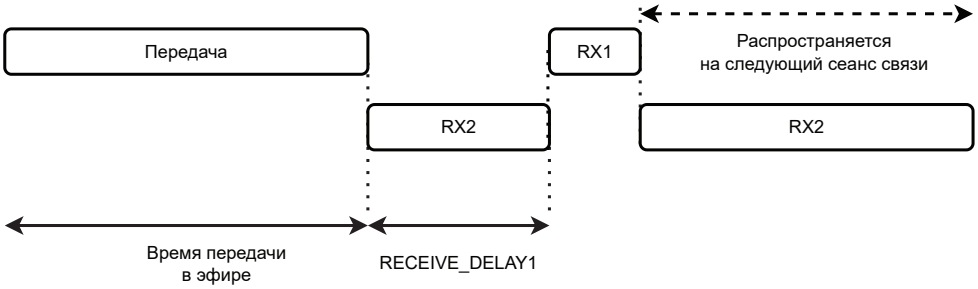


Рис. 22. Временная диаграмма процесса передачи устройств класса *C*

Класс *C* может применяться там, где не критично потребление энергии (например, в приборах учета) или где необходимо опрашивать оконечные устройства в произвольные моменты времени.

3.3. Оценка энергопотребления устройств интернета вещей

Технологии интернета вещей оказывают комплексное влияние на развитие сферы производства и потребления энергии. Их внедрение призвано способствовать экономии электроэнергии: так, умный кондиционер будет автоматически отключен, если жителей нет в по-

мещении. Превращение уже существующих устройств в умные путем добавления новых датчиков, электроники и дисплеев не вызывает значительного увеличения объемов энергопотребления, однако их использование предполагает энергоемкие процессы передачи больших объемов данных и их удаленной обработки. Кроме того, происходит стремительное наращивание производства и потребления устройств интернета вещей.

Возможно, в будущем благодаря интернету вещей в широком доступе появится умная одежда со встроенными датчиками, умная обувь с комфортной температурой, с шагомером, GPS и др. Однако встает вопрос: а от чего, собственно, должны питаться все эти полезные гаджеты? В идеале такой источник питания должен быть легким и миниатюрным, способным принимать любую форму и вид, умеющим заряжаться от любых типов энергии в окружающем пространстве и не требующим регулярной замены. Идеал пока недостижим, но первые шаги в этом направлении уже сделаны. Поскольку для датчиков и простых микрокомпьютеров не требуются источники большой мощности, с ними можно использовать устройства, способные генерировать энергию буквально из ничего, собирая ее по крупицам практически из воздуха, как, например, ветряные генераторы или солнечные батареи.

Эта идея положена в основу концепции Energy Harvesting — ее название пока не имеет общепринятого русского перевода, а по смыслу оно примерно означает «сбор энергетического урожая». Концепция заключается в сборе разнообразной энергии из окружающей среды и преобразовании ее в электрическую для питания автономных миниатюрных устройств. Источником энергии могут быть любые естественные природные и физические процессы и явления — от солнечного света до любых механических колебаний.

Создаваемые в рамках концепции Energy Harvesting устройства должны быть способны как генерировать, так и сохранять электрическую энергию — они смогут заменить тяжелые и громоздкие аккумуляторные батареи там, где не требуются большая мощность и высокое напряжение. В результате мы получим вшитые микрогаджеты, датчики в одежде и обуви с практически вечным питанием, способные работать автономно до своего физического износа.

Самый простой для преобразования в электричество вид внешних воздействий — это механические колебания и вибрации. Чаще всего для конвертации таких колебаний в электроэнергию применяют-

ся пьезоэлектрические материалы. Такие материалы используются, в частности, в так называемых микроэлектромеханических системах (МЭМС или MEMS), представляющих собой гибридные устройства на кремниевой подложке, в которых объединены микромеханические и микроэлектронные компоненты. МЭМС-чипы легко распаиваются на печатной плате и без проблем интегрируются в любую электронную схему.

Британская компания Perpetuum показала на выставке Energy Harvesting & Storage Europe вибрационный «сборщик энергии» Vibration Energy Harvester (VEH). Это беспроводной датчик, предназначенный для установки на вращающиеся детали, например на подшипники колес поездов. Датчик VEH выполняет одновременно три функции: он измеряет температуру, передает полученные сведения по беспроводной связи и вырабатывает необходимую для всего этого электроэнергию из механических колебаний. В Perpetuum уже опробовали VEH в действии, и оказалось, что это простое устройство, не требующее никакого обслуживания, чрезвычайно полезно именно для установки в колесах железнодорожных вагонов, поскольку оно способно мгновенно фиксировать критическое повышение температуры в подшипниках и тем самым предотвращать масштабный дорогостоящий ремонт.

Другой распространенный вид энергии — световая, и в частности солнечная. Фотовольтаика уже не первое десятилетие успешно применяется как в самых миниатюрных устройствах, например в микрокалькуляторах, так и в огромных космических станциях. Сегодня же ученые пытаются реализовать достижения фотовольтаики в довольно неожиданных предметах — например, в ткани. Создаются волокна, которые могут улавливать солнечную энергию, преобразуя ее в электрическую. В перспективе из них можно будет изготавливать ткани, работающие как единая система по выработке и хранению электроэнергии. Ведутся работы по созданию «электроткани», способной вырабатывать порядка 10 Вт энергии на квадратный метр. Если добиться таких показателей, то потенциальные способы использования ткани будут намного шире, чем у одежды со встроенными датчиками и автономными источниками питания. Ткань площадью в 100 квадратных метров даст уже целый киловатт электроэнергии. А это значит, что ее можно использовать в качестве материала для изготовления палаток, тентов или солнечных козырьков, которые будут способны выработать достаточно энергии для внутреннего освещения. Кроме

того, такую ткань можно сбрасывать с воздуха людям, терпящим бедствие, в то время как классические солнечные панели требуют бережного обращения.

Выделяют четыре механизма влияния интернета вещей на энергопотребление:

— прямое (локальное) воздействие компонентов интернета вещей на энергетические системы;
— удаленное потребление энергии инфраструктурой интернета вещей;
— потребление энергии в процессе производства устройств интернета вещей;
— косвенное воздействие интернета вещей на энергетические системы через изменение потребительского поведения, связанное с увеличением количества используемых устройств и объема потребляемых услуг.

При оценке прямого воздействия устройств интернета вещей на энергетические системы целесообразно опираться на данные об энергопотреблении бытовой электроники. По аналогии с бытовой электроникой системы интернета вещей обладают потенциалом для снижения энергопотребления при фиксированном объеме услуг за счет более разумного использования ресурсов. Например, при помощи собираемых устройствами интернета вещей данных можно оптимизировать промышленное энергопотребление. На базе технологий интернета вещей в транспортном секторе внедряются адаптивные системы управления движением, снижающие проблему пробок на дорогах, что способствует более эффективному расходованию топлива. В умных домах и офисах с помощью устройств интернета вещей оптимизируется управление температурой и освещением, обеспечивается более эффективное расходование электричества, внедряются системы на базе возобновляемых источников энергии.

В настоящее время сложно прогнозировать долгосрочные последствия внедрения технологий интернета вещей с точки зрения их воздействия на энергопотребление и оценить, какие именно устройства и приложения получат широкое распространение и как это повлияет на потребительское поведение.

3.4. Практические задания

Занятие № 8

Цель занятия: применить базовые технологии интернета вещей для реализации системы автополива растений.

Для выполнения работы понадобится следующее оборудование:

— контроллер STM32 Nucleo;
— плата расширения Troyka Slot Shield;
— Wi-Fi (Troyka-модуль);
— датчик влажности почвы;
— мини-реле (Troyka-модуль) или силовой ключ N-Channel v3 (Troyka-модуль);
— погружная помпа с трубкой;
— гнездо питания 2,1 мм с клеммником и штекер питания 2,1 мм с клеммником;
— импульсный блок питания 5–12 В;
— перемычки или трехпроводной шлейф «мама — мама» — 4 шт.;
— микрокомпьютер Raspberry Pi, настроенный в качестве точки доступа Wi-Fi и MQTT-сервера.

Техническое задание

Система должна получать значения с датчика влажности почвы. Показания публикуются на MQTT-сервере в топике /class/stand<id>/humidity, где <id> — номер кейса с лабораторным оборудованием.

Полив выполняется при низком уровне влажности (показания датчика влажности менее 15) и сопровождается отправкой сообщения «watering» в топик /class/stand<id>/pump, где <id> — номер кейса с лабораторным оборудованием. Реализуйте возможность запустить помпу принудительно (выполнить полив) при получении сообщения «do» в соответствующем топике.

Подача питания на погружную помпу производится с использованием модуля мини-реле или силового ключа.

Методические указания

Подключите оборудование согласно схеме на рис. 23.

Затем подключите плату ESP8266 к STM32Nucleo. Соединение должно проходить по четырем выводам: два коммуникационных вы-

вода TX и RX, питание (V) и «земля» (GND). Со стороны микроконтроллерной платы — выводы RX и TX (крест-накрест), питание 3,3 В и «земля» (GND).

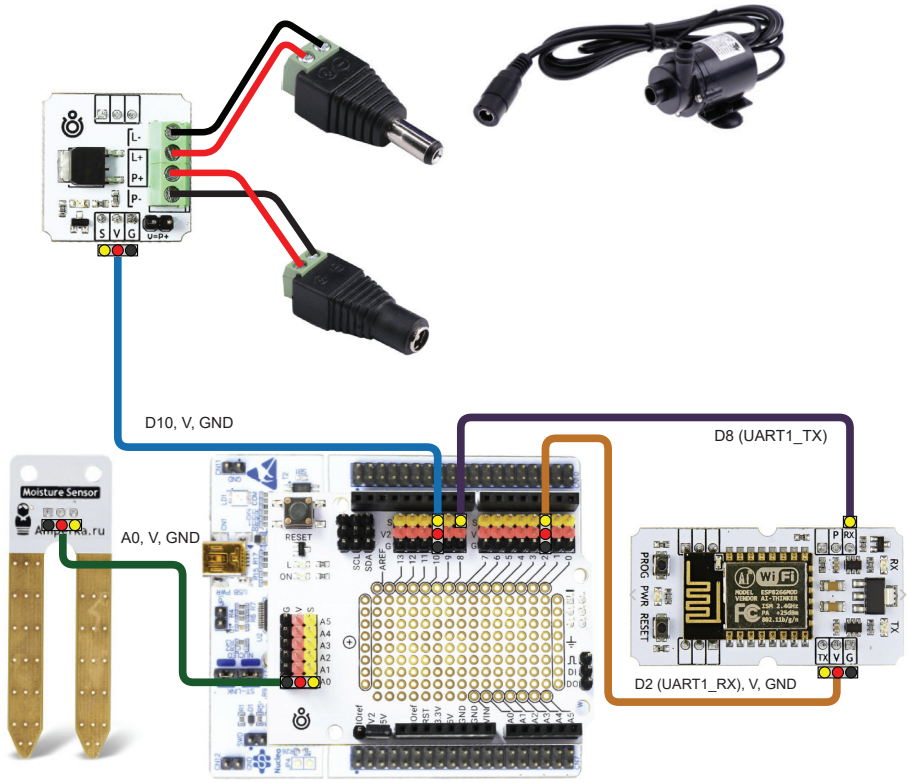


Рис. 23. Схема подключения оборудования

Протестируйте соединение. STM32Nucleo будет выступать в роли моста. Подключитесь к одному из интерфейсов UART — можно выбрать любой свободный, например UART1 (выводы D8 и D2).

Загрузите на плату скетч:

```
#include «mbed.h»
RawSerial pc (USBTX, USBRX);
RawSerial dev (D8, D2);
DigitalOut led1 (LED1);
void dev_recv ()
{
```

```
    led1 =! led1;
    while (dev.readable ()) {
    pc.putc (dev.getc ());
    }
}
void pc_recv ()
{
    while (pc.readable ()) {
    dev.putc (pc.getc ());
    }
}
int main () {
    pc.baud (115200);
    dev.baud (115200);
    pc.attach (&pc_recv, Serial:: RxIrq);
    dev.attach (&dev_recv, Serial:: RxIrq);
    while (1) {
    sleep ();
    }
}
```

Код перенаправляет данные от микроконтроллера ESP через микроконтроллер STM на компьютер и обратно, создавая прозрачный «мост». Выводы D2 и D8 MK STM используются для обмена данными с MK ESP, а выводы D1 и D0 (STM) — для обмена данными с компьютером.

Далее проверьте, что модуль Wi-Fi имеет необходимую прошивку. Для этого необходимо открыть терминал, настроить скорость 115200 бод и отправить команду AT и AT + GMR. После ввода каждой команды — нажать комбинацию Ctrl + M и Ctrl + J.

Создайте проект в среде Mbed. При работе в Mbed Studio используйте шаблон Mbed OS5 — mbed-os-example-wifi. Если вы работаете через браузерную версию, создайте пустой проект и импортируйте программу mbed-os-example-wifi.

Настройте параметры подключения по Wi-Fi. Для этого отредактируйте файл mbed_app.json. Файл должен содержать код:

```
{
"config": {
    "wifi-ssid": {
            "help": "WiFi SSID",
            "value": "\"myssid\""
},
    "wifi-password": {
            "help": "WiFi Password",
            "value": "\"mypassword\""},
    "tx": {
            "help": "TX pin for serial connection",
            "value": "D8"
            },
    "rx": {
            "help": "RX pin for serial connection",
            "value": "D2"
    },
    "socket-bufsize": {
            "help": "Max socket data heap usage",
            "value": "1024"}
            },
    "target_overrides": {
    "*": {
            "target.network-default-interface-type": "WIFI",
            "esp8266.tx": "D8",
            "esp8266.rx": "D2",
            "esp8266.socket-bufsize": "1024",
            "esp8266.debug": false,
            "platform.stdio-convert-newlines": true,
            «esp8266.provide-default»: true
            }
    }
}
```

Конфигурационные параметры можно посмотреть в драйвере[7] ESP8266.

[7] URL: https://github.com/ARMmbed/mbed-os/blob/master/connectivity/drivers/wifi/esp8266-driver/mbed_lib.json.

Настройте параметры точки доступа:

— Wi-Fi SSID: SmartAudIOTorium;
— Wi-Fi Password: Raspberry.

Проверьте подключение к точке доступа Wi-Fi. Для этого используйте пример кода, приведенный ниже. В этом примере создается MQTT-клиент, который подписывается на топик mbed-sample и сам посылает туда же сообщения. Callback-функция срабатывает при получении нового сообщения и увеличивает значение счетчика. Демосценарий: клиент посылает в свой же топик два сообщения и ждет третьего извне. Вы отправляете его сами, после чего работа программы завершается.

```
#define MQTTCLIENT_QOS2 1

#include "MQTTmbed.h"
#include "MQTTClientMbedOs.h"
int arrivedcount = 0;
void messageArrived (MQTT:: MessageData& md)
{
    MQTT:: Message &message = md.message;
  printf ("Message arrived: qos %d, retained %d, dup %d, packetid %d\r\n",
    message.qos, message.retained, message.dup, message.id);
  printf ("Payload %.*s\r\n", message.payloadlen, (char*)message.payload);
    ++arrivedcount;
}
void mqtt_demo (NetworkInterface *net)
{
    float version = 0.6;
    char* topic = "mbed-sample";

    TCPSocket socket;
    MQTTClient client (&socket);
    SocketAddress a;
    char* hostname = "192.168.0.11";
    net->gethostbyname (hostname, &a);
    int port = 1883;
    a.set_port (port);
    printf ("Connecting to %s: %d\r\n", hostname, port);
```

```
socket.open (net);
printf ("Opened socket\n\r");
int rc = socket.connect (a);
if (rc!= 0)
   printf ("rc from TCP connect is %d\r\n", rc);
printf ("Connected socket\n\r");

MQTTPacket_connectData data = MQTTPacket_connectData_initializer;
data.MQTTVersion = 3;
data.clientID.cstring = "mbed-sample";
data.username.cstring = "testuser";
data.password.cstring = "testpassword";
if ((rc = client.connect (data))!= 0)
   printf ("rc from MQTT connect is %d\r\n", rc);

if ((rc = client.subscribe (topic, MQTT:: QOS2, messageArrived))!= 0)
   printf ("rc from MQTT subscribe is %d\r\n", rc);

MQTT:: Message message;

//QoS 0
char buf [100];
sprintf (buf, "Hello World! QoS 0 message from app version %f\r\n", version);
message.qos = MQTT:: QOS0;
message.retained = false;
message.dup = false;
message.payload = (void*)buf;
message.payloadlen = strlen (buf)+1;
rc = client.publish (topic, message);
while (arrivedcount < 1)
   client.yield (100);

//QoS 1
sprintf (buf, "Hello World! QoS 1 message from app version %f\r\n",
   version);
message.qos = MQTT:: QOS1;
message.payloadlen = strlen (buf)+1;
rc = client.publish (topic, message);
```

```
    while (arrivedcount < 2)
        client.yield (100);

    while (arrivedcount < 3)
        client.yield (100);


    if ((rc = client.unsubscribe (topic))!= 0)
        printf ("rc from unsubscribe was %d\r\n", rc);

    if ((rc = client.disconnect ())!= 0)
        printf ("rc from disconnect was %d\r\n", rc);

    socket.close ();
    printf ("Version %.2f: finish %d msgs\r\n", version, arrivedcount);

    return;
}
```

Настройте подключение к MQTT-серверу, расположенному по IP-адресу 192.168.42.1 (адрес точки доступа); проверьте подключение к MQTT-серверу.

Реализуйте программу для работы устройства согласно требованиям технического задания.

Отчет должен содержать следующие данные:

— цель работы;
— постановка задачи;
— схема лабораторной установки;
— фотография собранной установки;
— исходный код реализованной программы в среде Mbed;
— описание программного алгоритма;
— выводы об актуальности технологий интернета вещей и предложения по дополнительной функциональности системы автополива.

3.5. Дополнительные материалы

Ознакомьтесь с видеолекцией «Взаимодействие устройств по протоколу Bluetooth Low Energy», читает Олег Пехов (ТУСУР).

IT Академия Samsung. Взаимодействие устройств по Bluetooth Low Energy : [видеолекция] // YouTube.com : [сайт]. URL: https://youtu.be/KSbfA3iEciY (дата обращения: 30.09.2021).

Контрольные вопросы и задания

1. Какие существуют типовые режимы энергосбережения микроконтроллера?
2. Какими способами можно уменьшить энергопотребление микропроцессора?
3. Опишите способы экономии энергии конечных устройств интернета вещей.
4. В чем отличие статического и динамического потребления в CMOS-технологии?
5. Какими способами можно снизить динамическое энергопотребление?
6. Каковы преимущества использования аппаратных таймеров?
7. Оцените протоколы передачи данных с точки зрения энергоэффективности.
8. Какие сети относятся к группе сетей с низким энергопотреблением и широкой зоной охвата?
9. На какой частоте работает Bluetooth и какая спецификация актуальна в настоящее время?
10. Опишите преимущества и недостатки BLE с пояснениями (примеры, значения).
11. Что влияет на скорость передачи данных в беспроводных сетях?

12. Перечислите возможные роли устройств BLE.
13. Какие уровни выделяют в архитектуре BLE? Опишите основные задачи уровней.
14. Какой профиль Bluetooth отвечает за установку соединения?
15. Какой профиль Bluetooth отвечает за передачу двухканального стерео-звука?
16. Чем отличается первичный сервис от вторичного?
17. Приведите примеры свойств и дескрипторов характеристики сервиса.
18. Нарисуйте жизненный цикл устройства BLE.
19. В чем особенности развертывания сети NB-IoT?
20. Перечислите примеры систем, в которых выгодно использовать сети Bluetooth, LoRa, NB-IoT, 5G.

Глава 4. Платформы и средства обработки данных

4.1. Вычислительные модели и их применимость для систем интернета вещей

Полноценная система интернета вещей, кроме получения и передачи данных, должна обязательно содержать уровень управления. На этом уровне происходит обработка и анализ данных, собранных на предыдущем уровне. Обычно данные поступают в некоторое хранилище, при этом они могут быть уже предобработаны или накапливаться в «сыром» виде. С развитием возможностей вычислительных систем появлялись новые способы организации компьютерных вычислений и хранения данных, причем централизованные и децентрализованные методы сменяли друг друга (рис. 24). Рассмотрим основные особенности этих вычислительных моделей и их применимость для систем интернета вещей.

К централизованным моделям относятся вычисления на суперкомпьютере (Mainframe Computing) и облачные вычисления (Cloud Computing). Вычисления на мейнфрейме осуществляются с использованием множества процессоров, способных параллельно производить сотни миллиардов операций. Большие объемы вычислений нужны для выполнения сложных математических операций — решения задач в аэродинамике, метеорологии, физике высоких энергий, геофизике и др. Мейнфрейм сравним с собственным ЦОДом. Такое дорогое решение обеспечивает высокий уровень безопасности, автономности, отсутствие зависимости от других систем, но и большие накладные расходы по содержанию оборудования. Для систем интернета вещей мейнфреймовая модель может быть использована на предприятии

с локально сосредоточенным оборудованием и высоким уровнем безопасности данных. К системам с мейнфреймовой вычислительной моделью предъявляются высокие требования по скорости и гарантированности доставки данных.

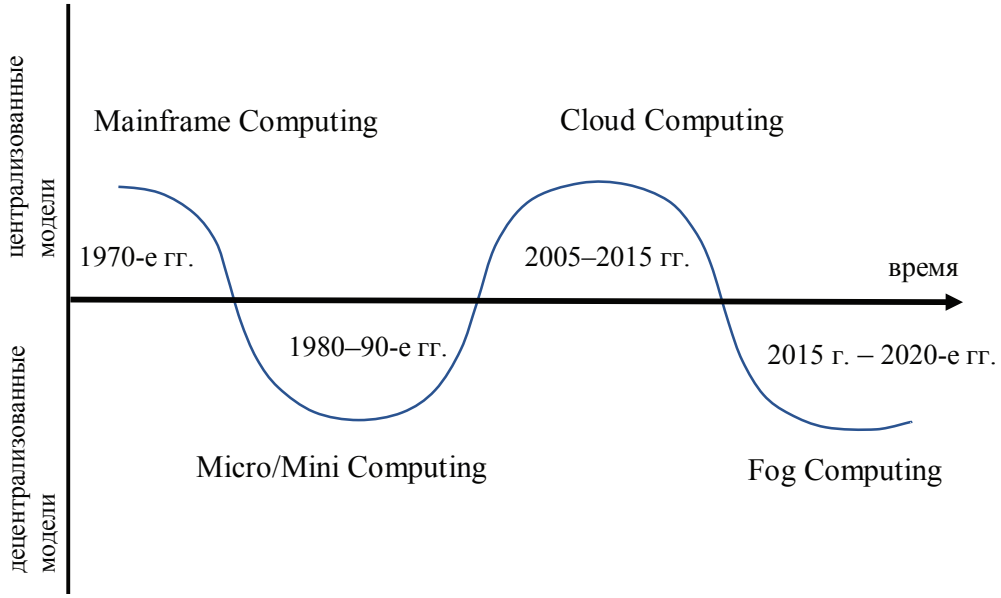


Рис. 24. Этапы возникновения и популярности вычислительных моделей

Другим вариантом централизованной модели является облачная система. Облачная модель имеет огромные преимущества для географически распределенных систем, а также снимает с потребителя задачу поддержания инфраструктуры. Еще одним преимуществом облачных систем является возможность быстрого масштабирования. Вертикальное масштабирование — увеличение вычислительных мощностей, горизонтальное масштабирование — кластеризация и параллельные вычисления.

Облачные вычисления — это технологии обработки данных, в которых компьютерные ресурсы предоставляются интернет-пользователю как онлайн-сервис. Слово «облако» здесь присутствует как метафора, олицетворяющая сложную инфраструктуру, скрывающую за собой все технические детали.

Современные облака появились в 2006 г., когда компания Amazon представила Amazon Web Services (AWS), положив начало движению облачных вычислений. AWS по сей день остается ведущей и очень надежной инфраструктурой платформ облачных веб-сервисов.

Уже в 2014 г. большинство крупнейших IT-вендоров на мировом рынке, включая Yandex, Microsoft, HP, Intel, SAP, IBM, Google, Oracle и др., имеет в своей линейке решения Cloud Computing.

Уровень безопасности в системах с моделью облачных вычислений в некоторых аспектах уступает мейнфреймовым, но для систем, в которых утечка данных не является критичной, такая модель имеет существенные преимущества: во-первых — отсутствие значительных капиталовложений; во-вторых — быстрый запуск системы; в-третьих — возможность быстро увеличивать ресурсы по мере необходимости.

К децентрализованным моделям относятся вычисления на персональных мини- или микрокомпьютерах (Micro/Mini Computing), а также пограничные (туманные) вычисления (Fog/Edge Computing).

Микровычисления зародились в эпоху развития персональных компьютеров. Мощность таких вычислений достаточна низкая, что не позволяет решать сложные вычислительные задачи в заданное время. Однако низкая стоимость владения решает вопрос доступности оборудования. Реализация систем интернета вещей на микрокомпьютере требует адаптации алгоритмов вычислений к особенностям устройства.

Для оптимизации вычислительной нагрузки используют технологию туманных вычислений. Данная модель построена в сервис-ориентированной архитектуре и использует все преимущества указанной технологии. Туман (туманные вычисления) возник с появлением умных устройств. Умные устройства генерируют данные и передают их по каналам связи в ЦОД, а затем они попадают в приложение пользователя.

Туман расположен вблизи производства или корпоративной сети. Серверы дают возможность предварительно обрабатывать данные для немедленного использования. Затем при необходимости предварительно обработанные данные могут быть отправлены в облако для более глубокого анализа или долгосрочного хранения. Туманные вычисления используются, когда для обработки данных устанавливаются критичные параметры. Это могут быть:

— высокая скорость отклика системы на состояние устройств;
— постоянная передача фиксированного объема данных в единицу времени;
— секретные сведения, содержащиеся в части данных.

Туман обеспечивает скорость, надежность и безопасность данных.

4.2. Облачные решения для разработки приложений

Иногда при работе с данными целесообразнее использовать виртуальное программное обеспечение или облачное решение. Облако — это не просто некое количество файлов и программ на некотором удаленном сервере. Облако является облаком, когда автоматизированы предоставление услуг и их администрирование.

Облачная платформа — это набор инструментов, предназначенных для удаленного запуска и использования приложений без затрат на приобретение серверного оборудования.

Распространены три основные модели облачных сервисов: инфраструктура как услуга (IaaS — Infrastructure as a Service), платформа как услуга (PaaS — Platform as a Service) и программное обеспечение как услуга (SaaS — Software as a Service). Модели отличаются глубиной погружения в систему и настройками, к которым пользователь имеет доступ. Чем больше глубина, тем больше возможностей. В IaaS пользователь получает доступ к данным, приложениям, базам данных и операционной системе, вычислительным ресурсам, на которых можно разворачивать и выполнять любое программное обеспечение. В PaaS пользователь получает готовую платформу, может управлять данными, устанавливать и конфигурировать свои приложения. В SaaS пользователю доступны только его данные и инструменты работы с ними.

В широком понимании облачная платформа есть не что иное, как средство для работы с различными программами и хранения информации.

Приведем несколько примеров существующих конструкторов, позволяющих разрабатывать приложения.

Supabase — облачная база данных с возможность авторизации. Когда создается проект, Supabase автоматически предоставляет базу данных Postgres SQL, аутентификацию пользователя и API. В системе доступны дополнительные функции, такие как подписки в реальном времени и хранение файлов.

Firebase — облачный сервис для разработки мобильных приложений с быстрой синхронизацией в базе данных NoSQL.

Deno Deploy — сервис для запуска серверного javascript-кода.

Glitch — сервис прототипирования веб-приложений, ботов с развитыми средствами групповой разработки.

Heroku — облачная PaaS-платформа, поддерживающая несколько языков программирования для запуска веб-приложений.

Перечисленные платформы призваны сделать разработку приложений доступной для широкого круга программистов и свести к минимуму подготовительные работы по установке и настройке программного обеспечения.

4.3. Платформы для разработки приложений IoT

Под термином «IoT-платформа» понимают инструмент разработки IoT-приложений и сервисов, объединяющий физические объекты и сеть.

В качестве примера рассмотрим платформу Mycodo — это программное обеспечение с открытым исходным кодом для микрокомпьютера Raspberry Pi, которое позволяет объединять входные данные и визуализировать их для восприятия и удобного управления. Приложение собирается при помощи описательного файла для сборки приложений docker-compose.yml, состоящего из нескольких контейнеров.

Контейнер — это помещенный на виртуальный диск файл, в который упаковывается приложение со всеми необходимыми для его работы зависимостями: кодом приложения, средой запуска, системными инструментами, библиотеками и настройками.

На рис. 25 представлен интерфейс сервиса, в котором визуализируются данные с нескольких датчиков в окне браузера.

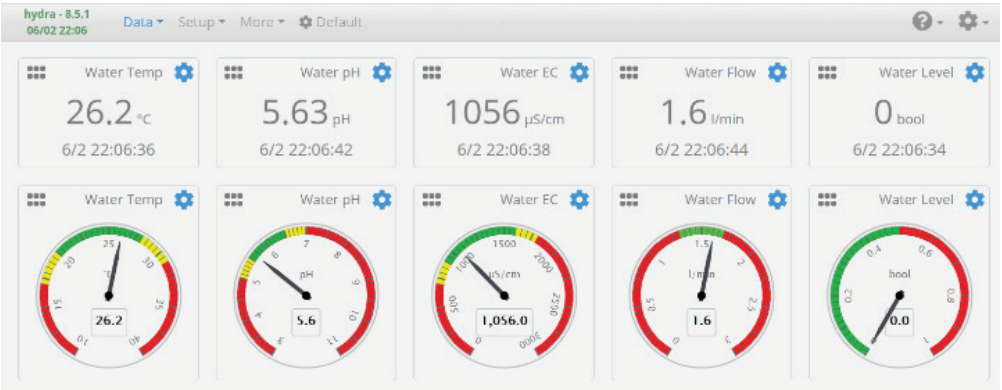


Рис. 25. Интерфейс Mycodo

Настройка датчиков осуществляется в веб-интерфейсе. Необходимо выбрать датчики, которые подключены к микроконтроллеру, и установить библиотеки для работы с ними. После добавления датчика нужно настроить его (прописать пины, к которым он подключен) и активировать. В результате настройки вывода в виде графиков и круговой шкалы интерфейс будет выглядеть как на рис. 25.

Характеристики полноценной IoT-платформы были сформулированы в компании Intel, которая продвигает свою платфору IoT Analytics:

1. Связь и нормализация (Connectivity & Normalization): различные протоколы и форматы данных должны быть сведены в один «программный» интерфейс с гарантией точной передачи данных и взаимодействия со всеми устройствами.
2. Управление устройствами (Device Management): должна быть обеспечена правильная, бесперебойная работа подключенных интернет-вещей, их конфигурирование, обновление ПО, причем не только ПО собственно вещей, но и приложений, работающих на устройстве или пограничных шлюзах.
3. База данных (Database): необходимо упорядочить обработку и перенос данных от разных IoT-платформ к информационным системам третьих лиц.
4. Обработка и управление действиями (Processing & Action Management): данные, полученные от вещей, влияют на со-

бытия в реальном мире, поэтому платформа должна уметь строить процессы, триггеры событий и выполнять прочие умные действия на основе конкретных данных датчиков.

5. Аналитика (Analytics): данные от вещей ценны сами по себе, поэтому существование комплекса средств их анализа является всенепременным требованием к платформе. Если сюда включить еще и средства кластеризации данных и глубокого машинного обучения вплоть до прогнозирующей аналитики, то ценность платформы, очевидно, растет.
6. Визуализация (Visualization): аналитику надо представить таким образом, чтобы пользователям было понятно, удобно и красиво.
7. Дополнительные инструменты (Additional Tools): должен быть набор инструментов, который позволяет разработчикам IoT создавать прототипы, тестировать и пробовать различные системы.
8. Внешние интерфейсы (External Interfaces): интеграция с помощью платформы — одна из главных возможностей. Всегда может потребоваться передача и обмен со сторонними системами, поэтому IoT-платформа обязана иметь интерфейсы прикладного программирования (API), комплекты разработки программного обеспечения (SDK) и шлюзы.

Этим требованиям полностью соответствует платформа WINNUM. Набор средств разработки позволяет создавать собственные приложения с использованием WINNUM Platform и WINNUM Cloud.

WINNUM Platform содержит полный набор инструментов для мониторинга, управления, оптимизации и создания автономных систем.

Она имеет встроенный редактор пользовательских интерфейсов с готовыми библиотеками графических элементов (виджеты, графики и пр.) и поддержкой сторонних элементов, доступных на специализированных ресурсах, для создания пользовательских дашбордов. Кроме того, в WINNUM Platform есть готовые инструменты для обмена данными с другими системами (BI, SCADA, ERP/MES и др.), встроенные серверы OPC UA и MTConnect для передачи необработанных данных и текущих статусов оборудования, поддержка HTTP/HTTPS и SQL-запросов.

4.4. Практические задания

Занятие № 9

Цель работы: приобрести опыт создания облачного приложения с использованием конструктора приложений.

Порядок выполнения работы:

1) выполните анализ существующих облачных решений для IoT;
2) обоснуйте выбор облачной платформы;
3) разверните в облаке проект с возможностью сбора данных;
4) настройте интерфейс пользователя по работе с данными;
5) разработайте мобильное или веб-приложение, используя конструктор для доступа к данным.

Занятие № 10

Цель работы: освоить разработку сервиса по сбору, анализу и визуализации данных, получаемых от устройств интернета вещей с использованием подхода Code-Less на основе Node-RED в облачной среде.

Порядок выполнения работы:

1) разверните ранее настроенный Node-RED;
2) соберите тестируемое устройство IoT;
3) выполните прошивку микроконтроллера;
4) подключите датчики;
5) создайте Dashboard для визуализации данных, приходящих с собранного устройства IoT;
6) настройте приложение для сбора и отображения данных;
7) настройте средства оповещения пользователя при выходе показаний датчика за допустимые пределы (мессенджер, почта, СМС);
8) разработайте телеграм-бот для получения показаний с сервера, изменения настроек, получения аварийных оповещений;
9) проведите сравнительный анализ облачного решения из предыдущей лабораторной работы с функционалом разработанного сервиса;
10) визуализируйте данные готового датасета.

4.5. Дополнительные материалы

Посмотрите видеолекцию «Облачные технологии в решениях IoT», читает Кирилл Валерьевич Святов (УлГТУ).

IT Академия Samsung. Облачные технологии в решениях IoT : [видеолекция] // YouTube.com: [сайт]. URL: https://youtu.be/UjWZ3FRkJ4M (дата обращения: 30.09.2021).

В лекции объяснены типовые архитектуры программных решений по работе с данными в IoT, технологии граничных и облачных вычислений на примере IBM Cloud, продемонстрировано создание сервиса по сбору, анализу и визуализации данных, получаемых от устройств интернета вещей с использованием подхода Code-Less на основе Node-RED в облачной среде.

Контрольные вопросы и задания

1. Перечислите вычислительные модели, на которых может быть реализована обработка данных в промышленных системах IoT.
2. Опишите централизованные вычислительные модели и примеры их использования.
3. Какие проблемы могут возникнуть при реализации системы интернета вещей с мейнфреймовой вычислительной моделью?
4. Какая децентрализованная вычислительная модель может быть построена в сервис-ориентированной архитектуре?
5. Перечислите преимущества использования облачной системы.
6. Какие проблемы могут возникать, если использовать публичное облако для управления умным домом? Как эти проблемы можно решить?
7. Зависит ли работоспособность IoT-системы от физического расположения облачного сервера, которым она пользуется? Ответ обоснуйте.

8. Чем туманные вычисления отличаются от облачных? Как можно использовать туманные и облачные сервисы?
9. Приведите примеры облачных сервисов: инфраструктура как услуга (IaaS), платформа как услуга (PaaS) и программное обеспечение как услуга (SaaS).
10. Перечислите решения для интернета вещей от российских облачных провайдеров.
11. Какие сервисы позволяет IBM добавить в приложение Node-RED?
12. Опишите функционал приложения Node-RED.
13. С каким протоколом передачи данных работает Node-RED?
14. Как из приложения Node-RED отправить данные в облако?
15. Где хранится информация об измерениях данных системы IoT?
16. Нарисуйте архитектуру системы IoT при использовании облачных сервисов.
17. Опишите минимальный функционал IoT-платформы.
18. Какой функционал имеет платформа Mycodo?
19. Перечислите известные IoT-платформы.
20. Какими характеристиками должна обладать полноценная IoT-платформа?

Библиографический список

Волкова, Т. Обучение преподавателей 2020 – Трек по интернету вещей : [видеолекции]. – Изображение (движущееся ; трехмерное) // YouTube.com : [сайт]. URL: https://www.youtube.com/playlist?list=PLJEYfuHbcEICFSHkSbgnzMDMf0UQ_rOmn (дата обращения: 30.09.2021).

Корнилов, А. Основы проектирования приложений интернета вещей : учебное пособие / А. Корнилов. — [Б. м.] : Ridero, 2018. – 173 с. : ил. ISBN 978-5-4490-4945-2. — Текст : электронный.

Кранц, М. Интернет вещей. Новая технологическая революция / М. Кранц ; перевод З. А. Мамедьярова. — Москва : БОМБОРА, 2018. — 332 с. : ил. — ISBN 978-5-04-090627-7. — Текст : непосредственный.

Крон, А. IoT и проблемы безопасности / А. Крон. — Текст : электронный // Habr.com : [сайт]. — 2018. — 15 марта. — URL: https://habr.com/ru/company/unet/blog/410849 (дата обращения: 30.09.2021).

Росляков, А. В. Интернет вещей : учебное пособие / А. В. Росляков, С. В. Ваняшин, А. Ю. Гребешков. — Самара : ПГУТИ, 2015. — 200 с. — URL: https://iotas.ru/files/documents/wg/учебник%20ИВ%20Росляков.pdf (дата обращения: 30.09.2021). — Текст : электронный.

Семенченко, П. И. Стандартизация взаимодействия устройств Интернета вещей / П. И. Семенченко. — Текст : электронный // Постулат. — 2017. — № 11. — С. 15–22. — URL: http://e-postulat.ru/index.php/Postulat/article/view/900/926 (дата обращения: 30.09.2021).

Стандарт NB-IoT Low-Power and Wide-Area, LPWA. — Текст : электронный // TAdviser.ru : [сайт]. — 2019. — URL: http://www.tadviser.ru/a/323705 (дата обращения: 30.09.2021).

Таненбаум, Э. Компьютерные сети / Э. Таненбаум, Д. Уэзеролл. — Санкт-Петербург : Питер, 2012. — 5-е изд. — 960 с. : ил. – ISBN 978-5-4461-1248-7. — Текст : непосредственный.

Шваб, К. Технологии Четвертой промышленной революции / К. Шваб, Н. Дэвис ; перевод с английского К. Ахметова, А. Врублев-

ского, В. Карпюка, А. Козлова. — Москва : Эксмо, 2018. — 320 с. — ISBN 978-5-04-095268-7. — Текст : непосредственный.

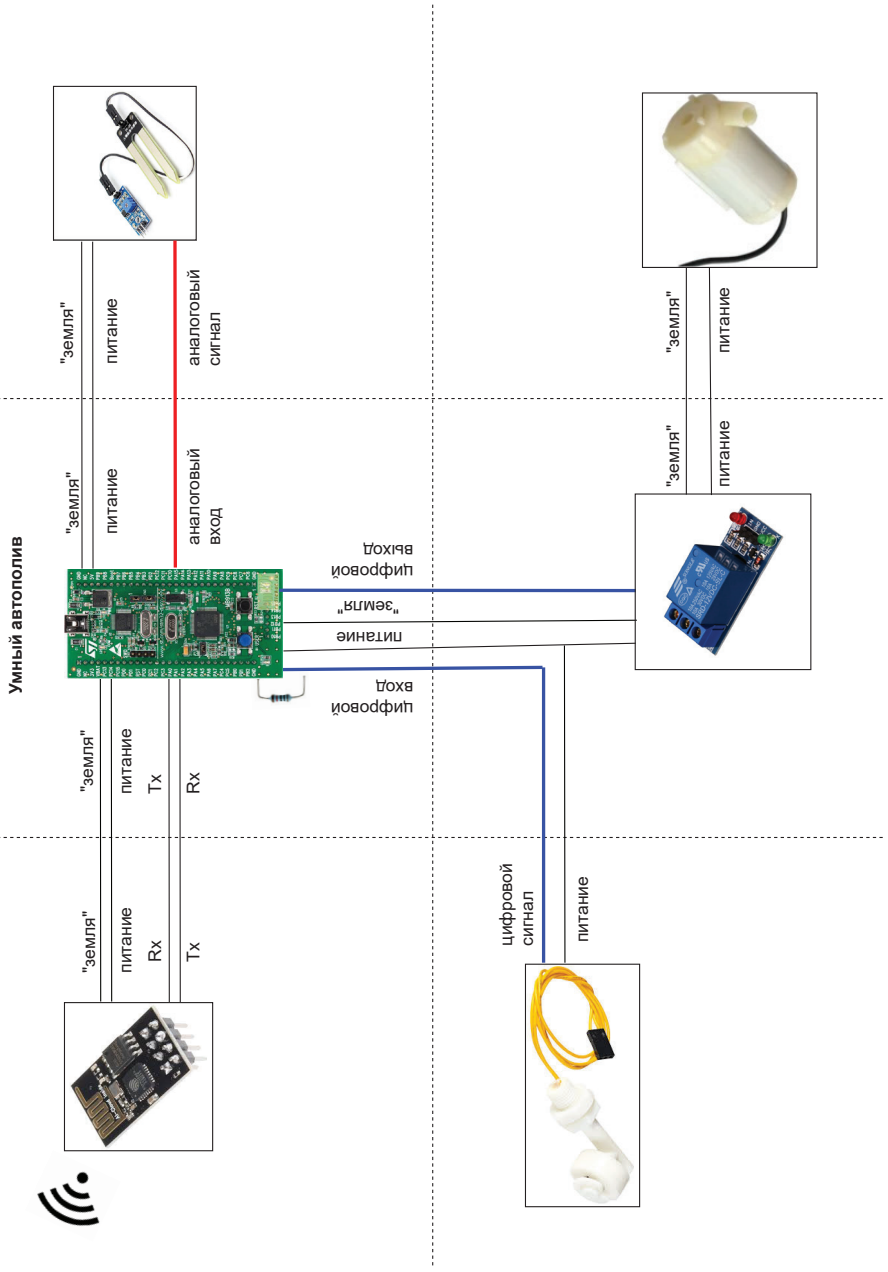
Шварц, М. Интернет вещей с ESP8266 / М. Шварц ; перевод с английского В. С. Яценкова. — Санкт-Петербург : БХВ-Петербург, 2018. — 224 с. : ил. — (Электроника). — ISBN 978-5-9775-4104-6. — Текст : непосредственный.

chrns [псевд.]. Си для встраиваемых систем / chrns. — [Б. м.] : Ridero, 2018. — ISBN 978-5-4493-6061-8. — URL: https://robotclass.ru/articles/c_for_embedded_systems/ (дата обращения: 30.09.2021). — Текст : электронный.

Gabriel, K. T. Mycodo Environmental Regulation System / K. T. Gabriel. — DOI 10.5281/zenodo.824199. — Текст : электронный // ResearchGate.net : [сайт]. — 2018. — 57 с. URL: https://www.researchgate.net/publication/328228021_Mycodo_Environmental_Regulation_System (дата обращения: 27.03.2022).

Приложения

1. Образец раздаточного материала для практических заданий



Датчик влажности почвы
предназначен для измерения содержания воды в почве.
В зависимости от содержания воды изменяется проводимость среды, что позволяет точно измерить уровень влажности аналоговым сигналом.

Модуль WiFi
предназначен для соединения оборудования по протоколу 802.11

Погружной насос
Работает от сети постоянного тока 6-12 В. Подает жидкость на высоту до 3 м.

Реле
Управляет подключением сети питания силовых агрегатов (двигатель, насос, нагревательный элемент, электромагнитный клапан и т.д.).

Датчик уровня жидкости
Содержит поплавок и герконовый переключатель. В зависимости от направления поплавка подключает/отключает сигнальный контакт к эталонному сигналу.

Умное освещение
"земля"
питание
цифровой вход
цифровой сигнал
"земля"
питание
цифровой выход
"земля"
питание
аналоговый сигнал
"земля"
питание
MOSI
MISO
SCLK
SS
"земля"
питание
MOSI
MISO
SCLK
SS
аналоговый сигнал
питание
"земля"
L
L
N
L
~ 220 В

PIR (Passive Infra Red) датчик движения
Выдает сигнал при изменении положения теплокровного объекта. Используется для определения нахождения человека в зоне действия.

Модуль Ethernet
предназначен для соединения оборудования по проводному протоколу 802.3

Светодиодный светильник
Работает от сети переменного тока 220 В.

Реле
Управляет подключением сети питания силовых агрегатов (двигатель, насос, нагревательный элемент, освещение, электромагнитный клапан и т.д.)

Датчик освещенности
Содержит фоторезистор, который изменяет сопротивление в зависимости от уровня освещенности. Выдает аналоговый сигнал, который позволяет точно определить уровень освещенности.

Умное отопление
"земля"
питание
MOSI
MISO
SCLK
SS
цифровой вход
цифровой выход
питание
"земля"
цифровой выход
+12В
+12В
L
L
N
~ 220 В
- 12В +
LoRa

Датчик температуры и влажности
Выдает цифровой сигнал о состоянии уровня влажности и температуры.

Модуль LoRa
предназначен для соединения оборудования по беспроводному протоколу LoRa.

Отопительный элемент (котёл)
Работает от сети переменного тока 220 В. Обеспечивает нагрев жидкости.

Реле
Управляет подключением сети питания силовых агрегатов (двигатель, насос, нагревательный элемент, освещение, электромагнитный клапан и т.д.).

Электромагнитный клапан
Устройство управления подачи жидкости и газа. При подаче питания постоянного тока 12В открывается для пропускания жидкости и газа.

2. Лабораторное оборудование для практических заданий

Базовое измерительное и отладочное оборудование:
— программатор;
— логический анализатор 16-канальный;
— осциллограф цифровой двухканальный, 100 МГц/1 ГГц;
— мультиметр цифровой;
— Wi-Fi маршрутизатор.

Состав типового рабочего места студента (перечень на 1 комплект, 1 рабочее место)

№	Наименование	Кол-во, шт
1	Отладочная плата на базе MCU STM32L152RET6 (ARM Cortex-M3), ST-LINK/V2–1, Arduino-интерфейс	1
2	Модуль электромагнитных реле	1
3	Модуль адаптера модема радиосети	3
4	Модуль универсальной монтажной платы	1
5	Модем радиосети LoRa	3
6	Модем радиосети 6LoWPAN	3
7	Модуль адаптера внешних датчиков	1
8	Интегрированный датчик: акселерометр + гироскоп + магнитометр + датчик давления + датчик температуры и влажности	1
9	Модуль аккумуляторного питания	1
10	Датчик естественной освещенности	1
11	Датчик температуры, влажности, давления воздуха	1
12	Модуль приемника геопозиционирования GPS/ГЛОНАСС	1
13	Модуль управления нагрузками постоянного тока	1
14	Модуль тактовых кнопок	1
15	Датчик обнаружения протечки с аналоговым или цифровым выходом	1
16	Датчик звукового давления	1
17	Датчик влажности почвы	1
18	Модуль ультразвукового дальномера	1
19	Модуль считывателя электронного ключа	1
20	Лампа светодиодная белая 12 В	5

№	Наименование	Кол-во, шт
21	Лампа светодиодная RGB 12 В	5
22	Источник питания 12 В, 1 А	1
23	Источник питания 5 В, 1 А	4
24	Кабель USB — microUSB	4
25	Датчик влажности почвы с аналоговым или цифровым выходом	1
26	Комплект монтажный (отвертка, провода монтажные, клеммы соединительные)	1
27	Микрокомпьютер на отладочной плате	1
28	Беспроводной модуль Bluetooth v4.0 & BLE	1

Учебное издание

Папуловская Наталья Владимировна

ОСНОВЫ ИНТЕРНЕТА ВЕЩЕЙ

Редактор *Т. Е. Мерц*
Верстка *О. П. Игнатьевой*

Подписано в печать 12.10.2022. Формат 70×100/16.
Бумага писчая. Цифровая печать. Усл. печ. л. 8,4.
Уч.-изд. л. 4,8. Тираж 30 экз. Заказ 171.

Издательство Уральского университета
Редакционно-издательский отдел ИПЦ УрФУ
620049, Екатеринбург, ул. С. Ковалевской, 5
Тел.: 8 (343) 375-48-25, 375-46-85, 374-19-41
E-mail: rio@urfu.ru

Отпечатано в Издательско-полиграфическом центре УрФУ
620075, Екатеринбург, ул. Тургенева, 4
Тел.: 8 (343)350-56-64, 350-90-13
Факс: 8 (343) 358-93-06
http://print.urfu.ru

9 785799 635374

ПАПУЛОВСКАЯ НАТАЛЬЯ ВЛАДИМИРОВНА
Кандидат педагогических наук, доцент кафедры информационных технологий и систем управления ИРИТ — РтФ УрФУ. Профессиональные интересы: отраслевые автоматизированные системы управления, моделирование процессов управления, методика преподавания учебных дисциплин в высшей профессиональной школе.