УДК 004.3 doi: 10.17586/2226-1494-2020-20-5-722-728

КОНФИГУРИРУЕМЫЕ IoT-УСТРОЙСТВА НА ОСНОВЕ SoC-СИСТЕМ ESP8266 И ПРОТОКОЛА MQTT

С.В. Корзухин, Р.Р. Хайдарова, В.Н. Шматков

Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
Адрес для переписки: shmatkovvlad@gmail.com

Информация о статье
Поступила в редакцию 18.05.20, принята к печати 05.08.20
Язык статьи — русский

Ссылка для цитирования: Корзухин С.В., Хайдарова Р.Р., Шматков В.Н. Конфигурируемые IoT-устройства на основе SoC-систем ESP8266 и протокола MQTT // Научно-технический вестник информационных технологий, механики и оптики. 2020. Т. 20. № 5. С. 722–728. doi: 10.17586/2226-1494-2020-20-5-722-728

Аннотация

Предмет исследования. Рассмотрены популярные протоколы прикладного уровня для устройств интернета вещей, использующиеся в сетях. Выполнен сравнительный анализ протоколов в контексте использования ресурсов сети и надежности передачи данных, выявлены их достоинства и недостатки применительно к использованию для передачи данных в системах интернета вещей. Проведен обзор аппаратных платформ для построения устройств интернета вещей. Большой практический интерес для создания оконечных устройств интернета вещей могут представлять SoC-системы, объединяющие на одном полупроводниковом кристалле вычислительный модуль, периферийные устройства и устройства связи. **Метод.** Предложен подход к построению конфигурируемого оконечного устройства интернета вещей на основе SoC-системы ESP8266. Для связи устройств с сервером управления и сбора данных применен протокол MQTT, который позволяет экономить ресурсы сети и логическим и иерархическим образом разделять устройства интернета вещей в сети. Предложена простая архитектура платформы на основе открытого программного обеспечения OpenHAB, MQTT-брокера Eclipse Mosquitto и протокола MQTT для объединения устройств интернета вещей в сеть. Преимуществом предложенного подхода является использование шаблонов приложений IoT-устройств. **Основные результаты.** Разработаны шаблоны приложений сенсора и актуатора, конфигурируемые посредством WEB-интерфейса. Реализован режим точки доступа для начальной настройки устройства. Получены зависимости времени отправки и приема MQTT-сообщения в зависимости от его длины, измерено время отклика устройства на сетевые запросы и потери сетевых пакетов и MQTT-сообщений. **Практическая значимость.** На основе шаблона приложений созданы устройства умного светильника, моторизированных штор, датчиков освещенности, сенсор газового состава (углекислый газ, метан), сенсор температуры, давления и влажности. Измерены параметры полученных устройств, характеризующие время обработки сообщений. Построен стенд, объединяющий созданные устройства. Преимуществом использованного подхода является возможность поддержки большого количества разнообразных внешних устройств и быстрота создания нового устройства на основе готового шаблона приложения. Показано, что использованный в работе подход позволяет создавать оконечные устройства интернета вещей с приемлемыми эксплуатационными характеристиками.

Ключевые слова
IoT-устройства, сенсоры, актуаторы, MQTT, SoC

doi: 10.17586/2226-1494-2020-20-5-722-728

CONFIGURABLE IoT DEVICES BASED ON ESP8266 SoC SYSTEM AND MQTT PROTOCOL

S.V. Korzukhin, R.R. Khaydarova, V.N. Shmatkov

ITMO University, Saint Petersburg, 197101, Russian Federation
Corresponding author: shmatkovvlad@gmail.com

Article info
Received 18.05.20, accepted 05.08.20
Article in Russian

For citation: Korzukhin S.V., Khaydarova R.R., Shmatkov V.N. Configurable IoT devices based on ESP8266 SoC system and MQTT protocol. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2020, vol. 20, no. 5, pp. 722–728 (in Russian). doi: 10.17586/2226-1494-2020-20-5-722-728

Abstract

Subject of Research. The paper considers popular application layer protocols for the Internet of Things devices used in TCP/IP networks. Comparative analysis of these protocols is performed in the context of network resources and reliable data transmission application. Drawbacks and benefits of these protocols for the data transfer inside the Internet of Things networks are identified. Review of hardware platforms for the Internet of Things devices is performed. SoC systems which combine a processor, peripherals and networking module on one semiconductor chip can have significant practical value. **Method.** An approach for the creation of the configurable IoT device using ESP8266 SoC system is proposed. MQTT protocol is used for connectivity with a management server and data collection which can save network bandwidth and arrange logically IoT devices. Simple IoT architecture based on MQTT protocol and OpenHAB and Eclipse Mosquito software is proposed for combination of IoT devices in a network. The advantage of the proposed approach is the use of IoT device application templates. **Main Results.** Template applications for web-configurable sensor and actuator are created. Access point mode for initial device setup is implemented. Parameters of these devices related to the MQTT message response time are measured. Dependencies of sending and receiving time from MQTT message are obtained depending on its length. Network response time, Transmission Control Protocol packet loss rate, and MQTT message loss rate are measured. **Practical Relevance.** The following IoT devices have been built based on the mentioned template applications: smart light, motorized curtains, light, gas, temperature, pressure and humidity sensors. The parameters of the received devices, which characterize the message processing time, have been measured. Demonstration stand combining developed devices has been built. The approach used in this work provides rapidly creation of a big variety of IoT devices built on IoT device application templates. The proposed approach also gives the possibility to build simple IoT devices with acceptable operating parameters.

Keywords

IoT, devices, sensors, actuators, MQTT, SoC

Введение

Концепция интернета вещей (Internet of Things, IoT) возникла в начале 2000-х годов и с тех пор является одним из наиболее популярных направлений в области информационных технологий. В основе концепции лежит идея о вычислительной сети устройств, способных взаимодействовать с внешним миром и друг с другом [1].

Решения интернета вещей включают в себя устройства для умных помещений, умных городов, медицины, сельского хозяйства, производства, безопасности и т. д. Вследствие этого для связи решений интернета вещей друг с другом и с другими узлами сети используются разнообразные, отвечающие конкретным задачам, стандарты и протоколы проводной и беспроводной связи [2, 3].

Для построения устройств интернета вещей также используется разнообразная элементная база, отвечающая предъявляемым требованиям вычислительной мощности, тепловыделению, энергопотреблению, совместимости. В отдельный класс устройств можно выделить SoC-устройства (System on Chip, или система на кристалле). Такие системы объединяют процессор, память, необходимые периферийные устройства и модули связи в одной полупроводниковой сборке [4, 5]. Они, как правило, имеют относительно невысокую вычислительную мощность и могут быть использованы в оконечных устройствах, для граничных и туманных вычислений.

Отдельную сложность в разработке IoT-устройств представляет необходимость поддержки разнообразных аппаратных решений для взаимодействия этих устройств с внешним миром. Это увеличивает трудоемкость создания устройств и время на их разработку. Сокращение времени разработки и снижение ее сложности является важной задачей.

В данной работе предложен подход к созданию оконечных устройств интернета вещей на основе одной из популярных SoC-систем ESP8266[1] с использованием протокола MQTT (Message Queuing Telemetry Transport)[2].

Протоколы прикладного уровня для устройств интернета вещей

На сегодняшний день существует большое разнообразие протоколов и стандартов связи, используемых в сетях интернета вещей. Применимость того или иного стандарта или протокола зависит от множества факторов, таких как необходимая пропускная способность, дальность связи, устойчивость к помехам, безопасность, потребление энергии и т. д. Среди проводных стандартов связи можно выделить IEEE 802.3 (Ethernet), ModBus, PowerLine (PLC), RS-232, RS-485, среди беспроводных – семейство IEEE 802.11 (Wi-Fi), ZigBee, Z-Wave, BLE, LTE, LoRa, SigFox, 6LoWPAN, NB-IoT [2, 3].

Перечисленные стандарты связи с точки зрения модели сетевого взаимодействия относятся к физическому и канальному уровням. С точки зрения практического использования интерес представляют протоколы прикладного уровня, основанные на стеке протоколов TCP/IP (Transmission Control Protocol/Internet Protocol)[3]. Примером такого протокола может служить широко используемый в сети Интернет протокол HTTP (Hypertext

[1] Страница продукта ESP8266 [Электронный ресурс]. Режим доступа: https://www.espressif.com/en/products/socs/esp8266ex/overview, свободный. Яз. англ. (дата обращения: 23.04.2020).

[2] Описание протокола MQTT [Электронный ресурс]. Режим доступа: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html, свободный. Яз. англ. (дата обращения: 23.04.2020).

[3] Описание стека протоколов TCP/IP [Электронный ресурс]. Режим доступа: https://tools.ietf.org/html/rfc1180, свободный. Яз. англ. (дата обращения: 23.04.2020).

Transfer Protocol)[1]. Однако его использование не всегда оправдано ввиду возможных высоких задержек передачи данных и большому количеству избыточной информации в сообщении [6].

В качестве альтернативы протоколу HTTP могут быть использованы AMQP (Advanced Message Queuing Protocol)[2], MQTT, JMS (Java Message Service)[3], CoAP (Constrained Application Protocol)[4], XMPP (Extensible Messaging and Presence Protocol)[5], DDS (Data Distribution Service)[6]. Протокол DDS является распределенным и предназначен для задач реального времени. AMQP, MQTT, JMS, CoAP, XMPP являются протоколами передачи сообщений, причем первые три из них требуют наличия в сети специального устройства-брокера [7]. Наличие брокера является одновременно преимуществом, поскольку избавляет узлы сети от необходимости выполнять работу по маршрутизации сообщений, проверке доставки и т. д., и недостатком, так как само наличие брокера приводит к присутствию в сети дополнительного узла.

Протокол MQTT является удобным с точки зрения организации взаимодействия устройств сети применительно к интернету вещей. MQTT-сообщение состоит из темы и тела сообщения. Тема сообщения имеет «иерархическую» структуру с символом — разделителем «/» и подстановочными символами «+» (произвольная тема на том же уровне иерархии) и «#» (все темы «справа» от себя). Это позволяет удобным и логичным образом адресовать сообщения одному или группе IoT-устройств. Также благодаря поддержке функции QoS (Quality of Service) протокол позволяет осуществлять надежную передачу данных [8, 9]. Показано что переход на протокол MQTT для обмена данными позволяет существенно экономить ресурсы сервера и сети при передаче одного и того же объема данных и при одинаковом количестве подключенных устройств, уступая по этим параметрам только протоколу CoAP [10, 11]. С точки зрения безопасности и надежности передачи данных MQTT в целом удовлетворяет требованиям, предъявляемым к устройствам интернета вещей [12, 13]. В качестве успешного примера применения протокола MQTT в IoT-устройствах можно привести работу [14]. Таким образом, применение этого протокола является оправданным, и в данной работе принято решение о его использовании.

Платформа для построения устройств интернета вещей

В качестве платформы для построения интернета вещей выбраны вычислительные модули ESP8266 (также известен как ESP12-E), как отвечающие требованиям производительности, энергопотребления, универсальности, компактности и простоты интеграции. Важным фактором стала поддержка данных модулей платформой прототипирования Arduino, а также поддержка устройством сети Wi-Fi. Подсистема управления интернета вещами в рамках работы построена на базе одноплатного мини-ПК (персонального компьютера) Raspberry Pi с установленным программным обеспечением (ПО) OpenHAB[7], на котором был развернут сервер выполнения команд и сбора данных. Каждое умное устройство строилось с использованием платформы Arduino на основе микроконтроллера ESP8266, объединенного в сеть Wi-Fi с сервером выполнения команд. Для взаимодействия сервера выполнения команд с IoT-устройствами использован протокол MQTT. В качестве брокера MQTT использовано по Eclipse Mosquitto. Схема платформы для построения устройств интернета вещей схематически отображено на рис. 1.

Разработка IoT-устройств

Существующие коммерческие IoT-устройства являются решениями с закрытым исходным кодом. Аппаратная и программная части устройств целиком разрабатываются производителем. Для обмена данными используются проприетарные протоколы. Разработка нового устройства требует написания программного кода для поддержки аппаратной части устройства, реализующей его взаимодействие с физическим окружением: разнообразных датчиков, модулей связи, двигателей, реле и т. д. Характерными представителями такого подхода к разработке являются устройства производства компаний Fibaro, Rubetek, Xiaomi.

В отличие от предлагаемых на рынке решений, в работе рассматривается подход к созданию умных устройств на основе открытых систем и уже разработанных библиотек поддержки внешних устройств. Основой IoT-устройства является универсальное приложение-шаблон сенсора или актуатора, в котором реализованы: работа с сетью; сбор данных о физическом окружении устройства; возможности конфигурирования и интеграции устройства с управляющей системой; обмен данными с ней. Для адаптации шаблона к работе с конкретным физическим устройством необходимо выполнить разработку части программы, отвечающей за логику работы с этим устройством, используя библиотеки поддержки (или драйверы устройств). Это позволяет сфокусироваться на логике работы интернета

[1] Описание протокола HTTP [Электронный ресурс]. Режим доступа: https://tools.ietf.org/html/rfc2616, свободный. Яз. англ. (дата обращения: 23.04.2020).

[2] Описание протокола AMQP [Электронный ресурс]. Режим доступа: https://www.amqp.org/resources/specifications, свободный. Яз. англ. (дата обращения: 23.04.2020).

[3] Описание протокола JMS [Электронный ресурс]. Режим доступа: https://www.ietf.org/rfc/rfc6167.txt, свободный. Яз. англ. (дата обращения: 23.04.2020).

[4] Описание протокола CoAP [Электронный ресурс]. Режим доступа: https://tools.ietf.org/html/rfc7252, свободный. Яз. англ. (дата обращения: 23.04.2020).

[5] Описание протокола XMPP [Электронный ресурс]. Режим доступа: https://tools.ietf.org/html/rfc6120, свободный. Яз. англ. (дата обращения: 23.04.2020).

[6] Описание протокола DDS [Электронный ресурс]. Режим доступа: https://www.omg.org/spec/DDS/About-DDS, свободный. Яз. англ. (дата обращения: 23.04.2020).

[7] Документация openHAB [Электронный ресурс]. Режим доступа: https://www.openhab.org/docs, свободный. Яз. англ. (дата обращения: 24.04.2020).

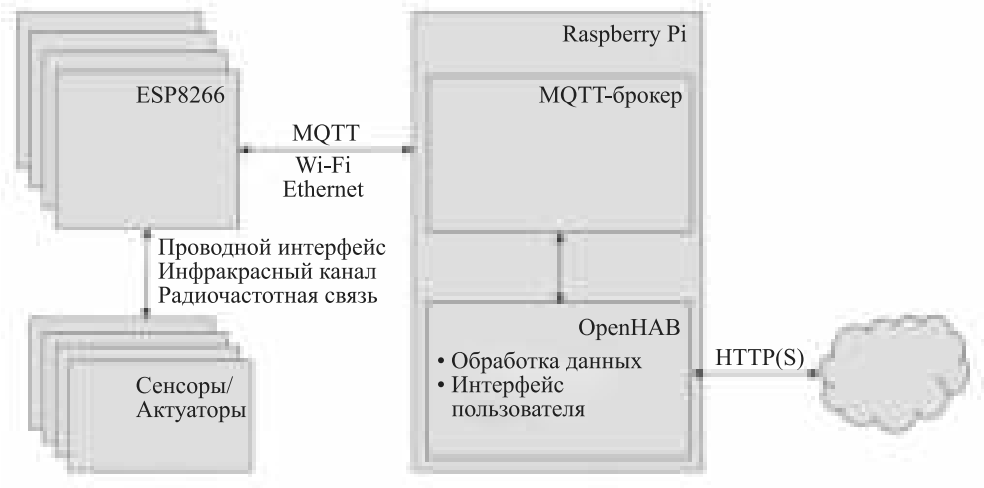

Рис. 1. Схема платформы для построения устройств интернета вещей

вещей и не разрабатывать библиотеки для работы с ее аппаратной частью, что снижает сложность и сокращает время разработки.

Для разработки IoT-устройств использована программная платформа прототипирования Arduino. Это обусловлено следующими факторами:

1) платформа поддерживает популярные микроконтроллеры и SoC-системы, в том числе ESP8266;
2) платформа поддерживает большое количество модулей внешних устройств, что упрощает разработку программного кода IoT-устройства и избавляет от самостоятельной разработки драйверов устройств;
3) платформа поддерживается разработчиком, обновляется и популярна в пользовательских сообществах;
4) для Arduino разработан большой объем библиотек.

Платформа не лишена недостатков: платой за совместимость с большим количеством внешних устройств является ограниченная совместимость внешних устройств между собой, благодаря использованию фреймворка Arduino пользователю доступно меньше аппаратных ресурсов (стек, память, процессорное время), и доступно всего лишь одно прерывание независимо от предоставляемых контроллером аппаратных возможностей.

Так как в качестве стандарта передачи данных выбрано сочетание сети Wi-Fi и протокола MQTT, устройство должно содержать клиент Wi-Fi и иметь возможность работать в режиме станции, и содержать клиент MQTT для обмена сообщениями с сервером с целью передачи данных сенсоров и получения команд от сервера. Также необходимо обрабатывать данные с внешних устройств – сенсоров, и (в некоторых случаях, например «умный выключатель») уметь обрабатывать команды от внешних устройств (нажата кнопка, переключен выключатель), поступающие способом, отличным от передачи MQTT-сообщения. Для реализации конфигурируемости устройство имеет WEB-интерфейс, который позволяет модифицировать значение некоторых его переменных. Хранение настроек осуществляется в энергонезависимой памяти устройства (EEPROM, Flash). Для первоначальной настройки устройства используется режим точки доступа. В случае невозможности подключения к Wi-Fi-сети устройство переходит в режим точки доступа и предоставляет свой WEB-интерфейс в собственной Wi-Fi-сети. Для обработки внешних команд используется аппаратное прерывание. Блок-схема работы устройства — актуатора представлена на рис. 2.

Работа сенсора аналогична работе актуатора, за исключением того, что сенсор не обрабатывает команды от внешних устройств и не обрабатывает команды по протоколу MQTT, а вместо этого осуществляет чтение данных из внешнего устройства и отправку их в виде MQTT-сообщений. Как сенсор, так и актуатор предоставляют пользовательский WEB-интерфейс для конфигурирования. Внешний вид интерфейса показан на рис. 3. Реализация настроек клиентов Wi-Fi и MQTT аналогична для сенсора и актуатора, настройки тем MQTT-сообщений являются специфичными для каждого из устройств и отвечают функциональности устройства и его аппаратным особенностям, а также предполагаемому расположению устройства.

Большинство библиотек, реализующих поддержку протокола MQTT для Arduino, предполагают синхронную обработку MQTT-сообщений во внутреннем цикле основного цикла кода микроконтроллера. Такой способ обработки приводит к тому, что во время приема MQTT-сообщений выполнение всего остального кода устройства блокировано. Это делает практически невозможным выполнение других задач, таких как выполнение кода WEB-сервера, чтение данных, и т. д. Полностью асинхронная работа кода в фреймворке Arduino также невозможна ввиду отсутствия абстракции потока. Одним из возможных способов переключения задач является использование прерывания. Этот способ использован в данной работе. Обработка и отправка сообщений MQTT на устройствах выглядит следующим образом. В экземпляре MQTT-клиента регистрируется функция обратного вызова для определенного типа события (прием, отправка сообщения, ошибка отправки и т. д.). Вся обработка события происходит внутри этой функции. Важной особенностью является недопустимость использования функций за-

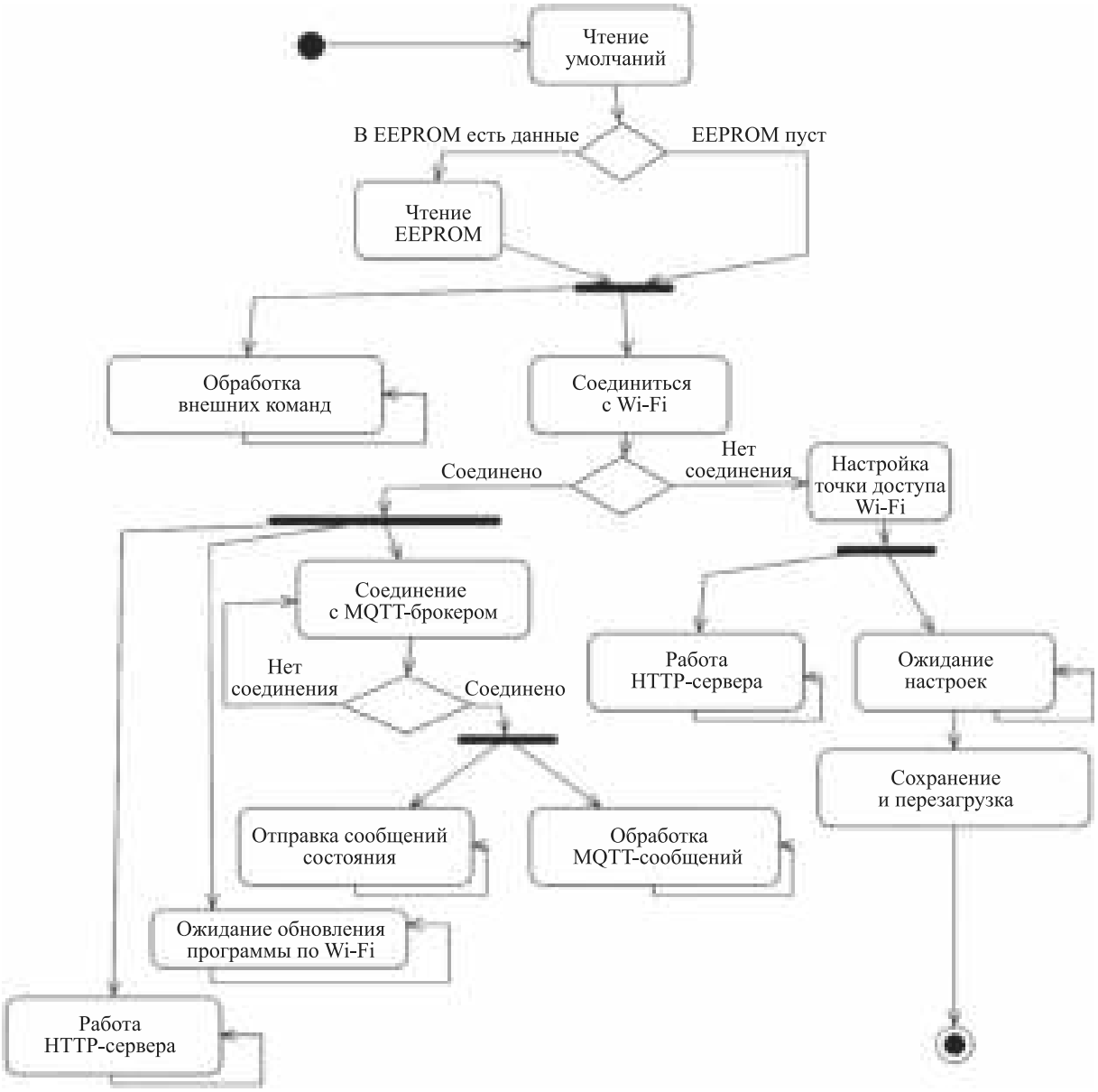

Рис. 2. Блок-схема актуатора

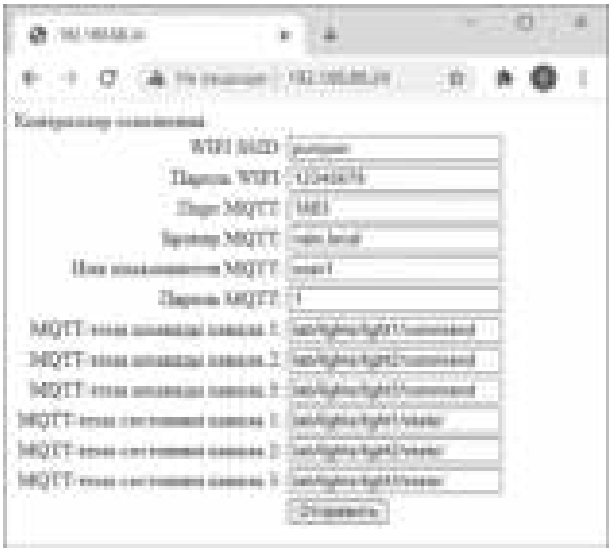

Рис. 3. Пример веб-интерфейса актуатора

держки delay() внутри функций обратного вызова и функций, вызываемых из них. Само время обработки события должно быть по возможности минимальным.

С использованием предложенного подхода построены следующие смарт-устройства: светильник, моторизированные шторы, датчики освещенности, сенсор газового состава (углекислый газ, метан), сенсор температуры, давления и влажности. Благодаря использованию разработанного шаблона приложения удалось сократить время разработки устройств.

В ходе тестирования устройств определены временные характеристики обработки MQTT-сообщений, и получены зависимости времени получения и отправки сообщений в зависимости от длины сообщения для параметров QoS: 1 и QoS: 2, а также оценено время обработки внешнего прерывания. Полученные зависимости показаны на рис. 4 (для передачи сообщений) и рис. 5 (для приема сообщений). Максимально возможная длина передаваемого сообщения составила

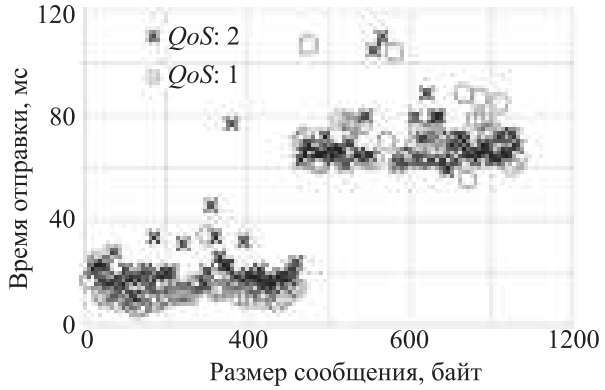

Рис. 4. Время отправки MQTT-сообщения

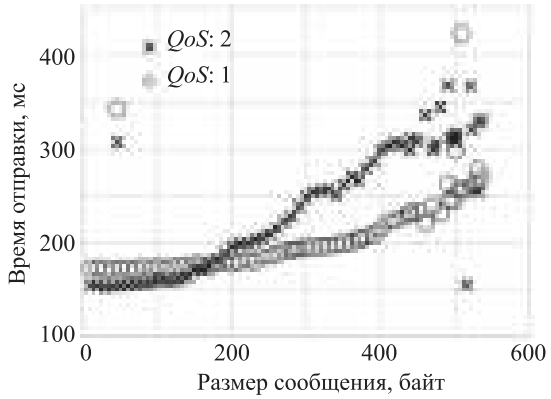

Рис. 5. Время приема MQTT-сообщения

приблизительно 1 000 Б, максимальная длина принимаемого — приблизительно 500 Б.

Время обработки внешнего прерывания составило в среднем 60–70 мс и определялось в основном значением параметра задержки для устранения дребезга.

Измерен показатель потери сообщений в случае, если на MQTT-брокере существует большая очередь непринятых сообщений. Для создания нагрузки использовался MQTT-клиент из пакета библиотек для фреймворка Node.js, который запускался на x86 совместимом ПК, значение QoS-клиента устанавливалось равным 2. Создавалась очередь из 1 000 сообщений. Потери определялись при помощи подсчета количества сообщений, принятых на устройстве, отнесенных к известной длине очереди, хранящейся на MQTT-брокере. Измененные потери сообщений составили не более 0,5 % при длине сообщения 100 Б.

Измерены времена ответа устройств на ICMP (Internet Control Message Protocol)-пакеты, которые составили: среднее время ответа 160 мс, минимальное 2 мс и максимальное 2500 мс. Доля потери пакетов при этом составила 0,2 %. Все результаты получены при тактовой частоте ядра SoC-системы ESP8266, равной 80 МГц. Потребление памяти устройства не измерялось.

Заключение

В работе разработан шаблон кода конфигурируемого IoT-устройства на основе SoC-системы ESP8266, использующего протокол MQTT для обмена данными и командами. Измерены временные характеристики устройств, относящиеся к обработке MQTT-сообщений и ответам на сетевые запросы. Согласно результатам измерений, на основе предложенного шаблона возможно строить IoT-устройства умного помещения с приемлемыми эксплуатационными характеристиками.

Благодаря использованию шаблонов приложений и свободно распространяемых библиотек поддержки внешних устройств сокращаются время и сложность разработки. С применением предложенного подхода построен ряд смарт-устройств. Данная работа развивает результаты, относящиеся к разработке интернета вещей, полученные в работе [15].

Литература

1. Gershenfeld N.A. When Things Start to Think. New York: Henry Holt and Company, 2000. 224 p.
2. Dragomir D., Gheorghe L., Costea S., Radovici A. A Survey on Secure Communication Protocols for IoT Systems // Proc. of the International Workshop on Secure Internet of Things (SIoT 2016). 2016. P. 49–62. doi: 10.1109/SIoT.2016.012
3. Hejazi H., Rajab H., Cinkler T., Lengyel L. Survey of platforms for massive IoT // Proc. of the IEEE International Conference on Future IoT Technologies (Future IoT 2018). 2018. doi: 10.1109/FIOT.2018.8325598
4. Polianytsia A., Starkova O., Herasymenko K. Survey of hardware IoT platforms // Proc Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T). 2016. P. 152–153. doi: 10.1109/INFOCOMMST.2016.7905364
5. Singh K.J., Kapoor D.S. Create Your Own Internet of Things: A survey of IoT platforms // IEEE Consumer Electronics Magazine. 2017. V. 6. N 2. P. 57–68. doi: 10.1109/MCE.2016.2640718
6. Naik N., Jenkins P. Web protocols and challenges of Web latency in the Web of Things // Proc. 8th International Conference on Ubiquitous and Future Networks (ICUFN 2016). 2016. P. 845–850. doi: 10.1109/ICUFN.2016.7537156
7. Селезнёв С.П., Яковлев В.В. Архитектура промышленных приложений IoT и протоколы AMQP, MQTT, JMS, REST, CoAP,

References

1. Gershenfeld N.A. *When Things Start to Think*. New York, Henry Holt and Company, 2000, 224 p.
2. Dragomir D., Gheorghe L., Costea S., Radovici A. A Survey on Secure Communication Protocols for IoT Systems. *Proc. of the International Workshop on Secure Internet of Things (SIoT 2016)*, 2016, pp. 49–62. doi: 10.1109/SIoT.2016.012
3. Hejazi H., Rajab H., Cinkler T., Lengyel L. Survey of platforms for massive IoT. *Proc. of the IEEE International Conference on Future IoT Technologies (Future IoT 2018)*, 2018. doi: 10.1109/FIOT.2018.8325598
4. Polianytsia A., Starkova O., Herasymenko K. Survey of hardware IoT platforms. *Proc Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T)*, 2016, pp. 152–153. doi: 10.1109/INFOCOMMST.2016.7905364
5. Singh K.J., Kapoor D.S. Create Your Own Internet of Things: A survey of IoT platforms. *IEEE Consumer Electronics Magazine*, 2017, vol. 6, no. 2, pp. 57–68. doi: 10.1109/MCE.2016.2640718
6. Naik N., Jenkins P. Web protocols and challenges of Web latency in the Web of Things. *Proc. 8th International Conference on Ubiquitous and Future Networks (ICUFN 2016)*, 2016, pp. 845–850. doi: 10.1109/ICUFN.2016.7537156
7. Seleznev S., Yakovlev V. Industrial Application Architecture IoT and protocols AMQP, MQTT, JMS, REST, CoAP, XMPP, DDS.

XMPP, DDS // International Journal of Open Information Technologies. 2019. V. 7. N 5. P. 17–28.
8. Hwang H.C., Park J., Shon J.G. Design and implementation of a reliable message transmission system based on MQTT protocol in IoT // Wireless Personal Communications. 2016. V. 91. N 4. P. 1765–1777. doi: 10.1007/s11277-016-3398-2
9. Roy D.G., Mahato B., De D., Buyya R. Application-aware end-to-end delay and message loss estimation in Internet of Things (IoT) – MQTT-SN protocols // Future Generation Computer Systems. 2018. V. 89. P. 300–316. doi: 10.1016/j.future.2018.06.040
10. Naik N. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP // Proc. 3rd Annual IEEE International Symposium on Systems Engineering (ISSE 2017). 2017. P. 8088251. doi: 10.1109/SysEng.2017.8088251
11. Yokotani T., Sasaki Y. Comparison with HTTP and MQTT on required network resources for IoT // Proc. 2nd International Conference on Control, Electronics, Renewable Energy, and Communications (ICCEREC 2016). 2016. P. 7814989. doi: 10.1109/ICCEREC.2016.7814989
12. Bonetto R., Bui N., Lakkundi V., Olivereau A., Serbanati A., Rossi M. Secure communication for smart IoT objects: Protocol stacks, use cases and practical examples // Proc. 13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2012). 2012. P. 6263790. doi: 10.1109/WoWMoM.2012.6263790
13. Dinculeană D., Cheng X. Vulnerabilities and limitations of MQTT protocol used between iot devices // Applied Sciences. 2019. V. 9. N 5. P. 848. doi: 10.3390/app9050848
14. Atmoko R.A., Riantini R., Hasin M.K. IoT real time data acquisition using MQTT protocol // Journal of Physics: Conference Series. 2017. V. 853. N 1. P. 012003. doi: 10.1088/1742-6596/853/1/012003
15. Шматков В.Н., Бонковски П., Медведев Д.С., Корзухин С.В., Голендухин Д.В., Спыну С.Ф., Муромцев Д.И. Взаимодействие с устройствами Интернета вещей с использованием голосового интерфейса // Научно-технический вестник информационных технологий, механики и оптики. 2019. Т. 19. № 4. С. 714–721. doi: 10.17586/2226-1494-2019-19-4-714-721

International Journal of Open Information Technologies, 2019, vol. 7, no. 5, pp. 17–28. (in Russian)
8. Hwang H.C., Park J., Shon J.G. Design and implementation of a reliable message transmission system based on MQTT protocol in IoT. *Wireless Personal Communications*, 2016, vol. 91, no. 4, pp. 1765–1777. doi: 10.1007/s11277-016-3398-2
9. Roy D.G., Mahato B., De D., Buyya R. Application-aware end-to-end delay and message loss estimation in Internet of Things (IoT) – MQTT-SN protocols. *Future Generation Computer Systems*, 2018, vol. 89, pp. 300–316. doi: 10.1016/j.future.2018.06.040
10. Naik N. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP // *Proc. 3rd Annual IEEE International Symposium on Systems Engineering (ISSE 2017)*, 2017, pp. 8088251. doi: 10.1109/SysEng.2017.8088251
11. Yokotani T., Sasaki Y. Comparison with HTTP and MQTT on required network resources for IoT. *Proc. 2nd International Conference on Control, Electronics, Renewable Energy, and Communications (ICCEREC 2016)*, 2016, pp. 7814989. doi: 10.1109/ICCEREC.2016.7814989
12. Bonetto R., Bui N., Lakkundi V., Olivereau A., Serbanati A., Rossi M. Secure communication for smart IoT objects: Protocol stacks, use cases and practical examples. *Proc. 13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2012)*, 2012, pp. 6263790. doi: 10.1109/WoWMoM.2012.6263790
13. Dinculeană D., Cheng X. Vulnerabilities and limitations of MQTT protocol used between iot devices. *Applied Sciences*, 2019, vol. 9, no. 5, pp. 848. doi: 10.3390/app9050848
14. Atmoko R.A., Riantini R., Hasin M.K. IoT real time data acquisition using MQTT protocol. *Journal of Physics: Conference Series*, 2017, vol. 853, no. 1, pp. 012003. doi: 10.1088/1742-6596/853/1/012003
15. Shmatkov V.N., Bąkowski P., Medvedev D.S., Korzukhin S.V., Golendukhin D.V., Spynu S.F., Mouromtsev D.I. Interaction with Internet of Things devices by voice control. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2019, vol. 19, no. 4, pp. 714–721. (in Russian). doi: 10.17586/2226-1494-2019-19-4-714-721

Авторы

Корзухин Сергей Владиславович — студент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, ORCID ID: 0000-0003-3163-9061, Sergey.korzukhin@gmail.com

Хайдарова Реседа Раитовна — инженер-исследователь, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, Scopus ID: 57031874500, ORCID ID: 0000-0001-8270-9192, mignolowa@gmail.com

Шматков Владислав Николаевич — заведующий лабораторией, инженер-исследователь, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, ORCID ID: 0000-0003-1391-1455, shmatkovvlad@gmail.com

Authors

Sergey V. Korzukhin — Student, ITMO University, Saint Petersburg, 197101, Russian Federation, ORCID ID: 0000-0003-3163-9061, Sergey.korzukhin@gmail.com

Rezeda R. Khaydarova — Research Engineer, ITMO University, Saint Petersburg, 197101, Russian Federation, Scopus ID: 57031874500, ORCID ID: 0000-0001-8270-9192, mignolowa@gmail.com

Vladislav N. Shmatkov — Laboratory Head, Research Engineer, ITMO University, Saint Petersburg, 197101, Russian Federation, ORCID ID: 0000-0003-1391-1455, shmatkovvlad@gmail.com