Russian Technological Journal.
2026;14(2):17–28

ISSN 2782-3210 (Print)
ISSN 2500-316X (Online)

УДК 004.415.53
https://doi.org/10.32362/2500-316X-2026-14-2-17-28
EDN XDBAIA

НАУЧНАЯ СТАТЬЯ

Архитектура распределенной системы тестирования устройств интернета вещей на этапе их разработки

Д.С. Беляков @

Томский государственный университет систем управления и радиоэлектроники, Томск, 634050 Россия
@ *Автор для переписки, e-mail: cauze4concern@yandex.ru*

• **Поступила:** 11.07.2025 • **Доработана:** 15.10.2025 • **Принята к опубликованию:** 06.02.2026

Резюме

Цели. Цель работы заключается в разработке архитектуры распределенной системы тестирования устройств интернета вещей (Internet of Things, IoT), обеспечивающей защищенную передачу тестовых сценариев и их изолированное исполнение на исполнительных модулях. Актуальность исследования обусловлена стремительным ростом числа IoT-устройств, функционирующих в недоверенных вычислительных средах, где процесс тестирования может создавать риски утечки конфиденциальных данных или несанкционированного вмешательства в программное обеспечение.

Методы. Проведен сравнительный анализ существующих решений, таких как *NI TestStand*, *MagicDAQ*, *PHiLIP* и *KEOLABS ContactLAB*. Выполнено сопоставление их архитектурных компонентов и процессов жизненного цикла тестовых сценариев.

Результаты. На основании анализа выделены основные этапы жизненного цикла, на которых применяются рассмотренные инструменты: подготовка и хранение, передача и интерпретация, взаимодействие с тестируемым устройством, регистрация и анализ результатов. Кроме того, проведено сравнение существующих и предложенного архитектурных решений по ключевым характеристикам: предметная область применения, тип архитектуры (распределенная или централизованная), среда исполнения тестовых сценариев, масштабируемость системы, уровень изоляции среды исполнения, наличие механизмов защиты и возможность удаленного управления. Результаты работы представлены в виде предложенной архитектуры, включающей управляющий модуль и автономные исполнительные модули с изолированной виртуальной средой исполнения MicroPython. Для обеспечения безопасности предусмотрена передача тестовых сценариев по зашифрованному каналу связи с использованием протоколов CoAP[1] и DTLS[2] , а также выполнение кода тестовых сценариев в ограниченной среде, изолированной от основной операционной системы.

[1] Constrained application protocol – облегченный протокол интернета вещей.
[2] Datagram transport layer security – протокол передачи данных, обеспечивающий защищенность соединений для протоколов, использующих датаграммы.

Выводы. Проведенный сравнительный анализ продемонстрировал, что предлагаемое решение устраняет ключевые ограничения аналогов, связанные с отсутствием механизмов шифрования и изоляции исполнения. Разработанная архитектура повышает безопасность и надежность процесса тестирования IoT-устройств и может использоваться в недоверенных вычислительных средах для защиты интеллектуальной собственности и логики тестовых сценариев.

Ключевые слова: интернет вещей, IoT, функциональное тестирование, архитектура тестирования, тестовые сценарии

Для цитирования: Беляков Д.С. Архитектура распределенной системы тестирования устройств интернета вещей на этапе их разработки. *Russian Technological Journal*. 2026;14(2):17–28. https://doi.org/10.32362/2500-316X-2026-14-2-17-28, https://www.elibrary.ru/XDBAIA

Прозрачность финансовой деятельности: Автор не имеет финансовой заинтересованности в представленных материалах или методах.

Автор заявляет об отсутствии конфликта интересов.

RESEARCH ARTICLE

Architecture of a distributed system for testing Internet of Things devices at the development stage

Danila S. Belyakov @

Tomsk State University of Control Systems and Radioelectronics, Tomsk, 634050 Russia
@ *Corresponding author, e-mail: cauze4concern@yandex.ru*

• **Submitted:** 11.07.2025 • **Revised:** 15.10.2025 • **Accepted:** 06.02.2026

Abstract

Objectives. The paper sets out to develop an architecture for a distributed testing system for Internet of Things (IoT) devices to ensure secure transmission and the isolated execution of test scenarios on dedicated execution modules. The study takes account of the rapid growth in the number of IoT devices operating in untrusted computing environments, in which the testing process can pose a risk of confidential data leakage or unauthorized interference with software components.

Methods. A comparative analysis of existing solutions such as *NI TestStand*, *MagicDAQ*, *PHiLIP*, and *KEOLABS ContactLAB* was conducted. Architectural components and test scenario life-cycle processes were examined and compared.

Results. The analysis identified the main stages of the test scenario life cycle, including preparation and storage of scripts, transmission and interpretation, interaction with the device under test, as well as registration and analysis of results. In addition, existing and proposed architectural solutions were compared according to the following key characteristics: application domain; type of architecture (distributed or centralized); test scenario execution environment; system scalability; level of execution isolation; availability of protection mechanisms; capability for remote management. The results of the study are presented in the form of a proposed architecture that includes a control module and autonomous execution modules with an isolated virtual MicroPython environment. To ensure security, test scenarios are transmitted over an encrypted communication channel using constrained application protocol and datagram transport layer security (protocol, while the execution of test code takes place in a restricted environment isolated from the main operating system.

Conclusions. The comparative analysis confirmed that the proposed solution eliminates the key limitations of existing solutions, namely the lack of encryption mechanisms and isolation of execution. The developed architecture enhances the security and reliability of the IoT device testing process, offering protection for intellectual property and test scenario logic in untrusted computing environments.

Keywords: Internet of Things, IoT, functional testing, testing architecture, test scenarios

For citation: Belyakov D.S. Architecture of a distributed system for testing Internet of Things devices at the development stage. *Russian Technological Journal.* 2026;14(2):17–28. https://doi.org/10.32362/2500-316X-2026-14-2-17-28, https://www.elibrary.ru/XDBAIA

Financial disclosure: The author has no financial or proprietary interest in any material or method mentioned.

The author declares no conflicts of interest.

ВВЕДЕНИЕ

Массовое распространение устройств интернета вещей обусловливает необходимость соблюдения высоких требований к их качеству, безопасности и надежности. В этих условиях тестирование приобретает особую значимость, т.к. позволяет не только выявить и устранить потенциальные неисправности, но и убедиться в том, что устройства соответствуют заданным техническим требованиям и способны стабильно функционировать в реальных условиях эксплуатации [1, 2].

В отличие от традиционного программного обеспечения, тестирование устройств интернета вещей сопровождается рядом специфических сложностей. Оно охватывает как аппаратные, так и программные компоненты, требует учета множества коммуникационных протоколов, ограниченных вычислительных ресурсов и особенностей низкоуровневых интерфейсов. Дополнительную сложность вносит тот факт, что тестирование нередко передается на аутсорсинг сторонним организациям, которые не всегда являются доверенными, что создает риски несанкционированного вмешательства и компрометации результатов [3].

В связи с этим цель данного исследования заключается в анализе существующих решений, направленных на тестирование устройств интернета вещей, и в разработке новой архитектуры распределенной системы тестирования, учитывающей выявленные недостатки.

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ТЕСТИРОВАНИЯ

С учетом особенностей устройств интернета вещей, тестирование можно разделить на три основных уровня [4–6]: устройство, сеть и система.

Тестирование устройства сосредоточено на проверке самого устройства как отдельной единицы. Оно охватывает как аппаратные компоненты (микроконтроллер, датчики, исполнительные модули), так и встроенное программное обеспечение. В первую очередь проверяется корректность работы периферийных интерфейсов (SPI[3], UART[4] и др.), обеспечивающих связь микроконтроллера с внешними модулями. Во вторую очередь оцениваются логика прошивки, алгоритмы обработки данных, производительность и надежность работы устройства в условиях нагрузок или помех.

Тестирование сети фокусируется на коммуникационной инфраструктуре. Измеряются такие параметры, как пропускная способность, задержки, надежность и масштабируемость используемых каналов связи (Wi-Fi, сотовые сети, LPWAN[5] и др.), чтобы гарантировать необходимое качество обслуживания устройств интернета вещей [7].

Тестирование системы, напротив, охватывает весь комплекс устройств и инфраструктуру: проверяется взаимодействие нескольких устройств друг с другом, с шлюзами и облачной платформой, а также корректность сквозной обработки данных и пользовательских сценариев. Это комплексное тестирование интеграции, при котором отрабатываются сценарии от события на устройстве до получения данных в облаке и обратной реакции.

Таким образом, каждый уровень тестирования необходим для комплексной оценки функционирования систем интернета вещей, поскольку охватывает различные аспекты их работы – от аппаратных компонентов до взаимодействия устройств в распределенной среде.

2. АНАЛИЗ АРХИТЕКТУРЫ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Инструменты тестирования представляют собой программные или аппаратно-программные средства, предназначенные для автоматизации и упрощения

[3] Serial peripheral interface – последовательный периферийный интерфейс.

[4] Universal asynchronous receiver-transmitter – универсальный асинхронный приемопередатчик.

[5] Low-power wide-area network – энергоэффективная сеть дальнего радиуса действия.

процесса проверки устройств интернета вещей. Их основная задача – поддержка специалистов при оценке различных характеристик, включая функциональность, производительность, устойчивость и безопасность. Такие инструменты позволяют воспроизводить широкий спектр сценариев эксплуатации и моделировать условия среды, обеспечивая тем самым более достоверную проверку поведения устройств в различных условиях.

В большинстве современных исследований и разработок в области тестирования интернета вещей основное внимание уделяется системному или сетевому уровням тестирования. Наиболее широко представлены решения, ориентированные на проверку корректности реализации протоколов (CoAP[6], MQTT[7], 6LoWPAN[8] и др.), совместимости узлов, производительности и надежности сетевых соединений. Платформы, такие как PatrIoT [8], Hector [9], F-Interop [10], MATTER [11], Eclipse IoT-Testware [12], как правило, используют механизмы виртуализации, что позволяет проводить масштабируемое тестирование архитектурных сценариев без участия реального физического оборудования.

Реализация проверок на уровне физических устройств обычно ограничивается узкоспециализированными решениями и менее формализована. Поэтому представляется целесообразным рассмотреть архитектуры систем, ориентированных именно на уровень устройства, и провести их сопоставление. Базовая архитектура системы тестирования, представленная на рис. 1, включает следующие компоненты:

- управляющий модуль – осуществляет координацию запуска тестов, а также распределение заданий между исполнительными модулями;
- исполнительный модуль – одно или несколько устройств, непосредственно взаимодействующих с тестируемым устройством через физические интерфейсы;
- тестируемое устройство – объект, в отношении которого проводится тестирование.

В результате анализа предметной области выявлены инструменты, применяемые на уровне тестирования устройств.

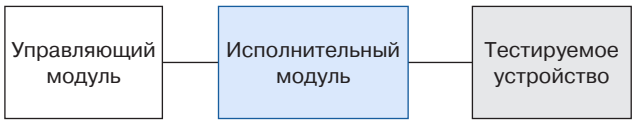

Рис. 1. Типовая архитектура системы тестирования

NI TestStand[9] представляет собой коммерческую систему управления тестированием, которая интегрируется со средой программирования измерительных процессов *NI LabView*. Архитектура системы представлена на рис. 2.

Управляющий модуль представляет собой программу, установленную на персональном компьютере (ПК) и предназначенную для централизованного управления автоматизированными исполнительными модулями. Управляющий модуль также занимается регистрацией результатов тестирования, обеспечивая сбор и визуализацию результатов.

Исполнительные модули представляют собой системы сбора данных, подключенные к управляющему модулю, которые взаимодействуют непосредственно с тестируемым устройством через физические интерфейсы, выполняя команды тестирования.

Среда исполнения сценариев находятся на управляющем модуле непосредственно в операционной системе (ОС). Безопасность системы реализуется через встроенные функции ОС, обеспечивая контроль ресурсов и результатов выполнения тестирования.

Функциональные возможности *NI TestStand* включают создание тестовых сценариев, которые могут выполняться как для одного устройства, так и для нескольких устройств одновременно, при этом тестирование координируется посредством параллельного взаимодействия нескольких управляющих модулей через локальную сеть. Однако такое взаимодействие ограничено внутренней сетью предприятия или лаборатории, т.к. система изначально не предусматривает полноценного удаленного управления через интернет-соединение. Тем не менее, *NI TestStand* предоставляет командный интерфейс, который позволяет инициировать запуск тестирования из внешних систем непрерывной интеграции (continuous integration, CI).

Таким образом, *NI TestStand* представляет собой локальную централизованную систему тестирования с развитым пользовательским интерфейсом и поддержкой процессов непрерывной интеграции, однако не обладающую встроенными механизмами распределенной обработки или изолированной работы тестовых модулей.

[6] Constrained application protocol – облегченный протокол интернета вещей.

[7] Message queuing telemetry transport – упрощенный протокол обмена данными. [Message queuing telemetry transport is a lightweight, publish–subscribe, machine-to-machine network protocol for message queue/message queuing service.]

[8] IPv6 over low power wireless personal area networks – стандарт взаимодействия по протоколу IPv6 поверх маломощных беспроводных персональных сетей стандарта IEEE 802.15.4. [IPv6 over Low-Power Wireless Personal Area Networks, based on the IEEE 802.15.4 standard.]

[9] TestStand Release Notes. https://www.ni.com/en/support/documentation/release-notes/product.teststand.html. Дата обращения 07.07.2025. / Accessed July 07, 2025.

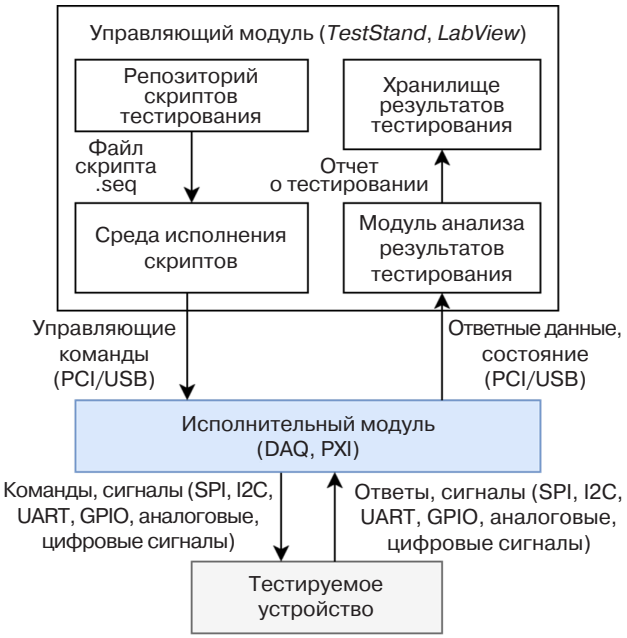

Рис. 2. Архитектура тестирования *NI TestStand* для одного устройства. PCI (peripheral component interconnect) – шина ввода-вывода для подключения периферийных устройств к материнской плате компьютера; USB (universal serial bus) – универсальная последовательная шина; I2C (inter-integrated circuit) – последовательная асимметричная шина; GPIO (general purpose input/output) – интерфейс ввода/вывода общего назначения; DAQ (data acquisition) – программно-аппаратный комплекс для сбора данных; PXI (PCI eXtensions for Instrumentation) – расширение PCI для измерительных систем

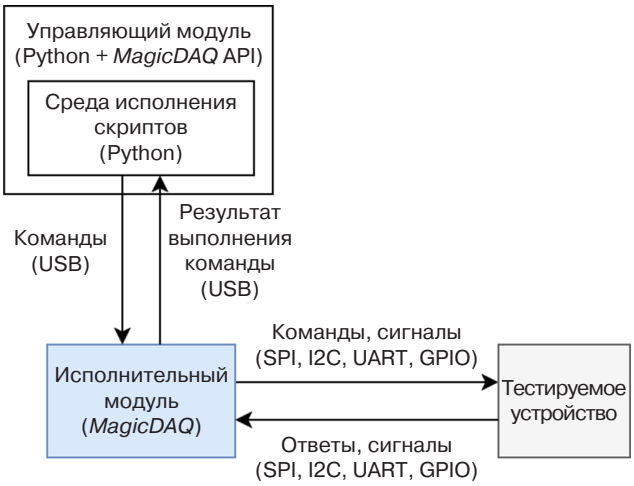

Рис. 3. Архитектура тестирования *MagicDAQ*. API (application programming interface) – программный интерфейс приложений

MagicDAQ[10] представляет собой исполнительный модуль, который соединяется по интерфейсу USB с управляющим модулем (ПК), как показано на рис. 3. Вся логика тестирования реализуется на управляющем модуле путем выполнения тестовых сценариев на языке Python, которые отправляют команды на исполнительный модуль по USB. *MagicDAQ* не является полноценно распределенной системой или облачным сервисом – это периферийный модуль для локального ПК. Масштабируемость определяется количеством доступных портов USB. Поскольку сценарии тестов пишутся на Python и передаются через локальную связь, безопасность реализуется через встроенные средства ОС.

Таким образом, *MagicDAQ* характеризуется как специализированное решение для локального применения в измерительных системах с ограниченными возможностями масштабирования и отсутствием механизмов распределенной обработки данных.

PHiLIP [8] представляет собой программно-аппаратную платформу, разработанную для автоматизированного тестирования периферийных интерфейсов микроконтроллеров во встроенных системах, включая устройства интернета вещей.

Архитектура системы тестирования *PHiLIP* построена на распределенной модели, в которой центральным управляющим модулем выступает сервер непрерывной интеграции, например, Jenkins, интегрированный с системой контроля версий Git для хранения и актуализации тестовых сценариев. Как показано на рис. 4, управляющий модуль отвечает за координацию тестовых запусков, хранение и распределение тест-кейсов, передачу прошивок на тестируемые устройства и сбор данных о результатах тестирования для их дальнейшего анализа.

Каждый исполнительный модуль реализован на базе одноплатного компьютера Raspberry Pi (производитель – Raspberry Pi Holdings plc, Великобритания) и выполняет роль промежуточного звена между CI-сервером и аппаратным интерфейсным модулем *PHiLIP*. На исполнительном модуле осуществляется запуск тестовых сценариев в среде Robot Framework[11], а также выполняется прошивка тестируемых устройств через интерфейсы SWD[12] или JTAG[13] для загрузки новых версий программного обеспечения. Взаимодействие с интерфейсным модулем *PHiLIP* происходит по интерфейсу UART.

Интерфейсный модуль *PHiLIP*, в свою очередь, отвечает за генерацию и регистрацию сигналов по физическим интерфейсам тестируемого устройства,

[10] *MagicDAQ* Docs. https://magicdaq.github.io/magicdaq_docs/. Дата обращения 07.07.2025. / Accessed July 07, 2025.

[11] Robot Framework. https://robotframework.org/. Дата обращения 07.07.2025. / Accessed July 07, 2025.

[12] Serial wire debug – двухпроводной протокол отладки.

[13] Joint test action group – специализированный аппаратный интерфейс.

включая SPI, I2C, UART и GPIO. Он эмулирует поведение внешней среды устройства и фиксирует ответы тестируемого устройства, которые затем возвращаются к исполнительному модулю и далее в управляющий модуль для анализа результатов тестирования.

Архитектура системы допускает масштабирование, как показано на рис. 5: к одному управляющему модулю могут быть подключены несколько исполнительных модулей, каждый из которых управляет своим экземпляром интерфейсного модуля и тестируемым устройством. Такая структура позволяет выполнять параллельное тестирование разных устройств, обеспечивая при этом централизованное управление и сбор результатов. Однако в работе [8] не описаны механизмы обеспечения информационной безопасности: не уточнено, каким образом обеспечивается изоляция исполнения тест-кейсов, защита от несанкционированного доступа и контроль целостности при передаче тестовых сценариев.

KEOLABS ContactLAB[14] представляет собой коммерческую программно-аппаратную платформу, предназначенную для функционального тестирования смарт-карт, микроконтроллеров с модулями безопасности (Secure Element), а также устройств с поддержкой бесконтактных интерфейсов NFC[15] и ISO 7816[16]. Платформа активно используется в сертификационных испытаниях и при разработке средств аутентификации и идентификации.

Архитектура решения представлена на рис. 6 и основана на использовании программного управляющего модуля *SCRIPTIS*[17], установленного на ПК, и аппаратного исполнительного модуля *ContactLAB HW*, выполняющего генерацию команд, измерение временных характеристик и регистрацию сигналов от тестируемого устройства. Взаимодействие между управляющим и исполнительным модулем происходит по интерфейсу USB или Ethernet. Исполнение тестов полностью сосредоточено на одном рабочем месте без возможности распределенного тестирования или удаленного доступа.

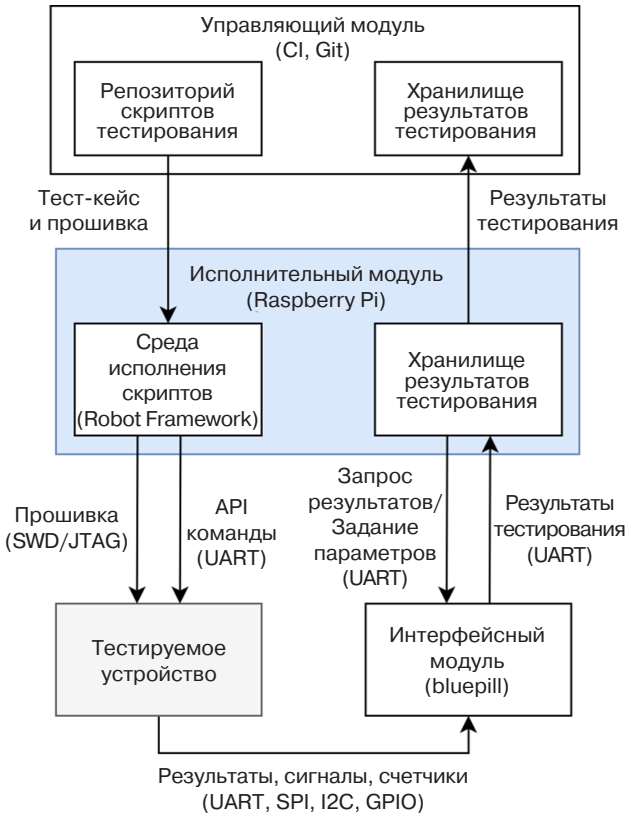

Рис. 4. Архитектура тестирования *PHiLIP* для одного тестируемого устройства

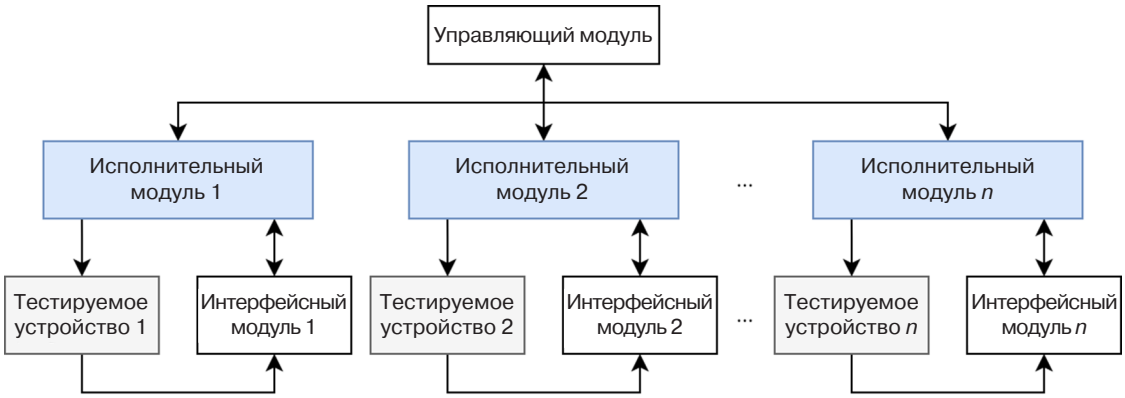

Рис. 5. Архитектура тестирования *PHiLIP* для нескольких тестируемых устройств

[14] Contact Tester. https://www.keolabs.com/products/platforms/contact-tester. Дата обращения 07.07.2025. / Accessed July 07, 2025.

[15] Near field communication – технология беспроводной передачи данных малого радиуса действия. [Near field communication (NFC) is a technology for transmitting data wirelessly over short distances.]

[16] ISO/IEC 7816-1 Identification cards – Integrated circuit cards. https://www.iso.org/standard/54089.html. Дата обращения 07.07.2025. / Accessed July 07, 2025.

[17] *SCRIPTIS*: an intuitive testing environment. https://www.keolabs.com/products/solutions/emvco-l1-payment-testing#. Дата обращения 07.07.2025. / Accessed July 07, 2025.

Масштабируемость возможна лишь путем увеличения количества стендов и ручной координации их работы. Архитектурно не предусмотрены ни централизованный управляющий модуль, ни параллельное выполнение сценариев на множестве устройств.

Безопасность логики тестирования обеспечивается в рамках закрытой среды *SCRIPTIS* и встроенных в нее механизмов доступа. Выполнение тестовых сценариев осуществляется в стандартной пользовательской среде ОС.

Таким образом, *KEOLABS* представляет собой платформу низкоуровневого тестирования защищенных устройств в условиях лабораторной среды. Ее архитектура ориентирована на централизованную эксплуатацию и не имеет механизмов удаленного доступа или защищенного распределенного исполнения тестов.

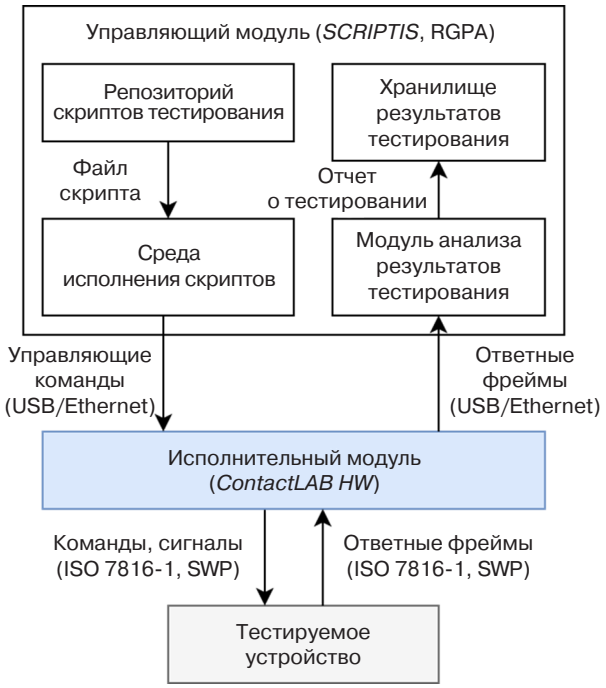

Рис. 6. Архитектура тестирования *KEOLABS ContactLAB.* RGPA (real-time general purpose analyzer) – анализатор общего назначения реального времени; SWP (single wire protocol) – однопроводной протокол

3. АНАЛИЗ ПРОЦЕССОВ ТЕСТИРОВАНИЯ

На основании проведенного обзора архитектур существующих систем тестирования можно выделить ряд типовых процессов. Эти процессы, представленные на рис. 7, формируют общий сценарий функционирования подобных систем независимо от конкретной реализации. Такие процессы можно определить как последовательность этапов жизненного цикла тестового сценария:

1) подготовка и хранение тестовых сценариев;
2) планирование и запуск тестовых сценариев;
3) передача тестовых сценариев на исполнительный модуль;
4) интерпретация тестовых сценариев;
5) взаимодействие с тестируемым объектом;
6) регистрация результатов тестирования;
7) анализ и формирование отчетов.

На этапе подготовки и хранения тестовых сценариев разрабатываются и описываются сценарии, представляющие собой последовательность действий, условий выполнения и критериев успешного прохождения теста. Сценарии создаются специалистами с использованием скриптовых языков программирования, таких как Python (например, в *MagicDAQ*), сценариев Robot Framework (в *PHiLIP*) и других средств. После разработки сценарии сохраняются в базе данных или репозитории для последующего использования управляющим модулем или системами непрерывной интеграции (например, Jenkins).

Планирование и запуск тестовых сценариев включает выбор необходимого набора тестов, определение последовательности их выполнения, синхронизацию и инициирование процесса тестирования. Запуск может осуществляться оператором через интерфейс управляющего модуля или автоматически из внешней системы непрерывной интеграции, как в случае с *PHiLIP*, использующим Jenkins.

Передача тестовых сценариев на исполнительный модуль в рассмотренных системах осуществляется через локальную сеть (например,

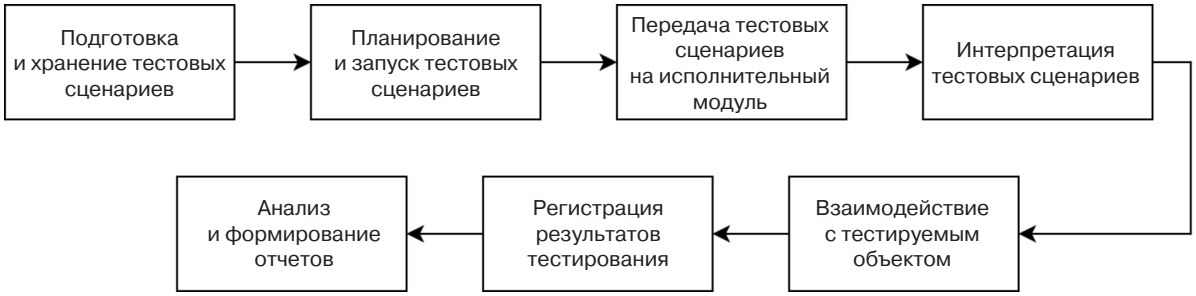

Рис. 7. Жизненный цикл тестового сценария

PHiLIP) или физические интерфейсы, такие как USB (*NI TestStand*, *MagicDAQ*, *KEOLABS ContactLAB*) и Ethernet (*KEOLABS ContactLAB*). При этом в проанализированных решениях не уточняется, применяются ли какие-либо механизмы шифрования или иные меры защиты передаваемых данных.

На этапе интерпретации тестовых сценариев происходит обработка сценария в среде исполнения, где логика теста преобразуется в конкретные команды для управления физическими интерфейсами. В зависимости от системы это может быть движок TestStand, Python-интерпретатор (в *MagicDAQ*), среда Robot Framework (в *PHiLIP*) или встроенная среда *SCRIPTIS* (в *KEOLABS ContactLAB*). Такой подход позволяет абстрагировать логику теста от аппаратной реализации.

Взаимодействие с тестируемым объектом осуществляется исполнительным модулем, который управляет физическими интерфейсами (SPI, I2C, UART, ISO 7816 и др.) и передает сигналы и команды непосредственно к тестируемому устройству. Например, в *PHiLIP* для этого используется отдельный интерфейсный модуль, взаимодействующий с Raspberry Pi и зависящий от него как управляющего модуля.

Регистрация результатов тестирования заключается в фиксации данных, полученных в ходе теста: откликов устройства, сигналов, временных меток, статусов и измеренных значений. Эти данные могут храниться локально на исполнительном модуле или сразу передаваться на управляющий модуль для последующего анализа.

На завершающем этапе – анализе и формировании отчетов – проводится обработка зарегистрированных данных, вычисление критериев успешного прохождения тест-кейса, формирование отчетов для операторов или автоматическая передача результатов во внешние системы анализа и управления качеством.

Анализ существующих платформ тестирования устройств интернета вещей и выделенных типовых процессов показал, что они обладают рядом ограничений, все рассмотренные решения объединяет ключевое ограничение – отсутствие механизма защищенного исполнения тестовых сценариев. Сценарии тестирования либо исполняются локально, либо доставляются вручную или через инфраструктуру непрерывной интеграции без явного использования криптографической защиты. Также многие инструменты являются локальными и не имеют поддержки удаленных исполнительных модулей. В современных условиях, где тестирование часто осуществляется в недоверенной среде и требует автономной работы исполнительных модулей, необходимо обеспечить возможность работы проведения тестов без риска вмешательства или конфликта.

Таким образом, необходимо разработать новую архитектуру тестирования, которая устранит выявленные недостатки и обеспечит необходимый уровень безопасности и масштабируемости при тестировании устройств интернета вещей.

4. ПРЕДЛАГАЕМАЯ АРХИТЕКТУРА СИСТЕМЫ ТЕСТИРОВАНИЯ

Предлагаемая архитектура представляет собой распределенную сеть тестирования устройств интернета вещей, в которой несколько исполнительных модулей взаимодействуют с управляющим модулем и обмениваются зашифрованными тестовыми сценариями. Архитектура представлена на рис. 8 и включает в себя следующие ключевые компоненты:

- управляющий модуль – отвечает за координацию распределенных исполнительных модулей, хранение тестовых сценариев;
- исполнительные модули – автономные узлы, физически подключенные к тестируемым устройствам. Модули выполняют полученные от управляющих модулей тестовые сценарии, регистрируют результаты и взаимодействуют с тестируемым объектом через подключенные интерфейсы. Каждая исполнительная станция содержит встроенную изолированную среду исполнения, в которой интерпретируются полученные от сервера сценарии тестирования;
- тестируемые устройства – подключенные к исполнительным модулям реальные устройства интернета вещей, тестируемые по физическим интерфейсам. Каждая испытательная станция подключается к тестируемому устройству через один из интерфейсов (SPI, I2C, UART и др.). Эти подключения в предлагаемой архитектуре обозначаются как каналы, каждый из которых представляет собой логически независимую единицу управления входными и выходными сигналами. Канал соответствует определенному типу интерфейса или протоколу взаимодействия.

Процесс тестирования начинается с того, что оператор инициирует выполнение сценария через управляющий модуль. Сценарий передается на исполнительный модуль по защищенному каналу, где исполняется в изолированной среде, формирующей команды для взаимодействия с интерфейсами тестируемого устройства. Результаты тестирования в зашифрованном виде пересылаются обратно на управляющий модуль для последующего анализа.

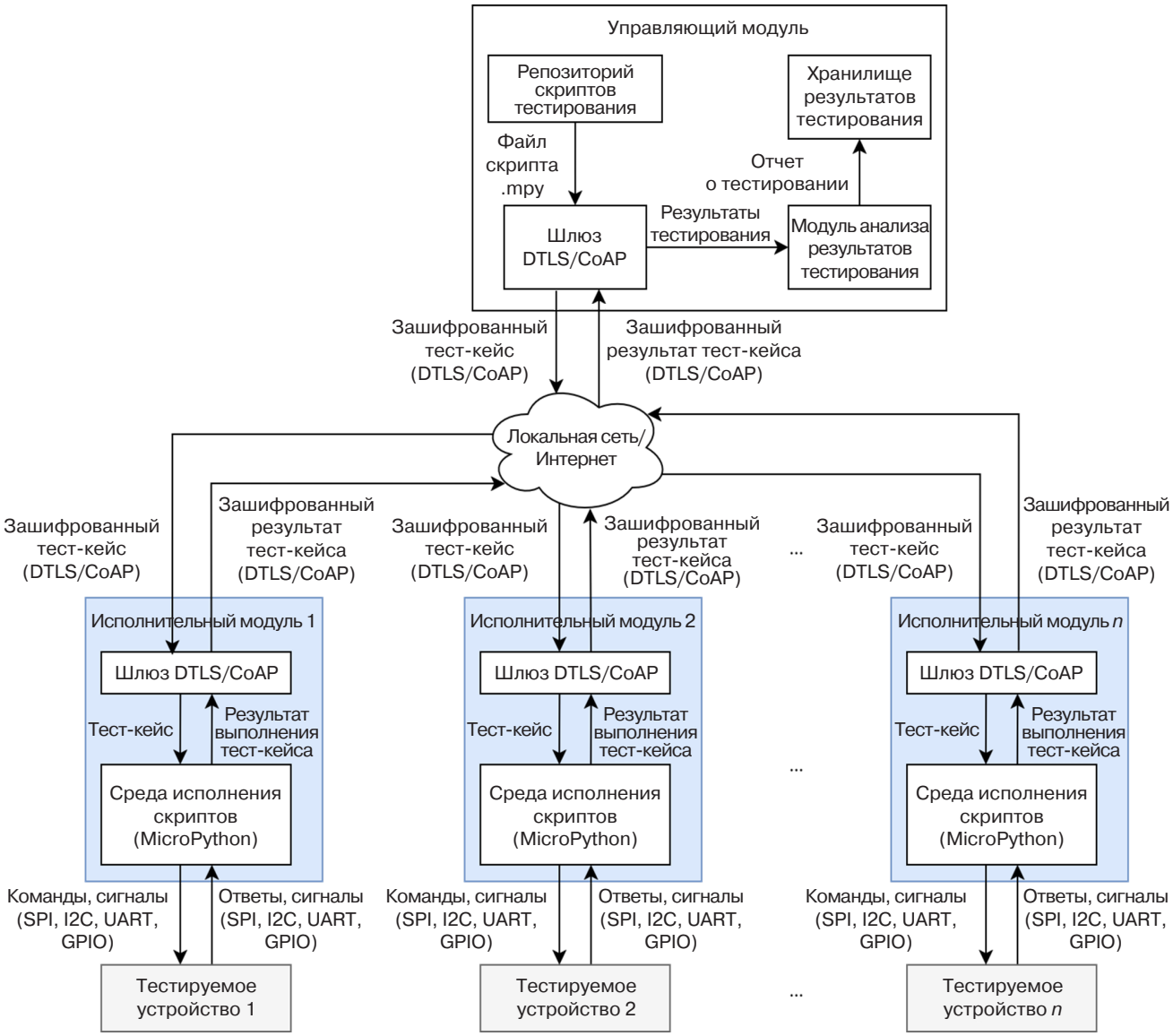

Рис. 8. Архитектура системы тестирования

При проектировании архитектуры особое внимание уделялось защите логики тестирования и снижению рисков, возникающих при эксплуатации в потенциально недоверенной среде. Как показано в исследованиях [13, 14], этапы жизненного цикла устройств сопровождаются угрозами несанкционированного доступа к программному обеспечению и утечки сценариев тестирования, а сами микроконтроллеры могут выступать как источником, так и объектом атак. Учитывая эти угрозы, архитектура системы предусматривает применение комплекса мер защиты на разных этапах жизненного цикла тестовых сценариев.

Для обеспечения конфиденциальности и целостности при передаче тестовых сценариев, а также при получении результатов тестирования используется протокол DTLS. Применение DTLS позволяет обеспечить защиту данных при работе в недоверенных сетях, что особенно важно для устройств с ограниченными вычислительными ресурсами [15].

Исполнение сценариев тестирования осуществляется в изолированной среде MicroPython[18]. Такая изоляция обеспечивает безопасное выполнение тестовых сценариев за счет компартментализации и ограничения доступа к ресурсам [16].

Система поддерживает масштабируемость за счет подключения новых исполнительных модулей, каждый из которых работает автономно. Это обеспечивает параллельное выполнение тестов и исключает единую точку отказа.

[18] Micropython. https://micropython.org/. Дата обращения 07.07.2025. / Accessed July 07, 2025.

В таблице приведены характеристики существующих и предложенного архитектурных решений. Как видно из таблицы, большинство существующих систем характеризуется централизованной архитектурой, ограниченными возможностями масштабирования и отсутствием механизмов защиты. Предлагаемое решение отличается распределенной архитектурой сети исполнительных модулей, применением изолированной среды выполнения и реализацией механизмов шифрования и аутентификации. Такой подход гарантирует не только эффективное масштабирование системы, но и существенно повышает уровень защиты тестовой логики.

Предлагаемое решение расширяет и усиливает отдельные процессы жизненного цикла тестового сценария. Рассмотрим ключевые особенности и отличия от других решений:

- Передача тестовых сценариев на исполнительный модуль. В отличие от аналогов, где передача осуществляется по USB/Ethernet (*NI TestStand*, *MagicDAQ*, *KEOLABS ContactLAB*) или локальной сети (*PHiLIP*) без использования механизмов защиты передачи данных, в предлагаемом решении сценарии передаются с управляющего модуля на исполнительные модули в зашифрованном виде с использование протокола DTLS. Это позволяет обеспечить конфиденциальность и целостность тестовых сценариев даже при эксплуатации в потенциально недоверенной сети.
- Интерпретация тестовых сценариев. Сценарии описываются в виде Python-скриптов и выполняются на виртуальной машине MicroPython на исполнительном модуле, изолированном от основной ОС. Такая изоляция не предусматривается в рассмотренных ранее решениях (*NI TestStand*, *MagicDAQ*), где исполнение происходит в общей пользовательской среде, что создает риск вмешательства в логику тестирования.
- Регистрация результатов тестирования. Во многих текущих решениях результаты тестирования остаются локальными (*KEOLABS ContactLAB*) или обрабатываются вручную (*MagicDAQ*), без единого централизованного анализа. Предлагаемая архитектура предусматривает отправку зашифрованных отчетов от модулей в управляющий сервер.

Эффективность и работоспособность предложенной архитектуры подтверждены в ходе интеграционных испытаний разработанной системы тестирования функционирования устройств. Испытания проводились на экспериментальном стенде, включающем управляющий модуль, несколько исполнительных модулей и тестируемое устройство. Управляющий модуль был реализован на ПК, а исполнительные модули – на платах SE-Discovery-GD32F427 (производитель – GigaDevice Semiconductor Inc., Китай) со встроенным программным обеспечением и средой MicroPython. Взаимодействие между компонентами осуществлялось по защищенным сетевым каналам, реализованным в рамках локальной сети,

Таблица. Сопоставление характеристик существующих и предложенного решения

Характеристика	*NI TestStand*	*MagicDAQ*	*PHiLIP*	*KEOLABS*	Предлагаемое решение
Тип архитектуры	Централизованная	Централизованная	Распределенная	Централизованная	Распределенная
Среда исполнения	ПК	ПК	Исполнительный модуль	ПК	Исполнительный модуль
Масштабируемость	Локальная сеть (удаленный запуск через CI)	Ограничена количеством портов USB	Высокая (добавление узлов Raspberry Pi)	Ограничено ручным развертыванием	Высокая (добавление исполнительных модулей)
Изоляция исполнения	ОС	ОС (Python)	Robot Framework	*SCRIPTIS*	MicroPython
Механизмы защиты	–	–	Не указано	–	DTLS
Удаленное управление	Да (CI)	Нет	Да (CI)	Нет	Да (управляющий модуль)
Область применения	Общие тесты электронных устройств	Общие тесты электронных устройств	API встроенных ОС	Смарт-карты, NFC	Устройства интернета вещей, интерфейсы микроконтроллера

при этом архитектура допускает возможность удаленного взаимодействия между модулями при подключении к сети Интернет. В ходе испытаний проверялись ключевые функции системы: защищенная передача тестовых сценариев и результатов выполнения, а также корректное функционирование виртуальной машины исполнительного модуля. Все тестовые сценарии завершились успешно, что подтвердило работоспособность и эффективность предложенной архитектуры.

Разработанное решение применяется для тестирования смарт-карт и аппаратных токенов, включая проверку корректности выполнения апплетов. В ходе испытаний моделировались сбои питания и нестабильность интерфейсов связи. Один исполнительный модуль управлял подачей питания на тестируемое устройство, а другой обеспечивал обмен с тестируемым устройством, что позволило воспроизвести реальные условия эксплуатации и подтвердить устойчивость системы к нештатным ситуациям.

ЗАКЛЮЧЕНИЕ

В статье предложена новая архитектура распределенной системы тестирования устройств интернета вещей, разработанная с учетом угроз эксплуатации в недоверенной среде. Проведенный анализ показал, что существующие системы тестирования устройств имеют ограниченную масштабируемость и не обеспечивают защищенной передачи и исполнения тестовых сценариев. Предложенная архитектура тестирования решает эти задачи за счет внедрения мер безопасности. Использование изолированных виртуальных сред исполнения и шифрованных каналов передачи защищает тестовые сценарии от несанкционированного доступа или подмены.

Таким образом, ключевые преимущества предлагаемой архитектуры тестирования – безопасность (шифрование, изоляция логики), масштабируемость (добавление узлов без изменения архитектуры) – делают ее подходящим решением для тестирования устройств интернета вещей в потенциально недоверенной среде.

СПИСОК ЛИТЕРАТУРЫ

1. Дьяков О.Н., Беляков Д.С., Калинин Е.О. Использование технологии ePKI для безопасного обновления встроенного программного обеспечения доверенных программно-аппаратных комплексов. *Безопасность информационных технологий.* 2025;32(2):152–177. https://doi.org/10.26583/bit.2025.2.12
2. Venugopal M., Nanda M., Anand G., Chandana Voora H. An integrated Hardware/Software Verification and Validation methodology for Signal Processing Systems. *ITM Web Conf.* 2022;50:02001. https://doi.org/10.1051/itmconf/20225002001
3. Bures M., Cerny T., Ahmed B.S. Internet of Things: Current Challenges in the Quality Assurance and Testing Methods. *arXiv*. arXiv:1805.01241. 2018. https://doi.org/10.48550/arXiv.1805.01241
4. Mazhar T., Talpur D.B., Shloul T.A., Ghadi Y.Y., Haq I., Ullah I. Analysis of IoT Security Challenges and its Solutions Using Artificial Intelligence. *Brain Sciences*. 2023;13(4):683. https://doi.org/10.3390/brainsci13040683
5. Minani J.B., Sabir F., Moha N., Guéhéneuc Y.G. A Multimethod Study of Internet of Things Systems Testing in Industry. *IEEE Internet Things J*. 2024;11(1):1662–1684. https://doi.org/10.1109/JIOT.2023.3291233
6. Папуловская Н.В., Изотов И.Н., Блиничкин Д.Ю., Катаев А.Ю. Разработка ядра платформы автоматизированного тестирования устройств интернета вещей. *Int. J. Open Inf. Technol.* 2021;9(6):38–45. https://elibrary.ru/ybxvtg
7. Castelo Branco K.D.S., Dantas V.L.L., Carvalho L.M. Interoperability Testing Guide for the Internet of Things. In: *Proceedings of the 30th Brazilian Symposium on Multimedia and the Web* (*WebMedia 2024*). (*Sociedade Brasileira de Computação*). 2024. P. 188–196. https://doi.org/10.5753/webmedia.2024.242058
8. Weiss K., Rottleuthner M., Schmidt T.C., Wählisch M. PHiLIP on the HiL: Automated Multi-Platform OS Testing with External Reference Devices. *ACM Trans. Embed. Comput. Syst*. (*TECS*). 2021;20(5s):1–26. https://doi.org/10.1145/3477040
9. Behnke I., Thamsen L., Kao O. Héctor: A Framework for Testing IoT Applications Across Heterogeneous Edge and Cloud Testbeds. In: *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion.* ACM; 2019. P. 15–20. https://doi.org/10.1145/3368235.3368832
10. Ziegler S., Fdida S., Viho C., Watteyne T. F-Interop – Online Platform of Interoperability and Performance Tests for the Internet of Things. In: Mitton N., Chaouchi H., Noel T., Watteyne T., Gabillon A., Capolsini P. (Eds.). *Interoperability, Safety and Security in IoT. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer; 2017. V. 190. P. 49–55. URL: https://link.springer.com/chapter/10.1007/978-3-319-52727-7_7. Accessed July 07, 2025.
11. Olianas D., Leotta M., Ricca F. MATTER: A tool for generating end-to-end IoT test scripts. *Software Qual. J.* 2021;30(2): 389–423. URL: https://link.springer.com/article/10.1007/s11219-021-09565-y. Accessed July 07, 2025.
12. Schieferdecker I., Kretzschmann S., Rennoch A., Wagner M. IoT-Testware – An Eclipse Project. In: *2017 IEEE International Conference on Software Quality, Reliability and Security* (*QRS*). IEEE; 2017. https://doi.org/10.1109/QRS.2017.59
13. Беляков Д.С., Калинин Е.О., Конев А.А., Шелупанов А.А., Мицель А.А. Модели жизненного цикла и угрозы безопасности микросхемы во время ее разработки и эксплуатации. *Доклады Томского государственного университета систем управления и радиоэлектроники* (*Доклады ТУСУР*). 2023;26(1):76–81. https://doi.org/10.21293/1818-0442-2023-26-1-76-81
14. Конев А.А. Модель угроз безопасности защищенного микроконтроллера и обрабатываемой им информации. *Доклады Томского государственного университета систем управления и радиоэлектроники* (*Доклады ТУСУР*). 2022;25(4):80–87. https://doi.org/10.21293/1818-0442-2022-25-4-80-87

15. Restuccia G., Tschofenig H., Baccelli E. Low-Power IoT Communication Security: On the Performance of DTLS and TLS 1.3. In: *Proceedings 2020 9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks* (*PEMWN*). https://doi.org/10.23919/PEMWN50727.2020.9293085
16. Lowther D., Jacob D., Trevor J., Singer J. Secure Scripting with CHERIoT MicroPython. In: *Proceedings of the 34th ACM SIGPLAN International Conference on Compiler Construction*. ACM; 2025. P. 180–191. https://doi.org/10.1145/3708493.3712694

REFERENCES

1. Dyakov O.N., Belyakov D.S., Kalinin E.O. Using ePKI technology to securely update of embedded software of trusted hardware and software system. *Bezopasnost' informatsionnykh tekhnologii = IT Security* (*Russia*). 2025;32(2):152–177 (in Russ.). https://doi.org/10.26583/bit.2025.2.12
2. Venugopal M., Nanda M., Anand G., Chandana Voora H. An integrated Hardware/Software Verification and Validation methodology for Signal Processing Systems. *ITM Web Conf.* 2022;50:02001. https://doi.org/10.1051/itmconf/20225002001
3. Bures M., Cerny T., Ahmed B.S. Internet of Things: Current Challenges in the Quality Assurance and Testing Methods. *arXiv*. arXiv:1805.01241. 2018. https://doi.org/10.48550/arXiv.1805.01241
4. Mazhar T., Talpur D.B., Shloul T.A., Ghadi Y.Y., Haq I., Ullah I. Analysis of IoT Security Challenges and its Solutions Using Artificial Intelligence. *Brain Sciences*. 2023;13(4):683. https://doi.org/10.3390/brainsci13040683
5. Minani J.B., Sabir F., Moha N., Guéhéneuc Y.G. A Multimethod Study of Internet of Things Systems Testing in Industry. *IEEE Internet Things J*. 2024;11(1):1662–1684. https://doi.org/10.1109/JIOT.2023.3291233
6. Papulovskaya N.V., Izotov I.N., Blinichkin D.Y., Kataev A.Y. Core Platform Development for IoT-devices Automated Testing. *Int. J. Open Inf. Technol*. 2021;9(6):38–45 (in Russ.). https://elibrary.ru/ybxvtg
7. Castelo Branco K.D.S., Dantas V.L.L., Carvalho L.M. Interoperability Testing Guide for the Internet of Things. In: *Proceedings of the 30th Brazilian Symposium on Multimedia and the Web* (*WebMedia 2024*). (*Sociedade Brasileira de Computação*). 2024. P. 188–196. https://doi.org/10.5753/webmedia.2024.242058
8. Weiss K., Rottleuthner M., Schmidt T.C., Wählisch M. PHiLIP on the HiL: Automated Multi-Platform OS Testing with External Reference Devices. *ACM Trans. Embed. Comput. Syst*. (*TECS*). 2021;20(5s):1–26. https://doi.org/10.1145/3477040
9. Behnke I., Thamsen L., Kao O. Héctor: A Framework for Testing IoT Applications Across Heterogeneous Edge and Cloud Testbeds. In: *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion.* ACM; 2019. P. 15–20. https://doi.org/10.1145/3368235.3368832
10. Ziegler S., Fdida S., Viho C., Watteyne T. F-Interop – Online Platform of Interoperability and Performance Tests for the Internet of Things. In: Mitton N., Chaouchi H., Noel T., Watteyne T., Gabillon A., Capolsini P. (Eds.). *Interoperability, Safety and Security in IoT. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer; 2017. V. 190. P. 49–55. Available from URL: https://link.springer.com/chapter/10.1007/978-3-319-52727-7_7. Accessed July 07, 2025.
11. Olianas D., Leotta M., Ricca F. MATTER: A tool for generating end-to-end IoT test scripts. *Software Qual. J*. 2021;30(2): 389–423. Available from URL: https://link.springer.com/article/10.1007/s11219-021-09565-y. Accessed July 07, 2025.
12. Schieferdecker I., Kretzschmann S., Rennoch A., Wagner M. IoT-Testware – An Eclipse Project. In: *2017 IEEE International Conference on Software Quality, Reliability and Security* (*QRS*). IEEE; 2017. https://doi.org/10.1109/QRS.2017.59
13. Belyakov D.S., Kalinin E.O., Konev A.A., Shelupanov A.A., Mitsel A.A. Life-cycle models and security threats to the microchip during its development and exploitation. *Doklady Tomskogo gosudarstvennogo universiteta sistem upravleniya i radioehlektroniki* (*Doklady TUSUR*) = *Proceedings of TUSUR University.* 2023;26(1):76–81 (in Russ.). https://doi.org/10.21293/1818-0442-2023-26-1-76-81
14. Konev A.A. Security threat model for protected microcontroller and the information it processes. *Doklady Tomskogo gosudarstvennogo universiteta sistem upravleniya i radioehlektroniki* (*Doklady TUSUR*) = *Proceedings of TUSUR University.* 2022;25(4):80–87 (in Russ.). https://doi.org/10.21293/1818-0442-2022-25-4-80-87
15. Restuccia G., Tschofenig H., Baccelli E. Low-Power IoT Communication Security: On the Performance of DTLS and TLS 1.3. In: *Proceedings 2020 9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks* (*PEMWN*). https://doi.org/10.23919/PEMWN50727.2020.9293085
16. Lowther D., Jacob D., Trevor J., Singer J. Secure Scripting with CHERIoT MicroPython. In: *Proceedings of the 34th ACM SIGPLAN International Conference on Compiler Construction*. ACM; 2025. P. 180–191. https://doi.org/10.1145/3708493.3712694

Об авторе

Беляков Данила Сергеевич, старший преподаватель, кафедра комплексной информационной безопасности электронно-вычислительных систем, факультет безопасности, ФГАОУ ВО «Томский государственный университет систем управления и радиоэлектроники» (634050, Россия, Томск, пр-т Ленина, д. 40). E-mail: cauze4concern@yandex.ru. Scopus Author ID 57359476400, ResearcherID AAQ-4613-2021, SPIN-код РИНЦ 3368-8751, https://orcid.org/0000-0002-6111-455X

About the Author

Danila S. Belyakov, Senior Lecturer, Department of Complex Information Security of Computer Systems, Faculty of Security, Tomsk State University of Control Systems and Radioelectronics (40, Lenina pr., Tomsk, 634050 Russia). E-mail: cauze4concern@yandex.ru. Scopus Author ID 57359476400, ResearcherID AAQ-4613-2021, RSCI SPIN-code 3368-8751, https://orcid.org/0000-0002-6111-455X