СИСТЕМНЫЙ АНАЛИЗ, УПРАВЛЕНИЕ И ОБРАБОТКА ИНФОРМАЦИИ, СТАТИСТИКА

УДК 004.45(1.6) Обзорная статья
https://doi.org/10.35330/1991-6639-2026-28-3-84-95
VIJDOE

Анализ используемых технологий межмодульного взаимодействия в одноплатных многопроцессорных системах

А. В. Смирнов

Калининградский государственный технический университет
236022, Россия, г. Калининград, Советский проспект, 1

Аннотация. Многопроцессорные системы используются для обеспечения вычислений во многих областях науки, однако их эффективность во многом зависит от того, как эти модули взаимодействуют друг с другом.

Цель исследования – провести обзор современного состояния и анализ используемых технологий межмодульного взаимодействия в одноплатных многопроцессорных системах, выявить тенденции их развития и использования, а также их ограничения.

Результаты. В данной работе был проведен комплексный анализ современного состояния и тенденций развития одноплатных вычислительных систем. Сделаны обзор и сравнение протоколов и моделей коммуникации, используемых для организации взаимодействия между одноплатными вычислительными системами. Также приведено описание дальнейших исследований по этой теме.

Заключение. Анализ показал, что существующее разнообразие архитектур процессоров и технологий организации межмодульного взаимодействия не позволяет выбрать некое одно универсальное решение, и выбор конкретного решения зависит от множества факторов, таких как необходимые задержка, пропускная способность, уровень безопасности сети, и всегда предполагает поиск компромисса для конкретной задачи.

Ключевые слова: межмодульное взаимодействие, многопроцессорные системы, одноплатные вычислительные системы

Поступила 29.12.2025, одобрена после рецензирования 21.04.2026, принята к публикации 11.06.2026

Для цитирования. Смирнов А. В. Анализ используемых технологий межмодульного взаимодействия в одноплатных многопроцессорных системах // Известия Кабардино-Балкарского научного центра РАН. 2026. Т. 28. № 3. С. 84–95. DOI: 10.35330/1991-6639-2026-28-3-84-95

MSC: 68M14 Review article

Analysis of inter-module communication technologies in single-board multiprocessor systems

A.V. Smirnov

Kaliningrad State Technical University
1, Sovetsky prospekt, Kaliningrad, 236022, Russia

Abstract. Multiprocessor systems are extensively used for scientific computing, yet their efficiency depends significantly on the mechanisms of inter-module interaction.

Aim. The study aims to review and analyze inter-module communication technologies in single-board multiprocessor systems, identifying their development trends, application scopes, and limitations.

Results. This paper provides a comprehensive analysis of the current state and development trends of single-board computing systems. It also includes a detailed benchmark of communication models and protocols designed for efficient interconnection between single-board computing systems. Further research on this topic is also described.

Conclusion. The analysis confirms that the extensive variety of processor architectures and intermodule communication technologies does not allow for a single universal solution. The selection of a specific solution relies on numerous factors, such as target latency, throughput, and network security, and invariably involves achieving a compromise for a given application.

Keywords: inter-module communication, multiprocessor systems, single-board computing systems

Submitted 29.12.2025, approved after reviewing 21.04.2026, accepted for publication 11.06.2026

For citation. Smirnov A.V. Analysis of inter-module communication technologies in single-board multiprocessor systems. *News of the Kabardino-Balkarian Scientific Center of RAS.* 2026. Vol. 28. No. 3. Pp. 84–95. DOI: 10.35330/1991-6639-2026-28-3-84-95

Введение

Одноплатные вычислительные системы (SBC, Single-Board Computer) находят широкое применение в различных областях, включая интернет вещей (IoT), системы SCADA, робототехнику, медицинское оборудование и системы искусственного интеллекта [1]. Их распространенность обусловлена масштабируемостью, портативностью, энергоэффективностью, а также поддержкой широкого спектра протоколов и интерфейсов для подключения периферийных устройств и датчиков, включая I2C, SPI, UART, PCI и USB, что обеспечивает возможность интеграции практически в любые аппаратно-программные комплексы.

Основными компонентами одноплатных вычислительных систем являются центральный процессор (CPU), который в зависимости от целевых задач характеризуется различным количеством ядер, тактовой частотой, наличием встроенного графического процессора (GPU) и в ряде случаев специализированного нейронного процессора (NPU) для ускорения алгоритмов машинного обучения, и оперативная память (RAM), выполненная по технологиям LPDDR4 или LPDDR5, для обеспечения более энергоэффективной работы. Для хранения данных используются накопители стандартов eMMC, UFS или SSD, а для размещения загрузчика и низкоуровневого программного обеспечения применяются распаянные на плате модули flash-памяти вида NAND/NOR. Практически все одноплатные

системы поддерживают стандартные интерфейсы, такие как GPIO, UART, I2C, SPI, PCIe, USB, а также более специализированные в зависимости от области применения.

Центральный процессор является основополагающим компонентом, определяющим вычислительную мощность SBC. На современном рынке доминируют три основные архитектуры CPU: ARM, AMD64 (x86-64) и RISC-V. Архитектура напрямую влияет на производительность, энергопотребление и совместимость с программным обеспечением.

Большинство современных SBC базируются на процессорах с архитектурой ARM, которая представляет собой усовершенствованную версию RISC-архитектуры и обеспечивает высокую энергоэффективность при достаточном уровне производительности. Основным ограничением ARM-архитектуры является ограниченная совместимость с программным обеспечением, разработанным для платформы x86-64, из-за различий инструкций данных архитектур. Существуют программные решения, частично снимающие это ограничение, однако их использование неминуемо ведет к снижению производительности [2]. Наиболее распространенными ARM-процессорами в этом сегменте являются решения компаний Broadcom, Rockchip, Amlogic и Allwinner.

Параллельно с ARM набирают популярность системы на основе открытой архитектуры RISC-V, которая не требует лицензионных отчислений. Отсутствие лицензионных отчислений является фактором, способным сделать RISC-V одной из наиболее экономически привлекательных архитектур в будущем. Однако на текущий момент поддержка данной архитектуры со стороны как аппаратного, так и программного обеспечения остается ограниченной, хотя и демонстрирует высокие темпы развития. Ключевыми участниками рынка, продвигающими RISC-V, являются компании SiFive и Espressif; решения последней, например, микроконтроллер ESP32-C6, уже активно применяются в IoT-устройствах [3, 4, 5].

В тех сегментах, где критически важна полная программная совместимость, продолжают применяться решения на основе архитектуры AMD64 (x86-64) от компаний Intel и AMD.

Сравнение описанных архитектур процессоров представлено в таблице 1.

Таблица 1. Сравнительная таблица архитектур одноплатных систем

Table 1. Comparative Analysis of Single-Board Computer Architectures

	ARM	**AMD64 (x86-64)**	**RISC-V**
Архитектура	Закрытая	Закрытая	Открытая
Энергопотребление	Низкое	Варьируется, но выше, чем ARM и RISC-V	Низкое
Производительность	Высокая	Очень высокая	Средняя, но активно развивается
Поддержка ПО	Высокая, однако требует использования эмуляторов	Очень высокая	Низкая, но активно развивается
Стоимость	Средняя	Высокая	Выше среднего, имеется перспектива снижения стоимости

Современные одноплатные вычислительные системы поддерживают широкий спектр операционных систем, включая RISC OS, RIOT, Contiki OS, а также Windows 10 IoT Core. Тем не менее доминирующее положение на рынке операционных систем для SBC занимает семейство ОС на ядре Linux. Данная тенденция обусловлена в первую очередь

открытостью исходного кода, что позволяет широкому кругу разработчиков участвовать в развитии системы и обеспечивает поддержку большого количества аппаратных конфигураций.

В отличие от проприетарных операционных систем ядро Linux может быть перекомпилировано с целью включения только необходимых драйверов и подсистем для конкретной SBC. Это позволяет минимизировать размер операционной системы, экономя дисковое пространство и оперативную память, что критически важно для встраиваемых систем с ограниченными ресурсами. Помимо этого, Linux предоставляет высокоуровневую абстракцию для аппаратных интерфейсов, таких как I2C, SMBus, SPI и PCI, представляя их в виде стандартизированных и доступных для разработчика API. Это позволяет взаимодействовать с датчиками, исполнительными механизмами и другими электронными компонентами без необходимости написания низкоуровневого кода, специфичного для конкретной аппаратной платформы, что значительно ускоряет процесс разработки и повышает переносимость программного обеспечения.

Обеспечение коммуникации между одноплатными вычислительными системами

Для обеспечения коммуникации между различными одноплатными вычислительными системами используются различные протоколы [6].

В настоящее время одним из наиболее распространенных стандартов в области организации взаимодействия между распределенными вычислительными системами является MPI (Message Passing Interface) [7]. В основе MPI лежит модель взаимодействия процессов посредством отправки и получения сообщений. С практической точки зрения MPI предоставляет примитивы для коммуникации между модулями поверх TCP/IP. Это предоставляет разработчикам стандартизированный программный интерфейс (API) для организации коммуникаций, что существенно упрощает разработку приложений. К ключевым преимуществам MPI можно отнести высокую масштабируемость, позволяющую эффективно использовать вычислительные ресурсы при увеличении числа узлов, богатую функциональность, а также наличие зрелой экосистемы. Однако функциональная полнота MPI обусловливает и ряд недостатков. К ним можно отнести повышение требования к памяти (объем пакета OpenMPI превышает 50 мегабайт, что может являться критическим ограничением для некоторых одноплатных вычислительных систем), а также повышение задержек при большом количестве процессов, что также может быть критично для низкопроизводительных одноплатных вычислительных систем [8]. Основным направлением использования MPI является распараллеливание вычислений на СуперЭВМ, однако он также используется для создания кластерных систем на основе SBC [9]. Но так как большинство SBC-систем используются для IoT, где необходимо отправлять файлы между сервером и клиентами, MPI не так сильно распространен, но все еще является хорошим выбором для создания кластерных систем вычислений.

Другая альтернатива MPI – это удаленный вызов процедур (Remote Procedure Calls, RPC), который позволяет программе выполнить код на удаленном узле так, как если бы это был локальный вызов функции [10]. Современные имплементации RPC, такие как gRPC или Apache Thrift, построены поверх транспортных протоколов. Чаще всего используется HTTP/2 с дополнительными протоколами сериализации данных и механизмами балансировки нагрузки между модулями. Тем не менее, несмотря на удобство разработки, модель RPC обладает и недостатками, связанными с производительностью [11].

Операции сериализации и десериализации данных являются вычислительно затратными. Кроме того, использование протоколов прикладного уровня (таких как HTTP) для инкапсуляции RPC-вызовов добавляет дополнительные накладные расходы на обработку заголовков и установление соединения.

Также существуют системы очередей сообщений (Message Queue, MQ), функционирование которых базируется на парадигме асинхронного обмена сообщениями. В зависимости от архитектуры и используемого протокола взаимодействие между компонентами распределенной системы может осуществляться как через централизованный промежуточный узел – брокер сообщений (например, RabbitMQ, Apache Kafka), так и в децентрализованной манере (например, ZeroMQ), при которой брокер отсутствует, а обмен происходит напрямую между узлами. Брокер-ориентированные архитектуры предоставляют развитую маршрутизацию и гарантии доставки, однако введение брокера создает потенциальные риски: он может стать «узким местом» (bottleneck) производительности, а также точкой отказа всей системы [12].

Также существует отдельная имплементация MQ под названием MQTT [13], который был спроектирован как чрезвычайно легковесный транспортный протокол, функционирующий поверх TCP/IP. Благодаря минимальному размеру служебных заголовков и низким требованиям к ресурсам MQTT является наиболее используемым решением для устройств с ограниченной пропускной способностью каналов связи или с ограниченными вычислительными ресурсами, таких как микроконтроллеры и одноплатные вычислительные системы. Для функционирования MQTT требуется от 8.92KB до 17KB RAM, что является чрезвычайно низким значением, это позволяет MQTT работать на абсолютном большинстве SBC и микроконтроллеров [14]. Но в то же время MQTT не поддерживает выставление приоритетов для сообщений и контроля очередности сообщений, и функционирование поверх TCP/IP добавляет дополнительную задержку [13, 14]. Также основной проблемой, по мнению исследователей, является безопасность протокола, несмотря на то, что протокол имеет встроенные механизмы шифрования, они не охватывают все части протокола, например, схема обмена сообщениями и информация для аутентификации пользователя передаются в открытом виде [15, 16].

Кроме того, существует проприетарное решение ESP-NOW, разработанное компанией Espressif Systems. Отличительными особенностями данного протокола являются низкая задержка передачи данных, обусловленная отказом от использования стека TCP/IP в пользу прямого соединения, простота API, высокая энергоэффективность и дальность передачи данных, а также наличие встроенных механизмов шифрования для обеспечения безопасности передачи. Однако к ограничениям ESP-NOW следует отнести его аппаратную зависимость: поддержка протокола реализована исключительно на устройствах производства Espressif. Кроме того, протокол характеризуется ограничением на максимальное количество одновременно подключаемых устройств в сети [17–19].

Почти все указанные выше протоколы коммуникации основаны на сокетах (sockets), следовательно, для одноплатных вычислительных систем с очень ограниченной вычислительной мощностью и небольшим объемом оперативной памяти можно использовать сокеты (sockets) как низкоуровневый протокол для обеспечения коммуникации между SBC. Сокеты предоставляют программный интерфейс для прямой и низкоуровневой передачи данных поверх TCP и UDP. Основным преимуществом использования сокетов является их легковесность по причине отсутствия высокоуровневых абстракций, однако по этой же причине использование сокетов требует от разработчика более высокого уровня понимания работы системы и ручной реализации дополнительных механизмов передачи данных [6].

Обобщение анализа представлено в таблице 2.

Таблица 2. Сравнительная таблица протоколов межмодульного взаимодействия

Table 2. Comparative Analysis of Inter-Module Interaction Protocols.

Протокол	**Преимущества**	**Недостатки**
MPI	Масштабируемость, стандартизированный API, богатая функциональность	Высокие требования к памяти (>50 МБ), рост задержек с увеличением числа узлов
RPC	Простота разработки, балансировка нагрузки	Высокие накладные расходы на сериализацию /десериализацию, задержки от HTTP/2
MQ	Гарантии доставки, маршрутизация, отказоустойчивость (при использовании кластеризации)	Брокер как «узкое место» (bottleneck) и единая точка отказа, сложность конфигурации
ESP-NOW	Сверхнизкая задержка, простота, энергоэффективность, встроенное шифрование	Проприетарный (только Espressif), ограниченное число устройств в сети, подверженность помехам
Сокеты	Легковесность, полный контроль над передачей данных	Низкоуровневый API, требует ручной реализации всех механизмов
MQTT	Чрезвычайно легковесный, оптимален для ограниченных устройств	Нет встроенных приоритетов и контроля очередности, существуют проблемы безопасности

Анализ используемых систем коммуникации в различных SBC-системах

В работе [20], посвященной созданию системы мониторинга агропромышленных параметров на базе одноплатного микроконтроллера STM32L051K8, реализована схема сбора и удаленной передачи данных. Информация с датчиков передавалась на сервер посредством протокола MQTT поверх соединения LTE. На серверном уровне поступившие данные агрегировались и записывались в базу данных InfluxDB для последующей визуализации и анализа оператором.

В исследовании [21], направленном на анализ концентрации CO_2 с использованием IoT-платформы на базе микроконтроллера Atmega2560, применялся иной подход к организации канала связи. Передача данных о газовом составе осуществлялась по каналу Wi-Fi с использованием проприетарного протокола, обозначенного авторами как IoT Json, функционирующего поверх стека HTTP over TCP. Отправка сообщений производилась строго через заданные временные интервалы.

В исследовании [22], посвященном изучению возможности применения большого количества микроконтроллеров для задач машинного обучения, были протестированы различные архитектуры для передачи данных между несколькими узлами с использованием стандарта IEEE 802. В рамках эксперимента были рассмотрены протоколы ESP-NOW, UDP, TCP и BLE. Каждая из архитектур обладает своими преимуществами и недостатками, что делает их применение зависимым от конкретных задач и условий эксплуатации. В результате исследования были сделаны следующие выводы: протокол BLE продемонстрировал наивысшую энергоэффективность, однако для передачи данных по этому протоколу требуется больше времени, чем при использовании других протоколов. Кроме того, BLE имеет ограничение на размер передаваемого пакета данных, который не может превышать 512 байт.

Протокол TCP позволяет передавать более крупные пакеты данных до 1460 байт, но имеет ограничение на количество одновременно активных клиентов (зависит от характеристик микроконтроллера). Также TCP требует использования трехстороннего рукопожатия для установления соединения между сервером и клиентом, что повышает надежность доставки сообщений, но увеличивает overhead. Протокол ESP-NOW, несмотря на отсутствие накладных расходов, связанных с использованием IP-стека, также имеет ограничения на количество одновременно активных клиентов и размер передаваемых пакетов, которые не могут превышать 250 байт. Протокол UDP теоретически позволяет поддерживать неограниченное количество клиентов, так как не требует предварительного установления соединения между отправителем и получателем. Это снижает задержки, но одновременно уменьшает надежность передачи данных, так как в UDP отсутствует механизм, обеспечивающий подтверждение доставки данных.

В процессе разработки системы мониторинга энергопотребления в режиме реального времени для устройств на базе микроконтроллера SiFive HiFive Premier P550 [23], функционирующего на архитектуре RISC-V, в качестве протокола передачи данных был выбран MQTT. Данный протокол, функционирующий поверх стека TCP/IP, по мнению авторов, является оптимальным для применения в системах с ограниченными вычислительными ресурсами и в условиях значительных сетевых задержек.

В рамках исследования, направленного на создание управляющего контроллера для лабораторного макета цифрового приемника PRATUSH [24], передача данных осуществлялась по каналу Ethernet. Для физического подключения использовался кабель категории 6 (RJ45 CAT6). Обмен информацией был реализован путем отправки UDP-дейтаграмм с пропускной способностью канала 8 Мбит/с. Ввиду ограниченной производительности одноплатного компьютера начального уровня Raspberry Pi 4B, используемого в качестве вычислительного ядра контроллера, программно была введена задержка длительностью в одну секунду. Данное решение позволило обеспечить необходимую синхронизацию процессов ввода данных и их последующей обработки, нивелируя асинхронность работы подсистем ввода-вывода и вычислений.

В работе [25], посвященной разработке активной системы видеонаблюдения, все модули были объединены в единую беспроводную сеть на базе Wi-Fi. Взаимодействие между компонентами системы реализовано по принципу клиент-серверной архитектуры. Отличительной особенностью предложенного решения является то, что каждый периферийный модуль функционирует как самостоятельный веб-сервер. Центральный (головной) управляющий модуль осуществляет подключение к ним по известным IP-адресам для сбора данных. Таким образом, несмотря на общую клиент-серверную модель взаимодействия, каждый узел сохраняет автономность и способность к независимой обработке запросов.

В исследовании, посвященном разработке открытой SCADA-системы для мониторинга солнечной панели [26], была реализована система передачи данных следующей архитектуры. Сбор и первичная передача данных осуществлялись посредством микроконтроллера ESP32. В качестве серверной платформы выступал одноплатный компьютер (SBC) Raspberry Pi 2. Обмен данными между оконечным устройством (ESP32) и сервером производился с использованием протокола MQTT, функционирующего поверх стека TCP/IP. Для обеспечения безопасности и идентификации клиента при подключении к брокеру применялся маркер доступа (Access Token), что является стандартным методом аутентификации в MQTT. Таким образом, в основе системы лежит брокерная архитектура, где Raspberry Pi 2 выполняет роль MQTT-брокера, а ESP32 – роль издателя данных.

Проведенный анализ позволяет сделать ряд обобщающих выводов. Прежде всего, очевидно отсутствие универсального подхода к организации каналов передачи данных: выбор конкретного стека протоколов (MQTT, HTTP, TCP, UDP, ESP-NOW, BLE) и физической среды (LTE, Wi-Fi, Ethernet) детерминируется исключительно спецификой решаемой задачи, требованиями к энергоэффективности и условиями эксплуатации. Ключевым выводом является необходимость поиска компромисса между надежностью доставки, скоростью передачи и энергопотреблением. Так, протокол BLE обеспечивает высокую автономность устройств ценой ограничений на размер пакета и скорость; TCP гарантирует целостность данных, но создает значительные накладные расходы, в то время как UDP минимизирует задержки при отсутствии механизмов подтверждения доставки. В рассмотренных работах прослеживается устойчивая тенденция к использованию брокерных архитектур на базе протокола MQTT, что обусловлено его эффективностью в условиях ограниченных вычислительных ресурсов и нестабильности сетевого соединения. Одновременно с этим наблюдается функциональное разделение оборудования: микроконтроллеры выполняют роль оконечных устройств сбора данных, а более производительные одноплатные компьютеры (Raspberry Pi) берут на себя функции шлюзов, брокеров и серверов баз данных. Ключевым трендом также является переход к автономным периферийным узлам для обеспечения большей отказоустойчивости.

НАПРАВЛЕНИЯ ПЕРСПЕКТИВНЫХ ИССЛЕДОВАНИЙ

Следующим шагом развития исследования является разработка методики адаптивного выбора протокола межмодульного взаимодействия для одноплатных компьютеров (SBC) и микроконтроллеров. Необходимо создание системы поддержки принятия решений, которая на основе формализованных требований к производительности, задержке, надежности доставки, энергопотреблению, дальности связи и доступным вычислительным ресурсам узлов будет автоматически рекомендовать оптимальный протокол или гибридную схему коммуникации. Ключевой особенностью такой методики должна стать способность учитывать динамически изменяющиеся условия эксплуатации, такие как производительность, качество канала связи и другие.

Базисом для построения подобных адаптивных механизмов должно стать количественное исследование производительности протоколов на гетерогенных аппаратных платформах. Планируется проведение серии экспериментов на стенде, включающем ARM-платы, RISC-V системы и популярные микроконтроллеры. В ходе экспериментальных исследований предполагается измерение ключевых метрик: производительности, задержки сети, джиттера, пропускной способности, энергопотребления для широкого спектра протоколов – MQTT, CoAP, gRPC, ESP-NOW, а также «сырых» UDP/TCP соединений в различных режимах работы. Результатом данной работы станет создание бенчмарка, который будет использоваться в дальнейших исследованиях.

В свете перехода к автономным распределенным системам необходимо исследовать методы обеспечения безопасности в гетерогенных SBC-системах. Критически важен анализ уязвимостей всего тракта передачи данных. В рамках этого направления предполагается разработка легковесных методов шифрования и аутентификации, адаптированных для маломощных узлов с жесткими ограничениями по памяти и вычислительной производительности.

Также необходимостью является разработка методов обеспечения отказоустойчивости и синхронизации. Перспективным видится направление по адаптации классических алгоритмов консенсуса (Raft, Paxos) для систем с узлами, имеющими кратно различающиеся

вычислительные ресурсы. Также требует изучения возможность построения децентрализованных систем управления без выделенного брокера, что позволит исключить единую точку отказа и повысить живучесть системы в целом.

После чего планируется разработка унифицированного фреймворка для межмодульного взаимодействия. Концептуально это программная прослойка, предоставляющая приложениям единый API и скрывающая сложность выбора транспортного протокола. Фреймворк должен автоматически выбирать или даже динамически переключать протокол «на лету» (например, переходить с MQTT на ESP-NOW при потере Wi-Fi-соединения) в зависимости от текущих сетевых условий и доступного энергетического бюджета, обеспечивая тем самым адаптивность и отказоустойчивость.

Опишем формальную постановку задачи исследования. Пусть имеется гетерогенная распределенная система, состоящая из множества вычислительных модулей (узлов) $\mathrm{U} = \{u_1, u_2, \dots, u_i, \dots, u_n\}$, где каждый узел u_i характеризуется:

- архитектурой процессора $A_i \in \{\text{ARM, x86-64, RISC-V}\}$;
- производительностью процессора P_i;
- объемом оперативной памяти R_i;
- типом энергопитания $P_i \in \{\text{battery, DC}\}$;
- поддерживаемыми интерфейсами связи $I_i \in \{\text{Wi-Fi, Ethernet, BLE, ESP-NOW}\}$.

Имеется множество протоколов $\mathrm{Pr} = \{pr_1, pr_2, \dots, pr_m\} \in \{\text{MQTT, CoAP, gRPC, ESP-NOW, RAW UDP, RAW TCP}\}$, где каждый протокол pr обладает следующим вектором характеристик: $ch(p,u) = \langle Latency, Performance, Loss, Security\rangle$.

Задача T, цель которой заключается в организации информационного обмена между модулями, задается вектором требований к качеству передачи информации

$$\mathrm{QoS}_\text{req} = \langle P_{min}, D_{max}, L_{max}, S_{min}\rangle,$$

где:

- P_{min} – минимально необходимая производительность;
- D_{max} – максимально допустимая задержка доставки сообщений (мс);
- L_{max} – максимально допустимые потери пакетов (%);
- S_{min} – минимальный уровень безопасности.

Требуется разработать метод, которой в режиме реального времени для каждого модуля u определяет оптимальный протокол межмодульного взаимодействия pr и минимизирует задержки D_{total}:

$$pr^* = \min_{pr\in Pr} \sum_{i=1}^{n} D_i(p,u_i) \text{ при } \begin{cases} Latency(p,\, u_i)) \le D_{max} \\ Performance(p,\, u_i)) \le P_{min} \\ Loss(p,\, u_i)) \le L_{max} \\ Security(p,\, u_i)) \le S_{min}. \end{cases}$$

Комплексная реализация перечисленных исследований позволит перейти от эмпирического подхода к выбору протоколов к созданию научно обоснованных, адаптивных и безопасных систем коммуникации для следующего поколения гетерогенных распределенных вычислительных систем.

Таким образом, следующий задачей является разработка и экспериментальная валидация метода адаптивного выбора протокола межмодульного взаимодействия в гетерогенных распределенных системах на базе одноплатных компьютеров и микроконтроллеров, обеспечивающего выполнение заданных требований к качеству обслуживания при минимизации энергопотребления, а также создание программного фреймворка, реализующего предложенный метод.

Заключение

В ходе выполнения работы был проведен анализ современного состояния одноплатных вычислительных систем. Выполненный анализ демонстрирует, что одноплатные вычислительные системы стали неотъемлемым компонентом современной электроники, находя применение в самых разных областях – от бытовой автоматизации до промышленной робототехники. Существующее разнообразие архитектур (ARM, AMD64 и RISC-V) предоставляет разработчикам возможность выбора оптимального баланса между вычислительной мощностью, энергопотреблением и стоимостью в соответствии с требованиями конкретного проекта. Анализ протоколов организации межмодульного взаимодействия показывает, что выбор конкретного решения всегда предполагает поиск компромисса. Если для вычислительных кластеров обосновано применение тяжеловесных решений вроде MPI, то в сфере интернета вещей доминируют легковесные протоколы. Среди них особое место занимает MQTT, который благодаря минимальным требованиям к памяти и пропускной способности каналов связи стал фактическим стандартом для SBC и микроконтроллеров. Анализ практических разработок выявляет тенденцию к формированию гибридных архитектур, в которых функции между узлами распределяются в зависимости от их возможностей. Простые микроконтроллеры используются как оконечные устройства сбора данных, в то время как более производительные одноплатные компьютеры выполняют функции шлюзов, брокеров сообщений или серверов. Такой подход позволяет оптимально сочетать экономичность периферийных устройств с вычислительной мощностью центральных узлов. Кроме того, прослеживается стремление к повышению автономности каждого элемента системы, что способствует росту отказоустойчивости и упрощению масштабирования. Также приведено описание дальнейших исследований по этой теме.

СПИСОК ЛИТЕРАТУРЫ / REFERENCES

1. Ossont S.J., Basford P.J., Perkins C.S. et al. Commodity single board computer clusters and their applications. *Future Generation Computer Systems*. 2018. Vol. 89. No. 2. Pp. 201–212. DOI: 10.1016/j.future.2018.06.048

2. Lombardi F., Recchia A. On abstract machines security and performance. *Procedia Computer Science*. 2024. Vol. 231. Pp. 111–118. DOI: 10.1016/j.procs.2023.12.182

3. Gul F.A.I., Tudose D., Ţurcanu T. A versatile IoT development board for environmental sensing and biometric applications. *23rd RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2024. Pp. 1–6. DOI: 10.1109/RoEduNet64292.2024.10722601

4. Pajkos J. et al. Esp32 microcontroller based lightweight tls 1.3 client for iot applications. *35th International Conference Radioelektronika (RADIOELEKTRONIKA).* IEEE, 2025. Pp. 1–6. DOI: 10.1109/RADIOEEKTRONIKA65656.2025.11008381

5. Mota A., Serodio C., Sá A.B., Valente A. Implementation of an Internet of Things architecture to monitor indoor air quality: a case study during sleep periods. *Sensors*. 2025. Vol. 25. No. 6. Art. 1683. DOI: 10.3390/s25061683

6. Dinari H. Inter-process communication (IPC) in distributed environments: An investigation and performance analysis of some middleware technologies. *International Journal of Modern Education and Computer Science*. 2020. Vol. 11. No. 2. P. 36. DOI: 10.5815/ijmecs.2020.02.05

7. Laguna I., Marshall R., Mohror K. et al. A large-scale study of MPI usage in open-source HPC applications. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2019. Pp. 1–14. DOI: 10.1145/3295500.3356176

8. Zhou H. et al. MPI progress for all. *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2024. Pp. 425–435. DOI: 10.1109/SCW63240.2024.00063

9. Misbahuddin S., Ibrahim M.M., Mahmoud A. et al. Automatic patients' vital sign monitoring by Single Board Computer (SBC) Based MPI Cluster. *2nd International Conference on Computer Applications & Information Security (ICCAIS). IEEE*, 2019. Pp. 1–5. DOI: 10.1109/CAIS.2019.8769551

10. Birrell A.D., Nelson B.J. Implementing remote procedure calls. *ACM Transactions on Computer Systems (TOCS).* 1984. Vol. 2. No. 1. Pp. 39–59. DOI: 10.1145/2080.357392

11. Seemakhupt K., Stephens B.E., Khan S. et al. A cloud-scale characterization of remote procedure calls. *Proceedings of the 29th Symposium on Operating Systems Principles*. 2023. Pp. 498–514. DOI: 10.1145/3600006.3613156

12. Fu G., Zhang Y., Yu G. A fair comparison of message queuing systems. *IEEE Access.* 2020. Vol. 9. Pp. 421–432. DOI: 10.1109/ACCESS.2020.3046503

13. Soni D., Makwana A. A survey on mqtt: a protocol of internet of things (iot). *International Conference on Telecommunication, Power Analysis and Computing Techniques (ICTPACT-2017)*. 2017. Vol. 20. No. April, 2017.

14. Serepas F., Papias I., Christakis K. et al. Lightweight embedded IoT gateway for smart homes based on an ESP32 microcontroller. *Computers*. 2025. Vol. 14. No. 9. P. 391. DOI: 10.3390/computers14090391

15. Dinculeană D., Cheng X. Vulnerabilities and limitations of MQTT protocol used between IoT devices. *Applied Sciences*. 2019. Vol. 9. No. 5. P. 848. DOI: 10.3390/app9050848

16. Yassein M.B., Shatnawi M.Q., Aljwarneh S. et al. Internet of Things: survey and open issues of MQTT protocol. *International Conference on Engineering & MIS (ICEMIS). IEEE*, 2017. Pp. 1–6. DOI: 10.1109/ICEMIS.2017.8273112

17. Urazayev D., Eduard A., Ahsan M. et al. Indoor performance evaluation of esp-now. *IEEE International Conference on Smart Information Systems and Technologies (SIST)*. *IEEE*, 2023. Pp. 1–6. DOI: 10.1109/SIST58284.2023.10223585

18. Eridani D., Rochim A.F., Cesara F.N. Comparative performance study of ESP-NOW, Wi-Fi, bluetooth protocols based on range, transmission speed, latency, energy usage and barrier resistance. *International Seminar on Application for Technology of Information and Communication (iSemantic). IEEE,* 2021. Pp. 322–328. DOI: 10.1109/iSemantic52711.2021.9573246

19. Becker B., Oberli Ch., Zobel J. et al. ESP-NOW performance in outdoor environments: Field experiments and analysis. *20th Wireless On-Demand Network Systems and Services Conference (WONS). IEEE*, 2025. Pp. 1–8.

20. Wang X. et al. SPARC-LoRa: A scalable, power-efficient, affordable, reliable, and cloud service-enabled LoRa networking system for agriculture applications. *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN). IEEE*, 2024. Pp. 151–156. DOI: 10.1109/DySPAN60163.2024.10632833

21. Flores-Cortez O., Cortez R., González B. Design and implementation of an IoT based LPG and CO gases monitoring system. *Computer Science; Information Technology – AIRCC Publishing Corporation*, 2021. Pp. 31–39. DOI: 10.5121/csit.2021.110803

22. Jenhani Z., Bensalem M., Dizdarevic J. et al. An experimental study of split-learning TinyML on Ultra-Low-Power Edge/IoT Nodes. *arXiv e-prints*. 2025. DOI: 10.48550/arXiv.2507.16594

23. Scionti A., Savio P., Lubrano F. et al. Runtime energy monitoring for RISC-V Soft-Cores. *International Conference on Complex, Intelligent, and Software Intensive Systems*. Cham: Springer Nature Switzerland, 2025. Pp. 283–292. DOI: 10.48550/arXiv.2509.26065

24. Srivani K.S., Girish B.S. et al. An SBC-based controller and processor for the laboratory model of PRATUSH digital receiver. *Experimental Astronomy*. 2025. Vol. 60. No. 1. P. 1. DOI: 10.1007/s10686-025-10013-z

25. Yamuna S., George S.N. An IoT based active building surveillance system using Raspberry Pi and NodeMCU. *arXiv e-prints*. 2020. DOI: 10.48550/arXiv.2001.11340

26. Aghenta L.O., Iqbal T. Design and implementation of a low-cost, open source IoT-based SCADA system using ESP32 with OLED, ThingsBoard and MQTT protocol. 2019. DOI: 10.3934/ElectrEng.2020.1.57

Финансирование. Исследование проведено без спонсорской поддержки.

Funding. The study was performed without external funding.

Информация об авторе

Смирнов Александр Владиславович, аспирант, Калининградский государственный технический университет;
236022, Россия, г. Калининград, Советский проспект, 1;
aleksandr.smirnov@klgtu.ru, ORCID: https://orcid.org/0009-0007-3580-3814, SPIN-код: 7142-4899

Information about the author

Alexander V. Smirnov, Postgraduate Student, Kaliningrad State Technical University;
1, Sovetsky prospekt, Kaliningrad, 236022, Russia;
aleksandr.smirnov@klgtu.ru, ORCID: https://orcid.org/0009-0007-3580-3814, SPIN-code: 7142-4899