Modern education and development
ISSN: 3060-4567

ИНФОРМАЦИОННАЯ СИСТЕМА СБОРА И ВИЗУАЛИЗАЦИИ ДАННЫХ С ГЕОРАСПРЕДЕЛЁННЫХ ДАТЧИКОВ
INFORMATION SYSTEM FOR COLLECTING AND VISUALIZING DATA FROM GEO-DISTRIBUTED SENSORS

Бобомуродов Б.С., *Ташкентский университет информационных технологий имени Мухаммада ал-Хоразмий*
Исломов Д.З*., Университет экономики и педагогики*

АННОТАЦИЯ: *В данной статье рассматривается проектирование и реализация информационной системы для сбора, обработки и визуализации данных с георасределённых датчиков. Современные промышленные и экологические задачи требуют непрерывного мониторинга физических параметров на больших территориях с использованием разнородных сенсорных сетей. Предложенная система обеспечивает централизованный сбор данных от датчиков, расположенных в различных географических точках, с последующей их обработкой и интерактивной визуализацией на картографической платформе. Разработана многоуровневая архитектура системы, включающая уровень полевых устройств, коммуникационный уровень, уровень обработки данных и уровень представления. Результаты эксперимента показали, что система успешно обеспечивает сбор данных в реальном времени с задержкой не более 2 секунд, высокую отказоустойчивость и масштабируемость до 10 000 узлов.*

Ключевые слова: *георасределённые датчики, IoT, сбор данных, визуализация, картографические системы, MQTT, веб-приложение, мониторинг в реальном времени.*

Keywords: *geo-distributed sensors, IoT, data collection, visualization, GIS, MQTT, web application, real-time monitoring.*

1. ВВЕДЕНИЕ

В последние годы стремительное развитие технологий Интернета вещей (IoT) открыло широкие возможности для построения распределённых сенсорных сетей, охватывающих обширные географические территории. Георазпределённые датчики находят применение в таких областях, как экологический мониторинг, умные города, сельское хозяйство, промышленная автоматизация, контроль инфраструктуры и системы раннего предупреждения о стихийных бедствиях [1]. Однако управление такими системами сопряжено с рядом технических трудностей: гетерогенность используемого оборудования, нестабильность каналов связи, большие объёмы потоковых данных и необходимость их интерпретации в режиме реального времени.

Актуальность настоящего исследования определяется растущей потребностью в унифицированных платформах, способных интегрировать разнородные источники данных и предоставлять операторам наглядный инструмент для анализа пространственно распределённой информации. Несмотря на наличие коммерческих решений в данной области, они зачастую характеризуются высокой стоимостью, закрытостью исходного кода и ограниченными возможностями кастомизации под специфические требования конкретного применения.

Целью данной работы является разработка масштабируемой информационной системы с открытой архитектурой, обеспечивающей надёжный сбор данных с георазпределённых датчиков, их хранение, первичную обработку и интерактивную визуализацию на географической карте.

Задачи исследования включают: анализ существующих решений для мониторинга георазпределённых датчиков; проектирование многоуровневой архитектуры системы; выбор и обоснование применяемых протоколов и технологий; реализацию прототипа системы; экспериментальную проверку производительности и масштабируемости разработанного решения.

1.1. Обзор существующих решений

Анализ литературы показывает, что существующие решения для мониторинга IoT-устройств можно разделить на три основные группы. К первой группе относятся коммерческие облачные платформы, такие как AWS IoT Core, Microsoft Azure IoT Hub и Google Cloud IoT. Эти платформы предоставляют широкий спектр сервисов, однако их использование связано со значительными операционными расходами и зависимостью от поставщика облачных услуг [2, 3].

Вторую группу составляют открытые платформы, такие как ThingsBoard, Grafana с InfluxDB и OpenHAB. Данные решения обеспечивают гибкость настройки, однако требуют значительных усилий по развёртыванию и администрированию [4]. Третья группа — специализированные геоинформационные системы (ГИС), например QGIS Server и GeoServer, которые ориентированы преимущественно на статические пространственные данные и не оптимизированы для работы с потоковыми телеметрическими данными [5].

Таким образом, в литературе отсутствует описание системы, которая органично сочетала бы возможности полноценной IoT-платформы с функциональностью современных ГИС при сохранении открытой архитектуры и низкого порога развёртывания. Настоящее исследование призвано восполнить данный пробел.

2. МАТЕРИАЛЫ И МЕТОДЫ

2.1. Архитектура системы

Разработанная система построена по принципу многоуровневой архитектуры (рис. 1), включающей четыре функциональных уровня: уровень полевых устройств (Field Layer), коммуникационный уровень (Communication Layer), уровень обработки и хранения данных (Data Processing Layer) и уровень представления (Presentation Layer).

На уровне полевых устройств располагаются датчики различных типов: датчики температуры и влажности (DHT22), датчики атмосферного давления (BMP280), датчики качества воздуха (MQ-135), а также модули GPS для определения координат. Каждый узел сети оснащён микроконтроллером ESP32, обеспечивающим беспроводную связь по протоколам Wi-Fi и LoRa.

Коммуникационный уровень реализован с использованием протокола MQTT (Message Queuing Telemetry Transport), который благодаря своей лёгковесности и асинхронной модели публикации/подписки (publish/subscribe) является де-факто стандартом для IoT-коммуникаций. В качестве брокера сообщений применяется Eclipse Mosquitto, развёрнутый на выделенном сервере. Для устройств с ограниченными возможностями беспроводной связи используются шлюзы LoRa-MQTT.

2.2. Технологический стек

Для реализации уровня обработки данных выбран язык программирования Python 3.11 в сочетании с асинхронным фреймворком FastAPI. Поступающие от брокера сообщения обрабатываются потоковым процессором Apache Kafka, обеспечивающим буферизацию и распределённую обработку высокочастотных потоков данных. Временны́е ряды хранятся в базе данных TimescaleDB, являющейся расширением PostgreSQL и оптимизированной для хранения и запроса временны́х данных.

Уровень представления реализован в виде одностраничного веб-приложения (SPA) на основе фреймворка React.js. Картографическая составляющая разработана с использованием библиотеки Leaflet.js с подключением тайловых карт OpenStreetMap. Для построения динамических графиков применяется библиотека Recharts. Взаимодействие клиента с сервером осуществляется через REST API и WebSocket-соединения, обеспечивающие обновление данных в режиме реального времени.

Таблица 1. Технологический стек разработанной системы

Уровень	Технология / Компонент	Назначение
Полевые устройства	ESP32 + DHT22, BMP280, MQ-135, GPS	Измерение и первичный сбор данных
Коммуникационный	MQTT (Eclipse Mosquitto), LoRa	Передача данных от узлов к серверу
Обработка данных	Apache Kafka, Python 3.11, FastAPI	Потоковая обработка и REST API
Хранение данных	TimescaleDB (PostgreSQL)	Хранение временных рядов
Представление	React.js, Leaflet.js, Recharts	Визуализация и картография
Безопасность	JWT, TLS/SSL, OAuth 2.0	Аутентификация и шифрование

2.3. Протокол сбора данных и обработки

Каждый датчик конфигурируется с уникальным идентификатором (Device ID) и публикует данные в MQTT-топик по шаблону: sensors/{device_id}/{sensor_type}/data. Формат сообщений — JSON, содержащий следующие поля: идентификатор устройства, тип измерения, значение, единица измерения, координаты (широта, долгота, высота) и метка времени в формате ISO 8601 (UTC). Период публикации данных конфигурируется индивидуально для каждого устройства и варьируется от 1 секунды до 60 минут в зависимости от типа применения.

На уровне обработки данных реализованы следующие функции: валидация входящих сообщений по JSON-схеме; обнаружение и обработка выбросов методом Z-оценки (порог $\sigma = 3$); агрегация данных по временным

окнам (1 мин, 10 мин, 1 ч, 24 ч); вычисление скользящего среднего для сглаживания шумовых компонент; генерация предупреждений при выходе параметров за установленные пороговые значения.

Для обеспечения надёжности передачи данных в условиях нестабильной связи в прошивку устройств интегрирован механизм буферизации с повторной отправкой (QoS Level 1), а также локальная запись на microSD-карту при отсутствии сетевого соединения. Синхронизация накопленных данных выполняется автоматически при восстановлении связи.

3. РЕЗУЛЬТАТЫ

3.1. Описание экспериментального стенда

Для оценки характеристик разработанной системы был развёрнут экспериментальный стенд, включающий 50 физических узлов, расположенных на территории трёх районов города Ташкента. Каждый узел оснащён датчиками температуры, влажности и качества воздуха. Серверная инфраструктура развёрнута на виртуальной машине с параметрами: 8 vCPU Intel Xeon Gold 6248, 32 ГБ ОЗУ, SSD-накопитель 500 ГБ. Операционная система — Ubuntu Server 22.04 LTS. Дополнительно было эмулировано от 100 до 10 000 виртуальных устройств с использованием инструмента MQTT-bench для нагрузочного тестирования.

3.2. Производительность системы

Проведённые испытания продемонстрировали высокую производительность разработанной системы. При нагрузке 1 000 одновременно подключённых устройств с частотой публикации данных 1 сообщение в секунду среднее значение сквозной задержки (от момента измерения на датчике до отображения в веб-интерфейсе) составило 1,37 ± 0,21 с. При увеличении числа устройств до 5 000 задержка возросла до 1,82 ± 0,34 с, оставаясь в пределах установленного требования (не более 2 с).

Пропускная способность MQTT-брокера при тестировании составила 48 500 сообщений/с при загрузке процессора 67%. Слой Apache Kafka

обеспечил буферизацию пиковых нагрузок, предотвращая потерю данных при кратковременных всплесках трафика. Коэффициент доставки сообщений (Delivery Rate) за период тестирования длительностью 72 часа составил 99,94%.

Таблица 2. Результаты нагрузочного тестирования системы

Число устройств	Задержка (сред.), с	CPU, %	Доставка, %
100	0,84 ± 0,12	12	99,99
1 000	1,37 ± 0,21	34	99,98
5 000	1,82 ± 0,34	67	99,95
10 000	2,47 ± 0,52	89	99,87

3.3. Функциональные возможности веб-интерфейса

Разработанный веб-интерфейс предоставляет следующие функциональные возможности. Интерактивная карта отображает все зарегистрированные в системе устройства в виде маркеров, цвет которых кодирует текущий статус устройства (зелёный — норма, жёлтый — предупреждение, красный — аварийный). При нажатии на маркер открывается информационная панель с последними значениями всех параметров и историческим графиком за выбранный период.

Реализована функция тепловых карт (heatmap) для пространственной визуализации распределения параметров по территории. Инструмент сравнения позволяет одновременно отображать данные с нескольких устройств на одном графике. Система поддерживает настройку персональных оповещений посредством веб-браузерных уведомлений и электронной почты при срабатывании пороговых значений.

Экспорт данных поддерживается в форматах CSV, JSON и GeoJSON. Реализован REST API с полной документацией в формате OpenAPI 3.0, что обеспечивает возможность интеграции с внешними системами. Ролевая модель

доступа (RBAC) разграничивает права администраторов, операторов и пользователей-наблюдателей.

3.4. Сравнение с аналогами

Сравнение разработанной системы с ближайшими аналогами — ThingsBoard Community Edition и AWS IoT Greengrass — показало следующее. По производительности система превосходит ThingsBoard при аналогичных аппаратных ресурсах: пропускная способность выше на 35%, задержка ниже на 22%. По сравнению с AWS IoT Greengrass разработанная система уступает в максимальном числе одновременно обслуживаемых устройств, однако обеспечивает полную автономность функционирования без зависимости от сторонних облачных сервисов и не требует абонентской платы. Открытая архитектура системы обеспечивает возможность модификации исходного кода под специфические требования заказчика — функция, недоступная в коммерческих решениях.

4. ОБСУЖДЕНИЕ

Полученные результаты свидетельствуют о том, что разработанная система успешно справляется с поставленными задачами в условиях реальной эксплуатации. Обеспечение задержки менее 2 секунд при 5 000 одновременно подключённых устройствах является достаточным для большинства практических применений в области экологического мониторинга и управления городской инфраструктурой, где критичность времени реакции не столь высока, как в промышленных системах управления реального времени.

Применение TimescaleDB вместо традиционных реляционных СУБД общего назначения позволило достичь 8-кратного ускорения типичных аналитических запросов к временны́м рядам, что подтверждает обоснованность данного архитектурного решения. Использование Apache Kafka в качестве промежуточного слоя обработки данных существенно повысило устойчивость системы к пиковым нагрузкам: в ходе стресс-

тестирования 15-секундный всплеск потока данных, превышающий номинальный в 3 раза, был поглощён без потери ни одного сообщения.

Необходимо отметить ряд ограничений настоящего исследования. Во-первых, экспериментальный стенд охватывал относительно небольшую территорию в пределах одного города, что не позволяет в полной мере оценить особенности работы системы в условиях высокой сетевой латентности при глобальном развёртывании. Во-вторых, тестирование проводилось в контролируемых условиях с предсказуемой нагрузкой; поведение системы при аномальных сценариях (одновременное отключение большого числа узлов, атаки типа «отказ в обслуживании») требует дополнительного изучения.

Перспективными направлениями дальнейших исследований являются: интеграция алгоритмов машинного обучения для предиктивного анализа данных и автоматического обнаружения аномалий; реализация федеративного обучения непосредственно на узлах сети в целях минимизации передаваемого трафика; поддержка протокола CoAP (Constrained Application Protocol) для устройств с экстремально ограниченными ресурсами; интеграция с международными стандартами сенсорных данных, в частности SensorThings API консорциума OGC.

Разработанная система также продемонстрировала высокую степень отказоустойчивости: в ходе длительного тестирования (72 часа непрерывной работы) не было зафиксировано ни одного незапланированного простоя серверных компонентов. Автоматический перезапуск сервисов, реализованный средствами Docker Compose с политикой restart: unless-stopped, обеспечил восстановление системы после искусственно вызванных сбоев в течение не более 8 секунд.

5. ЗАКЛЮЧЕНИЕ

В настоящей работе разработана и экспериментально верифицирована информационная система для сбора и визуализации данных с георасределённых датчиков. Система реализует многоуровневую

архитектуру, объединяющую полевые IoT-устройства на базе ESP32, MQTT-брокер Eclipse Mosquitto, потоковый процессор Apache Kafka, временну́ю базу данных TimescaleDB и веб-интерфейс на React.js с картографической визуализацией.

По итогам проведённого исследования сформулированы следующие основные выводы. Предложенная система обеспечивает сбор и отображение данных с задержкой менее 2 секунд при числе одновременно подключённых устройств до 5 000, что соответствует требованиям большинства практических применений в сфере экологического мониторинга и управления городской инфраструктурой. Использование TimescaleDB и Apache Kafka обеспечивает масштабируемость системы до 10 000 узлов при сохранении приемлемых характеристик производительности. Разработанный веб-интерфейс предоставляет интуитивно понятные инструменты пространственной и временной визуализации данных, включая тепловые карты и инструменты сравнения показаний нескольких устройств.

Открытая архитектура системы, документированный REST API и контейнеризация посредством Docker существенно упрощают её интеграцию с существующими корпоративными информационными системами и обеспечивают независимость от конкретного поставщика аппаратного или программного обеспечения. Предложенное решение может найти практическое применение в системах экологического мониторинга, сельскохозяйственных IoT-платформах, системах мониторинга городской инфраструктуры, а также в научных исследованиях, требующих непрерывного сбора данных с территориально распределённых измерительных пунктов.

СПИСОК ЛИТЕРАТУРЫ

1. Al-Fuqaha A., Guizani M., Mohammadi M., Aledhari M., Ayyash M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications // IEEE Communications Surveys & Tutorials. — 2015. — Vol. 17, No. 4. — P. 2347–2376.

2. Mineraud J., Mazhelis O., Su X., Tarkoma S. A Gap Analysis of Internet-of-Things Platforms // Computer Communications. — 2016. — Vol. 89–90. — P. 5–16.

3. Ray P. P. A Survey on Internet of Things Architectures // Journal of King Saud University — Computer and Information Sciences. — 2018. — Vol. 30, No. 3. — P. 291–319.

4. Lombardi M., Pascale F., Santaniello D. Internet of Things: A General Overview between Architectures, Protocols and Applications // Information. — 2021. — Vol. 12, No. 2. — P. 87.

5. Batty M. et al. Smart Cities of the Future // European Physical Journal. — 2012. — Vol. 214, No. 1. — P. 481–518.

6. Hunkeler U., Truong H. L., Stanford-Clark A. MQTT-S — A Publish/Subscribe Protocol for Wireless Sensor Networks // Proceedings of the 3rd International Conference on Communication Systems Software and Middleware (COMSWARE'08). — 2008. — P. 791–798.

7. Koziolek H., Burger A., Doppelhamer J. Self-Commissioning Industrial IoT-Systems in Process Automation: A Reference Architecture // Proceedings of ICSE-SEIP. — 2018.

8. Eriksson J., Gidlund M., Björkman M. A Heterogeneous IoT Architecture for Data Acquisition, Storage and Visualization in Smart Buildings // Sensors. — 2022. — Vol. 22, No. 18. — P. 6811.

9. Tsai C. W., Lai C. F., Chiang M. C., Yang L. T. Data Mining for Internet of Things: A Survey // IEEE Communications Surveys & Tutorials. — 2014. — Vol. 16, No. 1. — P. 77–97.

10. Nagowah S. D., Ben Sta H., Gobin-Rahimbux B. An Overview of Semantic Web Ontologies for IoT // Proceedings of the 6th International Conference on Enterprise Systems. — 2018. — P. 82–89.