Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Системное программное обеспечение вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

РАЗРАБОТКА И РЕАЛИЗАЦИЯ ИМИТАЦИОННОЙ МОДЕЛИ

ИСТОЧНИКА НЕОДНОРОДНЫХ ТЕЛЕМЕТРИЧЕСКИХ ДАННЫХ

(СЕРВЕР)

БГУИР КП 6-05-0611-05 109 ПЗ

Студент: группы 350501, М. В. Козлов

Руководитель: старший препода- ватель каф. ЭВМ Поденок Л. П.

МИНСК 2025

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой

\_\_\_\_\_\_\_\_ Б.В.Никульшин

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2025 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Козлову Максиму Владимировичу

1. Тема проекта: Разработка и реализация имитационной модели источника неоднородных телеметрических данных (клиент)

2. Срок сдачи студентом законченного проекта: *15 мая 2025 г.*

3. Исходные данные к проекту: *язык программирования C*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые

*Введение. 1. Обзор литературы. 2. Постановка задачи. 3. Системное проектирование. 4. Функциональное проектирование. 5. Разработка программных модулей. 6. Результат работы. Заключение. Список использованных источников. Приложения*

5. Перечень графического материала (с точным обозначением обязательных

чертежей и графиков):

1. *Схема структурная*
2. *Диаграмма классов*
3. *Ведомость документов*

6. Консультант по проекту *Поденок Л. П.*

7. Календарный график работы над проектом на весь период проектирования

(с обозначением сроков выполнения и трудоемкости отдельных этапов):

*раздел 1, 2 к 01.03 – 15 %;*

*разделы 3, 4 к 01.04 – 50 %;*

*разделы 5, 6, 7, 8, 9 к 01.05 – 80 %;*

*оформление пояснительной записки и графического материала к 15.05 – 100%*

*Защита курсового проекта с 29.05 по 09.06.*

РУКОВОДИТЕЛЬ  *Л. П. Поденок*

Задание принял к исполнению  *М. В. Козлов*

СОДЕРЖАНИЕ

[1 ОБЗОР ЛИТЕРАТУРЫ 7](#__RefHeading___Toc8822_1702547058)

[1.1 Обзор аналогов 7](#__RefHeading___Toc8824_1702547058)

[1.2 Постановка задачи 8](#__RefHeading___Toc8826_1702547058)

[2 ИСТЕМНОЕ ПРОЕКТИРОВАНИЕ 10](#__RefHeading___Toc8828_1702547058)

[2.1 Модуль конфигурации и управления 10](#__RefHeading___Toc8830_1702547058)

[2.2 Модуль имитации данных 10](#__RefHeading___Toc8832_1702547058)

[2.3 Модуль сетевого взаимодействия 10](#__RefHeading___Toc8834_1702547058)

[2.4 Модуль сериализации и формирования протокола 11](#__RefHeading___Toc8836_1702547058)

[2.5 Модуль вспомогательных средств (утилиты) 11](#__RefHeading___Toc8838_1702547058)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 12](#__RefHeading___Toc8840_1702547058)

[4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 12](#__RefHeading___Toc8842_1702547058)

[5 РЕЗУЛЬТАТ РАБОТЫ 12](#__RefHeading___Toc8844_1702547058)

[6 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 13](#__RefHeading___Toc8846_1702547058)

[7 ПРИЛОЖЕНИЕ А 14](#__RefHeading___Toc8848_1702547058)

[8 ПРИЛОЖЕНИЕ Б 15](#__RefHeading___Toc8850_1702547058)

[9 ПРИЛОЖЕНИЕ В 16](#__RefHeading___Toc8852_1702547058)

[10 ПРИЛОЖЕНИЕ Г 17](#__RefHeading___Toc8854_1702547058)

[11 ПРИЛОЖЕНИЕ Д 18](#__RefHeading___Toc8856_1702547058)

ВВЕДЕНИЕ

В современную эпоху цифровизации и повсеместного распространения IoT, систем промышленной автоматизации, аэрокосмических комплексов и систем мониторинга окружающей среды, непрерывный сбор и анализ данных от разнообразных датчиков и устройств стал критически важным. Эти данные, известные как телеметрия, предоставляют бесценную информацию о состоянии объектов, протекании процессов и параметрах окружающей среды в режиме реального времени. Эффективное управление, контроль и принятие обоснованных решений во многих отраслях напрямую зависят от своевременности и достоверности получаемых телеметрических данных.

Однако разработка, тестирование и отладка систем, предназначенных для сбора и обработки таких данных, сталкивается с существенными трудностями. Доступ к реальному оборудованию, генерирующему телеметрию, может быть ограничен, дорог или невозможен на ранних этапах разработки. Кроме того, реальные источники данных часто не позволяют гибко конфигурировать параметры генерируемых потоков для тестирования пограничных случаев, сценариев отказа или условий высокой нагрузки. Особенно остро эта проблема стоит при работе с **неоднородными** данными, когда одна система должна обрабатывать информацию разного типа (например, числовые показания температуры и давления, геолокационные координаты, текстовые статусные сообщения, бинарные данные изображений) и с разной частотой поступления от множества источников.

Для преодоления этих препятствий и обеспечения эффективного цикла разработки клиентских приложений необходимы инструменты, способные **имитировать** поведение реальных источников телеметрических данных. Создание реалистичной имитационной модели, способной генерировать потоки неоднородных данных, схожие по своим характеристикам (типам, частоте, диапазонам значений, динамике изменений) с реальными, становится ключевой задачей. Такая модель, реализованная в виде серверного приложения, может служить надежным и контролируемым источником данных для разработки и всестороннего тестирования клиентских систем сбора телеметрии.

**Целью данного проекта** является разработка и реализация имитационной модели источника неоднородных телеметрических данных в виде серверного приложения с использованием языка программирования C. Клиент должен быть способен принимать данные с одного или нескольких виртуальных устройств, принимать различные типы данных (числовые, строковые, структурированные, возможно, бинарные) с настраиваемой периодичностью и характеристиками, а также принимать эти данные по сети с использованием протокола TCP/IP для клиентского приложения сбора. Особое внимание будет уделено гибкости работы с этими данными для пользователя.

**Актуальность** данной работы обусловлена растущей потребностью в надежных и гибких инструментах для разработки и тестирования сложных систем сбора и обработки телеметрической информации в различных областях, от IoT и умных городов до промышленного мониторинга и научных исследований.

# ОБЗОР ЛИТЕРАТУРЫ

## Обзор аналогов

Для качественной оце­­нки разрабатываемого решения необходимо провести обзор аналогов. Важно отметить, что найти полный аналог, реализующий серверную имитацию­ неоднородных телеметрических данных на C++ с TCP-соединением для низколатентного сбора, затруднительно. Однако существуют классы инструментов, реализующие схожие концепции: предоставление серверной части для тестирования клиентских приложений и имитация источников данных, особенно в контексте тестирования API и разработки для IoT. Ниже рассмотрены некоторые из них.

В первую очередь стоит рассмотреть общие **Mock/Stub серверы**, предназначенные для тестирования API и сетевых клиентов. Ярким представителем является **MockServer** [1]. Это гибкий инструмент, позволяющий имитировать любые системы, с которыми код взаимодействует по протоколам HTTP/S. Он дает возможность детально настраивать ожидаемые запросы и соответствующие им ответы сервера, которые могут быть как статическими, так и динамически генерируемыми. Концептуально он схож с разрабатываемым проектом, так как предоставляет серверную часть для отладки и тестирования клиента. Ключевое отличие заключается в сильной ориентации MockServer (и аналогичных инструментов, таких как WireMock [2]) на веб-протоколы и часто на реализацию на Java или других языках, в то время как данный проект фокусируется на произвольном TCP-соединении, генерации специфичных телеметрических данных и реализации на C++.

Другим классом инструментов являются **симуляторы устройств и данных для платформ IoT**. Платформы вроде **AWS IoT Core [3]** или **Azure IoT Hub [4]** предлагают разработчикам SDK и примеры для создания виртуальных устройств, которые подключаются к облаку и отправляют потоки телеметрии (часто по MQTT или HTTPS), имитируя реальные датчики. Аналогично, open-source платформы, например, **ThingsBoard [5]**, могут включать компоненты для симуляции подключенных устройств. Эти инструменты полезны для тестирования облачной инфраструктуры или приложений, работающих с конкретной платформой. Их основная задача – генерация данных, схожих с телеметрическими – совпадает с целью проекта. Однако, они обычно функционируют как клиенты, привязанные к экосистеме своей платформы и её протоколам, в то время как разрабатываемый проект представляет собой автономный сервер, ожидающий подключения от кастомного клиента через TCP/IP и не зависящий от внешней инфраструктуры.

Наконец, часто для специфических задач или внутреннего тестирования разрабатываются **самописные скрипты-симуляторы** на языках вроде Python или Node.js. Они могут открывать TCP-порт и отправлять сгенерированные данные подключившемуся клиенту. Такие решения максимально гибки, но часто являются узкоспециализированными, могут не иметь удобных средств конфигурации и управления, а также могут уступать C++ реализации в производительности при генерации и отправке больших объемов данных.

Таким образом, хотя существуют инструменты для имитации серверов и генерации данных, они либо ориентированы на иные протоколы (HTTP, MQTT), либо являются частью крупных платформ, либо представляют собой менее структурированные решения. Разработка специализированного, автономного сервера на C++ для имитации источников неоднородных телеметрических данных с передачей по произвольному протоколу поверх TCP/IP остается актуальной задачей, особенно в контексте тестирования низколатентных систем сбора данных, что подтверждает практическую значимость разрабатываемого в данном проекте решения.

## Постановка задачи

После рассмотрения существующих аналогов можно заключить, что многие из них либо ориентированы на специфические протоколы (например, HTTP, MQTT) и платформы (AWS, Azure), либо представляют собой комплексные системы с широким функционалом, не всегда необходимым и реализуемым в рамках курсового проекта за отведенное время. В то же время, задача создания автономного, гибко настраиваемого сервера для имитации именно неоднородных телеметрических данных с передачей по чистому TCP/IP для тестирования специализированных клиентов остается актуальной.

Исходя из этого, были определены следующие ключевые задачи, которые будут решены в рамках данного курсового проекта:

* **Реализация серверного приложения:** Сервер должен быть способен прослушивать и принимать входящие TCP/IP соединения на заданном порту.
* **Имитация нескольких источников:** Сервер должен моделировать работу одного или нескольких виртуальных источников телеметрических данных.
* **Генерация неоднородных данных:** Должна быть реализована генерация данных различных предопределенных типов (например, числовые значения с плавающей точкой, целые числа, строки состояния, структурированные данные вроде GPS-координат).
* **Конфигурируемая генерация:** Необходимо предусмотреть возможность настройки параметров генерации для каждого источника и типа данных (например, частота отправки, диапазон значений, характер изменений).
* **Передача данных клиенту:** Сервер должен отправлять сгенерированные данные подключенным клиентам в соответствии с протоколом обмена.
* **Стабильная работа:** Сервер должен обеспечивать стабильную работу при подключении клиента и передаче данных в течение времени, необходимого для тестирования.

В качестве языка программирования выбран **C++**. Этот выбор обусловлен несколькими причинами: высокая производительность, которая может быть важна при генерации и отправке больших потоков данных; возможность низкоуровневой работы с сетевыми протоколами и представлением данных, что является ценным опытом в рамках курса "Операционные системы и сетевое программирование"; поддержка объектно-ориентированного подхода для логической организации имитационной модели (источники, типы данных, сервер); а также наличие у исполнителя опыта работы с данным языком.

Для реализации сетевого взаимодействия планируется использовать библиотеку **Boost.Asio**, предоставляющую мощные, кроссплатформенные и хорошо зарекомендовавшие себя абстракции для асинхронного (и синхронного) сетевого программирования на C++. Для эффективной упаковки (сериализации) неоднородных данных перед отправкой по сети будет рассмотрено применение библиотеки **Protocol Buffers**. Управление параметрами симуляции (например, количество источников, типы и частота данных) предполагается осуществлять через конфигурационные файлы, так как разработка полноценного графического интерфейса для серверного приложения не входит в основные задачи проекта. Сборка проекта будет управляться с помощью системы **CMake**.

Данный набор средств разработки и определенный перечень функциональных требований позволяют реализовать ключевые задачи проекта в установленные сроки и создать работоспособный сервер-симулятор телеметрических данных, необходимый для разработки и тестирования соответствующего низколатентного клиента сбора данных.

# СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

После определения требований к функционалу разрабатываемого серверного приложения его следует разбить на функциональные блоки. Такой подход упростит понимание архитектуры проекта, позволит более эффективно распределить задачи разработки и тестирования, обеспечит гибкость и масштабируемость программного продукта в будущем путем модификации или добавления новых блоков.

## ****Модуль конфигурации и управления****

Модуль конфигурации и управления необходим для задания начальных параметров работы сервера и координации запуска основных компонентов приложения. Его задача заключается в считывании настроек симуляции (например, из файла или аргументов командной строки), таких как сетевой порт для прослушивания, количество и типы имитируемых источников данных, их характеристики (частота, диапазон значений). Также этот модуль отвечает за инициализацию других функциональных блоков и управление основным жизненным циклом приложения.

## ****Модуль имитации данных****

Модуль имитации данных является ядром серверного приложения и необходим для генерации потоков телеметрической информации. Тут осуществляется весь функционал, связанный с созданием данных:

* Моделирование работы одного или нескольких виртуальных источников.
* Генерация значений различных типов (числовые, строковые, структурированные) в соответствии с заданными в конфигурации параметрами (периодичность, диапазон, характер изменений).
* Управление временными интервалами генерации данных для каждого источника.

## ****Модуль сетевого взаимодействия****

Модуль сети предназначен для обеспечения связи сервера с клиентскими приложениями и необходим для передачи сгенерированных телеметрических данных. Данный модуль должен отвечать за:

* Прослушивание указанного TCP-порта на предмет входящих подключений.
* Принятие и обработку соединений от клиентов.
* Управление активными клиентскими сессиями.
* Отправку подготовленных телеметрических данных подключенным клиентам.
* Обработку закрытия соединений и сетевых ошибок.

## ****Модуль сериализации и формирования протокола****

Модуль сериализации и формирования протокола необходим для преобразования внутренних структур данных приложения в формат, пригодный для передачи по сети. Его задача – упаковать сгенерированные модулем имитации данные (числа, строки, структуры) в последовательность байт в соответствии с согласованным с клиентом протоколом обмена.

## ****Модуль вспомогательных средств (утилиты)****

Модуль вспомогательных средств необходим для предоставления общих функций, используемых другими модулями приложения. Сюда входят компоненты для логирования событий и ошибок, классы для представления общих структур данных, вспомогательные функции для работы со временем и строками, что упрощает разработку и поддержку основного кода приложения.

# ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

# РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

# РЕЗУЛЬТАТ РАБОТЫ

ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] **MockServer** [Электронный ресурс]. – **MockServer.** – Режим доступа: https://www.mock-server.com/ – Дата доступа: 10.03.2025

[2] WireMock [Электронный ресурс]. – WireMock. – Режим доступа: https://wiremock.org/ – Дата доступа: 10.03.2025

[3] A**mazon** [Электронный ресурс]. – **AWS IoT Core** – Режим доступа: https://aws.amazon.com/ru/iot-core/ – Дата доступа: 10.03.2025

[4] M**icrosoft** [Электронный ресурс]. – **Azure IoT Hub** – Режим доступа: https://azure.microsoft.com/en-us/products/iot-hub – Дата доступа: 10.03.2025

[5] E**ngix-ltd** [Электронный ресурс]. – **ThingsBoard**. – Режим доступа: https://engix-ltd.github.io/ – Дата доступа: 10.03.2025

# ПРИЛОЖЕНИЕ А

(Обязательное)

**Ведомость документов**

# ПРИЛОЖЕНИЕ Б

(Обязательное)

**Структурная схема**

# ПРИЛОЖЕНИЕ В

(Обязательное)

**Диаграмма классов**

# ПРИЛОЖЕНИЕ Г

(Обязательное)

**Схема программы**

# ПРИЛОЖЕНИЕ Д

(Обязательное)

**Код программы**