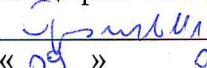


Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Радиофизический факультет

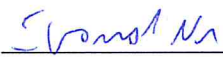
ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ООП
канд. физ.-мат. наук, доцент
 М.Л. Громов
« 09 » 06 20 23 г.

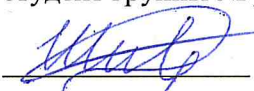
ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ ЗАДЕРЖЕК В СЕТЯХ

по направлению подготовки 03.03.03 Радиофизика
направленность (профиль) «Радиофизика, электроника и информационные системы»

Шилов Станислав Олегович

Руководитель ВКР
канд. физ.-мат. наук, доцент
 М.Л. Громов
подпись
« 09 » 06 20 23 г.

Автор работы
студент группы № _____
 С.О. Шилов
подпись
« 09 » 06 20 23 г.

Министерство науки и высшего образования Российской Федерации.
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Радиофизический факультет

УТВЕРЖДАЮ

Руководитель ООП

канд. физ.-мат. наук, доцент

 М.Л. Громов

« 03 » 10 2022 г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы бакалавра обучающемуся
Шилов Станислав Олегович

(Ф.И.О. обучающегося)

по направлению подготовки 03.03.03 Радиофизика, направленность (профиль)
«Радиофизика, электроника и информационные системы»

1. Тема выпускной квалификационной работы бакалавра
Имитационное моделирование задержек в сетях

2. Срок сдачи обучающимся выполненной выпускной квалификационной работы:

а) на кафедре – _____

б) в ГЭК – _____

3. Исходные данные к работе:

Объект исследования – Сеть вершины которой соответствуют узлам обработки (информации, материальных ресурсов, транспортного потока и т.д.), а дугам пути непосредственной передачи результатов обработки между узлами.

Предмет исследования – Временные задержки возникающие в сетях обработки.

Цель исследования – Создать программный инструмент имитационного моделирования сетей обработки с возможностью визуального контроля возникающих в сети временных задержек.

Задачи:

Рассмотреть подходы, оценить адекватность имитационного моделирования.

Рассмотреть инструменты имитационного моделирования.

Разработать инструмент имитационного моделирования обрабатывающей сети, с возможностью визуального контроля.

Провести эксперимент с построенной системой.

Методы исследования

Имитационное моделирование

4. Краткое содержание работы

В процессе работы будет разработан инструмент имитационного моделирования, с возможностью визуального контроля.

Руководитель выпускной квалификационной работы

Доцент, НИ ТГУ

(должность, место работы)

 (подпись)

М.Л. Громов

(И.О. Фамилия)

Задание принял к исполнению

Студент, НИ ТГУ

(должность, место работы)

 (подпись)

С.О. Шилов

(И.О. Фамилия)

АННОТАЦИЯ

Выпускная квалификационная работа бакалавра содержит 34 с., 30 рис., 11 источников, 1 приложение.

ИМИТАЦИОННАЯ МОДЕЛЬ, ЗАДЕРЖКИ, СИСТЕМЫ, СЕТИ, NODE-RED.

Целью данной работы является разработка инструмента позволяющего анализировать задержки в сетях с помощью имитационного моделирования.

В процессе выполнения выпускной квалификационной работы было сделано следующее:

- 1) Рассмотрены подходы для анализа задержек в сетях и был выбран свой;
- 2) Рассмотрены готовые инструменты для анализа задержек, в рамках выбранного подхода;
- 3) Изучена среда потокового программирования Node-RED;
- 4) Проведены эксперименты с целью исследования работы систем, моделируемых с помощью Node-RED;
- 5) Составлена формальная модель с помощью компьютерной игры Factorio;
- 6) Проведено имитационное моделирование формальной модели;
- 7) Расширен инструмент Node-RED;
- 8) Выявлены узкие места моделируемой системы;
- 9) Оптимизирована имитационная модель.

ОГЛАВЛЕНИЕ

ОГЛАВЛЕНИЕ.....	4
ВВЕДЕНИЕ	5
1 Анализ предметной области	6
1.2 Теоретический подход.....	6
1.2.1 Системы массового обслуживания	6
1.2.2 Поточковый граф	8
1.3 Обзор инструментов для проведения анализа	8
1.3.1 Arena.....	9
1.3.2 Simulink.....	10
1.3.3 AnyLogic	11
1.3.4 Среда разработки Node-RED.....	12
2 Рассматриваемые системы	14
3 Результаты проведенной работы.....	16
3.1 Проведение экспериментов.....	16
3.2 Проведение имитационного моделирование	23
3.3 Дополнение среды разработки Node-RED	27
ЗАКЛЮЧЕНИЕ.....	33
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ.....	34
ПРИЛОЖЕНИЕ А.....	35

ВВЕДЕНИЕ

Каждый завод, фабрика, создатель программного обеспечения в нашем мире борется с задержками в своем производстве или продуктах. Задержки в производстве влияют на объем изготавливаемой продукции, а, следовательно, на объемы продаж. Если рассматривать случай программного обеспечения, то можно говорить о положительном или отрицательном опыте использования какого-либо приложения пользователем, если в каких-либо местах приложение зависает, встает и т.д. то у пользователя формируется отрицательный опыт использования, и он попытается найти аналог такого приложения.

Таким образом уменьшая задержки в производстве или работе своего продукта, можно увеличить объемы производства, улучшить логистику или улучшить опыт использования какого-либо приложения.

Целью данной работы является разработать программный инструмент имитационного моделирования сетей обработки с возможностью визуального контроля возникающих в сети временных задержек.

Для достижения поставленной цели необходимо провести литературный обзор по теме теоретических и практических подходов при анализе систем на задержки. Выбрать инструмент для выполнения работы. Проанализировать исследуемые системы и провести эксперименты для того, чтобы рассмотреть поведение выбранного инструмента при проведении моделирования. Дополнить выбранную технологию новым функционалом, для облегчения анализа систем.

В первом разделе описан анализ предметной области.

Во втором разделе будет описание исследуемых систем и пример их моделирования с помощью технологии Node-RED.

В третьем разделе будут рассмотрены проведенные эксперименты, а также расширение среды Node-RED.

Заключение расскажет об итогах по проделанной работе.

1 Анализ предметной области

Производственные системы и программное обеспечение имеет различную структуру. В данной работе для изучения выбраны системы, представляющие собой сети, т.е. имеющие некоторые производственные узлы, соединёнными между собой чем-либо, будь то канал связи, конвейер или вызов функций в программном обеспечении. Для реализации анализа таких сетей на предмет задержек, можно использовать множество методов. Реализации всех существующих методов можно разбить на два подхода теоретический и практический.

1.2 Теоретический подход

Как сказано выше, методов для анализа систем существует достаточно большое множество. Оптимальным вариантом для теоретического подхода будет использование теоретических моделей. Теоретическое моделирование предполагает постановку гипотезы, обработку теорий и предположений, а также проведения анализа вариантов решений.

Так как рассматриваемые системы имеют вид сетей, то можно использовать математические модели, подходящие для анализа сетей на предмет задержек. Рассмотрим наиболее подходящие из них, чтобы понять какую математическую модель можно использовать для выполнения работы.

1.2.1 Системы массового обслуживания

Теория систем массового обслуживания (СМО) – это область прикладной математики, являющаяся обособленной частью теории случайных процессов, с комплексом задач для самостоятельного исследования. Она базируется на теории вероятностей и математической статистике. Теория систем массового обслуживания в настоящее время имеет широкую область применения не только в экономике, но и в других жизненно значимых отраслях: в социальной сфере, в военном деле, в области организации производства и обслуживания.

Теория массового обслуживания занимается анализом процессов в системах обслуживания, производства и управления, в которых однородные действия (события) повторяются многократно (предприятия торговли, коммерческие банки, автоматические линии производства, медицинские учреждения, страховые организации, транспортные системы и др.). Предметом теории массового обслуживания является установление зависимости между основными характеристиками системы обслуживания (число каналов

обслуживания, характер входного потока заявок, которые необходимо обслужить, производительность отдельно взятого канала, пр.) с целью улучшения управления системами. Структурная схема СМО показана на рисунке 1.

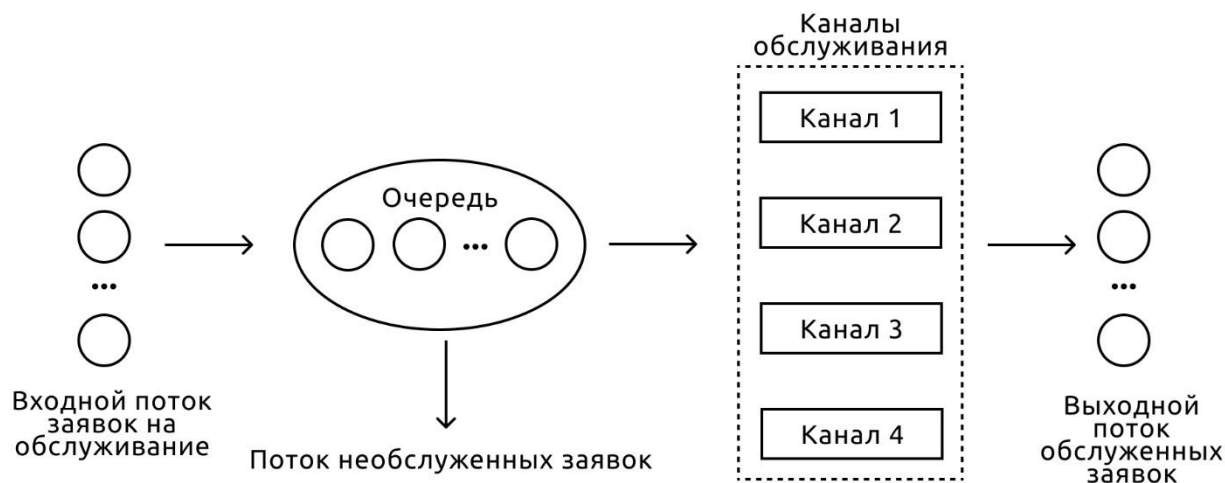


Рисунок 1 - Структурная схема система массового обслуживания

Основные задачи теории массового обслуживания заключаются в следующем:

- 10) Установление зависимости эффективности работы системы массового обслуживания от её организации.
- 11) Решение различных оптимизационных задач, связанных с функционированием системы массового обслуживания.
- 12) Выработка рекомендаций по рациональному построению системы массового обслуживания для обеспечения высокой эффективности ее функционирования.

Все задачи теории массового обслуживания носят оптимизационный характер и в конечном итоге направлены на определение такого варианта работы системы, при котором будет обеспечен минимум суммарных затрат от простоев каналов обслуживания, потерь времени и ресурсов на обслуживания и пр.

Методология теории массового обслуживания включает все основные элементы теории случайных процессов: Марковские случайные процессы, потоки заявок, граф состояний, системы обыкновенных дифференциальных уравнений для вероятностей состояний, др.

Таким образом можно сделать вывод по системам массового обслуживания - этот метод подходит для описания процессов, в которых клиенты сталкиваются с задержками при ожидании на обслуживание, и позволяет вычислить различные показатели, такие как среднее время ожидания, вероятность ожидания и т.д.

1.2.2 Поточковый граф

Поточковый граф – это математическая модель для моделирования и анализа производственных и информационных систем, представляемых в виде потоков, иначе сетей. Можно представить это как систему, в которой данные переходят от одной вершины к другой, в самих же вершинах над данными производятся некие операции. Сам граф является ориентированным и таким образом переход данных образуют поток.

Он состоит из вершин, которые представляют отдельные операции над данными, и ребер, которые соединяют вершины и определяют передачу данных между операциями. Как и говорилось ранее, такой граф является ориентированным, что обеспечивает его однонаправленность, но еще в нем есть входная точка, из которой идет начало системы-сети, а также выходная точка. Таким образом можно проследить производство каких-либо деталей от их входа в систему до самого выхода. Структурная схема графа потока управления показана на рисунке 2.

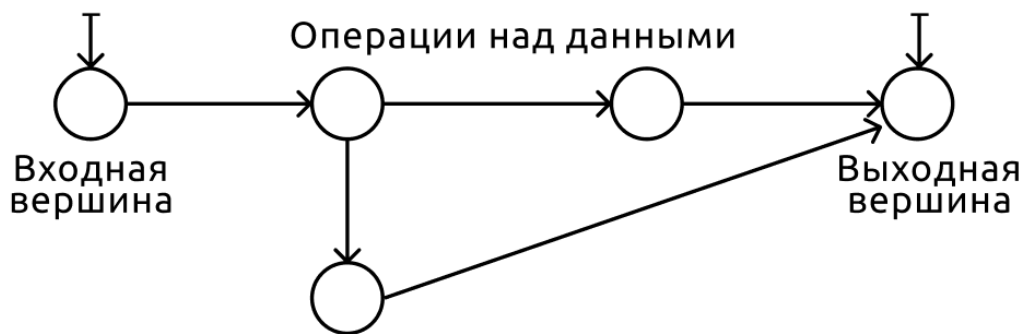


Рисунок 2 - Структурная схема потокового графа

Имея знания о том, какие операции над данными совершает каждая вершина, можно проанализировать структурированное производство систем в виде потоков, оптимизировать их и вычислить узкие места. Так, например с некоторой долей абстракции можно представить систему по обработке железной руды в виде потокового графа, где вершинами будут обрабатывающие узлы, выполняющие некие операции: переплавка, полировка и так далее. Ребрами будут являться конвейерные линии между обрабатывающими узлами.

1.3 Обзор инструментов для проведения анализа

Рассчитывать системы с помощью теоретических моделей достаточно трудоемкая задача. Поэтому для ускорения процесса анализа систем на предмет задержки, легче воспользоваться программным обеспечением для этого. Кроме облегчения процесса

анализа, такой способ намного ускорит получение результата, потому что в программное обеспечение сразу же заложен готовый функционал, покрывающий большую часть работы.

Из широкого выбора программного обеспечения для анализа систем приближенных к сетям, хорошим вариантом будет использование имитационного моделирования. Имея знание о структурном составе системы можно воссоздать симуляцию ее работы, для проведения анализа. Но еще одним важным преимуществом будет и то, что можно варьировать параметры системы либо для того, чтобы извлечь максимальное количество данных для анализа, либо для оптимизации работы системы выбрав наилучшие варианты распределения «материалов» участвующих в процессе производственной деятельности.

1.3.1 Arena

Arena – программное обеспечение для имитационного моделирования, созданное Systems Modeling Corporation. Позволяет создавать подвижные компьютерные модели, используя которые можно представить многие реальные системы. Данное программное обеспечение было выпущено в далеком 1993 году, но на данный момент последней версией является Arena 3.0 ориентированная на операционную систему Windows и использующая ее интерфейсные возможности.

Модель в среде Arena представляет собой граф, вершинами которого являются модули. Сами же модули связаны между собой, а между ними перемещаются транзакты – динамические объекты модели, в реальном мире это деталь, клиент, документ и т.д.

Для создания модели в среде Arena, необходимо:

- 1) Назначить транзакты, ресурсы и другие объекты системы;
- 2) Выбрать и связать между собой необходимые модули;
- 3) Задать для каждого модуля свои параметры;
- 4) Задать характеристики для всей системы в целом.

Преимущества:

- 1) Система очень проста в использовании;
- 2) Обладает объектно-ориентированным интерфейсом;
- 3) Используется язык программирования SIMAN для дополнительных возможностей моделирования;
- 4) Имеется визуализация результатов с помощью Cinema Animation;
- 5) Имеются некоторые шаблоны готовых решений.

Недостатки:

- 1) Даже третья версия вышла довольно давно и адаптирована к версии Windows XP;

2) Язык программирования SIMAN является не столь распространённым в широких массах, поэтому потребует дополнительного изучения, чтобы работать в среде Arena;

3) Визуализация Cinema Animation используемая в последней на данный момент версии Arena, является очень старой.

1.3.2 Simulink

Simulink – является приложением визуального программирования к пакету MATLAB. Данное приложение представляет собой среду динамического междисциплинарного моделирования сложных технических систем. Позволяет осуществлять моделирование во времени поведения динамических нелинейных систем. Эта среда базируется на модельно-ориентированном проектировании. Хотя Simulink и входит в пакет MATLAB, он является достаточно самостоятельным инструментом, поэтому при работе с ним нет необходимости знать весь MATLAB.

Графический интерфейс Simulink имеет достаточно широкую функциональность за счет возможности подключения различных библиотек и интегрирования с техническим языком программирования MatLab. Simulink имеет возможность автоматической генерации кода на языке C, что позволяет реализовывать системы в реальном времени. Библиотека этой среды имеет такие разделы: библиотека непрерывных элементов (дифференциатор, интегратор и т.д.), библиотека дискретных элементов, библиотеки источников и приемников сигналов, различные блоки подсистем и функции, как готовые математические, так и пользовательские, созданные с помощью средств программирования MatLab.

Преимущества:

- 1) Есть возможность к созданию пользовательских блоков с помощью средств программирования;
- 2) Присутствует 3D графика;
- 3) Широкий пакет инструментов в виде библиотек;
- 4) Есть возможности разработки систем в реальном времени.

Недостатки:

- 1) Является очень тяжеловесной программой, например процесс моделирования разделяется на два этапа: инициализация и само моделирование. Процесс инициализации очень похож на процесс компиляции программ и может продолжаться довольно длительное время;
- 2) Данная программа не является бесплатной и стоит достаточно дорого, кроме того, дополнительные пакеты для моделирования продаются за отдельную плату;

- 3) Не является кроссплатформенной;
- 4) Нет русской локализации.

1.3.3 AnyLogic

AnyLogic – платформа, предназначенная для имитационного моделирования любых бизнес систем, разработанная компанией AnyLogic Company. Первая версия вышла еще в 2000 году под индексом 4, т.к. продолжал индексацию предшественника – программы для моделирования COVERS. Однако хоть история продукции и начинается в 90-х годах прошлого столетия, популярен продукт стал только после выхода AnyLogic 5 в 2003 году. Пятая версия стала ориентированная на бизнес-моделирование, и разработка моделей расширила свой круг областей от сферы обслуживания до промышленного ракетостроения. На сегодняшний день последней вышедшей версией является восьмая.

Данный продукт обладает поистине обширным функционалом, даже его название говорит об этом, AnyLogic называли так, потому что поддерживает все три известные виды моделирования: системную динамику, дискретно-событийное моделирование и агентное моделирование. Моделирование в этой программе можно расширить с помощью языка программирования Java. Кроме расширения моделей есть возможность создания Java апплетов, которые могут быть открыты любым браузером, что позволяет размещать модели на веб-сайтах. AnyLogic включает в себя следующий набор стандартных библиотек: библиотека моделирования процессов, пешеходная библиотека, железнодорожная библиотека, библиотека моделирования потоков, библиотека производственных систем.

С восьмой версии стало возможна интеграция с AnyLogic Cloud. Это веб-сервис, позволяющий хранить, запускать и делиться имитационными моделями. Разработчики могут загружать готовые модели в облако, далее настроив панель управления моделью, с ней можно будет работать онлайн и не только на десктопном устройстве, а также на мобильном, включая смартфоны. Для уменьшения ресурсопотребления, сами модели исполняются на сервере, в клиентской части просто происходит анимированная отрисовка.

Преимущества:

- 1) Одни из лучших функциональных возможностей в мире;
- 2) Совместим с языком программирования Java, что дает бонус к расширению моделей;
- 3) Есть интеграция с AnyLogic Cloud, чтобы хранить и использовать свои модели в онлайн, экономя при этом свои ресурсы;
- 4) Java апплеты позволяют размещать модели на любых веб-ресурсах, не привязываясь к AnyLogic Cloud

5) Работа в онлайн не только экономит ресурсы, но еще и является кроссплатформенной

6) Есть полная локализация на русский язык.

Недостатки:

1) К сожалению, данное программное обеспечение является очень дорогостоящим;

2) Из-за обширного функционала, требует очень много времени для освоения.

Таким образом были разобраны самые популярные программные средства для проведения имитационного моделирования. Исходя из достоинств и недостатков, лучшим выбором для проведения анализа на предмет задержек был бы AnyLogic, за счет его огромной функциональности. Но целью данной работы является предоставить свой инструмент, который смог бы вобрать в себя лучшее из представленных программ.

1.3.4 Среда разработки Node-RED

Node-RED – это инструмент разработки на основе потоков для визуального программирования, первоначально разработанный IBM (от англ. International Business Machines) для объединения аппаратных устройств, API -интерфейсов и онлайн-сервисов в рамках Интернета вещей.

Node-RED предоставляет графический редактор работающий исключительно в веб-браузере, но разворачивается на устройстве пользователя. Развертку среды Node-RED возможно осуществить на всех современных операционных системах, включая множество дистрибутивов Linux, Raspberry Pi и даже облачные платформы, например IBM Cloud, AWS Cloud, Azure и другие. Данный инструмент никогда не рассматривался с точки зрения программного обеспечения для имитационного моделирования, но далее будет показано, как его применить в этой сфере.

В основе работы Node-RED лежит платформа Node.js. Данная платформа превращает язык программирования JavaScript из узкоспециализированного, работающего только в браузерах, в язык общего назначения. В современном мире Node.js активно используют для создания серверных приложений, руководствуясь парадигмой «JavaScript для всего». Так нет нужды использовать несколько разных языков программирования при создании приложений, когда все будет сделано с помощью одного. Исходный код Node-RED предоставлен в открытый доступ, так каждый желающий может опробовать его функционал, а также при желании помочь в разработке представленного программного продукта.

В Node-RED все программы представляются потоками. Кроме того, каждая программа в этой среде так и называется – *поток*. В этом потоке пользователи соединяют

доступные элементы графического интерфейса – узлы между собой. Существует достаточно большое множество доступных узлов, от ветвлений с различными условиями прохождения потока, до узлов предназначенных для организации связи по MQTT протоколу. Большим достоинством является и то, что можно создавать свои собственные узлы и способы как это сделать очень хорошо описаны в документации. В случае возникновения каких-либо ошибок, отладочная информация в веб-интерфейсе всегда даст ответ, в каких местах произошли ошибки и почему. Пример простого потока, выводящего время в консоль представлен на рисунке 3.

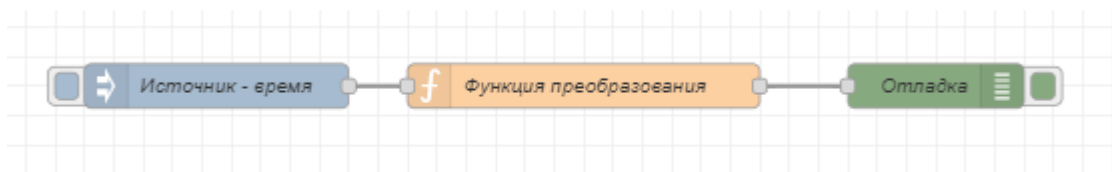


Рисунок 3 - пример программы в среде разработки Node-RED

В представленной среде в общем случае по потоку передвигаются некоторые данные, заданные пользователем. Данные представляются в виде JSON (от англ. JavaScript Object Notation) объекта - *сообщения*. Таким объектом во время выполнения программы, такими сообщениями можно манипулировать, для считывания данных, записи или удаления. Если рассматривать такой инструмент с некоторой долей абстракции, то в качестве данных можно представить исходные характеристики материалов какого-либо производства, например если в реальном производстве по конвейерным линиям проходит медная руда, то в сообщении можно задавать ее массу, качество очистки от других металлов и так далее.

По мере прохождения сообщения в потоке, его можно изменять на каждом узле производства, так чтобы на выходе получились данные о характеристиках готового продукта. Рассматривая данный способ и дополняя сообщения данными о временных характеристиках и других данных для анализа задержек, можно использовать Node-RED в качестве инструмента имитационного моделирования сетей на предмет задержек.

Достоинства:

- 1) Является полностью бесплатным программным обеспечением;
- 2) Написание программ в данной среде разработки намного быстрее, чем в других инструментах для имитационного моделирования;
- 3) Является полностью кроссплатформенным ПО;
- 4) Легок в освоении благодаря обширной документации, в которой описано не только использование Node-RED, но еще и его развертывание, дополнение новым функционалом и встраивание в другие приложения;

- 5) Достаточно обширные возможности встроенных узлов;
- 6) Локализован на русский язык;
- 7) Возможность экспорта своих потоков, а также импорта потоков других пользователей в несколько действий;
- 8) Возможность написания собственных узлов, либо использование узла *Function*, для написания необходимой функции на языке JavaScript.

Недостатки:

- 1) Исключена возможность применения асинхронности языка JavaScript в узле *Function*.

2 Рассматриваемые системы

Определившись с инструментом моделирования, необходимо понять, как структурно выглядят системы, с которыми происходит работа. Уже упоминалось, что в данной работе рассматриваются системы в виде потоков, для простоты будем рассматривать далее только производящие какой-либо готовый продукт предприятия. Получается, что такие системы могут иметь один или более входов, которые представляют из себя точки входа условных материалов для дальнейшего преобразования (обработки, слияния, отбрасывания неподходящих и т.п.). Выходов у такой системы тоже может быть один или более в зависимости от назначения производящего завода или устройства его логистики.

Между входными точками, с которых все начинается до выходов с готовой продукцией будут находиться производственные узлы. Узлы различаются между собой физикой своего поведения и являются главными причинами задержек в системе, так как на работу узла уходит некоторое время. В зависимости от характеристик приходящих на узел материалов время его работы будет либо увеличиваться, либо уменьшаться. Теперь можно сказать, что такие задержки будут являться случайной величиной независимой от величин задержек на предшествующих узлах.

Все узлы системы связаны между собой конвейерными линиями, по которым проходят различные материалы, детали и так далее. Для удобства можно абстрагироваться от реального мира и таким образом можно говорить, что по конвейерам всегда проходят некие сообщения. Входными точками системы тогда будут являться источники сообщений, а выходными точками приемники.

Теперь можно рассмотреть различия между производственными узлами. Такие узлы как уже сказано отличаются друг от друга физикой своего поведения. Можно выделить два типа. Первый тип представляет из себя узел, производящий обработку входящих

сообщений в системе. В реальном мире он работает с объектами одной природы, например плавильная печь, с помощью нее можно расплавить металлическую руду в какую-либо деталь для последующей обработки. Второй тип будет представлять сборочный узел, работающий с сообщениями различной природы, так, например для сборки школьных парт необходимо, чтобы с нескольких конвейеров пришли и ножки, и столешница для них.

Теперь рассмотрим, как выглядит изучаемая система, можно представить ее в виде структурной схемы.

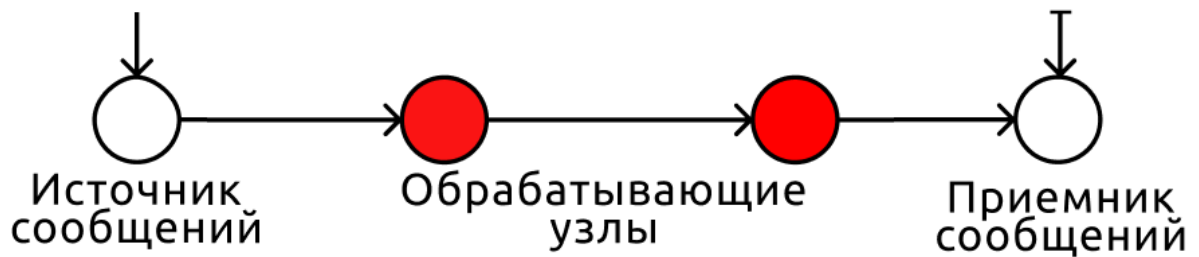


Рисунок 4 - Структурная схема системы, показывающая работу обрабатывающих узлов

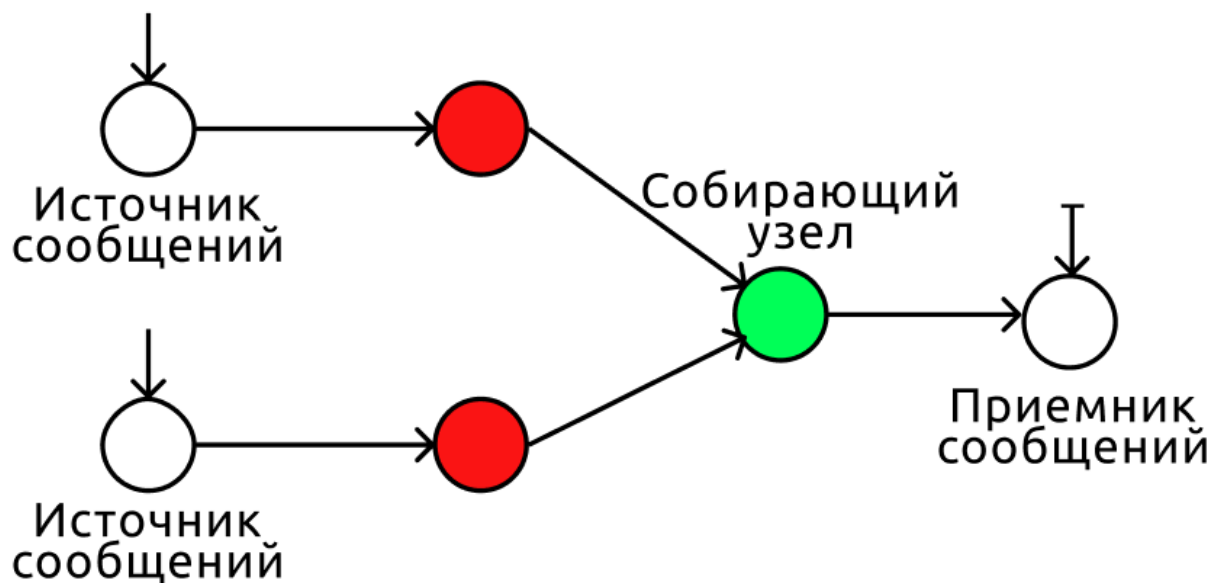


Рисунок 5 - Структурная схема системы, показывающая работу собирающего узла

Теперь покажем, как выглядят такие схемы с помощью среды-разработки Node-RED на рисунках 6 и 7.

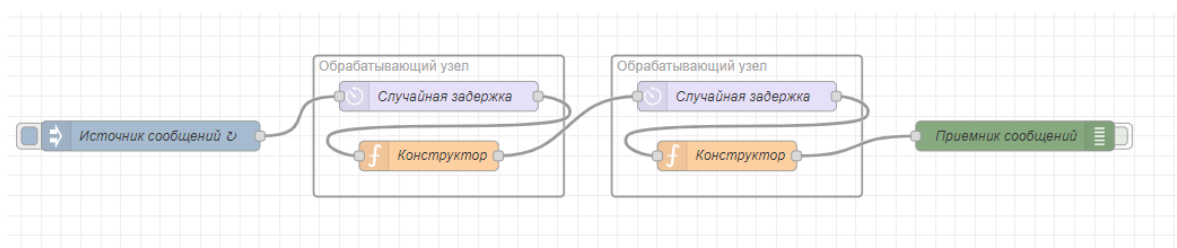


Рисунок 6 - Работа обрабатывающих узлов в среде разработки Node-RED

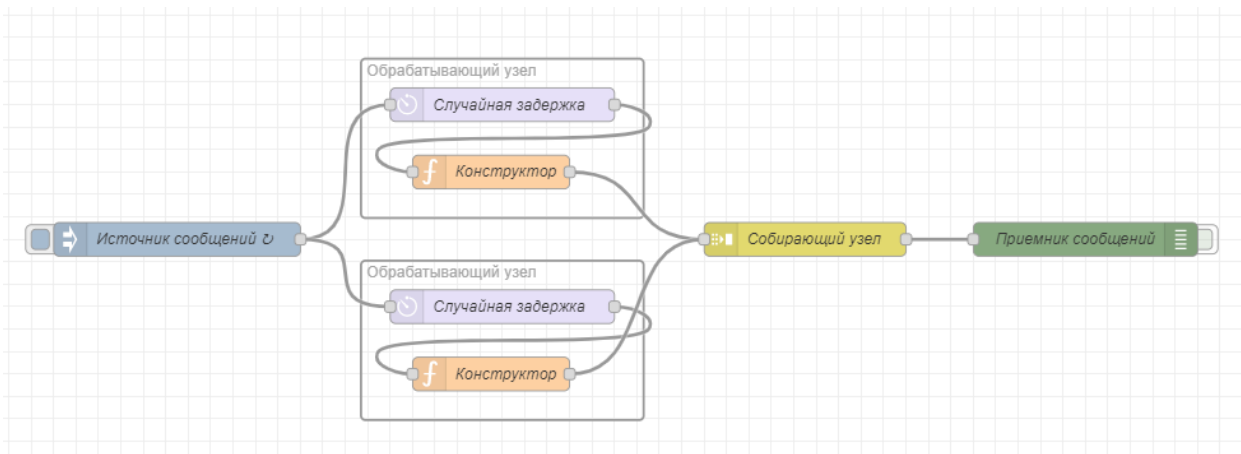


Рисунок 7 - Работа собирающего узла в среде разработки Node-RED

Источник сообщений представляется готовым узлом *Inject*, его можно использовать для ручного запуска потока, нажав кнопку на нем или для автоматического запуска потока, через равные промежутки времени. При этом элемент *Inject* может послать в систему некоторое сообщение имеющие вид контейнера, именуемым объектом из языка программирования JavaScript. Такие объекты формулируются контейнером с двумя основными параметрами - ключ и значение. Именно эти объекты и будут основным предметом в модели, т.е. сообщениями, несущими информацию о предметах, материалах, деталях и т.п. движущихся по конвейерным линиям.

Обрабатывающий узел представляется двумя узлами Node-RED, а именно *Delay* и *Function*. Узел *Delay* предоставляет возможность задерживания сообщений в себе или создания очереди. Время задержки можно настраивать в зависимости от нужд пользователя, например есть возможность задерживать каждое сообщение на определенное время или же настроить случайную задержку от минимальной до максимальной величины. Узел *Function* представляет из себя обычную функцию на языке JavaScript. Внутри функции пользователь имеет возможность написать любой код по своему желанию, но есть одно условие - в функцию в качестве аргументов можно передать лишь объект сообщения.

3 Результаты проведенной работы

3.1 Проведение экспериментов

Необходимо доказать, что выбранный инструмент для моделирования соответствует теории, это нужно, чтобы убедиться в его работе, как инструмента имитационного моделирования задержек. Для доказательства был выбран метод экспериментального исследования. Суть экспериментов состоит в снятии временных данных с сообщений, прошедших через систему. В объект сообщения будет записано время до его обработки и

после в формате миллисекунд, прошедших с 1 января 1970 года 00:00:00 по UTC, а разница между этими величинами и будет составлять задержку.

Во-первых, необходимо понять какой закон распределения плотности вероятности задержки будет у обрабатывающего узла. Для этого необходимо провести эксперимент по снятию статистического ряда. Далее проанализировать статистический ряд с помощью графика гистограммы и по ее форме определить форму плотности вероятности задержки. Для проведения эксперимента воспользуемся потоком, представленном на рисунке 8.

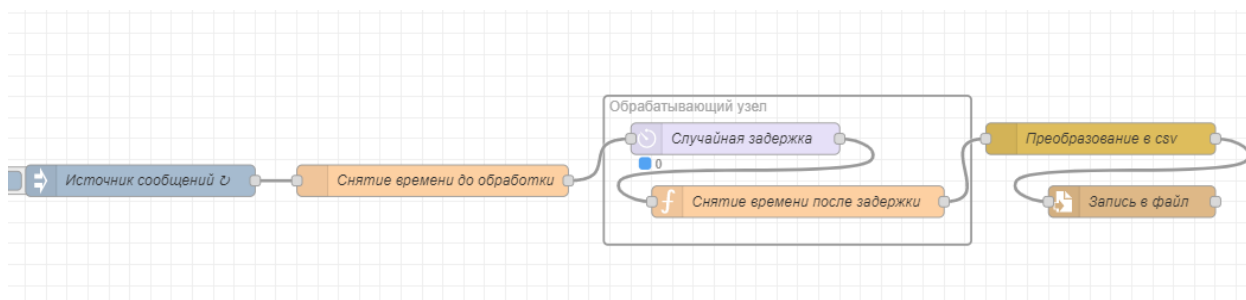


Рисунок 8 - Поток в среде Node-RED для снятия статистического ряда

В данном потоке представлены такие узлы как *CSV* и *Write File* они необходимы для корректной вставки данных в файлы, представляющих табличный вид. После снятия статистического ряда, он был разбит на интервальные ряды и по ним, была построена гистограмма, показывающая распределение задержки по интервальным рядам. Гистограмма представлена на рисунке 9.

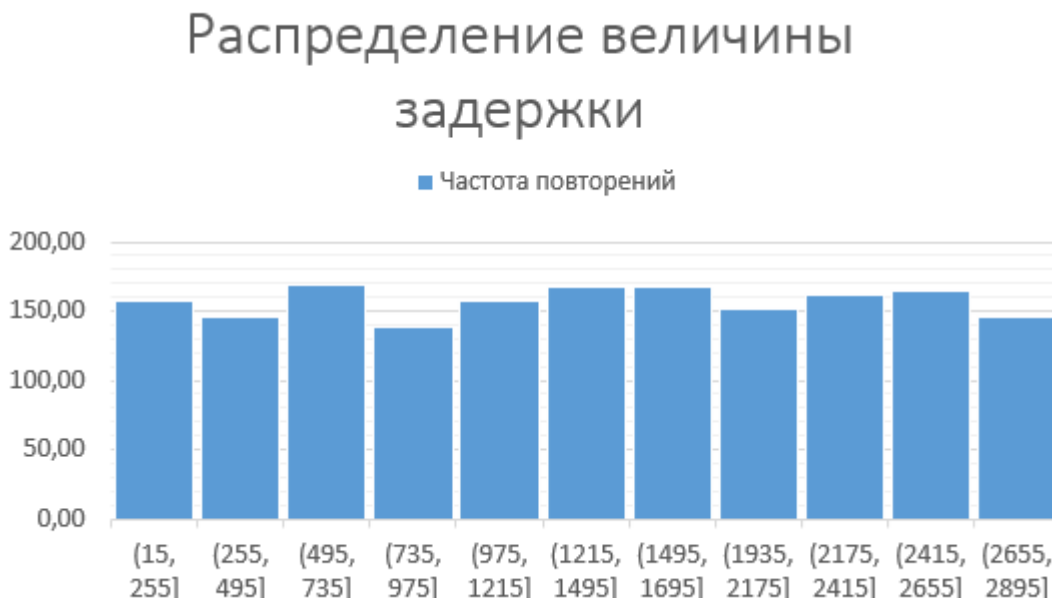


Рисунок 9 - Гистограмма, показывающая распределение задержки по интервальным рядам

Анализируя данную гистограмму, можно сказать, что функция распределения вероятности задержки имеет равновероятностный характер. При моделировании понадобится часто ставить обрабатывающие узлы последовательно друг за другом,

поэтому необходимо убедиться, что суммирование двух и более независимых задержек между собой будет удовлетворять теории, а точнее, что распределение будет относиться к распределению Симпсона. Для этого была построена еще одна простая модель, имеющая два обрабатывающих узла. Модель представлена на рисунке 10.

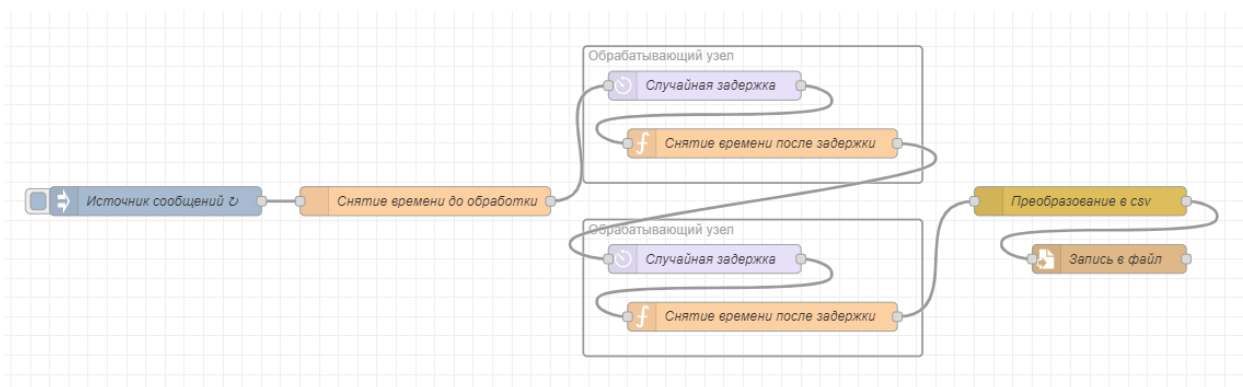


Рисунок 10 - Модель для снятия статистического ряда в случае двух обрабатывающих узлов

По прошлому алгоритму была построена гистограмма, для анализирования ее вида на форму распределения задержки. Гистограмма представлена на рисунке 11.



Рисунок 11 - Гистограмма, показывающая распределение задержки в случае с двумя последовательными обрабатывающими узлами

Анализируя данную гистограмму, можно сказать, что форма распределения задержки имеет форму распределения Симпсона, так называемого треугольного распределения. Для того, чтобы точно удостовериться в этом, можно выписать все количества вхождений величин задержек в один интервальный ряд, далее построить

точечный график, где по оси абсцисс будут также отложены интервальные ряды, а по оси ординат количество вхождений. С помощью программного обеспечения *Origin Pro* выполним аппроксимацию такого графика. График показан на рисунке 12.

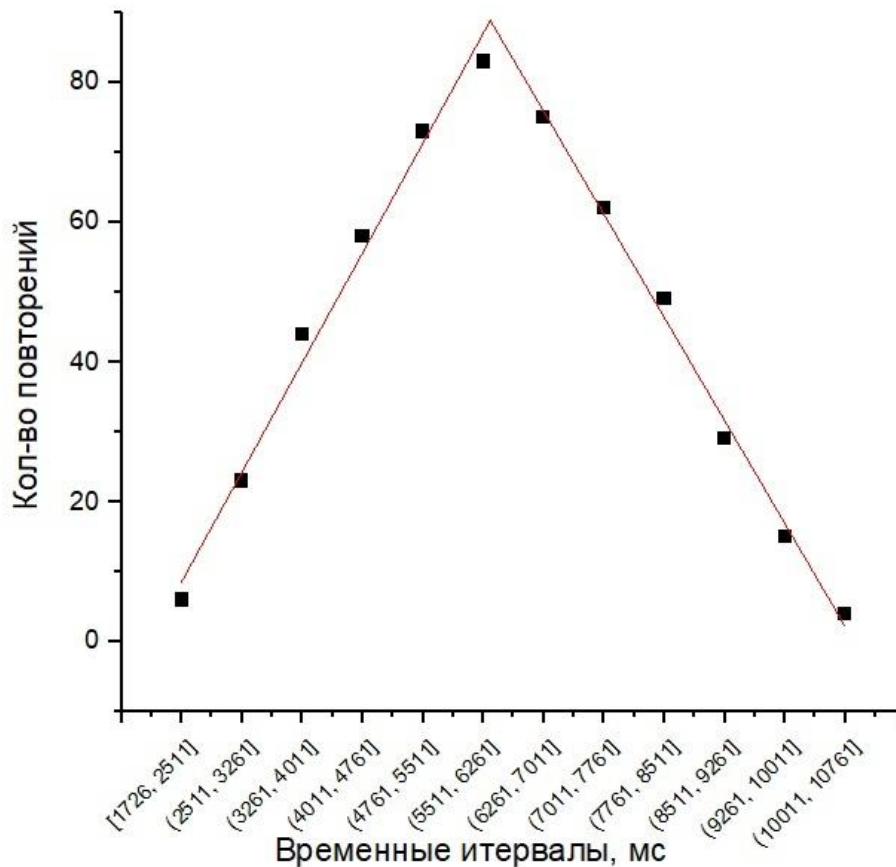


Рисунок 12 - График количества повторений в каждом интервальном ряде

При увеличении подряд следующих узлов обработки, треугольное распределение будет сглаживаться, стремясь к нормальному распределению, в соответствии с центральной предельной теоремой. Теперь можно точно убедиться, что моделирование в среде Node-RED точно соответствует теории.

Осталось рассмотреть, как с точки зрения влияния задержек на систему, ведет себя собирающий узел. Для этого можно выделить два случая. Первый, когда разные источники сообщений можно разбить на быстрые и медленные, с тем условием, что задержки на узлах обслуживания будут одинаковы. Т.е. за одинаковое время t , два источника сообщений сгенерируют разное количество сообщений в систему. Условия для первого случая можно записать так $t_1 < t_2$, где t_1 время возникновения сообщений для первого генератора, а t_2 время возникновения сообщений для второго генератора, который является более медленным. Времена задержки обозначим $\delta_1 = \delta_2$ для первого и второго случая соответственно. Графическое изображение такого примера изображено на рисунке 13.

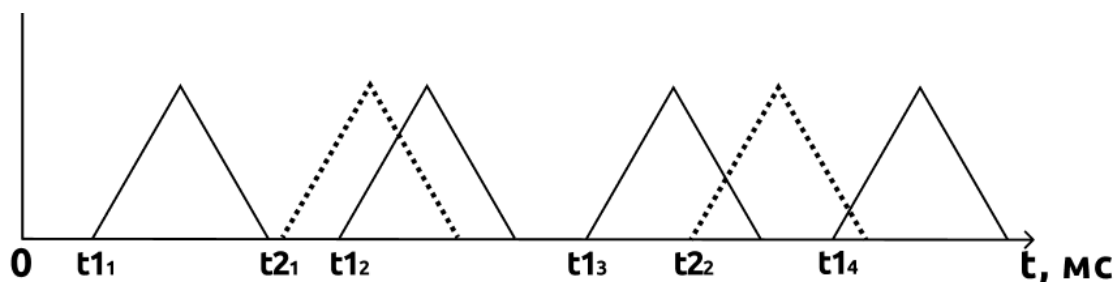


Рисунок 13 - Возникновение сообщений на временной линии в системе для первого случая

Для проверки, какие задержки будут вносить вклад в систему, также проведем эксперимент по снятию статистических рядов. И будем сравнивать разницу во времени между настоящим событием и предшествующим для обоих источников сообщений, а далее отобразим эти данные на гистограммах, заранее разбив величины задержек по интервальным рядам. Модель для эксперимента представлена на рисунке 14.

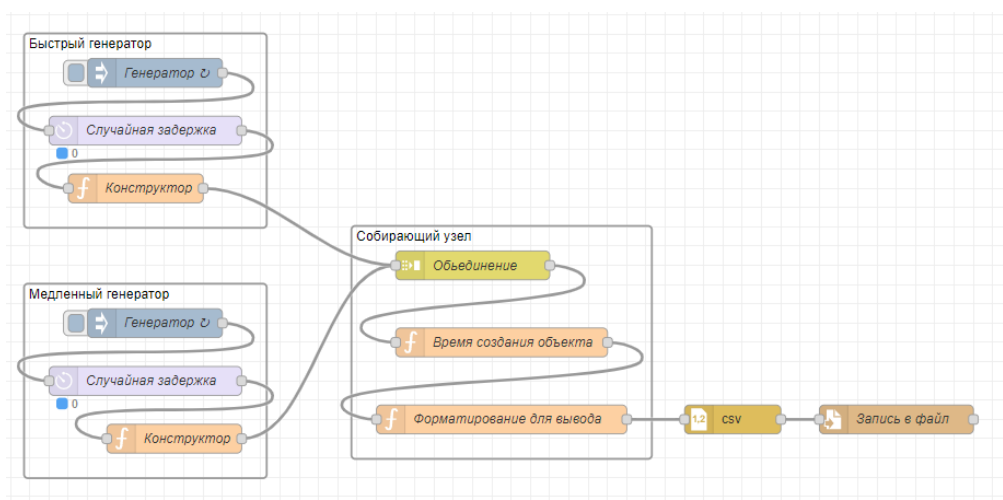


Рисунок 14 - Модель для проведения экспериментов с собирающими узлами

Данные проведения экспериментов показаны на рисунках 15 и 16 для разницы между временем выхода сообщений из системы и более медленного генераторов соответственно.

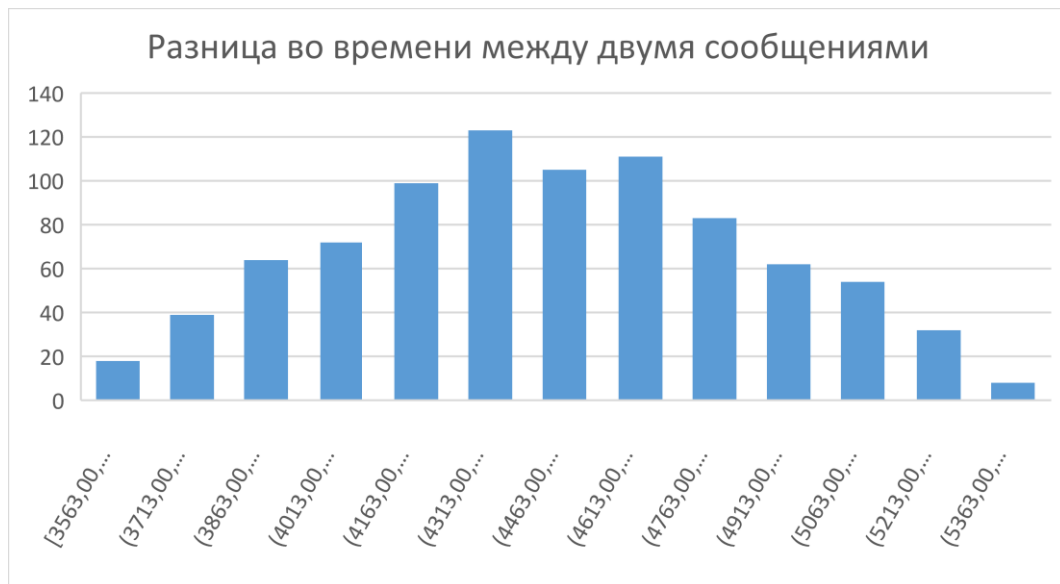


Рисунок 15 - Гистограмма, показывающая разницу между временем выхода сообщений из системы

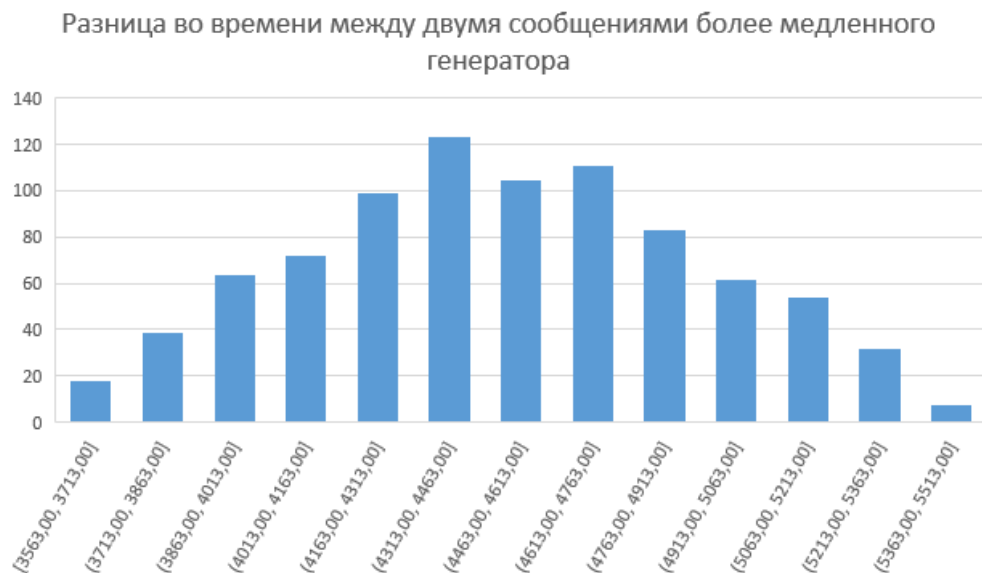


Рисунок 16 - Гистограмма, показывающая разницу во времени между двумя сообщениями с более медленного генератора

В данном случае получается, что две гистограммы одинаковы. Из этого можно сделать вывод, что быстрый генератор вообще не вносит задержку в систему. Это можно объяснить тем, что узел *Join*, который отвечает за объединение, настроен так, чтобы отбрасывать старые сообщения с одного потока, если к узлу подошло более новое сообщение. Такое допущение необходимо в следствие программных ограничений, т.к. на узле объединения может скопиться огромная очередь на операцию объединения и это приведет к переполнению оперативной памяти компьютера. Получается, что при приходе на узел объединения сообщения с более медленного источника, там всегда уже есть второе

сообщение с быстрого источника, но его временные значения задержек будут меньше и из-за этого никакого влияния на систему оказываться не будет.

Теперь рассмотрим второй случай, когда времена задержек на временной линии могут перекрываться в следствие большой их величины и их случайности. Таким образом условия для этого случая будут прописаны так: $t_1 < t_2$, но при этом $\delta_1 > \delta_2$. Графическое изображения временной линии для такого случая изображено на рисунке 17.

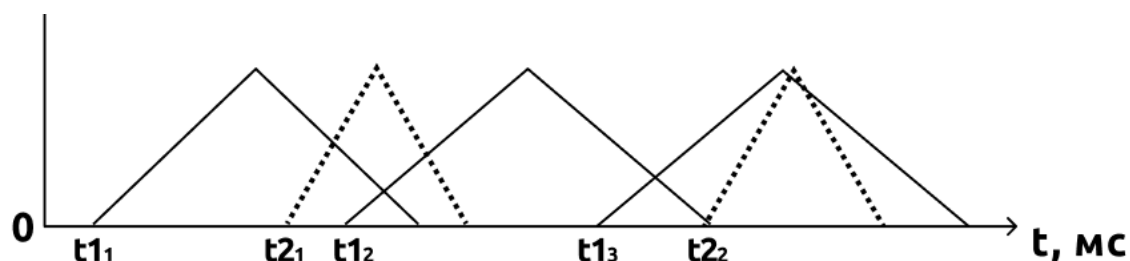


Рисунок 17 - Возникновение сообщений на временной линии для второго случая

Для моделирования были подобраны такие величины $t_1 = 8$ сек., $t_2 = 6$ сек., $\delta_1 = 0-1$ сек., $\delta_2 = 0-4$ сек. Результаты моделирования представлены на рисунках 18-20 для разницы во времени между более медленным источником сообщений быстрым источником и сообщениями на выходе соответственно.

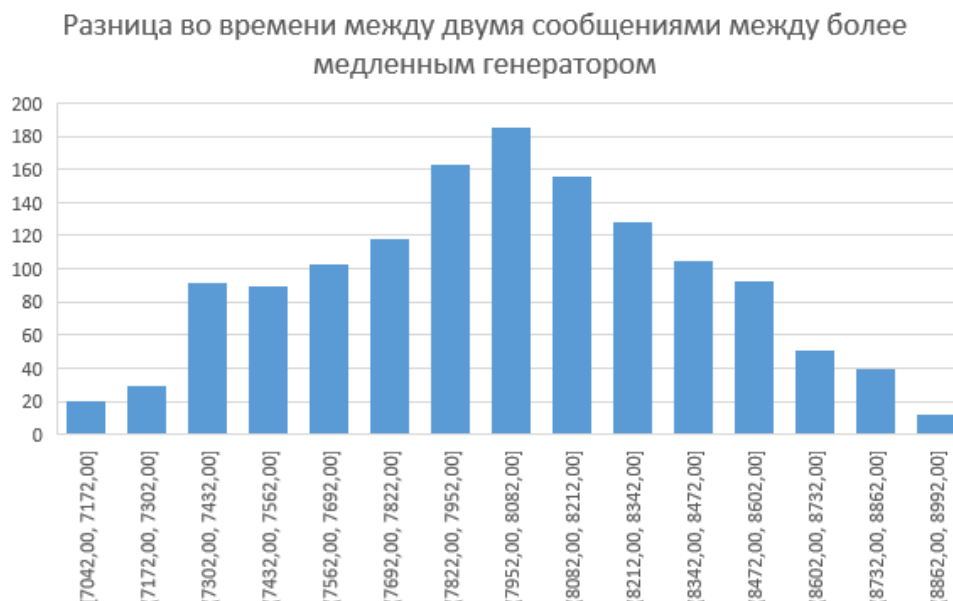


Рисунок 18 - Гистограмма, показывающая разницу между двумя сообщениями у более медленного генератора



Рисунок 19 - Гистограмма, показывающая разницу между двумя сообщениями у более быстрого генератора



Рисунок 20 - Гистограмма, показывающая разницу между двумя сообщениями на выходе системы

Как можно видеть по представленным гистограммам, в этом случае задержка в системе определяется не только благодаря медленному генератору, но и быстрый вносит вклад в нее. Такой результат кроется в перекрытие времен задержек у быстрого генератора. Так как задержка является независимой случайной величиной, то с некоторой вероятностью может возникнуть такая ситуация, что на узел сборки, сообщение от быстрого генератора придет позже, чем сообщение от быстрого генератора.

3.2 Проведение имитационного моделирование

Для проведения имитационного моделирования необходимо иметь структурную схему какого-либо производства. Она нужна для того, чтобы точно знать расположение

всех узлов обработки и сборки, а кроме того, знать время обработки и посылки материалов в систему. По таким данным можно построить модель для симуляции работы производства и задать точные параметры для всех объектов моделирования. Так как практически все предприятия скрывают свои структурные схемы из-за конкуренции, то придется воспользоваться формальной моделью. Формальная модель была создана с помощью компьютерной игры *Factorio*.

Factorio - компьютерная игра в жанре симуляции строительства и управления, изданная чешской игровой студией Wube Software. Основополагающая задача игры состоит в том, чтобы строить заводы, фабрики и т.д., правильно управлять ими, распределять ресурсы и совершенствовать свое производство. Такая игра очень хорошо моделирует изучаемые в данной работе системы. Формальная модель для изучения, построенная с помощью данной игры представлена на рисунке 21.

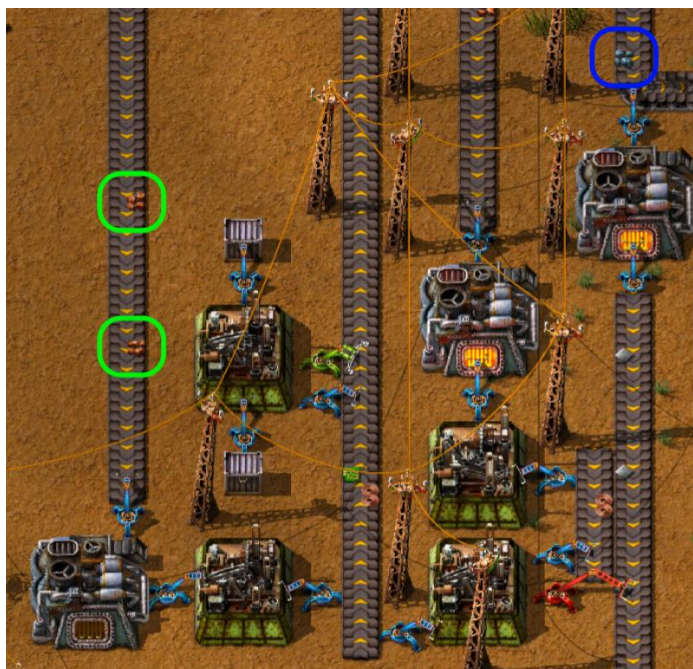


Рисунок 21 - Формальная модель для последующего имитационного моделирования

На представленной формальной модели, можно видеть производство таких игровых объектов как «Красные микросхемы». Для их производства требуются ресурсы двух видов, а именно стальная руда, на рисунке 21 обведена в синий контур, и медная руда, обведена в зелёный контур. Следовательно, изначально будет два источника сообщений в систему. Далее каждым типом руды необходимо произвести преобразования, для этого понадобятся обрабатывающие узлы, каждый из которых будет выполнять свою роль и иметь различное время выполнения. Также для сборки микросхем понадобится несколько собирающих узлов.

Имея логистическую схему производства, следующим шагом для моделирования будет снятие временных характеристик с каждого узла и источников руды. Движок игры запрограммирован так, что каждый объект, выполняющий преобразования над материалами, имел некоторую вероятностную задержку и даже узел добычи руды, с которого она поступает на конвейерную линию также обладает такой характеристикой.

Обозначим $t_{\text{медь}}$ и $t_{\text{сталь}}$ времена для поступления медной и стальной руды в систему соответственно и засечем эти величины. Получились такие значения $t_{\text{медь}} = 0,6-0,8$ сек. и $t_{\text{сталь}} = 1-1,4$ сек. Далее необходимо снять временные характеристики с обрабатывающих узлов: $t_{\text{плавильня меди}} = 1,35 - 1,7$ сек., $t_{\text{плавильня стали}} = 1,5-1,9$ сек., $t_{\text{создание проволоки}} = 0,2-0,3$ сек., где $t_{\text{плавильня меди}}$ - время за которое медная руда перелавливается в медную пластину, $t_{\text{плавильня стали}}$ - время за которое стальная руда перелавливается в стальную пластину, $t_{\text{создание проволоки}}$ - время за которое из медной пластины создается медная проволока. И последним этапом необходимо снять время сборки микросхем: $t_{\text{зеленые микросхемы}} = 1,5-1,8$ сек., $t_{\text{красные микросхемы}} = 7-8$ сек., где $t_{\text{зеленые микросхемы}}$ - время необходимо для сборки зеленных микросхем, которые собираются из стальных пластин и медной проволоки, $t_{\text{красные микросхемы}}$ - время необходимо для сборки красных микросхем, которые собираются из зеленных микросхем и медной проволоки.

Для представления получившейся имитационной модели, она будет разбита на части и показана ниже.

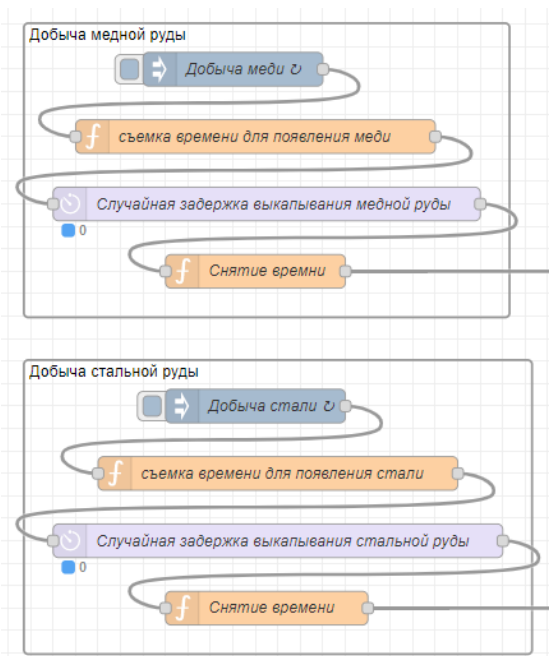


Рисунок 22 - Часть имитационной модели, отвечающая за добычу медной и стальной руды

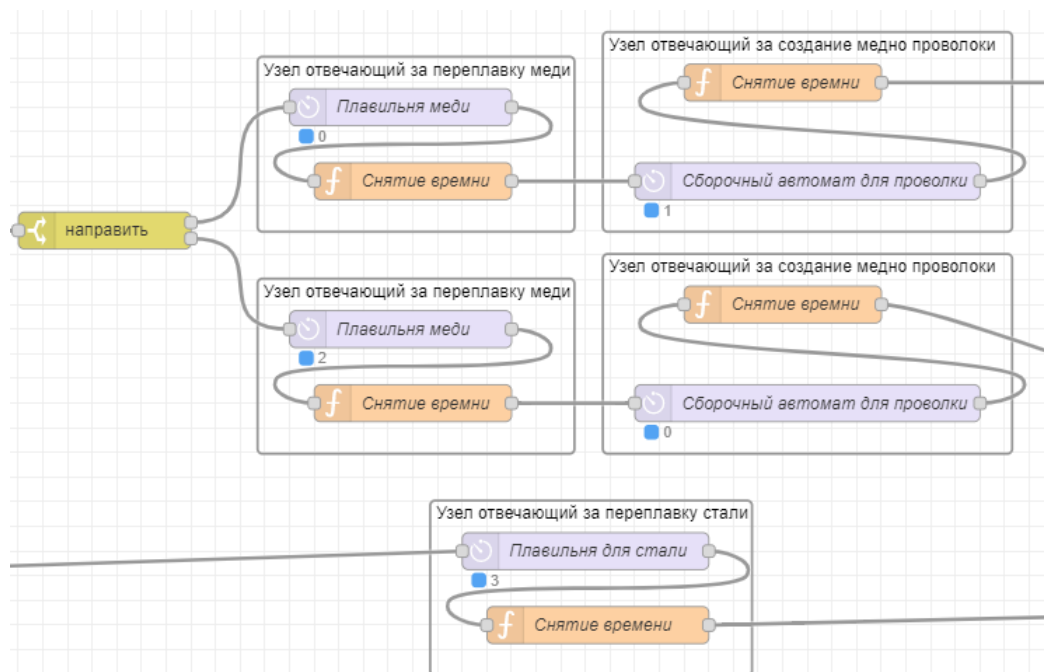


Рисунок 23 - Часть имитационной модели, отвечающая за обработку руды

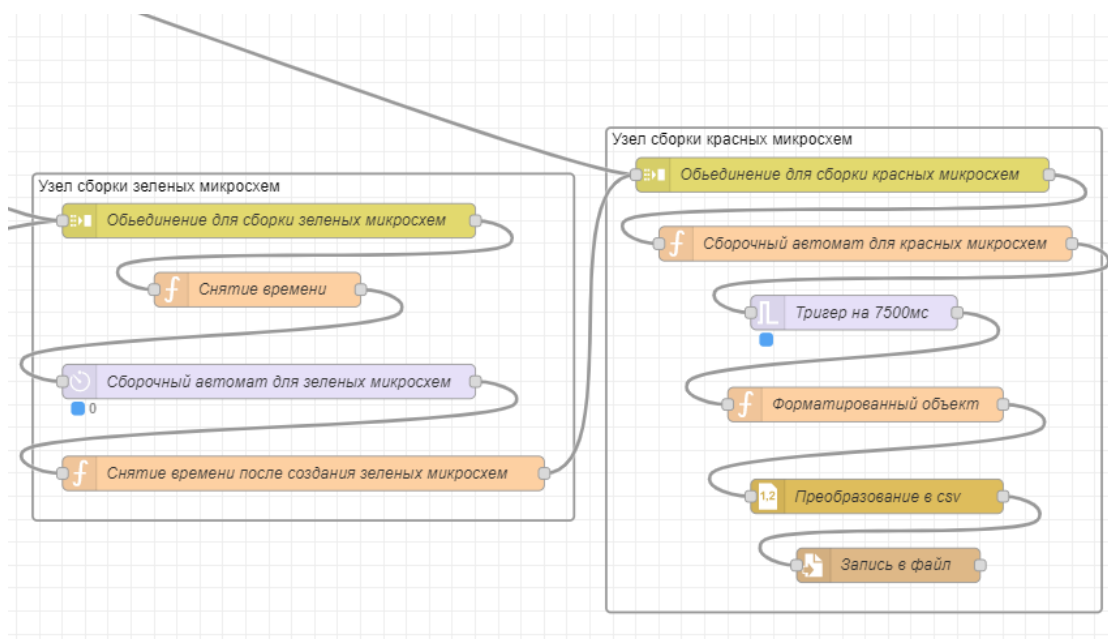


Рисунок 24 - Часть имитационной модели, отвечающая за сборку микросхем

Так была построена имитационная модель в среде Node-RED, для анализа задержек в производстве. Каждое сообщение в системе несет информацию о времени до входа в узел обработки и после него, на узле сборки красных микросхем, все сообщения для создания красной микросхемы объединяются в одно и записываются в файл формата .csv. Проводя моделирование достаточно долго время, можно снять несколько достаточных для анализа статистических рядов.

3.3 Дополнение среды разработки Node-RED

Как уже было сказано выше, среду Node-RED можно дополнить плагинами. Для этого в документации к среде есть подробные объяснения. Можно воспользоваться этой возможностью, чтобы расширить инструмент. Чтобы облегчить взаимодействие с моделями, можно выводить данные в виде различных графиков прямо в веб-интерфейс. Чтобы реализовать такую функциональность можно воспользоваться готовыми библиотеками, написанными на языке JavaScript, для отображения данных.

Важными параметрами для выбора библиотеки должны быть:

- 1) Подходящая функциональность, позволяющая отображать данные в реальном времени;
- 2) Возможность интегрирования в другие программные обеспечения;
- 3) Работа с данными из программного кода, а не только из форматированных файлов формата .csv, .json, .html и так далее;
- 4) Не должны быть дополнительных условий для развертывания такой библиотеки, не считая условия, под которые подходит инструмент Node-RED.

Под данные критерии хорошо подходит библиотека Chart.js. Это простая библиотека для визуализации различных диаграмм и графиков. На данный момент является самой популярной по количеству скачиваний с GitHub и по скачиванию npm пакетов из всех визуальных библиотек для графиков. Из ее преимуществ можно выделить, что для отображения данных, она не использует SVG для каждого объекта, а пользуется инструментом HTML 5, таким как тэг CANVAS, что очень положительно влияет на производительность приложения, например для отображения массива из более чем тысячи элементов, в DOM дерево страницы не будет вставлено столько же узлов SVG.

После интегрирования Chart.js в среду разработки Node-RED, можно в реальном времени выводить данные с помощью созданных по документации узлов. Расширив имитационную модель, показанную выше, с помощью новых узлов, рассмотрим результаты графического представления данных ниже.

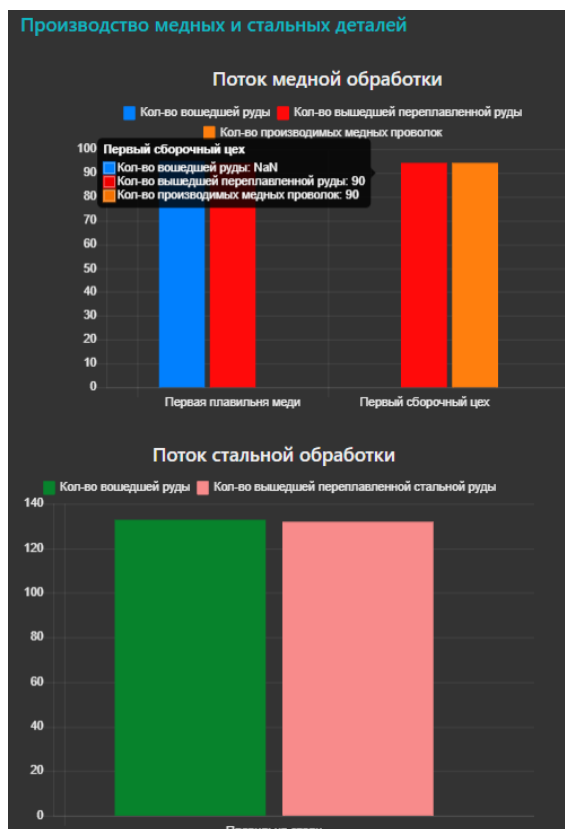


Рисунок 25 - Диаграммы, показывающие количество входящих и выходящих материалов на узлы обработки

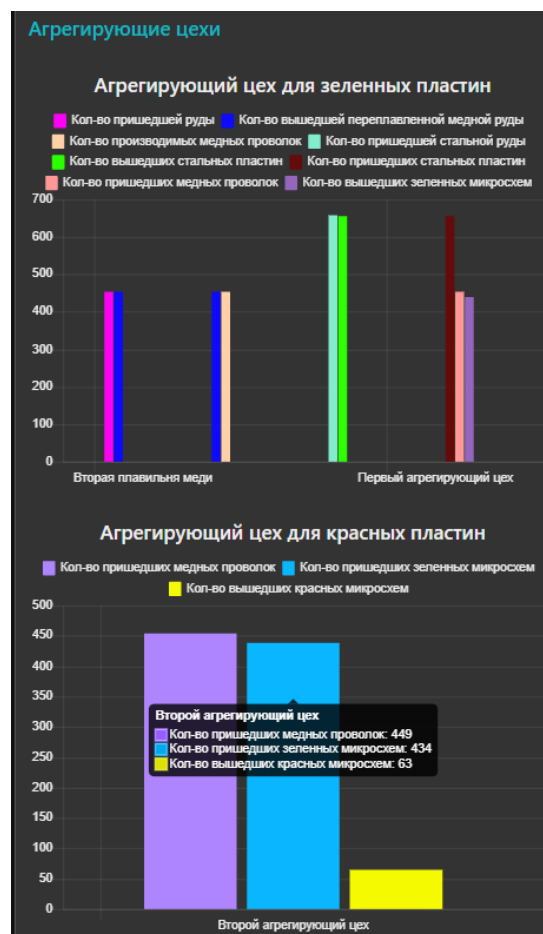


Рисунок 26 - Диаграммы, показывающие количество входящих и выходящих материалов для сборочных узлов

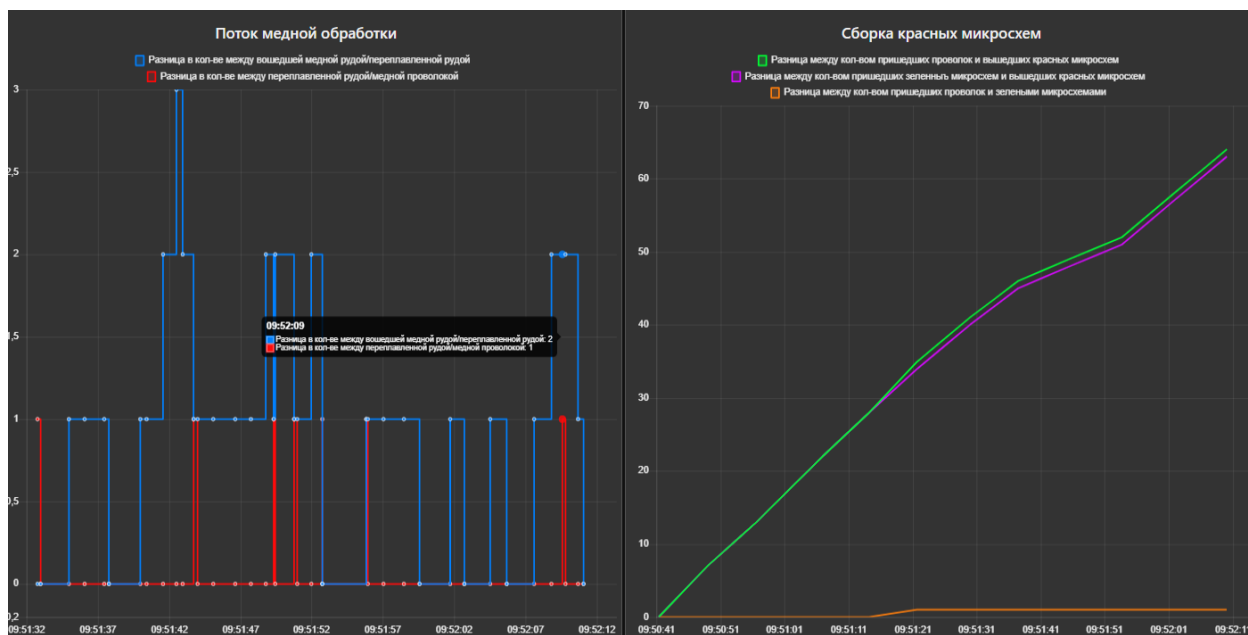


Рисунок 27 - Графики, показывающие разницу между входящими материалами и выводящими деталями от времени

Теперь, когда данные сразу отображаются в реальном времени на графиках, проще анализировать условия, выставленные для моделирования. Как можно видеть по представленным диаграммам и графикам, в системе имеются зависания при сборке зеленных и красных микросхем.

Для случая сборки зеленных микросхем можно видеть, что количество пришедших стальных микросхем много больше, чем медных проволок, т.е. здесь нарушается условие баланса материалов. Хотя сообщения с медной рудой и появляется в системе быстрее, чем сообщения со стальной рудой, но за время движения сообщения по системе, оно проходит больше узлов обработки и при этом поток таких сообщений делиться поровну для отдельного создания медных проволок и тех, кто идет на сборку зеленных микросхем. Для случая сборки красных микросхем видно, что сборочный узел не справляется со своей нагрузкой и поэтому видна огромная разница на правом графике рисунка 25. Точно такие же затыки показывает реальная система, в этом можно убедиться с помощью рисунка 28.

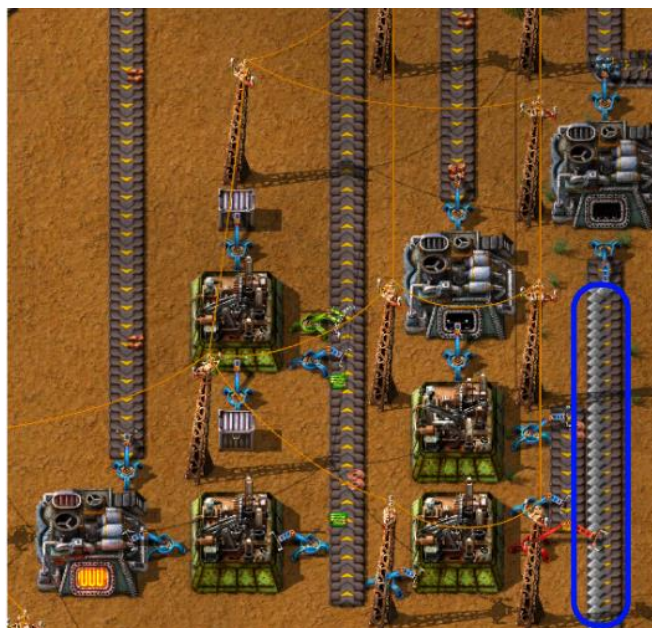


Рисунок 28 - Застревания стальных пластин в формальной модели

Такие узкие места системы необходимо оптимизировать для качественного производства. Варьируя временные параметры имитационной модели и немного перестроив ее структуру, можно достичь оптимальной системы. Чтобы не изменять временные характеристики узлов обработки и сборки, был использован метод баланса потоков материалов и продуктов. С помощью этого метода было вычислено время новое время для источника сообщений стальной руды $t_{\text{сталь}2} = 3-4$ сек. Далее с помощью анализа движения потока, было принято решение расширить схему и дополнить ее узлом сборки зеленых микросхем. Чтобы разобраться с зависаниями на узле сборки красных микросхем, были добавлены производственные мощности в виде еще двух узлов сборки. Дополненная имитационная модель, поделенная на части показана на рисунках ниже.

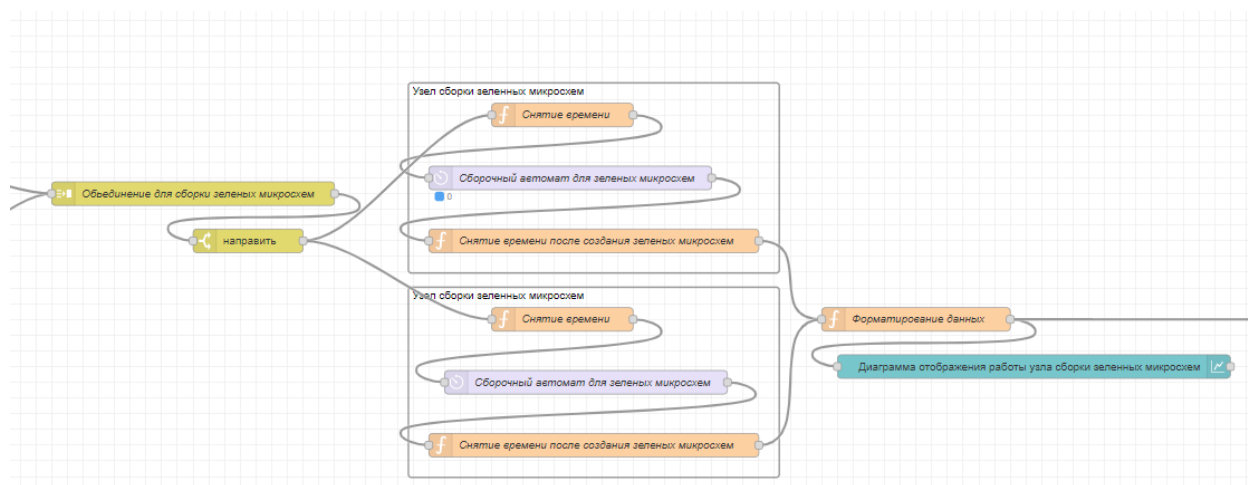


Рисунок 29 - Часть имитационной модели, отвечающей за сборку зеленых микросхем

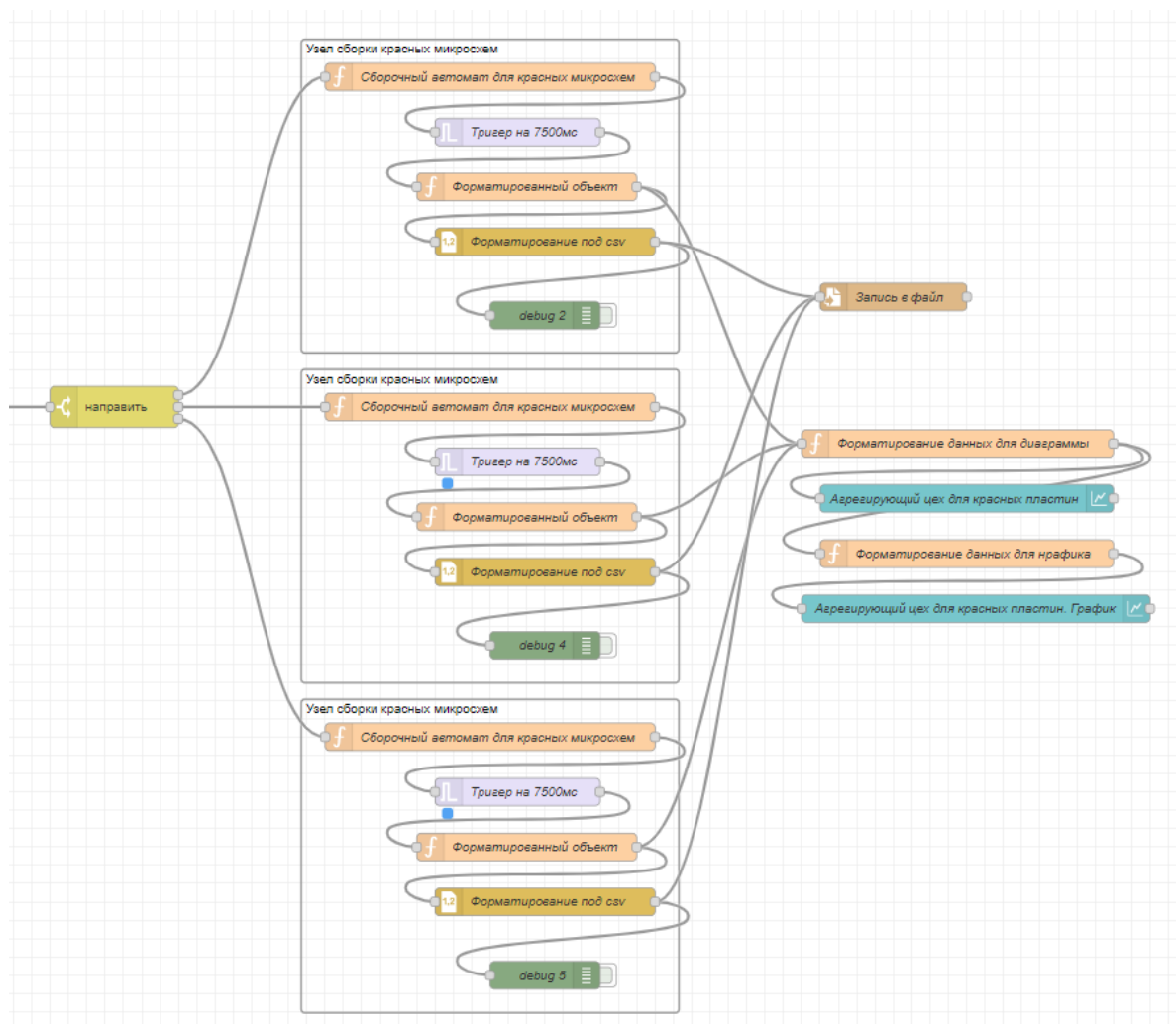


Рисунок 30 - Часть имитационной модели, отвечающей за сборку красных микросхем

Перестроив модель, можно снова вывести данные на графиках и убедиться в ее оптимизации. Графическое представление данных представлено на рисунках 31 и 32.

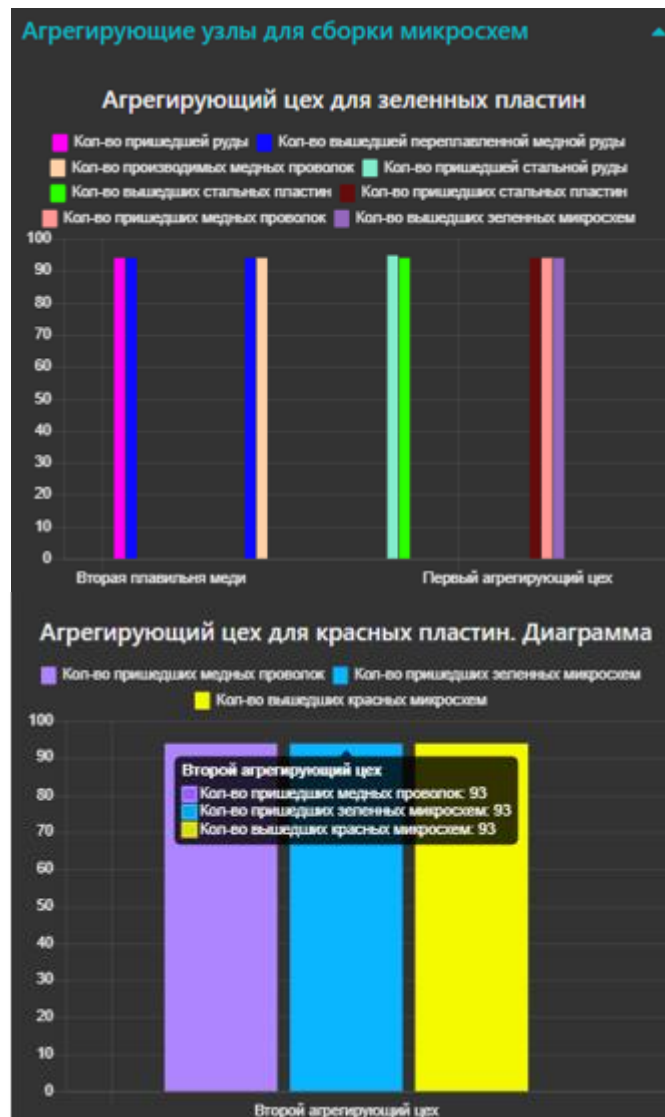


Рисунок 31 - Диаграммы показывающие различие между входящими и выходящими сообщениями на узлах сборки

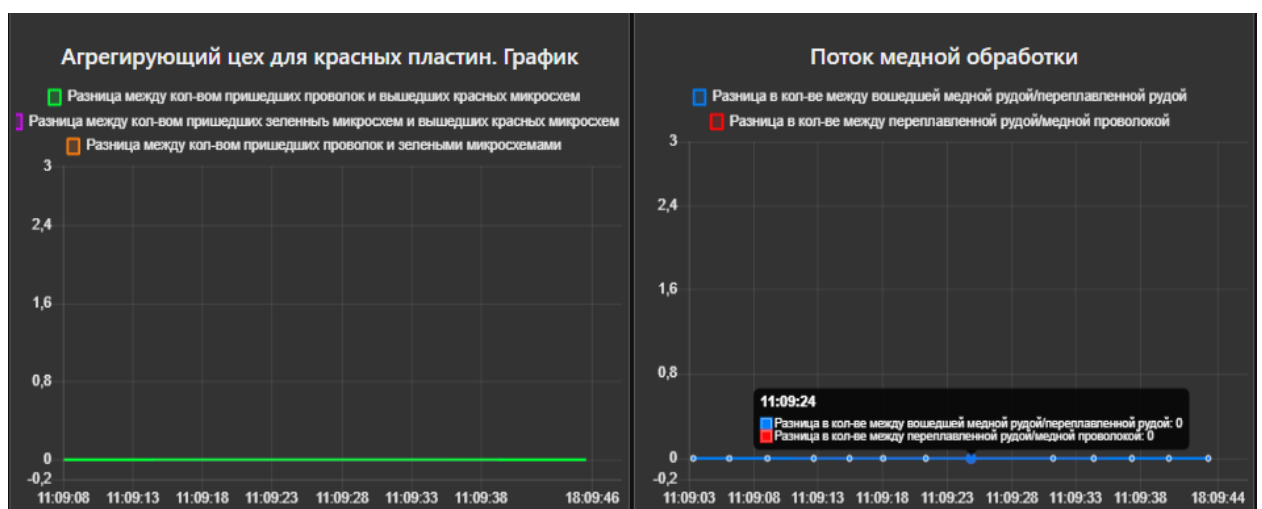


Рисунок 32 - Графики показывающие разницу между входящими и выходящими сообщениями на узлах сборки

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены подходы по анализованию сетей на предмет задержек. Из множества вариантов был выбран практический подход, а именно проведение моделирования систем. Для таких целей были рассмотрены различные инструменты и выбран один из них, а именно среда потокового программирования Node-RED. Было проведено описание моделируемых систем и поставлены условия экспериментов. Далее были созданы имитационные модели для проведения экспериментов, а также для проведения моделирования готовой системы.

Эксперименты доказали, что выбранный инструмент для моделирования Node-RED, удовлетворяет стохастической теории. Кроме того, эксперименты показали, что наибольшее влияние на задержки в системе вносит тот случай, когда с некоторой вероятностью все сообщения приходят на сборочные узлы в различном порядке относительно друг друга.

На примере формальной модели, созданной с помощью игры Factorio, было проведено имитационное моделирование. Также была расширена функциональность среды Node-RED, что позволило без подробного анализа задержек в системе, вычислить ее узкие места и провести оптимизацию для их исправления.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

- 1) Солнышкина, И.В. Теория систем массового обслуживания: учеб. пособие / И. В. Солнышкина. – Комсомольск-на-Амуре: ФГБОУ ВПО «КнАГТУ», 2015. – 76 с.
- 2) Романенко, В. А. Системы и сети массового обслуживания: учебное пособие / В.А. Романенко. – Самара: Издательство Самарского университета, 2021. – 68 с.
- 3) Горцев А. М. Управление и адаптация в системах массового обслуживания / А. М. Горцев, А. А. Назаров, А. Ф. Терпугов; Ред. А. П. Рыжаков. - Томск: Издательство Томского университета, 1978. - 207, [1] с.
- 4) Дакетт Дж. HTML и CSS. Разработка и дизайн веб-сайтов / Дж. Дакетт. – Москва: Эксмо, 2013. – 480 с.
- 5) Марейн Х. Выразительный JavaScript. Современное веб-программирование / Х. Марейн. – 3-е изд. – СПб.: Питер, 2019. – 480 с.
- 6) Современный учебник Javascript [Электронный ресурс]. — URL: <https://learn.javascript.ru/> (дата обращения: 06.09.2022).
- 7) Официальный сайт технологии Node-RED [Электронный ресурс]. – URL: <https://nodered.org/> (дата обращения: 28.09.22).
- 8) Якубов В.П. Статистическая радиофизика: учебное пособие / В. П. Якубов. - Томск: Изд-во Науч.-технической лит., 2006. - 127 с.
- 9) Рогов А.Ю., Графовые методы анализа в дискретной математике: учебное пособие / А.Ю. Рогов, В.И. Халимон, О.В. Проститенко, СПб: СПбГТИ(ТУ), 2012.- 88 с.
- 10) Официальный сайт программного обеспечения Arena [Электронный ресурс]. – URL: <http://www.interface.ru/sysmod/arena.htm> (дата обращения: 05.04.23).
- 11) Официальный сайт библиотеки Chart.js [Электронный ресурс]. – URL: <https://www.chartjs.org/> (дата обращения: 29.04.23).

ПРИЛОЖЕНИЕ А

ИМИТАЦИОННАЯ МОДЕЛЬ В ВИДЕ. JSON ФАЙЛА

```
[
  {
    "id": "780c730ed6e5eaa5",
    "type": "tab",
    "label": "Имитационная модель",
    "disabled": false,
    "info": "",
    "env": []
  },
  {
    "id": "f6834f732ef75fda",
    "type": "inject",
    "z": "780c730ed6e5eaa5",
    "name": "Добыча меди",
    "props": [
      {
        "p": "payload"
      },
      {
        "p": "topic",
        "vt": "str"
      }
    ],
    "repeat": "0.6",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "topic": "cuprum",
    "payload": "",
    "payloadType": "date",
    "x": 100,
    "y": 360,
    "wires": [
      [
        "e6e4282403d3a8fb"
      ]
    ]
  },
  {
    "id": "752c9017cd214b31",
    "type": "inject",
    "z": "780c730ed6e5eaa5",
    "name": "Добыча стали",
    "props": [
      {
        "p": "payload"
      },
      {
        "p": "topic",
        "vt": "str"
      }
    ],
    "repeat": "0.8",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "topic": "ferum",
    "payload": "",
    "payloadType": "date",
    "x": 100,
    "y": 580,
    "wires": [
      [
        "090fb519d8993496"
      ]
    ]
  },
  {
    "id": "6d0163fa6a367383",
    "type": "delay",
    "z": "780c730ed6e5eaa5",
    "name": "Случайная задержка выкапывания стальной руды",
  }
]
```

```

    "pauseType": "random",
    "timeout": "5",
    "timeoutUnits": "seconds",
    "rate": "1",
    "nbRateUnits": "1",
    "rateUnits": "second",
    "randomFirst": "0",
    "randomLast": "1",
    "randomUnits": "seconds",
    "drop": false,
    "allowrate": false,
    "outputs": 1,
    "x": 740,
    "y": 580,
    "wires": [
      [
        "dffbae60919031f2"
      ]
    ]
  },
  {
    "id": "9bfcc519407fef6a",
    "type": "delay",
    "z": "780c730ed6e5eaa5",
    "name": "Случайная задержка выкапывания медной руды",
    "pauseType": "random",
    "timeout": "5",
    "timeoutUnits": "seconds",
    "rate": "1",
    "nbRateUnits": "1",
    "rateUnits": "second",
    "randomFirst": "0",
    "randomLast": "1",
    "randomUnits": "seconds",
    "drop": false,
    "allowrate": false,
    "outputs": 1,
    "x": 750,
    "y": 360,
    "wires": [
      [
        "9d45fe5d0dbe9d5e"
      ]
    ]
  },
  {
    "id": "3c0e3d4751bd857c",
    "type": "delay",
    "z": "780c730ed6e5eaa5",
    "name": "Плавильня для стали",
    "pauseType": "random",
    "timeout": "5",
    "timeoutUnits": "seconds",
    "rate": "1",
    "nbRateUnits": "1",
    "rateUnits": "second",
    "randomFirst": "1.2",
    "randomLast": "1.5",
    "randomUnits": "seconds",
    "drop": false,
    "allowrate": false,
    "outputs": 1,
    "x": 1300,
    "y": 580,
    "wires": [
      [
        "3d17933c61a8031c"
      ]
    ]
  },
  {
    "id": "8145f104bfe6483b",
    "type": "switch",
    "z": "780c730ed6e5eaa5",
    "name": "",
    "property": "flagCuprum",
    "propertyType": "global",
    "rules": [
      {

```

```

        "t": "true"
    },
    {
        "t": "false"
    }
],
"checkall": "true",
"repair": false,
"outputs": 2,
"x": 1270,
"y": 360,
"wires": [
    [
        "29405a74e0677693",
        "71341043e19bcb86"
    ],
    [
        "4c27a0527d73745d",
        "8de5f16c54b0b0f0"
    ]
]
},
{
    "id": "e6e4282403d3a8fb",
    "type": "function",
    "z": "780c730ed6e5eaa5",
    "name": "съемка времени для появления меди",
    "func": "msg.payload = {};\nmsg.payload['cuprumBegin'] = Date.now();\n\nswitch(global.get('flagCuprum')){\n  case true:\n    global.set('flagCuprum', false);\n    break;\n  case false:\n    global.set('flagCuprum', true);\n    break;\n  default:\n    break;\n}\n\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 360,
    "y": 360,
    "wires": [
        [
            "9bfcc519407fef6a"
        ]
    ]
},
{
    "id": "20d2418cbca9e7da",
    "type": "change",
    "z": "780c730ed6e5eaa5",
    "name": "Установка глобальных переменных",
    "rules": [
        {
            "t": "set",
            "p": "flagCuprum",
            "pt": "global",
            "to": "true",
            "tot": "bool"
        },
        {
            "t": "set",
            "p": "flagRed",
            "pt": "global",
            "to": "true",
            "tot": "bool"
        },
        {
            "t": "set",
            "p": "initTime",
            "pt": "global",
            "to": "Date.now()",
            "tot": "jsonata"
        },
        {
            "t": "set",
            "p": "cumpurBeforeFirstPlav",
            "pt": "global",
            "to": "",
            "tot": "str"
        }
    ],
    "action": "",

```

```

    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 390,
    "y": 880,
    "wires": [
      []
    ]
  },
  {
    "id": "d61f1a9198b2c44c",
    "type": "inject",
    "z": "780c730ed6e5eaa5",
    "name": "",
    "props": [
      {
        "p": "payload"
      },
      {
        "p": "topic",
        "vt": "str"
      }
    ],
    "repeat": "",
    "crontab": "",
    "once": true,
    "onceDelay": 0.1,
    "topic": "",
    "payload": "",
    "payloadType": "date",
    "x": 120,
    "y": 880,
    "wires": [
      [
        "20d2418cbca9e7da"
      ]
    ]
  },
  {
    "id": "4c27a0527d73745d",
    "type": "delay",
    "z": "780c730ed6e5eaa5",
    "name": "Плавильня меди",
    "pauseType": "random",
    "timeout": "5",
    "timeoutUnits": "seconds",
    "rate": "1",
    "nbRateUnits": "1",
    "rateUnits": "second",
    "randomFirst": "1.2",
    "randomLast": "1.4",
    "randomUnits": "seconds",
    "drop": false,
    "allowrate": false,
    "outputs": 1,
    "x": 1470,
    "y": 440,
    "wires": [
      [
        "4b37934fb4de9c6c"
      ]
    ]
  },
  {
    "id": "090fb519d8993496",
    "type": "function",
    "z": "780c730ed6e5eaa5",
    "name": "съемка времени для появления стали",
    "func": "msg.payload = { };\nmsg.payload['ferumBegin'] = Date.now();\n\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 360,
    "y": 580,
    "wires": [
      [

```

```

        "6d0163fa6a367383"
    ]
}
},
{
    "id": "71341043e19bcb86",
    "type": "delay",
    "z": "780c730ed6e5eaa5",
    "name": "Плавильня меди",
    "pauseType": "random",
    "timeout": "5",
    "timeoutUnits": "seconds",
    "rate": "1",
    "nbRateUnits": "1",
    "rateUnits": "second",
    "randomFirst": "1.2",
    "randomLast": "1.4",
    "randomUnits": "seconds",
    "drop": false,
    "allowrate": false,
    "outputs": 1,
    "x": 1490,
    "y": 240,
    "wires": [
        [
            "18afd7736bdd3e98"
        ]
    ]
},
{
    "id": "381cdef1ffadf32c",
    "type": "delay",
    "z": "780c730ed6e5eaa5",
    "name": "Сборочный автомат для проволоки",
    "pauseType": "random",
    "timeout": "5",
    "timeoutUnits": "seconds",
    "rate": "1",
    "nbRateUnits": "1",
    "rateUnits": "second",
    "randomFirst": "0.2",
    "randomLast": "0.35",
    "randomUnits": "seconds",
    "drop": false,
    "allowrate": false,
    "outputs": 1,
    "x": 1980,
    "y": 440,
    "wires": [
        [
            "433895ed84988bf7"
        ]
    ]
},
{
    "id": "e1f62959b6421f90",
    "type": "delay",
    "z": "780c730ed6e5eaa5",
    "name": "Сборочный автомат для проволоки",
    "pauseType": "random",
    "timeout": "5",
    "timeoutUnits": "seconds",
    "rate": "1",
    "nbRateUnits": "1",
    "rateUnits": "second",
    "randomFirst": "0.2",
    "randomLast": "0.35",
    "randomUnits": "seconds",
    "drop": false,
    "allowrate": false,
    "outputs": 1,
    "x": 2160,
    "y": 240,
    "wires": [
        [
            "6e8b2ae2d1f322bf"
        ]
    ]
},
},

```

```

{
  "id": "18120c2d6e2c2dfe",
  "type": "delay",
  "z": "780c730ed6e5eaa5",
  "name": "Сборочный автомат для зеленых микросхем",
  "pauseType": "random",
  "timeout": "5",
  "timeoutUnits": "seconds",
  "rate": "1",
  "nbRateUnits": "1",
  "rateUnits": "second",
  "randomFirst": "0.15",
  "randomLast": "0.3",
  "randomUnits": "seconds",
  "drop": false,
  "allowrate": false,
  "outputs": 1,
  "x": 2520,
  "y": 580,
  "wires": [
    [
      "dc7151a135219ce7"
    ]
  ]
},
{
  "id": "b2bef7d662065e87",
  "type": "join",
  "z": "780c730ed6e5eaa5",
  "name": "Объединение для сборки зеленых микросхем",
  "mode": "custom",
  "build": "object",
  "property": "payload",
  "propertyType": "msg",
  "key": "topic",
  "joiner": "\\n",
  "joinerType": "str",
  "accumulate": false,
  "timeout": "0",
  "count": "2",
  "reduceRight": false,
  "reduceExp": "",
  "reduceInit": "",
  "reduceInitType": "num",
  "reduceFixup": "",
  "x": 1880,
  "y": 580,
  "wires": [
    [
      "c77e9843c4205b9f"
    ]
  ]
},
{
  "id": "dc7151a135219ce7",
  "type": "join",
  "z": "780c730ed6e5eaa5",
  "name": "Объединение для сборки красных микросхем",
  "mode": "custom",
  "build": "object",
  "property": "payload",
  "propertyType": "msg",
  "key": "topic",
  "joiner": "\\n",
  "joinerType": "str",
  "accumulate": false,
  "timeout": "0",
  "count": "2",
  "reduceRight": false,
  "reduceExp": "",
  "reduceInit": "",
  "reduceInitType": "num",
  "reduceFixup": "",
  "x": 2940,
  "y": 420,
  "wires": [
    [
      "9b1b01bb4ed02f9d"
    ]
  ]
}

```



```

    ]
  },
  {
    "id": "31e4b94c87b1cfdc",
    "type": "function",
    "z": "780c730ed6e5eaa5",
    "name": "Форматированный объект",
    "func": "const obj = { };\n\nobj.payload = {\n  'cuprum': msg.payload['cuprum']['cuprumBegin'],\n  'ferum':\nmsg.payload['green']['ferum']['ferumBegin'],\n  'green': msg.payload.green.green,\n  'end': Date.now(),\n  'cuprumInSystem':\nmsg.payload['cuprum']['cuprumInSystem'],\n  'cuprumFire': msg.payload['cuprum']['cuprumFire'],\n  'cuprumLink':\nmsg.payload['cuprum']['cuprumLink'],\n  'ferumInSystem': msg.payload['green']['ferum']['ferumInSystem'],\n  'ferumFire':\nmsg.payload['green']['ferum']['ferumFire'],\n  'cuprumFireInFerumSystem': msg.payload['green']['cuprum']['cuprumFireInFerumSystem'],\n  'cuprumLinkInFerumSystem': msg.payload['green']['cuprum']['cuprumLinkInFerumSystem'],\n}\n\nreturn obj;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 2960,
    "y": 700,
    "wires": [
      [
        "3cad786cd3998637",
        "5bf515184366a473"
      ]
    ]
  }
],
{
  {
    "id": "3cad786cd3998637",
    "type": "csv",
    "z": "780c730ed6e5eaa5",
    "name": "Форматирование под csv",
    "sep": ",",
    "hdrin": "",
    "hdrout": "once",
    "multi": "one",
    "ret": "\\r\\n",
    "temp": "cuprum, ferum, green, end, cuprumInSystem, cuprumFire, cuprumLink, ferumInSystem, ferumFire, cuprumFireInFerumSystem,\ncuprumLinkInFerumSystem",
    "skip": "0",
    "strings": true,
    "include_empty_strings": "",
    "include_null_values": "",
    "x": 2960,
    "y": 780,
    "wires": [
      [
        "6999262651c3f502"
      ]
    ]
  }
],
{
  {
    "id": "6999262651c3f502",
    "type": "debug",
    "z": "780c730ed6e5eaa5",
    "name": "debug 1",
    "active": false,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 2960,
    "y": 860,
    "wires": []
  }
},
{
  {
    "id": "deacc956f297d372",
    "type": "file",
    "z": "780c730ed6e5eaa5",
    "name": "Запись в файл",
    "filename": "C:\\Scince\\NotOptimizationModel\\model.csv",
    "filenameType": "str",
    "appendNewline": false,
    "createDir": true,
    "overwriteFile": "false",
    "encoding": "setbymsg",

```

```

    "x": 3260,
    "y": 780,
    "wires": [
      []
    ]
  },
  {
    "id": "c77e9843c4205b9f",
    "type": "function",
    "z": "780c730ed6e5eaa5",
    "name": "Снятие времени",
    "func": "global.set(\n  \"green\", \n  global.get(\"green\") + 1 \n); \n \n msg.topic = 'green'; \n msg.payload.green = Date.now(); \n \n return msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "// Добавленный здесь код будет исполняться \n// однократно при развертывании узла. \n global.set(\n  \"green\", \n  0 \n);",
    "finalize": "",
    "libs": [],
    "x": 2190,
    "y": 580,
    "wires": [
      [
        "18120c2d6e2c2dfe",
        "de1e31c80178bb1b"
      ]
    ]
  },
  {
    "id": "9d45fe5d0dbe9d5e",
    "type": "function",
    "z": "780c730ed6e5eaa5",
    "name": "Снятие времени",
    "func": "msg.payload.cuprumInSystem = Date.now(); \n global.set(\"flagCuprum\", !global.get(\"flagCuprum\")); \n \n return msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 1080,
    "y": 360,
    "wires": [
      [
        "8145f104bfe6483b"
      ]
    ]
  },
  {
    "id": "18afd7736bdd3e98",
    "type": "function",
    "z": "780c730ed6e5eaa5",
    "name": "Снятие времени",
    "func": "global.set(\n  \"cuprumAfterPlav1\", \n  global.get(\"cuprumAfterPlav1\") + 1 \n); \n \n msg.payload.cuprumFire = Date.now(); \n \n return msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "// Добавленный здесь код будет исполняться \n// однократно при развертывании узла. \n global.set(\"cuprumAfterPlav1\", 0);",
    "finalize": "",
    "libs": [],
    "x": 1740,
    "y": 240,
    "wires": [
      [
        "e1f62959b6421f90"
      ]
    ]
  },
  {
    "id": "6e8b2ae2d1f322bf",
    "type": "function",
    "z": "780c730ed6e5eaa5",
    "name": "Снятие времени",
    "func": "msg.payload.cuprumLink = Date.now(); \n \n return msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],

```

```

"x": 2480,
"y": 240,
"wires": [
  [
    "dc7151a135219ce7",
    "d035a7ad4c034c58",
    "dbf9e1615096a050"
  ]
]
},
{
  "id": "3d17933c61a8031c",
  "type": "function",
  "z": "780c730ed6e5eaa5",
  "name": "Снятие времени",
  "func": "global.set(\n  \"ferrumPlav\", \n  global.get(\"ferrumPlav\") + 1\n);\n\nmsg.payload.ferrumFire = Date.now();\n\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "initialize": "// Добавленный здесь код будет исполняться\n// однократно при развертывании узла.\nglobal.set(\"ferrumPlav\", 0);",
  "finalize": "",
  "libs": [],
  "x": 1530,
  "y": 580,
  "wires": [
    [
      "b2bef7d662065e87",
      "6b616bd9d90f1219"
    ]
  ]
},
{
  "id": "dffb6e60919031f2",
  "type": "function",
  "z": "780c730ed6e5eaa5",
  "name": "Снятие времени",
  "func": "global.set(\n  \"ferrum\", \n  global.get(\"ferrum\") + 1\n);\n\nmsg.payload.ferrumInSystem = Date.now();\n\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "initialize": "// Добавленный здесь код будет исполняться\n// однократно при развертывании узла.\nglobal.set(\"ferrum\", 0);\n",
  "finalize": "",
  "libs": [],
  "x": 1070,
  "y": 580,
  "wires": [
    [
      "3c0e3d4751bd857c"
    ]
  ]
},
{
  "id": "4b37934fb4de9c6c",
  "type": "function",
  "z": "780c730ed6e5eaa5",
  "name": "Снятие времени",
  "func": "global.set(\n  \"cuprumAfterPlav2\", \n  global.get(\"cuprumAfterPlav2\") + 1\n);\n\nmsg.payload.cuprumFireInFerumSystem = Date.now();\n\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "initialize": "// Добавленный здесь код будет исполняться\n// однократно при развертывании узла.\nglobal.set(\n  \"cuprumAfterPlav2\", \n  0\n);\n",
  "finalize": "",
  "libs": [],
  "x": 1700,
  "y": 440,
  "wires": [
    [
      "381cdef1ffadf32c"
    ]
  ]
},
{
  "id": "433895ed84988bf7",
  "type": "function",
  "z": "780c730ed6e5eaa5",
  "name": "Снятие времени",
  "func": "global.set(\"cuprumWire2\", global.get(\"cuprumWire2\") + 1);\n\nmsg.payload.cuprumLinkInFerumSystem = Date.now();\n\nreturn msg;",
  "outputs": 1,
  "noerr": 0,

```

```

"initialize": "// Добавленный здесь код будет исполняться\n// однократно при развертывании узла.\nglobal.set(\\"cuprumWire2\\", 0);",
"finalize": "",
"libs": [],
"x": 2240,
"y": 440,
"wires": [
  [
    "b2bef7d662065e87"
  ]
]
},
{
  "id": "9b1b01bb4ed02f9d",
  "type": "function",
  "z": "780c730ed6e5eaa5",
  "name": "Сборочный автомат для красных микросхем",
  "func": "\n\nconst min = 7000;\nconst max = 7500;\n\n// setTimeout() => {\n\n  node.send(msg);\n\n  node.done()\n\n },\ndelay);\n\nmsg.delay = Math.floor(min + Math.random() * (max - min));\n\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "initialize": "// Добавленный здесь код будет исполняться\n// однократно при развертывании узла.\nglobal.set(\n  \\"red\\",\n  0\n);",
  "finalize": "",
  "libs": [],
  "x": 2960,
  "y": 500,
  "wires": [
    [
      "cfb64abc8ef46227"
    ]
  ]
},
{
  "id": "cfb64abc8ef46227",
  "type": "trigger",
  "z": "780c730ed6e5eaa5",
  "name": "Тригер на 7500мс",
  "op1": "",
  "op2": "",
  "op1type": "pay",
  "op2type": "nul",
  "duration": "250",
  "extend": false,
  "overrideDelay": true,
  "units": "ms",
  "reset": "",
  "bytopic": "all",
  "topic": "topic",
  "outputs": 1,
  "x": 2950,
  "y": 600,
  "wires": [
    [
      "31e4b94c87b1cfdc"
    ]
  ]
},
{
  "id": "29405a74e0677693",
  "type": "function",
  "z": "780c730ed6e5eaa5",
  "name": "function 1",
  "func": "global.set(\n  \\"cuprumBeforePlav1\\",\n  global.get( \\"cuprumBeforePlav1\\") + 1\n );\n",
  "outputs": 1,
  "noerr": 0,
  "initialize": "// Добавленный здесь код будет исполняться\n// однократно при развертывании узла.\nglobal.set(\\"cuprumBeforePlav1\\", 0);",
  "finalize": "",
  "libs": [],
  "x": 1440,
  "y": 140,
  "wires": [
    []
  ]
},
{
  "id": "d035a7ad4c034c58",
  "type": "function",
  "z": "780c730ed6e5eaa5",
  "name": "function 2",

```

```

"func": "global.set(\`cuprumWire1\`, global.get(\`cuprumWire1\`) + 1);\nvar m = [\n  { label: \"Первая плавильня меди\", payload:
global.get(\`cuprumBeforePlav1\`), series: \"Кол-во вошедшей руды\" },\n  { label: \"Первая плавильня меди\", payload:
global.get(\`cuprumAfterPlav1\`), series: \"Кол-во вышедшей переплавленной руды\" },\n  { label: \"Первый сборочный цех\", payload:
global.get(\`cuprumAfterPlav1\`), series: \"Кол-во вышедшей переплавленной руды\" },\n  { label: \"Первый сборочный цех\", payload:
global.get(\`cuprumWire1\`), series: \"Кол-во производимых медных проволок\" },\n];\nmsg.cuprum = m;\nreturn [msg.cuprum];",
"outputs": 1,
"noerr": 0,
"initialize": "/// Добавленный здесь код будет исполняться\n// однократно при развертывании узла.\nglobal.set(\`cuprumWire1\`, 0);",
"finalize": "",
"libs": [],
"x": 2720,
"y": 120,
"wires": [
  [
    "6966d05d14392ed0"
  ]
]
},
{
  "id": "6966d05d14392ed0",
  "type": "ui_chart",
  "z": "780c730ed6e5eaa5",
  "name": "Переплавка медной руды",
  "group": "1fa81e61ec2e7298",
  "order": 1,
  "width": "10",
  "height": "7",
  "label": "Поток медной обработки",
  "chartType": "bar",
  "legend": "true",
  "xformat": "HH:mm:ss",
  "interpolate": "linear",
  "nodata": "",
  "dot": false,
  "ymin": "",
  "ymax": "",
  "removeOlder": 1,
  "removeOlderPoints": "",
  "removeOlderUnit": "3600",
  "cutout": 0,
  "useOneColor": false,
  "useUTC": false,
  "colors": [
    "#0180fe",
    "#ff0a0a",
    "#ff7f0e",
    "#1ed21e",
    "#98df8a",
    "#640c0c",
    "#ff9896",
    "#9467bd",
    "#c5b0d5"
  ],
  "outputs": 1,
  "useDifferentColor": false,
  "className": "",
  "x": 3020,
  "y": 120,
  "wires": [
    []
  ]
},
{
  "id": "8de5f16c54b0b0f0",
  "type": "function",
  "z": "780c730ed6e5eaa5",
  "name": "function 3",
  "func": "global.set(\`cuprumBeforePlav2\`,\n  global.get(\`cuprumBeforePlav2\`) + 1);\n",
  "outputs": 1,
  "noerr": 0,
  "initialize": "/// Добавленный здесь код будет исполняться\n// однократно при развертывании
узла\nnglobal.set(\`cuprumBeforePlav2\`, 0);",
  "finalize": "",
  "libs": [],
  "x": 1420,
  "y": 500,
  "wires": [
    []
  ]
}

```

```

    },
    {
      "id": "de1e31c80178bb1b",
      "type": "function",
      "z": "780c730ed6e5eaa5",
      "name": "function 4",
      "func": "var m = [\n  { label: \"Вторая плавильня меди\", payload: global.get(\"cuprumBeforePlav2\"), series: \"Кол-во пришедшей руды\" },\n  { label: \"Вторая плавильня меди\", payload: global.get(\"cuprumAfterPlav2\"), series: \"Кол-во вышедшей переплавленной медной руды\" },\n  { label: \"Второй сборочный цех\", payload: global.get(\"cuprumAfterPlav2\"), series: \"Кол-во вышедшей переплавленной медной руды\" },\n  { label: \"Второй сборочный цех\", payload: global.get(\"cuprumWire2\"), series: \"Кол-во производимых медных проволок\" },\n  { label: \"Плавильня стали\", payload: global.get(\"ferrum\"), series: \"Кол-во пришедшей стальной руды\" },\n  { label: \"Плавильня стали\", payload: global.get(\"ferrumPlav\"), series: \"Кол-во вышедших стальных пластин\" },\n  { label: \"Первый агрегирующий цех\", payload: global.get(\"ferrumPlav\"), series: \"Кол-во пришедших стальных пластин\" },\n  { label: \"Первый агрегирующий цех\", payload: global.get(\"cuprumWire2\"), series: \"Кол-во пришедших медных проволок\" },\n  { label: \"Первый агрегирующий цех\", payload: global.get(\"green\"), series: \"Кол-во вышедших зеленых микросхем\" },\n];\nreturn [m];",
      "outputs": 1,
      "noerr": 0,
      "initialize": "",
      "finalize": "",
      "libs": [],
      "x": 2200,
      "y": 740,
      "wires": [
        [
          "6f415b0d7541249b",
          "a6a67233afa11a8e"
        ]
      ]
    },
    {
      "id": "6f415b0d7541249b",
      "type": "ui_chart",
      "z": "780c730ed6e5eaa5",
      "name": "Агрегирующий цех для зеленных пластин",
      "group": "ac1a8e9aa4a27e05",
      "order": 1,
      "width": "10",
      "height": "8",
      "label": "Агрегирующий цех для зеленных пластин",
      "chartType": "bar",
      "legend": "true",
      "xformat": "HH:mm:ss",
      "interpolate": "linear",
      "nodata": "",
      "dot": false,
      "ymin": "",
      "ymax": "",
      "removeOlder": 1,
      "removeOlderPoints": "",
      "removeOlderUnit": "3600",
      "cutout": 0,
      "useOneColor": false,
      "useUTC": false,
      "colors": [
        "#fe01f5",
        "#0e0aff",
        "#ffd1a8",
        "#83eccc",
        "#2bff00",
        "#640c0c",
        "#ff9896",
        "#9467bd",
        "#c5b0d5"
      ],
      "outputs": 1,
      "useDifferentColor": false,
      "className": "",
      "x": 2470,
      "y": 740,
      "wires": [
        []
      ]
    },
    {
      "id": "6b616bd9d90f1219",
      "type": "function",
      "z": "780c730ed6e5eaa5",
      "name": "Сбор данных для мониторинга",

```

```

"func": "var m = [\n  { label: \"Плавильня стали\", payload: global.get(\"ferrum\"), series: \"Кол-во вошедшей руды\" },\n  { label: \"Плавильня стали\", payload: global.get(\"ferrumPlav\"), series: \"Кол-во вышедшей переплавленной стальной руды\" },\n];\nreturn [m];",
"outputs": 1,
"noerr": 0,
"initialize": "/// Добавленный здесь код будет исполняться\n// однократно при развертывании узла.\nglobal.set(\"ferrumPlav\", 0);",
"finalize": "",
"libs": [],
"x": 1610,
"y": 740,
"wires": [
  [
    "02f296c47cc94e9b"
  ]
]
},
{
  "id": "02f296c47cc94e9b",
  "type": "ui_chart",
  "z": "780c730ed6e5eaa5",
  "name": "Поток стальной обработки",
  "group": "1fa81e61ec2e7298",
  "order": 1,
  "width": "10",
  "height": "8",
  "label": "Поток стальной обработки",
  "chartType": "bar",
  "legend": "true",
  "xformat": "HH:mm:ss",
  "interpolate": "linear",
  "nodata": "",
  "dot": false,
  "ymin": "",
  "ymax": "",
  "removeOlder": 1,
  "removeOlderPoints": "",
  "removeOlderUnit": "3600",
  "cutout": 0,
  "useOneColor": false,
  "useUTC": false,
  "colors": [
    "#07832c",
    "#f88b8b",
    "#e704b6",
    "#1ed21e",
    "#98df8a",
    "#640c0c",
    "#ff9896",
    "#9467bd",
    "#c5b0d5"
  ],
  "outputs": 1,
  "useDifferentColor": false,
  "className": "",
  "x": 1920,
  "y": 740,
  "wires": [
    []
  ]
},
{
  "id": "5bf515184366a473",
  "type": "function",
  "z": "780c730ed6e5eaa5",
  "name": "function 9",
  "func": "global.set(\n  \"red\", \n  global.get(\"red\") + 1\n);\n\nvar m = [\n  { label: \"Второй агрегирующий цех\", payload: global.get(\"cuprumWire2\"), series: \"Кол-во пришедших медных проволок\" },\n  { label: \"Второй агрегирующий цех\", payload: global.get(\"green\"), series: \"Кол-во пришедших зеленных микросхем\" },\n  { label: \"Второй агрегирующий цех\", payload: global.get(\"red\"), series: \"Кол-во вышедших красных микросхем\" },\n];\nreturn [m];",
  "outputs": 1,
  "noerr": 0,
  "initialize": "/// Добавленный здесь код будет исполняться\n// однократно при развертывании узла.\nglobal.set(\"cuprumWire1\", 0);",
  "finalize": "",
  "libs": [],
  "x": 3320,
  "y": 580,
  "wires": [
    [
      "0ef50c6aa30f82d6",
      "2b5fd90e04e93e1a"
    ]
  ]
}

```

```

    ]
  ],
},
{
  "id": "0ef50c6aa30f82d6",
  "type": "ui_chart",
  "z": "780c730ed6e5eaa5",
  "name": "Агрегирующий цех для красных пластин",
  "group": "ac1a8e9aa4a27e05",
  "order": 1,
  "width": "10",
  "height": "7",
  "label": "Агрегирующий цех для красных пластин",
  "chartType": "bar",
  "legend": "true",
  "xformat": "HH:mm:ss",
  "interpolate": "linear",
  "nodata": "",
  "dot": false,
  "ymin": "",
  "ymax": "",
  "removeOlder": 1,
  "removeOlderPoints": "",
  "removeOlderUnit": "3600",
  "cutout": 0,
  "useOneColor": false,
  "useUTC": false,
  "colors": [
    "#af85ff",
    "#0ab6ff",
    "#f6fa00",
    "#83eccc",
    "#2bff00",
    "#640c0c",
    "#ff9896",
    "#9467bd",
    "#c5b0d5"
  ],
  "outputs": 1,
  "useDifferentColor": false,
  "className": "",
  "x": 3420,
  "y": 680,
  "wires": [
    []
  ]
},
{
  "id": "a6a67233afa11a8e",
  "type": "debug",
  "z": "780c730ed6e5eaa5",
  "name": "debug 3",
  "active": false,
  "tosidebar": true,
  "console": false,
  "tostatus": false,
  "complete": "true",
  "targetType": "full",
  "statusVal": "",
  "statusType": "auto",
  "x": 2380,
  "y": 860,
  "wires": []
},
{
  "id": "87091d1e63534a59",
  "type": "inject",
  "z": "780c730ed6e5eaa5",
  "name": "",
  "props": [
    {
      "p": "payload"
    },
    {
      "p": "topic",
      "vt": "str"
    }
  ]
},
"repeat": "1",

```



```

    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "topic": "",
    "payload": "",
    "payloadType": "date",
    "x": 110,
    "y": 980,
    "wires": [
      [
        "acb9c1496456ec5a"
      ]
    ]
  },
  {
    "id": "acb9c1496456ec5a",
    "type": "function",
    "z": "780c730ed6e5eaa5",
    "name": "function 14",
    "func": "const now = Date.now();\nlet time = now - flow.get(\"initTime\");\n\nlet sec = Math.round(time / 1000)%60;\nlet mins = Math.round(time / 60000);\nlet hours = Math.floor(mins / 60);\nmins %= 60;\nif (mins < 10) {\n  mins = '0' + mins;\n}\nif (sec < 10) {\n  sec = '0' + sec;\n}\nif (hours < 10) {\n  hours = '0' + hours;\n}\n\nmsg.text = hours + ':' + mins + ':' + sec;\nflow.set(\"timeNow\", msg.text);\n\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "/// Добавленный здесь код будет исполняться\n// однократно при развертывании узла.\nflow.set(\"initTime\", Date.now());\nflow.set(\"timeNow\", null);\n",
    "finalize": "",
    "libs": [],
    "x": 310,
    "y": 980,
    "wires": [
      []
    ]
  },
  {
    "id": "defb92ddca4ef642",
    "type": "ui_chart",
    "z": "780c730ed6e5eaa5",
    "name": "Переплавка медной руды",
    "group": "1fa81e61ec2e7298",
    "order": 1,
    "width": "15",
    "height": "15",
    "label": "Поток медной обработки",
    "chartType": "line",
    "legend": "true",
    "xformat": "HH:mm:ss",
    "interpolate": "step",
    "nodata": "",
    "dot": true,
    "ymin": "-0.2",
    "ymax": "",
    "removeOlder": "40",
    "removeOlderPoints": "1000",
    "removeOlderUnit": "1",
    "cutout": 0,
    "useOneColor": false,
    "useUTC": true,
    "colors": [
      "#0180fe",
      "#ff0a0a",
      "#ff7f0e",
      "#1ed21e",
      "#98df8a",
      "#640c0c",
      "#ff9896",
      "#9467bd",
      "#c5b0d5"
    ],
    "outputs": 1,
    "useDifferentColor": false,
    "className": "",
    "x": 3020,
    "y": 180,
    "wires": [
      []
    ]
  },
  {

```

```

{
  "id": "4866241f668096a6",
  "type": "ui_chart",
  "z": "780c730ed6e5eaa5",
  "name": "Переплавка медной руды",
  "group": "ac1a8e9aa4a27e05",
  "order": 1,
  "width": "15",
  "height": "15",
  "label": "Сборка красных микросхем",
  "chartType": "line",
  "legend": "true",
  "xformat": "HH:mm:ss",
  "interpolate": "bezier",
  "nodata": "",
  "dot": false,
  "ymin": "",
  "ymax": "",
  "removeOlder": "40",
  "removeOlderPoints": "1000",
  "removeOlderUnit": "60",
  "cutout": 0,
  "useOneColor": false,
  "useUTC": true,
  "colors": [
    "#01fe34",
    "#d60aff",
    "#ff7f0e",
    "#1ed21e",
    "#98df8a",
    "#640c0c",
    "#ff9896",
    "#9467bd",
    "#c5b0d5"
  ],
  "outputs": 1,
  "useDifferentColor": false,
  "className": "",
  "x": 3620,
  "y": 420,
  "wires": [
    []
  ]
},
{
  "id": "dbf9e1615096a050",
  "type": "function",
  "z": "780c730ed6e5eaa5",
  "name": "function 16",
  "func": "var m = [\n  { label: \"Первая плавильня меди\", payload: global.get(\"cuprumBeforePlav1\") -\n    global.get(\"cuprumAfterPlav1\"), series: \"Разница в кол-ве между вошедшей медной рудой/переплавленной рудой\" },\n  { label: \"Первый сборочный цех\", payload: global.get(\"cuprumAfterPlav1\") - global.get(\"cuprumWire1\"), series: \"Разница в кол-ве между\n    переплавленной рудой/медной проволокой\" }\n];\nmsg.cuprumGraph = m;\nreturn [msg.cuprumGraph];\n",
  "outputs": 1,
  "noerr": 0,
  "initialize": "",
  "finalize": "",
  "libs": [],
  "x": 2730,
  "y": 180,
  "wires": [
    [
      "defb92ddca4ef642"
    ]
  ]
},
{
  "id": "2b5fd90e04e93e1a",
  "type": "function",
  "z": "780c730ed6e5eaa5",
  "name": "function 23",
  "func": "\nvar m = [\n  { label: \"Второй агрегирующий цех\", payload: global.get(\"cuprumWire2\") - global.get(\"red\"), series: \"Разница между кол-вом\n    пришедших проволок и вышедших красных микросхем\" },\n  { label: \"Второй агрегирующий цех\", payload: global.get(\"green\") - global.get(\"red\"), series: \"Разница между кол-вом\n    пришедших зеленных микросхем и вышедших красных микросхем\" },\n  { label: \"Второй агрегирующий цех\", payload: global.get(\"cuprumWire2\") - global.get(\"green\"), series: \"Разница\n    между кол-вом пришедших проволок и зелеными микросхемами\" },\n];\nreturn [m];",
  "outputs": 1,
  "noerr": 0,
  "initialize": "",

```

```

    "finalize": "",
    "libs": [],
    "x": 3430,
    "y": 500,
    "wires": [
      [
        "4866241f668096a6"
      ]
    ]
  },
  {
    "id": "7f39473fdcede843",
    "type": "ui_spacer",
    "z": "780c730ed6e5eaa5",
    "name": "spacer",
    "group": "1fa81e61ec2e7298",
    "order": 1,
    "width": 1,
    "height": 1
  },
  {
    "id": "1fa81e61ec2e7298",
    "type": "ui_group",
    "name": "Производство медных и стальных деталей",
    "tab": "dcd11462903a0a8a",
    "order": 1,
    "disp": true,
    "width": "15",
    "collapse": true,
    "className": ""
  },
  {
    "id": "ac1a8e9aa4a27e05",
    "type": "ui_group",
    "name": "Агрегирующие цехи",
    "tab": "dcd11462903a0a8a",
    "order": 3,
    "disp": true,
    "width": "15",
    "collapse": true,
    "className": ""
  },
  {
    "id": "dcd11462903a0a8a",
    "type": "ui_tab",
    "name": "Имитационная модель",
    "icon": "dashboard",
    "order": 1,
    "disabled": false,
    "hidden": false
  }
]

```

СПРАВКА

Томский Государственный Университет

о результатах проверки текстового документа
на наличие заимствований

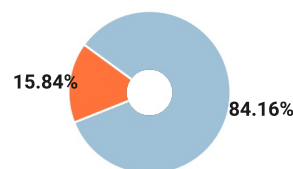
ПРОВЕРКА ВЫПОЛНЕНА В СИСТЕМЕ АНТИПЛАГИАТ.ВУЗ

Автор работы: Шилов Станислав Олегович
Самоцитирование
рассчитано для: Шилов Станислав Олегович
Название работы: Шилов С.О. ВКР
Тип работы: Выпускная квалификационная работа
Подразделение:

РЕЗУЛЬТАТЫ

СОВПАДЕНИЯ	<div><div></div></div>	15.84%
ОРИГИНАЛЬНОСТЬ	<div><div></div></div>	84.16%
ЦИТИРОВАНИЯ	<div><div></div></div>	0%
САМОЦИТИРОВАНИЯ	<div><div></div></div>	0%

ДАТА ПОСЛЕДНЕЙ ПРОВЕРКИ: 07.06.2023



Структура документа:

Проверенные разделы: основная часть с.2-4, 6-34, приложение с.36-52

Модули поиска:

ИПС Адилет; Библиография; Сводная коллекция ЭБС; Интернет Плюс*; Сводная коллекция РГБ; Цитирование; Переводные заимствования (RuEn); Переводные заимствования по eLIBRARY.RU (EnRu); Переводные заимствования по коллекции Гарант: аналитика; Переводные заимствования по коллекции Интернет в английском сегменте; Переводные заимствования по Интернету (EnRu); Переводные заимствования по коллекции Интернет в русском сегменте; Переводные заимствования издательства Wiley ; eLIBRARY.RU; СПС ГАРАНТ: аналитика; СПС ГАРАНТ: нормативно-правовая документация; Медицина; Диссертации НББ; Коллекция НБУ; Перефразирования по eLIBRARY.RU; Перефразирования по СПС ГАРАНТ: аналитика; Перефразирования по Интернету; Перефразирования по Интернету (EN); Перефразированные заимствования по коллекции Интернет в английском сегменте; Перефразированные заимствования по коллекции Интернет в русском сегменте; Перефразирования по коллекции

Работу проверил: Лапутенко Андрей Владимирович

ФИО проверяющего

Дата подписи: 07-06-2023



Подпись проверяющего



Чтобы убедиться
в подлинности справки, используйте QR-код,
который содержит ссылку на отчет.

Ответ на вопрос, является ли обнаруженное заимствование
корректным, система оставляет на усмотрение проверяющего.
Предоставленная информация не подлежит использованию
в коммерческих целях.