**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования «Южно-Российский государственный политехнический университет (НПИ) имени М.И. Платова»**



Факультет **информационных технологий и управления**

Кафедра **«Программное обеспечение вычислительной техники»**

Направление **09.04.01 –** **Информатика и вычислительная техника**

**ОТЧЕТ**

**по Лабораторной работе №3**

по дисциплине: **Программное и аппаратное обеспечение**

**информационных систем**

**Выполнил студент** 1 **курса, группы** ТИСа-о24 Блохин Э.Е.

Фамилия, имя, отчество

**Принял доцент, кандидат технических наук** Рыбалкин А.Д.

Фамилия, имя, отчество

«\_\_\_\_\_» **\_\_**\_\_\_\_\_\_\_\_\_\_\_\_2024 г. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Подпись

Новочеркасск, 2024 г

**Лабораторная работа №3**

**«Разработка аппаратной части ИС и АС или ее модели»**

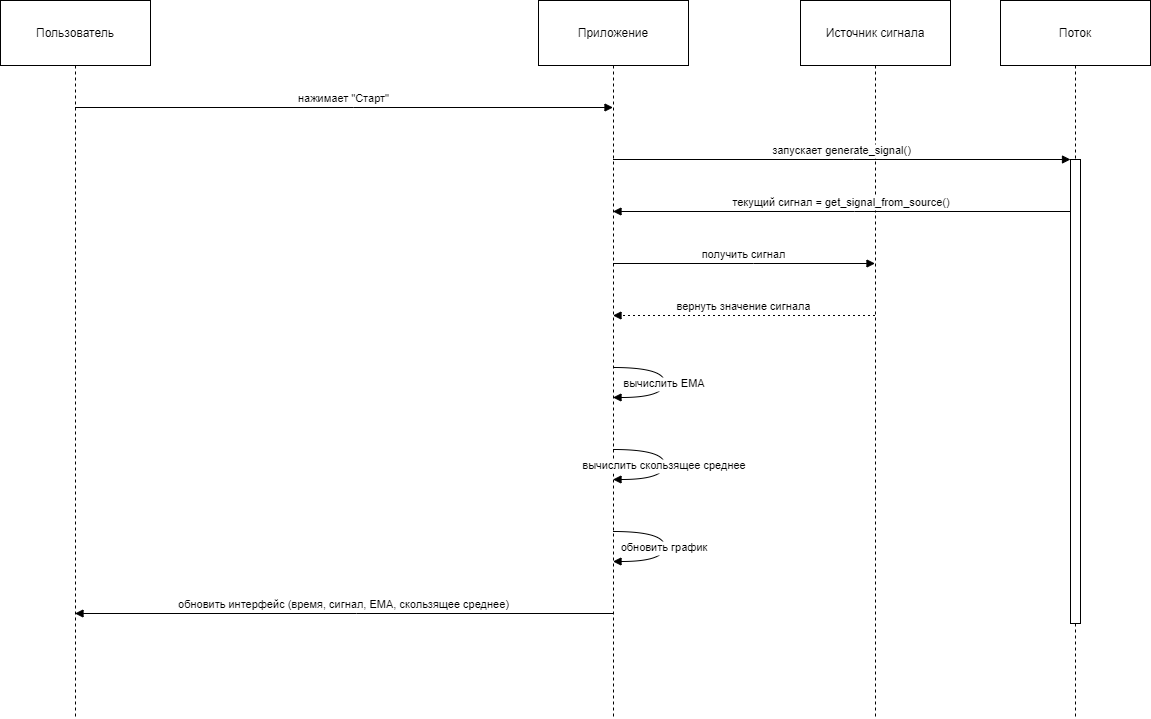
**Цель работы:** Изучить способы управления версиями программного кода, разработать каркас приложений, научиться работать с цифровыми данными, применять алгоритмы предварительной обработки информации, подготавливая для дальнейшего применения, применять CASE-средства поддержки процесса разработки программного обеспечения.

**Теоретический материал:** Современные методы разработки программного обеспечения включают нативные и кроссплатформенные инструменты. Нативные инструменты, такие как C++ и Java, предназначены для создания приложений на определённых платформах. Кроссплатформенные решения, например, JavaScript и Python, позволяют разрабатывать программы, работающие на разных операционных системах.

Современные методы разработки программного обеспечения используют нативные и кроссплатформенные инструменты. Нативные инструменты, такие как C++, C#, Java, применяются для создания приложений под конкретные платформы, обеспечивая высокую производительность и полное использование возможностей системы. Кроссплатформенные инструменты, такие как Python и JavaScript, позволяют разрабатывать приложения, работающие на разных операционных системах, что облегчает их перенос и поддержку.

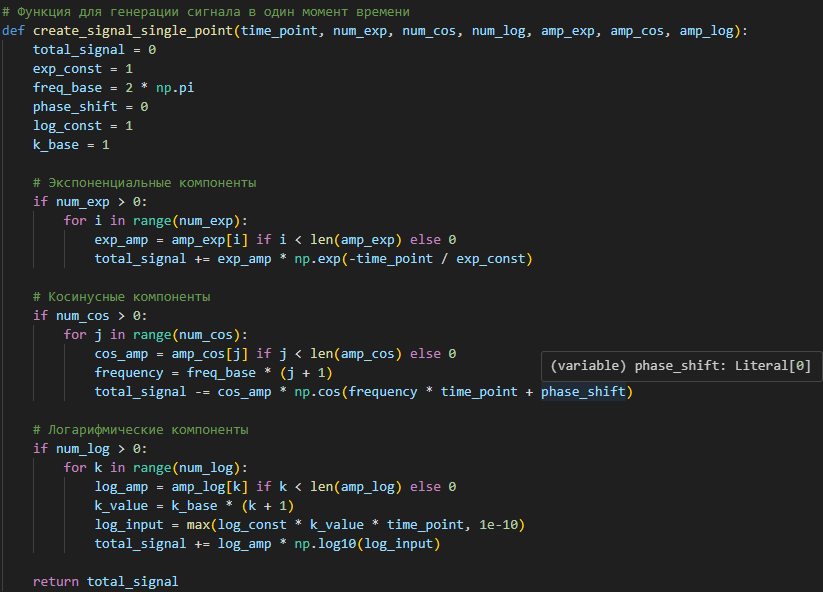
Предварительная обработка данных включает методы фильтрации и сглаживания, которые помогают улучшить качество данных. Алгоритмы, такие как скользящее среднее и экспоненциальное сглаживание (EMA), устраняют шумы и делают сигнал более пригодным для анализа. Эти методы широко применяются при работе с цифровыми сигналами и помогают обеспечить более точные результаты на этапе последующего анализа.

**Ход работы:**

1) Разработана схема последовательности, которая описывает пошаговый процесс взаимодействия компонентов системы после нажатия пользователем кнопки "Старт". На схеме показано, как SignalApp инициирует процесс генерации сигнала, запуская его в отдельном потоке (Thread). Поток вызывает метод get\_signal\_from\_source() для получения новых данных сигнала. Для этого SignalApp обращается к классу SignalSource, который отвечает за предоставление данных сигнала на основе источника. После получения сигнала в программе выполняются расчёты EMA и скользящего среднего. Затем интерфейс обновляется для отображения текущих данных, включая сигнал, EMA и скользящее среднее. Этот процесс повторяется до тех пор, пока пользователь не нажмёт кнопку "Стоп", что завершает работу приложения. Всё это показано на рисунке 1.

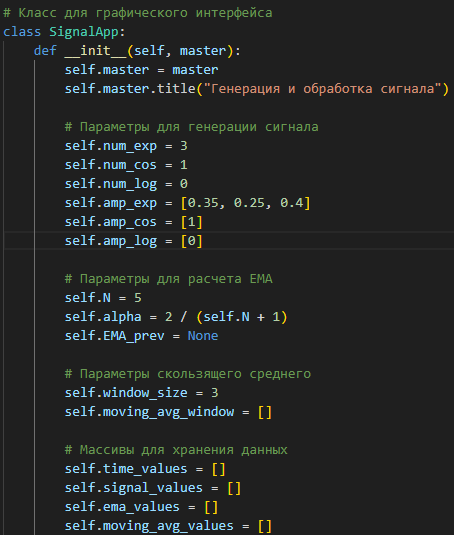
.

*Рисунок 1 – Схема последовательности*

2) В первой части кода, помимо импорта необходимых библиотек, таких как NumPy, добавляется библиотека для работы с API, которая может использоваться для получения сигнала из внешнего источника. Основной функцией является create\_signal\_single\_point, которая принимает параметры, включая временную точку и количество членов для различных компонентов сигнала. Функция суммирует значения этих компонентов для получения итогового сигнала, что показано на рисунке 2.

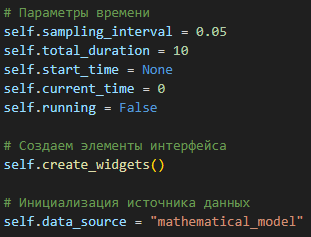
*Рисунок 2 – Импорт библиотек и создание функции генерации сигнала*

3) В этой части кода создаётся класс SignalApp, который отвечает за создание графического интерфейса для генерации и обработки сигнала. В конструкторе класса инициализируются параметры для генерации сигнала, такие как количество экспоненциальных, косинусных и логарифмических членов, а также их амплитуды. Задаются параметры для расчёта EMA (экспоненциального скользящего среднего) и скользящего среднего. Инициализируются массивы для хранения значений времени, сигнала, EMA и скользящего среднего. Также задаётся источник данных, который может быть либо математической моделью, либо внешним сенсором или API (рисунок 3).



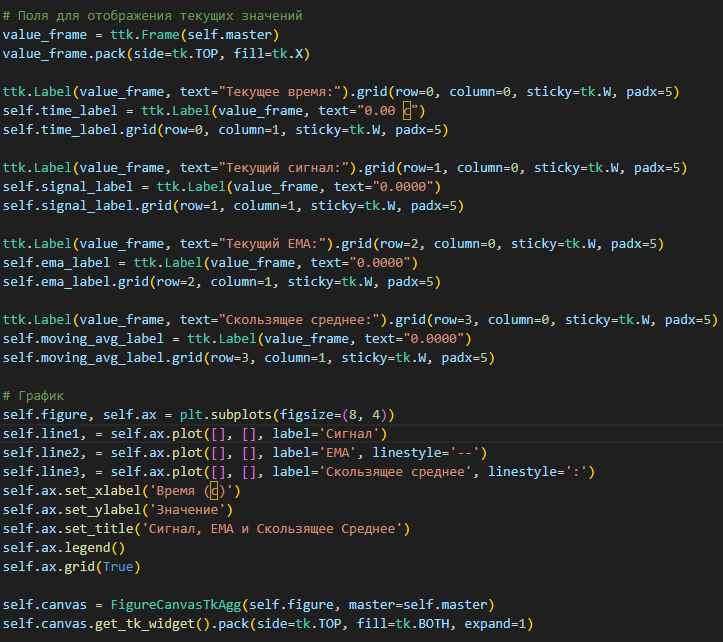
*Рисунок 3 – Создание класса SignalApp и инициализация параметров*

4) Метод create\_widgets отвечает за создание и размещение элементов управления в графическом интерфейсе. В нём создаются кнопки для управления процессом генерации сигнала («Старт» и «Стоп»), а также поля для отображения текущих значений времени, сигнала, EMA и скользящего среднего. Структурированный интерфейс позволяет управлять процессом генерации и наблюдать за изменениями в данных в режиме реального времени, что продемонстрировано на рисунке 4.

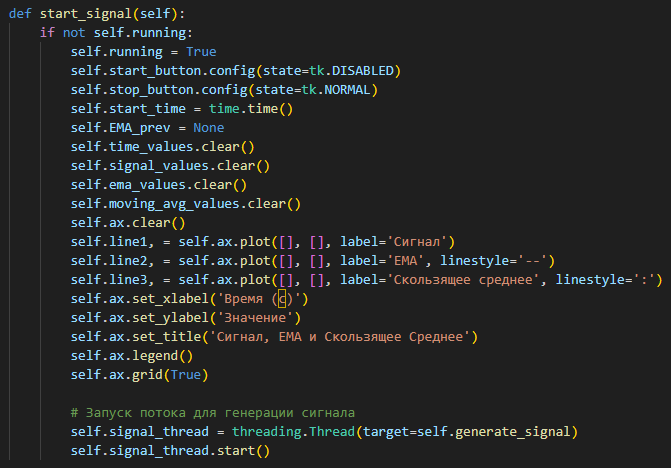


*Рисунок 4 – Создание элементов управления и отображения интерфейса*

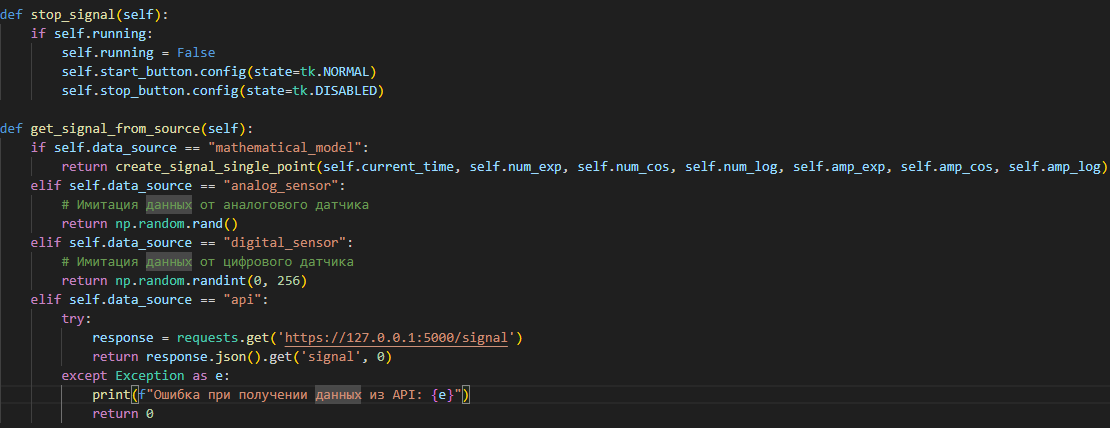
5) Здесь создаётся график для визуализации данных. Используя библиотеку Matplotlib, создаётся фигура и оси для графика, на которых будут отображаться линии для сигнала, EMA и скользящего среднего. Устанавливаются метки осей («Время» и «Значение») и заголовок графика. Линии для сигнала, EMA и скользящего среднего создаются с различными стилями (сплошная, пунктирная и точечная линии). Этот график будет обновляться в реальном времени во время генерации сигнала (рисунок 5).

*Рисунок 5 – Настройка графика для визуализации сигнала и EMA*

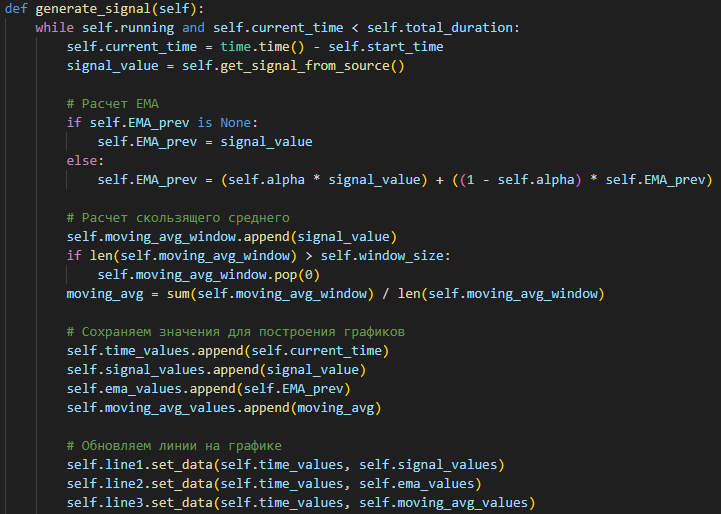
6) Методы start\_signal и stop\_signal управляют процессом генерации сигнала. Метод start\_signal запускает генерацию, инициализируя параметры времени, обнуляя массивы данных и создавая отдельный поток для генерации сигнала. Также происходит обновление интерфейса для отображения нового сигнала. Метод stop\_signal останавливает процесс генерации, деактивируя поток и меняя состояние кнопок управления, что показано на рисунке 6.

*Рисунок 6 – Управление процессом генерации сигнала*

7) Метод get\_signal\_from\_source отвечает за выбор источника данных для сигнала. В зависимости от выбранного источника (математическая модель, аналоговый или цифровой сенсор, либо API), метод возвращает соответствующее значение сигнала. Например, для математической модели используется функция create\_signal\_single\_point, для сенсоров — имитация случайных данных, а для API — запрос на сервер для получения сигнала, что показано на рисунке 7.

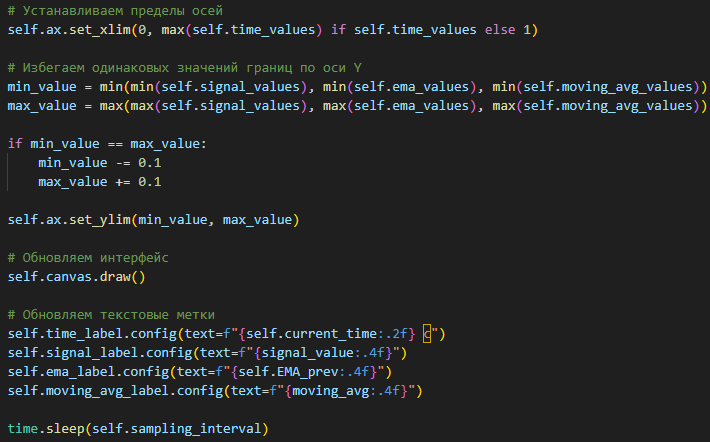


*Рисунок 7 – Получение данных сигнала из различных источников*

8) Метод generate\_signal отвечает за непрерывную генерацию сигнала в реальном времени. Во время работы этого метода вычисляется текущее время, получаются новые значения сигнала, вычисляются EMA и скользящее среднее. Значения сохраняются в массивы и используются для обновления графика, отображаемого на экране. Также обновляются текстовые метки, показывающие текущие значения сигнала, EMA и скользящего среднего, что продемонстрировано на рисунке 8.

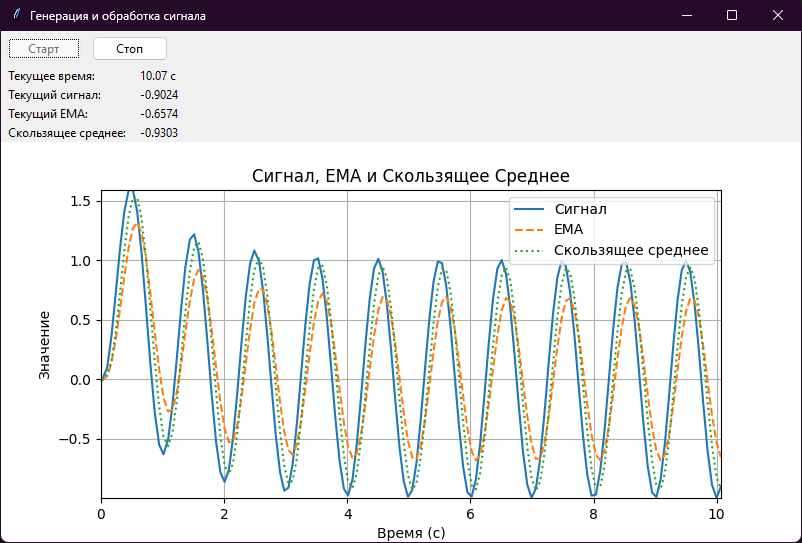
*Рисунок 8 – Генерация сигнала и обновление графика в реальном времени*

9) В последней части задаются динамические пределы осей для графика в зависимости от текущих значений сигнала, EMA и скользящего среднего. Это предотвращает ситуацию, когда график становится неподвижным из-за одинаковых значений по оси Y. Также обновляется визуальное отображение графика с помощью метода canvas.draw, который перерисовывает содержимое. Всё это показано на рисунке 9.

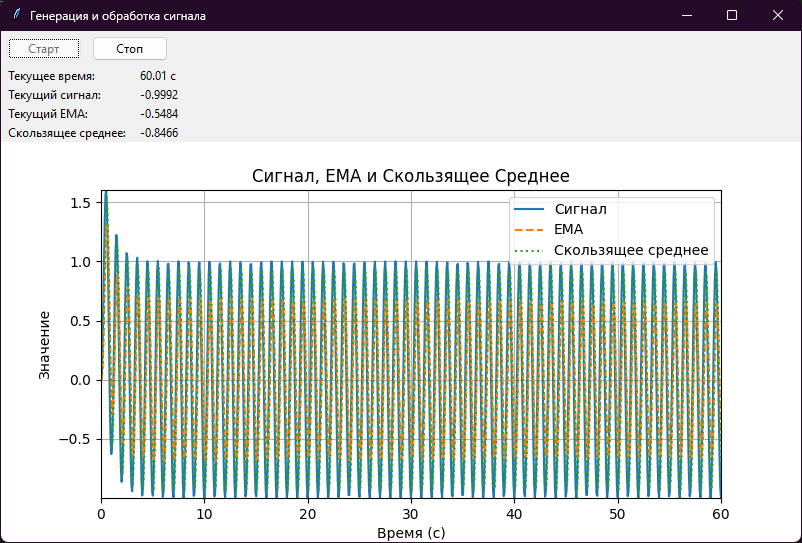
*Рисунок 9 – Обновление пределов осей и перерисовка графика*

**Результаты работы:**

После нажатия кнопки "Старт" происходит генерация сигнала с использованием функции create\_signal\_single\_point, которая создает сигнал, состоящий из экспоненциальных, косинусных и логарифмических компонентов. В отдельном потоке сигнал обновляется в реальном времени с заданным интервалом. На графике, отображаемом в интерфейсе, по оси X показывается время, а по оси Y — значения сигнала. Основной сигнал представлен сплошной линией, EMA — пунктирной, а скользящее среднее — точечной линией. В реальном времени обновляются текущие значения времени, сигнала, EMA и скользящего среднего, которые отображаются в соответствующих текстовых полях интерфейса. График автоматически обновляется с учётом текущих данных, что позволяет пользователю наблюдать за динамикой изменения сигнала. Пользователь может остановить генерацию сигнала в любой момент, нажав кнопку "Стоп", при этом поток прекращает свою работу. Результаты отображения динамики сигнала и его обработки показаны на рисунках 10 и 11.



*Рисунок 10 – Визуализированный сигнал после 10 секунд*

*Рисунок 11 – Визуализированный сигнал после 1 минуты*

**Вывод:** Разработана программа для генерации сложных сигналов. Она позволяет в реальном времени рассчитывать скользящее среднее и EMA, предоставляя удобный интерфейс для анализа динамических процессов.