**Содержание**

Введение 6

1 Модель физического уровня канала связи OSI 8

1.1 Концептуальная модель линии передачи 8

1.2 Математическая модель длинной линии связи 9

1.3 Переходные процессы в длинной линии связи 10

1.4 Физическое кодирование 12

1.5 Дискретизация модели длинной линии связи 13

1.6 Постановка задачи 17

2 Алгоримты моделирования цифровой связи 19

2.1 Задача Коши и численный метод решения 19

2.2 Выбор метода численного интегрирования и конкретизация алгоритма 19

2.3 Нахождение начальных условий задачи 21

2.4 Алгоритмы физического кодирования 22

3 Программная реализация стенда исследования цифровых каналов связи 25

3.1 Выбор структур данных 25

3.2 Выбор и обзор инструментов 25

3.2.1 Выбор языка программирования, библиотек, среды разработки 26

3.2.3 Опыт использования инструментов 28

3.3 Ожидаемый пользовательский сценарий взаимодействия с ПС 28

3.4 Объектно-ориентированное проектирование ПС 29

3.4.1 Выбор языка программирования, библиотек, среды разработки 26

3.4.2 Опыт использования инструментов 28

3.5 Выводы по третьей главе 31

4 Тестирование 32

5 Безопасность и экологичность выпускной квалификационной работы 35

5.1 Анализ вредных и опасных факторов 35

5.2 Оценка экологичности работы 36

5.3 Устойчивость к чрезвычайным ситуациям. Пожаробезопасность 37

5.4 Расчет системы искусственного освещения помещений 38

5.5 Выводы по главе 42

6 Экономическая часть выпускной квалификационной работы 43

6.1 Основные аспекты реализации проекта 43

6.2 Стратегический маркетинговый анализ целесообразности применения программного средства 44

6.3 Определение продолжительности работ по разработке программного средства 48

6.4 Построение сетевой модели комплекса работ по созданию программного средства 50

6.5 Оценка потенциальной социально-экономической результативности внедрения программного средства 52

6.6 Определение затрат на разработку и внедрение программного средства 55

6.7 Определение экономической эффективности создания и внедрения программного средства 58

6.8 Выводы по разделу 61

Заключение 62

Перечень использованных информационных ресурсов 63

Приложение А Техническое задание на программное средство 64

Приложение Б Исходный код программного средства 71

Приложение В Акт приемки/сдачи программного средства 84

Введение

Одним из главных фактором работы предприятий телекоммуникаций, отрасли информационных технологий, а также повседневной жизни обычного человека являются физические каналы или линии связи, использующиеся для обмена радиочастотными электрическими сигналами.

При передаче данных в физической линии передачи, возникают падения напряжения на конце кабеля и переходные процессы [5, с. 306], которые требуют тщательного изучения, так как если не учитывать нюансы передачи сигналов по проводникам, это может привести к искажению, зашумлению и полной потере передаваемой информации.

Также учет свойств физический уровень передачи информации важен при разработке и тестировании эффективных и надежных АЦП и ЦАП, проектировании алгоритмов канального кодирования. В этих и многих других случаях возникает необходимость оценить их работу в реальной ЛП.

Обычно трудности возникают при работе с высокочастотными и импульсными сигналами, которые часто используются в современной технике.

Именно поэтому построение моделей явлений, происходящих при передаче данных на физическом уровне, важно для изучения свойств физического кодирования.

Исследование переходных процессов и поведения напряжения на другом конце линии связи может быть произведено двумя путями:

* осуществление измерений на эксплуатируемых кабелях при помощи осциллографа;
* численное моделирование процессов, протекающих в электрических кабелях, при помощи средств вычислительных устройств;

Измерения на линиях связи позволяет получить реальное представление о происходящем, однако требует дорогостоящего осциллографа и навыков работы с ним, а также, повреждения кабеля, что в некоторых случаях недопустимо.

По этим причинам лучше проводить исследования не на реальных объектах, а на физических моделях или с помощью компьютерных программ. Применение физических моделей позволяет получить необходимые данные в исследуемой области, но имеет ряд недостатков, таких как большие временные и финансовые затраты. Также физическое моделирование сложно осуществить при большом числе возможных вариантов исследуемой длинной линии, так как для каждого варианта необходимо произвести физическую модель и провести на ней серию опытов.

Компьютерное моделирование позволяет отказаться от изготовления физических моделей, сократить временные затраты на проведение экспериментов и предоставляет возможность быстро и многократно изменять параметры исследуемого объекта.

# 1 Модель физического уровня канала связи OSI

В данном разделе приведены теоретические сведения, относящиеся к аспектам процесса передачи информации на физическом уровне.

## 1.1 Концептуальная модель линии передачи

Линия передачи (ЛП) – устройство, используемое для передачи электрических сигналов от источника электрических импульсов к нагрузке. Как правило, представляет собой систему проводов, либо в виде кабелей, используется в различных устройствах на частотах от нуля до соответствующих оптическому диапазону.

Конструкция ЛП определяется областью её применения. Простейшая ЛП состоит из двух проводников, каждый из которых обладает определенным сопротивлением и вносит некоторое затухание.

При длинах волн , превышающих длину линии, электрические процессы в ЛП описываются законом Ома. С уменьшением (практически с ), проявляется эффект запаздывания в линии.

Главными свойствами ЛП являются диапазон рабочих частот, максимально допустимая средняя и импульсная мощность, волновое сопротивление.   
 Рабочий диапазон частот ЛП обычно выбирается так, чтобы импульсы в ней распространялись с малым коэффициентом затухания и небольшой дисперсией на конкретный тип волны.

Максимально допустимая мощность зависит от тепловых потерь и эффективности теплоотвода при передаче электрических импульсов.

ЛП хорошо изучены и для них существует физико-математическая модель, которая называется “модель длинной линии”.

## 1.2 Математическая модель длинной линии связи

Длинная линия (ДЛ) связи представляет собой линию передачи, протяженность которой превышает, либо сравнима с длиной волны, распространяющейся в ней, а поперечные размеры значительно меньше длины волны.

Математическая модель длинной линии связи описывается Телеграфными уравнениями [6, с. 249] - парой дифференциальных уравнений, представленных в формуле 1.1, со следующими параметрами:

* погонное сопротивление металла проводников R Ом/м;
* погонная индуктивность L Гн/м;
* погонная ёмкость C Ф/м;
* паразитная проводимость изоляции двух проводников G См/м.

, (1.1)

В ДЛ при передаче электрических импульсов от генератора к нагрузке возникает падающая и отраженная от нагрузки волна, которые характеризуются функциями в зависимости от свойств которых выделяют режим работы ЛП:

* бегущей волны, определяемый наличием только падающей волны, распространяющейся от генератора к нагрузке;
* стоячей волны, характеризующийся тем, что амплитуда отраженной волны равна амплитуде падающей;
* смешанных волн, в котором одна часть падающей волны уходит в нагрузке, а другая в виде отраженной волны возвращается обратно в генератор.

Данная модель учитывает основные эффекты при передаче импульсов по проводникам, в числе которых падение напряжение, переходные процессы и интерференция падающей и отраженной волны.

### 1.3 Переходные процессы в длинной линии связи

Понятие переходного процесса в общем смысле относится к теории систем и обозначает реакцию динамической системы на приложенное внешнее воздействие с момента воздействия до некоторого установившегося значения во времени.

Длинную линию можно рассматривать как такую систему. В процессе передачи информации на физическом уровне в форме электрических импульсов, протекают явления в электрических цепях, возникающие при резких воздействиях, переводящих их новое стационарное состояние, описываемых единичными функциями [5. с. 351]. То есть, — при действии какой-то коммутационной техники, например, ключей, переключателей источника, или нагрузки, при обрывах в электрической цепи, а также при коротких замыканиях отдельных участков цепи и так далее.

В теоретическом анализе переходных процессов в электрических цепях для однозначного определения понятия “резкий”, закон изменения напряжения U на входе в длинную линию связи выбирают по аналогии с формулой 1.2.

, (1.2)

Таким образом, в момент времени напряжение мгновенно изменяется от до величины , которая далее остается неизменной сколь угодно долго.

Важным свойством электрических цепей является то, что из-за наличия так называемых паразитных эффектов: индуктивностей и ёмкостей, - в реальных электрических цепях получение мгновенного скачка напряжения или тока физически невозможно, так как требует применения схем формирования тока бесконечно большой мощности.

При переходных процессах могут происходить перенапряжения, сверхтоки, электромагнитные колебания, препятствующие корректному функционированию устройства, способные привести к его поломке. С другой стороны, переходные процессы находят применение, например, в различных цифровых схемах, генерирующих импульсы.

В передаче данных на физическом уровне участвуют передатчик (генератор прямоугольных электрических импульсов) и приемник. Приемник преобразует входной аналоговый сигнал в цифровой, передатчик, по аналогии с законом из формулы 1.2, многократно для передачи информации многократно меняет выдаваемый уровень напряжения. Эти устройства выполняют работу с некоторый периодичностью секунд, при этом величина называется тактовой частотой такой цифровой схемы.

Особое внимание при передаче данных на физическом уровне уделяется синхронизации этих устройств., которое, как правило, осложняется влиянием потерь электрической энергии в сопротивлении, а также наличием индуктивности линии.

При резком изменении уровня напряжения на входе длинной линии, поведение напряжения на другом конце имеет вид “затуханий” по характеру напоминающих затухание по показательному закону, как на рисунке 1.1. Таким образом, если приемник и передатчик работают не в такт, это может привести к тому, что вместо одного переданного бита приемником будет распознано несколько или данные вовсе будут пропущены.

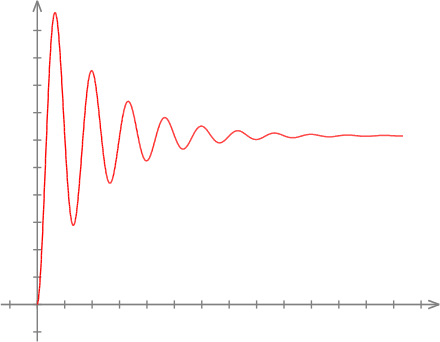


Рисунок 1.1 – Затухающие колебания

Как правило, двоичные данные перед передаче в физическом канале, подвергаются предобработке по некоторому алгоритму физического кодирования.

Такое преобразование способствует решению проблем синхронизации, обеспечения приемлемой скорости передачи данных, управления частотами передаваемых импульсов, а также может позволить более наглядно раскрыть свойства ЛП и повысить оперативность исследования их характеристик.

## 1.4 Физическое кодирование

Физическое кодирование дискретных данных осуществляется с использованием потенциальных или импульсных кодов. Для представления двоичных нулей и единиц в кодах используются разные значения уровня напряжения, изменяющиеся с некоторой частотой по аналогии с законом в формуле 1.2.

С точки зрения физического кодирования цифровой сигнал может иметь два, три, четыре, пять и т. д. уровней амплитуды напряжения, амплитуды тока.

Надежность и достоверность передачи данных, возможность обнаружения и коррекции возникающих ошибок, пропускная способность среды передачи, стоимость реализации во многом зависят от выбранного метода физического кодирования.

Поэтому, для обеспечения качества передачи данных к методам кодирования предъявляются следующие требования:

* уменьшение спектра сигнала при одинаковой битовой скорости;
* поддержка синхронизации между передатчиком и приемником за счёт наличия в сигналах признаков, на основе которых реализуется самосинхронизация;
* отсутствие постоянной составляющей в сигнале;
* низкая стоимость реализации метода.

Требования к методам цифрового кодирования противоречивы. При этом каждый из методов цифрового кодирования по сравнению с другими обладает своими конкретными достоинствами и недостатками.

## 1.5 Дискретизация модели длинной линии связи

Для решения системы телеграфных уравнений в частных производных, представленной в формуле 1.1, с помощью программных средств, необходимо прежде всего сделать замену неким конечным, дискретным соотношением. Это осуществляется применением дискретной по пространству ячеечной модели. Схема ячеечной модели приведена на рисунке 1.2, где i – номер ячейки, n – количество ячеек, t – время.

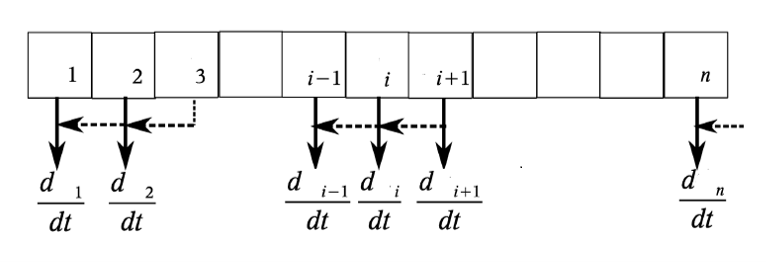


Рисунок 1.2 – Схема метода дискретизации

Длинную линию связи можно рассматривать как распределённую активную и реактивную нагрузку. Выделенный бесконечно короткий участок такой линии характеризуется постоянным активным, ёмкостным и индуктивным сопротивлением переменному току — R, L и C. Такой участок представляет собой классический линейный четырехполюсник.

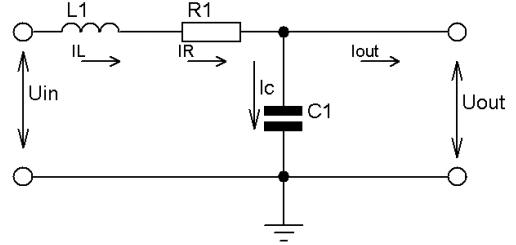


Рисунок 1.3 – Пассивный четырехполюсник

При допущении, что помимо эффектов ёмкости и самоиндукции и рассеяния энергии электричества в линии связи не наблюдается других физических эффектов, участок линии связи может быть приближённо смоделирован пассивным RLC четырёхполюсником, который изображен на рисунке 1.3.

Для такого четырехполюсника выполняются законы Кирхгофа [5, с. 42-33] из формул 1.3, 1.4, а также формулы 1.5, 1.6.

, (1.3)

, (1.4)

, (1.5)

, (1.6)

, (1.7)

, (1.8)

, (1.9)

Необходимо отметить, что в формулах 1.3 – 1.9 все параметры I и U - функции времени t.

Из формул 1.3 – 1.6 можно вывести систему ОДУ, характеризующую пассивный RLC четырехполюсник. Система представлена в формуле 1.7, где – входное напряжение, – сила тока на выходе.

Систему из формулы 1.7 можно привести к задаче Коши, которая представлена в формуле 1.9, если найти начальные условия, решив СЛАУ при условии регулярности линии, которое представлено в формуле 1.8.

При предельном переходе система уравнений из формул 1.3 – 1.6 должна быть эквивалентна системе телеграфных уравнений для линии электрической связи. Таким образом, на метров кабеля придётся , , , а на входе приложено напряжение , правее конденсатора в горизонтальной ветви течёт выходной ток .

После такого преобразования получим цепь из последовательно соединенных четырехполюсников, изображенная рисунке 1.4.

Данная модель является хорошим приближением к реальной линии передачи.

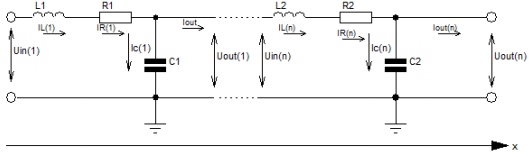


Рисунок 1.4 – Схема модели длинной линии

Математическая модель, характеризующая явления, происходящие, в такой цепи, представляет собой систему дифференциальных уравнений, которая представлена в формуле 1.10.

, (1.10)

Эти уравнения построены при допущении, что помимо эффектов ёмкости и самоиндукции и рассеяния энергии электричества в линии связи не наблюдается других физических эффектов и являются неким конечным дискретным соотношением для телеграфных уравнений.

Система ОДУ из формулы 1.10 и будет использоваться в качестве математической модели физического уровня передачи информации.

Полученная задача Коши может быть решена двумя способами: аналитическим и численным, имеющим программно-алгоритмическое воплощение.

## 1.6 Постановка задачи

Создание программного стенда важно для обеспечения обучающихся удобным средством изучения переходных процессов, импульсно-частотных характеристик линии связи, влияния выбора метода физического кодирования на качество передачи данных.

Целью данной работы является создание программного стенда моделирования передачи информации на физическом уровне.

Наиболее полным и точным описание процессов в линии передачи является её математическая модель. Модель представляет собой систему дифференциальных уравнений в полных производных. Чтобы создать программную реализацию такой модели, нужны алгоритмы, реализующие методы вычислительной математики. Таким образом, можно выделить перечень задач для создания программного стенда:

* знакомство с численным моделированием физико-технических объектов и систем распределёнными параметрами, методами цифрового кодирования;
* изучение применяемых математических моделей и алгоритмов;
* анализ посылок и допущений, определяющих адекватность модели явлению, вычислительных целей и алгоритмических способов их достижения;
* выбор необходимых и совместимых программных средств, инструментального ПО при реализации и комплексировании программной системы;
* разработка компьютерной модели и её учебно-исследовательского интерфейса.

Важным требованием к программному стенду является наличие графического интерфейса с формой ввода параметров кодера, линии и моделирования, а также вывод результатов моделирования в виде графиков.

# 2 Алгоритмы моделирования цифровой связи

## 2.1 Задача Коши и численный метод решения

В ходе выполнения данной работы, возникла необходимость решения задачи Коши, общая форма которой представлена в формуле 2.1.

, (2.1)

Прямоугольные импульсы представлены сложной функцией от времени, поэтому рационально использовать численный метод решения.

Главная идея численных методов заключается в следующем: рассматривается последовательность точек , (0 возможно с переменным шагом .

В каждой точке решение аппроксимируется значением , которое вычисляется по предыдущим значениям . Такие методы называются k-шаговыми. Точность и сложность вычислений, во многом, зависит от выбора такого метода.

## 2.2 Выбор метода численного интегрирования и конкретизация алгоритма

Для численного решения задачи Коши существует много различных методов, которые в общем случае делятся на 2 класса:

* дискретные методы, позволяющие найти решение только в узлах сетки. В этих методах каждая вновь вычисленная точка решения уравнения определяется через одну или несколько предшествующих точек;
* непрерывно-дискретные методы, основанные на использовании дискретных методов и сплайн-функций для восполнения численных результатов.

В этой работе достаточно дискретных методов, поэтому далее рассмотрены только они.

* метод Эйлера. Существенным недостатком простого метода Эйлера является слишком большая погрешность, при этом погрешность имеет тенденцию накапливаться, что не подходит при моделировании на больших промежутках времени;
* усовершенствованный метод Эйлера. Показывает лучшие результаты в сравнении с обыкновенным методом Эйлера, но при этом недостаточно точен для данной задачи;
* классический метод Рунге-Кутты 4-го порядка. Обеспечивает высокую точность решения, которая оправдывает увеличение объема вычислений. Более высокая точность метода Рунге – Кутта часто позволяет увеличить шаг интегрирования, что необходимо в программном стенде.

Так как методы Рунге-Кутты высокого порядка, в общем случае, имеют большую точность [7], для решения задачи был выбран метод Рунге-Кутты 4-го порядка.

Рассмотрим алгоритм метода Рунге-Кутта 4-го порядка [8, с. 134] для функции f с переменной x и величиной шага h для каждого номера шага по времени i. По этому алгоритму, блок-схема которого изображена на рисунке 2.1, необходимо выполнить:

1. вычислить коэффициенты k:

* ;
* ;
* ;
* ;

1. найти поправку: (;
2. вычислить очередное значение: ;

Для решения поставленной задачи используется Классический метод Рунге-Кутты четвертого порядка с постоянным шагом, так как:

* этот метод достаточно устойчив;
* четвертый порядок обеспечивает приемлемую точность;
* простой, удобен для программирования.

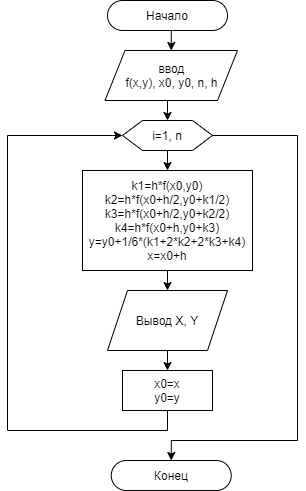


Рисунок 2.1 – Блок-схема алгоритма Рунге-Кутты 4 порядка.

Найти начальные условия можно используя систему ДУ и условие регулярности длинной линии из формул 1.10 и 1.8 соответственно, и применив элементарные преобразования к полученной СЛАУ, можно получить формулы для нахождения начальной силы тока и напряжения i - го четырехполюсника:

*,* (2.2)

, (2.3)

где – функция силы тока на нагрузке, – функция напряжения подаваемого на вход линии, R – общее сопротивление линии передачи, i – индекс четырехполюсника, – общее число четырехполюсников.

## 2.4 Алгоритмы физического кодирования

Для лучшего изучения процессов, происходящих при передаче данных на физическом уровне, желательно рассмотреть методы кодирования с различной максимальной частоты несущей.

Поэтому в данной работе будут построены следующие алгоритмы преобразования цифровых последовательностей в сигналы сложной структуры:

* RZ;
* MLT-3;
* Manchester IEEE 802.3.

В RZ формирование сигнала происходит по следующему правилу: возврат к нулевому уровню происходит после передачи значащего интервала. Информационный переход осуществляется в начале значащего интервала, возврат к нулевому уровню — в середине значащего интервала, далее уровень является нулевым до конца значащего интервала [9, с. 34]. Блок-схема данного алгоритма представлена на рисунке 2.2.

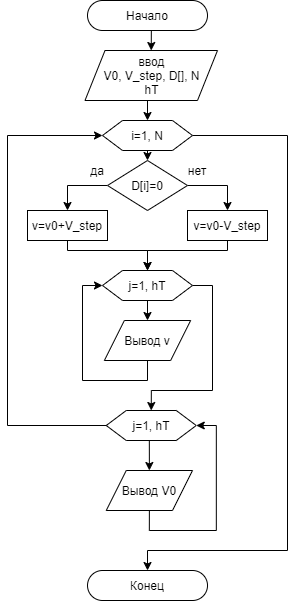


Рисунок 2.2 – Блок-схема алгоритма RZ кодирования

В MLT-3 формирование сигнала происходит по следующему правилу: последовательно переключаются уровни напряжения -1, 0, +1, 0, -1. Он переходит в следующее состояние для передачи 1 бита и остается в том же состоянии для передачи 0 бита [10].

В Manchester IEEE 802.3 логическому нулю соответствует переход на верхний уровень в центре битового интервала, логической единице - переход на нижний уровень. Логика кодирования хорошо видна на примере передачи последовательности единиц или нулей. При передаче чередующихся битов частота следования импульсов уменьшается в два раза. Блок-схема данного алгоритма представлена на рисунке 2.3.

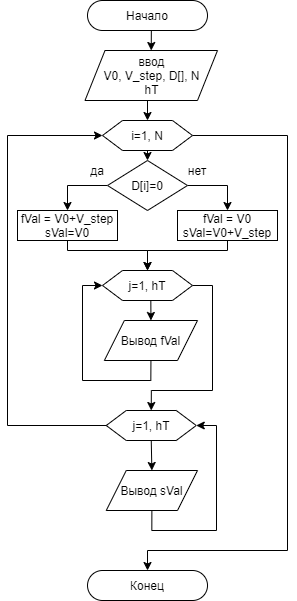


Рисунок 2.3 – Блок-схема алгоритма Manchester IEEE 802.3

Изменение сигнала в центре каждого бита позволяет легко выделить синхросигнал. Самосинхронизация дает возможность передачи больших пакетов информацию без потерь из-за различий тактовой частоты передатчика и приемника [10].

# 3 Программная реализация стенда исследования цифровых каналов связи

В данном разделе представлен ожидаемый сценарий взаимодействия с программным стендом, выбраны инструменты для реализации, а также разделена функциональность приложения на классы в объектно-ориентированном стиле.

### 3.1 Выбор структур данных

Процесс моделирования заключается в нахождении численного решения задачи Коши из системы 1.10 с некоторыми начальными условиями. Система описывает последовательно соединенных четырехполюсников – узлов сетки, каждый со своим состоянием (величиной силы тока и напряжения) в моменты времени. Поведение такой системы зависит от функций входного напряжения линии и силы тока на нагрузке , которые представлены задачи представлены таблицами значений на временной сетке.  
 Моделирование длинной линии средствами ЭВМ предполагает итерирование по ячейкам двумерной расчетной сетки по пространству и времени, поэтому динамический массив является наиболее подходящей коллекцией для хранения значений состояний узлов и табличных функций, так как представляет собой непрерывную область в памяти и имеет небольшие относительно других структур данных задержки чтения и записи.

### 3.2 Выбор и обзор инструментов

Выбор программного обеспечения обуславливается спецификой языка программирования, удобством использования, возможностью запуска на различных платформах, содержанием инструмента для решения сложных математических задач, взаимодействия с пользователем и осуществлением ввода-вывода, а также доступностью.

### 3.2.1 Выбор языка программирования, библиотек, среды разработки

Главными требованиями к языку программирования является наличие библиотек для создания графического пользовательского интерфейса, библиотек компьютерной математики, средства построения графиков функций.

Рассмотрим наиболее популярные в настоящий момент языки программирования:

Язык Python часто используется в научных вычислениях. К его преимуществам относят: простой синтаксис, наличие библиотек, предназначенных для инженерных расчётов. Существенным недостатком Python является интерпретируемость, что отражается на производительности.

Язык C++ широко используется для разработки программного обеспечения с повышенными требованиями к производительности. Также, как и для Python, на C++ написано множество библиотек, предназначенных для научных расчетов.

Учитывая все преимущества и недостатки этих языков, был сделан выбор в пользу C++ ввиду его производительности, которая является самым важным требованием при большой разрешающей способности разностной схемы.

Помимо этого, C++ обладает следующими положительными качествами [4]:

* модульность и раздельная компиляция;
* обработка исключений;
* абстракция данных;
* объявление типов (классов) объектов;
* наличие стандартной библиотеки с широкими возможностями;
* сочетание свойств высокоуровневых и низкоуровневых языков;
* компиляция программного кода непосредственно в машинный код.

Разработка осуществляется в интегрированной среде разработки (IDE) Qt Creator. Данная среда разработки является кроссплатформенной и свободно распространяемой, поддерживает возможность расширения функционала через установку дополнительны плагинов.

Для компиляции используется компилятор GCC, работающий под операционной системой Linux и также свободно распространяемый.

Для реализации численного решения математической модели, являющейся системой дифференциальных уравнений, необходимо найти решение системы линейных алгебраических уравнений, а также численно проинтегрировать систему дифференциальных уравнений.

Для первой задачи выбрана библиотека Eigen [2], которая содержит определения подходящих типов данных и определение кода подпрограмм (функций) для решения СЛАУ, является кроссплатформенной и свободно распространяемой. Она также имеет развернутую документацию.

Для решения задачи Коши выбрана библиотека boost::odeint [3], которая содержит реализации алгоритмов решения дифференциальных уравнений, в том числе классический метод Рунге-Кутты 4-го порядка. Является кроссплатформенной и свободно распространяемой, имеет отличную документацию.

Для реализации взаимодействия с пользователем, ввода им исходных данных и вывода результатов моделирования необходимы дополнительные библиотеки с возможностью вывода результатов в графической форме, выбрана библиотека (фреймворк) Qt [1], которая содержит компоненты графического интерфейса, функции ввода-вывода информации, является кроссплатформенной и распространяемой по универсальной общественной лицензии (GNU GPL).

### 3.2.2 Опыт использования инструментов

Выбор C++ в качестве языка программирования оказался верным решением. Стандартная библиотека libstdc++ содержит реализации всех необходимых структур данных, кросс-платформенные обёртки над средствами ОС и примитивы синхронизации.

Были использованы потоки для асинхронных вычислений модели линии связи, а также атомарные переменные для остановки расчёта по запросу пользователя из потока интерфейса.

Среда разработки Qt Creator позволила без особых усилий сверстать пользовательский графический интерфейс, обеспечила удобным редактором текста с подсветкой синтаксиса, помогла провести отладку программы.

В фреймворке Qt есть все необходимые инструменты для считывания пользовательского ввода – текстовые, числовые, выпадающие списки выбора, гибкая система обработки событий – SLOTS, а также средства вывода графиков. Также, важно отметить, что это динамическая библиотека, однако все требуемые в Т.З. платформы поддерживаются.

Библиотеки Eigen и boost::odeint предоставляют все необходимые примитивы для численных решений задач линейной алгебры и систем дифференциальных уравнений – реализации матриц, векторов и операций над ними, а также удобный программный интерфейс вызова методов решения дифференциальных уравнений.

## 3.3 Ожидаемый пользовательский сценарий взаимодействия с ПС.

Сценарий взаимодействия с программным средством представляет собой простую последовательность шагов:

Шаг 1. Ввод параметров моделирования.

Шаг 2. Запуск моделирования.

Шаг 3. Ожидание результата или остановка и переход к шагу 1.

Шаг 4. Отображение результата в виде графиков.

Блок-схема шагов этого сценария представлена на рисунке 3.1.

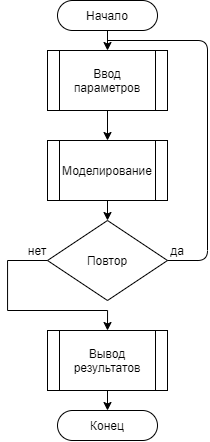


Рисунок 3.1 – Главный пользовательский сценарий

### 3.4 Объектно-ориентированное проектирование ПС

Весь функционал, требуемый от данного ПС можно условно разбить на несколько классов:

От класса “MainWindow” требуется обеспечение ввода параметров, запуска и остановки моделирования, должен содержать следующие методы:

* метод init должен инициализировать поля параметрами по умолчанию и устанавливающий ограничения;
* метод start должен запускать процесс моделирования;
* метод stop должен останавливать моделирование;
* метод draw должен выводить выходное напряжение в виде графиков;
* метод clear должен графики;
* поле stopped должно отражать состояние – моделирование или ввод параметров.

Интерфейс “Encoder”, представляющий схему кодирования данных, содержащий следующие методы:

* метод encode - это главный метод, который должен отвечать за кодирование. Принимает на вход данные, возвращает последовательность напряжений для моментов времени ;
* поле zero – уровень напряжения, взятый за ноль;
* поле step – шаг в уровнях напряжения кодера.

Классы “RZEncoder”, “ManchesterEncoder”, “MLT3Encoder”, должны реализовать интерфейс “Encoder” согласно алгоритму цифрового кодирования.

Главный класс программы “Machine”, должен отвечать за численное моделирование со следующими методами:

* поле opt – параметры проводника;
* поле N – число четырехполюсников;
* метод initState должен принимать значение входного напряжение и силы тока на выходе в момент и возвращать начальные условия;
* поле payload хранящее нагрузку в виде массивов значений силы тока и напряжения в моменты времени , и шага h;
* метод processNextPayload должен возвращать последовательность напряжений для моментов времени .

Связи между классами удобно изображать на основе стандарта языка UML, который используется для спецификации, визуализации, конструирования и документирования системы.

UML-диаграмма данных классов, представленная на рисунке 3.2, а также описание классов, представленное выше, отражает общую схему классов в ПС и не является строгим требованием.

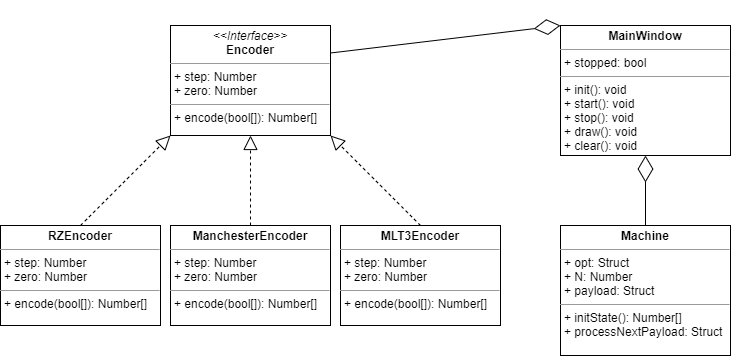


Рисунок 3.2 – UML-диаграмма классов ПС.

## 3.5 Вывод по третьей главе

В данном разделе был описан типовой сценарий взаимодействия пользователя с ПС, создана объектно-ориентированная модель ПС, осуществлен обоснованный выбор инструментов для реализации, а также описан опыт исполнителя при работе с инструментами.

# 4 Тестирование

В данном разделе представлены примеры работы программного средства: ввод параметров, ожидание окончания расчетов, вывод графиков, краткие пояснения к наблюдаемым результатам моделирования.

После запуска исполняемого файла и прорисовки компонентов интерфейса, пользователь может увидеть форму для ввода параметров, это отражено на рисунке 4.1

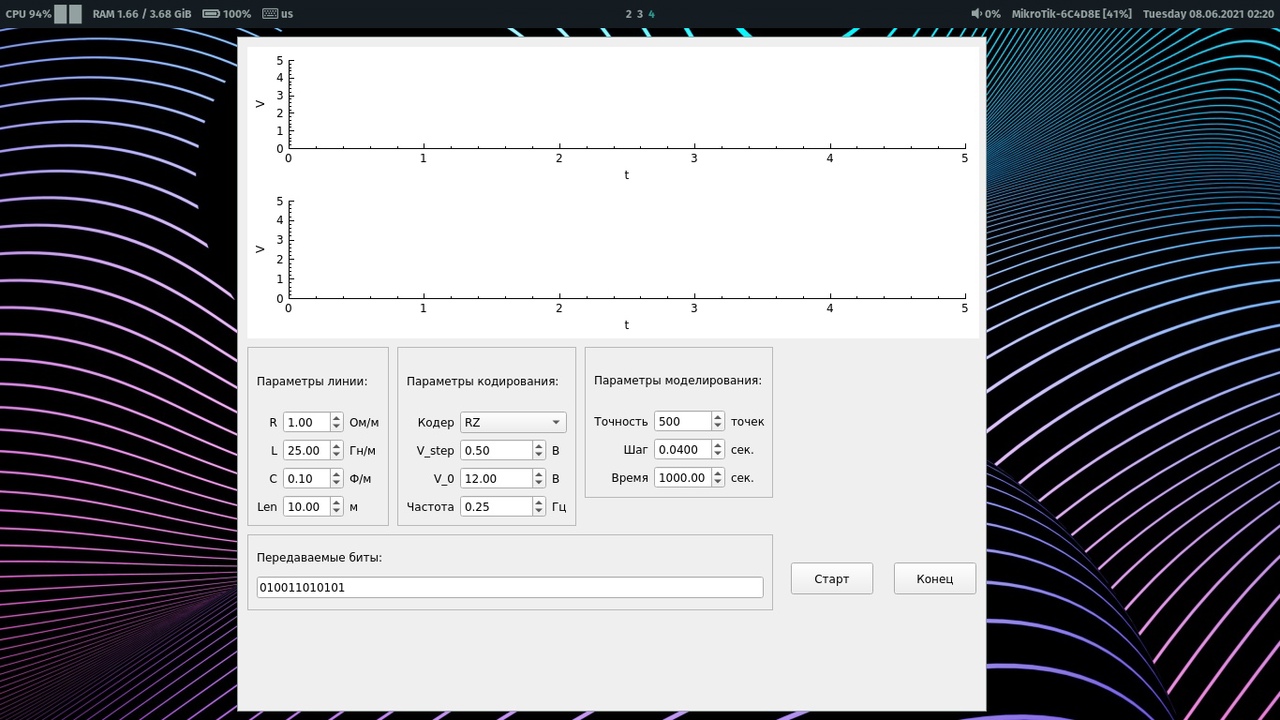


Рисунок 4.1 – Форма ввода параметров

Перед моделированием пользователю необходимо ввести параметры линии передачи, свойств кодирования, по нажатию кнопки “Старт” запускается процесс моделирования с индикатором прогресса, это отражено на рисунке 4.2.

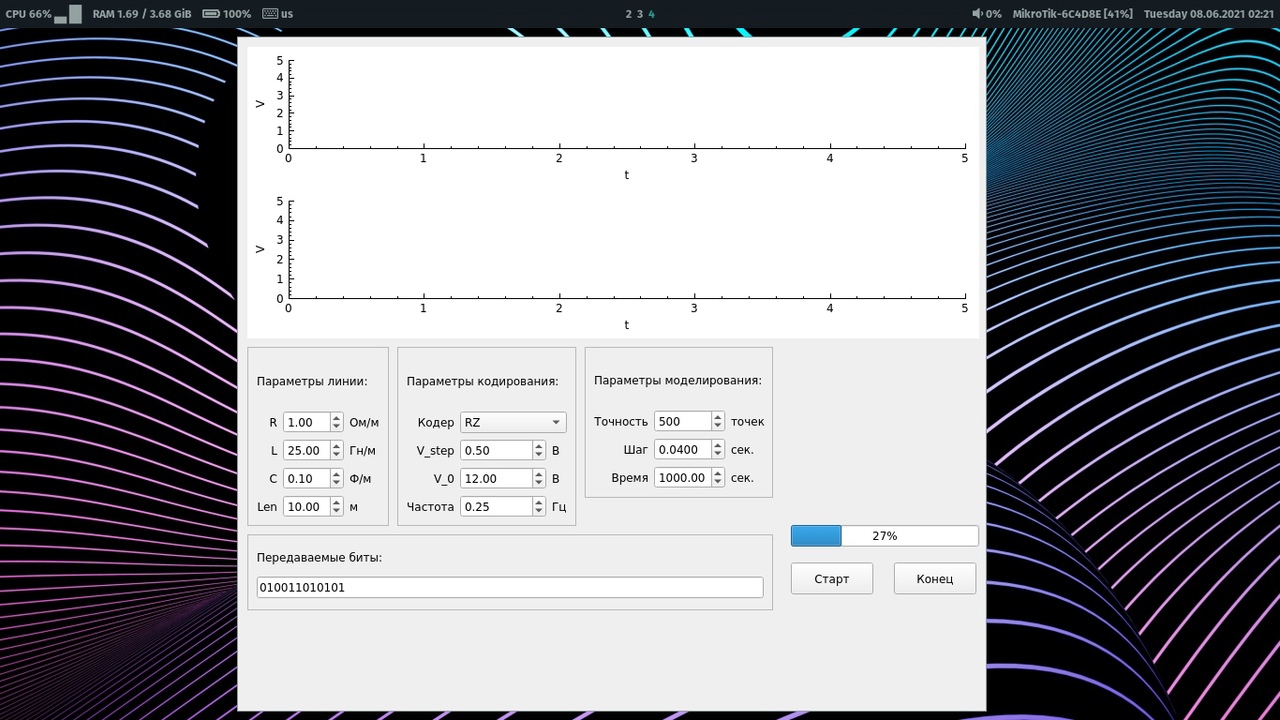


Рисунок 4.2 – Ожидание результата

После ожидания результата, на верхнем и нижнем графиках будет изображен сигнал, переданный на вход длинной линии и полученный на выходе соответственно, примеры представлены на рисунках 4.3 и 4.4.

Важным для пользователя свойством работы стенда является возможность наблюдения переходного процесса в длинной линии. Как было упомянуто в разделе 1.3, при воздействии в виде резкой смены уровня напряжения, подаваемого на вход, поведение уровня напряжения на выходе напоминает затухающие колебания по показательному закону, что отражено на рисунках 4.3 и 4.4.

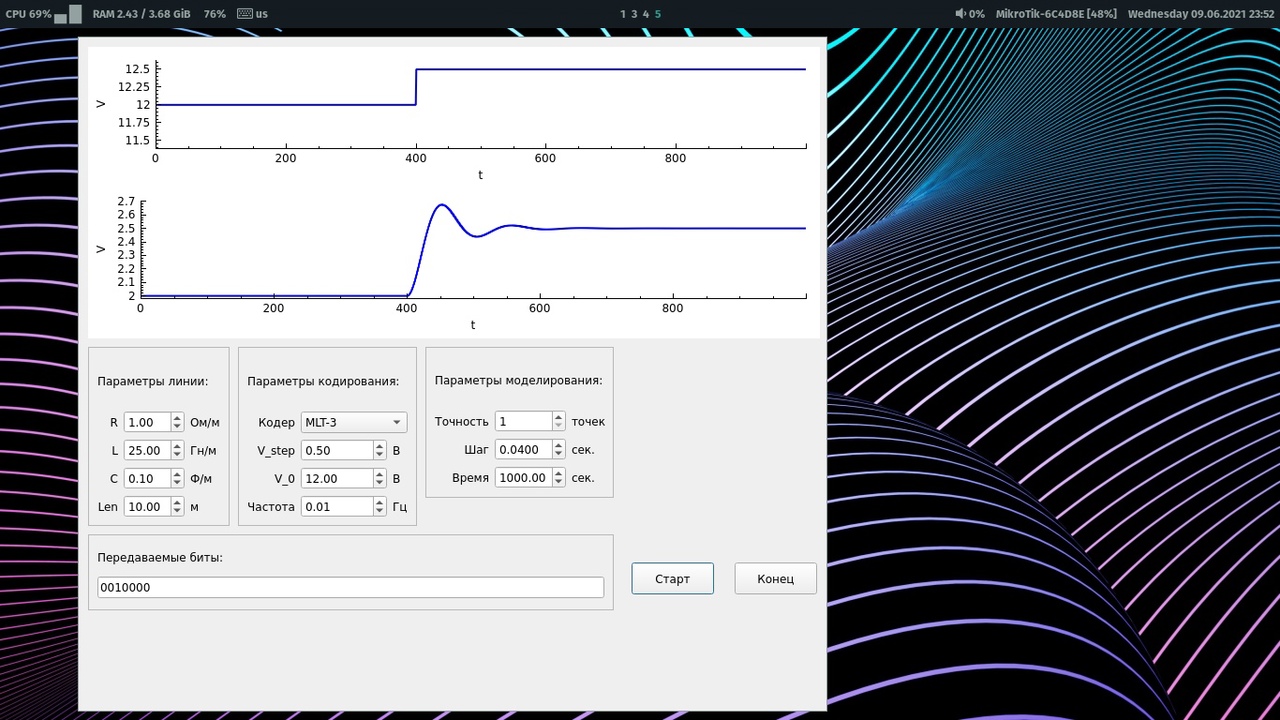


Рисунок 4.3 – Результат для одного четырехполюсника

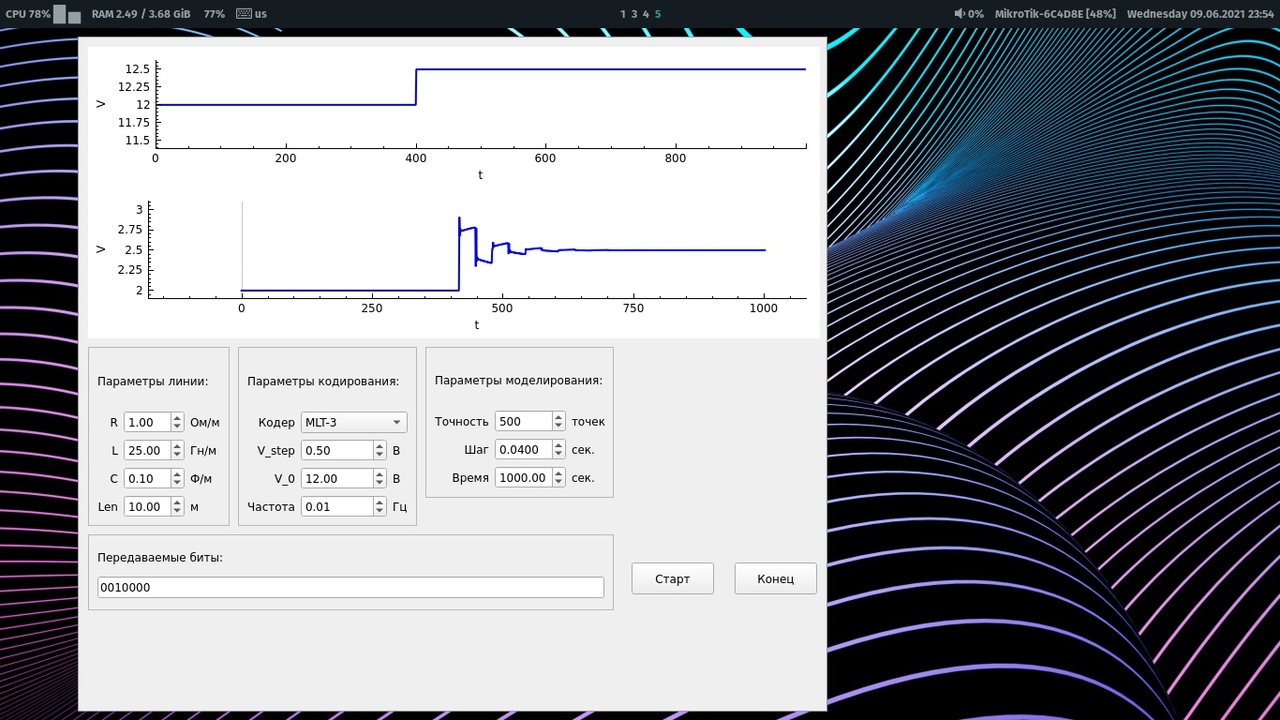


Рисунок 4.4 – Результат для пятисот четырехполюсников

# 5 Безопасность и экологичность выпускной квалификационной работы

## 5.1 Анализ вредных и опасных факторов

Так как выполнение этой работы производится в офисном помещении с использованием ПК, на работников могут оказывать неблагоприятное воздействие следующие опасные и вредные производственные факторы:

* повышенная или пониженная яркость света;
* повышенная или пониженная температура;
* повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека;
* статические перегрузки костно-мышечного аппарата и динамические локальные перегрузки мышц кистей рук;
* перенапряжение зрительного анализатора;
* умственное перенапряжение;
* эмоциональные перегрузки;
* монотонность труда.

Учитывая эти факторы, предлагаются следующие мероприятия по улучшение условию труда:

* проведение специальной оценки условий труда (СОУТ);
* организация обучения и проверки знаний по охране труда работников;
* проведение обязательных медицинских осмотров и психиатрических освидетельствований;
* приведение уровней естественного и искусственного освещения на рабочих местах, в бытовых помещениях, местах прохода работников в соответствие с действующими нормами;
* устройство новых и (или) реконструкция имеющихся мест организованного отдыха, помещений и комнат релаксации, психологической разгрузки;
* обучение лиц, ответственных за эксплуатацию опасных источников электроэнергии;
* оборудование по установленным нормам помещения для оказания медицинской помощи и (или) создание санитарных постов с аптечками, укомплектованными набором лекарственных средств и препаратов для оказания первой помощи;
* организация и проведение производственного контроля;
* издание (тиражирование) инструкций по охране труда.

## 5.2 Оценка экологичности работы

Каждое предприятие вынуждено обеспечивать экологическую безопасность во избежание внереализационных расходов, связанных с визитами государственных инстанций, а также для предотвращения несчастных случаев на производстве, влекущих издержки, связанные с расследованиями и компенсациями.

Экологическая безопасность в РФ обеспечивается следующими правовыми актами: законом «Об охране окружающей среды» и ГОСТом Р ИСО 14000. Этот ГОСТ является аналогом оригинального стандарта ISO 14000.

Выпускная квалификационная работа выполнялась в офисе за персональным компьютером.

Конструкция дисплея должна обеспечивать возможность фронтального наблюдения экрана путем поворота корпуса в горизонтальной плоскости вокруг вертикальной оси с фиксацией в заданном положении. Дизайн мониторов должен предусматривать окраску в спокойные мягкие тона с диффузным рассеиванием света. Корпус монитора, клавиатура должны иметь матовую поверхность одного цвета с коэффициентом отражения 0,4 - 0,6 и не иметь блестящих деталей, способных создавать блики.

Дисплей должен обеспечивать возможность регулировки яркости, контрастности, разрешения от минимальных до максимальных значений.

Таким образом при следовании рекомендациям, можно избежать возможных проблем со здоровьем и выжать из сотрудника максимальный КПД.

## 5.3 Устойчивость к чрезвычайным ситуациям. Пожаробезопасность

Ввиду того, что работа проводится в офисе, существенным требованием является пожаробезопасность.

Реализация требований пожаробезопасности, предъявляемых к рабочему пространству, существенно повышает уровень защищенности работника. Обозначаются они в распорядительном документе, разработанном ответственным за пожарную безопасность лицом и утвержденном руководителем организации.

Основными документами, регулирующими пожарную безопасность в офисе, являются Федеральный закон № 123-ФЗ от 22 июля 2008 г. «Технический регламент о требованиях пожарной безопасности» и Правила противопожарного режима Российской Федерации, утвержденные постановлением Правительства РФ от 25 июля 2012 г. № 390 «О противопожарном режиме».

На любом предприятии должен быть документ, в соответствии с которым организуется пространство помещения и устанавливается противопожарный режим. Его основные положения включают:

* наличие табличек с номером телефона вызова пожарной охраны и фамилиями ответственных за противопожарное состояние помещений. За размещение таких табличек несет ответственность руководитель организации;
* наличие планов эвакуации при пожаре. План должен быть разработан для каждого этажа офисного здания. Копии всех поэтажных планов используются для создания сводного плана, хранящегося у ответственного лица. В случае если офис находится в небольшом арендованном помещении в бизнес-центре, изготавливается локальный план эвакуации;
* наличие знаков пожарной безопасности в офисе. К ним относятся информационные стенды и плакаты по пожарной безопасности и информационные таблички нескольких видов;
* наличие первичных средств пожаротушения в офисе. Это средства, которые эффективны на начальной стадии возгорания;
* доступ к эвакуационным выходам и путям эвакуации. Одной из действенных мер пожарной безопасности в офисе является обеспечение свободного доступа к эвакуационным путям и выходам.

Наличие и регулярная проверка огнетушителей, плана эвакуации — в случае пожара помогут избежать материального ущерба и человеческих жертв.

## 5.4 Расчет системы искусственного освещения помещений

Необходимо произвести расчет искусственного освещения, выполненного светильниками ЛСПО1 с одной люминесцентной лампой для помещения, схема помещения изображена на рисунке 5.2, схема освещения на рисунке 5.1, а исходные данные для расчета представлены в таблице 5.1.

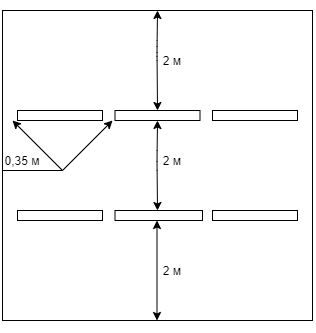


Рисунок 5.1 – Эскиз размещения светильников

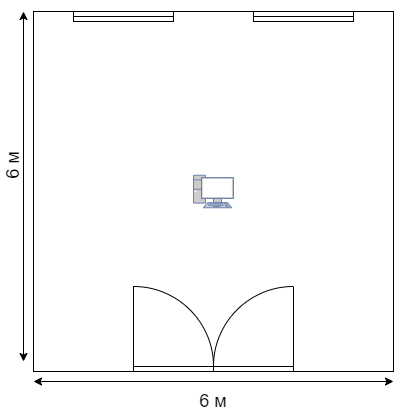


Рисунок 5.2 – Эскиз помещения

Таблица 5.1 – Исходные данные

|  |  |
| --- | --- |
| Наименование | Величина |
| Длина помещения, А, м | 6 |
| Ширина помещения, В, м | 6 |
| Высота помещения, h, м | 4 |
| Коэффициент отражения потолка | 30 |
| Коэффициент отражения стен | 10 |
| Коэффициент отражения пола | 10 |
| Нормируемая освещенность, , лк | 300 |
| Число ламп в светильнике, n, шт. | 2 |
| Длина ламп, , м | 1,534 |
| Коэффициент запаса, К, | 1,5 |
| Коэффициент использования светового потока, | 48 |

Высота подвеса светильника над рабочей поверхностью вычисляется по формуле 5.1.

, (5.1)

где - высота рабочей поверхности.

м.

Найдём индекс помещения по формуле 5.2:

, (5.2)

Необходимо определить оптимальное расстояние между рядами светильников по формуле 5.3:

, (5.3)

где L – оптимальное расстояние между рядами светильников,

λ=1,1 – коэффициент оптимального расстояния между светильниками.

м.

Определим общее количество светильников, а также отдельно количество светильников по ширине и во длине:

, (5.4)

, (5.5)

, (5.6)

шт.

шт.

Определим световой поток одной лампы по следующей формуле:

, (5.7)

, лм

Также определим расчетный световой поток:

, (5.8)

где – расчетный световой поток, лк.

, лм

Для определения соответствия проектируемой системы освещения требованиям СНиП 23-05-95 выполним проверочный расчет по следующей формуле:

, (5.9)

, лк

Так как , то освещение удовлетворяет требованиям.

Определим потребляемую мощность осветительной установки:

, (5.10)

, Вт

где p – мощность лампы, Вт,

- коэффициент, учитывающий потери пускорегулирующей аппаратуры. Для ЛЛ .

## 5.5 Вывод по главе

В данной главе было рассчитана система искусственного освещения помещения, а также выявлено, что освещение удовлетворяет требованиям СНиП 23-05-95.

Построен эскиз размещения источников освещения помещения с указанием расстояний от стен по длине, ширине, а также между рядами светильников.

Также в главе были описаны меры, предлагаемые для обеспечения безопасной работы.

# 6 Экономическая часть выпускной квалификационной работы

## 6.1 Основные аспекты реализации проекта

Целью работы является создание программного стенда моделирования передачи информации на физическом уровне.

Актуальность разработки программного стенда заключается в предоставлении удобного средства для изучения явлений, происходящих в физическом канале связи с различными свойствами.

Программный стенд моделирования передачи информации на физическом уровне предназначен для изучения свойств физического кодирования, а также переходных процессов, происходящих в линии. Он должен предоставлять пользователю возможность ввода погонных параметров, шага моделирования, свойств кодирования, а также выводить результаты в виде графиков.

К программному стенду предъявляются следующие требования:

* он должен обеспечивать надежность выполнения операций;
* дизайн интерфейса должен обеспечивать удобство для работы пользователя;
* при применении стенда не должно быть специальных требований к пользователям программного средства, ими могут быть любые пользователи, имеющие общие знания вычислительной техники и начальные знания физики, электротехники.

## 6.2 Стратегический маркетинговый анализ целесообразности применения программного средства

Для определения целесообразности применения программного стенда позиций стратегического маркетинга необходимо провести SWOT, PEST и SNW анализы.

SWOT - метод анализа заключается в разделении факторов и явлений на четыре категории: возможности, угрозы, сильные и слабые стороны. SWOT-анализ может быть проведен с использованием стратегических определяющих, приведенных в таблице 6.1.

Таблица 6.1 – SWOT-анализ возможности внедрения

|  |  |
| --- | --- |
| Возможности | Угрозы |
| 1 Внедрение современных методов моделирования  2 Визуальное отображение начальных параметров и результатов расчетов | 1 Отрицательный результат от применения программного средства в случае расхождения свойств модели и реального объекта |
| Сильные стороны | Слабые стороны |
| 1 Экономия материальных и временных ресурсов в решении схожих задач  2 Удобный интерфейс | 1 Узкая специализированная направленность программного средства |

Для оптимального представления результатов SWOT-анализа определяется взвешенная бальная оценка факторов SWOT, приведенная в таблице 6.2 и позволяющая количественно оценить значения элементов SWOT.

Таблица 6.2 – Взвешенная бальная оценка факторов SWOT

|  |  |  |  |
| --- | --- | --- | --- |
| Факторы | Экспертная оценка важности фактора  (по пятибальной шкале) | Коэффициент весомости фактора (сумма  равна 1) | Взвешенная оценка |
| Возможности | | | |
| 1 Внедрение современных методов моделирования | 4 | 0,8 | 3,2 |

Продолжение таблицы 6.2

|  |  |  |  |
| --- | --- | --- | --- |
| Факторы | Экспертная оценка важности фактора  (по пятибальной шкале) | Коэффициент весомости фактора (сумма  равна 1) | Взвешенная оценка |
| Возможности | | | |
| 2 Визуальное отображение начальных параметров и результатов расчетов | 5 | 0,2 | 1 |
| Итого: | - | 1 | Σ=4,2 |
| Угрозы | | | |
| 1 Отрицательный результат от применения программного средства в случае расхождения свойств модели и реального объекта | 2 | 1 | 2 |
| Итого: | - | 1 | Σ=2 |
| Сильные стороны | | | |
| 1 Экономия материальных и временных ресурсов в решении схожих задач | 5 | 0,9 | 4,5 |
| 2 Удобный интерфейс | 4 | 0,1 | 0,4 |
| Итого: | - | 1 | Σ=4,9 |
| Слабые стороны | | | |
| 1 Узкая специализированная направленность программного средства | 4 | 1 | 4 |
| Итого: | - | 1 | Σ=4 |

В результате проведенного SWOT-анализа было выявлено, что слабая сторона является достаточно весомой, поэтому можно предложить создание модульной архитектуры программного кода, поддержка и сопровождение которой позволят при минимальных затратах дорабатывать программный стенд. Также можно добавить другие видов физических каналов связи. Сильные стороны и возможности имеют значения, большие чем слабые стороны и угрозы соответственно, что предполагает успешное внедрение программного стенда моделирования передачи информации на физическом уровне.

PEST - метод анализа заключается в выявлении политических, экономических, социальных и технологических аспектов внешней среды, которые влияют на использование программного средства. PEST-анализ может быть проведен с использованием стратегических определяющих, приведенных в таблице 6.3.

Таблица 6.3 – PEST-анализ возможности внедрения

|  |  |
| --- | --- |
| Политические факторы | Влияние экономики |
| 1 Никакие политические факторы не влияют на внедрение | 1 Финансовые ресурсы  2 Материальные ресурсы |
| Социокультурные тенденции | Технологические инновации |
| 1 Имидж технологии | 1 Развитие технологий  2 Применение инновационных методов совместно с традиционными |

В целях определения степени влияния PEST-факторов на процесс внедрения программного средства составляется матрица влияния факторов и ответной реакции на них, приведенная в таблице 6.4.

Таблица 6.4 – матрица влияния и ответной реакции на PEST -факторы

|  |  |  |  |
| --- | --- | --- | --- |
| Факторы | Степень и временные горизонты влияния | Характер влияния фактора | Ответная реакция на влияние фактора |
| Политические факторы | Незначительное, долгосрочное | Не влияет | Нет |
| Влияние экономики | Сильное, долгосрочное | Развитие проводных линий связи увеличивает потребности в специалистах | Возможность применения программного средства для учебных целей |
| Социокультурные тенденции | Незначительное, краткосрочное | Применение программного средства зависит от уровня образованности населения | Наличие понятной документации и простого интерфейса |
| Технологические инновации | Среднее, краткосрочное | Внедрение новых технологий приведет к необходимости доработки программного средства | Возможность доработки программного средства с минимальными дополнительными затратами |

В результате проведенного PEST-анализа было определено, что политические факторы не оказывают влияние на внедрение программного средства, социокультурные факторы оказывают незначительное влияние, а экономические и технологические факторы должны положительно повлиять на результат внедрения данного программного средства.

SNW-анализ - это анализ сильных, нейтральных и слабых сторон организации. Обычно SNW-анализ применяют для более глубокого изучения внутренней среды организации после проведения SWOT-анализа. Необходимость проведения SNW-анализа внутренней среды организации обусловлена нецелесообразностью использования результатов анализа внешней маркетинговой среды в отрыве от внутренних факторов, влияющих на возможность внедрения. Пример проведения SNW-анализа представлен в таблице 6.5.

Таблица 6.5 – SNW-анализ внутренней среды организации

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование стратегической позиции | Качественная оценка позиций | | |
| S  сильная | N нейтральная | W  слабая |
| Финансовые ресурсы | × |  |  |
| Мотивация персонала |  | × |  |
| Обеспеченность вычислительными средствами и информационными технологиями | × |  |  |
| Осуществление ребрендинга |  |  | × |
| Активность проведения научно-исследовательских и опытно-конструкторских работ | × |  |  |
| Материально-техническая база | × |  |  |
| Разработка и внедрение инновационных программ |  | × |  |
| Наличие высококвалифицированных специалистов и т.д. |  | × |  |

В результате проведенного SNW-анализа было определено, что из множества стратегических позиций в аспекте определения целесообразности внедрения самую низкую оценку имеет осуществление ребрендинга, а самые высокие – стратегические позиции, связанные с финансовыми и материальными ресурсами.

В результате проведения стратегического маркетингового анализа целесообразности применения информационной системы, были получены результаты трех видов анализа SWOT, PEST, SNW.

По их результатам можно сделать следующие выводы:

* возможно успешное внедрение программного средства для решения специальных задач;
* в перспективе возможно создание более универсального программного средства;
* экономические и технологические факторы могут положительно повлиять на результат внедрения данного программного средства;
* наименьшее влияние на внедрение программного средства оказывают финансовые и материальные ресурсы организации.

## 6.3 Определение продолжительности работ по разработке программного средства

Перед расчетом сроков работ и внедрения, необходимо принять о составе персонала, участвующего в разработке ПС. Для разработки программного стенда средства необходим следующий персонал: руководитель проекта, программист, руководитель по экономической части, руководитель по БЖД.

Для определения продолжительности работ по разработке программного средства составлен перечень работ и определена их продолжительность, которые приведены в таблице 6.6.

Таблица 6.6 – Определение этапов внедрения и их продолжительности

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование этапа | Наименование работ | Длительность работ, час | | | Исполнители |
| t min | t max | T |
| Подготовительный этап | Постановка задачи | 4 | 16 | 9 | Руководитель, программист |
| Теоретический этап | Изучение основ теории электрических цепей, модели линии передачи | 9 | 16 | 12 | Руководитель, программист |
| Изучение модели длинной линии связи | 9 | 16 | 12 | Руководитель, программист |

Продолжение таблицы 6.6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование этапа | Наименование работ | Длительность работ, час | | | Исполнители |
| t min | t max | T |
|  | Изучение и выбор алгоритмов решения СЛАУ и систем ДУ | 9 | 16 | 12 | Руководитель, программист |
| Практический этап | Выбор и поиск инструментов для создания программного стенда | 7 | 10 | 8 | Программист |
| Установка и тестирование инструментов разработки | 12 | 17 | 14 | Программист |
| Этап разработки программного средства | Проектирование программного средства | 16 | 32 | 22 | Программист |
| Создание кода программного стенда | 70 | 90 | 78 | Программист |
| Тестирование программного средства | 9 | 16 | 12 | Программист |
| Этап подготовки основной части | Подготовка отчета | 24 | 32 | 27 | Программист |
| Проверка отчета | 8 | 16 | 11 | Руководитель |
| Доработка отчета | 24 | 32 | 27 | Руководитель,  Программист |
| Дополнительный этап | Подготовка экономической части | 8 | 32 | 18 | Программист |
| Проверка экономической части | 1 | 4 | 2 | Руководитель по экономике |
| Доработка экономической части | 8 | 32 | 18 | Программист, руководитель по экономике |
| Подготовка экологической части | 8 | 32 | 18 | Программист |
| Проверка экологической части | 1 | 4 | 2 | Руководитель по БЖД |
| Дополнительный этап | Доработка экологической части | 8 | 32 | 18 | Программист, руководитель по БЖД |

Продолжение таблицы 6.6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование этапа | Наименование работ | Длительность работ, час | | | Исполнители |
| t min | t max | T |
| Заключительный этап | Оформление отчета | 24 | 32 | 27 | Программист |
| Всего: | | 259 | 477 | 347 |  |

Определение продолжительности работ по разработке программного средства осуществляется на основании данных таблицы 6.6 по формуле 6.1

|  |  |
| --- | --- |
| , | (6.1) |

где – длительность работы,

– наименьшая длительность работы,

– наибольшая длительность работы.

## 6.4 Построение сетевой модели комплекса работ по созданию программного средства

Процесс создания программного средства включает в себя выполнение большого количества работ, которые должны быть увязаны между собой в определенной последовательности по срокам, ресурсам и информационным потокам. Для достижения координации всех работ применяется сетевая модель планирования и управления, которая отражает оптимальное распределение задач по исполнителям. Планирование работ с применением сетевой модели ведется в следующем порядке:

* составляется перечень событий и работ;
* строится сетевой график работ;
* определяется продолжительность работ;
* рассчитываются основные параметры сетевой модели;
* оптимизируется сетевая модель комплекса работ.

Перечень событий и работ по разработке приведен в таблице 6.7.

Таблица 6.7 – Перечень событий и работ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код события | Событие | Код работы | Наименование работы | Исполнители |
| 0 | Сформулирована идея создания программного стенда | 0 – 1 | Постановка задачи | Руководитель,  программист |
| 1 | Разработано задание | 1 – 2 | Изучение основ теории электрических цепей и модели линии передачи | Руководитель |
| 2 | Изучены основы теории электрических цепей, модель линии передачи | 2 – 3 | Изучение модели длинной линии связи | Руководитель, программист, |
| 3 | Изучена модель длинной линии связи | 3 – 4 | Изучение и выбор алгоритмов численного решения СЛАУ и систем ДУ | Руководитель, программист |
| 4 | Изучены алгоритмы решения СЛАУ и систем ДУ | 4 – 5 | Выбор и поиск инструментов для создания программного стенда | Программист |
| 5 | Определены средства разработки | 5 – 6 | Установка и тестирование инструментов разработки | Программист |
| 6 | Установлены средства разработки | 6 – 7 | Проектирование программного средства | Программист |
| 7 | Разработана структура программного средства | 7 – 8 | Создание кода программного стенда | Программист |
| 8 | Создан программный код стенда моделирования передачи информации на физическом уровне | 8 – 9 | Тестирование программного стенда | Программист |
| 9 | Протестирована работоспособность программного стенда | 9 – 10 | Подготовка отчета | Программист |
| 10 | Написан предварительный вариант основной части отчета | 10 – 11 | Подготовка экологической части | Программист |
| 10 – 12 | Проверка отчета | Руководитель |
| 11 | Написан предварительный вариант экологической части отчета | 11 – 12 | Подготовка экономической части | Программист |
| 11 – 13 | Проверка экологической части | Руководитель по БЖД |

Продолжение таблицы 6.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код события | Событие | Код работы | Наименование работы | Исполнители |
| 12 | Написан предварительный вариант экономической части отчета | 12 – 13 | Доработка отчета | Руководитель, программист |
| 12 – 14 | Проверка экономической части | Руководитель по экономике |
| 13 | Доработана основная часть отчета | 13 – 14 | Доработка экологической части | Руководитель по БЖД, программист |
| 14 | Доработана экологическая часть отчета | 14 – 15 | Доработка экономической части | Руководитель по экономике, программист |
| 15 | Доработана экономическая часть отчета | 15 – 16 | Оформление отчета | Программист |

На основании перечня событий и работ построен сетевой график работ, приведенный на рисунке 6.1.

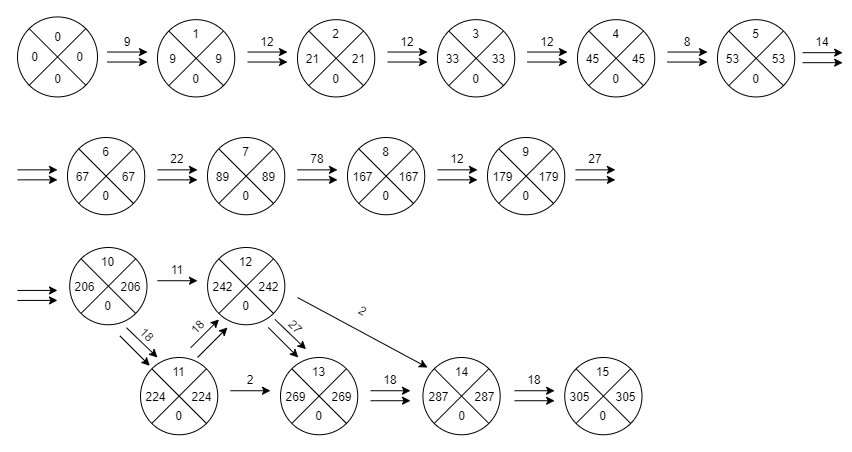


Рисунок 6.1 – Сетевой график работ по разработке программного средства

## 6.5 Оценка потенциальной социально-экономической результативности внедрения программного средства

Оценка потенциальной социально-экономической результативности внедрения программного средства производится с помощью коэффициента, рассчитываемого по формуле 6.2.

|  |  |
| --- | --- |
| , | (6.2) |

где  *-* коэффициент социально-экономической результативности,

*i* – порядковый номер фактора,

*m* – количество факторов,

– коэффициент значимости фактора,

– коэффициент достигнутого уровня.

Также для оценки целесообразности внедрения программного средства необходимо определить коэффициенты для каждого фактора социально-экономической результативности.

Для получения этих значений применяются различные способы определения значимости параметров. Основной особенностью данных параметров является то, что их значимость не связана с отдельными актами принятия решений, а должна отражать общую важность характеристики для итогового результата бизнеса.

По каждому из факторов экспертным путем устанавливаются значения коэффициентов значимости и достигнутого уровня. Сумма коэффициентов значимости должна быть равна 1,0. Чем ближе к 1,0, тем выше потенциальная социально-экономическая результативность внедрения разработки. Факторы, их характеристики и коэффициенты оценки результативности приведены в таблице 6.8.

Таблица 6.8 – Оценка потенциальной социально-экономической результативности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фактор  социально-экономической результативности | Коэффициент значимости фактора, | Качество фактора | Характеристика фактора | Коэффициент достигнутого уровня, |
| Вероятность коммерческой реализации технологии создания и внедрения | 0,35 | Вероятность большая | Разработанная технология является универсальной, может заинтересовать других участников рынка | 1,0 |
| Вероятность умеренная | Технология требует незначительной адаптации для использования другими участниками рынка | 0,6 |
| Вероятность малая | Технология является высокоспециализированной, ориентированной на особенности заказчика | 0,1 |
| Улучшение качества деятельности | 0,35 | Однозначно произойдёт | Внедрение достаточно широкое, охватывает большую часть бизнес-процессов | 1,0 |
| Возможно произойдет | Внедрение в ряде подразделений | 0,6 |
| Вероятность улучшения низкая | Внедрение в формате «пилотного проекта» | 0,1 |
| Фактор  социально-экономической результативности | Коэффициент значимости фактора, | Качество фактора | Характеристика фактора | Коэффициент достигнутого уровня, |
| Вероятность повышения квалификации сотрудников | 0,15 | Вероятность большая | Внедрение однозначно стимулирует повышение квалификации сотрудников | 1,0 |
| Вероятность умеренная | Внедрение вероятно потребует повышение квалификации сотрудников | 0,6 |
| Вероятность малая | Внедрение возможно без повышения квалификации сотрудников | 0,1 |

Продолжение таблицы 6.8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фактор  социально-экономической результативности | Коэффициент значимости фактора, | Качество фактора | Характеристика фактора | Коэффициент достигнутого уровня, |
| Рост престижности организации | 0,15 | Однозначно произойдёт | Внедрение послужит росту престижности организации в связи с внедрением прогрессивных технологий | 1,0 |
| Возможно произойдет | Внедрение подтвердит престижность организации в сфере внедрения прогрессивных технологий | 0,6 |
| Вероятность роста низкая | Внедрение не повлияет на репутации организации из-за узости области применения | 0,1 |

Таблица 6.9 – Коэффициенты потенциальной социально-экономической результативности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фактор  социально-экономической результативности | Коэффициент значимости фактора, | Качество фактора | Характеристика фактора | Коэффициент достигнутого уровня, |
| Вероятность коммерческой реализации технологии | 0,35 | Вероятность умеренная | Технология требует незначительной адаптации для использования другими участниками рынка | 0,6 |
| Улучшение качества деятельности | 0,35 | Возможно произойдет | Внедрение в ряде подразделений | 0,6 |
| Вероятность повышения квалификации | 0,15 | Вероятность умеренная | Внедрение вероятно потребует повышение квалификации сотрудников по узкоспециализированным технологиям | 0,6 |
| Рост престижности организации | 0,15 | Однозначно произойдёт | Внедрение послужит росту престижности организации в связи с внедрением прогрессивных технологий | 1,0 |

Коэффициенты потенциальной социально-экономической результативности внедрения разработанного программного средства приведены в таблице 6.9. Сумма коэффициентов значимости должна быть равна 1.

Рассчитаем коэффициенты для оценки социально-экономической результативности:

Чем ближе к 1, тем выше потенциальная социально-экономическая результативность внедрения разработки. Рассчитанный коэффициент показывает хорошую социально-экономическую результативность внедрения программного средства.

## 6.6 Определение затрат на разработку и внедрение программного средства

Затраты, необходимые на разработку и внедрение программного средства, включают в себя:

* материальные затраты;
* заработная плата сотрудников и страховые взносы;
* стоимость специального оборудования и программного обеспечения;
* амортизация основного оборудования и программного обеспечения;
* накладные расходы.

Материальные затраты, как одна из основных статей расходов любого производства, обычно находятся под пристальным вниманием работников планово-экономического отдела, подразделений, занятых прогнозированием издержек и анализом эффективности деятельности.

В состав материальных затрат включаются стоимость бумаги, канцелярских товаров, справочной литературы, устройств хранения информации, работ и услуг по обслуживанию компьютерной техники, услуг связи и других затрат, связанных с разработкой и отладкой программного средства. Расчет материальных затрат приведен в таблице 6.10.

Таблица 6.10 – Материальные затраты, связанные с созданием и внедрением программного средства

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование материалов и ресурсов | Единица измерения | Количество | Цена за единицу, руб. | Сумма, руб. |
| Бумага писчая | Пачка | 2 | 250 | 500 |
| Канцтовары | Комплект | 1 | 1000 | 1000 |
| USB-накопители | Шт. | 2 | 200 | 400 |
| Заправка картриджа | - | - | - | 400 |
| Услуги связи и интернета | - | - | - | 600 |
| Прочие | - | - | - | 2000 |
| Всего: | | | | 4900 |

Заработная плата сотрудников включает затраты на постановку задачи, изучение предметной области, разработку алгоритмов, программную реализацию, отладку программного средства и другие виды работ, выполняемые задействованными сотрудниками. Заработная плата рассчитывается на основании месячного должностного оклада сотрудника. Расчет заработной платы приведен в таблице 6.11.

Таблица6.11 – Заработная плата сотрудников, связанных с созданием и внедрением программного средства

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Коды работ | Должность сотрудника | Количество отработанных часов | Часовая ставка, руб. | Сумма, руб. |
| 0-1, 1-2, 2-3, 3-4, 10-12, 12-13 | Руководитель | 83 | 284 | 23572 |
| 0–1, 1-2, 2-3, 3-4, 4-5, 5-6, 6-7, 7-8, 8-9, 9-10, 10-11, 11-12, 12-13, 13-14, 14-15, 15-16 | Программист | 343 | 236 | 80948 |
| 11-13, 13-14 | Руководитель по БЖД | 20 | 256 | 5120 |
| 12-14, 14-15 | Руководитель по экономической части | 20 | 256 | 5120 |
| ВСЕГО: | | | | 114760 |

Страховые взносы в государственные внебюджетные фонды определяются в процентном отношении от суммы заработной платы (30%) согласно законодательству.

Расчет страховых взносов: (руб.).

Стоимость специального оборудования и программного обеспечения рассчитывается только в случае, когда имеется необходимость в его приобретении. При выборе программного обеспечения для разработки программного средства были выбраны свободно распространяемые программные продукты. Приобретение специального оборудования не требуется.

Амортизация основного оборудования и программного обеспечения рассчитывается, если используется уже имеющееся оборудование и программное обеспечение. Амортизация определяется исходя из стоимости задействованного в разработке оборудования и программного обеспечения, приведенной в таблице 6.12, и рассчитанной длительности разработки.

Таблица6.12 – Стоимость основного оборудования и программного обеспечения

|  |  |  |
| --- | --- | --- |
| Наименование оборудования, программного обеспечения | Количество | Сумма, руб. |
| Компьютер в сборе | 1 | 50000 |
| Установленное программное обеспечение | 3 | 20000 |
| ВСЕГО: | | 70000 |

Амортизация основного оборудования и программного обеспечения рассчитывается на основании данных, приведенных в таблице 6.13.

Таблица 6.13 – Амортизационные отчисления за год

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование оборудования, программного обеспечения | Общая первоначальная стоимость, руб. | Годовая норма амортизации, % | Сумма годовой амортизации, руб. |
| Компьютер в сборе | 50000 | 20 | 10000 |
| Установленное программное обеспечение | 20000 | 20 | 4000 |
| ВСЕГО: | | | 14000 |

Для расчета суммы амортизации основного оборудования и программного обеспечения за срок разработки программного средства применяется формула 6.3.

|  |  |
| --- | --- |
| , | (6.3) |

где – сумма амортизации за срок разработки,

– сумма годовой амортизации,

– обще количество рабочих часов в году.

Рассчитаем амортизацию для разработки программного средства:

= 2165 (руб.)

Накладные расходы, в состав которых включают расходы по содержанию зданий, коммунальные расходы, представительские и другие расходы, определяются в процентах от заработной платы сотрудников.

При проценте накладных расходов 25% рассчитаем сумму накладных расходов: (руб.)

Общая себестоимость разработки программного средства рассчитана и приведена в таблице 6.14.

Таблица 6.14 – Сводная таблица себестоимости проекта

|  |  |
| --- | --- |
| Статьи расходов | Сумма, руб. |
| Материальные затраты | 4900 |
| Заработная плата | 114760 |
| Страховые взносы | 34428 |
| Амортизация | 2165 |
| Накладные расходы | 28690 |
| Итого - стоимость разработки: | 184943 |

## 6.7 Определение экономической эффективности создания и внедрения программного средства

Экономическая эффективность проекта зависит от вида эффекта, получаемого при внедрении программного средства. Рассчитаем экономическую эффективность, получаемую за счет интенсификации труда сотрудников - роста производительности труда.

В результате внедрения программного средства могут быть автоматизированы процессы и операции, которые до этого выполнялись вручную, что приведет к сокращению времени на выполнение аналогичной работы.

Если пользователь при выполнении работы с использованием программного средства сэкономит рабочее время, то повышение производительности труда определяется по формуле 6.4.

|  |  |
| --- | --- |
| , | (6.4) |

где – коэффициент повышения производительности труда,

– сэкономленное время,

– время, требующееся для выполнения работы до внедрения.

Разработанное программное средство позволяет выполнять работы со значительной экономией рабочего времени. Рассчитаем коэффициент производительности труда, если сэкономленное время составляет 78 ч. в месяц, а среднемесячное рабочее время: 164 ч.:

.

Если пользователь применяет программное средство при выполнении одного вида работ, тогда дополнительный эффект в виде годовой условной экономии, связанной с повышением производительности труда, определяется по формуле 6.5.

|  |  |
| --- | --- |
| , | (6.5) |

где – эффект годовой условной экономии,

– средняя годовая заработная плата пользователя.

Если программное средство используют различные категории пользователей, то расчеты следует выполнять отдельно для каждой категории. Рассчитаем условную экономию при использовании программного средства пользователями одной категории при выполнении одного вида работ, если среднемесячная зарплата пользователя 34000 руб.:

(руб.)

Расчет годового экономического эффекта производится по формуле 6.6.

|  |  |
| --- | --- |
| , | (6.6) |

где – годовой экономический эффект, (руб.),

– нормативный коэффициент экономической эффективности, равный 0,25,

– затраты, связанные с разработкой программного средства, (руб.).

Рассчитаем годовой экономический эффект:

(руб.)

Коэффициент сравнительной эффективности рассчитывается по формуле 6.7.

|  |  |
| --- | --- |
| , | (6.7) |

Рассчитаем коэффициент сравнительной эффективности для разработанного программного средства:

Срок окупаемости проекта определяется по формуле (6.8):

|  |  |
| --- | --- |
| , | (6.8) |

Рассчитаем срок окупаемости разработанного проекта:

.49

Из полученных результатов можно сделать вывод об экономической эффективности создания и внедрения программного средства, так как;

* годовой экономический эффект больше нуля ( > 0);
* коэффициент сравнительной эффективности выше нормативного (2,00 > 0,25);
* срок окупаемости не превышает четырех лет (0,49 < 4).

## 6.8 Выводы по разделу

В данном разделе ВКР описаны основные аспекты реализации проекта, приведены характеристики программного средства с позиций маркетинга, проведен стратегический анализ целесообразности применения SWOT, PEST и SNW методами, определена продолжительность работ по разработке с построением сетевой модели, проведена оценка потенциальной социально-экономической результативности внедрения.

Также в данном разделе определены затраты на разработку и внедрение программного средства и сделан вывод об их экономической эффективности.

Заключение

В ходе выполнения выпускной квалификационной работы были решены следующие задачи.

Проведен обзор предметной области, построена математическая модель длинной линии связи, выбраны необходимые инструменты для создания программного средства.

Обоснован выбор программного инструментария. В качестве языка программирования был выбран C++, для создания графического интерфейса была выбрана библиотека Qt, для решения систем линейных алгебраических уравнений – библиотека Eigen, для решения системы дифференциальных уравнений – библоитека boost::odeint.

Проведено конструирование программного стенда моделирования передачи информации на физическом уровне, отвечающее всем требованиям технического задания.

Разработана документация на программное средство, обоснована безопасность и экологичность, экономическая целесообразность реализуемого продукта.

Перечень использованных информационных ресурсов

1. Официальная документация по фреймворку Qt [Электронный ресурс], URL: https://doc.qt.io/qt-6/reference-overview.html (дата обращения: 15.05.2021).
2. Официальная документация по библиотеке Eigen [Электронный ресурс], URL: https://eigen.tuxfamily.org/dox/index.html (дата обращения: 15.05.2021).
3. Официальная документация по библиотеке boost::odeint [Электронный ресурс] URL: <https://www.boost.org/doc/libs/1_66_0/libs/numeric/odeint/doc/html/>index.html (дата обращения: 15.05.2021).
4. Peter Van Weert, Marc Gregoire «C++17 Standard Library Quick Reference: A Pocket Guide to Data Structures, Algorithms, and Functions 2nd ed. Edition», «Apress», 2019.
5. В.П. Попов «Основы теории цепей», ФГУП «Издательство «Высшая школа», 2003.
6. David H. Staelin «Electromagnetics and Applications», «MIT OpenCourseWare», Spring 2009.
7. Картавченко А.Е., Тархов Д.А «Сравнение методов построения приближенных аналитических решений дифференциальных уравнений на примере элементарных функций», Санкт-Петербургский политехнический университет Петра Великого, г. Санкт-Петербург, Россия, 2017.
8. E. Hairer, S.P. Nørsett, G. Wanner «Solving Ordinary Differential Equations I», Springer, Second Revised Edition, 1987.
9. John B. Anderson Б.В «Digital Transmission engineering», Lund University, 2005.
10. Статья по методам кодирования [Электронный ресурс] URL: https://lanberry.ru/lan/standarti\_otkritih\_kabeljnih\_sistem (дата обращения: 15.05.2021).

Приложение А Техническое задание на программное средство

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Доцент кафедры «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_ Н.В. Кудинов  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. | УТВЕРЖДЕНО  Зав. кафедрой «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_ В.В. Долгов  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2021 г. |

## А.1. Общие сведения

### А.1.1 Имя программы

«PLDTM»

### А.1.2 Полное наименование ПС

Программный стенд моделирования передачи информации на физическом уровне «PLDTM»

### А.1.3 Перечень документов, на основании которых создается система

Основанием для разработки является задание к выпускной квалификационной работе, согласованное с руководителем ВКР доцентом кафедры «ПОВТиАС» Кудиновым Никитой Валерьевичем с одной стороны, студентом гр. ВПР41 Бачуриным Данилой Дмитриевичем, именуемым в дальнейшем исполнителем с другой стороны, утвержденному заведующим кафедрой «ПОВТиАС» Долговым Василием Валерьевичем.

### А.1.4 Порядок оформления и предъявления заказчику результатов работ по созданию системы

Система передается в виде функционирующего комплекса на базе средств вычислительной техники Заказчика и Исполнителя. Приемка системы осуществляется комиссией в составе уполномоченных представителей Заказчика и Исполнителя.

### А.1.5 Перечень нормативно–технических документов, методических материалов, использованных при разработке ТЗ

При разработке автоматизированной системы и создании проектно–эксплуатационной документации Исполнитель должен руководствоваться требованиями следующих нормативных документов:

ГОСТ 19.201–78. Техническое задание. Требования к содержанию и оформлению.

ГОСТ 34.601–90. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания;

ГОСТ 34.201–89. Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплексность и обозначение документов при создании автоматизированных систем;

РД 50–34.698–90. Методические указания. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов.

## А.2 Назначение программного средства

Функциональным назначением программного средства является изучение свойств линий передачи информации. Эксплуатироваться программное средство будет в учебной организации.

## А.3.Требования к программному средству

### А.3.1 Требования к программе в целом

#### А.3.1.1 Требования к дизайну графического интерфейса ПС

Дизайна интерфейса ПС находится в ответственности Исполнителя.

#### А.3.1.2 Требования к шрифтовому оформлению ПС

Основным шрифтом должен стать Times New Roman*.*

Размер (кегль) шрифтов должен обеспечивать удобство восприятия текста при минимально допустимом размере экрана.

#### А.3.1.3 Требования к средствам выполнения ПС

Программа должна корректно выполняться в следующих ОС:

* Linux (версия 2.6 и выше);
* Windows 10;
* macOS (версия 10.0 и выше).

#### А.3.1.4 Требования к архитектуре ПС

ПС должна представляться в виде исходных кодов, функции главного окна, модулей физического кодирования и моделирования, должны был разделены на классы.

#### А.3.1.5 Язык интерфейса программы

Русский.

### А.3.2 Требования к функциям (возможностям), выполняемых программой

#### А.3.2.1 Основные требования

Программа должна представлять собой исходные коды, размещенные на физическом накопителе.

## A.4 Требования к программе или программному изделию

### А.4.1 Требования к функциональным характеристикам

Программное средство должно осуществлять следующие функции:

* предоставлять пользователю форму для ввода параметров линии передачи, кодера, моделирования;
* вывод результата моделирования в виде графиков функций входного напряжения и выходного, зависящих от времени.

### А.4.2 Требования к надежности

Надежное функционирование программы должно быть обеспечено выполнением совокупности нижеописанных мероприятий:

* выполнение требований ГОСТ 51188-98;
* защита информации;
* организацией бесперебойного питания технических средств;
* использованием лицензионного программного обеспечения.

### А.4.3 Условия эксплуатации

Для функционирования программного продукта необходимо соблюдение всех требований и правил эксплуатации компьютерной техники.

Высокая квалификация пользователя программного средства не требуется. Дополнительных требований и ограничений не вводится.

Требования к персоналу, работающему с данным программным продуктом – общие знания вычислительной техники.

### А.4.4 Требования к составу и параметрам технических средств

### В состав технических средств должен входить IВМ-совместимый персональный компьютер (ПЭВМ), выполняющий роль сервера, включающий в себя:

### процессор Pentium - 2.0 GHz, не менее;

### оперативную память объемом, 2 Гигабайт, не менее;

* ПЗУ объемом не менее 256 Гигабайт, со свободным дисковым пространством не менее 100Мегабайт.

### А.4.5 Требования к информационной и программной совместимости

#### А.4.5.1 Язык программирования

Для разработки программного средства должен быть использован язык программирования С++.

#### А.4.5.2 Операционная система

Программное средство должно работать под управлением любой ОС из списка:

* Linux (версия 2.6 и выше);
* Windows 10;
* macOS (версия 10.0 и выше).

### А.4.6 Требования к упаковке и маркировке

Требования к упаковке и маркировке программного средства не предъявляется.

### А.4.7 Требования к транспортировке и хранению

Условия транспортирования, места хранения, условия складирования и сроки хранения в различных условиях должны соответствовать требованиям, предъявляемым к носителям информации, на которых будет содержаться данное программное изделие.

Допустимы все способы транспортирования и хранения, не нарушающие целостность используемого носителя данных. Программное средство может храниться на любом носителе информации, имеющее возможность подключения к персональному компьютеру.

### А.4.8 Специальные требования

Специальных требований не предъявляется.

## А.5 Требования к программной документации

Программная документация должна состоять из следующих листов:

* титульный лист;
* пояснительная записка к производственной практике;
* техническое задание по ГОСТ 19.201-78 ЕСПД;
* исходный код программного средства по ГОСТ 19.401-79 ЕСПД.

## А.6 Стадии и этапы разработки

* постановка задачи (с 20.04.21 по 22.04.21);
* изучение предметной области (с 23.04.21 по 26.04.21);
* разработка алгоритмов решения задачи (с 27.04.21 по 02.05.21);
* разработка программы (с 04.05.21 по 11.05.21);
* тестирование программы (с 11.05.21 по 14.05.21).

## А.7 Порядок контроля и приемки

Порядок и контроль приёмки определяются заведующим кафедрой «ПОВТиАС» и основаны на демонстрации знаний технологии и умении создавать программные средства для различных предметных областей.

Главным требованием к приемке является наличие правильно работающего программного средства с тестовым примером и отчета, представленного в печатном виде.

Разработчик технического задания /Бачурин Д.Д./

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Приложение Б Исходный код программного средства

Листинг Б.1 – Заголовочный файл кодеров

#ifndef ENCODER\_H

#define ENCODER\_H

#include <vector>

#include <string>

using namespace std;

class Encoder {

public:

virtual vector<double> encode(const string& s) = 0;

virtual ~Encoder();

};

class RZEncoder : public Encoder {

private:

double uPos;

double uNeg;

double zero;

size\_t halfCnt;

public:

vector<double> encode(const string& s) override;

RZEncoder(double zero, double step, size\_t halfCnt) : uPos(zero+step), uNeg(zero-step), zero(zero), halfCnt(halfCnt) {}

};

class MLT3Encoder : public Encoder {

private:

double uPos;

double uNeg;

double zero;

size\_t tactCnt;

public:

vector<double> encode(const string& s) override;

MLT3Encoder(double zero, double step, size\_t tactCnt) : uPos(zero+step), uNeg(zero-step), zero(zero), tactCnt(tactCnt) {}

};

class ManchesterEncoder : public Encoder {

private:

double uPos;

double zero;

size\_t halfCnt;

public:

vector<double> encode(const string& s) override;

ManchesterEncoder(double zero, double step, size\_t halfCnt) : uPos(zero+step), zero(zero), halfCnt(halfCnt) {}

};

#endif // ENCODER\_H

Листинг Б.2 – Исходный код кодеров

#include "encoder.h"

Encoder::~Encoder() {};

vector<double> RZEncoder::encode(const string &s) {

auto res = vector<double>();

for (auto c : s) {

double val = c == '0' ? uPos : uNeg;

for (int i = 0; i < halfCnt; i++) {

res.push\_back(val);

}

for (int i = 0; i < halfCnt; i++) {

res.push\_back(zero);

}

}

return res;

};

vector<double> MLT3Encoder::encode(const string &s) {

auto res = vector<double>();

bool isInc = true;

double val = zero;

for (auto c : s) {

if (c != '0') {

if (isInc) {

if (val == uPos) {

val = zero;

isInc = false;

} else if (val == zero) {

val = uPos;

} else {

val = zero;

}

} else {

if (val == uNeg) {

val = zero;

isInc = true;

} else if (val == zero) {

val = uNeg;

} else {

val = zero;

}

}

}

for (int i = 0; i < tactCnt; i++) {

res.push\_back(val);

}

}

return res;

};

vector<double> ManchesterEncoder::encode(const string &s) {

auto res = vector<double>();

double fVal;

double sVal;

for (auto c : s) {

if (c == '0') {

fVal = uPos;

sVal = zero;

} else {

fVal = zero;

sVal = uPos;

}

for (int i = 0; i < halfCnt; i++) {

res.push\_back(fVal);

}

for (int i = 0; i < halfCnt; i++) {

res.push\_back(sVal);

}

}

return res;

};

Листинг Б.3 – Заголовочный файл расчетного модуля моделирования

#ifndef MACHINE\_H

#define MACHINE\_H

#include <array>

#include <iostream>

#include <eigen3/Eigen/Dense>

#include "types.h"

#include <thread>

#include <stop\_token>

#include <optional>

#include <atomic>

#include "types.h"

using namespace std;

using namespace Eigen;

class Machine {

    typedef vector<double> state\_type;

    RLC opt;

    int N;

    double RN;

    double NL;

    double NC;

    auto initState(double U\_in, double I\_out);

    public:

        optional<Payload> processNextPayloadStoppable(Payload& payload, reference\_wrapper<atomic\_bool> shouldStop, function<void(int)> cb);

        Machine(int amount, const RLC& lOptions);

};

#endif // MACHINE\_H

Листинг Б.4 – Исходный код расчетного модуля моделирования

#include <array>

#include <iostream>

#include <eigen3/Eigen/Dense>

#include "types.h"

#include "machine.h"

#include <boost/array.hpp>

#include <boost/numeric/odeint.hpp>

#include <functional>

using namespace std;

using namespace Eigen;

using namespace boost::numeric::odeint;

auto initState(double U\_in, double I\_out) {

vector<double> I0 = vector<double>(N);

vector<double> U0 = vector<double>(N);

for (size\_t i = 0; i < U0.size(); i++) {

I0[i] = I\_out;

U0[i] = U\_in - (i+1)\*RN\*I\_out;

}

}

optional<Payload> Machine::processNextPayloadStoppable(Payload& payload, reference\_wrapper<atomic\_bool> shouldStop, function<void(int)> cb) {

Payload res;

res.tau = payload.tau;

cb(0);

auto [I0, U0] = initState(payload.U[0], payload.I[0]);

state\_type state = vector<double>(2\*N);

boost::numeric::odeint::runge\_kutta4\_classic<state\_type> rk;

for (int i = 0; i < N; i++) {

const int idx = i\*2;

state[idx] = I0[i];

state[idx+1] = U0[i];

}

auto& U\_in = payload.U;

auto& I\_out = payload.I;

double h = payload.tau;

size\_t l = min(U\_in.size(), I\_out.size());

vector<double> I\_last = vector<double>(l);

vector<double> U\_last = vector<double>(l);

size\_t i = 0;

const int lIdx = N\*2 - 1;

int percent = 1;

do {

atomic\_bool& stop = shouldStop.get();

if (stop.load()) {

return nullopt;

}

double progress = (i / (double) l) \* 100;

int p = (int) progress;

if (p != percent - 1) {

cb(progress);

percent = p;

}

const double t = i\*h;

I\_last[i] = state[lIdx-1];

U\_last[i] = state[lIdx];

rk.do\_step(

[&](const state\_type &x, state\_type &dxdt, double t) {

double time = (t/h);

size\_t i = (size\_t)time;

if (N == 1) {

dxdt[0] = (U\_in[i] - RN\*x[0] - x[1])/NL;

dxdt[1] = (x[0] - I\_out[i])/NC;

return;

}

dxdt[0] = (U\_in[i] - RN\*x[0] - x[1])/NL;

dxdt[1] = (x[0] - x[2])/NC;

for (int i = 1; i < N-1; i++) {

const int idx = i\*2;

dxdt[idx] = (x[idx - 1] - RN\*x[idx] - x[idx+1])/NL;

dxdt[idx+1] = (x[idx] - x[idx+2])/NC;

}

const int lIdx = N\*2 - 1;

dxdt[lIdx - 1] = (x[lIdx - 2] - RN\*x[lIdx - 1] - x[lIdx])/NL;

dxdt[lIdx] = (x[lIdx - 1] - I\_out[i])/NC;

},

state,

t,

h

);

i++;

} while(i < l);

res.I = I\_last;

res.U = U\_last;

return res;

}

Machine::Machine(int amount, const RLC& lOptions)

: opt(lOptions),

N(amount)

{

RN = lOptions.R/N;

NL = lOptions.L/N;

NC = lOptions.C/N;

}

Листинг Б.5 – Заголовочный файл главного окна

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "qcustomplot.h"

#include <atomic>

#include <QVector>

#include "encoder.h"

#include "types.h"

using namespace std;

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_startButton\_clicked();

void on\_stopButton\_clicked();

void on\_freqSpinBox\_valueChanged(double arg1);

void on\_bitsLineEdit\_textChanged(const QString &arg1);

private:

Ui::MainWindow \*ui;

atomic\_bool stopped;

void setOscilatorStylePlot(QCustomPlot\* plt);

void clearPlot(QCustomPlot\* plt);

void setLoading(bool);

void showResult(QCustomPlot\* plt, QVector<double> x, QVector<double> y);

Payload getPayload(Encoder\* enc);

void dispatchAppState(bool stopped, bool clearPlots);

void setLoadingPercent(size\_t percent);

void invokeOnWindowThread(function<void()> fun);

void displayVoltageResult(QCustomPlot\* plt, QVector<double> time, vector<double> voltage);

RLC getRLC();

Encoder\* getEncoder();

void setState(int);

};

#endif // MAINWINDOW\_H

Листинг Б.6 – Исходный код главного окна

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QPushButton>

#include <QtConcurrent>

#include "qcustomplot.h"

#include "encoder.h"

#include "machine.h"

#include "types.h"

#include <thread>

#include <atomic>

#include <limits>

void MainWindow::setOscilatorStylePlot(QCustomPlot \*plt) {

//plt->setBackground( QColor(0,0,0) );

QColor gridColor(255, 255, 255);

QPen gridPen;

gridPen.setWidthF(0.5);

gridPen.setColor(gridColor);

gridPen.setStyle(Qt::PenStyle::SolidLine);

plt->xAxis->grid()->setPen(gridPen);

plt->yAxis->grid()->setPen(gridPen);

plt->xAxis->setLabel("t");

plt->yAxis->setLabel("V");

QCPGraph\* graph = plt->addGraph();

QColor graphColor(0, 0, 255);

QPen graphPen;

graphPen.setWidthF(2);

graphPen.setColor(graphColor);

graph->setPen(graphPen);

plt->graph(0)->rescaleAxes(true);

plt->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom | QCP::iSelectPlottables);

//plt->axisRect()->setAutoMargins(QCP::msNone);

//plt->axisRect()->setMargins(QMargins(-5,0,0,0));

}

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

setOscilatorStylePlot(ui->inputPlot);

setOscilatorStylePlot(ui->outputPlot);

dispatchAppState(true, true);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::setLoading(bool state) {

if (state) {

ui->progressBar->setVisible(true);

} else {

ui->progressBar->setVisible(false);

}

ui->progressBar->setValue(0);

}

void MainWindow::setState(int p) {

ui->progressBar->setValue(p);

}

Payload MainWindow::getPayload(Encoder\* enc) {

double T = ui->timeSpinBox->value();

double step = ui->stepSpinBox->value();

double uNull = ui->nullUSpinBox->value();

double R = ui->rSpinBox->value();

double len = ui->lenSpinBox->value();

string bits = ui->bitsLineEdit->text().toStdString();

size\_t tCount = (T/step);

Payload p = {

.tau = step,

.I = vector<double>(tCount),

.U = vector<double>(tCount),

};

vector<double> vals = enc->encode(bits);

double fullR = (R\*len);

//std::fill(p.I.begin(), p.I.end(), I\_out);

for (size\_t i = 0; (i < vals.size()) && (i < tCount); i++) {

p.U[i] = vals[i];

}

//padding

for (size\_t i = vals.size(); i < tCount; i++) {

p.U[i] = uNull;

}

for (size\_t i = 0; i < p.U.size(); i++) {

//нагрузка

p.I[i] = 1;

}

return std::move(p);

}

RLC MainWindow::getRLC() {

double R = ui->rSpinBox->value();

double L = ui->lSpinBox->value();

double C = ui->cSpinBox->value();

double length = ui->lenSpinBox->value();

RLC rlc = {

.R = R\*length,

.L = L\*length,

.C = C\*length,

};

return rlc;

}

Encoder\* MainWindow::getEncoder() {

size\_t codeIndex = ui->codeComboBox->currentIndex();

double frequency = ui->freqSpinBox->value();

double uStep = ui->stepUSpinBox->value();

double uNull = ui->nullUSpinBox->value();

double step = ui->stepSpinBox->value();

//число точек на половину такт(?)

double val = 1/(frequency\*step);

Encoder\* enc;

//mlt-3, rz, manchester

switch (codeIndex) {

case 0:

enc = new MLT3Encoder(uNull, uStep, 2\*val);

break;

case 1:

enc = new RZEncoder(uNull, uStep, val);

break;

case 2:

enc = new ManchesterEncoder(uNull, uStep, val);

break;

//to supress warnings?

default:

enc = new RZEncoder(uNull, uStep, val);

}

return enc;

}

void MainWindow::dispatchAppState(bool stopped, bool clearPlots) {

this->stopped.store(stopped);

setLoading(!stopped);

ui->progressBar->setVisible(!stopped);

ui->status->setVisible(!stopped);

if (stopped && clearPlots) {

clearPlot(ui->inputPlot);

clearPlot(ui->outputPlot);

}

}

void MainWindow::setLoadingPercent(size\_t percent) {

ui->progressBar->setValue(percent);

}

void MainWindow::invokeOnWindowThread(function<void()> fun) {

QMetaObject::invokeMethod(this, fun);

}

void MainWindow::displayVoltageResult(QCustomPlot \*plt, QVector<double> time, vector<double> voltage) {

double maxU = 0;

double minU = std::numeric\_limits<double>::max();

double T = ui->timeSpinBox->value();

QVector<double> V(voltage.size());

for (size\_t i = 0; i < voltage.size(); ++i) {

V[i] = voltage[i];

if (voltage[i] > maxU)

maxU = voltage[i];

if (voltage[i] < minU)

minU = voltage[i];

}

plt->graph(0)->setData(time, V);

plt->xAxis->setRange(0, T);

plt->yAxis->setRange(minU\*0.99, maxU\*1.01);

plt->replot();

}

void MainWindow::on\_startButton\_clicked()

{

bool running = !stopped.load();

if (running)

return;

dispatchAppState(false, true);

auto trd = std::thread([&]() {

Encoder\* enc = getEncoder();

Payload payload = getPayload(enc);

RLC rlc = getRLC();

size\_t circuitCount = ui->accuracySpinBox->value();

Machine machine(circuitCount, rlc);

double T = ui->timeSpinBox->value();

double step = ui->stepSpinBox->value();

optional<Payload> lastOpt = machine.processNextPayloadStoppable(payload, ref(stopped), [=](size\_t percent) {

invokeOnWindowThread([=]() {

if (percent == 0) {

ui->status->setText("Поиск начальных условий..");

} else if (percent == 1) {

ui->status->setText("Моделирование..");

}

setLoadingPercent(percent);

});

});

if (!lastOpt.has\_value()) {

return;

}

Payload last = lastOpt.value();

QVector<double> time(T/step);

for (size\_t i = 0; i < time.size(); i++) {

time[i] = last.tau \* i;

}

delete enc;

invokeOnWindowThread([=]() {

displayVoltageResult(ui->inputPlot, time, payload.U);

displayVoltageResult(ui->outputPlot, time, last.U);

dispatchAppState(true, false);

});

});

trd.detach();

}

void MainWindow::clearPlot(QCustomPlot \*plt) {

plt->graph(0)->data()->clear();

plt->replot();

}

void MainWindow::on\_stopButton\_clicked()

{

dispatchAppState(true, true);

}

void MainWindow::on\_freqSpinBox\_valueChanged(double newValue)

{

double step = ui->stepSpinBox->value();

double maxFreq = 1/step;

if (newValue > maxFreq) {

ui->freqSpinBox->setValue(maxFreq);

}

}

void MainWindow::on\_bitsLineEdit\_textChanged(const QString &arg1)

{

if (arg1.size() == 0) {

return;

}

auto last = arg1[arg1.size() - 1];

auto newStr = arg1;

if (last != "0" && last != "1") {

ui->bitsLineEdit->setText( newStr.remove(newStr.size() - 1, 1) );

}

}

Листинг Б.8 – Исходный код приложения

#include "mainwindow.h"

#include <QApplication>

#include <stdlib.h>

int main(int argc, char\* argv[])

{

QApplication app(argc, argv);

MainWindow w;

w.show();

return app.exec();

}

Приложение В

**Акт приемки/сдачи программного средства**

|  |  |
| --- | --- |
|  | Зав. кафедрой «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Долгов  (подпись) «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021г. |

**Студент Бачурин Данила Дмитриевич**

**Руководитель Кудинов Никита Валерьевич**

**Наименование темы Программный стенд моделирования передачи информации на физическом уровне**

**Перечень подлежащих проверке функций программного обеспечения**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Основные функции, выполняемые программным продуктом | Выполнение | |
| Алгоритм | ПС |
| 1 | Моделирование линии связи посредством решения системы дифференциальных уравнений |  |  |
| 2 | Моделирование кодера физического уровня, формирование сигнала с заданной частотой тактирования |  |  |
| 3 | Отображение графиков зависимости входного и выходного напряжения от времени |  |  |

**Оценка экспертом свойств программного средства**

|  |  |
| --- | --- |
| ***Проверяемые свойства представленного ПО*** | ***Оценка эксперта*** |
| Соответствие заявленных к проверке функций техническому заданию |  |
| Размер программного кода *(В том числе соотношение кода созданного вручную к общему объему кода)* |  |
| Удобство эксплуатации *(Трудоемкость установки программного средства, удобство пользовательского интерфейса, наличие контекстной и полнотекстовой помощи)* |  |
| Сопровождаемость ПО  *(Сложность/простота конструкции, структурированность программного кода, оформление текста программ, наличие комментариев-заголо­вков для модулей и коммен­тариев к основным процедурам и функциям)* |  |
| Надежность ПО  *(Устойчивость функциониро­вания ПО при наличии ошибок во входных данных и ошибочных ситуациях)* |  |
| Уровень владения студентом исходным кодом |  |

Краткий отзыв эксперта на представленное программное обеспечение:

Исходные тексты в полном объеме представлены

Эксперт / А.П. Кузин /

Перечень подлежащих проверке свойств ПС подтверждаю

Руководитель / Н.В. Кудинов /