

Федеральное государственное бюджетное  
образовательное учреждение высшего образования



«Московский государственный технический  
университет имени Н. Э. Баумана (национальный  
исследовательский университет)»



Факультет «Машиностроительные технологии»

Кафедра «Электронные технологии в  
машиностроении»

## ОТЧЁТ

по дисциплине «Анализ и синтез технических решений»

на тему:

«Способы автоматизации вакуумных систем»

Выполнил:

студент группы МТ11-62Б

Зотов М.С.

Руководитель:

Колесник Л.Л.

Дисциплина	Результат защиты (нужное выделить)	
	НЕЗАЧЁТ	ЗАЧЁТ
Научно-исследовательская работа		
Зачёт принял _____ подпись	( _____ ) расшифровка	

Москва

2021 г.

## РЕФЕРАТ

Отчёт по дисциплине «Анализ и синтез технических решений» содержит 60 страниц, 8 таблиц, 27 рисунков, 41 уравнение, 15 листингов, список литературы из 10 источников.

АВТОМАТИЗАЦИЯ, ВАКУУМНАЯ СИСТЕМА, MODBUS, TCP, QT, ПРОЕКТИРОВАНИЕ, МОДЕЛИРОВАНИЕ, СИМУЛЯЦИЯ, ПРОТОКОЛЫ.

Отчёт выполнен с использованием среды L<sup>A</sup>T<sub>E</sub>X

Работа посвящена изучению работы с библиотеками Qt и основам протокола *Modbus*.

В работе описывается создание программы для обмена сигналами между классами, работа в сетях TCP/IP, создание модели нанесения покрытия на цилиндрическую трубку и анализ её методами техники эксперимента. Приведены фрагменты кода и описаны принципы работы программ.

*Целью работы является изучение библиотеки Qt, протокола Modbus и разработка ПО по моделированию поведения частиц методом Монте-Карло.*

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b>	<b>5</b>
<b>1 Протоколы взаимодействия вакуумного оборудования</b>	<b>6</b>
1.1 Модель OSI . . . . .	6
1.2 Протокол Modbus . . . . .	8
1.3 Принцип работы протокола Modbus . . . . .	9
1.3.1 Физический уровень . . . . .	9
1.3.2 Логический уровень . . . . .	9
1.3.2.1 Modbus RTU . . . . .	10
1.3.2.2 Modbus TCP . . . . .	11
<b>2 Составление документации по Modbus на Gitlab</b>	<b>14</b>
2.1 Знакомство с языками разметки документации . . . . .	14
2.1.1 HTML . . . . .	14
2.1.2 Markdown . . . . .	15
2.2 Структурирование и перенос информации . . . . .	16
<b>3 Реализация обмена сообщениями в среде QT</b>	<b>18</b>
3.1 Сигналы и слоты . . . . .	18
3.2 Концепция программы . . . . .	19
3.3 Реализация кода на языке Qt . . . . .	19
3.3.1 Структура класса . . . . .	19
3.3.2 Связь сигналов и слотов . . . . .	21
3.3.3 Графический интерфейс и окно вывода . . . . .	22
<b>4 Создание сервера обмена сообщениями по протоколу TCP</b>	<b>24</b>
4.1 Принцип работы в сетях TCP/IP . . . . .	24
4.2 Принцип работы программы . . . . .	25

4.3	Реализация кода программы . . . . .	25
4.4	Терминал сервера . . . . .	27
<b>5</b>	<b>ПО для моделирования нанесения покрытия</b>	<b>30</b>
5.1	Краткое резюме . . . . .	30
5.2	Расчёт геометрии . . . . .	30
5.2.1	Эскиз . . . . .	31
5.2.2	Граничные условия . . . . .	31
5.2.2.1	Нижние граничные условия . . . . .	32
5.2.2.2	Верхние граничные условия . . . . .	33
5.2.2.3	Нахождение интервала попадания . . . . .	34
5.3	Принцип работы программы . . . . .	35
5.3.1	Преобразование Бокса-Мюллера . . . . .	35
5.3.2	Генерация случайных чисел в библиотеке Qt . . . . .	37
5.3.3	Реализация программы на языке C++ . . . . .	37
5.4	Анализ технологического процесса . . . . .	41
5.4.1	Краткое описание технологического процесса . . . . .	41
5.4.1.1	Схема . . . . .	41
5.4.1.2	Назначение процесса . . . . .	41
5.4.1.3	Сущность процесса . . . . .	41
5.4.1.4	Этапы . . . . .	42
5.4.1.4.1	При моделировании эксперимента . . . . .	42
5.4.1.4.2	При проведении эксперимента на практике . . . . .	42
5.4.1.5	Используемое оборудование . . . . .	43
5.4.2	Сравнительный анализ входных и выходных параметров, присущих данной операции . . . . .	43
5.4.3	Выбор наиболее существенных входных и выходных параметров . . . . .	44

5.4.3.1	Входные параметры . . . . .	44
5.4.3.2	Выходные параметры . . . . .	45
5.5	Построение схемы контроля . . . . .	45
5.6	Проведение математического моделирования технологического процесса . . . . .	45
5.6.1	Обоснование необходимости проведения процесса . . .	45
5.6.2	Разработка плана эксперимента . . . . .	46
5.6.3	Построение математической модели . . . . .	48
5.6.4	Обработка результатов эксперимента . . . . .	50
5.6.4.1	Выборочное среднее и дисперсия . . . . .	50
5.6.4.2	Проверка эксперимента на воспроизводимость	50
5.6.4.3	Определение коэффициентов полинома . . . . .	51
5.6.4.4	Оценка значимости коэффициентов . . . . .	51
5.6.4.5	Проверка модели на адекватность . . . . .	52
5.7	Перерасчёт с другим значением критерия Стьюдента . . . . .	53
5.8	Выводы . . . . .	55
5.8.1	Выводы по модели с 3 значимыми коэффициентами . . .	55
5.8.2	Выводы по модели с 4 значимыми коэффициентами . . .	55
<b>6</b>	<b>Дальнейшая работа</b>	<b>56</b>
	<b>ВЫВОДЫ И РЕЗУЛЬТАТЫ</b>	<b>57</b>
	<b>ЗАКЛЮЧЕНИЕ</b>	<b>58</b>
	<b>Список литературы</b>	<b>59</b>

## ВВЕДЕНИЕ

Требование точности к продукции выдвигает критерий повторяемости на первый план. Повторяемость может быть достигнута путём автоматизации технологического процесса, исключая человеческий фактор при его проведении. С этим может справиться дистанционное автоматизированное управление, которое решает следующие проблемы:

- простой оборудования в связи с отсутствием оператора;
- уменьшение производительности в связи с необходимостью ручного управления;
- ухудшение качества продукции в связи с неточным следованием инструкциям.

*Целью работы является изучение библиотеки Qt, протокола Modbus и разработка ПО по моделированию поведения частиц методом Монте-Карло.*

### **Задачи:**

- изучить выбранный по результатам научной работы протокол передачи данных *Modbus TCP*;
- освоить принцип работы в одной из современных сред разработки и программирования Qt;
- разработать программу обмена сигналами между объектами;
- разработать клиент-серверную модель передачи данных по протоколу TCP;
- разработать ПО для моделирования процесса нанесения плёнки на тонкостенную трубку по методу Монте-Карло; результаты моделирования будут использоваться как входные данные для дисциплины «Техника эксперимента в электронике и нанoeлектронике».

# 1 Протоколы взаимодействия вакуумного оборудования

## 1.1 Модель OSI

Модель *OSI/ISO* – идеальная модель построения сети.

На рис. 1 показана структура идеальной модели *OSI*. Модель определяет различные уровни взаимодействия систем. Каждый уровень выполняет определённые функции при таком взаимодействии.

Данные	Прикладной доступ к сетевым службам
Данные	Представления представление и кодирование данных
Данные	Сеансовый Управление сеансом связи
Блоки	Транспортный безопасное и надёжное соединение точка-точка
Пакеты	Сетевой Определение пути и IP (логическая адресация)
Кадры	Канальный MAC и LLC (Физическая адресация)
Биты	Физический кабель, сигналы, бинарная передача данных

Рис. 1. Уровни модели OSI

Уровни модели OSI делятся на [1] :

- **физический**: предназначен непосредственно для передачи потока данных. Осуществляет передачу электрических или оптических сигналов в кабель или в радиоэфир и, соответственно, их приём и преобразование в

биты данных в соответствии с методами кодирования цифровых сигналов;

- **канальный**: предназначен для обеспечения взаимодействия сетей на физическом уровне. Полученные с физического уровня данные проверяет на ошибки, если нужно исправляет, упаковывает во фреймы, проверяет на целостность, и отправляет на сетевой уровень;
- **сетевой**: определяет пути передачи данных. Отвечает за трансляцию логических адресов и имён в физические, за определение кратчайших маршрутов, коммутацию и маршрутизацию, за отслеживание неполадок и заторов в сети;
- **транспортный**: организует доставку данных без ошибок, потерь и дублирования (не все) в той последовательности, как они были переданы. Разделяет данные на фрагменты равной величины, объединяя короткие и разбивая длинные (размер фрагмента зависит от используемого протокола);
- **сеансовый**: управляет созданием/завершением сеанса связи, обменом информацией, синхронизацией задач, определением права на передачу данных и поддержанием сеанса в периоды неактивности приложений. Синхронизация передачи обеспечивается помещением в поток данных контрольных точек, начиная с которых возобновляется процесс при нарушении взаимодействия;
- **представления**: на этом уровне может осуществляться преобразование протоколов и сжатие/распаковка или кодирование/декодирование данных, а также перенаправление запросов другому сетевому ресурсу, если они не могут быть обработаны локально;
- **прикладной**: уровень приложений (англ. Application layer). Обеспечива-



ет взаимодействие сети и приложений пользователя, выходящих за рамки модели OSI. На этом уровне работают изученные в работе протоколы автоматизации промышленных сетей.

## 1.2 Протокол Modbus

В настоящее время существует огромное множество промышленных протоколов, используемых в автоматизации. На рис. 2 показано распределение рынка между протоколами [2]:

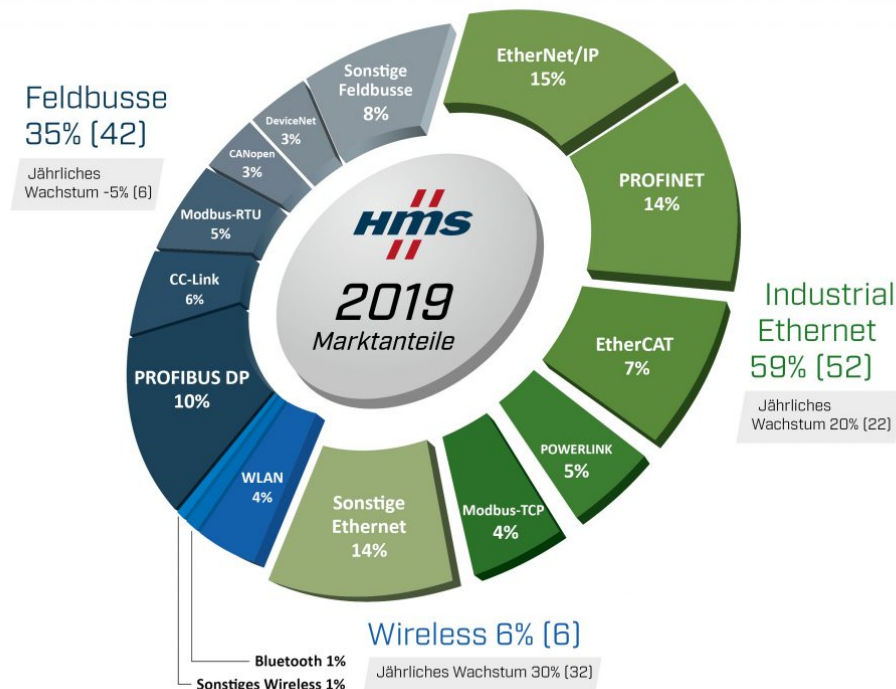


Рис. 2. Диаграмма долей рынка промышленных протоколов

В России основную долю рынка занимают протоколы *Modbus* и *Profibus* [3], однако *Profibus* уступает своему конкуренту в ряде вещей:

- стандарт открыт не полностью;
- сложнее в освоении, чем *Modbus*;
- в СНГ *Modbus* более популярен.

## 1.3 Принцип работы протокола Modbus

### 1.3.1 Физический уровень

Протокол *Modbus* может быть использован со следующими интерфейсами:

- **RS-232/422/485**: последовательные интерфейсы, широко распространенные в промышленности. Интерфейсы RS-422/485 обеспечивают дальность сигнала до 1200 метров. Используются протоколы *Modbus RTU/ASCII*
- **TCP/IP**: физическим каналом передачи данных могут любые ethernet-интерфейсы. Используется протокол *Modbus TCP*.

Существует 3 разновидности протокола *Modbus* [4]:

1. *Modbus ASCII*: в котором данные кодируются символами из таблицы ASCII (рис. 1) и передаются в шестнадцатеричном формате и данный формат протокола встречается довольно редко;
2. *Modbus RTU*: самый распространенный вариант протокола Modbus, который кодирует данные в двоичном формате и разделяет пакеты с помощью временного интервала;
3. *Modbus TCP*: данные кодируются в двоичном формате и упаковываются в TCP - пакет, для передачи по IP-сетям и предназначен для работы в локальных сетях.

На рис. 3 показан пример построения схемы контроля за неким объектом при помощи протокола *Modbus* [5].

### 1.3.2 Логический уровень

Протокол *Modbus* предполагает одно ведущее устройство и до 247 ведомых. Обмен данными начинается ведущим устройством. Ведомые не могут на-

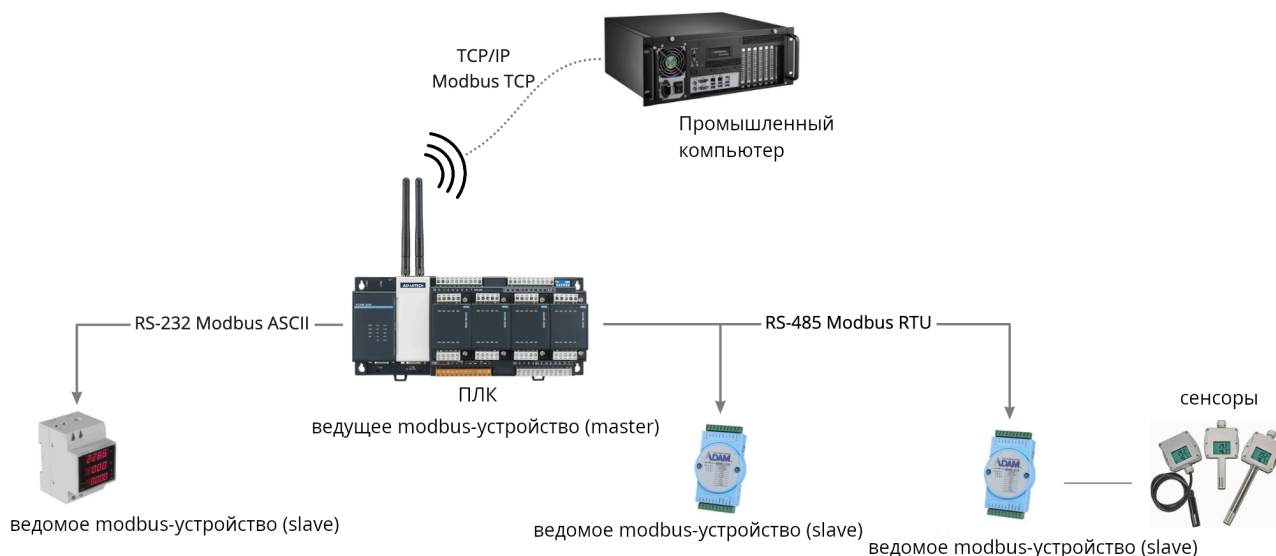


Рис. 3. Физический уровень протокола *Modbus*

чинать передачу и обмениваться данными между собой. В любой момент времени может происходить только один акт обмена. Структуры пакетов *Modbus* при работе 3 способами приведены на рис. 4.

### 1.3.2.1 Modbus RTU

Сообщение начинает восприниматься как новое после паузы длиной в 14 бит. На рис. 5 показан формат пакета. У пакета есть следующие поля [3]:

- **Адрес:** содержит адрес ведомого устройства. Адрес отправляется даже при ответе на запрос мастера, тем самым всегда понятно откуда пришёл ответ;
- **Код функции:** говорит модулю о том, что ему необходимо сделать;
- **Данные:** тут может содержаться информация о параметрах, которые используются в исполнении команд мастера или показания, передаваемые мастеру;
- **Контрольная сумма:** используется для проверки целостности пакета.

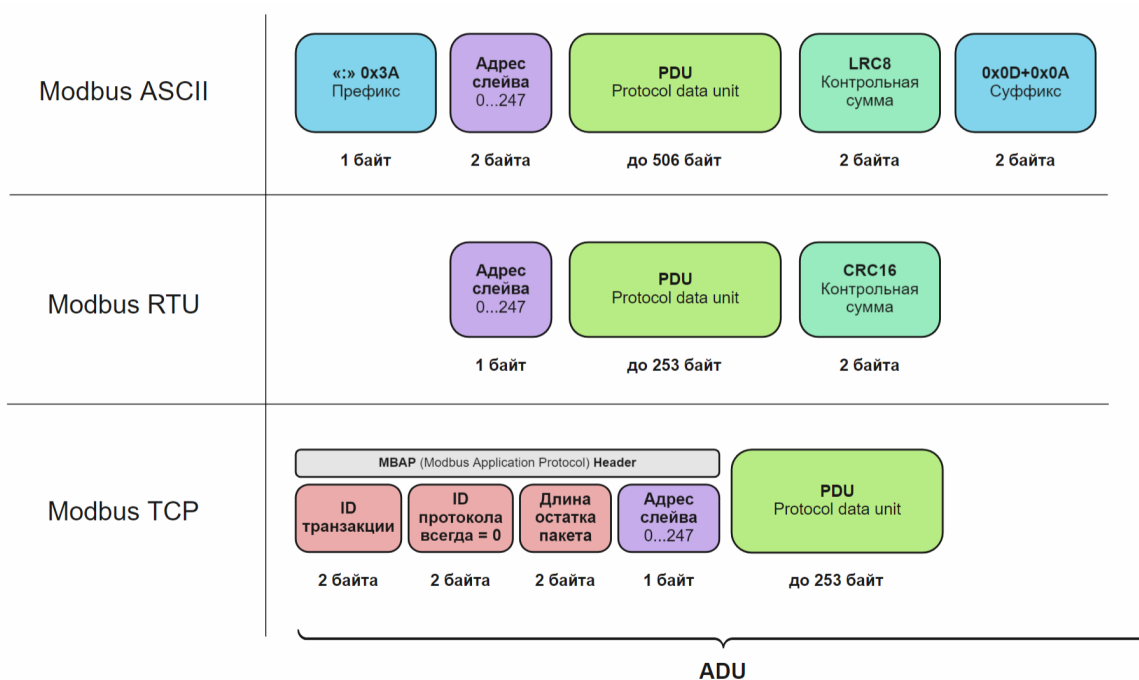


Рис. 4. Структура пакета *Modbus*



Рис. 5. Формат кадра протокола *Modbus RTU*

### 1.3.2.2 Modbus TCP

Данный протокол используется для того, чтобы подключить устройства, работающие по протоколу *Modbus* к сети *Internet* [6]. То есть, в соответствии со стандартом *OSI/ISO* (см. рис. 1) на транспортном уровне используется протокол *TCP*, а на прикладном – *Modbus*. В этом случае проверка целостности пакета ложится на протокол *TCP*. Структура протокола *Modbus TCP* приведена на рис. 6. У пакета есть следующие поля [3]:

- **Идентификатор обмена:** используется для идентификации сообщения в случае, когда в пределах одного *TCP* - соединения клиент посылает серверу

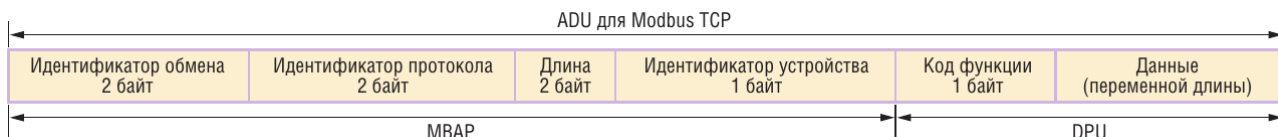


Рис. 6. Формат кадра протокола *Modbus TCP*

ру несколько сообщений без ожидания ответа после каждого сообщения;

- **Идентификатор протокола:** всегда выставлен на 0 (как и протокола *TCP*);
- **Длина:** указывает количество следующих байтов;
- **Идентификатор устройства:** адрес slave - устройства;
- **Код функции:** аналогично 1.3.2.1. **Modbus RTU**;
- **Данные:** аналогично 1.3.2.1. **Modbus RTU**.

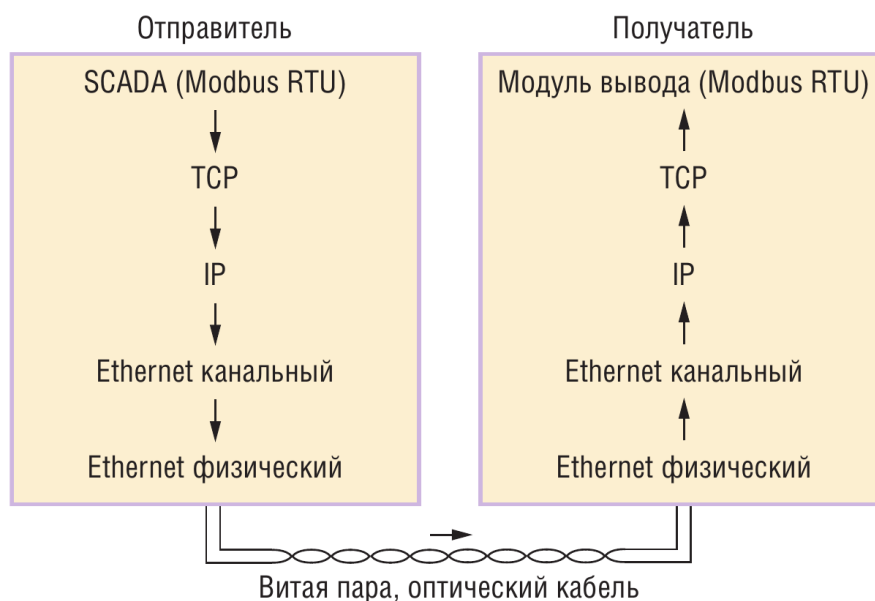


Рис. 7. Процесс передачи пакетов *Modbus RTU* по сети *TCP*

Как можно заметить, *Modbus RTU*, оказывается “вшит” в пакет *TCP*, тем самым получается *Modbus TCP*. На рис. 7 показан принцип работы такой системы [3]:

- коды функций передаются с прикладного уровня на транспортный (*Modbus-TCP*), добавление заголовка *TCP*;

- передача на сетевой уровень, добавление блока *IP*;
- передача на канальный уровень, а затем на физический (*Ethernet*)

После прохождения через канал связи пакет начинает обратное движение согласно модели *OSI/ISO* (см. рис. 1).

Более подробно о работе с протоколом *Modbus* можно узнать в документации [7].

## 2 Составление документации по Modbus на Gitlab

### 2.1 Знакомство с языками разметки документации

Документация на *Gitlab* оформляется на языке разметки. **Язык разметки** – набор символов или последовательностей символов, вставляемых в текст для передачи информации о его отображении или строении.

Текстовый документ, написанный с использованием языка разметки, содержит не только сам текст (как последовательность слов и знаков препинания), но и дополнительную информацию о различных его участках:

- заголовки;
- выделения;
- нумерованные и маркированные списки;
- таблицы.

Наличие языка разметки позволяет иметь документ одинакового вида на всех устройствах, что крайне важно для документации.

#### 2.1.1 HTML

**HTML** – стандартизированный язык разметки документов в сети Интернет. Большинство веб-страниц содержат описание разметки на языке HTML. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

В листинге 1 приведён пример кода на языке HTML. На рис. 8 приведён пример разметки, выполненной на языке HTML.

```
1 | <html>
2 | <head>
```

```

3  <meta http-equiv="Content-Type" content="text/html; charset=
    utf-8">
4  <title>Пример веб – страницы</title>
5  </head>
6  <body>
7  <h1>Заголовок</h1>
8  <!-- Комментарий -->
9  <p>Первый абзац.</p>
10 <p>Второй абзац.</p>
11 </body>
12 </html>

```

Листинг 1. Пример кода на языке HTML

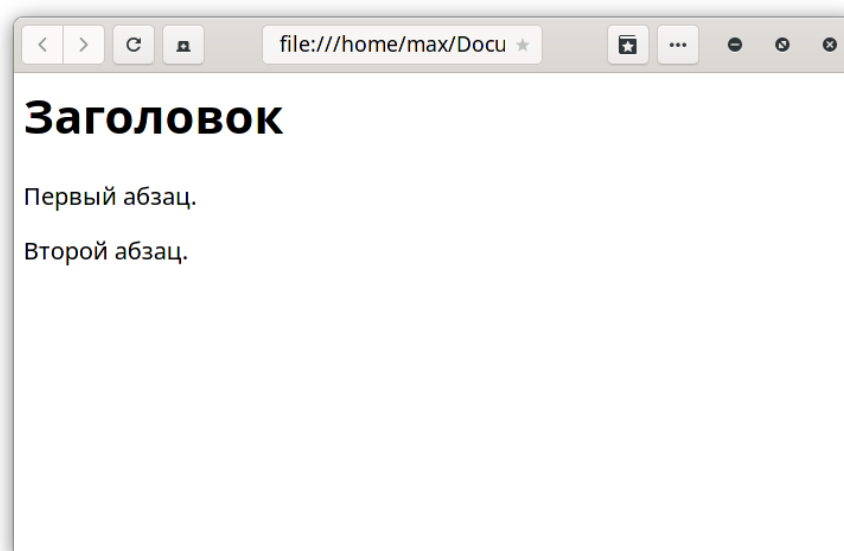


Рис. 8. Результат выполнения примера

### 2.1.2 Markdown

**Markdown** – облегчённый язык разметки, созданный с целью обозначения форматирования в простом тексте, с максимальным сохранением его читаемости человеком, и пригодный для машинного преобразования в языки для продвинутых публикаций (HTML, Rich Text и других).



**Это H1**

**Это H2**

**Это H3**

**Жирный**

**Тоже жирный**

*Курсив*

*Тоже курсив*

```
7 # Это H1
6
5
4 ## Это H2 ##
3
2
1 ### Это H3
8
1
2   Жирный
3
4
5 **Тоже жирный**
6
7
8 *Курсив*
9
10
11 _Тоже курсив_
12
```

Рис. 9. Пример кода на языке разметки Markdown

Этот язык разметки намного проще  $\text{\LaTeX}$  и HTML, поэтому документация была перенесена на него.

На рис. 9 приведён пример оформления документа на языке Markdown.

## 2.2 Структурирование и перенос информации

Информация со старого сервера была проработана и структурирована с использованием документации Modbus Foundation и наработок Л.Л. Колесника. На рис. 10 показана часть переведённой с языка HTML на язык Markdown документации по протоколу *Modbus*.

# Описание протокола MODBUS

Документ находится в стадии разработки и не является окончательной версией

Описание основано на MODBUS Application Protocol Specification V1.1b.

- Введение
- Обозначения и сокращения
- Общее описание протокола
  - Блок данных протокола для отправки запроса
  - Блок данных протокола для отправки ответа
  - Блок данных протокола для отправки ответа в случае возникновения ошибки
- Порядок передачи данных
  - Пример
- Модель хранения данных
- Примеры реализации хранения данных на сервере
  - Пример 1. Устройство использует 4 независимых блока памяти
  - Пример 2. Устройство использует 1 блок памяти
- Адресация данных в протоколе
- Транзакция протокола
- Коды функций
  - Коды функций, определяемые спецификацией
  - Коды функций, определяемые разработчиками
  - Зарезервированные коды функций
- Описание функций
  - 01 (0x01) Чтение массива значений дискретных выходов
    - Структура запроса
    - Структура ответа
    - Информирование о возникшей ошибке
    - Пример запроса и ответа
    - Диаграмма состояния
  - 02 (0x02) Чтение массива значений дискретных входов
    - Структура запроса

Рис. 10. Информация в структурированном виде

## 3 Реализация обмена сообщениями в среде QT

### 3.1 Сигналы и слоты

**Сигналы и слоты** используются для коммуникации между объектами в Qt:

- **сигнал**: вырабатывается когда происходит определенное событие;
- **слот**: функция, которая вызывается в ответ на определенный сигнал.

На рис. 11 показан механизм работы сигналов и слотов. Здесь

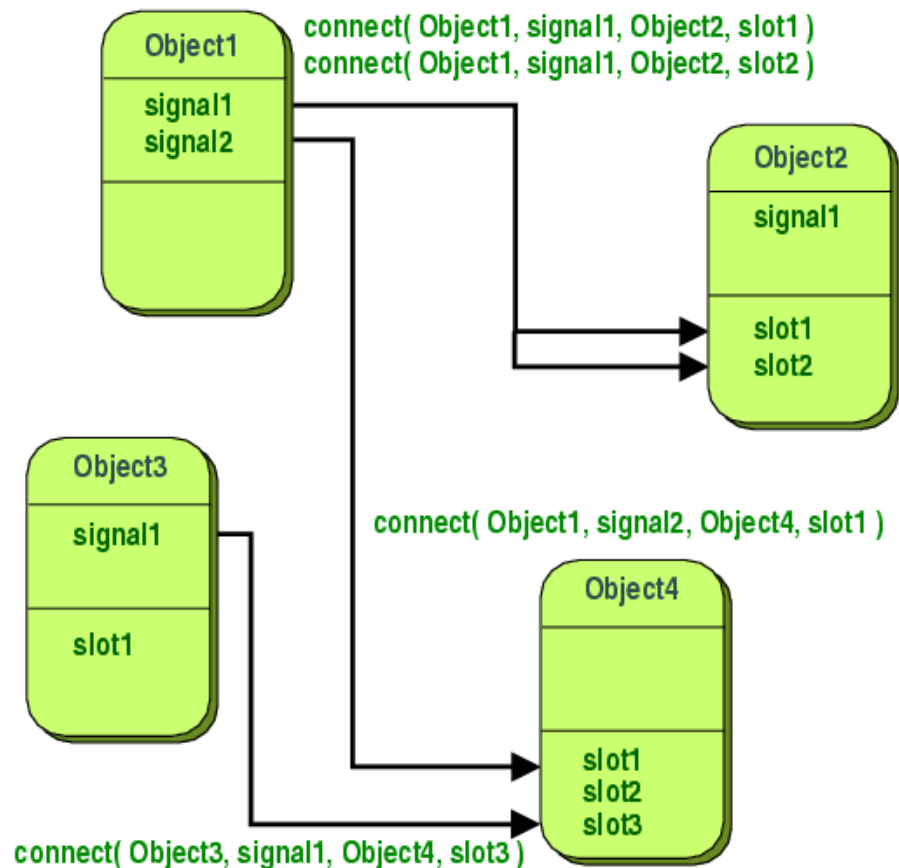


Рис. 11. Схема действия сигналов и слотов

- $Object_1$ ,  $Object_2$  – объекты, имеющие в себе сигналы и слоты;
- $connect(args)$  – функция, соединяющая сигналы и слоты в первом и втором объекте

## 3.2 Концепция программы

**Концепция программы** – реализовать пересылку сообщений между двумя объектами одного класса.

Первый объект будет отправлять второму сигнал *ping*, а второй первому – сигнал *pong*.

## 3.3 Реализация кода на языке Qt

### 3.3.1 Структура класса

В листинге 2 приведён отрывок программы *ping - pong*.

```
1 public slots:
2     void processPing(const QString& msg);
3     void processPong(const QString& msg);
4     void sendPing();
5
6 signals:
7     void ping(QString = "", int = 0);
8     void pong(QString);
9     void msgPinged(QString);
10    void msgPonged(QString);
```

Листинг 2. Сигналы и слоты класса

Здесь содержатся 4 сигнала и 3 слота:

— **Сигналы:**

1. *ping* – сигнал, отправляемый слотом *sendPing*;
2. *pong* – сигнал, отправляемый слотом *processPing* (см. листинг 3);
3. *msgPinged* – сигнал графического интерфейса, отправляемый слотом *sendPing* (см. листинг 5);

4. *msgPonged* – сигнал графического интерфейса, отправляемый слотом *processPong* (см. листинг 4).

— Слоты:

1. *processPing* – получает сигнал *ping* и отправляет сигнал *pong*:

```
1 void MainClass::processPing(const QString& msg)
2 {
3     emit pong(msg);
4 }
```

Листинг 3. Реализация слота processPing

2. *processPong* – получает сигнал *pong* и приступает к его обработке: записывает в строку номер сообщения и время его получения:

```
1 void MainClass::processPong(const QString& msg)
2 {
3     // ? Display pong process
4     QString m;
5     m.append("Message ponged # ");
6     m.append(QString::number(a));
7     m.append(". Time: ");
8     m.append(QTime::currentTime().toString("hh:mm:ss
9         "));
10    emit msgPonged(m);
11 }
```

Листинг 4. Реализация слота processPong

3. *sendPing* – слот, соединённый с таймером, отправляет сигналы *pong* и *msgPinged*:

```
1 void MainClass::sendPing()
2 {
3     ++a;
```

```

4      emit ping(QString::number(a));
5      QString m;
6      m.append("Message pinged # ");
7      m.append(QString::number(a));
8      m.append(". Time: ");
9      m.append(QTime::currentTime().toString("hh:mm:ss
      "));
10     emit msgPinged(m);
11 }

```

Листинг 5. Реализация слота sendPing

### 3.3.2 Связь сигналов и слотов

Для работы с сигналами и слотами, они должны быть соединены между собой. Соединение сигналов и слотов класса между собой приведены в листинге 6:

```

1  QObject::connect(&timer, SIGNAL(timeout()),
2      &test1, SLOT(sendPing()));
3
4  QObject::connect(&test1, SIGNAL(ping(QString)),
5      &test2, SLOT(processPing(QString)));
6
7  QObject::connect(&test2, SIGNAL(pong(QString)),
8      &test1, SLOT(processPong(QString)));
9
10 QObject::connect(&test1, SIGNAL(msgPinged(QString)),
11     ui->plainTextEdit, SLOT(appendPlainText(QString)));
12
13 QObject::connect(&test1, SIGNAL(msgPonged(QString)),
14     ui->plainTextEdit_2, SLOT(appendPlainText(QString)));

```

Листинг 6. Соединение сигналов и слотов

Принцип работы программы следующий (см. рис. 12):

1. Вызывается таймер;
2. Таймер вызывает слот *sendPing*;
3. Слот *sendPing* отправляет сигнал *ping*, а также отправляет сигнал графическому интерфейсу *msgPinged*;
4. Сигнал *ping* вызывает слот *processPing*;
5. Слот *processPing* отправляет сигнал *pong*;
6. Сигнал *pong* вызывает слот *processPong*;
7. Слот *processPong* отправляет сигнал графическому интерфейсу *msgPonged*

### 3.3.3 Графический интерфейс и окно вывода

На рис. 13 показан интерфейс программы до начала работы и после её окончания.

В левом поле пишутся сообщения класса *ping*, в правом – *pong*. Нижнее поле предназначено для задания интервалов отправки сообщений в миллисекундах.

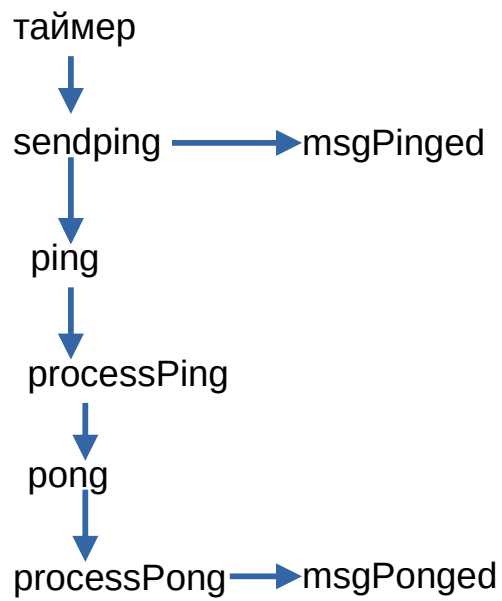
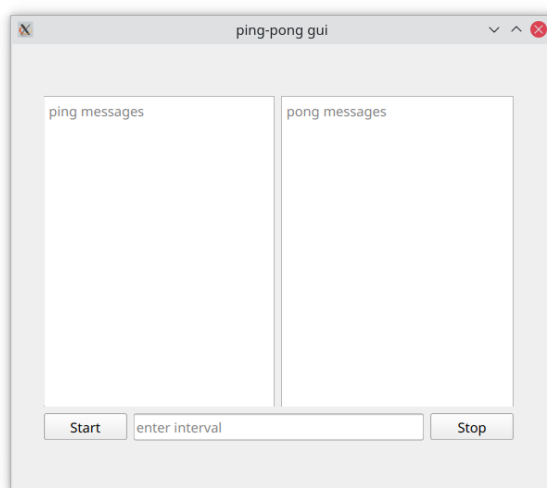
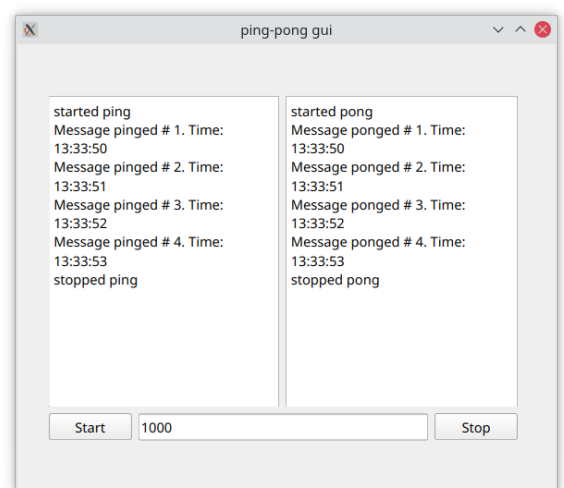


Рис. 12. Структурная схема программы



а) Интерфейс при открытии



б) Интерфейс после исполнения

Рис. 13. Графический интерфейс программы



## 4 Создание сервера обмена сообщениями по протоколу TCP

### 4.1 Принцип работы в сетях TCP/IP

**TCP/IP** — сетевая модель передачи данных, представленных в цифровом виде. Протокол *TCP* — протокол транспортного уровня согласно модели *OSI/ISO* (см. рис. 1).

Основное достоинство протокола – ориентированность на “качество” соединения. *TCP* устанавливает предварительное соединение, а затем проверяет, действительно ли файл дошёл до получателя [1].

На рис. 14 показано, как устанавливается соединения между устройствами по протоколу *TCP* (технология тройного рукопожатия):

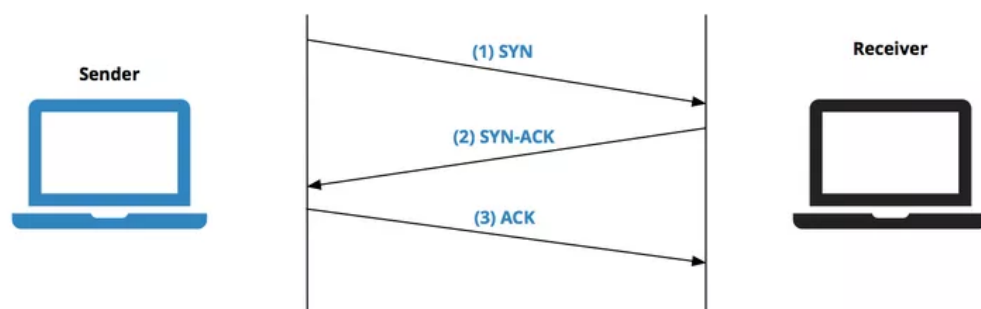


Рис. 14. Установка соединения протоколом *TCP*

1. Клиент отправляет серверу пакет с флагом SYN. Этот начальный пакет позволяет клиенту установить, каким должен быть первый порядковый номер для пакетов запроса;
2. Сервер, получив пакет, добавляет к нему флаг ACK и отправляет клиенту. Этот пакет подтверждает порядковый номер, отправленный клиентом, подтверждая его. Однако сервер также должен отправить SYN и порядко-

вый номер обратно клиенту, чтобы установить, каким должен быть первый порядковый номер для пакетов ответа, исходящих от сервера;

3. Наконец, клиент отвечает на пакет SYN / ACK пакетом ACK, который подтверждает запрос порядкового номера сервера.

## 4.2 Принцип работы программы

Программа состоит из двух частей:

1. Клиент;
2. Сервер;

Программа работает следующим образом (см. рис. 15):

1. Производится запуск сервера;
2. Производится запуск клиента;
3. С клиента поступает запрос на подключение, который всегда одобряется;
4. После подключения сервер отправляет клиенту сообщение об успешном подключении (этого можно было не делать, поскольку успешность подключения обеспечивается протоколом *TCP*).

После этого клиент может передать серверу файл. Контроль за передачей файла ведётся из консоли сервера, а также в графическом интерфейсе клиента.

## 4.3 Реализация кода программы

1. **Запуск сервера.** На листинге 7 приведён метод, осуществляющий запуск сервера на порт 9999.

```

1      void myserver::startServer ()
2      {
3          if ( this->listen (QHostAddress::Any ,9999))
4          {
5              qDebug() << "Listening";
6          }
7          else
8          {
9              qDebug() << "Not listening";
10         }
11     }

```

Листинг 7. Метод запуска сервера

2. При поступлении входящего запроса реализуется метод, приведённый в листинге 8:

```

1      void myserver::incomingConnection ( qintptr
           socketDescriptor )
2      {
3          socket = new QTcpSocket ( this );
4          socket->setSocketDescriptor ( socketDescriptor );
5          connect ( socket , SIGNAL ( readyRead ( ) ) , this , SLOT (
           sockReady ( ) ) );
6          connect ( socket , SIGNAL ( disconnected ( ) ) , this , SLOT (
           sockDisc ( ) ) );
7          qDebug() << socketDescriptor << " Client connected";

           socket->write ( "You are connect" );
8          qDebug() << "Send client connect status - YES";
9
10     }

```

Листинг 8. Метод входящего подключения

## 4.4 Терминал сервера

Сервер не нуждается в графической оболочке, ему достаточно выводить текст в консоль посредством встроенных в Qt средств (`qDebug()`). На рис. 16 показан вывод при запуске сервера.

На рис. 17 показан вывод сообщения при подключении клиента к сервера. 5 – уникальный номер соединения.

На последнем рис. 18 выведено сообщение при получении сервером файла.

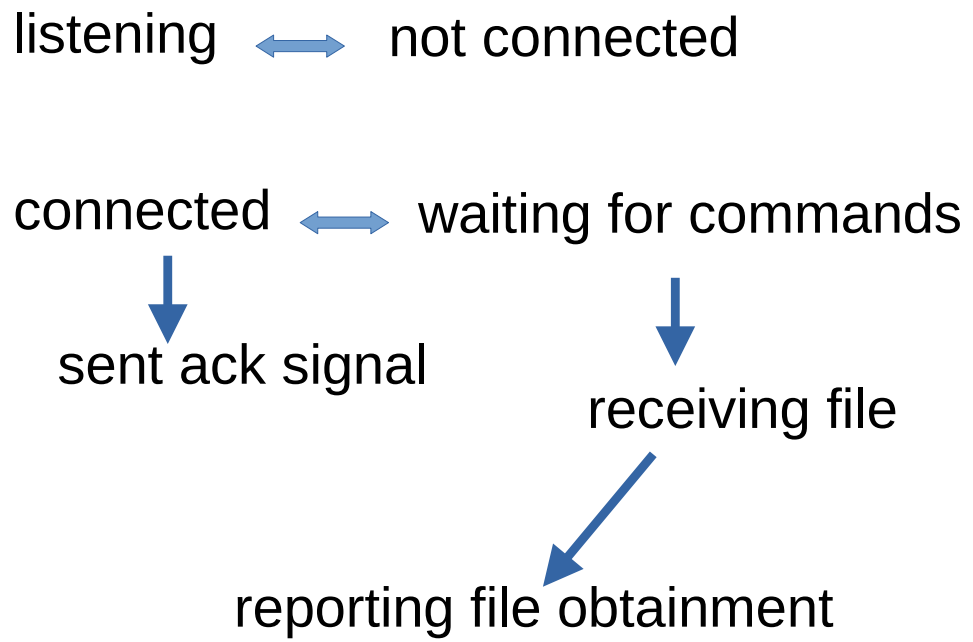


Рис. 15. Схема работы серверной части

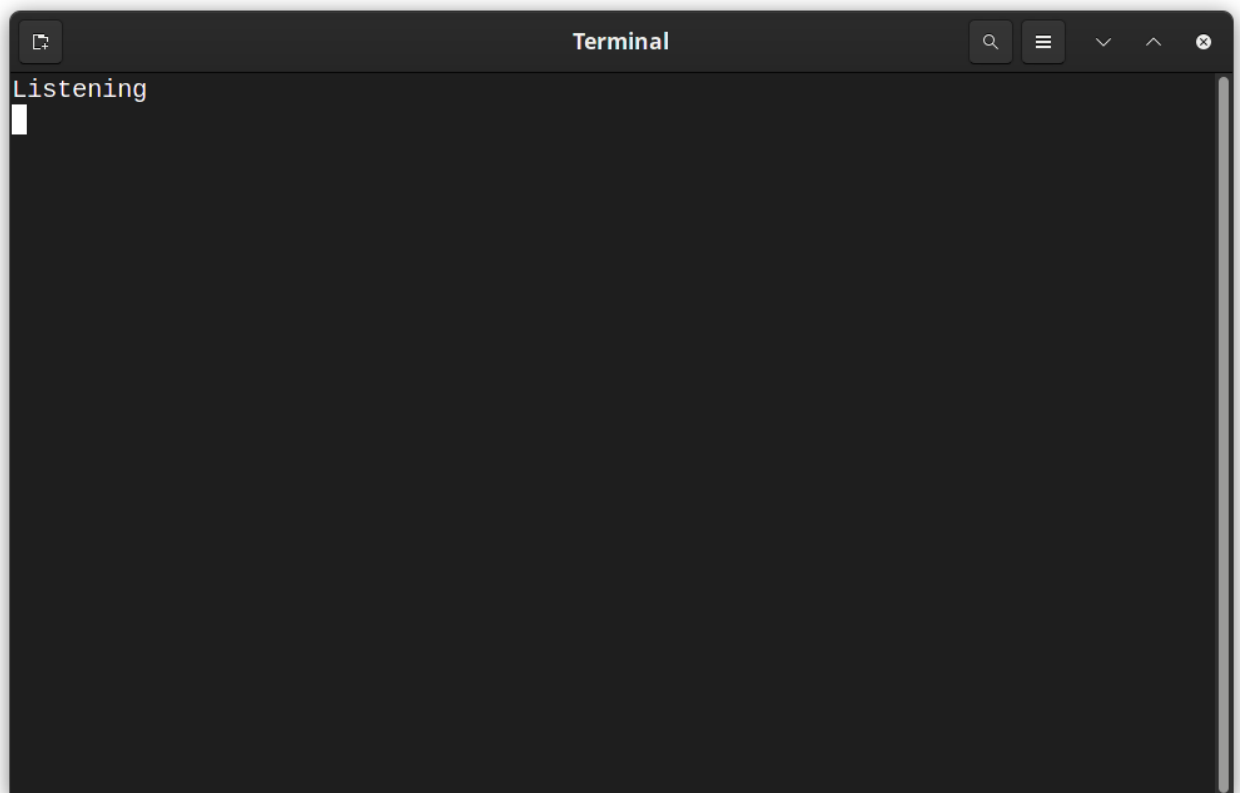
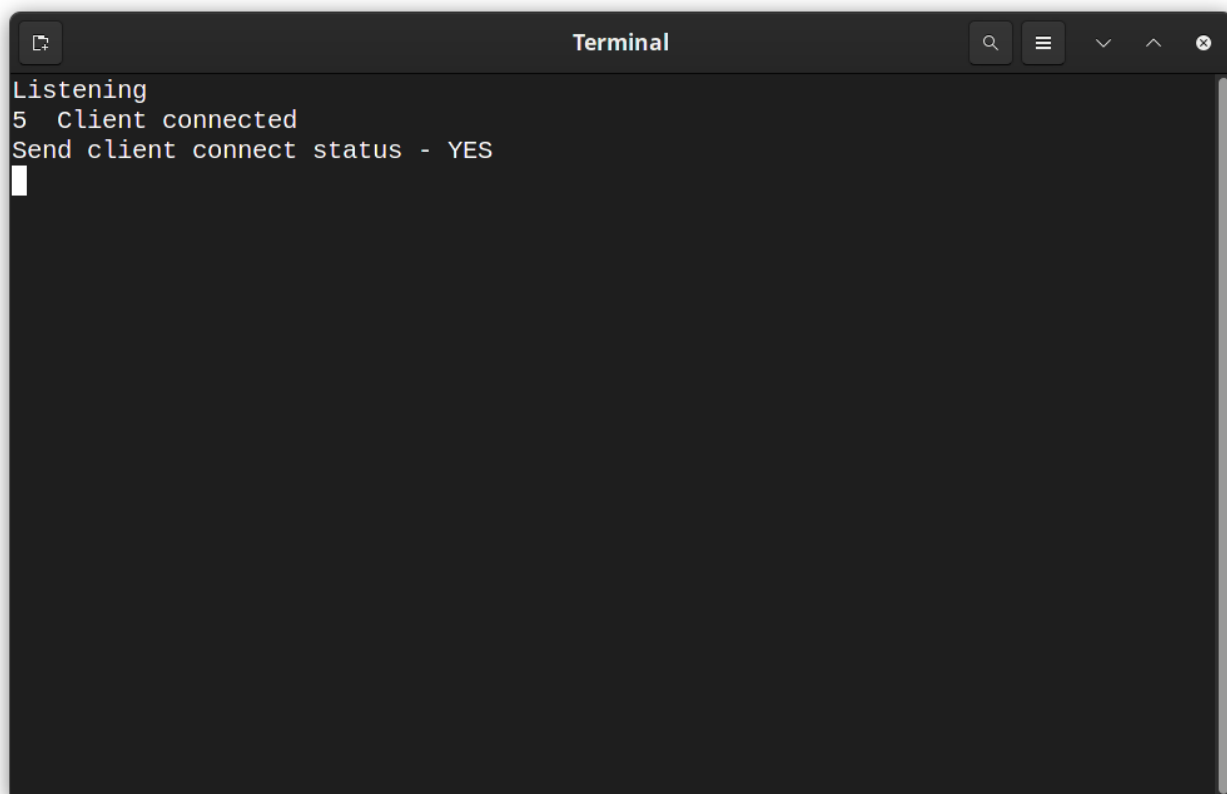


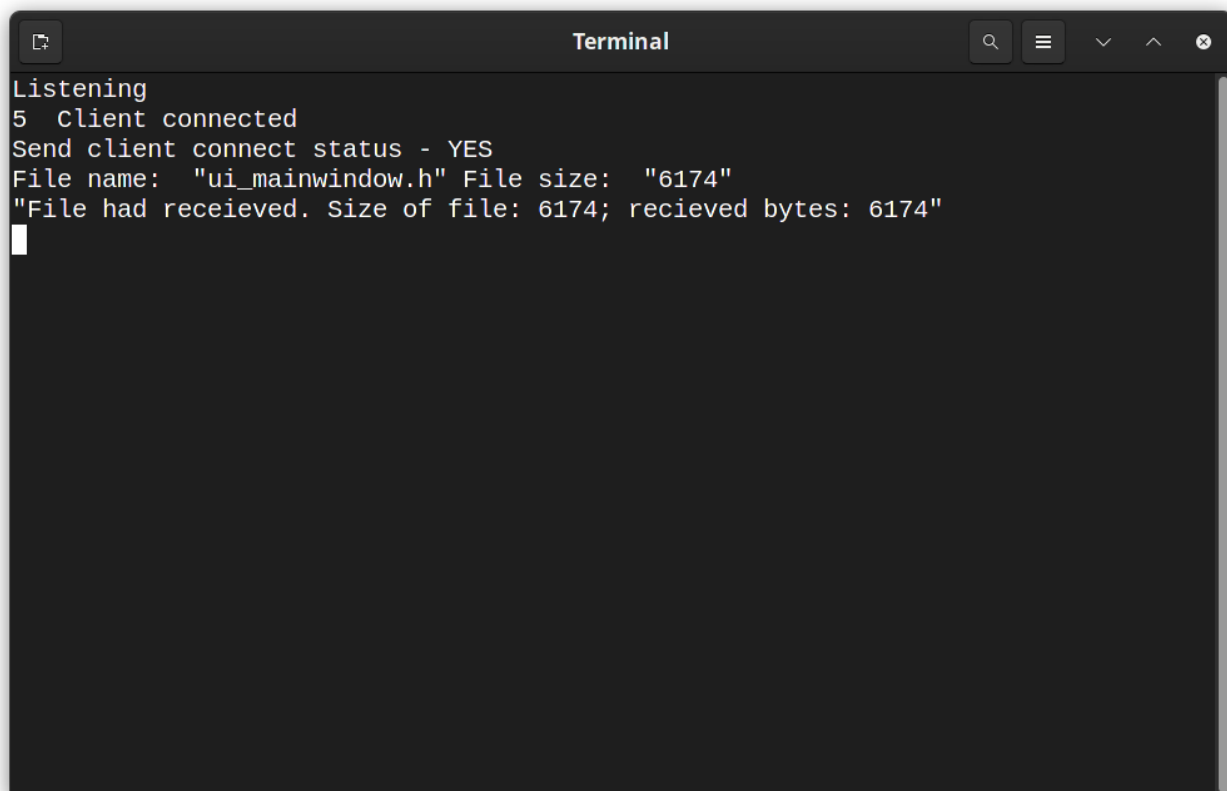
Рис. 16. Запуск сервера

A terminal window titled "Terminal" with a dark background. The text displayed is: "Listening", "5 Client connected", and "Send client connect status - YES". A white cursor is on the line following the last message.

```
Listening
5 Client connected
Send client connect status - YES

```

Рис. 17. Подключение клиента к серверу

A terminal window titled "Terminal" with a dark background. The text displayed is: "Listening", "5 Client connected", "Send client connect status - YES", "File name: 'ui\_mainwindow.h' File size: '6174'", and "File had received. Size of file: 6174; recieved bytes: 6174". A white cursor is on the line following the last message.

```
Listening
5 Client connected
Send client connect status - YES
File name: "ui_mainwindow.h" File size: "6174"
File had received. Size of file: 6174; recieved bytes: 6174

```

Рис. 18. Получение файла от сервера

## 5 ПО для моделирования нанесения покрытия

### 5.1 Краткое резюме

Описанное ниже программное обеспечение является учебным и позволяет смоделировать нанесение покрытия на внутреннюю часть тонкой цилиндрической трубки, расположенной под наклоном методом Монте-Карло.

Программа генерирует случайные числа (в данном случае случайным числом является синус угла распыления).

Для записи результатов применяется словарь (число-структура). Ключ словаря – нижнее значение интервала разбиения длины трубки. Например, если молекула попала на отрезок  $[10, 20]$ , то ключём будет 10.

Значением является структура *values* (см. листинг 9). Структура содержит в себе поле значения количества попавших в интервал молекул, а также координаты каждой из молекул по стороне  $L$ .

```
1 struct value
2 {
3     int i;
4     vector<double> vec;
5 };
6 map<int, value> right_values;
7 map<int, value> left_values;
```

Листинг 9. Запись значений

Программа совершает большое количество итераций по описанному далее методу и выводит результаты в файл (см. листинг 15). В файле содержится количество попавших в каждый интервал на каждой стороне молекул.

### 5.2 Расчёт геометрии

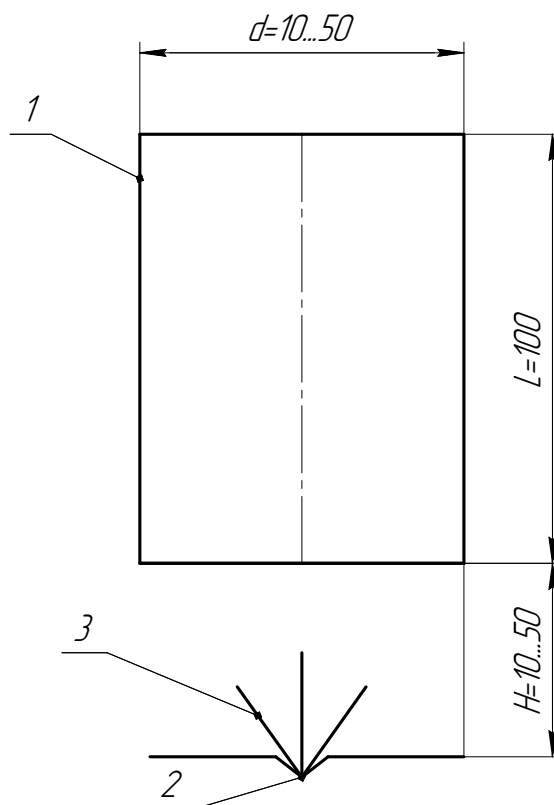


Рис. 19. Эскиз расчёта

### 5.2.1 Эскиз

На рис. 19 изображён эскиз будущего расчёта.

1. Трубка, на внутреннюю поверхность которой наносятся частицы;
2. Точечный источник, из которого вылетают частицы;
3. Частицы.

### 5.2.2 Граничные условия

Молекулы будут попадать на внутреннюю поверхность в том случае, если (см. рис. 20):

— **слева:** угол распыления

$$\psi_{\text{расп}} \in [\mu_{\text{пр}}, \psi_{\text{пр}}] \quad (1)$$



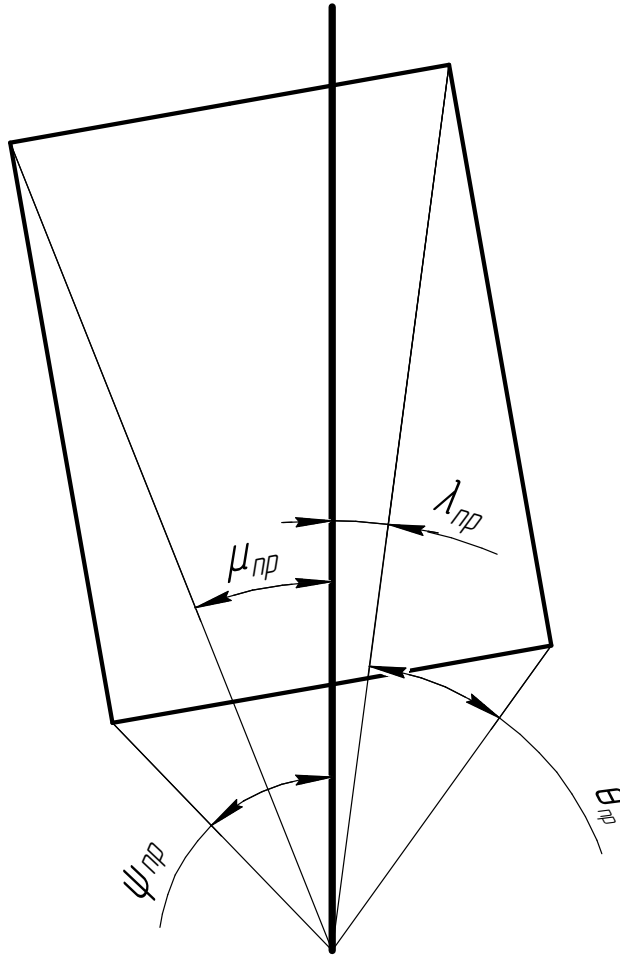


Рис. 20. Граничные условия

— **справа:** угол распыления

$$\psi_{\text{расп}} \in [\lambda_{\text{пр}}, \theta_{\text{пр}}] \quad (2)$$

### 5.2.2.1 Нижние граничные условия

Рассмотрим нижнюю часть трубки для нахождения углов  $\mu_{\text{пр}}$  и  $\lambda_{\text{пр}}$  (см. рис. 21). В треугольнике  $ACB$ :

$$\psi_{\text{пр}} = \arctg \left( \frac{AM}{BM} \right) = \arctg \left( \frac{\frac{d \cos \varphi}{2}}{H - \frac{d}{2} \sin \varphi} \right) = \arctg \left( \frac{d}{(2 (H - \frac{d}{2} \sin \varphi))} \right) \quad (3)$$

Похожим образом найдём правое граничное условие из треугольника  $BNE$ :

$$\theta_{\text{пр}} = \arctg \left( \frac{NE}{BN} \right) = \arctg \left( \frac{\frac{d \cos \varphi}{2}}{H + \frac{d}{2} \sin \varphi} \right) = \arctg \left( \frac{d}{(2 (H + \frac{d}{2} \sin \varphi))} \right) \quad (4)$$

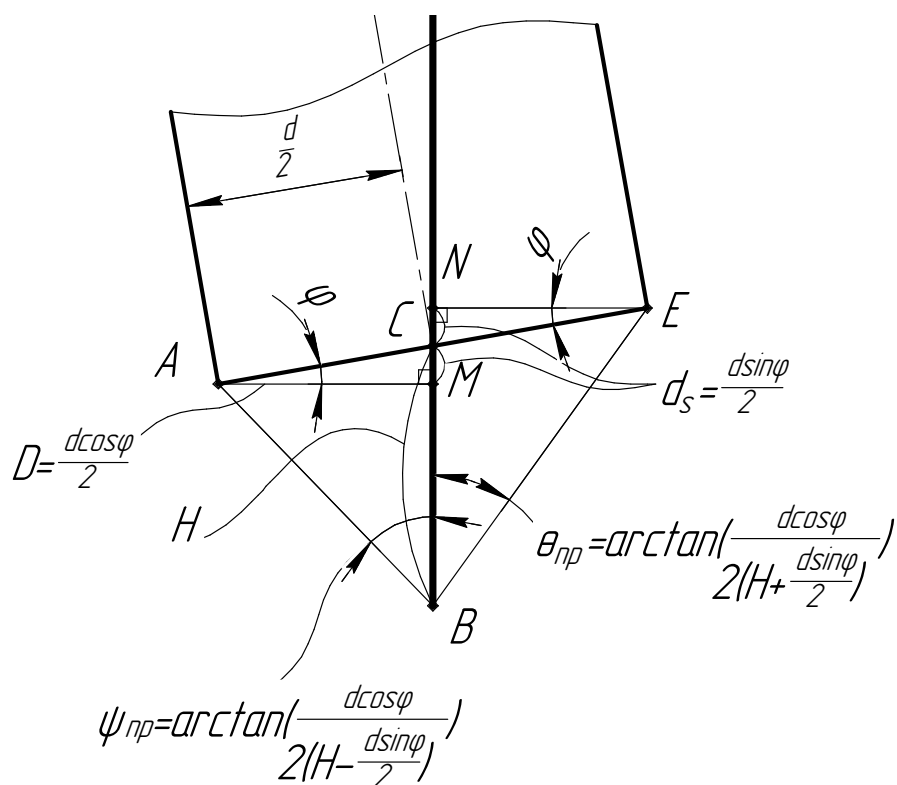


Рис. 21. Нижние граничные условия

#### 5.2.2.2 Верхние граничные условия

Рассмотрим верхнюю часть трубки (см. рис. 22). Найдём левую границу (рис. 22а):

1. Угол  $\gamma$  из треугольника  $BHF$ :

$$\gamma = \operatorname{arctg} \left( \frac{BF}{BH} \right) = \operatorname{arctg} \left( \frac{\frac{d}{2}}{b} \right) = \operatorname{arctg} \left( \frac{d}{2\sqrt{\left(\frac{d}{2}\right)^2 + L^2}} \right) \quad (5)$$

2. Угол  $AHB$ :

$$AHB = \delta = 180^\circ - \varphi - \gamma \quad (6)$$

3. В треугольнике  $AHB$  по теореме косинусов находим сторону  $a$ :

$$a = b^2 + H^2 - 2bH \cos(180^\circ - \varphi - \delta) \quad (7)$$

4. По теореме косинусов треугольника  $AHB$  находим искомый угол:

$$\begin{aligned} b^2 &= a^2 + H^2 = 2aH \cos(\mu_{\text{пр}}) \\ \mu_{\text{пр}} &= \arccos\left(\frac{a^2 + H^2 - b^2}{2aH}\right) \end{aligned} \quad (8)$$

Правое граничное условие может быть найдено по аналогии (см. рис. **22б**):

$$\begin{aligned} b^2 &= a^2 + H^2 = 2aH \cos(\lambda_{\text{пр}}) \\ \lambda_{\text{пр}} &= \arccos\left(\frac{a^2 + H^2 - b^2}{2aH}\right) \end{aligned} \quad (9)$$

### 5.2.2.3 Нахождение интервала попадания

Как уже было сказано в **5.1. Краткое резюме**, программа вычисляет координату попадания молекулы на сторону. На рис. **23** изображены расчётные схемы для координат молекул.

1. В треугольнике  $BCD$  (рис. **23а**) известны стороны  $AB$  (можно найти из рис. **21**),  $BD = \frac{d}{2}$  и граничный угол  $\psi_{\text{пр}}$  (см. уравнение (3)). Находим углы  $\varepsilon$  и  $\angle CAB$ :

$$\begin{aligned} \varepsilon &= \arcsin\left(\frac{2k \sin \psi_{\text{пр}}}{d}\right) \\ \angle CAB &= 180^\circ - \psi_{\text{пр}} - \varepsilon \end{aligned} \quad (10)$$

2. В треугольнике  $DAB$  (см. рис. **23б**) находим углы  $\angle ADB$  и  $\angle ABD$ :

$$\begin{aligned} \angle ABD &= \psi_{\text{пр}} - \psi_{\text{расп}} \\ \angle DAB &= 90^\circ + \angle CAB = 90^\circ + 180^\circ - \psi_{\text{пр}} - \varepsilon = 270^\circ - \psi_{\text{пр}} - \varepsilon \\ \angle ADB &= 180^\circ - \angle DAB - \angle ABD = \\ &= 180^\circ - 270^\circ + \psi_{\text{пр}} + \varepsilon - \psi_{\text{пр}} + \psi_{\text{расп}} = \varepsilon + \psi_{\text{расп}} - 90^\circ \end{aligned} \quad (11)$$

3. По теореме синусов треугольника  $ADB$  находим искомую сторону (координату распыления):

$$AD = \frac{AB \sin(\angle ABD)}{\sin(\angle ADB)} \quad (12)$$

## 5.3 Принцип работы программы

### 5.3.1 Преобразование Бокса-Мюллера

**Преобразование Бокса-Мюллера** – метод моделирования стандартных нормально распределённых случайных величин:

1. Выбираются 2 независимые случайные величины, равномерно распределённые на отрезке  $[-1, 1]$ :

$$x \in [-1, 1], \quad y \in [-1, 1] \quad (13)$$

2. Вычисляем  $s$ :

$$s = x^2 + y^2 \quad (14)$$

3. Если

$$0 < s \leq 1 \quad (15)$$

то вычисляем  $z_0$  и  $z_1$ :

$$\begin{aligned} z_0 &= x \sqrt{\frac{-2 \ln s}{s}} \\ z_1 &= y \sqrt{\frac{-2 \ln s}{s}} \end{aligned} \quad (16)$$

в противном случае повторяем пункт **1** и уравнение (13).

В листинге **10** приведена реализация алгоритма Бокса - Мюллера на языке C++.

```
1  double box_muller ()
2  {
3      double s;
4      static quint64 i = 0;
5      double x;
6      double y;
7      double q;
8      double w;
```

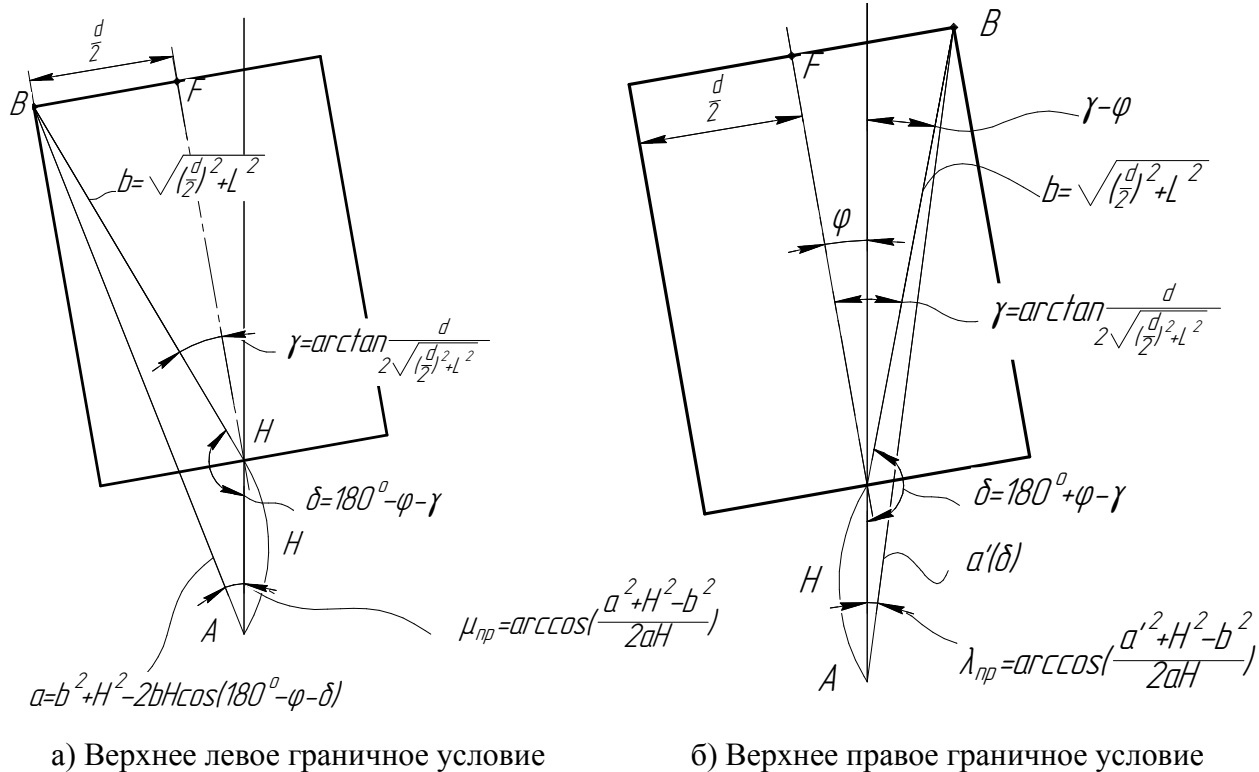


Рис. 22. Верхние граничные условия

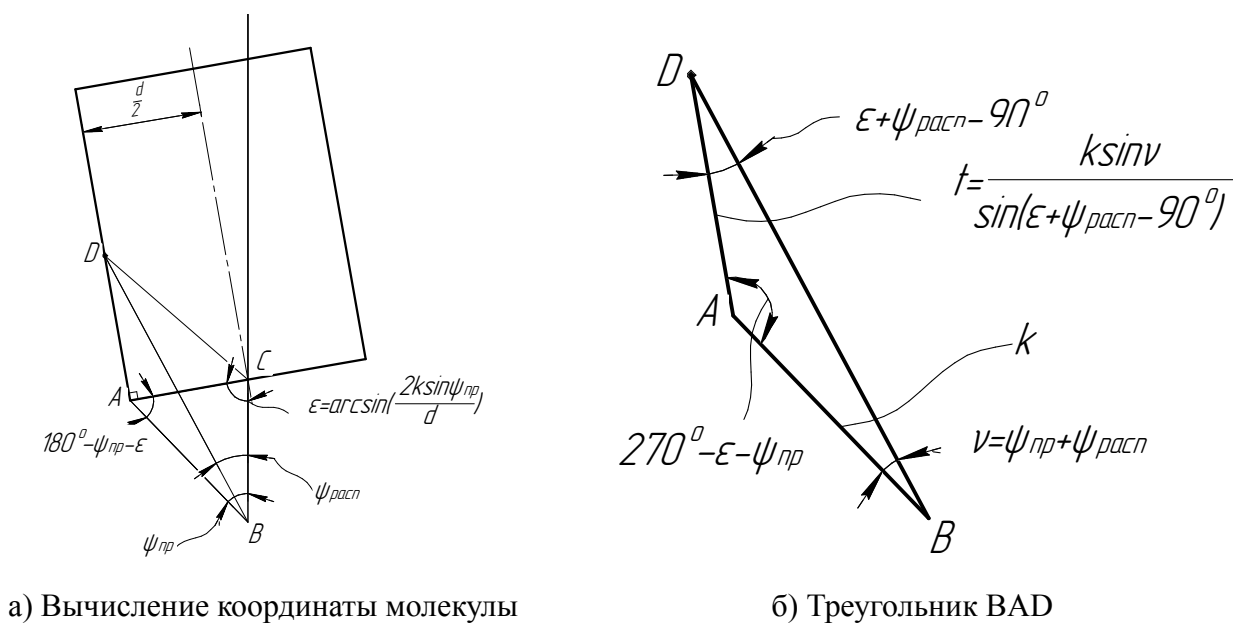


Рис. 23. Расчётная схема вычисления координаты молекулы

```

9      do
10     {
11         x = randomnum();
12         y = -randomnum();
13         s = x * x + y * y;
14         q = x * qSqrt((-2 * qLn(s))/s);
15         w = y * qSqrt((-2 * qLn(s))/s);
16     }
17     while (s>1 || s==0 || qAbs(q)>1 || qAbs(w) > 1 || qAbs(q)>1);
18     i++;
19     if (i % 2 == 0)
20         return w;
21     else
22         return q;
23 }

```

Листинг 10. Реализация алгоритма Бокса-Мюллера

### 5.3.2 Генерация случайных чисел в библиотеке Qt

Qt предоставляет класс *QRandomGenerator* для генерации случайных чисел. Функция *randomnum()* из листинга 10 реализуется как раз при помощи этого класса. Реализация функции приведена в листинге 11.

```

1 double randomnum()
2 {
3     QRandomGenerator generator;
4     return generator.global()->generateDouble();
5 }

```

Листинг 11. Генерация случайных чисел

### 5.3.3 Реализация программы на языке C++

Первым делом задаются начальные значения. Было принято, что расстояние от источника до центра основания  $H$ , угол наклона оси трубки  $\varphi$  и её диа-

метр  $d$  лежат в следующих интервалах:

$$\begin{aligned} H &\in [10, 50] \text{ мм} \\ \varphi &\in [0, 10]^\circ \\ d &\in [10, 50] \text{ мм} \end{aligned} \tag{17}$$

```
1 double H = 30;
2 double L = 100;
3 double fi_pi = 5; // tut
4 double fi = fi_pi * M_PI / 180;
5 double d = 40; // tut
6 double r = 145e-12;
```

Листинг 12. Исходные значения

Затем вычисляются величины по формулам, указанным в **5.2. Расчёт геометрии**.

```
1 double left_down_angle = qAtan(d*qCos(fi)/(2*(H-d/2*qSin(fi))));
2 double right_down_angle = qAtan(d*qCos(fi)/(2*(H+d/2*qSin(fi))));
3 double left_down_dlina = d*qCos(fi)/(2*(qSin(left_down_angle)));
4 double right_down_dlina = d*qCos(fi)/(2*(qSin(right_down_angle)));
5 double left_up_dlina = sqrt(pow(d/2,2)+pow(L,2)+pow(H,2)-2*sqrt(pow(d,2)/4+pow(L,2))*H*qCos(M_PI-fi-qAtan(d/(2*L))));
6 double left_up_angle = qAcos(-(pow(d,2)/4+pow(L,2)-pow(left_up_dlina,2)-pow(H,2))/(2*H*left_up_dlina));
7 double right_up_dlina = sqrt(pow(d/2,2)+pow(L,2)+pow(H,2)-2*sqrt(pow(d,2)/4+pow(L,2))*H*qCos(M_PI+fi-qAtan(d/(2*L))));
8 double right_up_angle = qAcos(-(pow(d,2)/4+pow(L,2)-pow(right_up_dlina,2)-pow(H,2))/(2*H*double left_down_dlina = d*qCos(fi)/(2*(qSin(left_down_angle)));
```

```

9   double right_down_dlina = d*qCos( fi )/(2*(qSin(right_down_angle
    )));
10  qDebug() << right_down_dlina << "right_down_dlina";
11  double left_up_dlina = sqrt(pow(d/2,2)+pow(L,2)+pow(H,2)-2*
    sqrt(pow(d,2)/4+pow(L,2))*H*qCos( M_PI-fi-qAtan( d/(2*L) ) ) );
12  double left_up_angle = qAcos(-(pow(d,2)/4+pow(L,2)-pow(
    left_up_dlina ,2)-pow(H,2))/(2*H*left_up_dlina));
13  double right_up_dlina = sqrt(pow(d/2,2)+pow(L,2)+pow(H,2)-2*
    sqrt(pow(d,2)/4+pow(L,2))*H*qCos( M_PI+fi-qAtan( d/(2*L) ) ) );
14  double right_up_angle = qAcos(-(pow(d,2)/4+pow(L,2)-pow(
    right_up_dlina ,2)-pow(H,2))/(2*H*right_up_dlina));_up_dlina
    ));
15  double eps_left = qAsin(left_down_dlina*2*qSin(left_down_angle
    )/d);
16  double eps_right = M_PI-eps_left;

```

Листинг 13. Вычисление граничных условий

Далее начинается цикл из десяти миллионов итераций (листинг 14). Именно здесь реализуются алгоритмы, описанные выше.

```

1   for (int j = 0; j < 10000000; ++j)
2   {
3       double buff = box_muller();
4       if (j%1000000 == 0)
5           qDebug() << j/1000000 <<"/10";
6       if (qAsin(buff)>-right_down_angle && qAsin(buff)<
            -right_up_angle)
7       {
8           double rh = dlina(right_down_angle ,right_down_dlina ,
                eps_right ,qAsin(qAbs( buff) ) );
9           right_values[ address( rh) ].i++;
10          right_values[ address( rh) ].vec.push_back( rh );
11      }
12      else if (qAsin(buff)<left_down_angle && qAsin(buff)>
            left_up_angle)

```



```

13     {
14         double lh = dlina(left_down_angle , left_down_dlina , eps_left
            , qAsin(buff));
15         left_values[address(lh)].i++;
16         left_values[address(lh)].vec.push_back(lh);
17     }
18 }

```

Листинг 14. Тело программы

После исполнения программы данные из массивов записываются в файл (см. листинг 15):

```

1     QFile left_file(filename);
2     QTextStream left_data(&left_file);
3     for (const auto& amount:left_values) {
4         left_data << amount.second.i*2*r*pow(10,6)/10 << "\n";
5     }
6     left_data << "sep" << "\n" << "\n";
7     for (const auto& amount:right_values) {
8         left_data << amount.second.i*2*r*pow(10,6)/10 << "\n";
9     }
10    left_file.close();

```

Листинг 15. Запись данных в файл

## 5.4 Анализ технологического процесса

### 5.4.1 Краткое описание технологического процесса

#### 5.4.1.1 Схема

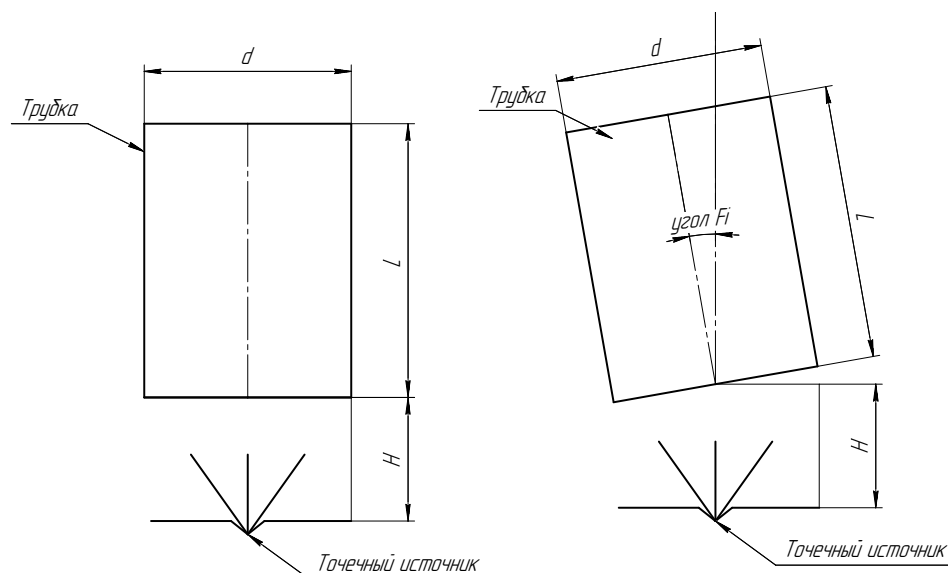


Рис. 24. Расчётная схема

#### 5.4.1.2 Назначение процесса

Тонкопленочные покрытия широко и многообразно применяются в различных сферах деятельности человека. В данном эксперименте на внутреннюю часть тонкой трубки наносился тонкий слой меди.

#### 5.4.1.3 Сущность процесса

Формирование тонкопленочных покрытий методом термического испарения осуществляется при сообщении материалу вещества энергии, которая затрачивается на его нагрев. С увеличением температуры колебательная энергия частиц возрастает и становится больше энергии связи с другими частицами, в результате чего происходит их испускание (испарение) и дальнейшая конденсация на поверхности изделия.

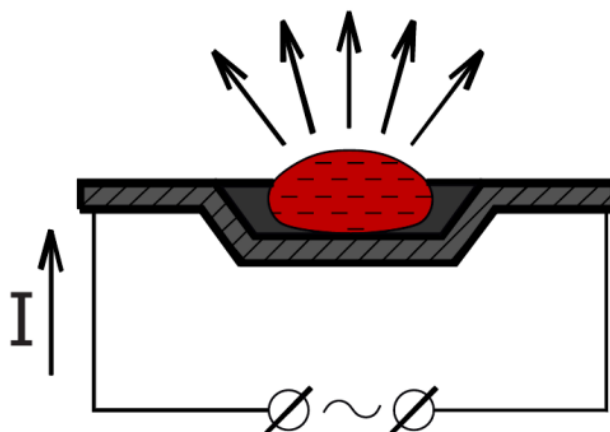


Рис. 25. Схема резистивного термического испарения

При резистивном испарении материал помещается в проводник с высоким электрическим сопротивлением (испаритель). При протекании тока через испаритель происходит его нагрев и передача тепла испаряемому веществу.

#### **5.4.1.4 Этапы**

##### **5.4.1.4.1 При моделировании эксперимента**

1. Ввод необходимых параметров в исходный код программы;
2. Компиляция и запуск исполняемого файла;
3. Получение результатов в виде текстового файла.

##### **5.4.1.4.2 При проведении эксперимента на практике**

1. Подготовка заготовки:
  - Резка исходной трубы на трубы заданных параметров.
2. Очистка заготовки.
3. Нанесение медного покрытия:
  - 3.1. Установка трубки в держатель;

- 3.2. Откачка газа из камеры;
- 3.3. Установка параметров для термического испарения;
- 3.4. Нанесение покрытия;
- 3.5. Остановка насосов, напуск атмосферы;
- 3.6. Изъятие заготовки из камеры.

#### **5.4.1.5 Используемое оборудование**

Эксперименты проводились в рамках НИР в лаборатории кафедры «Электронные технологии в машиностроении». Процесс нанесения покрытия был реализован при помощи специально разработанного программного обеспечения, моделирующее нанесение покрытия методом магнетронного распыления.

#### **5.4.2 Сравнительный анализ входных и выходных параметров, присущих данной операции**

##### **1. Входные контролируемые и управляемые факторы:**

- Расстояние от источника до заготовки;
- Время нанесения;
- Угол наклона заготовки относительно источника;
- Внутренний диаметр трубки.

##### **2. Входные контролируемые и неуправляемые:**

- Чистота заготовки;
- Чистота распыляемого материала.

##### **3. Входные неконтролируемые и неуправляемые:**

- Квалификация оператора;

- Состояние оператора;
- Калибровка и настройка инструментов измерения.

#### 4. Выходные параметры:

- Отношение толщин нанесённого покрытия на противоположных сторонах стенки (отношение толщины на левой проекции к толщине покрытия на правой).

##### Входные к.у.

- Расстояние от источника до заготовки
- Время нанесения
- Угол наклона
- Внутренний диаметр

##### Входные к.ну.

- Чистота заготовки
- Чистота распыляемого материала

##### Входные нк.ну.

- Квалификация оператора
- Состояние оператора
- Калибровка и настройка инструментов измерения

**Выходной параметр:**  
Отношение толщин на  
противоположных  
проекциях

Рис. 26. Схема действия факторов

### 5.4.3 Выбор наиболее существенных входных и выходных параметров

#### 5.4.3.1 Входные параметры

Поскольку задачей эксперимента является нахождение зависимости отношения толщин (4) напылённого материала от диаметра трубки и угла её наклона относительно источника, в качестве входных параметров были выбраны:

- Диаметр трубки;
- Угол наклона оси трубки относительно нормали к источнику.

### 5.4.3.2 Выходные параметры

Из-за наклона трубки толщина покрытия на противоположных сторонах будет отличаться, в связи с чем предлагается в качестве выходного параметра взять:

- Отношение толщин покрытий на противоположных стенках (какую долю от толщины покрытия на правой стороне составляет толщина покрытия на левой).

## 5.5 Построение схемы контроля

Моделирование будет производиться на компьютере и контроль будет осуществляться на нём.

1. Цилиндр высотой  $L$  разделяется на 10 равных участков, считается число попавших атомов;
2. Вычисляется средняя толщина покрытия на участке по формуле:

$$\delta = \frac{a \cdot d}{\frac{L}{10}}, \text{ где} \quad (18)$$

- $a$  — количество попавших в данный интервал молекул;
- $\frac{L}{10}$  — длина данного интервала.

## 5.6 Проведение математического моделирования технологического процесса

### 5.6.1 Обоснование необходимости проведения процесса

Основной целью проведения эксперимента является разработка математической модели, адекватно описывающей влияние параметров процесса нанесения покрытий ИТО на значение коэффициента отражения. Проведение моделирования выбранного процесса необходимо для того, чтобы определить какие

из входных факторов наиболее существенно влияют на выходной параметр, а какие влияют на выходной параметр в меньшей степени. Математическое описание процесса, обычно, представляется в виде полинома:

$$Y = b_0 + \sum_{j=1}^k b_j X_j + \sum_{j \neq u}^k b_{ju} X_j X_l + \sum_{j=1}^k b_j X_j^2 + \dots, \text{ где} \quad (19)$$

—  $X$  – факторы эксперимента;

—  $Y$  – функция отклика.

### 5.6.2 Разработка плана эксперимента

1. В качестве метода исследования процесса выбран полный факторный эксперимент (ПФЭ). В этом случае учитывается влияние на функцию отклика исследуемого процесса не только каждого рассматриваемого в эксперименте фактора в отдельности, но и их взаимодействий.
2. Планирование эксперимента начнём с предположения о том, что модель имеет вид полинома первого порядка:

$$Y = b_0 + \sum_{j=1}^k b_j X_j + \sum_{j \neq u}^k b_{ju} X_j X_l \quad (20)$$

3. Определение числа опытов:

$$N = u^k = (1 + 1)^2 = 4, \text{ где} \quad (21)$$

—  $u$  – число, на единицу большее порядка полинома

—  $k$  – число исследуемых факторов

4. Для линейной модели и 2 факторов достаточно будет провести 4 опыта, модель будет иметь вид:

$$Y = b_0 + b_1 X_1 + b_2 X_2 + b_{12} X_1 X_2, \text{ где} \quad (22)$$

- $b_0$  – значение функции отклика в центре плана;
- $b_1, b_2$  – характеристика степени влияния соответствующих факторов на функцию отклика;
- $b_{12}$  – характеристика влияния взаимодействия факторов.

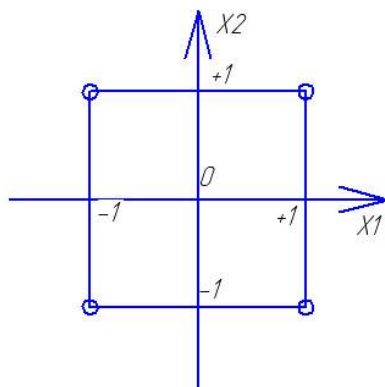


Рис. 27. Геометрическое изображение экспериментальных точек ПФЭ

5. Были выбраны следующие диапазоны варьирования факторов:

Уровень	Угол наклона оси трубки, град	Диаметр трубки, мм	В безразмерной системе координат
Верхний	6	30	+1
Нижний	4	20	–1

Таблица 1. Диапазоны варьирования входных факторов

6. В качестве центра плана принимается центр исследуемой области. Значения входных параметров в центре плана:

- Угол наклона оси трубки  $\varphi = 5^\circ$ ;
- Диаметр трубки  $d = 25$  мм.

7. Выходной параметр по 5.4.3.2 измеряется в процентах.



### **5.6.3 Построение математической модели**

Матрица планирования эксперименты имеет вид:

№	$x_1$	$x_2$	$X_0$	$X_1$	$X_2$	$X_1X_2$	$Y_1$	$Y_2$	$Y_3$	$Y_4$	$Y_5$	$Y_6$	$Y_7$	$Y_8$	$Y_9$	$Y_{10}$	$\bar{Y}_i$	$S_i^2$
1	4	20	1	-1	-1	1	86.17	84.34	82.82	80.43	75.6	75.52	74.92	74.8	75.35	75.53	78.55	19.75
2	6	20	1	1	-1	-1	78.66	71.49	70.17	68.43	65.82	65.12	66.66	64.85	65.85	66.02	68.31	18.12
3	4	30	1	-1	1	-1	88.3	86.75	85.26	82.65	82.31	82.13	82.08	82.62	82.14	82.79	83.7	5.05
4	6	30	1	1	1	1	82.83	78.04	75.93	75	74.52	74.92	73.99	74.5	74.01	74.86	75.86	7.39

Таблица 2. Матрица планирования ПФЭ

№	$x_1$	$x_2$	$X_0$	$X_1$	$X_2$	$X_1X_2$	$Y_1$	$Y_2$	$Y_3$	$Y_4$	$Y_5$	$Y_6$	$Y_7$	$Y_8$	$Y_9$	$Y_{10}$	$\bar{Y}_i$	$S_i^2$
5	6	27.5	1	1	0.5	0.5	81.01	76.09	74.10	72.98	73.33	72.34	72.81	72.11	73.02	73.14	74.09	7.15

Таблица 3. Дополнительный опыт для определения дисперсии адекватности

Для исключения влияния систематических погрешностей эксперимент проводился в случайном порядке.

## 5.6.4 Обработка результатов эксперимента

### 5.6.4.1 Выборочное среднее и дисперсия

Поскольку в каждом опыте было проведено 10 параллельных наблюдений (отношение толщин на десяти интервалах), определим:

#### 1. Выборочное среднее:

$$\bar{Y}_i = \frac{\sum_{l=1}^n Y_{il}}{n} \quad (23)$$

#### 2. Выборочная дисперсия (число степеней свободы определяется число параллельных наблюдений в каждом опыте):

$$S_i^2 = \sum_{l=1}^n \frac{(Y_{il} - \bar{Y}_i)^2}{n - 1} \quad (24)$$

### 5.6.4.2 Проверка эксперимента на воспроизводимость

Проверку на воспроизводимость определим по критерию Кохрена:

#### 1. Экспериментальное значение критерия Кохрена:

$$G_9 = \frac{\max(S_i^2)}{\sum_{i=1}^N S_i^2} = \frac{19.75}{19.75 + 18.12 + 5.05 + 7.39} = 0.39 \quad (25)$$

#### 2. Критическое значение критерия Кохрена:

$$G_{кр} = G(\beta = 0.05, N = 4, n = 10) = 0.5, \text{ где} \quad (26)$$

- $\beta$  – уровень значимости;
- $N$  – число проведённых опытов;
- $n$  – число параллельных наблюдений.

### 3. Сравнение экспериментального и критического:

$$G_{\text{э}} < G_{\text{кр}} \quad (27)$$

Таким образом дисперсии являются однородными, а эксперимент – воспроизводимым.

#### 5.6.4.3 Определение коэффициентов полинома

Коэффициенты полинома вычисляются по формуле:

$$b_j = \frac{\sum_{i=1}^N X_{ji} \bar{Y}_i}{N} \quad (28)$$

$b_0$	$b_1$	$b_2$	$b_{12}$
76.61	−4.52	3.18	0.60

Таблица 4. Значения коэффициентов полинома

#### 5.6.4.4 Оценка значимости коэффициентов

Незначимость коэффициента может быть вызвана следующими причинами:

- интервал варьирования соответствующей переменной мал,;
- уровень базового режима по данной переменной близок к точке частного экстремума;
- данный фактор не влияет на функцию отклика.

Проведём оценку значимости коэффициентов:

1. **Дисперсия воспроизводимости** ((среднее арифметическое группы выборочных дисперсий (т.е. дисперсий функции отклика по каждому опыту))):

$$S^2(Y) = \frac{\sum_{i=1}^N S_i^2}{N} = 12.57 \quad (29)$$

2. **Дисперсия ошибки определения коэффициента:**

$$S^2(b_j) = \frac{S^2(Y)}{nN} = \frac{12.57}{10 \cdot 4} = 0.31 \quad (30)$$

3. **Оценка значимости по критерию Стьюдента**, значение которого рассчитывается по формуле:

$$t_j = \frac{|b_j - 0|}{\sqrt{S^2(b_j)}} \quad (31)$$

4. **Критическое значение критерия Стьюдента:**

$$t_{кр} = t(\beta = 0.05, \nu = N(n - 1)) = t(0.05, 36) = 2.03 \quad (32)$$

5. **Отбрасывание незначимых коэффициентов:**

	$b_0$	$b_1$	$b_2$	$b_{12}$
$t_j$	243.67	14.38	10.11	1.91
$t_{кр}$	2.03	2.03	2.03	2.03

Таблица 5. Критерий Стьюдента

Из табл. 5 отбрасываем  $b_{12}$  как незначимый.

6. **Окончательный вид модели.** Из (22) и табл. 5:

$$Y = 76.61 - 4.52X_1 + 3.18X_2 \quad (33)$$

#### 5.6.4.5 Проверка модели на адекватность

1. Для оценки адекватности модели необходимо определить **значения функции отклика** в каждом опыте согласно математической модели (33).

$\hat{Y}_1$	$\hat{Y}_2$	$\hat{Y}_3$	$\hat{Y}_4$
77.95	68.91	84.30	75.26

Таблица 6. Значения функции отклика, рассчитанные по математической модели

2. **Дисперсия адекватности** (оценка отклонения, предсказанного моделью значения выходного параметра (функции отклика) от результатов эксперимента в каждой точке факторного пространства):

$$S_{\text{ад}}^2 = \frac{\sum_{i=1}^N (\bar{Y}_i - \hat{Y}_i)^2}{N - \alpha_{\text{зн}}} n = 14.38, \text{ где} \quad (34)$$

—  $\alpha_{\text{зн}} = 3$  – число значимых коэффициентов в полиноме;

—  $n$  – число параллельных измерений повторений.

3. **Проверка на адекватность при помощи критерия Фишера.** Сравним дисперсию адекватности (34) с дисперсией воспроизводимости (29):

$$F_9 = \frac{\max(S_{\text{ад}}^2, S^2(Y))}{\min(S_{\text{ад}}^2, S^2(Y))} = \frac{S_{\text{ад}}^2}{S^2(Y)} = \frac{14.38}{12.57} = 1.14 \quad (35)$$

4. **Критическое значение критерия Фишера:**

$$F_{\text{кр}} = F(\beta, N(n-1), N - \alpha_{\text{зн}}) = F(0.05, 36, 1) = 4.11 \quad (36)$$

5. **Сравнение экспериментального и критического:**

$$F_9 < F_{\text{кр}} \Rightarrow \text{модель адекватна} \quad (37)$$

Таким образом, полученная модель имеет вид, показанный в (33)

## 5.7 Перерасчёт с другим значением критерия Стьюдента

1. В разделе 5.6.4.4 в п.5 коэффициент, отвечающий за взаимодействие факторов был отброшен в связи с выбором уровня значимости  $\beta = 0.05$ .

Однако, если принять уровень значимости  $\beta = 0.1$ , значения критерия Стьюдента будет:

$$t_{\text{кр}} = t(\beta = 0.1, \nu = N(n - 1)) = t(0.1, 36) = 1.69 \quad (38)$$

2. Мы получим **таблицу 5** в изменённом виде:

	$b_0$	$b_1$	$b_2$	$b_{12}$
$t_j$	243.67	14.38	10.11	1.91
$t_{\text{кр}}$	1.69	1.69	1.69	1.69

Таблица 7. Критерий Стьюдента с  $\beta = 0.1$

3. **Модель** примет вид, отличный от (33):

$$Y = 76.61 - 4.52X_1 + 3.18X_2 + 0.59X_1X_2 \quad (39)$$

4. Необходимо будет провести **дополнительный опыт**, поскольку число значимых коэффициентов  $\alpha_{\text{зн}}$  стало равно число проведённых опытов. Предлагается провести опыт в центре плана. Результаты опыта см. в таблице 3.

5. Таблица 6 примет вид:

$\hat{Y}_1$	$\hat{Y}_2$	$\hat{Y}_3$	$\hat{Y}_4$	$\hat{Y}_5$
78.54	68.31	83.70	75.86	73.97

Таблица 8. Значения функции отклика, рассчитанные по математической модели

6. **Дисперсия адекватности:**

$$S_{\text{ад}}^2 = \frac{\sum_{i=1}^N (\bar{Y}_i - \hat{Y}_i)^2}{N - \alpha_{\text{зн}}} n = 0.15 \quad (40)$$

Дисперсия адекватности получилась меньше, чем в (34), поскольку значения выходных параметров во всех точках кроме центральной полученные по модели и по опыту совпали.

7. **Проверка на адекватность** (проверка с дисперсией воспроизводимости (29)):

$$S_{ад}^2 < S^2(Y) \Rightarrow \text{модель адекватна} \quad (41)$$

## 5.8 Выводы

### 5.8.1 Выводы по модели с 3 значимыми коэффициентами

Сравнение дисперсии адекватности и дисперсии воспроизводимости показало, что полученная математическая модель адекватно описывает процесс нанесения покрытий на внутреннюю часть трубки.

Проанализировав полученную модель, можно сделать следующие выводы:

1. Угол наклона оси трубки и её диаметр влияют на отношение толщин покрытий на противоположащих стенках;
2. Влияние фактора 1 (угол наклона  $\varphi$ ) более значимо, чем влияние фактора 2 (диаметр трубки  $d$ );
3. Выявлено, что эффект взаимодействия двух факторов является незначимым (эффект взаимодействия был бы значимым при другом значении уровня значимости).

### 5.8.2 Выводы по модели с 4 значимыми коэффициентами

Полученная математическая модель адекватно описывает зависимость, отношения толщин на левой и правой проекции трубки. Значение дисперсии



адекватности, полученной с 3 значимыми коэффициентами (34) получилось больше, чем в случае с 4 значимыми коэффициентами (40).

## 6 Дальнейшая работа

Дальнейшим направлением работы является усовершенствование программы обмена данными между устройствами и осуществление автоматизации оборудования по протоколу *Modbus TCP*, поскольку дистанционное автоматизированное управление позволяет минимизировать такие проблемы, как:

- простой оборудования в связи с отсутствием оператора;
- уменьшение производительности в связи с необходимостью ручного управления;
- ухудшение качества продукции в связи с неточным следованиям инструкциям.

Результатом работы может стать создание SCADA системы для управления вакуумным оборудованием.

## ВЫВОДЫ И РЕЗУЛЬТАТЫ

1. Был изучен выбранный по результатам литературного обзора протокол передачи данных между автоматизируемым оборудованием;
2. Была создана программа обмена сигналами между двумя объектами класса с использованием встроенного в библиотеки Qt функционала сигналов и слотов;
3. Разработаны приложения клиента и сервера для обмена данными между устройствами или между человеком и оборудованием по сети *TCP*;
4. Написана программа для моделирования нанесения покрытий на внутреннюю часть тонкой трубки методом Монте-Карло с использованием библиотек проекта Qt;
5. По результатам моделирования была составлена математическая модель отношения толщин наносимого покрытия;
6. Математическая модель, полученная в результате анализа данных по методике, описанной в курсе «Техника эксперимента в электронике и нанoeлектронике» была проверена на адекватность;
7. Благодаря проведению моделирования было установлено, что угол наклона трубки и её диаметр на отношение толщин нанесенного покрытия.

## ЗАКЛЮЧЕНИЕ

По результатам работы были изучены различные протоколы, используемые для связи оборудования в сетях, а также промышленные протоколы.

Было выбрано наиболее оптимальное сочетание протоколов: *Modbus* и *TCP*.

Были разработаны программы для отправки и принятия сигналов и данных от различного оборудования, а также проведено моделирование нанесения покрытия на внутреннюю часть тонких трубок по методу Монте-Карло.

Дальнейшим направлением работы является усовершенствование программы обмена данными между устройствами и осуществление автоматизации оборудования по протоколу *Modbus TCP*, поскольку дистанционное автоматизированное управление позволяет минимизировать такие проблемы, как:

- простой оборудования в связи с отсутствием оператора;
- уменьшение производительности в связи с необходимостью ручного управления;
- ухудшение качества продукции в связи с неточным следованиям инструкциям.

## Список литературы

1. *Kumar S., Rai S.* Survey on Transport Layer Protocols: TCP & UDP // International Journal of Computer Applications. — 2012. — 1 мая. — Т. 46, № 7. — С. 6. — URL: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.734.7346&rep=rep1&type=pdf> (дата обр. 10.04.2021).
2. *Promwad.* Обзор современных протоколов в системах промавтоматики. — 31.10.2019. — URL: <https://habr.com/ru/post/473992/> (дата обр. 16.04.2021).
3. *Денисенко В.* Протоколы и сети Modbus и Modbus TCP // Современные технологии автоматизации. — 2010. — Т. 4. — С. 90—94. — URL: <https://www.reallab.ru/images/editor/downloads/articles/Modbus.pdf> (дата обр. 12.04.2021).
4. *Евдокимов Д. А., Ярыш Р. Ф.* Modbus - Протокол Передачи Данных //. — Наука и Просвещение, 2021. — С. 66—68. — URL: <https://www.elibrary.ru/item.asp?id=44684618> (дата обр. 12.04.2021).
5. *IoT А.* Как общаются машины: протокол Modbus. — 06.05.2019. — URL: <https://habr.com/ru/company/advantech/blog/450234/> (дата обр. 12.04.2021).
6. *Юшин О. Е., Кудрявцев Д. Н., Концов А. В.* ОБЗОР ТЕХНОЛОГИЙ В INDUSTRIAL ETHERNET // Инноватика-2018. — 2018. — С. 532—534. — URL: <https://elibrary.ru/item.asp?id=35627705> (дата обр. 11.04.2021).
7. *Swales A.* Open modbus/tcp specification // Schneider Electric. — 1999. — Т. 29. — С. 25. — URL: [http://www.dankohn.info/projects/Fieldpoint\\_module/Open\\_ModbusTCP\\_Standard.pdf](http://www.dankohn.info/projects/Fieldpoint_module/Open_ModbusTCP_Standard.pdf) (дата обр. 10.04.2021).

8. Индивидуальный лабораторный практикум. Часть 1 / А. Беликов [и др.] ; под ред. Ю. Панфилова. — Издательство МГТУ им. Н.Э. Баумана, 2018. — 80 с. — (Дата обр. 10.04.2021).
9. *Шлее М.* Qt 5.10. Профессиональное программирование на C++. — БХВ-Петербург, 2018. — 1072 с. — (В подлиннике). — ISBN 5-9775-3678-X.
10. *Панфилова Е.* Организация контроля и моделирование технологического процесса. — Москва, 2018.