

КАФЕДРА Системы обработки информации и управления

Локальная безадаптерная сеть

(Подпись, дата)

(И.О.Фамилия)

2021 г.

**Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ
Заведующий кафедрой ИУ5
(Индекс)
В.М. Черненко
(И.О.Фамилия)
« ____ » _____ 2021 г.

**ЗАДАНИЕ
на выполнение курсовой работы**

по дисциплине Сетевые технологии в АСОИУ

Студенты группы ИУ5-64Б

Алпеев Владислав Сергеевич, Калашникова Анастасия Викторовна
(Фамилия, имя, отчество)

Тема курсовой работы Локальная безадаптерная сеть

Направленность КР учебная

Источник тематики кафедра

График выполнения работы: 25% к ____ нед., 50% к ____ нед., 75% к ____ нед., 100% к ____ нед.

Задание Разработать протоколы взаимодействия объектов до прикладного уровня локальной сети, состоящей из 2-х ПК, соединенных нуль-модемно через интерфейс RS232C, и реализующей функцию передачи текста диалога абонентов. Принимаемый и передаваемый тексты отображать в разных окнах. Скорость обмена и параметры СОМ-порта выбирает пользователь одного из ПК. Передаваемую информацию защитить [15,11]-кодом Хэмминга.

Оформление курсовой работы:

Расчетно-пояснительная записка на 19 листах формата А4. Содержит:

Приложение 1: Техническое задание на 5 листах формата А4.

Приложение 2: Описание программы на 25 листах формата А4.

Приложение 3: Руководство пользователя на 6 листах формата А4.

Приложение 4: Программа и методика испытаний на 9 листах формата А4.

Приложение 5: Графическая часть на 7 листах формата А4.

Дата выдачи задания « ____ » _____ 2021 г.

Руководитель курсовой работы

В.А. Галкин
(Подпись, дата) (И.О.Фамилия)

Студент

В.С. Алпеев
(Подпись, дата) (И.О.Фамилия)

Студент

А.В. Калашникова
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

1. Введение	4
2. Требования к программе	4
3. Определение структуры программного продукта	4
4. Физический уровень.....	4
4.1. Функции физического уровня.....	4
4.2. Описание физического уровня.....	5
4.3. Нуль-модемный интерфейс	8
4.4. Настройка COM-порта средствами библиотеки jSerialComm	9
4.4.1. Описание класса SerialPort	9
4.4.2. Описание класса SerialPortEvent	12
4.4.3. Описание класса SerialPortDataListener	12
5. Канальный уровень.....	13
5.1. Функции канального уровня	13
5.2. Протокол связи	13
5.3. Защита передаваемой информации	14
5.4. Формат кадров	15
5.4.1. Информационные кадры (I).....	15
5.4.2. Супервизорные кадры.....	16
6. Прикладной уровень	16
6.1. Функции прикладного уровня	16
6.2. Окно настроек	16
6.3. Окна чата	18

1. Введение

Данная программа, «Chat», выполненная в рамках курсовой работы по предмету «Сетевые технологии в АСОИУ», предназначена для организации обмена текстовыми сообщениями между соединёнными с помощью интерфейса RS232C компьютерами. Программа позволяет обмениваться 2-ум компьютерам, соединенным через COM-порты, текстовыми сообщениями, при условии запуска этой программы на обоих компьютерах.

2. Требования к программе

К программе предъявляются следующие требования. Программа должна:

- 1) устанавливать соединение между компьютерами и контролировать его целостность;
- 2) обеспечивать правильность передачи и приема данных с помощью кодирования пакета алгоритмом Хэмминга;
- 3) обеспечивать функцию передачи сообщений;
- 4) выполняться под управлением MS Windows 7/8/10. Было решено выполнить реализацию программы с помощью среды разработки IntelliJ IDEA.

3. Определение структуры программного продукта

При взаимодействии компьютеров между собой выделяются несколько уровней: нижний уровень должен обеспечивать соединение компьютера со средой передачи, а верхний – обеспечить интерфейс пользователя. Программа разбивается на три уровня: физический, канальный и прикладной (см. Приложение «Структурная схема программы»).

- Физический уровень предназначен для сопряжения компьютера со средой передачи.
- Канальный уровень предназначен для установки и поддержания соединения, формирования и проверки пакетов обмена протоколов верхних модулей.
- Прикладной уровень предназначен для выполнения задач программы.

4. Физический уровень

4.1. Функции физического уровня

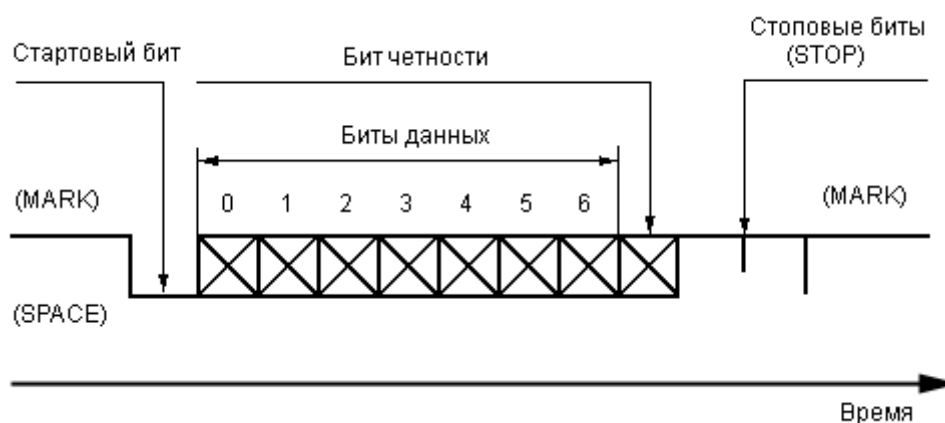
Основными функциями физического уровня являются:

- 1) Задание параметров COM-порта.
- 2) Установление физического канала.
- 3) Разъединение физического канала.
- 4) Передача информации из буфера в интерфейс.

5) Прием информации и ее накопление в буфере.

4.2. Описание физического уровня

Последовательная передача данных означает, что данные передаются по единственной линии. При этом биты байта данных передаются по очереди с использованием одного провода. Для синхронизации группе битов данных обычно предшествует специальный *стартовый бит*, после группы битов следуют *бит проверки на четность* и один или два *стоповых бита*, как показано на рисунке 1. Иногда бит проверки на четность может отсутствовать.



(Рис. 1)

Из рисунка видно, что исходное состояние линии последовательной передачи данных – уровень логической 1. Это состояние линии называют отмеченным – **MARK**. Когда начинается передача данных, уровень линии переходит в 0. Это состояние линии называют пустым – **SPACE**. Если линия находится в таком состоянии больше определенного времени, считается, что линия перешла в состояние разрыва связи – **BREAK**.

Стартовый бит **START** сигнализирует о начале передачи данных. Далее передаются биты данных, вначале младшие, затем старшие.

Контрольный бит формируется на основе правила, которое создается при настройке передающего и принимающего устройства. Контрольный бит может быть установлен с контролем на четность, нечетность, иметь постоянное значение 1 либо отсутствовать совсем.

Если используется бит четности **P**, то передается и он. Бит четности имеет такое значение, чтобы в пакете битов общее количество единиц (или нулей) было четно или нечетно, в зависимости от установки регистров порта. Этот бит служит для

обнаружения ошибок, которые могут возникнуть при передаче данных из-за помех на линии. Приемное устройство заново вычисляет четность данных и сравнивает результат с принятым битом четности. Если четность не совпала, то считается, что данные переданы с ошибкой. Конечно, такой алгоритм не дает полноценной гарантии обнаружения ошибок. Так, если при передаче данных изменилось четное число битов, то четность сохраняется, и ошибка не будет обнаружена. Поэтому на практике применяют более сложные методы обнаружения ошибок.

В самом конце передаются один или два стоповых бита **STOP**, завершающих передачу байта. Затем до прихода следующего стартового бита линия снова переходит в состояние **MARK**.

Использование бита четности, стартовых и стоповых битов определяют формат передачи данных. Очевидно, что передатчик и приемник должны использовать один и тот же формат данных, иначе обмен будет невозможен.

Другая важная характеристика – скорость передачи. Она также должна быть одинаковой для передатчика и приемника.

Скорость изменения информативного параметра сигнала обычно измеряется в бодах.

Иногда используется другой термин – биты в секунду (bps). Здесь имеется в виду эффективная скорость передачи данных, без учета служебных битов.

Интерфейс RS232C описывает несимметричный интерфейс, работающий в режиме последовательного обмена двоичными данными. Интерфейс поддерживает как асинхронный, так и синхронный режимы работы.

Интерфейс называется несимметричным, если для всех цепей обмена интерфейса используется один общий возвратный провод – сигнальная «земля».

Интерфейсы 25-ти (DB25) или 9-ти (DB9) контактный разъем.

Наименование сигнала	Цепь	Номер контакта	
		DB25P	DB9S
DCD (Data Carrier Detect)	109	8	1
RD (Receive Data)	104	3	2
TD (Transmit Data)	103	2	3
DTR (Data Terminal Ready)	108	20	4
GND (Signal Ground)	102	7	5
DSR (Data Set Ready)	107	6	6
RTS (Request To Send)	105	4	7
CTS (Clear To Send)	106	5	8
RI (Ring Indicator)	125	22	9

В интерфейсе реализован биполярный потенциальный код на линиях между DTE и DCE. Напряжения сигналов в цепях обмена симметричны по отношению к уровню сигнальной «земли» и составляют не менее +3В для двоичного нуля и не более -3В для двоичной единицы.

Входы TD и RD используются устройствами DTE и DCE по-разному. DTE использует вход TD для передачи данных, а вход RD для приема данных. И наоборот, устройство DCE использует вход TD для приема, а вход RD для передачи данных. Поэтому для соединения двух DTE необходимо перекрестное соединение линий TD и RD в нуль-модемном кабеле.

Самый низкий уровень управления связью – подтверждение связи.

В начале сеанса связи компьютер (DTE) должен удостовериться, что модем (DCE) находится в рабочем состоянии. Для этой цели компьютер подает сигнал по линии DTR. В ответ модем подает сигнал по линии DSR. Затем, после вызова абонента, модем подает сигнал по линии DCD, чтобы сообщить компьютеру, что он произвел соединение с удаленной системой.

Более высокий уровень используется для управления потоком (скоростью обмена данными) и также реализуется аппаратно. Этот уровень необходим для того, чтобы предотвратить передачу большего числа данных, чем то, которое может быть обработано принимающей системой.

В полудуплексных соединениях DTE подает сигнал RTS, когда оно желает передать данные. DCE отвечает сигналом по линии CTS, когда оно готово, и DTE начинает передачу данных. До тех пор, пока оба сигнала RTS и CTS не примут активное состояние, только DCE может передавать данные.

Для решения всех этих проблем для соединения двух устройств типа DTE используется специальный нуль-модемный кабель.

4.3. Нуль-модемный интерфейс

Обмен сигналами между адаптером компьютера (DTE) и модемом (DCE) (или вторым компьютером, присоединенным к исходному посредством кабеля стандарта RS-232C) строится по стандартному сценарию, в котором каждый сигнал генерируется сторонами лишь после наступления определенных условий. Такая процедура обмена информацией называется запрос/ответным режимом, или “рукопожатием” (handshaking). Большинство из приведенных в таблице сигналов как раз и нужны для аппаратной реализации “рукопожатия” между адаптером и модемом.

Обмен сигналами между сторонами интерфейса RS-232C выглядит так:

- 1) Компьютер после включения питания и открытия COM-порта выставляет сигнал DTR, который удерживается активным. Если модем включен в электросеть и исправен, он отвечает компьютеру сигналом DSR. Этот сигнал служит подтверждением того, что DTR принят, и информирует компьютер о готовности модема к приему информации;
- 2) Если компьютер получил сигнал DSR и хочет передать данные, он выставляет сигнал RTS;
- 3) Если модем готов принимать данные, он отвечает сигналом CTS. Он служит для компьютера подтверждением того, что RTS получен модемом и модем готов принять данные от компьютера. С этого момента адаптер может бит за битом передавать информацию по линии TD;
- 4) Получив байт данных, модем может сбросить свой сигнал CTS, информируя компьютер о необходимости “притормозить” передачу следующего байта, например, из-за переполнения внутреннего буфера;

программа компьютера, обнаружив сброс CTS, прекращает передачу данных, ожидая повторного появления CTS.

Модем может передать данные в компьютер, когда он обнаружит несущую в линии и выставит сигнал — DCD. Программа компьютера, принимающая данные, обнаружив этот сигнал, читает приемный регистр, в который сдвиговый регистр “собрал” биты, принятые по линии приема данных RD. Когда для связи используются только приведенные в таблице данные, компьютер не может попросить модем “повременить” с передачей следующего байта. Как следствие, существует опасность переопределения помещенного ранее в приемном регистре байта данных вновь “собранным” байтом. Поэтому при приеме информации компьютер должен очень быстро освобождать приемный регистр адаптера. В полном наборе сигналов RS-232C есть линии, которые могут аппаратно «приостановить» модем.

Нуль-модемный интерфейс характерен для прямой связи компьютеров на небольшом расстоянии (длина кабеля до 15 метров). Для нормальной работы двух непосредственно соединенных компьютеров нуль-модемный кабель должен выполнять следующие соединения:

1. RI-1 + DSR-1 – DTR-2;
2. DTR-1 – RI-2 + DSR-2;
3. CD-1 – CTS-2 + RTS-2;
4. CTS-1 + RTS-1 – CD-2;
5. RD-1 – TD-2;
6. TD-1 – RD-2;
7. SG-1 – SG-2;

Знак ‘+’ обозначает соединение соответствующих контактов на одной стороне кабеля.

4.4. Настройка COM-порта средствами библиотеки jSerialComm

4.4.1. Описание класса SerialPort

Класс SerialPort дает возможность управления последовательными портами компьютера. Он определяет минимальную функциональность для работы с ними.

Поля класса:

EVEN_PARITY – дополнение до четности;

FLOW_CONTROL_CTS_ENABLED – управление потоком CTS включено;

FLOW_CONTROL_DISABLED – управление потоком выключено;

FLOW_CONTROL_DSR_ENABLED – управление потоком отправки данных (DSR) включено;

FLOW_CONTROL_DTR_ENABLED – управление потоком DTR включено;

FLOW_CONTROL_RTS_ENABLED – управление потоком DTR включено;

FLOW_CONTROL_XONXOFF_IN_ENABLED – XON/XOFF управление обменом данными;

FLOW_CONTROL_XONXOFF_OUT_ENABLED – XON/XOFF управление обменом данными;

LISTENING_EVENT_DATA_AVAILABLE

LISTENING_EVENT_DATA_RECEIVED

LISTENING_EVENT_DATA_WRITTEN

MARK_PARITY – бит четности всегда единица;

NO_PARITY – бит четности отсутствует;

ODD_PARITY – дополнение до нечетности;

ONE_POINT_FIVE_STOP_BITS – полтора стоп-бита;

ONE_STOP_BIT – один стоп-бит;

SPACE_PARITY – бит четности всегда ноль;

TIMEOUT_NONBLOCKING

TIMEOUT_READ_BLOCKING

TIMEOUT_READ_SEMI_BLOCKING

TIMEOUT_SCANNER

TIMEOUT_WRITE_BLOCKING

TWO_STOP_BITS – два стоп-бита.

Методы класса:

addDataListener(SerialPortDataListener listener) – добавляет SerialPortDataListener к интерфейсу COM-порта;

bytesAvailable() – возвращает количество доступных байт;

`clearDTR()` – сбрасывает состояние линии DTR на 0;

`clearRTS()` – сбрасывает состояние линии RTS на 0;

`closePort()` – закрыть COM-порт;

`getBaudRate()` – получение текущей скорости передачи COM-порта;

`static SerialPort[] getCommPorts()` – возвращает список всех доступных COM-портов на данном устройстве;

`int getDeviceReadBufferSize()` – возвращает размер принимающего буфера, используемого COM-портом;

`boolean getDSR()` – проверка состояния сигнала DSR;

`boolean getDTR()` – проверка состояния сигнала DTR;

`int getNumDataBits()` – получение текущего количества бит данных в слове;

`getNumStopBits()` – получение текущего количества стоп-битов в слове;

`int getParity()` – получение текущей схемы обнаружения ошибок четности;

`getPortDescription()` – получение описания порта, установленного самим устройством;

`boolean getRTS()` – проверка состояния сигнала RTS;

`isOpen()` – получение состояния порта;

`openPort()` – открытие порта;

`setDTR()` – изменение состояния сигнала DTR;

`int readBytes(byte[] buffer, long bytesToRead)` – считывание необработанных данных `bytesToRead` из COM-порта и сохранение их в буфер;

`void removeDataListener()` – удаляет прослушиватель данных COM-порта;

`boolean setBaudRate(int newBaudRate)` – устанавливает желаемую скорость передачи данных для COM-порта;

`boolean setComPortParameters(int newBaudRate, int newDataBits, int newStopBits, int newParity)` – установка всех параметров COM-порта одновременно;

`boolean setComPortTimeouts(int newTimeoutMode, int newReadTimeout, int newWriteTimeout)` – задание параметров тайм-аута чтения и записи COM-порта;

`boolean setFlowControl(int newFlowControlSettings)` - указание типа управления потоком;

`boolean setNumDataBits(int newDataBits)` – задание желаемого количества битов данных в слове;

`boolean setNumStopBits(int newStopBits)` – задание желаемого количества стоп-битов в слове;

`boolean setParity(int newParity)` – задание желаемой схемы обнаружения ошибок четности;

`setRTS()` – изменение состояния линии RTS;

`int writeBytes(byte[] buffer, long bytesToWrite)` – запись необработанных данных `bytesToWrite` из буфера в COM-порт;

`getOutputStream()` – возвращает указатель на входной буфер;

`getInputStream()` - возвращает указатель на выходной буфер.

4.4.2. Описание класса `SerialPortEvent`

Класс определяет возможные события, происходящие на COM-порте.

Поля класса:

CTS - состояние линии CTS (0 - ВЫКЛ, 1 - ВКЛ);

DSR - состояние линии DSR (0 - ВЫКЛ, 1 - ВКЛ);

ERR - маска ошибок.

Методы класса:

`int getEventType()` – возвращает тип события COM-порта;

`byte[] getReceivedData()` – возвращает все необработанные байты данных;

`SerialPort getSerialPort()` - возвращает COM-порт.

4.4.3. Описание класса `SerialPortDataListener`

Методы класса:

`int getListeningEvents()` - необходимо переопределить, чтобы вернуть одну или несколько требуемых констант события, для которых должен быть иницирован обратный вызов `serialEvent(SerialPortEvent)`.

`void serialEvent(SerialPortEvent event)` – вызывается всякий раз, когда происходит одно из событий COM-порта, указанных методом `getListeningEvents()`.

5. Канальный уровень

5.1. Функции канального уровня

На канальном уровне выполняются следующие функции:

- 1) Установление логического соединения;
- 2) Управление передачей кадров;
- 3) Обеспечение необходимой последовательности блоков данных, передаваемых через межуровневый интерфейс;
- 4) Контроль ошибок и ретрансмиссия;
- 5) Разрыв логического соединения.

5.2. Протокол связи

В основном протокол содержит набор соглашений или правил, которого должны придерживаться обе стороны связи для обеспечения получения и корректной интерпретации информации, передаваемой между двумя сторонами. Таким образом, помимо управления ошибками и потоком протокол связи регулирует также такие вопросы, как формат передаваемых данных – число бит на каждый элемент и тип используемой схемы кодирования, тип и порядок сообщений, подлежащих обмену для обеспечения (свободной от ошибок и дубликатов) передачи информации между двумя взаимодействующими сторонами.

Перед началом передачи данных требуется установить соединение между двумя сторонами, тем самым проверяется доступность приемного устройства и его готовность воспринимать данные. Для этого передающее устройство посылает специальную команду – запрос на соединение, и ожидает ее приема с другого COM-порта.

Сначала соединение устанавливается, используя параметры COM-порта по умолчанию. В случае, когда пользователь первого ПК меняет параметры, от первого ПК посылается кадр PRM второму ПК с новыми параметрами соединения. Таким образом, скорость обмена и параметры COM-порта выбирает пользователь одного ПК.

Также необходимо информировать пользователя о неисправностях в физическом канале, поэтому для поддержания логического соединения необходимо

предусмотреть специальный кадр, который непрерывно будет посылаться с одного компьютера на другой, сигнализируя тем самым, что логическое соединение активно. В протоколе этот кадр и кадр запроса на соединение может быть один и тот же.

5.3. Защита передаваемой информации

При передаче данных по линиям могут возникать ошибки, вызванные электрическими помехами, связанными, например, с шумами, порожденными коммутирующими элементами сети. Эти помехи могут вызвать множество ошибок в цепочке последовательных битов. Контроль ошибок осуществляется применением кода Хэмминга [15,11].

При декодировании кадра возможны ошибки. Если ошибка обнаружена, то отправляется кадр RET для сообщения о возникшей ошибке и необходимости повторения отправки последнего кадра.

Рассмотрим алгоритм кодирования Хэмминга.

Коды Хэмминга являются самоконтролирующимися кодами (позволяют автоматически обнаруживать ошибки при передаче данных). Для их построения достаточно приписать к каждому слову один добавочный (контрольный) двоичный разряд и выбрать цифру этого разряда так, чтобы общее количество единиц в изображении любого числа было, например, нечетным. Одиночная ошибка в каком-либо разряде передаваемого слова изменит четность общего количества единиц. Счетчики по модулю 2, подсчитывающие количество единиц, которые содержатся среди двоичных цифр числа, могут давать сигнал о наличии ошибок. При этом невозможно узнать, в каком именно разряде произошла ошибка, и, следовательно, нет возможности исправить её. Остаются незамеченными также ошибки, возникающие одновременно в четном количестве разрядов.

К корректирующим кодам относятся коды Хэмминга с кодовым расстоянием $d=3$. Для кодов Хэмминга выбрано предельное значение разрешенных кодовых комбинаций $N = 2n \cdot (1 + n) - 1$, а число информационных разрядов (k) определится как:

$$k = \log[2n \cdot (1 + n) - 1] = n - \log(n + 1).$$

Данное уравнение имеет целочисленные решения $k = 0, 1, 4, 11, 26, \dots$, которые и определяют соответствующие коды Хэмминга [3,1] - код, [7,4] - код, [15,11] - код и т.д.

Алгоритм кодирования: все номера позиций кода нумеруются в двоичной системе счисления, проверочные разряды размещаются в позициях кода, кратных целой степени двойки $2^0, 2^1, \dots, 2^{(r-1)}$, где r – число проверочных разрядов. Значение (i)-го проверочного разряда определяется как сумма по mod2 тех разрядов кода, в номере которых двоичный разряд с (i)-ым весом равен единице.

Алгоритм декодирования: вычисляется значение синдрома ошибки. Разрядность синдрома равна числу проверочных разрядов. Значение (i)-го разряда синдрома определяется как сумма по mod2 тех разрядов принятого кода, включая проверочные, в номере которых вес двоичного разряда совпадает с весом разряда синдрома.

5.4. Формат кадров

Кадры могут иметь одинаковую структуру:

Стартовый байт	Адрес получателя	Адрес отправителя	Тип кадра	Длина поля данных	Данные	Стоповый байт
1 байт	1 байт	1 байт	1 байт	1 байт	N байт	1 байт

Стартовый и стоповый байт – служат для определения начала и конца кадра. Для них принято значение 0xFF;

Адрес получателя – байт, содержащий адрес получателя кадра; адрес отправителя – байт, содержащий адрес отправителя кадра (адреса в системе назначаются динамически и лежат в промежутке от 0x01 до 0x7E);

Тип кадра – байт, содержащий код типа кадра;

Длина поля данных – необязательное поле, содержит длину поля Данные;

Данные – необязательное поле, содержит какие-либо данные, передаваемые в кадре.

5.4.1. Информационные кадры (I)

Может служить для передачи коротких сообщений между компьютерами. Поля Длина поля данных и Данные – присутствуют. Поле Данные содержит текст передаваемого короткого сообщения.

5.4.2. Супервизорные кадры

Эти кадры используются для передачи служебной информации и реализуют следующие функции канального уровня: установка и разрыв логического канала, подтверждение приема информационного кадра без ошибок, запрос на повторную передачу принятого с ошибкой кадра. Поля Длина поля данных и Данные отсутствуют.

Для передачи параметров СОМ-порта от одного пользователя другому используются супервизорные кадры с параметрами СОМ-порта в поле Данные.

6. Прикладной уровень

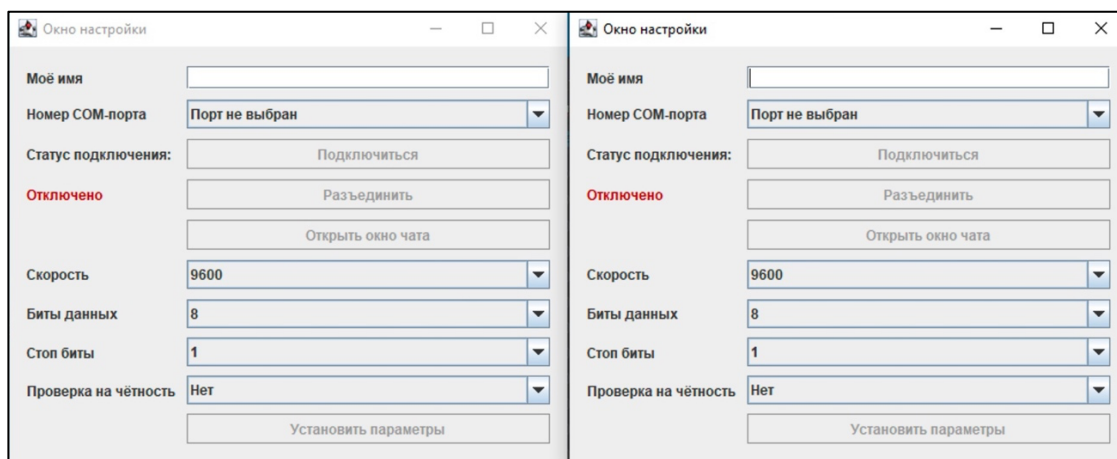
6.1. Функции прикладного уровня

На прикладном уровне выполняются следующие функции:

- 6) Обеспечение интерфейса с пользователем через систему форм;
- 7) Предоставление нижнему уровню текстового сообщения;
- 8) Выбор номера СОМ-порта для канала;
- 9) Установка параметров СОМ-порта;

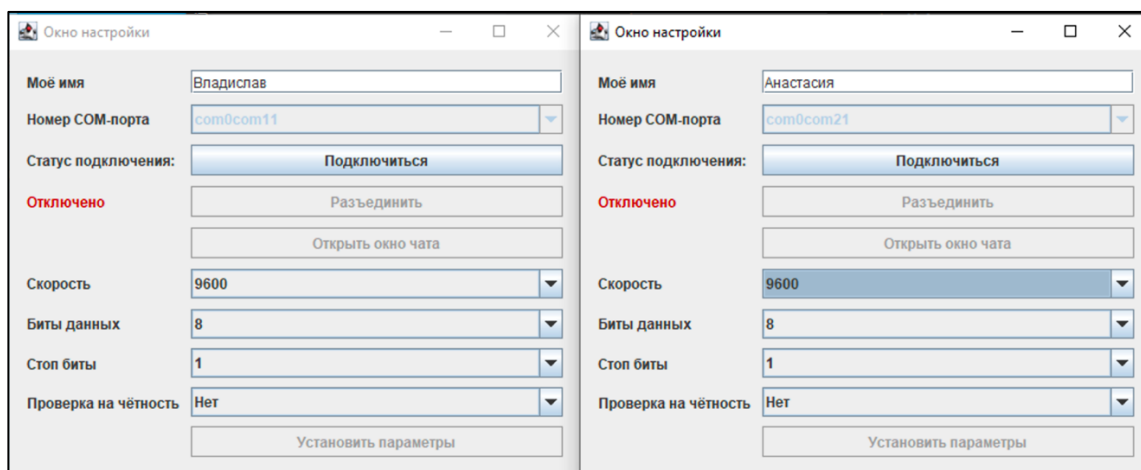
6.2. Окно настроек

При запуске программы появляется по одной форме настроек соединения на каждого пользователя. (см. Рис. 2)



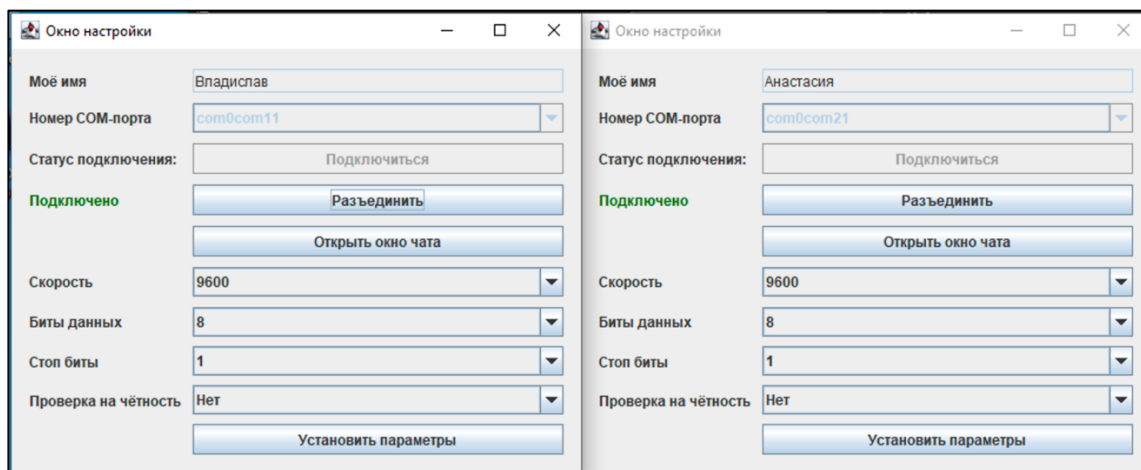
(Рис. 2)

Пользователям необходимо ввести имена и выбрать СОМ-порты для подключения. (см. Рис. 3)



(Рис. 3)

Чтобы установить соединение одному из пользователей нужно нажать кнопку «Подключиться». (см. Рис. 4)

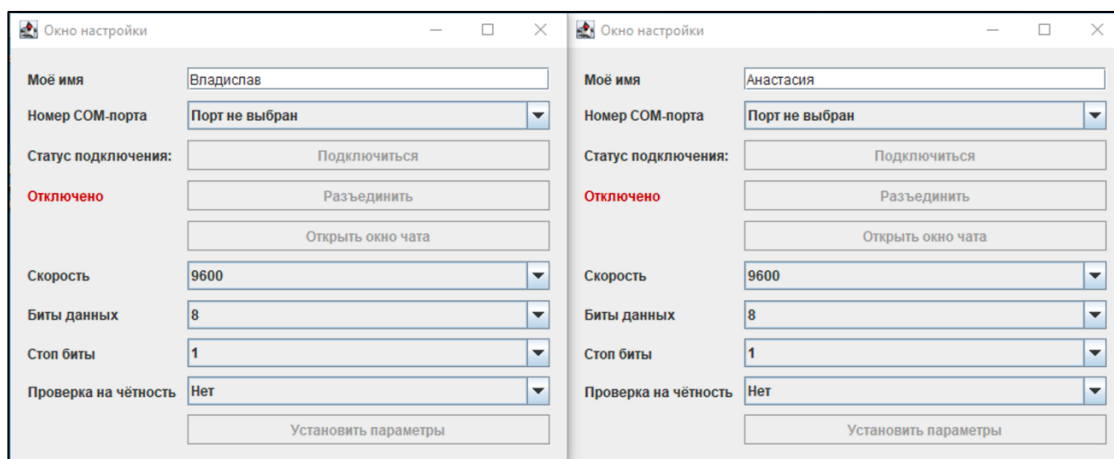


(Рис. 4)

В данном окне пользователи могут изменять скорость обмена, параметры COM-порта и фиксировать эти изменения с помощью кнопки «Установить параметры».

Из окна настроек доступно окно чата (п. 6.3)

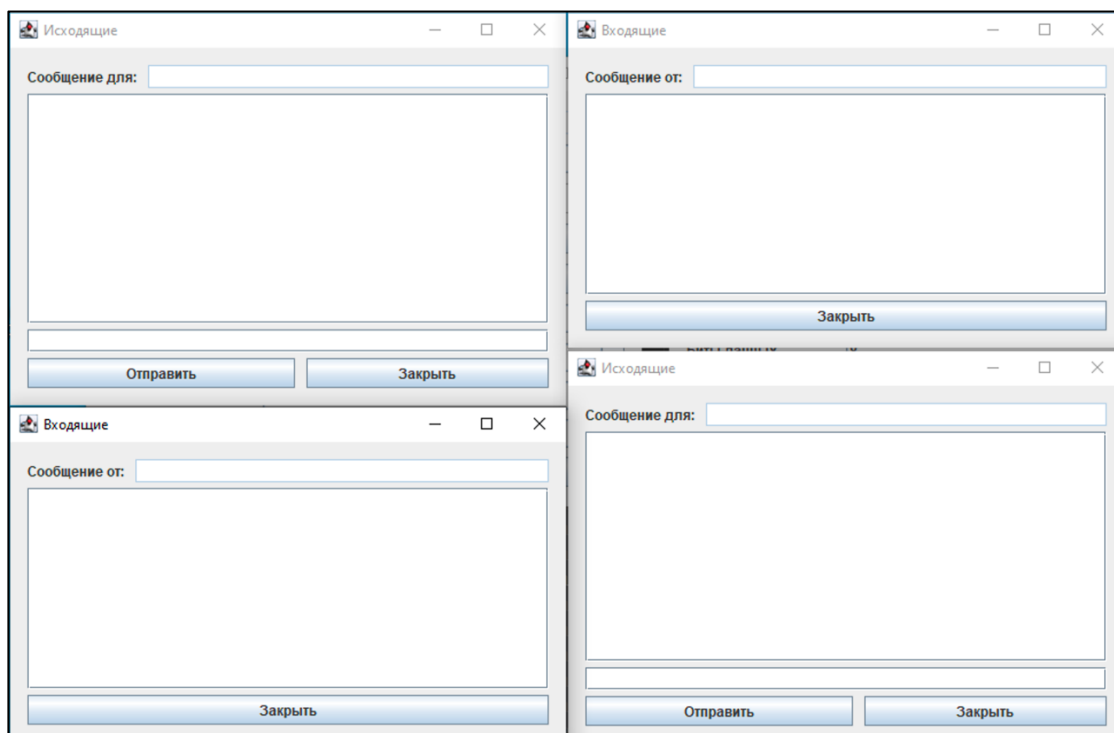
Чтобы прекратить соединение одному из пользователей нужно нажать кнопку «Разъединить». (см. Рис. 5)



(Рис. 5)

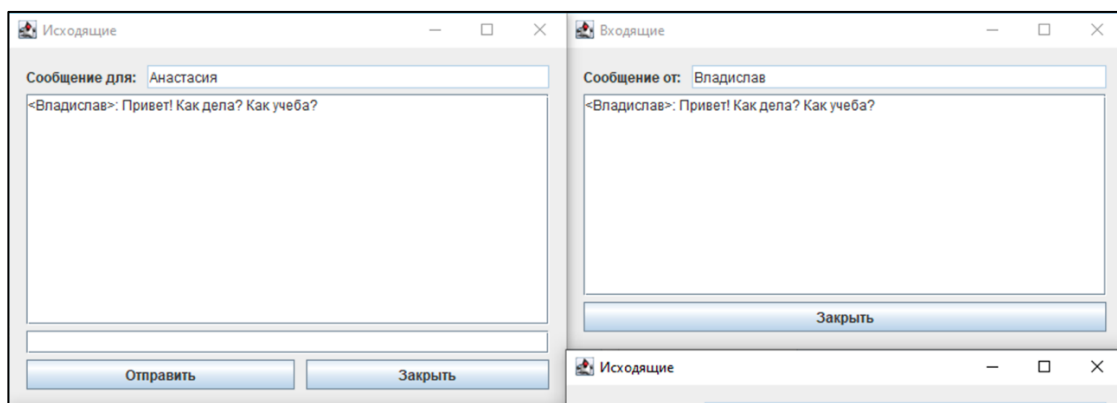
6.3. Окна чата

При нажатии на кнопку «Открыть окно чата» в окне настроек появятся две формы для пользователя: входящих и исходящих сообщений. (см. Рис. 6)



(Рис. 6)

В форме исходящих сообщений пользователь может набрать текст сообщения и отправить его с помощью кнопки «Отправить» формы «Исходящие». Тогда сообщение станет доступно в форме «Входящие» другого пользователя. (см. Рис. 7)



(Рис. 7)