**Московский государственный технический университет им. Н.Э.Баумана**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Утверждаю: |  |
| Галкин В.А. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2018 г. |

**Курсовая работа по дисциплине**

**«Сетевые технологии»**

**«Локальная безадаптерная сеть»**

Расчётно-пояснительная записка

(вид документа)

писчая бумага

(вид носителя)

17

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛИ: |  |
| студенты группы ИУ5-62 |  |
| Злобина С.В.  Кучаева К.И.  Заровная Н.А. | "\_\_\_"\_\_\_\_\_\_\_\_\_2018 г. |
|  |  |

Москва - 2018

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Содержание**

**Введение 3**

**Требования к программе 3**

**Определение структуры программного продукта 3**

**Физический уровень 3**

**Настройка COM-порта средствами Python 3.6 6**

**Канальный уровень. 7**

**Прикладной уровень………………………………………………………………………….12**

# **Введение**

Данная программа, выполненная в рамках курсовой работы по предмету «Сетевые технологии», предназначена для организации обмена файлов и текстовых сообщений между соединёнными с помощью интерфейса RS232C компьютерами. Программы позволяет обмениваться компьютерам, соединенным через COM-порты, текстовыми сообщениями и файлами.

# **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 алгоритма кодирования пакета с помощью кода Хэмминга [7,4];
3. Обеспечивать функцию передачи файлов.
4. Реализовать функцию просмотра содержимого файла
5. Реализовать управление темпом передачи

Программа выполняется под управлением OS Windows или Linux. Поэтому было решено выполнить реализацию программы с помощью среды разработки PyCharm на языке Python 3.6.

# **Определение структуры программного продукта**

При взаимодействии компьютеров между собой выделаются 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 занимается установлением и поддержанием соединения, формированием и проверкой пакетов обмена протоколов верхних модулей.

- прикладной уровень занимается выполнением задач программы.

# **Физический уровень**

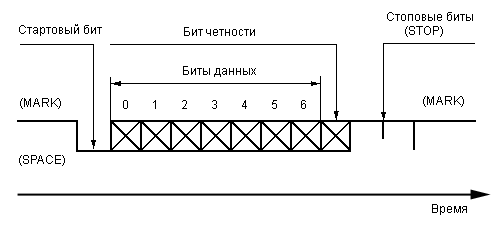
**Функции физического уровня**

Основными функциями физического уровня являются:

1. Установление физического канала.
2. Разъединение физического канала.
3. Передача информации из буфера в интерфейс.
4. Прием информации и накопление её в буфере.

**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*. Иногда бит проверки на четность может отсутствовать.



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 стопроцентной 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 данных. Она также должна быть одинаковой для передатчика и приемника.

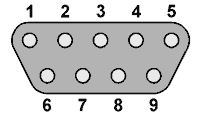
Скорость передачи данных обычно измеряется в бодах.

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.**

**Последовательная передача данных означает, что данные передаются по единственной линии. При этом биты байта данных передаются по очереди с использованием одного провода. Интерфейс называется несимметричным, если для всех цепей обмена интерфейса используется один общий возвратный провод – сигнальная «земля».**

**Интерфейс 9-ти контактный разъем**



**Рис. 3.** Назначение контактов разъема DB9.

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер контакта** | **Обозначение** | **Назначение** | **Обозначение CCITT** |
| 1 | DCD | Обнаружение несущей | 109 |
| 2 | RD | Принимаемые данные | 104 |
| 3 | TD | Отправляемые данные | 103 |
| 4 | DTR | Готовность терминала к работе | 108/2 |
| 5 | SG | Земля сигнала (схемная) | 102 |
| 6 | DSR | Готовность DCE | 107 |
| 7 | RTS | Запрос передачи | 105 |
| 8 | TD | Отправляемые данные | 103 |
| 9 | RI | Индикатор вызова | 125 |

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.

Каждый байт данных сопровождается специальными сигналами «старт» – стартовый бит и «стоп» – стоповый бит. Сигнал «старт» имеет продолжительность в один тактовый интервал, а сигнал «стоп» может длиться один, полтора или два такта.

При синхронной передаче данных через интерфейс передаются сигналы синхронизации, без которых компьютер не может правильно интерпретировать потенциальный код, поступающий по линии RD.

**Нуль-модемный интерфейс.**

Обмен сигналами между адаптером компьютера и модемом (или 2-м 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.

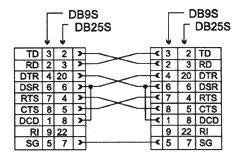


Рис. 2Полный нуль-модемный кабель.

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

1. Компьютер после включения питания выставляет сигнал **DTR**, который постоянно 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**.
5. Когда модему необходимо передать данные в компьютер, он (модем) выставляет сигнал на разъеме 8 – **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.

# **Настройка COM-порта средствами Python 3.6**

Пространство имен и библиотек TKINTER, SERIAL, RE и OS предлагают широкие возможности по настройке COM-порта, а также созданию и настройке интерфейсов для работы программы.

**Описание класса PhysicalLayer**

Этот класс используется для управления файловым ресурсом последовательного порта. Данный класс предоставляет возможности управления вводом-выводом в синхронном режиме или на основе событий, доступа к состоянию линии и состоянию разрыва, слежением за Time Out’ом, а также доступа к свойствам последовательного драйвера.

*Методы класса:*

|  |  |
| --- | --- |
| **Имя** | **Описание** |
|  |  |
| send\_bytes | Запись байтов строки bytes\_s в порт |
| receive\_byte | Чтение одного байта из порта, timeout – время ожидания байта в секундах, возвращение None, если байт не был считан |
| оreceive\_bytes | Чтение amount байт из порта, timeout - время ожидания байта в секундах, возвращает None, если байты не были считаны |
| close\_port | закрытие порта |
| open\_port | закрытие порта |
| set\_connection | закрытие порта |

# **Канальный уровень**

**Функции канального уровня**

На канальном уровне выполняются следующие функции:

1. Запрос логического соединения;
2. Разбивка данных на кадры;
3. Управление передачей кадров;
4. Обеспечение необходимой последовательности блоков данных, передаваемых через межуровневый интерфейс;
5. Контроль и обработка ошибок;
6. Проверка поддержания соединения;
7. Запрос на разъединение логического соединения.

**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 битов 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, сопровождаемую ответом приемного устройства, например о приеме или отклонении вызова.

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.

**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.

Метод четности/нечетности контрольная сумма блока не обеспечивают надежного обнаружения нескольких (например, двух) ошибок. Для этих случаев чаще всего применяется альтернативный метод, основанный на полиномиальных кодах. Полиномиальные коды используются в схемах покадровой (или поблочной) передачи. Это означает, что для каждого передаваемого кадра формируется (вырабатывается) один-единственный набор контрольных разрядов, значения которых зависят от фактического содержания кадра и присоединяются передатчиком к “хвосту” кадра. Приемник выполняет те же вычисления с полным содержимым кадра; если при передаче ошибки не возникли, то в результате вычислений должен быть получен заранее известный ответ. Если этот ответ не совпадает с ожидаемым, то это указывает на наличие ошибок.

Опишем кратко математический аппарат кодирования [7,4]-кодом Хэмминга.

**Формат кадров**

Кадры, передаваемые с помощью функций канального уровня, имеют различное назначение. Выделены супервизорные и информационные кадры.

**Служебные супервизорные кадры**

Эти кадры используются для передачи служебной информации и реализуют следующие функции канального уровня: установление и разъединение логического канала, подтверждение приема информационного кадра без ошибок, запрос на повторную передачу принятого с ошибкой кадра. Формат эти кадров:

|  |  |  |
| --- | --- | --- |
| StartByte | Type | StopByte |
| Флаг начала кадра | Тип супервизорного кадра | Флаг конца кадра |

**Супервизорные кадры передачи параметров**

Супервизорные кадры передачи параметров используются для синхронизации параметров COM-портов, как принимающего, так и отправляющего. Кадр данного типа формируется когда на одном из компьютеров изменяются параметры. Формат эти кадров:

|  |  |  |  |
| --- | --- | --- | --- |
| StartByte | Type | Data | StopByte |
| Флаг начала кадра | Тип супервизорного кадра | Параметры СОМ-порта | Флаг конца кадра |

**Информационные кадры**

Информационные кадры применяются для передачи закодированных кодом Хэмминга пользовательских сообщений. Формат эти кадров:

|  |  |  |  |
| --- | --- | --- | --- |
| StartByte | Type | Data | StopByte |
| Флаг начала кадра | Тип супервизорного кадра | Закодированные данные (текстовая строка) | Флаг  конца  кадра |

Кадр можно разделить на несколько блоков – флаг начала кадра, тип кадра, данные и флаг конца кадра.

Флаги начала и конца кадра представляют собой байты, с помощью которых программа выделяет кадр, определяя соответственно начало и конец кадра.

Поле типа кадра обеспечивает правильное определение и распознавание разновидностей кадров и обработки их соответствующими процедурами.

Данные представляют собой либо закодированную строку в информационном кадре или параметры порта в супервизорном кадре передачи параметров.

**Описание класса DataLinkLayer:**

Данный класс предназначен для работы с кадрами, отслеживает начало и конец кадров. Реализована функция для бит стаффинга. Реализованы функции по определению информационных кадров, положительных и отрицательных квитанций, об окончании передачи.   
  
**Методы класса:**

|  |  |
| --- | --- |
| **Имя** | **Описание** |
|  |  |
| check\_received | Проверяем, есть ли в физическом канале байты. Если да, то начинаем собирать кадр. В итоге в поле self.message может появиться принятое сообщение **:return**: Принятое сообщение |
| send\_msg | Отправляет сообщение, полученное извне в физический канал. **:param** msg: Сообщение для отправки **:return**: Булевое значение. Успешно передано сообщение или нет |
| receive\_msg | Запрос сообщения из канала. В отличие от check\_received мы точно знаем, что сообщение должно придти. После считывания, сообщение оказывается в поле self.message **:param** first\_frame: Первый кадр сообщения. Мог быть считан в других функциях **:return**: Полученное сообщение |
| \_send\_frames | Отправка кадров на физический уровень. **:param** frames: список кадров для отправки **:return**: |
| \_receive\_frame | Получение кадра из физического уровня **:return**: Словарь {'frame': кадр в виде строки байтов, 'broken': бинарное значение} |
| \_send\_ack | Отправление положительной квитанции **:return**: |
| \_send\_nak | Отправление отрицательной квитанции **:return**: |
| \_form\_frame | Формирование кадра **:param** f\_type: Тип отправляемого кадра **:param** data: Данные в кадре. Максимальная длина = 256 байт **:return**: кадр в виде строки байтов |
| \_deform\_frame | Расформирование кадров. Также проверяет кадр на наличие ошибок. **:param** frame: Кадр для расформировывания **:return**: кортеж ( тип\_кадра, данные\_в\_кадре) |
| \_cycle\_cipher | Напрямую в коде нигде не вызывается Функция циклического кодирования [7,4]. **:param** m: целое число в диапазоне от 0 до 15 включительно **:return**: целое число в диапазоне от 0 до 127 включительно |
| \_reminder | Вычисление "остатка" для алгоритма циклического кодирования **:param** x: Данные для кодирования. Целое число от 0 до 127 включительно **:param** y: Образующий полином **:return**: "Остаток" от деления x на y в формате целого числа |
| \_byte\_staff | Процедура байт-стаффинга **:param** x: строка для обработки **:return**: изменённая строка |
| \_debyte\_staff | Процедура, обратная байт-стаффингу **:param** x: строка для отменения байт-стаффинга **:return**: строка без байт-стаффа |

# **Прикладной уровень**

**Описание класса AppLayer:**

**Функции прикладного уровня:**

1. Обеспечивают интерфейс программы с пользователем через систему форм и меню
2. Предоставление нижнему уровню текстового файла.
3. Обеспечение вывода принятого файла в окно диалога пользователей.
4. Управление темпом передачи

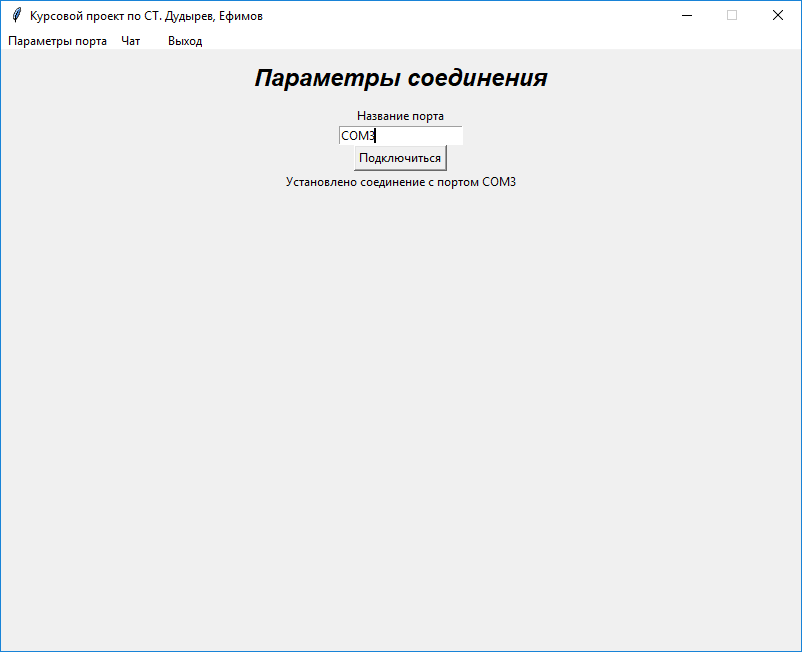
**Методы класса:**

|  |  |
| --- | --- |
| **Имя** | **Описание** |
|  |  |
| check\_received | Проверка, было ли получено какое-либо сообщение на канальном уровне. **:return**: кортеж (отправитель сообщения "А" или "В", само сообщениие в utf-8) |
| send\_file\_propose | Отправление сообщения с предложением принять файл. **:param** fname: абсолютное имя файла **:return**: |
| receive\_file\_proposal | Получение и обработка предложения принять файл. С помощью исключения FileProposal сообщение "Принять файл или отказаться" передаётся в интерфейс приложения **:param** bytes\_str: Строка с предложением **:return**: |
| send\_file\_ack | Передача сообщения с согласием принять файл. **:param** fname: Абсолютное имя файла **:return**: |
| send\_file\_nak | Передача сообщения с отказом принять файл **:param** fname: Абсолютное имя файла **:return**: |
| send\_msg | Передача "сообщения", в смысле сообщения в чате **:param** msg: текст "сообщения" **:return**: |
| receive\_msg | Получение "сообщения", в смысле сообщения в чате **:param** bytes\_str: данные из канального уровня **:return**: кортеж (отправитель, текст сообщения) |
| send\_file | Отправка файла через сеть. **:param** bytes\_str: строка с согласием или отказом принять файл от другого пользователя **:return**: кортеж (отправитель "А", сообщение об отправке файла) |
| receive\_file | Получение файла из сети. **:param** bytes\_str: Строка из канального уровня. **:return**: кортеж (отправитель "В", сообщение о получении файла) |
| set\_connection | Установка соединения с заданным портом **:param** port\_name: имя порта для подключения **:return**: результат подключения. None - если всё хорошо |
| short\_fname | Генерация "короткого" относительного пути файла **:param** fname: абсолютный путь к файлу **:return**: относительный путь к файлу |
| \_send\_message | Общий алгоритм для отправки сообщения через сеть. **:param** msg\_type: тип сообщения **:param** fname: абсолютный путь к файлу, если требуется **:param** data: данные для передачи, если требуются **:return**: |
| \_form\_message | Обобщённое формирование сообщения на основе его типа и содержания **:param** msg\_type: тип сообщения **:param** data: данные сообщения, если требуется **:param** fname: абсолютное имя файла, если требуется **:return**: сформированное сообщение для отправки через сеть |
| \_deform\_message | Извлечение полезной информации из сообщения **:param** message: исходное сообщение, полученное процедурой \_form\_message **:return**: словарь с полями: msg\_type - тип сообщения  fname - название файла (опционально)  data - данные файла (опционально)  msg - содержимое сообщения из чата (опционально) |
| **class** FileNotAcknowledged | Особое исключение, если собеседник из чата отказывается принимать файл. Используется для передачи этой информации в интерфейс программы. |
| **class** FileProposal | Особое исключение, если собеседник из чата предлагает получить файл. Используется для передачи этой информации в интерфейс программы. |

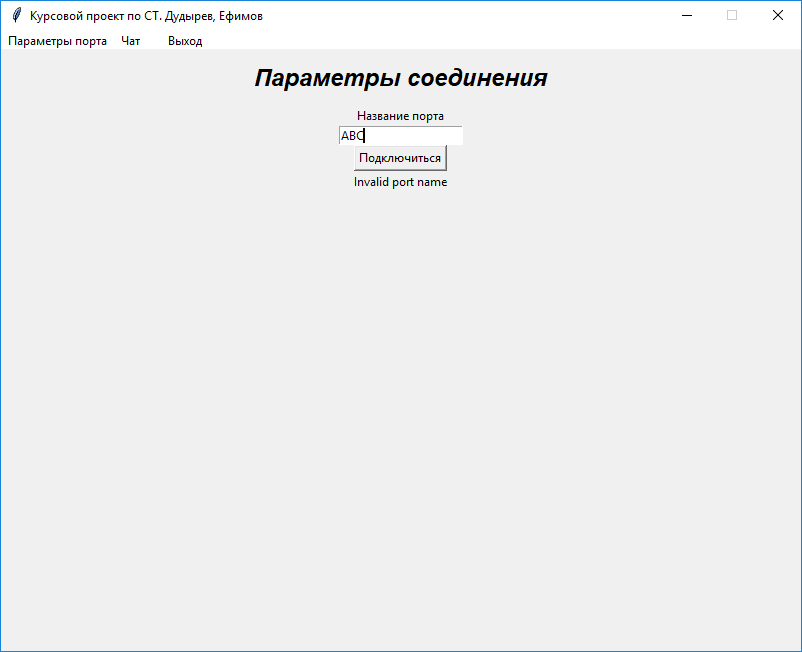
**Пользовательский интерфейс**

Пользовательский интерфейс выполнен в среде PyCharm. При его разработке учитывались рекомендации по простоте, удобству и функциональности интерфейса.

При запуске программы появляется форма, в которой необходимо ввести параметры соединения.



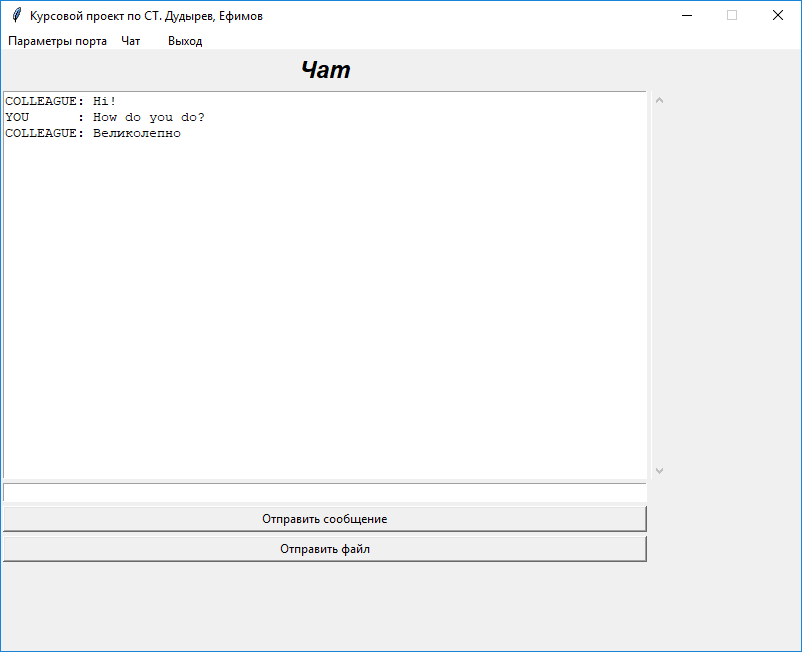
Далее появится окно с выбором соединения, в поле которого необходимо ввести COM порт, к которому хотите подключиться.



После этого будет произведено открытие портом для дальнейшей работы и осуществится переход на главное окно.

Главным окном программы является окно «Чат». В данной форме есть следующие возможности:

1. Соединение с компьютером в сети.
2. Выбор файла
3. Отправка файла
4. Разъединение соединения
5. Управление темпом передачи
6. Окно вывода файла
7. Использование меню



Выбор и отправка файла.

