

Утверждаю:

Галкин В.А. " __ " _____ 2020 г.

**Курсовая работа по дисциплине
«Сетевые технологии в АСОИУ»
«Локальная безадаптерная сеть»**

Пояснительная записка

(вид документа)

писчая бумага

(вид носителя)

14

(количество листов)

ИСПОЛНИТЕЛИ:

студенты группы ИУ5-61Б

Болгова А.В. _____

Фонканц Р.В. _____

Попов И.А. _____

Содержание

1. Введение.....	3
2. Требования к программе.....	3
3. Определение структуры программного продукта	3
4. Физический уровень.	3
4.1 Функции физического уровня.	3
4.2 Описание физического уровня.	3
4.3 Нуль-модемный интерфейс.....	5
4.4 Настройка COM-порта средствами Python.	7
4.5 Описание функций физического уровня. Основные функции PySerial.....	8
4.5.1. Задание параметров COM-порта.....	8
4.5.2. Установление и поддержание/разъединение физического канала.....	8
4.5.3. Прием информации и ее накопление в буфере/Передача информации из буфера в интерфейс.....	8
5. Канальный уровень.	8
5.1 Функции канального уровня.	8
5.2 Протокол связи.	9
5.3 Защита передаваемой информации.	9
5.4. Процедуры взаимодействия.	10
5.4.2. Невозможность установления физического соединения.....	10
5.4.3. Успешная передача сообщения.....	10
5.4.4. Передача сообщения с ошибкой.	10
5.4.5. Успешное разъединение физического соединения.....	10
5.5. Формат кадров.....	10
5.5.1 Информационные кадры.....	11
5.5.2 Служебные кадры.....	11
6. Прикладной уровень.	12
6.1 Функции прикладного уровня.....	12
6.2 Пользовательский интерфейс программы.	12

1. Введение

Данная программа, выполненная в рамках курсовой работы по предмету «Сетевые технологии», предназначена для организации обмена короткими текстовыми сообщениями и файлами между двумя соединёнными с помощью интерфейса RS232C компьютерами.

2. Требования к программе

К программе предъявляются следующие требования. Программа должна:

2.1. Устанавливать соединение между компьютерами и контролировать его целостность

2.2. Обеспечивать правильность передачи и приема данных с помощью кодирования пакета циклическим [7,4]-кодом

2.3. Обеспечивать функцию передачи коротких сообщений

2.4. Обеспечивать функцию передачи файлов

Программа выполняется под управлением OS Windows 7 и выше. Было решено реализовать программу с помощью среды разработки Python.

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 занимается установлением и поддержанием соединения, формированием и проверкой пакетов обмена протоколов верхних модулей.
- Прикладной уровень занимается выполнением задач программы.

4. Физический уровень.

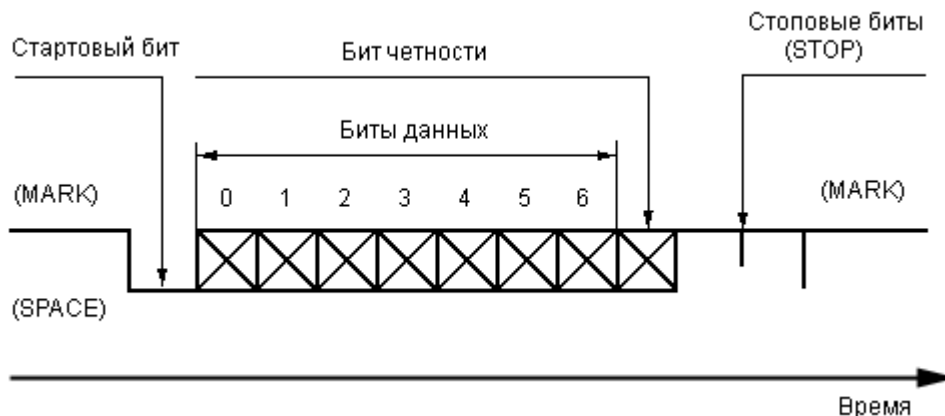
4.1 Функции физического уровня.

Основными функциями физического уровня являются:

1. Задание параметров СОМ-порта.
2. Установление физического канала.
3. Поддержание соединения.
4. Разъединение физического канала.
5. Передача информации из буфера в интерфейс.
6. 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 группы битов данных обычно предшествует специальный *стартовый бит*, после группы битов следуют *бит проверки на четность* и один или два *стоповых бита* (см. рисунок). 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.

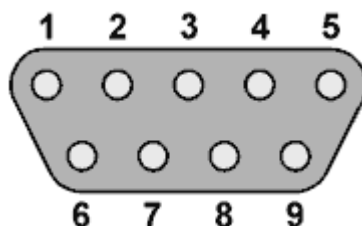
Иногда используется другой термин — биты в секунду (bps). Здесь имеется в виду эффективная скорость передачи данных, без учета служебных битов.

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».

Таблица 1. Разъемы и сигналы интерфейса RS-232C.

Обозначение цепи	Контакт разъема	Направление	Название цепи
RS232	DB9S	Вход/Выход	
PG	-	-	Protect Ground - Защитная земля
TD	3	Выход	Transmit Data - Передаваемые данные
RD	2	Вход	Receive Data - Принимаемые данные
RTS	7	Выход	Request To Send - Запрос на передачу
CTS	8	Вход	Clear To Send - Готовность модема к приему данных для передачи
DSR	6	Вход	Data Set Ready - Готовность модема к работе
SG	5	-	Signal Ground - Схемная земля
DCD	1	Вход	Data Carrier Detect - Несущая обнаружена
DTR	4	Выход	Data Terminal Ready - Готовность терминала (PC) к работе
RI	9	Вход	Ring Indicator - Индикатор вызова



Назначение контактов разъема DB9.

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.

4.3 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.

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

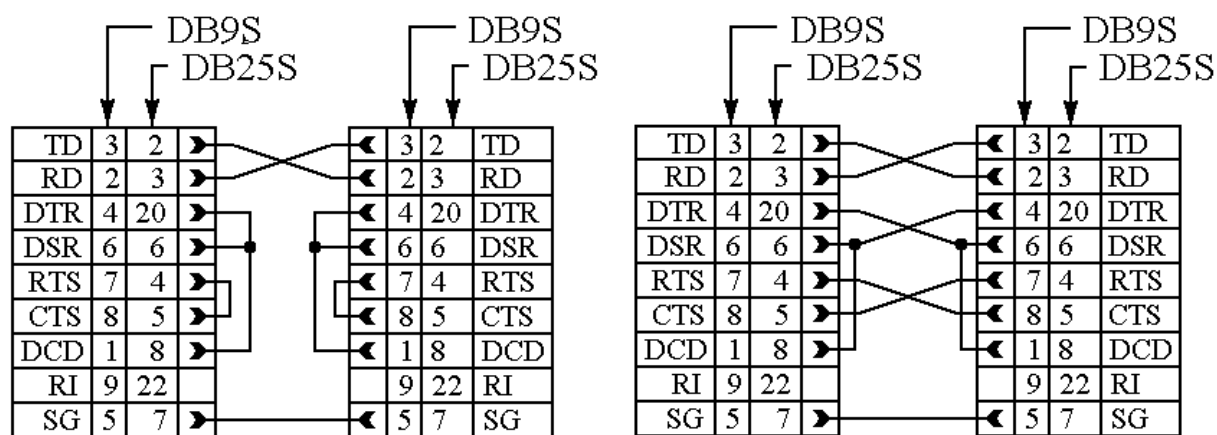
1. компьютер после включения питания выставляет сигнал **DTR**, который постоянно удерживается активным. Если модем включен в электросеть и справен, он отвечает компьютеру сигналом **DSR**. Этот сигнал служит подтверждением того, что **DTR** принят, и информирует компьютер о готовности модема к приему 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**.

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.

Возможный вариант взаимодействия рабочей станции и модема:

1. Установкой **DTR** компьютер указывает на желание использовать модем.
2. Установкой **DSR** модем сигнализирует о своей готовности и установлении соединения.
3. Сигналом **RTS** компьютер запрашивает разрешение на передачу и заявляет о своей готовности принимать данные от модема.
4. Сигналом **CTS** модем уведомляет о своей готовности к приему данных от компьютера и передаче их в линию.
5. Снятием **CTS** модем сигнализирует о невозможности дальнейшего приема (например, буфер заполнен) — компьютер должен приостановить передачу данных.
6. Сигналом **CTS** модем разрешает компьютеру продолжить передачу (в буфере появилось место).
7. Снятие **RTS** может означать как заполнение буфера компьютера (модем должен приостановить передачу данных в компьютер), так и отсутствие данных для передачи в модем. Обычно в этом случае модем прекращает пересылку данных в компьютер.
8. Модем подтверждает снятие **RTS** сбросом **CTS**.
9. Компьютер повторно устанавливает **RTS** для возобновления передачи.
10. Модем подтверждает готовность к этим действиям.
11. Компьютер указывает на завершение обмена.
12. Модем отвечает подтверждением.
13. Компьютер снимает **DTR**, что обычно является сигналом на разрыв соединения (“повесить трубку”).
14. Модем сбросом **DSR** сигнализирует о разрыве соединения.

Нуль-модемный интерфейс характерен для прямой связи компьютеров на небольшом расстоянии (длина кабеля до 15 метров). В случае, когда аппаратура DTE (в нашем случае – рабочие станции) соединяется без модемов, то разъемы устройств (вилки) соединяются между собой нуль-модемным кабелем, имеющим на обоих концах розетки, контакты которых соединяются перекрестно схеме, приведенной на рисунке:



а б
Нуль-модемный кабель: а - минимальный, б – полный

4.4 Настройка COM-порта средствами Python.

Язык программирования Python содержит модуль PySerial. Этот модуль инкапсулирует доступ для последовательного порта. Он предоставляет бэкэнды для Python, работающего в Windows и Linux. PySerial использует прикладной программный интерфейс Win32 API, который предлагает широкие возможности по настройке COM-порта.

Основные параметры и настройки COM-порта при помощи PySerial:

1. **port** – имя устройства от `None`.
2. **baudrate** (*int*) – частота бод(пропускная способность): 9600, 115200...
3. **bytesize** – количество бит данных. Возможные значения:
`FIVEBITS`, `SIXBITS`, `SEVENBITS`, `EIGHTBITS`
4. **parity** – бит четности. Возможные значения:
`PARITY_NONE`, `PARITY_EVEN`, `PARITY_ODD`, `PARITY_MARK`, `PARITY_SPACE`
5. **stopbits** – номер стоп-бит. Возможные значения:
`STOPBITS_ONE`, `STOPBITS_ONE_POINT_FIVE`, `STOPBITS_TWO`
6. **timeout** (*float*) – Set a read timeout value. (Установите значение времени ожидания чтения)
7. **xonxoff** (*bool*) – Enable software flow control. (Включить программное управление потоком)
8. **rtscts** (*bool*) – Enable hardware (RTS/CTS) flow control. (Включить аппаратное (RTS/CTS) управление потоком)
9. **dsrdtr** (*bool*) – Enable hardware (DSR/DTR) flow control. (Включить аппаратное (DSR / DTR) управление потоком)
10. **write_timeout** (*float*) – Set a write timeout value. (Установить значение времени ожидания записи)
11. **inter_byte_timeout** (*float*) – Inter-character timeout, `None` to disable (default). (Межсимвольный тайм-аут, None для отключения (по умолчанию).)

4.5 Описание функций физического уровня. Основные функции PySerial

4.5.1. Задание параметров COM-порта

Сер – объект класса Serial, который является основным классом библиотеки PySerial и используется для определения COM-порта.

В Python для задания параметров используется вызов класса Serial(), в круглые скобки мы записываем значения порта. Используются следующие функции:

port(**self**, port) – *название порта (COM6, COM7)*
baudrate(**self**, baudrate) – *скорость передачи в бодах*
bytesize(**self**, bytesize) – *размер байта*
parity(**self**, parity) – *бит четности*
stopbits(**self**, stopbits) – *стопбит*

4.5.2. Установление и поддержание/разъединение физического канала

После инициализации порта и задания его параметров, для открытия порта и поддержания соединения необходимо использовать следующую функцию:

open(**self**) – *функция открытия COM-порта*

Соответственно, для разъединения необходимо закрыть порт при помощи функции:

close(**self**) – *функция закрытия COM-порта*

4.5.3. Прием информации и ее накопление в буфере/Передача информации из буфера в интерфейс

Существующие порты имеют в своем распоряжении входные и выходные буферы. Буферы порта можно очистить с помощью функций flushInput(**self**) и flushOutput(**self**).

Для записи данных в порт и чтения данных из порта используются следующие функции:

write(**self**, data) – *функция для передачи данных в порт*

read(**self**, size) – *функция приема данных из порта*

in_waiting(**self**) – *функция, возвращающая количество байт во входном буфере*

При помощи функции list_ports.comports(**self**) можно получить данные о доступных COM-портах системы.

5. Канальный уровень.

5.1 Функции канального уровня.

На канальном уровне выполняются следующие функции:

1. Запрос физического соединения
2. Управление передачей кадров

3. Обеспечение необходимой последовательности блоков данных, передаваемых через межуровневый интерфейс
4. Контроль и исправление ошибок
5. Запрос на разъединение физического 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 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.

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.

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 кадровой (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.

Циклические коды относятся к линейным кодам. Специфические свойства данного вида кодов помогают как при кодировании/декодировании, так и при аппаратной реализации этих процессов. Процедура построения таких кодов гораздо более управляемая. Однако, нам потребуется перейти от векторного описания кодов к полиномиальному. Последовательность символов основного алфавита (0'ки и 1'ки в простейшем случае), составляющие сообщения и кодовые слова мы будем интерпретировать как коэффициенты полиномов. Например, считая, что коэффициенты записаны в порядке возрастания степени, сообщение [1010] запишем в виде многочлена $1 + x^2$. Кодирование сообщения в более “длинное” кодовое слово будет проводиться умножением этого многочлена на другой, что дает в результате многочлен более высокой степени.

Рассмотрим алгоритм циклического кода:

Информационное слово – 1001 соответствует полиному $x^3 + 1$.

Сдвиг слова влево на 3 – 1001000 или умножение полинома на x^3 : $x^6 + x^3$.

Деление информационного полинома на порождающий полином:

$$\begin{array}{r} x^6 + x^3 \\ \underline{x^6 + x^4 + x^3} \\ -x^4 \\ \underline{x^4 + x^2 + x} \\ x^2 + x \end{array} \quad \begin{array}{r} | x^3 + x + 1 \end{array}$$

приписываем остаток к сдвинутому полиному и получаем закодированный полином: $x^6 + x^3 + x^2 + x$, что соответствует двоичному слову 1001110 или десятичному 78. Зная слово, которое было закодировано, программа может декодировать полученное из канала связи слово с ошибками.

Реализация защиты передаваемой информации с помощью циклического кода [7,4]:

Алгоритм кодирования данных с помощью циклического кода [7,4] использует кодирующий вектор [1011] или десятичное число 11. Для кодирования, из кадра берётся поочерёдно по 4 байта (32 бита) с конца, которые в свою очередь кодируются по 4 бита, после кодирования получается блок в 7 байт. Из таких блоков собирается кадр с уже закодированными данными.

Декодирование:

При декодировании кадр так же делится по 7 байт с конца, где каждые 7 бит декодируются отдельно, превращаясь в 4 байта с уже исправленными ошибками (если такие возникли при передаче), и затем, из таких пачек собирается кадр с уже расшифрованными данными.

5.4. Процедуры взаимодействия.

5.4.1. Успешное установление физического соединения.

На прикладном уровне необходимо нажать на кнопку «Подключиться». После этого на физический уровень подается команда занять СОМ-порт, заранее прописанный в конфигурационном файле. При этом присутствует проверка на занятость СОМ-порта другим процессом, если данный порт занят, то будет занят резервный порт (удобно при симуляции работы мессенджера, запуская 2 клиента на одном и том же компьютере. На прикладном уровне отображается уведомление «Соединение установлено».

5.4.2. Невозможность установления физического соединения.

В случае же если программе не удаётся занять ни один из прописанных портов, на прикладном уровне отображается сообщение о том, что соединение установить невозможно.

5.4.3. Успешная передача сообщения.

На прикладном уровне необходимо ввести сообщение в главное окно и нажать «Отправить». После этого на канальный уровень подается команда послать кадр(ы) с сообщением или файлом. На канальном уровне происходит разбиение на кадры и шифрование сообщения. На прикладном уровне, в окне исходящих сообщений отображается сообщение. На физическом уровне передаются биты. Второй компьютер на канальном уровне получает кадр(ы), ориентируясь на количество принятый байт и формирует сообщение из полученных кадров. На канальном уровне получателя происходит расшифровка данных и исправление ошибок, составление текста или создание файла. На прикладном уровне отображается в главном окне сообщение от первого компьютера.

5.4.4. Передача сообщения с ошибкой.

Ошибки при передаче исправляются канальным уровнем на этапе дешифрации кадров.

5.4.5. Успешное разъединение физического соединения.

На прикладном уровне необходимо нажать на кнопку «Выход». После этого на физический уровень подается команда закрыть занятый СОМ-порт.

5.5. Формат кадров.

Кадры, передаваемые с помощью функций канального уровня, имеют различное назначение. Кадры имеют фиксированную длину в 133 байта.

PRIM-кадр – заголовочный кадр стоит в начале сообщения, содержит информацию о сообщении и часть передаваемой информации.

DAT-кадр – кадр содержит передаваемые данные.

5.5.1 Информационные кадры

- **PRIM-кадр**

Заголовочный кадр, стоит в самом начале, содержит идентификатор сообщения, информацию о количестве кадров в сообщении, типе сообщения (текстовое сообщение или файл)

2 бита	6 бит	24 бита	1 бит	7 бит	128 б
Флаг типа кадра	Идентификатор сообщения	Количество кадров в сообщении	Тип сообщения	Количество значащих символов в кадре	данные

Флаг заголовочного кадра **11**

- **DATA-кадр**

Кадр данных передаёт информацию клиента

2 бита	6 бит	24 бита	1 бит	7 бит	128 б
Флаг типа кадра	Идентификатор сообщения	Номер текущего кадра	Тип сообщения	Количество значащих символов в кадре	данные

Флаг кадра данных **00**

Тип сообщения **0** – текстовое сообщение; **1** – файл

Идентификатор сообщения может распределить кадры от разных сообщений, если они пришли в одно время. Номер кадра служит для сортировки кадров по порядку, если они пришли не в правильном порядке.

В случае передачи файла в заголовочном кадре содержится только название передаваемого файла, кадры данных передают тело файла. В случае текстового сообщения, часть его содержится в заголовочном кадре. Кадры имеют фиксированную длину. Если данные в кадре не заняли полный объём в 128 байт, то оставшееся место будет занято символом – заглушкой, так же как и в случае заголовочного кадра при передаче файла.

5.5.2 Служебные кадры

Флаг служебного кадра **01** – одинаковый для всех служебных кадров

2 бита	6 бит	132 байта
Флаг типа кадра	Тип служебного кадра	Заполнитель

Следующие после флага кадра 6 бит характеризуют тип служебного кадра:

- **RET** – тип кадра **000000**; - кадр запроса повторной передачи сообщения при ошибке в сообщении
- **ACK** – тип кадра **000001**; - квитанция на успешно доставленное сообщение
- **UPLINK** – тип кадра **000010**; – кадр запроса на установление логического соединения
- **ACK UPLINK** – тип кадра **000011**; - положительная квитанция на UPLINK-кадр
- **DOWNLINK** – тип кадра **000100**; – кадр разрыва логического соединения
- **ACK DOWNLINK** – тип кадра **000101**; – положительная квитанция на DOWNLINK-кадр

- **LINKACTIVE** – тип кадра **001000**; – кадр поддержания логического соединения
- **ACK LINKACTIVE** – тип кадра **001001**; – положительная квитанция на LINKACTIVE-кадр

6. Прикладной уровень.

6.1 Функции прикладного уровня.

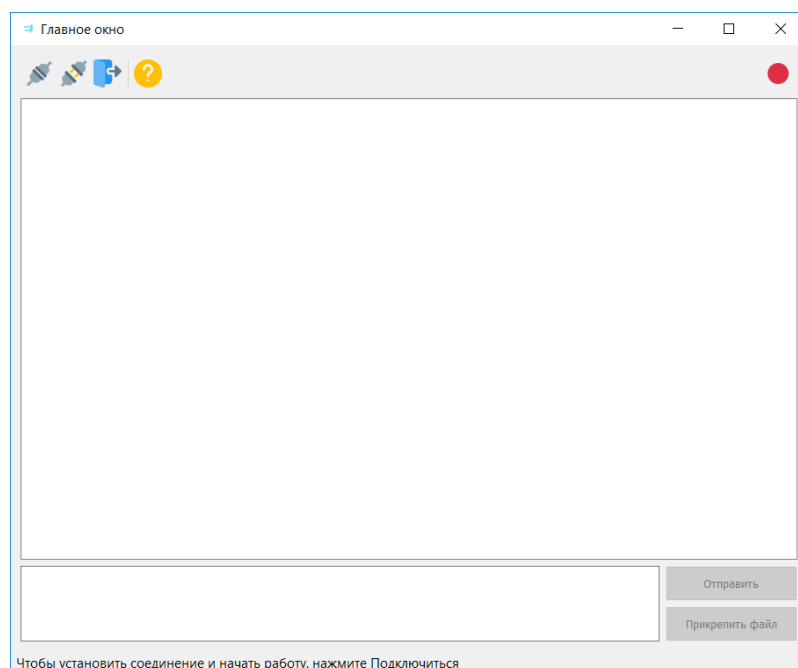
На данном уровне должны выполняться следующие функции:

- обеспечение интерфейса программы с пользователем через систему форм и меню;
- набор и редактирование текстовых сообщений;
- отображение истории сообщений, вывод принятых и отправленных сообщений в окно диалога пользователей;
- отправка сообщения, предоставление сообщения нижнему уровню;
- отправка файла;
- передаваемый файл должен указываться на передающей ПЭВМ.






6.2 Пользовательский интерфейс программы.

Пользовательский интерфейс выполнен с использованием графического фреймворка Qt для языка программирования Python – PyQt версии 5.13.0, а также входящего в состав PyQt среды для разработки графических интерфейсов (GUI) – Qt Designer. При разработке интерфейса основными целевыми характеристиками приложения были простота, понятность, наглядность и удобство использования.

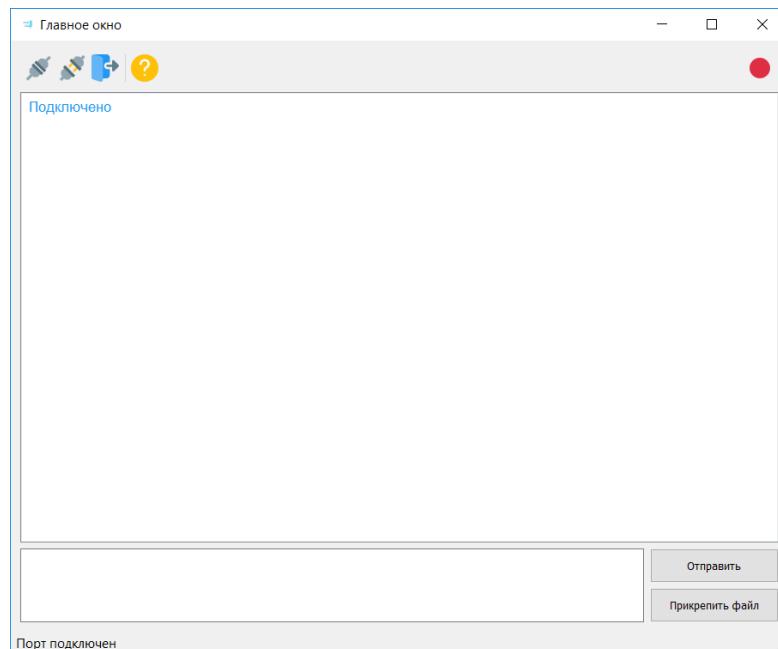
При запуске программы появляется основная форма «Главное окно», размер которой можно свободно изменять.



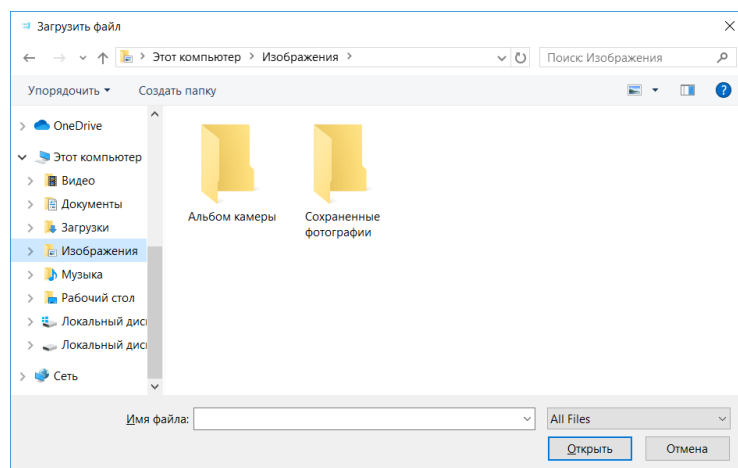
В данной форме реализованы следующие функции:

1. Инициализация и открытие одного из заданных по умолчанию портов по нажатию 
2. Разъединение порта по нажатию 
3. Закрытие рабочего порта и завершение работы приложения по нажатию 
4. Вызов информационной справки по нажатию 
5. Мониторинг соединения с портом и пользователем при помощи Индикатора  или текстовых сообщений

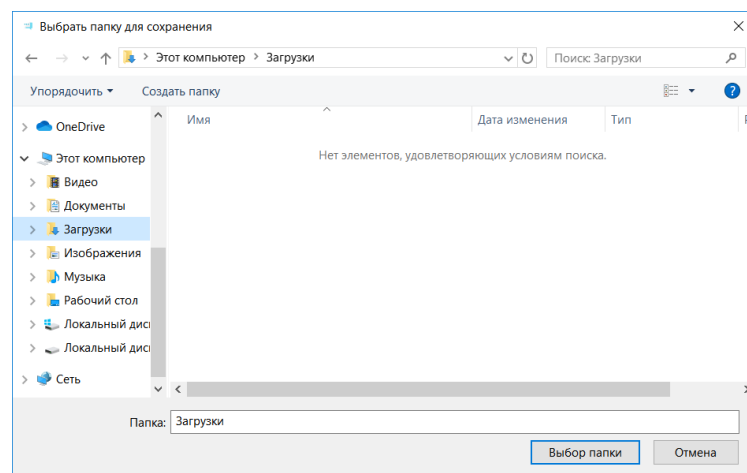
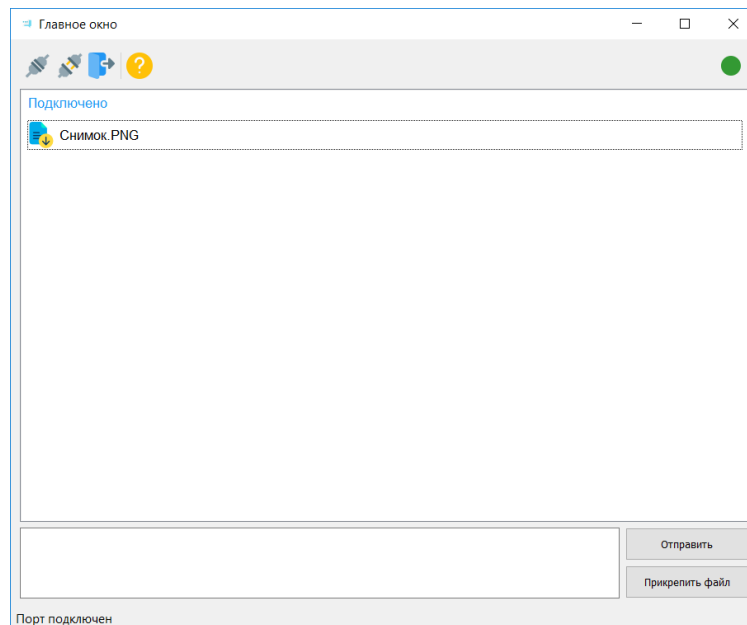
6. Набор и редактирование сообщений
7. Отображение текущей истории сообщений
8. Отправка новых сообщений по нажатию кнопки «Отправить»
9. Отправка файла по нажатию кнопки «Прикрепить файл»
10. Скачивание файла в указанную папку по двойному нажатию на файл
11. Вывод служебных сообщений пользователю при ошибках или успешном выполнении




При нажатии на кнопку «Прикрепить файл» вызывается стандартное диалоговое окно загрузки файла системы «Загрузить файл»:



При получении файла необходимо кликнуть два раза по сообщению с файлом, чтобы открыть стандартное диалоговое окно выбора директории «Выбрать папку для сохранения» и указать директорию, куда будет сохранен файл. По умолчанию файл сохраняется в папку downloads в корневом каталоге программы.



Нажатие кнопки Информация  открывает диалоговое окно «Информация», содержащее описание выполненной работы и ФИО исполнителей. Кнопка «ОК» закрывает диалоговое окно.

