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)

ФАКУЛЬТЕТ _____

КАФЕДРА _____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

_____

_____

_____

_____

_____

_____

Студент _____ _____ _____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы _____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(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 _____
(Индекс)
_____ _____
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине _____

Студент группы _____

_____
(Фамилия, имя, отчество)

Тема курсовой работы _____
_____
_____

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
_____
Источник тематики (кафедра, предприятие, НИР) _____

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

Задание _____
_____
_____
_____
_____

Оформление курсовой работы:

Расчетно-пояснительная записка на _____ листах формата А4.

_____
_____
_____
_____

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсовой работы _____ _____
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____
(Подпись, дата) (И.О.Фамилия)

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

Оглавление

1. Введение

Данная программа, выполненная в рамках курсовой работы по предмету «Сетевые технологии», предназначена для пересылки файлов между соединёнными с помощью интерфейса RS232C компьютерами.

2. Требования к программе

Программное изделие выполняется на Java под управлением Linux и MS Windows.

Для работы программы требуются 2 ПЭВМ типа IBM PC AT (/XT), соединенные нульмодемным кабелем через интерфейс RS-232C.

3. Определение структуры программного продукта

См. Схему «Структурная схема программы»

4. Физический уровень

4.1. Сигналы интерфейса RS-232-C.

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.

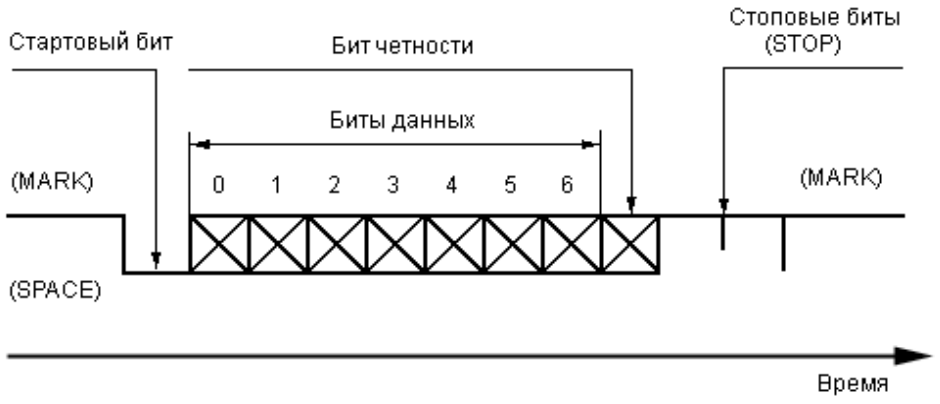


Рисунок 1.

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 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. 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 (Reguest 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.

Рассмотрим 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. Иногда для соединения двух устройств DTE эти линии (RTS и CTS) соединяются вместе на каждом конце кабеля. В результате получаем то, что другое устройство всегда готово для получения данных (если при большой скорости передачи принимающее устройство не успевает принимать и обрабатывать данные, возможна потеря данных).

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

4.2. Нуль-модемный интерфейс.

Обмен сигналами между адаптером компьютера (DTE) и модемом (DCE) (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 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.

4.3. Настройка COM-порта средствами библиотеки Jscc.

4.3.1. Описание класса SerialPort

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

Поля класса:

Частоты порта:

BAUDRATE_110

BAUDRATE_115200

BAUDRATE_1200

BAUDRATE_128000

BAUDRATE_14400

BAUDRATE_19200

BAUDRATE_256000

BAUDRATE_300

BAUDRATE_38400

BAUDRATE_4800

BAUDRATE_57600

BAUDRATE_600

BAUDRATE_9600

DATABITS_5 – 5 бит

DATABITS_6 – 6 бит

DATABITS_7 – 7 бит

DATABITS_8 – 8 бит

FLOWCONTROL_NONE – отключить управление обменом данными.

FLOWCONTROL_RTSCTS_IN – RTS/CTS управление обменом данными.

FLOWCONTROL_RTSCTS_OUT – RTS/CTS управление обменом данными.

FLOWCONTROL_XONXOFF_IN – XON/XOFF управление обменом данными.

FLOWCONTROL_ XONXOFF_OUT – XON/XOFF управление обменом данными.

MASK_BREAK

MASK_CTS

MASK_DSR

MASK_ERR

PARITY_EVEN - Дополнение до четности.

PARITY_MARK - Бит четности всегда 1.

PARITY_NONE - Бит четности отсутствует.

PARITY_ODD – Дополнение до нечетности.

PARITY_SPACE – Бит четности всегда 0.

STOPBITS_1 - один стоп-бит.

STOPBITS_1_5 - полтора стоп-бита.

STOPBITS_2 – два стоп-бита.

Методы класса:

addEventListener(SerialPortEventListener listener)

добавляет обработчика событий.

addEventListener(SerialPortEventListener listener, int mask)

Добавить обработчика событий.

closePort()

Закрыть порт

getEventsMask()

Получить маски событий для порта

getFlowControlMode()

Получить режим управления потоком

getInputBufferBytesCount()

Получить количество байтов во входном буфере

getLinesStatus()

Получение статуса линий.

getOutputBufferBytesCount()

Получить количество байтов в выходном буфере

getPortName()

Получение имени порта в процессе работы

isCTS()

Получить состояние линии CTS

isDSR()

Получить состояние линии DSR

isOpened()

Получение состояния порта

isRING()

Получить состояние линии RING

isRLSD()

Получить состояние линии RLSD

openPort()

Открытие порта

purgePort(int flags)

Очистка буфера ввода и вывода.

readBytes()

Прочитать все доступные байты из порта как массив байтов

readBytes(int byteCount)

Чтение байтового массива из порта

readBytes(int byteCount, int timeout)

Чтение байтового массива из порта

readHexString()

Прочитать все доступные байты из порта как шестнадцатеричную строку

readHexString(int byteCount)

Считать шестнадцатеричную строку из порта (пример: FF 0A FF).

readHexString(int byteCount, int timeout)

Считать шестнадцатеричную строку из порта (пример: FF 0A FF).

readHexString(int byteCount, java.lang.String separator)

Прочитать шестнадцатеричную строку из порта с установленным разделителем (например, если разделитель "::": FF :: 0A :: FF)

readHexString(int byteCount, java.lang.String separator, int timeout)

Прочитать шестнадцатеричную строку из порта с установленным разделителем (например, если разделитель "::": FF :: 0A :: FF)

readHexString(java.lang.String separator)

Прочитать все доступные байты из порта как шестнадцатеричную строку с установленным разделителем

readHexStringArray()

Прочитать все доступные байты из порта как массив Hex StringreadHexStringArray(int byteCount)

Чтение массива шестнадцатеричных строк из порта

readHexStringArray(int byteCount, int timeout)

Чтение массива шестнадцатеричных строк из порта

readIntArray()

Прочитать все доступные байты из порта как массив int (значения в диапазоне от 0 до 255)

readIntArray(int byteCount)

Чтение массива int из порта

readIntArray(int byteCount, int timeout)

Чтение массива int из порта

readString()

Прочитать все доступные байты из порта как строку

readString(int byteCount)

Прочитать строку из порта

readString(int byteCount, int timeout)

Прочитать строку из порта

removeEventListener()

Удалить прослушиватель событий.

sendBreak(int duration)

Отправить сигнал перерыва на установленный срок

setDTR(boolean enabled)

Изменить состояние линии DTR.

setEventsMask(int mask)

Установить маску событий.

setFlowControlMode(int mask)

Установите режим управления потоком.

setParams(int baudRate, int dataBits, int stopBits, int parity)

Установка параметров порта.

setParams(int baudRate, int dataBits, int stopBits, int parity, boolean setRTS, boolean setDTR)

Настройка параметров порта

setRTS(boolean enabled)

Изменить состояние линии RTS.

writeByte(byte singleByte)

Запись одного байта в порт

writeBytes(byte[] buffer)

Записать байтовый массив в порт

writeInt(int singleInt)

Записать значение int (в диапазоне от 0 до 255 (0x00 - 0xFF)) в порт

writeIntArray(int[] buffer)

Записать массив int (в диапазоне от 0 до 255 (0x00 - 0xFF)) в порт

writeString(java.lang.String string)

Записать строку в порт

writeString(java.lang.String string, java.lang.String charsetName)

Записать строку в порт

4.3.2. Описание класса SerialPortEvent

Поля класса:

BREAK - 0
CTS - состояние линии CTS (0 - ВЫКЛ, 1 - ВКЛ)
DSR - состояние линии DSR (0 - ВЫКЛ, 1 - ВКЛ)
ERR - маска ошибок
RING - состояние линии ЗВОНОК (0 - ВЫКЛ, 1 - ВКЛ)
RLSD - состояние линии RLSD (0 - ВЫКЛ, 1 - ВКЛ)
RXCHAR - флаг байтов во входном буфере
RXFLAG - флаг байтов во входном буфере (не поддерживается в Linux)
TXEMPTY - флаг байтов в буфере вывода

Методы класса:

getEventType()

Получение типа события

getEventValue()

Получение ценности события

getPortName()

Получение имени порта, отправившего событие

isBREAK()

Метод возвращает true, если получено событие типа «BREAK», иначе false.

isCTS()

Метод возвращает true, если получено событие типа «CTS», иначе false.

isDSR()

Метод возвращает true, если получено событие типа "DSR", иначе false.

isERR()

Метод возвращает true, если получено событие типа "ERR", иначе false.

isRING()

Метод возвращает true, если получено событие типа «RING», иначе false.

isRLSD()

Метод возвращает true, если получено событие типа "RLSD", иначе false.

isRXCHAR()

Метод возвращает true, если получено событие типа "RXCHAR", иначе false.

isRXFLAG()

Метод возвращает true, если получено событие типа "RXFLAG", иначе false.

isTXEMPTY()

Метод возвращает true, если получено событие типа "TXEMPTY", иначе false.

4.3.3. Описание класса SerialPortListener

Методы класса:

serialEvent(SerialPortEvent serialPortEvent)

4.3.4. Описание класса SerialPortList

Методы класса:

getPortNames()

Получить отсортированный массив последовательных портов в системе с настройками по умолчанию:

5. Канальный уровень

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.

5.1. 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 и ожидает ее приема с другого COM-порта.

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.

5.2. 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 ERROR для сообщения о возникшей ошибке.

5.3. Формат кадров.

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

5.3.1. Информационный BINARY DATA кадр

Информационные кадры применяются для передачи закодированных кодом Хэмминга данных.

Тип кадра: 00000100.

Длина блока данных кадра: XXXXXXXX XXXXXXXX

Блок данных: закодированные кодом Хэмминга данные.

5.3.2. Супервизорный LINK кадр

Используется для поднятия логического соединения.

Тип кадра: 00000000.

Длина блока данных кадра: 00000000 00000000.

5.3.3. Супервизорный ACK кадр

Используется для подтверждения получения данных.

Тип кадра: 00000001.

Длина блока данных кадра: 00000000 00000000.

5.3.4. Супервизорный SYNC кадр

Используется для синхронизации скорости соединения. В блоке данных содержится информация про скорость COM порта.

Тип кадра: 00000010.

Длина блока данных кадра: 00000000 00000100.

Данные: XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX

5.3.5. Супервизорный DOWN LINK кадр

Используется для разрыва логического соединения.
Тип кадра: 00000101.

Длина блока данных кадра: 00000000 00000000.

5.3.6. Супервизорный ERROR кадр

Используется для сообщения об ошибке в кадре данных.

Тип кадра: 00000110.

Длина блока данных кадра: 00000000 00000000.

5.3.7. Информационный FILE HEADER кадр

Используется для имени файла.

Тип кадра: 00000011.

Длина блока данных кадра: XXXXXXXX XXXXXXXX

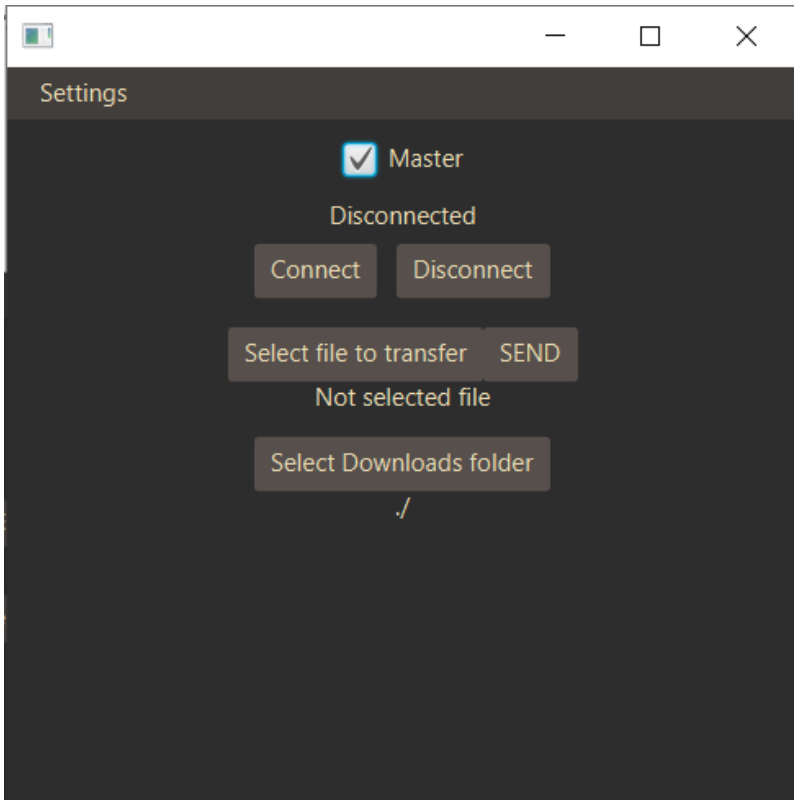
Данные: 8 байт на длину файла + m байт на название файла.

6. Прикладной уровень.

Функции прикладного уровня:

1. интерфейс с пользователем через систему меню,
2. выбор режима работы,
3. выбор номера COM-порта для канала,
4. установка параметров COM-порта,
5. имя передаваемого файла указывается на передающем ПК, а имя подкаталога для размещения полученного файла указывается на ПК-получателе.

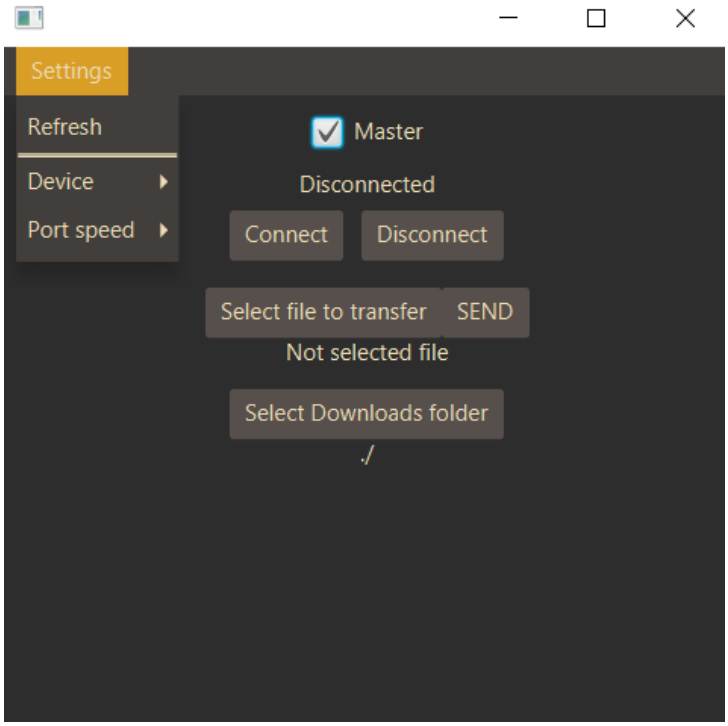
6.1. Окно главного меню



По середине есть опция выбрать файл, выбрать папку, куда отправится файл, также указать, кто Master для инициации логического соединения, подключиться и отключиться.

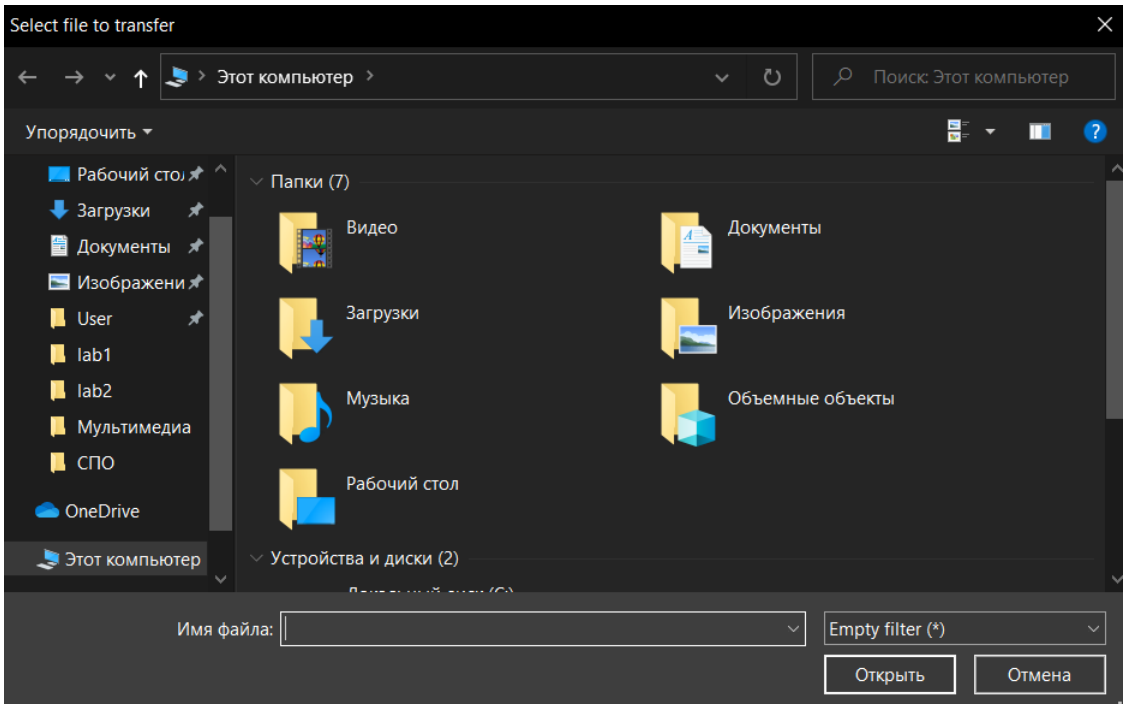
6.2. Окна настройки подключения

Здесь мы можем выбрать порт, скорость порта, а также обновить список доступных устройств.

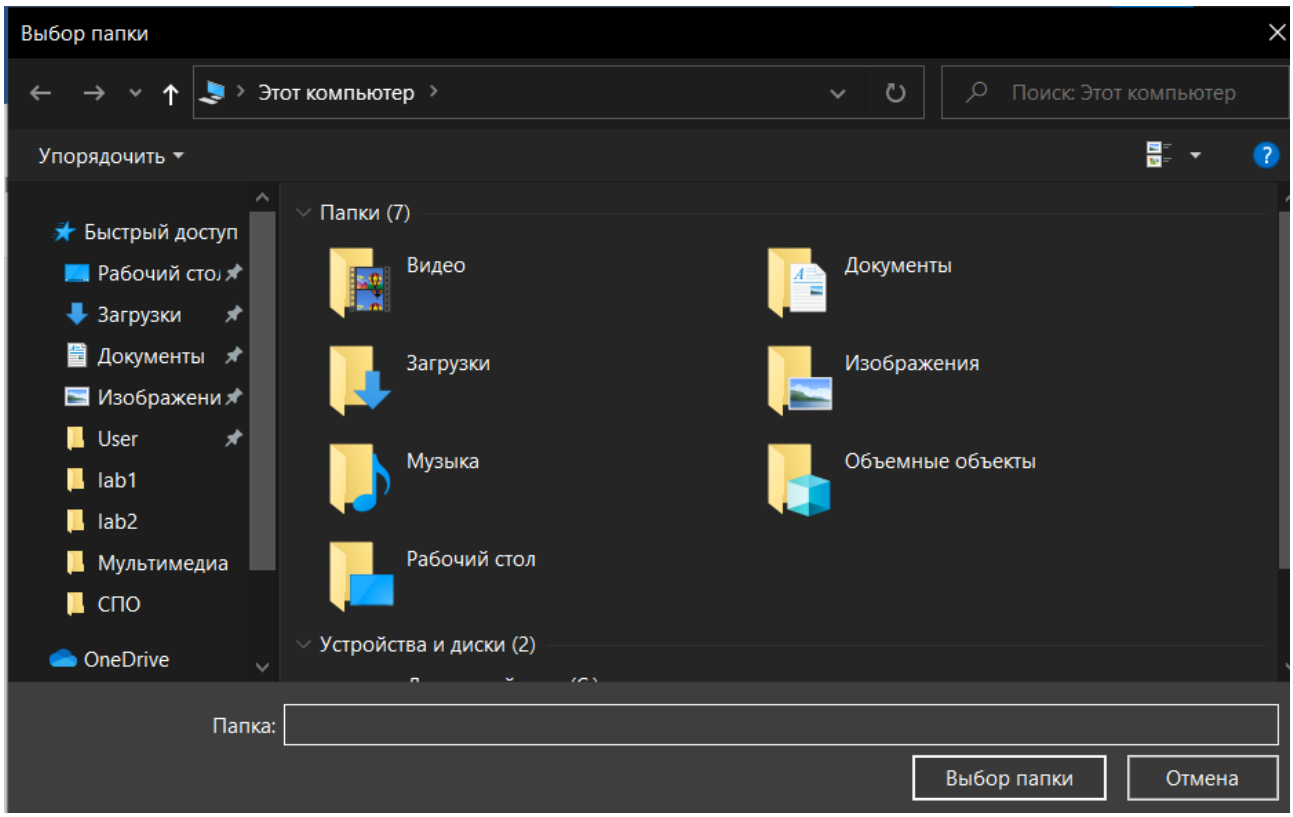


6.3. Окно выбора файла и окно выбора папки.

По нажатии кнопки «Select file to transfer» откроется меню выбора файла.

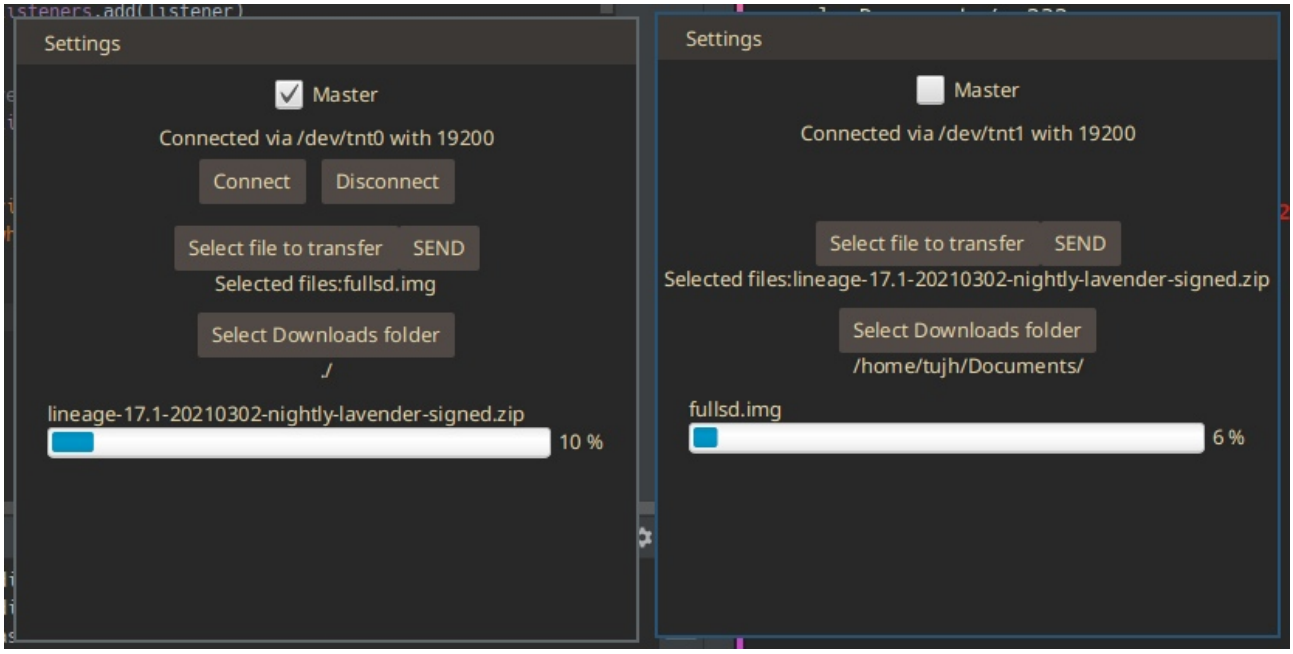


По нажатии кнопки «Select Downloads folder» откроется меню выбора папки для загрузки и сохранения файла.



6.4. Окно отправки файла

При отправке файлов появится статус прогресса.



6.5. Доказательство работоспособности кодирования

Обнаруживающая способность используемого кода.

```
1  :  16  :  16.0   :   1.0
2  :  120  :  120.0   :   1.0
3  :  517  :  560.0   :   0.9232142857142858
4  :  1681  :  1820.0   :   0.9236263736263737
5  :  4119  :  4368.0   :   0.9429945054945055
6  :  7548  :  8008.0   :   0.9425574425574426
7  :  10679  :  11440.0   :   0.933479020979021
8  :  12020  :  12870.0   :   0.933954933954934
9  :  10776  :  11440.0   :   0.941958041958042
10  :  7535  :  8008.0   :   0.9409340659340659
11  :  4065  :  4368.0   :   0.9306318681318682
12  :  1697  :  1820.0   :   0.9324175824175824
13  :  536  :  560.0   :   0.9571428571428572
14  :  114  :  120.0   :   0.95
15  :  13  :  16.0   :   0.8125
16  :  1  :  1.0   :   1.0
TOTAL PECENTAGE    0.9374685282673381
```