Московский государственный технический университет им. Н.Э. Баумана

Утверждаю:

Галкин В.А

"__"_____2014 г.

Курсовая работа
по курсу «Сетевые технологии»:
«Программа передачи коротких сообщений»

Расчётно-пояснительная записка
(вид документа)

писчая бумага
(вид носителя)

13
(количество листов)

ИСПОЛНИТЕЛИ:

студенты групп
ИУ5-63 и ИУ5-64
Леонтьев А.В. _____
Корнуков Н.О. _____
Латкин И.И. _____

"__"_____2014 г.

Москва - 2014

Оглавление

1. ФИЗИЧЕСКИЙ УРОВЕНЬ

1.1. Интерфейс RS-232C

Интерфейс RS-232C (официальное название "EIA/TIA–232–E") является наиболее широко распространенной стандартной последовательной связью между микрокомпьютерами и периферийными устройствами. Интерфейс, определенный стандартом Ассоциации электронной промышленности (EIA), подразумевает наличие оборудования двух видов: терминального DTE и связного DCE.

Терминальное оборудование, например микрокомпьютер, может посылать или принимать данные по последовательному интерфейсу. Оно как бы оканчивает (terminate) последовательную линию. Связное оборудование - устройства, которые могут упростить передачу данных совместно с терминальным оборудованием. Наглядным примером связного оборудования служит модем (модулятор-демодулятор). Он оказывается соединительным звеном в последовательной цепочке между компьютером и телефонной линией.

Конечной целью подключения является соединение двух устройств DTE. Полная схема соединения включает в себя устройства DCE соединённые с DTE через интерфейс RS-232 и линию удалённой связи. Интерфейс позволяет исключить канал удаленной связи вместе с парой устройств DCE (модемов), соединив устройства непосредственно с помощью нуль-модемного кабеля (рис. 1).

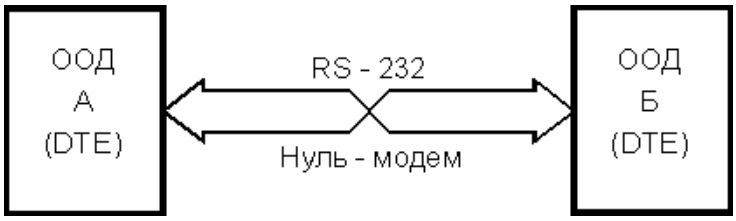


Рис. 1. Соединение по RS-232C нуль-модемным кабелем

1.2. Физические параметры интерфейса RS-232C

Стандарт RS-232C регламентирует типы применяемых разъемов, что обеспечивает высокий уровень совместимости аппаратуры различных производителей. На аппаратуре DTE (в том числе, и на COM-портах PC) принято устанавливать вилки (male - "папа") DB25-P или DB9-P. Девятиштырьковые разъемы не имеют контактов для дополнительных сигналов, необходимых для синхронного режима.

В случае, когда аппаратура DTE соединяется без модемов ("Короткозамкнутая петля"), то разъемы устройств (вилки) соединяются между собой нуль-модемным кабелем (Zero modem или Z-modem), имеющим на обоих концах розетки, контакты которых соединяются перекрестно схеме, приведенной на рис. 2.

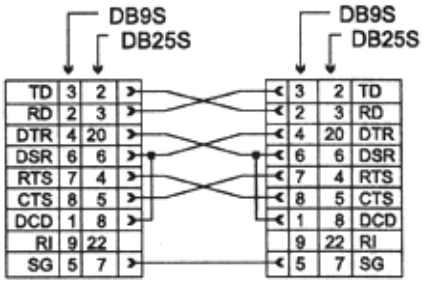


Рис. 2. Полный нуль-модемный кабель.

В таблице 1 приведено назначение контактов разъемов COM-прортов (и любой другой аппаратуры DTE). Назначение контактов разъема DB9S (рис. 3) определено стандартом EIA/TIA-574.

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

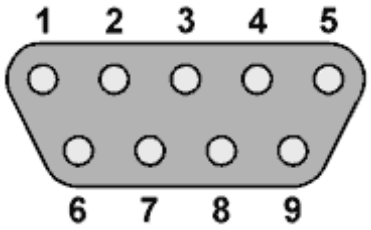


Рис. 3. Назначение контактов разъема DB9.

Функциональное назначение сигнальных линий интерфейса RS-232C:

1. Установкой DTR компьютер указывает на желание установить соединение.
2. Установкой DSR второй компьютер сигнализирует о своей готовности и установлении соединения.
3. Сигналом RTS компьютер запрашивает разрешение на передачу и заявляет о своей готовности принимать данные от второго компьютера.
4. Сигналом CTS второй компьютер уведомляет о своей готовности к приему данных от первого компьютера и передаче их в линию.
5. Снятием CTS второй компьютер сигнализирует о невозможности дальнейшего приема (например, буфер заполнен) — первый компьютер должен приостановить передачу данных.
6. Сигналом CTS второй компьютер разрешает первому 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 другой компьютер. Обычно в этом случае прекращается пересылка данных в компьютер.
8. Второй компьютер подтверждает снятие RTS сбросом CTS.
9. Компьютер повторно устанавливает RTS для возобновления передачи.
10. Второй компьютер подтверждает готовность к этим действиям.
11. Компьютер указывает на завершение обмена.
12. Второй компьютер отвечает подтверждением.
13. Компьютер снимает DTR, что обычно является сигналом на разрыв соединения (“повесить трубку”).
14. Второй компьютер сбросом DSR сигнализирует о разрыве соединения.

1.3. Асинхронная передача данных

Асинхронный режим передачи является байт-ориентированным (символьно-ориентированным): минимальная пересылаемая единица информации — один байт (один символ). Формат посылки байта иллюстрирует рис. 4. Передача каждого байта начинается со старт-бита, сигнализирующего приемнику о начале посылки, за которым следуют биты данных и, возможно, бит четности (Parity). Завершает посылку стоп-бит, гарантирующий паузу между посылками. Старт-бит следующего байта посылается в любой момент после стоп-бита, то есть между передачами возможны паузы произвольной длительности. Старт-бит, имеющий всегда строго определенное значение, обеспечивает простой механизм синхронизации приемника по сигналу от передатчика. Подразумевается, что приемник и передатчик работают на одной скорости обмена. Внутренний генератор синхронизации приемника использует счетчик-делитель опорной частоты, обнуляемый в момент приема начала старт-бита. Этот счетчик генерирует внутренние стробы, по которым приемник фиксирует последующие принимаемые биты. В идеале стробы располагаются в середине битовых интервалов, что позволяет принимать данные и при незначительном рассогласовании скоростей приемника и передатчика. Очевидно, что при передаче 8 бит данных, одного контрольного и одного стоп-бита предельно допустимое рассогласование скоростей, при котором данные будут распознаны верно, не может превышать 5 %. С учетом фазовых искажений и дискретности работы внутреннего счетчика синхронизации реально допустимо меньшее отклонение частот. Чем меньше коэффициент деления опорной частоты внутреннего генератора (чем выше частота передачи), тем больше погрешность привязки стробов к середине битового интервала, и требования к

согласованности частот становятся более строгие. Чем выше частота передачи, тем больше влияние искажений фронтов на фазу принимаемого сигнала. Взаимодействие этих факторов приводит к повышению требований к согласованности частот приемника и передатчика с ростом частоты обмена.

Рис. 4. Формат асинхронной передачи RS-232C

Формат асинхронной посылки позволяет выявлять возможные ошибки передачи.

- Если принят перепад, сигнализирующий о начале посылки, а по стробу старт-бита зафиксирован уровень логической единицы, старт-бит считается ложным и приемник снова переходит в состояние ожидания. Об этой ошибке приемник может не сообщать.
- Если во время, отведенное под стоп-бит, обнаружен уровень логического нуля, фиксируется ошибка стоп-бита.
- Если применяется контроль четности, то после посылки бит данных передается контрольный бит. Этот бит дополняет количество единичных бит данных до четного или нечетного в зависимости от принятого соглашения. Прием байта с неверным значением контрольного бита приводит к фиксации ошибки.
- Контроль формата позволяет обнаруживать обрыв линии: как правило, при обрыве приемник “видит” логический нуль, который сначала трактуется как старт-бит и нулевые биты данных, но потом срабатывает контроль стоп-бита.

Для асинхронного режима принят ряд стандартных скоростей обмена: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 и 115200 бит/с. Иногда вместо единицы измерения “бит/с” используют “бод” (baud), но при рассмотрении двоичных передаваемых сигналов это некорректно. В бодах принято измерять частоту изменения состояния линии, а при недвоичном способе кодирования (широко применяемом в современных модемах) в канале связи скорости передачи бит (бит/с) и изменения сигнала (бод) могут отличаться в несколько раз.

Количество бит данных может составлять 5, 6, 7 или 8 (5- и 6-битные форматы распространены незначительно). Количество стоп-бит может быть 1, 1,5 или 2 (“полтора бита” означает только длительность стопового интервала).

В данной работе настройки параметров передачи производятся пользователем в окне “Connection”.

1.4. Реализация физического уровня

1.4.1. Открытие порта

В ОС Windows и Linux доступ к COM-портам предоставляется посредством файловых дескрипторов. Для работы с портом используется библиотека RXTX. Используются 2 класса: **SerialPort** и **SerialPortEventListener**. Первый – для чтения и записи в порт, а от второго нужно наследовать свой класс для обработки событий изменения логический уровней на сигнальных линиях интерфейса RS-232C.
Для открытия порта используется функция **SerialPort.open()**.

После открытия порта устанавливаются параметры передачи, выбранные пользователем через графический интерфейс. Для этого используется метод **setSerialPortParams()** и **setFlowControlMode()**.

Затем библиотека предоставляет 2 потока:

- **serialPort.getOutputStream();**
  для записи
- **serialPort.getInputStream();**
  для чтения

И текущий объект класса ComPort передаётся в библиотеку RXTX и назначается обработчиком событий изменения уровня сигнальных линий интерфейса RS-232C.

1.4.2. Закрытие порта

Вначале закрываются оба потока данных (входной и выходной), сбрасываются значения RTS и DTR, а затем закрывается и сам объект ком-порта:

```
outStream.close();
inStream.close();
serialPort.setRTS(false);
serialPort.setDTR(false);
serialPort.close();
```

1.4.3. Передача и прием данных

Для отправки используется выходной поток и его метод **write()**:
public void write(byte b[])
Для получение данных – **read()**:
public abstract int read()

2. КАНАЛЬНЫЙ УРОВЕНЬ

2.1. Защита передаваемой информации

При передаче данных по линиям, входящим в коммутируемую сеть, чаще всего возникают ошибки, обусловленные электрическими помехами. Эти помехи в свою очередь могут вызвать ошибки в цепочке или пакете последовательных битов.

Для обнаружения ошибок применяют разнообразные корректирующие коды. Например: линейный код, код Хемминга, циклический код, логический код 4B/5B.

Для практической реализации циклических кодов могут быть использованы схемы с регистрами сдвига и обратными связями, или таблицы (в программе используется полином X^3+X+1).

В рамках данной курсовой работы необходима реализация алгоритма кодирования циклическим кодом.

Рассмотрим классический алгоритм циклического кода:

1. Задана информационная последовательность m(x). Умножить заданный полином степени (k-1) на $x^{(n-k)}$, т.е. сдвинуть в сторону старших разрядов на (n-k); где n = r+k , r - степень образующего полинома , k - число информационных разрядов данной последовательности;
2. Получить остаток от деления полинома $x^{(n-k)}*m(x)$ на g(x) - образующий полином. Степень остатка ≤ n-k-1
3. Объединить остаток p(x) и исходный полином $x^{(n-k)}*m(x)$ для получения кодового слова; p(x)@ $x^{(n-k)}*m(x)$, где @ - конкатенация;

Декодирование циклического кода:

V(x) - передаваемый кодовый полином; r(x) - принятый;

r(x)=g(x)*q(x)+S(x), где q(x) - частное, S(x) - остаток от деления переданного полинома на порождающий полином;

$S(x)=S0+S1*x+...+S(n-k-1)*x^{(n-k-1)}$ - синдром ошибки (если S(x) = 0, ошибки нет или она не обнаружена)

r(x)=V(x)+e(x), где e(x) - вектор ошибки;

e(x)=V(x)+q(x)*g(x)+S(x)=[m(x)+q(x)]*g(x)+S(x)

Задача декодирующего устройства - оценка вектора g(x) по синдрому ошибки.

По значению синдрома ошибки определяется бит, в котором была обнаружена ошибка:

Синдром	Номер бита
001	0
010	1
100	2
011	3
110	4
111	5
101	6

В данной работе применяется табличный метод кодирование и декодирования, основанный на том, что каждые 4 бита входного потока кодируются независимо от остальных данных и шифрование можно выполнить простым отображением по таблице, которую нужно было лишь единожды сгенерировать классическим алгоритмом. Декодирование осуществляется как обратное отображение из множества закодированных значений в множество раскодированных (по 4 бита, т.е. всего 16 возможных комбинаций). Таким образом, если обратное отображение невозможно, то кадр считается повреждённым и в ответ отсылается кадр RET.

2.2. Передача данных

На обоих компьютерах задаются параметры COM-порта (скорость передачи, количество информационных и стоповых бит, проверка чётности). Скорости COM-портов на передатчике и приемнике должны совпадать для корректной передачи.

После подключения второго компьютера начинается обмен кадрами. На канальном уровне предусмотрены две очереди на отправку кадров. Это позволяет дать больший приоритет служебным (управляющим кадрам). Таким образом, при получении сообщения с канального уровня оно десериализуется в байтовую последовательность, разделяется на

части, в случае превышение максимального размера кадра, затем всё это кодируется циклическим кодом и помещается в очередь на отправку, откуда извлекается по одному и при возможности передаётся отдельным поток на физический уровень.

Кадр не удаляется из очереди при отправке до получения положительной квитанции (кадра ACK) от другого компьютера. В случае истечения таймаута или получения негативной квитанции (RET) последний кадр вновь пересылается.

На приёме кадр расшифровывается, в случае успеха из кадра собирается сообщение (как массив байт) и результат передаётся на прикладной уровень. В противном случае отсылается RET.

2.3. Функции кодирования/декодирования

Кодирование и декодирование данных в программе осуществляется циклическим кодом с помощью класса **CycleCoder** и его методов:

- **public byte[] encode(byte[] data)**
- **public byte[] decode(byte[] data) throws DecodeException**

2.4. Форматы кадров

Типы кадров:
I - информационный;
S – управляющий;

Формат кадра:
[стартовый байт][тип кадра][ACK][RET][CH][END][RESERVED][Информационные байты][стоповый байт].

Стартовый байт: признак начала кадра (1 байт);
Тип кадра: 0 – информационный, 1 – управляющий (1 байт);
ACK: признак того, что данный кадр – кадр подтверждения (1 бит);
RET: признак того, что данный кадр – кадр повтора отсылки (1 бит);
CH: признак того, что данный кадр – чанк целого кадра (1 бит);
END: признак того, что данный кадр – конец цепочки чанков (1 бит);
RESERVED: 4 зарезервированных бита;
Информационные байты: (<= 126 байт в незакодированном виде)
Стоповый байт: признак конца кадра (1 байт);

3. ПРИКЛАДНОЙ УРОВЕНЬ

Работа прикладного уровня является практически прозрачной.

3.1. Передача сообщений

Передача сообщений осуществляется при помощи метода **send()**:
public int send(Message.Type type, String msg)
Здесь сообщению присваивается уникальный номер, если его тип – **Msg**, сообщение сериализуется и передаётся на канальный уровень.

3.2. Приём сообщений

Для приёма сообщений предусмотрена предварительная подписка на оповещение о доставке сообщения:
public void subscribeToReceive(final Consumer<Message> receiver).
При приёме сообщения с канального уровня оно десериализуется и передаётся в обработчики всех подписчиков в методе **receive()**:
public void receive(byte[] data).

3.3. Типы сообщений

Предусмотрено три типа сообщений:

- **Msg** – обычное информационное сообщение
- **Auth** – передаётся при подключении и содержит ник пользователя
  Ack – шлётся в ответ на **Msg** для графического обозначения доставки сообщения пользователю

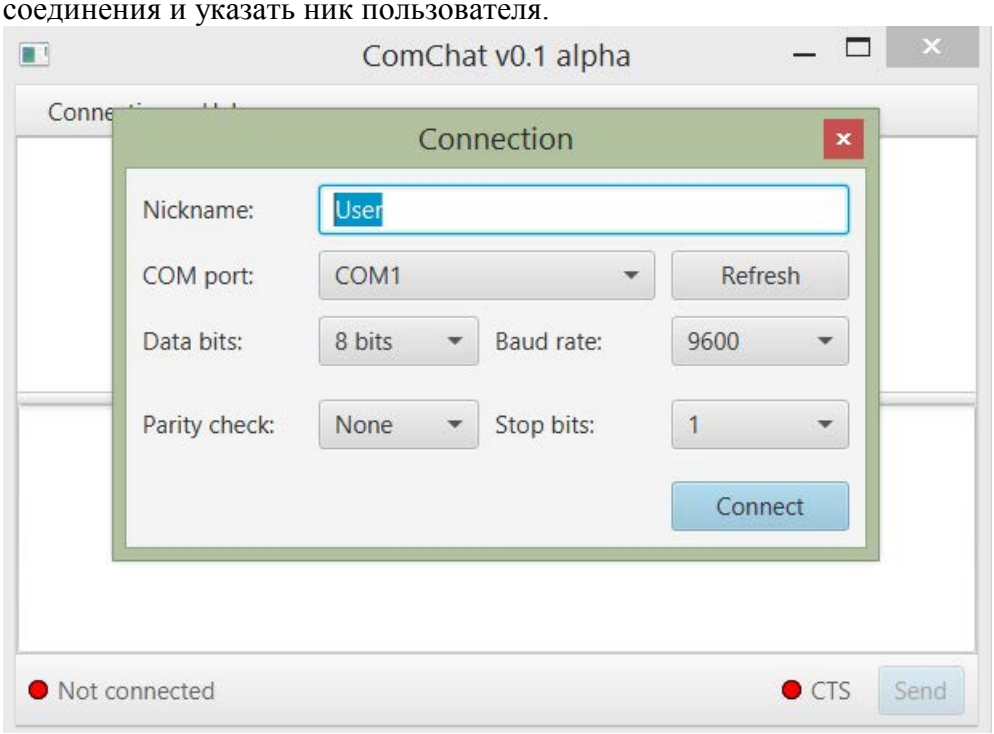
4. ГРАФИЧЕСКИЙ ИНТЕРФЕЙС

Графический интерфейс реализован при помощи библиотек JavaFX и ControlsFX.

4.1. Формы

4.1.1. Форма подключения

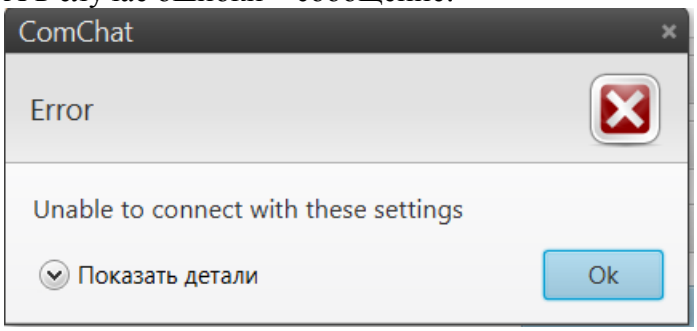
Данная форма открывается сразу при запуске приложения и позволяет выбрать настройки соединения и указать ник пользователя.



В списке **COM-port** выбирается один из доступных ком-портов. При использовании виртуальных портов бывает актуальной кнопка **Refresh**, которая обновляет список комп-портов. В списке **Baud rate** выбирается скорость передачи данных, в **Data bits** – количество информационных битов, в **Stop bits** – стоповых. В списке **Parity check** выбирается способ проверки чётности:

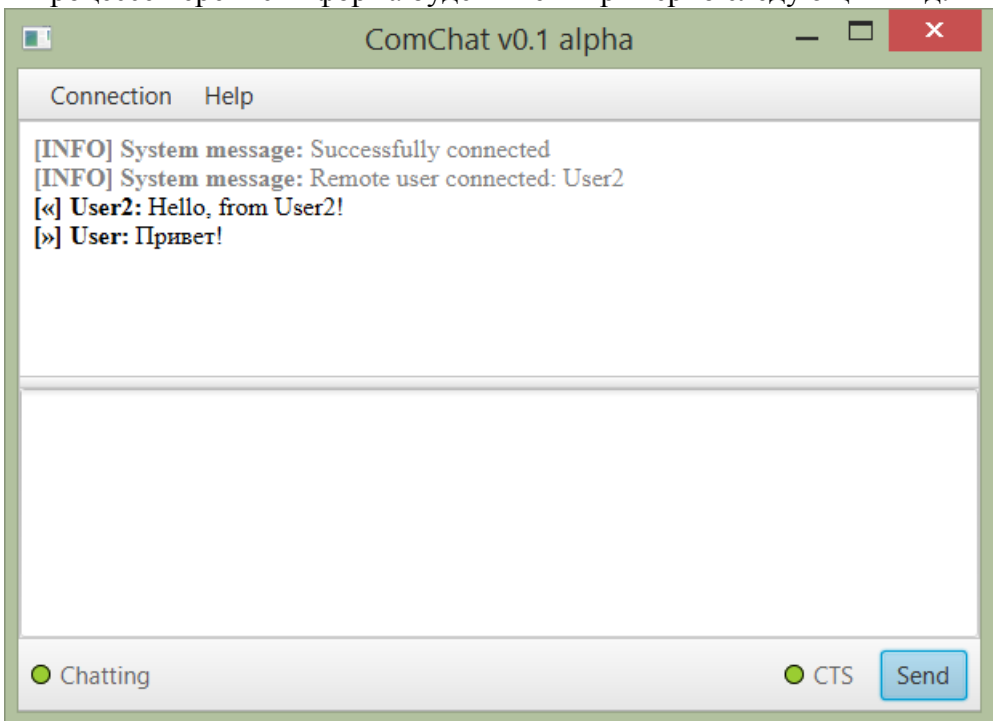
- **Even** - число единиц всегда чётно
- **Mark** - бит четности всегда = 1
- **None** - нет контроля четности
- **Odd** - нечетное число единиц
- **Space** - бит четности = 0

В поле **Nickname** требуется указать произвольное имя пользователя, оно будет передано на другой узел.
После того как все настройки выбраны можно нажать Connect и в случае успеха в главном окне чата выведется строка:
[INFO] System message: Successfully connected.
А в случае ошибки – сообщение:

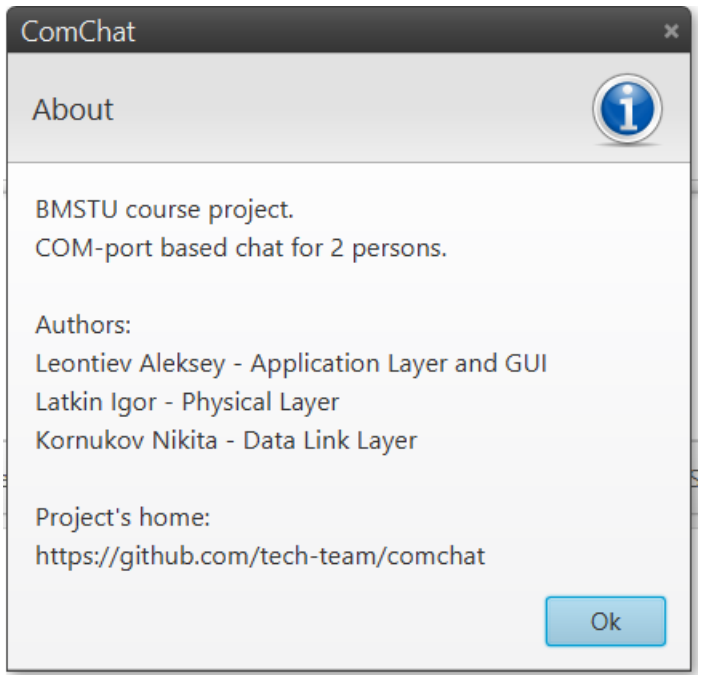


4.1.2. Главная форма

В процессе переписки форма будет иметь примерно следующий вид:



В верхней части меню расположено меню. В меню **Connection** есть два подпункта: **Connect**, который вызывает окно настройки подключения, описанное выше, и **Disconnect**, который принудительно разрывает связь. В меню Help всего один подпункт – **About**, который выводит информацию о программе:

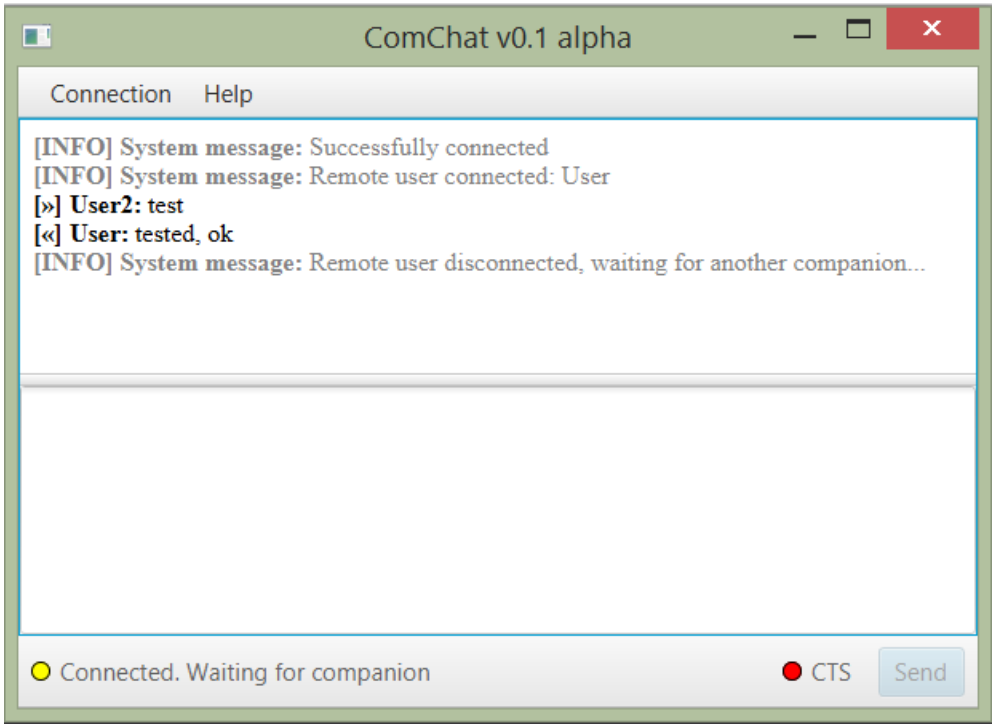


Под меню расположено основное окно чата, где выводятся пользовательские и системные сообщения. При отправке сообщение почемчается символом [×], как только оно будет доставлено на противоположный узел и обратно придёт сообщение Ack, значёк смениться на [»]. Все полученные сообщения помечаются значком [«].
Под основным окном расположено поле ввода сообщения, отправка сообщения производится либо по надатию кнопки **Send**, либо **Enter**.
Под полем ввода расположены два индикатора. Левый индикатор имеет три состояния:

- **Not connected** – (красный) Нет соединения
- **Connected. Waiting for companion…** - (жёлтый) Ожидание собеседника
- **Chatting** – (зелёный) Собеседник подключен и возможна переписка

Правый индикатор имеет два состояния (красный или зелёный) и отображает состояние линии CTS, полученное от физического уровня.

Для разрыва соединения можно либо сделать это вручную через меню **Connection -> Disconnect**, либо просто закрыть программу. Программа на удалённом узле перейдёт в состояние ожидание собеседника:


ComChat v0.1 alpha
Connection
Help
[INFO] System message: Successfully connected
[INFO] System message: Remote user connected: User
[»] User2: test
[«] User: tested, ok
[INFO] System message: Remote user disconnected, waiting for another companion...
Connected. Waiting for companion
CTS
Send