Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет ,информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Основы компьютерных сетей

ОТЧЕТ

по лабораторной работе № 2

на тему

ПАКЕТНАЯ ПЕРЕДАЧА ДАННЫХ

Студент: Ю. Л. Спасёнов

Преподаватель: В. А. Марцинкевич

МИНСК 2024

**1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

**1.1 Исходные данные**

Для написания и отладки программы использовался IntelliJ IDEA. В качестве языка программирования был выбран язык Java и библиотека JavaFX для создания графического интерфейса (GUI).

Для эмуляции COM-портов использовался socat.

**1.2 Алгоритм работы бит стаффинга**

Алгоритм стаффинга, реализует добавление нулевых битов после пяти последовательных единичных битов в исходных данных (сторона передатчика), а затем их удаление (сторона приемника). Это делается для предотвращения ошибок при передаче данных, если последовательности единиц могут привести к ошибочной интерпретации данных как контрольных флагов.

Описание работы:

Сторона передатчика:

1. Чтение каждого байта данных.
2. Пробег по каждому биту байта, начиная с самого значащего (бит 7).
3. Добавление бита в результирующий буфер: Если бит 1, увеличивается счетчик последовательных единиц. Если достигнуто 5 последовательных единиц, добавляется 0 (для стаффинга), и счетчик сбрасывается. Если бит 0, счетчик последовательных единиц сбрасывается.
4. Если в конце останутся неполные байты (меньше 8 бит), они заполняются нулями и записываются в выходные данные.

Сторона приемника:

1. Чтение каждого байта застаффированных данных.
2. Пробег по каждому биту байта, начиная с самого значащего (бит 7).
3. Добавление бита в исходные данные: Если бит 1, увеличивается счетчик последовательных единиц. Если достигнуто 5 последовательных единиц, следующий бит пропускается (поскольку это застаффированный 0), и счетчик сбрасывается. Если бит 0, счетчик последовательных единиц сбрасывается.
4. Если в конце останутся неполные байты, они дополняются нулями.

Схема передачи данных (передатчик): Чтение бита → 2. Проверка на единицу → 3. Если 5 подряд — вставка 0 → 4. Запись бита в выходной поток → 5. Если меньше 8 бит — дополнение байта → Конец

Схема приема данных (приемник): Чтение бита → 2. Проверка на единицу → 3. Если 2 подряд — пропуск следующего бита → 4. Запись бита в исходный поток → 5. Если меньше 8 бит — дополнение байта → Конец

Алгоритм работы бит-стаффинга использовавшийся в данной работе изображен на рисунке 1.1.

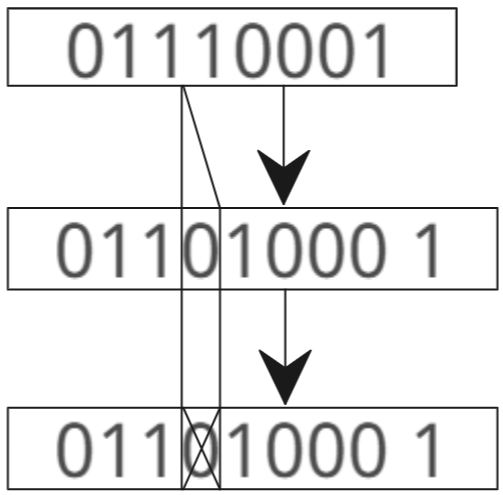


Рисунок 1.1 – Алгоритм работы бит-стаффинга

Результат работы компьютерной программы изображен на рисунке 1.2.

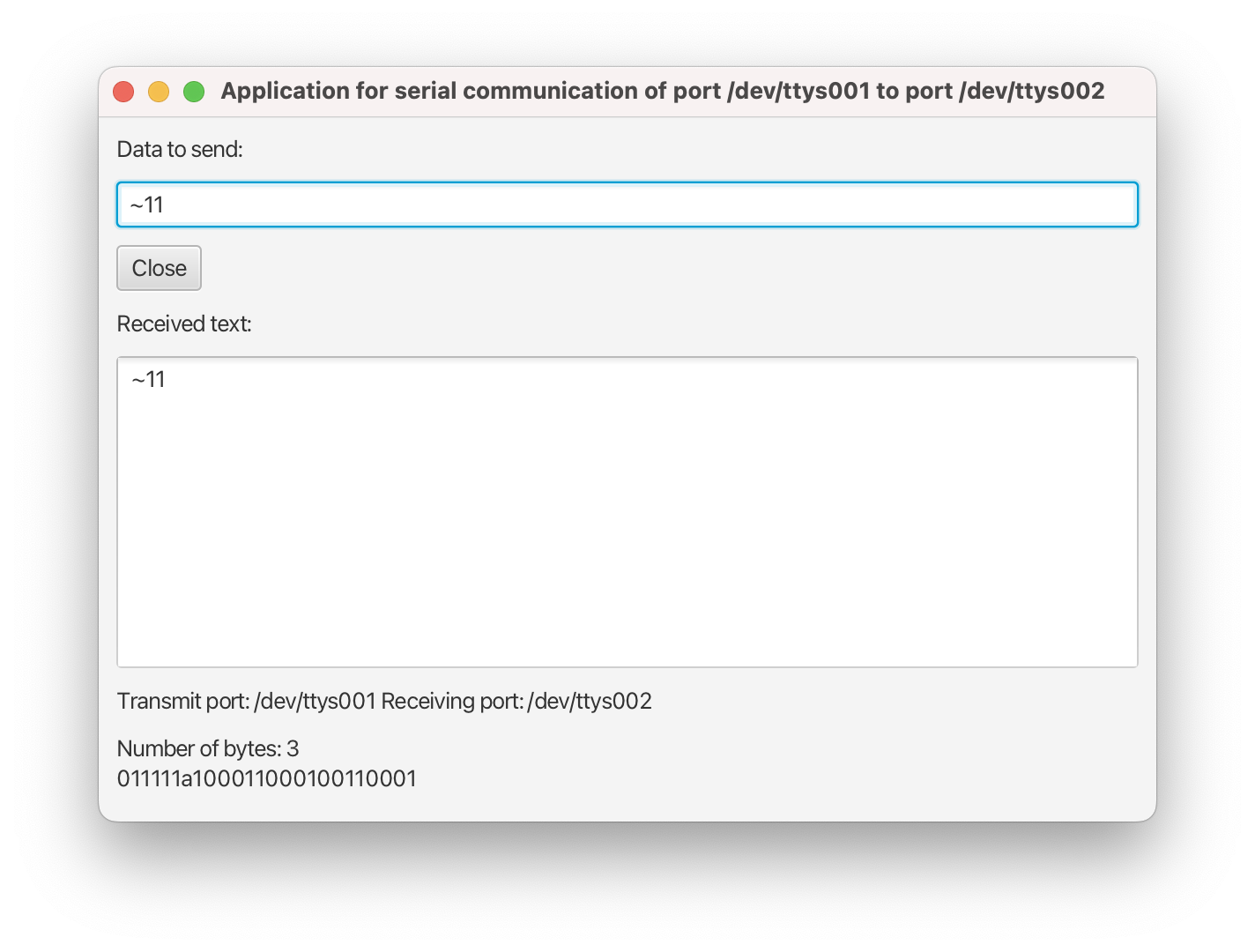


Рисунок 1.2 – Результат работы компьютерной программы

**2 ПРАКТИЧЕСКАЯ ЧАСТЬ**

**2.1 Код программы**

public void serialEvent(SerialPortEvent event) {

if (event.getEventType() != SerialPort.LISTENING\_EVENT\_DATA\_AVAILABLE) {

return;

}

byte[] newData = new byte[port.bytesAvailable()];

ArrayList<String> dynamicStringArray = new ArrayList<>();

ArrayList<String> extractedArray = new ArrayList<>();

int index = 0;

while (port.readBytes(newData, newData.length) > 0) {

String receivedData = new String(newData);

String extracted = extractDataFromPackets(receivedData);

extractedArray.add(extracted);

String debitStaffing = debitStaffing(extracted);

dynamicStringArray.add(binaryStringToAscii(debitStaffing));

index++;

newData = new byte[port.bytesAvailable()];

}

bitStuffing = extractedArray;

dataReceivedTextArea.clear();

Platform.runLater(() -> {

StringBuilder sb = new StringBuilder();

for (String str : dynamicStringArray) {

sb.append(str);

}

dataReceivedTextArea.appendText(sb.toString());

});

}

public static String binaryStringToAscii(String binaryString) {

if (!binaryString.matches("[01]+")) {

return "Некорректная строка бинарных данных";

}

StringBuilder asciiBuilder = new StringBuilder();

int remainder = binaryString.length() % 8;

if (remainder != 0) {

binaryString = "0".repeat(8 - remainder) + binaryString;

}

for (int i = 0; i < binaryString.length(); i += 8) {

String binaryChar = binaryString.substring(i, i + 8);

int asciiValue = Integer.parseInt(binaryChar, 2);

char asciiChar = (char) asciiValue;

asciiBuilder.append(asciiChar);

}

return asciiBuilder.toString();

}

public static String extractDataFromPackets(String dataStream) {

String marker = "200000000";

StringBuilder extractedData = new StringBuilder();

int markerIndex = dataStream.indexOf(marker);

while (markerIndex != -1) {

int skipIndex = markerIndex + marker.length() + 1;

markerIndex = dataStream.indexOf(marker, skipIndex);

if (markerIndex == -1) {

markerIndex = dataStream.length();

}

extractedData.append(dataStream, skipIndex, markerIndex);

if (markerIndex == dataStream.length())

break;

}

return extractedData.toString();

}

public static boolean[] bytesToBits(byte[] bytes) {

boolean[] bits = new boolean[bytes.length \* 8];

int bitIndex = 0;

for (byte b : bytes) {

for (int i = 7; i >= 0; i--) {

bits[bitIndex++] = (b & (1 << i)) != 0;

}

}

return bits;

}

public static String ByteArrayToString(byte[] bytes) {

StringBuilder sb = new StringBuilder();

for (byte b : bytes) {

sb.append(String.format("%X", b));

}

return sb.toString();

}

public byte[] convertStringToByteArray(String port) {

byte[] portBytes = new byte[port.length()];

for (int i = 0; i < port.length(); i++) {

portBytes[i] = Byte.parseByte(String.valueOf(port.charAt(i)));

}

return portBytes;

}

public String booleanArrayToString(boolean[] booleanArray) {

StringBuilder sb = new StringBuilder();

for (boolean b : booleanArray) {

sb.append(b ? '1' : '0');

}

return sb.toString();

}