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 № 1

на тему

«работа с COM-портом»

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 Параметры инициализации COM-порта**

Java предоставляет классы SerialPort для работы с COM-портами.

Объект класса SerialPort имеет следующие параметры:

* BaudRate – выходная частота бод-генератора или скорость передачи данных. Для UART 16550 один бод (baud) равен одному биту в секунду. Задается методом setBaudRate(). Возможные значения могут высчитываться по формуле:

Fout = Fin / (16 \* DL),

где Fin – входная частота;

DL – шестнадцатибитная константа, старшая и младшая части которой хранятся в двух регистрах UART (DLL и DLM).

Стандартные значения (Hz): 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.

По умолчанию задана частота 9600 Hz.

* DataBits – количество битов в одном байте. Устанавливается с помощью метода setDataBits(). Может принимать значения от 5 до 8. По умолчанию размер байта равен 8 битам.
* Parity – паритет (четность). Этот параметр нужен для проверки целостности информационной части. Паритет может принимать следующие значения, определенные в классе QSerialPort: NoParity (без четности), OddParity (проверка на нечетность единиц в байте, EvenParity (проверка на четность единиц в байте), MarkParity (проверка на четность/нечетность отсутствует, но бит паритета равен единице), SpaceParity (проверка на четность/нечетность отсутствует, но бит паритета равен нулю). При OddParity бит паритета устанавливается в 0 или 1 в зависимости от количества нечетных единиц в байте, при EvenParity – в зависимости от количества четных единиц. По умолчанию – NoParity.
* StopBits – стоп-биты. По умолчанию линия находится в состоянии логической единицы. При наличии байта для передачи передатчик переводит линию в состояние логического нуля, т. е. передает старт-бит, что говорит приемнику о том, что на следующем такте нужно “ловить” первый информационный бит. Стоп-бит необходим для того, чтобы после передачи информационной последовательности гарантированно вернуть линию в исходное, т.е. единичное состояние. Старт-бит всегда один, а стоп-битов может быть 1, 1.5 либо 2. По умолчанию устанавливается 1 стоп-бит.
* FlowControl – управление потоком. Этот параметр, к примеру, позволяет избегать переполнения приемника, приостанавливая “быстрый” передатчик.

Возможные значения: NoFlowControl (без управления потоком), HardwareControl (аппаратное управление потоком), SoftwateControl (программное управление потоком). По умолчанию устанавливается в NoFlowControl.

**2 ПРАКТИЧЕСКАЯ ЧАСТЬ**

**2.1 Код программы**

Файл main.java

package com.example.lab1;

import com.example.lab1.port.SerialPortManager;

import javafx.application.Application;

import javafx.stage.Stage;

import static com.example.lab1.UI.ConfigWindow.createConfigWindow;

public class Main extends Application {

@Override

public void start(Stage stage) {

SerialPortManager portManager = new SerialPortManager();

createConfigWindow(portManager);

}

public static void main(String[] args) {

launch();

}

}

Файл PortWindow.java

package com.example.lab1.UI;

import com.example.lab1.port.SerialPortManager;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextArea;

import javafx.scene.control.TextField;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import static com.example.lab1.UI.ConfigWindow.createConfigWindow;

public class PortWindow {

public static void createPortWindow(SerialPortManager portManager, TextArea dataReceivedTextArea) {

Stage stage = new Stage();

stage.setTitle("Application for serial communication of port " +

portManager.getPortTransfer().getSystemPortPath() +

" to port " +

portManager.getPortReceiving().getSystemPortPath());

TextField dataToSendTextArea = new TextField();

Label dataToSendTextAreaLabel = new Label("Select the COM port for data transfer:");

Button closeButton = new Button("Close");

Label statusInformationPortsLabel = new Label("Transfer port: " +

portManager.getPortTransfer().getSystemPortPath() +

" Receiving port: " +

portManager.getPortReceiving().getSystemPortPath());

Label statusInformationBytesLabel = new Label();

dataToSendTextArea.setOnAction(actionEvent -> {

String data = dataToSendTextArea.getText();

portManager.SendAction(data);

statusInformationBytesLabel.setText(" Number of bytes: " +

dataToSendTextArea.getText().getBytes().length);

});

closeButton.setOnAction(actionEvent -> {

stage.close();

portManager.closePorts();

createConfigWindow(portManager);

dataReceivedTextArea.clear();

});

dataReceivedTextArea.setEditable(false);

Label dataReceivedTextAreaLabel = new Label("Select the COM port for data transfer:");

VBox root = new VBox();

root.setSpacing(10);

root.setPadding(new Insets(10));

root.getChildren().addAll(dataToSendTextAreaLabel,

dataToSendTextArea,

closeButton,

dataReceivedTextAreaLabel,

dataReceivedTextArea,

statusInformationPortsLabel,

statusInformationBytesLabel);

Scene scene = new Scene(root, 600, 400);

stage.setScene(scene);

stage.show();

}

}

Файл ErrorWindow.java

package com.example.lab1.UI;

import javafx.scene.control.Alert;

import javafx.scene.control.ButtonType;

public class ErrorWindow {

public static void createErrorWindow(String errorMessage) {

Alert alert = new Alert(Alert.AlertType.ERROR, errorMessage, ButtonType.OK);

alert.setTitle("Error");

alert.setHeaderText(null);

alert.showAndWait();

}

}

Файл ConfigWindow.java

package com.example.lab1.UI;

import com.example.lab1.port.SerialPortManager;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ComboBox;

import javafx.scene.control.Label;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.List;

import static com.example.lab1.UI.PortWindow.createPortWindow;

import static com.example.lab1.UI.ErrorWindow.createErrorWindow;

public class ConfigWindow {

public static void createConfigWindow(SerialPortManager portManager) {

Stage stage = new Stage();

stage.setTitle("Config Serial Communication App");

try {

List<String> devFiles = Files.list(Paths.get("/dev"))

.map(Path::getFileName)

.map(Path::toString)

.filter(s -> s.matches("ttys00[0-9]"))

.sorted()

.toList();

Label portTransferLabel = new Label("Select the COM port for data transfer:");

ComboBox<String> portTransferComboBox = new ComboBox<>();

portTransferComboBox.getItems().addAll(devFiles);

portTransferComboBox.setValue("No port");

Label portReceivingLabel = new Label("Select the COM port for receiving data:");

ComboBox<String> portReceivingComboBox = new ComboBox<>();

portReceivingComboBox.getItems().addAll(devFiles);

portReceivingComboBox.setValue("No port");

Label SpeedTxLabel = new Label("Speed Tx");

ComboBox<Integer> speedTxComboBox = new ComboBox<>();

speedTxComboBox.getItems().addAll(9600, 19200, 38400, 57600, 115200);

speedTxComboBox.setValue(9600);

Label SpeedRxLabel = new Label("Speed Rx");

ComboBox<Integer> speedRxComboBox = new ComboBox<>();

speedRxComboBox.getItems().addAll(9600, 19200, 38400, 57600, 115200);

speedRxComboBox.setValue(9600);

Button saveButton = new Button("Save");

saveButton.setOnAction(actionEvent -> {

if (speedTxComboBox.getValue() == null) {

createErrorWindow("speedTx is empty");

}

if (speedRxComboBox.getValue() == null) {

createErrorWindow("speedRx is empty");

} portManager.initSerialPorts(portTransferComboBox.getValue(),

portReceivingComboBox.getValue(),

speedTxComboBox.getValue(),

speedRxComboBox.getValue());

createPortWindow(portManager, portManager.getDataReceivedTextArea());

stage.close();

});

Button closeButton = new Button("Close");

closeButton.setOnAction(actionEvent -> {

stage.close();

portManager.closePorts();

});

VBox root = new VBox();

root.getChildren().addAll(

portTransferLabel, portTransferComboBox,

portReceivingLabel, portReceivingComboBox,

SpeedTxLabel, speedTxComboBox,

SpeedRxLabel, speedRxComboBox,

saveButton, closeButton

);

root.setSpacing(10);

root.setPadding(new Insets(10));

Scene scene = new Scene(root, 400, 350);

stage.setScene(scene);

stage.show();

} catch (IOException e) {

e.printStackTrace();

}

}

}

**2.2 Интерфейс программы**

Интерфейс настройки портов представлен на рисунке 2.1.

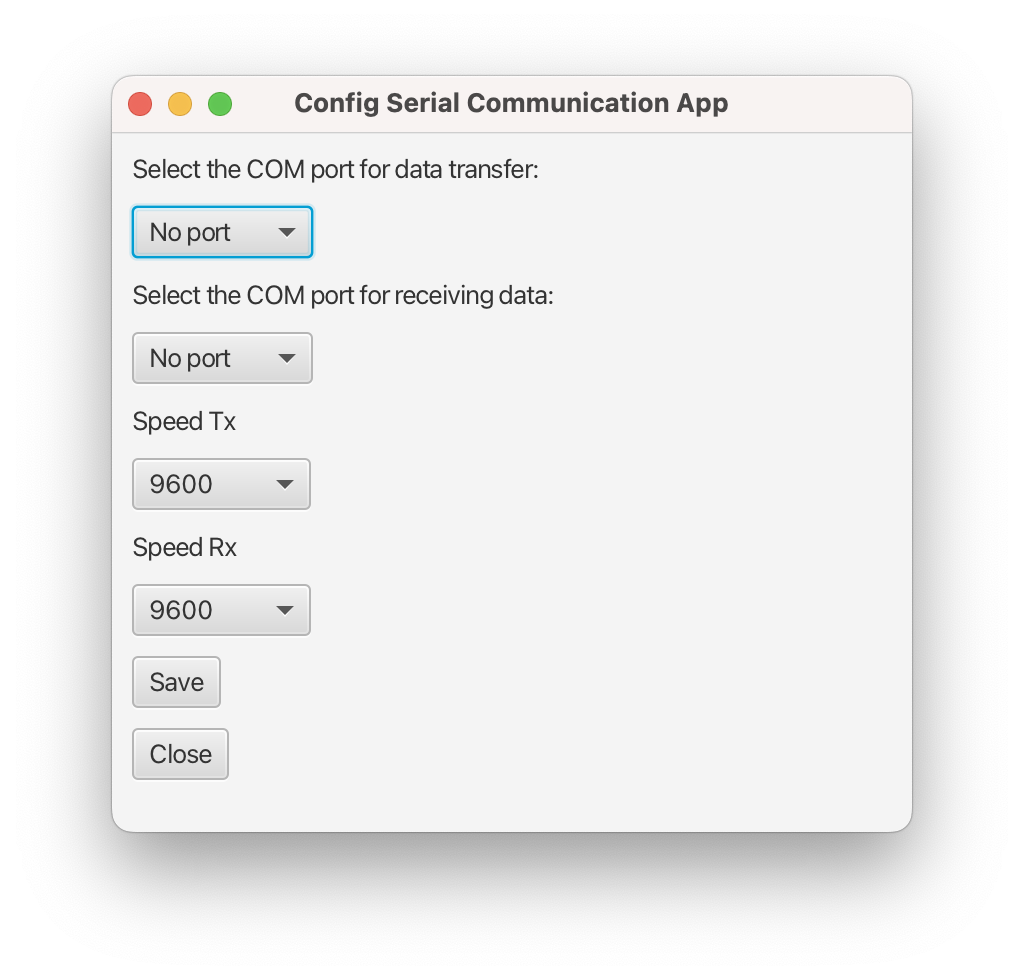


Рисунок 2.1 – Интерфейс настройки портов

Интерфейс передачи передачи представлен на рисунке 2.2.

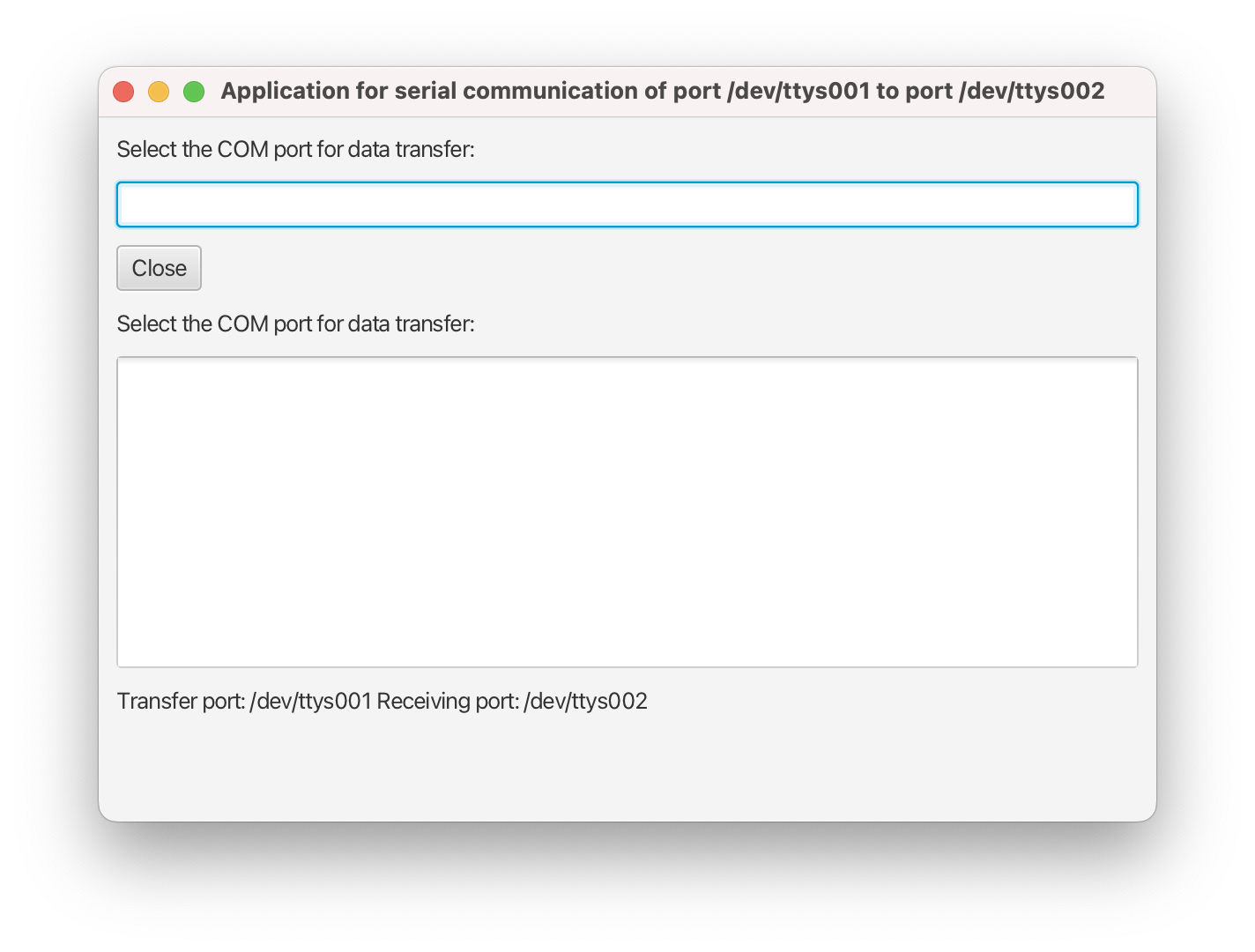


Рисунок 2.2 – Интерфейс передачи передачи