Міністерство освіти та науки України

НТУ «Дніпровська політехніка»

Звіт

з лабораторної

Виконав

Студент групи 123-17-1

Ікол Данило

Дніпро 2020р

Завдання

Реализация Мастера (графический интерфс две кнопки с фиксацией, две метки) один раз в секунду опрашивает состояние Слейва с помощью протокола модбас тсп, слейв отвечает состояние своих двух кнопок с фиксацией, и в зависимости от их состояния (нажато/ не нажато) мастер индицирует их в соотвестующих метках меня цвет. При нажатии кнопки на мастере отправляется комманда для слейва, который в свою очередь меняет цвет соответвующих своих меток

Код програми

Client:

```java
public class App extends Application {
    private double xOffset = 0.0D;
    private double yOffset = 0.0D;
    private Parent root;

    public App() { root = null; }

    public void start(Stage primaryStage) {
        try {
            root = FXMLLoader.load(Objects.requireNonNull(getClass().getClassLoader().getResource( name: "client/client_ui.fxml")));
        } catch (IOException e) {
            System.out.println(("File with UI not found!"));

        }

        initMouseEvents(root, primaryStage);

        Scene mainScene = new Scene(root);

        mainScene.setFill(Color.TRANSPARENT);

        primaryStage.setTitle("Client");
        primaryStage.setScene(mainScene);
        primaryStage.initStyle(StageStyle.TRANSPARENT);
        primaryStage.show();
    }

    private void initMouseEvents(Parent root, Stage primaryStage) {
        root.setOnMousePressed(event -> {
            xOffset = primaryStage.getX() - event.getScreenX();
            yOffset = primaryStage.getY() - event.getScreenY();
            primaryStage.setOpacity(0.9D);

        });
        root.setOnMouseReleased(event -> {
            primaryStage.setOpacity(1.0D);

        });
        root.setOnMouseDragged(event -> {
            primaryStage.setX(event.getScreenX() + xOffset);
            primaryStage.setY(event.getScreenY() + yOffset);

        });
    }

    public static void main(String[] args) { launch(args); }
}
```

```java
public class ClientController {

    private final ModbusClient client = new ModbusClient();
    private final ClientService service = new ClientService();
    private Timeline timeline = new Timeline();

    @FXML
    private JFXToggleButton firstToggle;
    @FXML
    private JFXToggleButton secondToggle;
    @FXML
    private JFXButton connectionBtn;
    @FXML
    private JFXButton exitBtn;
    @FXML
    private JFXTextField ipField;
    @FXML
    private JFXTextField portField;
    @FXML
    private Label headLabel;

    public void connect(ActionEvent event) {
        if (connectionBtn.getText().equals("CONNECT")) {
            service.init(client, ipField.getText(), portField.getText());
            service.connect(client);
            setConnectStatus();
            updateStatus();
        } else if (connectionBtn.getText().equals("DISCONNECT")) {
            try {
                service.disconnect(client);
                setDisconnectedStatus();
                timeline.stop();
            } catch (Exception e) {
                headLabel.setText(e.getMessage());
            }

        }
    }

    public void exit(ActionEvent event) { System.exit( status: 0); }
```

```
public void toggleAction(ActionEvent event) {
    if (firstToggle.isSelected()) {
        service.setCoilValue(client,  coil: 0,  value: true);
    } else {
        service.setCoilValue(client,  coil: 0,  value: false);
    }

    if (secondToggle.isSelected()) {
        service.setCoilValue(client,  coil: 1,  value: true);
    } else {
        service.setCoilValue(client,  coil: 1,  value: false);
    }
}

private void setConnectStatus() {
    headLabel.setText("CONNECTED");
    connectionBtn.setStyle("-fx-background-color: red");
    connectionBtn.setText("DISCONNECT");
    firstToggle.setDisable(false);
    secondToggle.setDisable(false);
    ipField.setEditable(false);
    portField.setEditable(false);
}


private void setDisconnectedStatus() {
    headLabel.setText("CLIENT");
    connectionBtn.setStyle("-fx-background-color: black");
    connectionBtn.setText("CONNECT");
    firstToggle.setDisable(true);
    secondToggle.setDisable(true);
    ipField.setEditable(true);
    portField.setEditable(true);
}


private void updateStatus() {
    timeline = new Timeline(new KeyFrame(Duration.seconds(1.0D), event -> {
        try {
            boolean[] toggleState = client.ReadCoils( startingAddress: 0,  quantity: 2);
            System.out.println("First coil value: " + toggleState[0] + " \n" +
                    "Second coil value: " + toggleState[1]);
            firstToggle.setSelected(toggleState[0]);
            secondToggle.setSelected(toggleState[1]);
        } catch (Exception e) {
            headLabel.setText("UNKNOWN ERROR");
            System.out.println(e.getMessage());
        }
    }
    ));
    timeline.setCycleCount(-1);
    timeline.play();
}
```

```java
public class ClientService {

    public void init(ModbusClient client, String ip, String stringPort) {
        int port = Integer.parseInt(stringPort);
        if (ip.matches( regex: "\\b(?:\\d{1,3}\\.){3}\\d{1,3}")) {
            if (stringPort.matches( regex: "\\d+") && port > 1023 && port < 65536) {
                client.setipAddress(ip);
                client.setPort(port);
            } else {
                throw new IllegalArgumentException("INVALID PORT NUMBER");
            }
        } else {
            throw new IllegalArgumentException("INVALID IP ADDRESS");
        }
    }

    public void connect(ModbusClient client) {
        try {
            client.Connect();
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }

    public void disconnect(ModbusClient client) throws IOException {
        client.Disconnect();
    }

    public void setCoilValue(ModbusClient client, int coil, boolean value) {
        try {
            client.WriteSingleCoil(coil, value);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

CLIENT

127.0.0.1 504

CONNECT

Server:

```
public class App extends Application {
    private double xOffset = 0.0D;
    private double yOffset = 0.0D;
    private Parent root;

    public App() { root = null; }


    public void start(final Stage primaryStage) {
        try {
            root = FXMLLoader.load(Objects.requireNonNull(getClass().getClassLoader().getResource( name: "server/server_ui.fxml")));
        } catch (IOException e) {
            System.out.println(("File with UI not found!"));

        }
        initMouseEvents(root, primaryStage);

        Scene mainScene = new Scene(root);
        mainScene.setFill(Color.TRANSPARENT);
        primaryStage.setTitle("SERVER");
        primaryStage.setScene(mainScene);
        primaryStage.initStyle(StageStyle.TRANSPARENT);
        primaryStage.show();
    }

    private void initMouseEvents(Parent root, Stage primaryStage) {
        root.setOnMousePressed(event -> {
            xOffset = primaryStage.getX() - event.getScreenX();
            yOffset = primaryStage.getY() - event.getScreenY();
            primaryStage.setOpacity(0.9D);
        });
        root.setOnMouseReleased(event -> {
            primaryStage.setOpacity(1.0D);
        });

        root.setOnMouseDragged(event -> {
            primaryStage.setX(event.getScreenX() + App.this.xOffset);

            primaryStage.setY(event.getScreenY() + App.this.yOffset);

        });


    }

    public static void main(String[] args) { launch(args); }
}
```

```java
public class ServerController implements ICoilsChangedDelegator {

    private ModbusServer server = new ModbusServer();
    private ServerService service = new ServerService();

    @FXML
    private JFXToggleButton firstToggle;
    @FXML
    private JFXToggleButton secondToggle;
    private FadeTransition fadeOut = new FadeTransition(Duration.millis(3000.0D));
    @FXML
    private JFXButton startServerBtn;
    @FXML
    private JFXButton exitBtn;
    @FXML
    private JFXTextField ipField;
    @FXML
    private JFXTextField portField;
    @FXML
    private Label headLabel;

    @FXML
    public void initialize() throws IOException {
        server.setNotifyCoilsChanged(this);
        portField.setText(generateEmptyPort());
        fadeOut.setFromValue(1.0D);
        fadeOut.setToValue(0.0D);
        fadeOut.setCycleCount(1);
        fadeOut.setAutoReverse(false);
    }

    public void start(ActionEvent event) {
        try {
            service.init(server, portField.getText());
            service.start(server);
            setServerStatus();
        } catch (Exception e) {
            headLabel.setText(e.getMessage());
        }
    }

    public void exit(ActionEvent event) { System.exit( status: 0); }
```

```
    public void toggleAction(ActionEvent event) {
        if (firstToggle.isSelected()) {
            service.setCoilValue(server,  coil: 1,  value: true);
        } else {
            service.setCoilValue(server,  coil: 1,  value: false);
        }

        if (secondToggle.isSelected()) {
            service.setCoilValue(server,  coil: 2,  value: true);
        } else {
            service.setCoilValue(server,  coil: 2,  value: false);
        }
    }

    private void setServerStatus() {
        headLabel.setText("SERVER STARTED");
        firstToggle.setDisable(false);
        secondToggle.setDisable(false);
        portField.setEditable(false);
        startServerBtn.setDisable(true);
    }

    public void coilsChangedEvent() {
        firstToggle.setSelected(server.coils[1]);
        secondToggle.setSelected(server.coils[2]);
    }

    private String generateEmptyPort() throws IOException {
        String port;
        ServerSocket socket = new ServerSocket( port: 0);
        port = String.valueOf(socket.getLocalPort());
        socket.close();
        return port;
    }
```

```
public class ServerService {
    public void init(ModbusServer server, String stringPort) {
        int port = Integer.parseInt(stringPort);

        if (stringPort.matches( regex: "\\d+") && port > 1023 && port < 65536) {
            server.setPort(port);
        } else {
            throw new IllegalArgumentException("INVALID PORT NUMBER");
        }
    }

    public void start(ModbusServer server) {
        try {
            server.Listen();
        } catch (IOException e) {
            System.out.println("Can't start server");
        }
    }

    public void setCoilValue(ModbusServer server, int coil, boolean value) { server.coils[coil] = value; }
}
```

SERVER

127.0.0.1

64473

START