```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package seafarm;



//Java FX////
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import com.sun.deploy.trace.TraceLevel;
import javafx.application.Application;
import javafx.geometry.HPos;
import javafx.geometry.VPos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.ColumnConstraints;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.RowConstraints;
import javafx.stage.Stage;
import eu.hansolo.medusa.Gauge;
import eu.hansolo.medusa.GaugeBuilder;
import java.io.ByteArrayInputStream;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Observable;
import java.util.Observer;
import javafx.animation.AnimationTimer;
import javafx.application.Platform;
import javafx.geometry.Insets;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Slider;
import javafx.scene.control.Tab;
import javafx.scene.control.TabPane;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
```

```
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
//Open CV//
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.videoio.VideoCapture;
import org.opencv.videoio.VideoWriter;
import de.re.easymodbus.modbusclient.ModbusClient;
/**
 *
 * @author Ole Morken
 */
public class SeaFarm extends Application implements Observer {

    //VideoStreamReciever
    //udpVideoReciever vidUdpOverview;
    udpVideoReciever vidUdpPlatform;
    udpVideoReciever vidUdpROV;
    maps maps;

    // MODBUS comunication///
    ModbusClient modbusClientRead;
    ModbusClient modbusClientSend;
    static final String MODBUSIPADRESS = "192.168.0.112";
    //recieve modbus
    ModbusReciever recieverModbus;
    RecieveDataObserver recieveObserver;
    //send modbus
    ModbusSender sendModbus;
    SendDataObserver sendObserver;


    //ROV variables///
    static final String ROVIPADDRESS = "192.168.0.100";
    static final int ROVRECEIVEPORT = 8765;
    static final int ROVSENDPORT = 9876;
    static final int ROVVIDEOPORT = 12342;
    static RovSendEventState enumStateEvent = RovSendEventState.FALSE;
    RovGUIController rovGuicontroller;
```

```
RovDatahandler rovDatahandler;
RovUDPreceiver rovUdpReceiver;
RovReceiveDataObservable rovReceiveDataObserver;



//platform video
static final int PLATVIDEOPORT = 12345;
autoMode autMode;









private static final int SCENE_W = 640;
private static final int SCENE_H = 480;

boolean stoprecPlatform;
boolean startRecPlatform;
boolean stoprecROV;
boolean startRecROV;
//VideoCapture videoCapture;
//Canvas canvas;

GraphicsContext g2dPlatform;
GraphicsContext g2dROV;
Stage window;

AnimationTimer timerPlatform;
AnimationTimer timerROV;
VideoWriter writerPlatform;
VideoWriter writerROV;

Gauge depthMeter;
Gauge temp;
double autoModeLatSetpoint;
double autoModeLongSetpoint;
float longitude;
float latitude;
float currentSpeed=0;
float heading=0;
```

```
float pitch=0;
float roll=0;
boolean enabled=false;
//winch variables//
boolean rovLocked=false;
boolean rovUpperPos=false;
boolean dpModeActivated = false;
boolean autoModeActivated = false;
boolean manualModeActivated = false;
int platformSpeedVal=0;

//mode indicators platform

Circle recCirclePlatform ;
Circle dpCirclePlatform ;
Circle autoCirclePlatform ;
Circle manualCirlcePlatform ;
Circle lockCircleROV ;
Circle upperCircleROV;

//textFields for sensordata

//text fields
   TextField headingPlatTextField ;
   TextField curretnSpeenTextField ;
   TextField pitchTextField ;
   TextField rollTextField;

   //rov
   TextField deptRovTextField;
   TextField tempRovTextField;
   TextField headingRovTextField;
   TextField tempInWaterTextField;
   TextField oxygenInWaterTextField;

@Override
public void start(Stage primaryStage) {
   window = primaryStage;
   initOpenCv();

   depthMeter = makeGauge("Depth", "Meter");
   temp = makeGauge("Temperature", "C");
   temp.setSubTitle("ROV");
   depthMeter.setSubTitle("ROV");
```

```
        temp.setValue(0);

         //modbus threads
        modbusClientRead= new ModbusClient(MODBUSIPADRESS, 502);
        modbusClientSend= new ModbusClient(MODBUSIPADRESS, 502);
        recieveObserver= new RecieveDataObserver();
        recieveObserver.addObserver(this);
        recieverModbus = new ModbusReciever(modbusClientRead
,recieveObserver);
        recieverModbus.start();

        //sender modbus
        sendModbus=new ModbusSender(modbusClientSend);
        sendObserver = new SendDataObserver();
        sendObserver.addObserver(sendModbus);
        sendModbus.start();

        //ROV ///
        rovDatahandler=new RovDatahandler();
        rovDatahandler.setThreadStatus(true);
        rovGuicontroller=new RovGUIController();
        rovGuicontroller.setDatahandler(rovDatahandler);
        rovGuicontroller.setStart(true);

        // create the object  of receiving data to GUI
        rovReceiveDataObserver = new RovReceiveDataObservable();
        // create the UDP thread, puts data into subject
        rovUdpReceiver = new RovUDPreceiver(rovReceiveDataObserver,
ROVRECEIVEPORT);
        rovUdpReceiver.start();
        rovReceiveDataObserver.addObserver(this);

        //creates a map object
        maps = new maps();
        /// UDP VideoStream
        vidUdpPlatform = new udpVideoReciever(PLATVIDEOPORT);
        vidUdpPlatform.start();
        vidUdpROV = new udpVideoReciever(ROVVIDEOPORT);
        vidUdpROV.start();
        DateFormat dateFormat = new SimpleDateFormat("yyyy_MM_dd_HH_mm_ss");

        //making automode object...//
        autMode=new autoMode();
```

```
        //vidUdp.start();
        // TAB PANE
        TabPane tabPane = new TabPane();
        tabPane.setTabClosingPolicy(TabPane.TabClosingPolicy.UNAVAILABLE);
        Tab overviewTab = new Tab("OVERVIEW");
        Tab platformTab = new Tab("PLATFORM");
        Tab rovTab = new Tab("ROV");
        Tab settingsTab = new Tab("SETTINGS");

        ///////// Overview



        // Buttons for GPS data
        Button closebtn = new Button("Close");
        closebtn.setOnAction(e -> closeProgram());
        Button getLat = new Button("Get GPS position");
        getLat.setOnAction(e -> System.out.println("Latitude: " +
maps.getNextPosLat() + " Langditude: " + maps.getNextPosLng()));


        // Labels
        Label sensorData = new Label("SENSOR DATA");
        Label rov = new Label("ROV");
        Label platform = new Label("PLATFORM");
        Label depthRov = new Label("Depth ROV: ");
        Label rovTemp = new Label("Temp in ROV:");
        Label headingRov = new Label("Heading ROV:");
        Label headingPlat = new Label("Heading Platform:");
        Label currenSpeedPlat = new Label("Speed Platform:");
        Label pitchPlat = new Label("Pitch Platform:");
        Label rollPlat = new Label("ROLL Platform:");
        Label oxygenLabel = new Label("Oxygen in water:");
        Label waterTempLabel = new Label("Temp in water:");
        //text fields in overview tab
         headingPlatTextField = new TextField(Float.toString(heading));
         curretnSpeenTextField = new
TextField(Float.toString(currentSpeed));
         pitchTextField = new TextField(Float.toString(pitch));
         rollTextField = new TextField(Float.toString(roll));
         deptRovTextField = new TextField(Float.toString(0));
         tempRovTextField = new TextField(Float.toString(0));
         headingRovTextField = new TextField(Float.toString(0));
```

```
        tempInWaterTextField = new TextField(Float.toString(0));
        oxygenInWaterTextField = new TextField(Float.toString(0));




       // Panes in overview tab
        GridPane gridpaneOverview = new GridPane();
        GridPane gridpaneValues = new GridPane();
        BorderPane mapPane = new BorderPane();
        mapPane.setCenter(maps.getMapView());
        mapPane.setPrefSize(640, 480);

        //Canvas overviewCanvas = new Canvas(SCENE_W, SCENE_H);
        // Overall constraints
        gridpaneOverview.getColumnConstraints().add(new
ColumnConstraints(300));
        gridpaneOverview.getColumnConstraints().add(new
ColumnConstraints(300));
        gridpaneOverview.getColumnConstraints().add(new
ColumnConstraints(300));
        gridpaneOverview.getColumnConstraints().add(new
ColumnConstraints(300));
        gridpaneOverview.getRowConstraints().add(new RowConstraints(50));

        // Constraints Buttons
        GridPane.setConstraints(closebtn, 0, 4);
        GridPane.setValignment(closebtn, VPos.BOTTOM);
        GridPane.setHalignment(closebtn, HPos.LEFT);
        GridPane.setConstraints(getLat, 2, 0, 1, 1, HPos.LEFT,
VPos.CENTER);

        // Constraints Lables
        GridPane.setConstraints(sensorData, 0, 0);
        GridPane.setHalignment(sensorData, HPos.CENTER);
        GridPane.setConstraints(depthMeter, 3, 4);
        GridPane.setConstraints(temp, 2, 4);

        // Panes inside GridPane
        //GridPane.setConstraints(overviewCanvas, 2, 1);
        //GridPane.setHalignment(overviewCanvas, HPos.RIGHT);
       GridPane.setConstraints(mapPane, 2, 1, 2, 1, HPos.RIGHT,
VPos.CENTER);
       GridPane.setConstraints(gridpaneValues, 0, 1, 2, 2, HPos.CENTER,
```

```
VPos.CENTER);
	gridpaneValues.getColumnConstraints().add(new
ColumnConstraints(300));
	gridpaneValues.getColumnConstraints().add(new
ColumnConstraints(300));
	gridpaneValues.getRowConstraints().add(new RowConstraints(50));
	gridpaneValues.getRowConstraints().add(new RowConstraints(50));
	gridpaneValues.getRowConstraints().add(new RowConstraints(50));
	gridpaneValues.getRowConstraints().add(new RowConstraints(50));
	gridpaneValues.getRowConstraints().add(new RowConstraints(50));
	gridpaneValues.getRowConstraints().add(new RowConstraints(50));
	gridpaneValues.getRowConstraints().add(new RowConstraints(50));

	GridPane.setConstraints(depthRov, 0, 0);
	GridPane.setConstraints(rovTemp, 0, 1);
	GridPane.setConstraints(headingRov, 0, 2);
	GridPane.setConstraints(headingPlat, 0, 3);
	GridPane.setConstraints(currenSpeedPlat, 0, 4);
	GridPane.setConstraints(pitchPlat, 0, 5);
	GridPane.setConstraints(rollPlat, 0, 6);
	GridPane.setConstraints(oxygenLabel, 0, 7);
	GridPane.setConstraints(waterTempLabel, 0, 8);

	GridPane.setConstraints(deptRovTextField, 1, 0);
	GridPane.setConstraints(tempRovTextField, 1, 1);
	GridPane.setConstraints(headingRovTextField, 1, 2);
	GridPane.setConstraints(headingPlatTextField, 1, 3);
	GridPane.setConstraints(curretnSpeenTextField, 1, 4);
	GridPane.setConstraints(pitchTextField, 1, 5);
	GridPane.setConstraints(rollTextField, 1, 6);
	GridPane.setConstraints(oxygenInWaterTextField, 1, 7);
	GridPane.setConstraints(tempInWaterTextField, 1, 8);

	 // Videostream


	 // Add to the differet panes
	 gridpaneValues.getChildren().addAll(depthRov, rovTemp, headingRov,
		headingPlat, currenSpeedPlat, pitchPlat,
rollPlat,oxygenLabel,waterTempLabel,deptRovTextField,tempRovTextField,headi
ngRovTextField, headingPlatTextField,
		curretnSpeenTextField, pitchTextField,
rollTextField,oxygenInWaterTextField,tempInWaterTextField);
	 //mapPane.getChildren().addAll(maps.getMapView());
```

```
        //valueBox.getChildren().addAll(headingPlatValue);
        //dataBox.getChildren().addAll(rov, depthRov, rovTemp, headingRov,
platform,
                //headingPlat, currenSpeedPlat, yawPlat, rollPlat);
        gridpaneOverview.getChildren().addAll(closebtn, sensorData,
gridpaneValues, depthMeter,
        mapPane, temp, getLat);

        ////////////// Platform TAB

         recCirclePlatform = new Circle(10, Color.TRANSPARENT);
         dpCirclePlatform = new Circle(10, Color.RED);
         autoCirclePlatform = new Circle(10, Color.RED);
         manualCirlcePlatform = new Circle(10, Color.RED);

       Color redDotPlatform = Color.RED;
       Color orangeDotPlatform = Color.ORANGE;
       Color notRecPlatform = Color.TRANSPARENT;




        // Buttons
        Button fwdButtonPlatform = new Button("FWD");
        fwdButtonPlatform.setMinSize(100, 75);
        fwdButtonPlatform.setOnTouchPressed(e -> {
                                        sendObserver.setFwdMotion(true);
                                        sendObserver.notifyObs();});
        fwdButtonPlatform.setOnTouchReleased(e -> {
                                        sendObserver.setFwdMotion(false)
;
                                        sendObserver.notifyObs();});

        Button bckButtonPlatform = new Button("AFT");
        bckButtonPlatform.setMinSize(100, 75);
        bckButtonPlatform.setOnTouchPressed(e -> {
                                        sendObserver.setBckMotion(true);
                                        sendObserver.notifyObs();});
        bckButtonPlatform.setOnTouchReleased(e -> {
                                        sendObserver.setBckMotion(false)
;
                                        sendObserver.notifyObs();});
        Button lftButtonPlatform = new Button("LEFT");
        lftButtonPlatform.setMinSize(125, 75);
```

```
        lftButtonPlatform.setOnTouchPressed(e -> {
                                            sendObserver.setLeftMotion(true)
;
                                            sendObserver.notifyObs();});
        lftButtonPlatform.setOnTouchReleased(e -> {
                                            sendObserver.setLeftMotion(false
);
                                            sendObserver.notifyObs();});
        Button rgtButtonPlatform = new Button("RIGHT");
        rgtButtonPlatform.setMinSize(125, 75);
        rgtButtonPlatform.setOnTouchPressed(e -> {
                                            sendObserver.setRightMotion(true
);
                                            sendObserver.notifyObs();});
        rgtButtonPlatform.setOnTouchReleased(e -> {
                                            sendObserver.setRightMotion(fals
e);
                                            sendObserver.notifyObs();});

        Button clwButtonPlatform = new Button("CLW");
        clwButtonPlatform.setMinSize(100, 50);
        clwButtonPlatform.setOnTouchPressed(e -> {
                                            sendObserver.setClockWMotion(tru
e);
                                            sendObserver.notifyObs();});
        clwButtonPlatform.setOnTouchReleased(e -> {
                                            sendObserver.setClockWMotion(fal
se);
                                            sendObserver.notifyObs();});

        Button cClwButtonPlatform = new Button("CCLW");
        cClwButtonPlatform.setMinSize(100, 50);
        cClwButtonPlatform.setOnTouchPressed(e -> {
                                            sendObserver.setCounterClockWMot
ion(true);
                                            sendObserver.notifyObs();});
        cClwButtonPlatform.setOnTouchReleased(e -> {
                                            sendObserver.setCounterClockWMot
ion(false);
                                            sendObserver.notifyObs();});
        Button startPump = new Button("Empety tanks");
        startPump.setOnTouchPressed(e -> {
                                           sendObserver.setStartPump(true);
                                           sendObserver.notifyObs(); });
```

```
        startPump.setOnTouchReleased(e -> {
                                            sendObserver.setStartPump(false)
;
                                            sendObserver.notifyObs();});
            Button flute = new Button("Flute");
        flute.setOnTouchPressed(e -> {
                                           sendObserver.setEnableFlute(true)
;
                                           sendObserver.notifyObs(); });
        flute.setOnTouchReleased(e -> {
                                            sendObserver.setEnableFlute(fals
e);
                                            sendObserver.notifyObs();});

        Button dpModeButton = new Button("DP-Mode");
        dpModeButton.setMaxWidth(160);
        dpModeButton.setOnTouchPressed(e -> {
            autoModeLatSetpoint=latitude;
            autoModeLongSetpoint=longitude;
            sendObserver.setDpModeEnable(true);
            sendObserver.setEnableAuto(false);
            sendObserver.setEnableManual(false);
            sendObserver.notifyObs();
            dpCirclePlatform.setFill(orangeDotPlatform);
            autoCirclePlatform.setFill(redDotPlatform);
            manualCirlcePlatform.setFill(redDotPlatform);
                });

        Button autopilotModeButton = new Button("Autopilot");
        autopilotModeButton.setMaxWidth(160);
        autopilotModeButton.setOnTouchPressed(e -> {
            autoModeLatSetpoint=latitude;
            autoModeLongSetpoint=longitude;
            sendObserver.setDpModeEnable(false);
            sendObserver.setEnableAuto(true);
            sendObserver.setEnableManual(false);
            sendObserver.notifyObs();
            dpCirclePlatform.setFill(redDotPlatform);
            autoCirclePlatform.setFill(orangeDotPlatform);
            manualCirlcePlatform.setFill(redDotPlatform);
                });

        Button manualModeButton = new Button("Manual Mode");
        manualModeButton.setMaxWidth(160);
```

```
        manualModeButton.setOnTouchPressed(e -> {
            sendObserver.setDpModeEnable(false);
            sendObserver.setEnableAuto(false);
            sendObserver.setEnableManual(true);
            sendObserver.notifyObs();
            dpCirclePlatform.setFill(redDotPlatform);
            autoCirclePlatform.setFill(redDotPlatform);
            manualCirlcePlatform.setFill(orangeDotPlatform);
                });


       Button startRecordBtnPlatform = new Button("Record");
       startRecordBtnPlatform.setOnAction(e -> {
           writerPlatform = new
VideoWriter("C:\\Users\\Platform\\Desktop\\ROVVideo\\Platform\\"
                   +dateFormat.format(new Date())+".avi",
VideoWriter.fourcc
                    ('M', 'J', 'P', 'G'), 15,  new Size(320, 240));
           startRecPlatform = true;
           recCirclePlatform.setFill(redDotPlatform);
       });
       startRecordBtnPlatform.setMaxSize(150, 20);

       Button stopRecordBtnPlatform = new Button("Stop");
       stopRecordBtnPlatform.setMaxSize(150, 20);
       stopRecordBtnPlatform.setOnMousePressed(e -> {
           stoprecPlatform = true;
           startRecordBtnPlatform.disarm();
           startRecPlatform = false;
           recCirclePlatform.setFill(notRecPlatform);
       });
       stopRecordBtnPlatform.setOnMouseReleased(e -> stoprecPlatform =
false);

        // Slider
        Slider speedPlatform = new Slider(0, 100, 50);

        speedPlatform.setShowTickMarks(true);
        speedPlatform.setShowTickLabels(true);
        speedPlatform.setMajorTickUnit(25f);
        speedPlatform.setBlockIncrement(10f);
        Label platformLabel = new Label("Platform Speed");
```

```
        speedPlatform.valueProperty().addListener(new
ChangeListener<Number>(){
        @Override
        public void changed(ObservableValue<? extends Number> observable,
                Number oldValue, Number newValue){
            System.out.println("New Value: " + newValue);

            sendObserver.setThrusterSpeed(newValue.intValue());
            sendObserver.notifyObs();
        }
        });


        // Panes
        GridPane platformLayout = new GridPane();
        GridPane platformGrid = new GridPane();
        Canvas platformCanvas = new Canvas(SCENE_W, SCENE_H);


        // Constraints
        platformLayout.getColumnConstraints().add(new
ColumnConstraints(300));
        platformLayout.getColumnConstraints().add(new
ColumnConstraints(250));
        platformLayout.getColumnConstraints().add(new
ColumnConstraints(300));
        platformLayout.getColumnConstraints().add(new
ColumnConstraints(350));
        platformLayout.getRowConstraints().add(new RowConstraints(50));
        platformLayout.getRowConstraints().add(new
RowConstraints(SCENE_H));
        platformLayout.getRowConstraints().add(new RowConstraints(150));


        GridPane.setConstraints(platformCanvas, 0, 1);
        GridPane.setHalignment(platformCanvas, HPos.LEFT);
        GridPane.setConstraints(fwdButtonPlatform, 3, 2, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(bckButtonPlatform, 3, 2, 1, 1, HPos.CENTER,
VPos.BOTTOM);
        GridPane.setConstraints(lftButtonPlatform, 3, 2, 1, 1, HPos.LEFT,
VPos.BOTTOM);
        GridPane.setConstraints(rgtButtonPlatform, 3, 2, 1, 1, HPos.RIGHT,
VPos.BOTTOM);
```

```
        GridPane.setConstraints(clwButtonPlatform, 3, 2, 1, 1, HPos.RIGHT,
VPos.TOP);
        GridPane.setConstraints(speedPlatform, 2, 2, 1, 1, HPos.LEFT,
VPos.BOTTOM);
        GridPane.setConstraints(cClwButtonPlatform, 3, 2, 1, 1, HPos.LEFT,
VPos.TOP);
        GridPane.setConstraints(startPump, 0, 2, 1, 1, HPos.CENTER,
VPos.BOTTOM);
        GridPane.setConstraints(flute, 1, 2, 1, 1, HPos.CENTER,
VPos.BOTTOM);
        GridPane.setConstraints(startRecordBtnPlatform, 0, 0, 1, 1,
HPos.LEFT, VPos.CENTER);
        GridPane.setConstraints(stopRecordBtnPlatform, 0, 0, 1, 1,
HPos.RIGHT, VPos.CENTER);
        GridPane.setConstraints(recCirclePlatform, 0, 1, 1, 1, HPos.LEFT,
VPos.TOP);
        GridPane.setConstraints(platformLabel, 2, 2, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(platformGrid, 2, 1, 2, 1, HPos.LEFT,
VPos.CENTER);



        platformGrid.getColumnConstraints().add(new
ColumnConstraints(300));
        platformGrid.getColumnConstraints().add(new
ColumnConstraints(350));
        platformGrid.getRowConstraints().add(new RowConstraints(50));
        platformGrid.getRowConstraints().add(new RowConstraints(50));
        platformGrid.getRowConstraints().add(new RowConstraints(50));
        platformGrid.getRowConstraints().add(new RowConstraints(50));
        platformGrid.getRowConstraints().add(new RowConstraints(50));



        GridPane.setConstraints(dpModeButton, 1, 0, 1, 1, HPos.CENTER,
VPos.CENTER);
        GridPane.setConstraints(autopilotModeButton, 1, 1, 1, 1,
HPos.CENTER, VPos.CENTER);
        GridPane.setConstraints(manualModeButton, 1, 2, 1, 1, HPos.CENTER,
VPos.CENTER);
        GridPane.setConstraints(dpCirclePlatform, 1, 0, 1, 1, HPos.RIGHT,
VPos.CENTER);
        GridPane.setConstraints(autoCirclePlatform, 1, 1, 1, 1, HPos.RIGHT,
VPos.CENTER);
        GridPane.setConstraints(manualCirlcePlatform, 1, 2, 1, 1,
```

```
HPos.RIGHT, VPos.CENTER);


        g2dPlatform = platformCanvas.getGraphicsContext2D();
        timerPlatform = new AnimationTimer() {
        Mat matPlatform = new Mat();
        Image imagePlatform;
            @Override
            public void handle(long now) {
               Mat vidPlatform = vidUdpPlatform.getImage();
                if (vidPlatform == null){
                       imagePlatform = new
Image(getClass().getResourceAsStream("disconnectedScreen2.png"));

                }
                else if(vidPlatform != null){
                    imagePlatform = mat2Image(vidPlatform);

                }

                g2dPlatform.drawImage(imagePlatform, 0, 0);

                if (startRecPlatform && !stoprecPlatform) {
                    try {
                         writerPlatform.write(vidPlatform);
                    } catch (Exception e) {
                        writerPlatform.release();
                        System.out.println("stoppingThe recording
Platform");
                        stoprecPlatform = true;
                        startRecordBtnPlatform.disarm();
                        startRecPlatform = false;
                        recCirclePlatform.setFill(notRecPlatform);
                    }
                }
                if (stoprecPlatform) {
                    writerPlatform.release();
                }
            }
        };
        timerPlatform.start();

        // Add children
        platformGrid.getChildren().addAll(dpCirclePlatform, dpModeButton,
```

```
manualCirlcePlatform,
                manualModeButton, autoCirclePlatform, autopilotModeButton);

        platformLayout.getChildren().addAll(speedPlatform,
fwdButtonPlatform, bckButtonPlatform,
                lftButtonPlatform, rgtButtonPlatform, platformCanvas,
cClwButtonPlatform, clwButtonPlatform,
                startPump,flute, startRecordBtnPlatform,
stopRecordBtnPlatform, recCirclePlatform,
                platformLabel, platformGrid);

        ////////////// ROV TAB
        // Buttons
        Button fwdButtonROV = new Button("FWD");
        fwdButtonROV.setMinSize(100, 75);
        fwdButtonROV.setOnTouchPressed(e -> {
                                        rovGuicontroller.setFwd(true);})
;
        fwdButtonROV.setOnTouchReleased(e -> {
                                        rovGuicontroller.setFwd(false);}
);

        Button bckButtonROV = new Button("AFT");
        bckButtonROV.setMinSize(100, 75);
        bckButtonROV.setOnTouchPressed(e -> {
                                        rovGuicontroller.setRev(true);
                                        System.out.println("test av ROV
AFT knapp");});
        bckButtonROV.setOnTouchReleased(e -> {
                                        rovGuicontroller.setRev(false);}
);
        Button lftButtonROV = new Button("LEFT");
        lftButtonROV.setMinSize(125, 75);
        lftButtonROV.setOnTouchPressed(e -> {
                                        rovGuicontroller.setLeft(true);}
);
        lftButtonROV.setOnTouchReleased(e -> {
                                        rovGuicontroller.setLeft(false);
});
        Button rgtButtonROV = new Button("RIGHT");
        rgtButtonROV.setMinSize(125, 75);
         rgtButtonROV.setOnTouchPressed(e -> {
                                        rovGuicontroller.setRight(true);
});
```

```
         rgtButtonROV.setOnTouchReleased(e -> {
                                          rovGuicontroller.setRight(false)
;});
        Button clwButtonROV = new Button("CLW");
        clwButtonROV.setMinSize(100, 50);
        clwButtonROV.setOnTouchPressed(e -> {
                                          rovGuicontroller.setSlideLeft(tr
ue);});
        clwButtonROV.setOnTouchReleased(e -> {
                                          rovGuicontroller.setSlideLeft(fa
lse);});

        Button cClwButtonROV = new Button("CCLW");
        cClwButtonROV.setMinSize(100, 50);
        cClwButtonROV.setOnTouchPressed(e -> {
                                          rovGuicontroller.setSlideRight(t
rue);});
        cClwButtonROV.setOnTouchReleased(e -> {
                                          rovGuicontroller.setSlideRight(f
alse);});
        Button upButtonROV = new Button("UP");
        upButtonROV.setMinSize(125, 75);
         upButtonROV.setOnTouchPressed(e -> {
                                           sendObserver.setWinchUp(true);
                                           sendObserver.notifyObs();});
         upButtonROV.setOnTouchReleased(e -> {
                                           sendObserver.setWinchUp(false);
                                           sendObserver.notifyObs();});
        Button downButtonROV = new Button("DOWN");
        downButtonROV.setMinSize(125, 75);
        downButtonROV.setOnTouchPressed(e -> {
                                           sendObserver.setWinchDown(true)
;
                                           sendObserver.notifyObs();});
        downButtonROV.setOnTouchReleased(e -> {
                                           sendObserver.setWinchDown(false
);
                                           sendObserver.notifyObs();});




       Circle recCircleROV = new Circle(10, Color.TRANSPARENT);
```

```
        lockCircleROV = new Circle(10, Color.ORANGE);
        upperCircleROV = new Circle(10, Color.ORANGE);
        Color redDotROV = Color.RED;
        Color clearDotROV = Color.TRANSPARENT;
        Color greenDotROV = Color.GREEN;



        Button startRecordBtnROV = new Button("Record");
        startRecordBtnROV.setOnAction(e -> {
            writerROV = new
VideoWriter("C:\\Users\\Platform\\Desktop\\ROVVideo\\ROV\\"
                    +dateFormat.format(new Date())+".avi",
VideoWriter.fourcc
                     ('M', 'J', 'P', 'G'), 15,  new Size(320, 240));
            startRecROV = true;
            recCircleROV.setFill(redDotROV);
        });
        startRecordBtnROV.setMaxSize(125, 20);

        Button stopRecordBtnROV = new Button("Stop");
        stopRecordBtnROV.setMaxSize(125, 20);
        stopRecordBtnROV.setOnMousePressed(e -> {
            stoprecROV = true;
            startRecordBtnROV.disarm();
            startRecROV = false;
            recCircleROV.setFill(clearDotROV);
        });
        stopRecordBtnROV.setOnMouseReleased(e -> stoprecROV = false);

        // Labels
        Label rovLockLabel = new Label("ROV is LOCKED:");
        Label rovUpperPosLabel = new Label("Upper position:");

          // Slider
         Slider speedROV = new Slider(0, 100, 50);
         speedROV.setShowTickMarks(true);
         speedROV.setShowTickLabels(true);
         speedROV.setMajorTickUnit(25f);
         speedROV.setBlockIncrement(10f);


         //Speed for the ROV slider
         speedROV.valueProperty().addListener(new ChangeListener<Number>(){
```

```
        @Override
        public void changed(ObservableValue<? extends Number> observable,
                Number oldValue, Number newValue){
            System.out.println("New Value: " + newValue);

            rovGuicontroller.setThrusterValue(newValue.intValue());

        }
        });

         // Slider
        Slider lightRovSlider = new Slider(0, 100, 50);
        lightRovSlider.setShowTickMarks(true);
        lightRovSlider.setShowTickLabels(true);
        lightRovSlider.setMajorTickUnit(25f);
        lightRovSlider.setBlockIncrement(10f);


        //Speed for the ROV slider
        lightRovSlider.valueProperty().addListener(new
ChangeListener<Number>(){
        @Override
        public void changed(ObservableValue<? extends Number> observable,
                Number oldValue, Number newValue){
            System.out.println("New Value: " + newValue);

            rovGuicontroller.setLightValue(newValue.intValue());

        }
        });







        Slider speedWinch = new Slider(0, 100, 50);
        speedWinch.setShowTickMarks(true);
        speedWinch.setShowTickLabels(true);
        speedWinch.setMajorTickUnit(25f);
        speedWinch.setBlockIncrement(10f);

           speedWinch.valueProperty().addListener(new
```

```
ChangeListener<Number>(){
        @Override
        public void changed(ObservableValue<? extends Number> observable,
                Number oldValue, Number newValue){
            System.out.println("New Value: " + newValue);

            sendObserver.setWinchSpeed(newValue.intValue());
            sendObserver.notifyObs();
        }
        });

        Label winchLabel = new Label("Winch Speed");
        Label rovLabel = new Label("ROV Speed");
        Label rovLightLabel = new Label("Light Power");
        // Panes
        GridPane ROVLayout = new GridPane();
        GridPane ROVValuePane = new GridPane();
        Canvas ROVCanvas = new Canvas(SCENE_W, SCENE_H);

        // Constraints
        ROVLayout.getColumnConstraints().add(new ColumnConstraints(250));
        ROVLayout.getColumnConstraints().add(new ColumnConstraints(300));
        ROVLayout.getColumnConstraints().add(new ColumnConstraints(300));
        ROVLayout.getColumnConstraints().add(new ColumnConstraints(350));
        ROVLayout.getRowConstraints().add(new RowConstraints(50));
        ROVLayout.getRowConstraints().add(new RowConstraints(SCENE_H));
        ROVLayout.getRowConstraints().add(new RowConstraints(150));


        GridPane.setConstraints(ROVCanvas, 0, 1);
        GridPane.setHalignment(ROVCanvas, HPos.LEFT);
        GridPane.setConstraints(fwdButtonROV, 3, 2, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(bckButtonROV, 3, 2, 1, 1, HPos.CENTER,
VPos.BOTTOM);
        GridPane.setConstraints(lftButtonROV, 3, 2, 1, 1, HPos.LEFT,
VPos.BOTTOM);
        GridPane.setConstraints(rgtButtonROV, 3, 2, 1, 1, HPos.RIGHT,
VPos.BOTTOM);
        GridPane.setConstraints(clwButtonROV, 3, 2, 1, 1, HPos.RIGHT,
VPos.TOP);
        GridPane.setConstraints(speedROV, 2, 2, 1, 1, HPos.LEFT,
VPos.BOTTOM);
        GridPane.setConstraints(cClwButtonROV, 3, 2, 1, 1, HPos.LEFT,
```

```
VPos.TOP);
        GridPane.setConstraints(startRecordBtnROV, 0, 0, 1, 1, HPos.LEFT,
VPos.CENTER);
        GridPane.setConstraints(stopRecordBtnROV, 0, 0, 1, 1, HPos.RIGHT,
VPos.CENTER);
        GridPane.setConstraints(recCircleROV, 0, 1, 1, 1, HPos.LEFT,
VPos.TOP);
        GridPane.setConstraints(upButtonROV, 0, 2, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(downButtonROV, 0, 2, 1, 1, HPos.CENTER,
VPos.BOTTOM);
        GridPane.setConstraints(speedWinch, 1, 2, 1, 1, HPos.LEFT,
VPos.BOTTOM);
        GridPane.setConstraints(winchLabel, 1, 2, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(rovLabel, 2, 2, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(ROVValuePane, 3, 1, 1, 1);

        ROVValuePane.getColumnConstraints().add(new
ColumnConstraints(350));
        ROVValuePane.getRowConstraints().add(new RowConstraints(50));
        ROVValuePane.getRowConstraints().add(new RowConstraints(50));
        ROVValuePane.getRowConstraints().add(new RowConstraints(50));
        ROVValuePane.getRowConstraints().add(new RowConstraints(200));
        ROVValuePane.getRowConstraints().add(new RowConstraints(50));
        ROVValuePane.getRowConstraints().add(new RowConstraints(50));

        GridPane.setConstraints(lockCircleROV, 0, 0, 1, 1, HPos.RIGHT,
VPos.CENTER);
        GridPane.setConstraints(upperCircleROV, 0, 1, 1, 1, HPos.RIGHT,
VPos.CENTER);
         GridPane.setConstraints(lightRovSlider, 0, 5, 1, 1, HPos.CENTER,
VPos.BOTTOM);
        GridPane.setConstraints(rovLightLabel, 0, 4, 1, 1, HPos.CENTER,
VPos.TOP);
        GridPane.setConstraints(rovLockLabel, 0, 0, 1, 1, HPos.CENTER,
VPos.CENTER);
        GridPane.setConstraints(rovUpperPosLabel, 0, 1, 1, 1, HPos.CENTER,
VPos.CENTER);
```

```
        g2dROV = ROVCanvas.getGraphicsContext2D();
        timerROV = new AnimationTimer() {
        Mat matROV = new Mat();
        Image imageROV;
            @Override
            public void handle(long now) {
               Mat vidROV = vidUdpROV.getImage();
                if (vidROV == null){
                       imageROV = new
Image(getClass().getResourceAsStream("disconnectedScreen2.png"));
                }
                else if(vidROV != null){
                    imageROV = mat2Image(vidROV);
                }
                g2dROV.drawImage(imageROV, 0, 0);
                if (startRecROV && !stoprecROV) {
                    try {
                         writerROV.write(vidROV);
                    } catch (Exception e) {
                        writerROV.release();
                        System.out.println("stoppingThe recording");
                        stoprecROV = true;
                        startRecordBtnROV.disarm();
                        startRecROV = false;
                        recCircleROV.setFill(clearDotROV);
                    }

                }
                if (stoprecROV) {
                    writerROV.release();
                }
            }
        };
        timerROV.start();

        // Add children
        ROVValuePane.getChildren().addAll(rovLockLabel,rovLightLabel,lightR
ovSlider, rovUpperPosLabel,
                upperCircleROV, lockCircleROV);

        ROVLayout.getChildren().addAll(fwdButtonROV, bckButtonROV,
lftButtonROV,
                rgtButtonROV, clwButtonROV, cClwButtonROV, speedROV,
ROVCanvas,
```

```
                stopRecordBtnROV, startRecordBtnROV,recCircleROV,
upButtonROV,
                downButtonROV, speedWinch, winchLabel, rovLabel,
ROVValuePane);
        ////////////// Settings TAB

        //Buttons
        Button saveFileButton = new Button("Select");
        Button setRovIpButton = new Button("Set IP");
        Button setPlatformIpButton = new Button("Set IP");
        // TextFields
        TextField setSaveLocation = new TextField();
        TextField setRovIp = new TextField();
        TextField setPlatformIp = new TextField();
        GridPane settingsLayout = new GridPane();
        // Constraints
        settingsLayout.getColumnConstraints().add(new
ColumnConstraints(300));
        settingsLayout.getColumnConstraints().add(new
ColumnConstraints(150));

        settingsLayout.getRowConstraints().add(new RowConstraints(75));
        settingsLayout.getRowConstraints().add(new RowConstraints(75));
        settingsLayout.getRowConstraints().add(new RowConstraints(75));


        GridPane.setConstraints(setSaveLocation, 0, 0);
        GridPane.setConstraints(setRovIp, 0, 1);
        GridPane.setConstraints(setPlatformIp, 0, 2);

        GridPane.setConstraints(saveFileButton, 1, 0, 1, 1,
HPos.CENTER,VPos.CENTER);
        GridPane.setConstraints(setRovIpButton, 1, 1, 1, 1,
HPos.CENTER,VPos.CENTER);
        GridPane.setConstraints(setPlatformIpButton, 1, 2, 1, 1,
HPos.CENTER,VPos.CENTER);

        // Padding
        settingsLayout.setPadding(new Insets(20, 20, 20, 20));



        settingsLayout.getChildren().addAll(setSaveLocation,saveFileButton,
                setRovIp, setRovIpButton, setPlatformIp,
```

```
setPlatformIpButton);


        gridpaneOverview.setGridLinesVisible(false);
        settingsLayout.setGridLinesVisible(false);
        platformLayout.setGridLinesVisible(false);
        ROVLayout.setGridLinesVisible(false);
        ROVValuePane.setGridLinesVisible(false);
        platformGrid.setGridLinesVisible(false);

        // Add panes to tab
        overviewTab.setContent(gridpaneOverview);
        settingsTab.setContent(settingsLayout);
        platformTab.setContent(platformLayout);
        rovTab.setContent(ROVLayout);
        tabPane.getTabs().addAll(overviewTab, platformTab, rovTab,
settingsTab);

        Scene scene = new Scene(tabPane);
        scene.getStylesheets().add
        (SeaFarm.class.getResource("seafarm.css").toExternalForm());



        window.setOnCloseRequest(e -> closeProgram());
        window.setTitle("Aquaculture Seafarm");
        window.setFullScreen(true);
        window.setScene(scene);
        window.show();

    }

    //change this to, check each sensor and set value, so more  than one
can be
    // green at the same time

    private void updateMOdeIndicators(boolean dp,boolean auto, boolean
manual){
        Color redDotPlatform = Color.RED;
        Color greenDotPlatform = Color.GREEN;


        if (manual) {
```

```
            this.dpCirclePlatform.setFill(redDotPlatform);
            this.autoCirclePlatform.setFill(redDotPlatform);
            this.manualCirlcePlatform.setFill(greenDotPlatform);
        }

          else if (auto) {

            this.dpCirclePlatform.setFill(redDotPlatform);
            this.autoCirclePlatform.setFill(greenDotPlatform);
            this.manualCirlcePlatform.setFill(redDotPlatform);
        }
            else if (dp) {

            this.dpCirclePlatform.setFill(greenDotPlatform);
            this.autoCirclePlatform.setFill(redDotPlatform);
            this.manualCirlcePlatform.setFill(redDotPlatform);
        }
            else  {

            this.dpCirclePlatform.setFill(redDotPlatform);
            this.autoCirclePlatform.setFill(redDotPlatform);
            this.manualCirlcePlatform.setFill(redDotPlatform);
        }


    }

        private void updateRovLockingIndicator(boolean upperPos,boolean
locked){
        Color redDotRov = Color.RED;
        Color greenDotRov = Color.GREEN;




        if (upperPos) {

            this.upperCircleROV.setFill(greenDotRov);

        }

          else  {
```

```
            this.upperCircleROV.setFill(redDotRov);

        }


        if (locked) {

            this.lockCircleROV.setFill(greenDotRov);

        }
            else  {

            this.lockCircleROV.setFill(redDotRov);

        }


    }



    private void initOpenCv() {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        //videoCapture = new VideoCapture();
        //videoCapture.open(0);

        //System.out.println("Camera open: " + videoCapture.isOpened());

    }

    public static Image mat2Image(Mat mat) {
        Mat resizeimage = new Mat();
        Size sz = new Size(640,480);
        Imgproc.resize( mat, resizeimage, sz );

        MatOfByte buffer = new MatOfByte();
        Imgcodecs.imencode(".jpg", resizeimage, buffer);
        //Image imToRe = new Image(new
ByteArrayInputStream(buffer.toArray()));

        return new Image(new ByteArrayInputStream(buffer.toArray()));
    }
```

```
    private Gauge makeGauge(String title, String unit){
              Gauge gauge = GaugeBuilder.create()
               .foregroundBaseColor(Color.WHITE)
               .title(title)
               .titleColor(Color.WHITE)
               .subTitle("Subtitle")
               .unit(unit)
               .unitColor(Color.WHITE)
               .valueColor(Color.WHITE)

               .build();

        return gauge;
    }

    private void setGaugeValue(double temperature,double dept){
        temp.setValue(temperature);
        depthMeter.setValue(dept);
    }


   private void setRovTextField(float headingVal,float t, float dpt,float
tW,float oxy){
        //text fields
        headingRovTextField.setText(Float.toString(headingVal));
        tempRovTextField.setText(Float.toString(t));
        deptRovTextField.setText(Float.toString(dpt));
        tempInWaterTextField.setText(Float.toString(tW));
        oxygenInWaterTextField.setText(Float.toString(oxy));
    }

    private void setPlatformTextField(float headingVal,float speed, float
ptc,float roll){
        //text fields
        headingPlatTextField.setText(Float.toString(headingVal));
        curretnSpeenTextField.setText(Float.toString(speed));
        pitchTextField.setText(Float.toString(ptc));
        rollTextField.setText(Float.toString(roll));
    }

    private void closeProgram(){
```

```
        //timerOverview.stop();
        timerPlatform.stop();
        timerROV.stop();
        //vidUdpOverview.stop();
        vidUdpPlatform.stop();
        vidUdpROV.stop();
        System.out.println("Printing positions");
        maps.printList();
        //videoCapture.release();

        window.close();
        System.out.println("Closed program");
        Platform.exit();
        System.exit(0);
    }

    @Override
    public void update(Observable o, Object arg) {
        if(o instanceof RecieveDataObserver){
            //System.out.println("updatemodbus in seafarm class");
            RecieveDataObserver recieve = (RecieveDataObserver) o;
            longitude = recieve.getCurrentLongitude();
            latitude = recieve.getCurrentLatitude();
            dpModeActivated = recieve.getDpMode();
            autoModeActivated = recieve.getAutoMode();
            manualModeActivated = recieve.getManualMode();

            //check if maps i initialized, will be false with no internet
            if (maps != null){
            //System.out.println("map is not 0");
                try {
                  maps.setLng(longitude);
                  maps.setLat(latitude);
                  maps.setRovPosition();



                //Function for calculating if the plaform should get a new
position.
                //while in DP or Autopilot
                if(dpModeActivated || autoModeActivated ){

                    if
((!dpModeActivated)&&((autMode.shouldGetNewPos(latitude, longitude,
```

```
                    autoModeLongSetpoint,
autoModeLatSetpoint))||((autoModeLatSetpoint==0)||(autoModeLongSetpoint==0)
)))
                        //if the platform is close enough to the desire
point,
                        //check if there is another point plotted on the
map
                  {     double mapSetpointLat=maps.getNextPosLat();
                        double mapSetpointLong=maps.getNextPosLng();
                        //if there is no marker plotted stay in current
position
                        if((mapSetpointLat==0)||(mapSetpointLong==0)){
                            autoModeLatSetpoint= latitude;
                            autoModeLongSetpoint=longitude;
                        }
                        else{
                            //Request the platform to hold a new position
                            autoModeLatSetpoint= mapSetpointLat;
                            autoModeLongSetpoint=mapSetpointLong;
                         }

                  }
                        sendObserver.setLatitude((float)autoModeLatSetpoint
);
                        sendObserver.setLongitude((float)autoModeLongSetpoi
nt);
                        //System.out.println(autoModeLongSetpoint
+"longSet");
                        sendObserver.notifyObs();
                }

                }
                catch (Exception e) {

                    //System.out.println(e+"in update maps");

                }



            updateMOdeIndicators(dpModeActivated, autoModeActivated,
manualModeActivated);
            currentSpeed = recieve.getCurrentSpeed();
            heading = recieve.getCurrentYaw();
```

```
            pitch = recieve.gettCurrentPitch();
            roll = recieve.getCurrentRoll();
            rovLocked = recieve.getRovIsLocked();
            rovUpperPos = recieve.getRovInupperPos();
            updateRovLockingIndicator(rovUpperPos, rovLocked);
            setPlatformTextField(heading,currentSpeed,pitch,roll);


            }
            }
                if(o instanceof RovReceiveDataObservable){
            float dpt=0;
            float tmpInrov=0;
            float tmpInSea=0;
            float headrov=0;
            float oxygen=0;



            RovReceiveDataObservable recieve = (RovReceiveDataObservable)
o;
            dpt=recieve.getDepth();
            tmpInrov=recieve.getTempInROV();
            tmpInSea=recieve.getTemperature();
            headrov=recieve.getHeading();
            oxygen=recieve.getOxygen();
            sendObserver.setDpt(dpt);
            sendObserver.setTmpInSea(tmpInSea);
            sendObserver.setTmpInrov(tmpInrov);
            sendObserver.setHeadrov(headrov);
            sendObserver.setOxygen(oxygen);
            sendObserver.notifyObs();

            //updating the textField for the ROV
            setRovTextField(headrov,tmpInrov,dpt,tmpInSea,oxygen);
            setGaugeValue((double)recieve.getTempInROV(),(double)recieve.ge
tDepth());



        }


    }
```

```
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```