ДОДАТОК А

Конфігурація проекту. Конфігурація інтерфейсу користувача. Текст програми

1. Конфігурація проекту:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.konstantinp</groupId>
  <artifactId>spectrMask</artifactId>
  <version>1.0-SNAPSHOT</version>

  <dependencies>
    <dependency>
      <groupId>com.zenjava</groupId>
      <artifactId>javafx-maven-plugin</artifactId>
      <version>8.8.3</version>
    </dependency>
    <dependency>
      <groupId>org.apache.poi</groupId>
      <artifactId>poi</artifactId>
      <version>4.1.1</version>
    </dependency>
    <dependency>
      <groupId>org.apache.poi</groupId>
      <artifactId>poi-ooxml</artifactId>
      <version>4.1.1</version>
    </dependency>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <version>1.18.10</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.apache.commons</groupId>
      <artifactId>commons-lang3</artifactId>
      <version>3.9</version>
    </dependency>
    <dependency>
      <groupId>log4j</groupId>
```

```
        <artifactId>log4j</artifactId>
        <version>1.2.17</version>
      </dependency>

    </dependencies>
    <build>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-compiler-plugin</artifactId>
          <configuration>
            <source>8</source>
            <target>8</target>
          </configuration>
        </plugin>
        <plugin>
          <groupId>com.zenjava</groupId>
          <artifactId>javafx-maven-plugin</artifactId>
          <version>8.8.3</version>
          <configuration>
            <mainClass>Main</mainClass>
          </configuration>
        </plugin>
        <plugin>
          <!-- Build an executable JAR -->
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-jar-plugin</artifactId>
          <version>3.1.0</version>
          <configuration>
            <archive>
              <manifest>
                <addClasspath>true</addClasspath>
                <classpathPrefix>lib/</classpathPrefix>
                <mainClass>Main</mainClass>
              </manifest>
            </archive>
          </configuration>
        </plugin>
      </plugins>
    </build>

</project>
```

2. Конфігурація інтерфейсу користувача:

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.chart.LineChart?>
<?import javafx.scene.chart.NumberAxis?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<GridPane maxHeight="750.0" maxWidth="1200.0" minHeight="500.0" minWidth="850.0"
prefHeight="500.0" prefWidth="850.0"
        xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1"
        fx:controller="controller.Controller">
   <children>
      <MenuBar GridPane.columnSpan="2">
        <menus>
          <Menu mnemonicParsing="false" text="Файл">
            <items>
              <MenuItem fx:id="openFileMenuItem" mnemonicParsing="false"
onAction="#openFileButton"
                        text="Открыть"/>
              <SeparatorMenuItem mnemonicParsing="false"/>
              <MenuItem mnemonicParsing="false" onAction="#saveFileButton" text="Сохранить"/>
            </items>
          </Menu>
          <Menu mnemonicParsing="false" text="График">
            <items>
              <MenuItem mnemonicParsing="false" onAction="#cleanChart" text="Очистить
график"/>
              <CheckMenuItem fx:id="compareCharts" mnemonicParsing="false"
onAction="#compareCharts"
                             text="Сравнивать графики"/>
            </items>
          </Menu>
        </menus>
      </MenuBar>
      <TreeView fx:id="devicesTreeView" onContextMenuRequested="#mouseClick"
onMouseClicked="#mouseClick"
              GridPane.rowIndex="1">
        <GridPane.margin>
          <Insets bottom="5.0" left="5.0" right="5.0" top="5.0"/>
        </GridPane.margin>
      </TreeView>
      <LineChart fx:id="spectrMaskChart" GridPane.columnIndex="1" GridPane.hgrow="ALWAYS"
GridPane.rowIndex="1"
```

```
            GridPane.vgrow="ALWAYS">
          <xAxis>
            <NumberAxis side="BOTTOM"/>
          </xAxis>
          <yAxis>
            <NumberAxis side="LEFT"/>
          </yAxis>
          <GridPane.margin>
            <Insets bottom="5.0" left="5.0" right="5.0" top="5.0"/>
          </GridPane.margin>
        </LineChart>
        <HBox prefHeight="100.0" prefWidth="200.0" GridPane.columnIndex="1"
GridPane.rowIndex="2">
          <children>
            <Label text="-30 db ">
              <HBox.margin>
                <Insets left="25.0"/>
              </HBox.margin>
            </Label>
            <Label fx:id="frequency30db">
              <HBox.margin>
                <Insets right="75.0"/>
              </HBox.margin>
            </Label>
            <Label text="-3 db "/>
            <Label fx:id="frequency3db"/>
          </children>
        </HBox>
        <VBox prefHeight="200.0" prefWidth="100.0" GridPane.rowIndex="2">
          <children>
            <HBox prefHeight="100.0" prefWidth="200.0">
              <children>
                <Label text="Елементів у базі: ">
                  <HBox.margin>
                    <Insets left="10.0"/>
                  </HBox.margin>
                </Label>
                <Label fx:id="countInBase"/>
              </children>
            </HBox>
            <HBox prefHeight="100.0" prefWidth="200.0">
              <children>
                <Label text="Розмір бази: ">
                  <HBox.margin>
                    <Insets left="10.0"/>
                  </HBox.margin>
                </Label>
                <Label fx:id="sizeOfBase"/>
```

```
            </children>
          </HBox>
        </children>
      </VBox>
    </children>
    <columnConstraints>
      <ColumnConstraints/>
      <ColumnConstraints hgrow="ALWAYS"/>
    </columnConstraints>
    <rowConstraints>
      <RowConstraints/>
      <RowConstraints/>
      <RowConstraints minHeight="10.0" prefHeight="30.0"/>
    </rowConstraints>
</GridPane>
```

3. Текст програми:

```
        import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;


public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
        Parent root = new
FXMLLoader().load(getClass().getResourceAsStream("/sample.fxml"));
        primaryStage.setTitle("SpectrMask");
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }


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

```
        package controller;

import dom.Device;
import excel.DocumentProcessor;
import javafx.embed.swing.SwingFXUtils;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.SnapshotParameters;
import javafx.scene.chart.LineChart;
import javafx.scene.control.CheckMenuItem;
import javafx.scene.control.Label;
import javafx.scene.control.TreeItem;
import javafx.scene.control.TreeView;
import javafx.scene.image.WritableImage;
import javafx.scene.input.MouseEvent;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import lineChart.ChartCreator;
import lombok.extern.log4j.Log4j;

import javax.imageio.ImageIO;
import javax.xml.ws.Action;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Log4j
public class Controller {
    @FXML
    private LineChart spectrMaskChart;
    @FXML
    private TreeView devicesTreeView;
    @FXML
    private CheckMenuItem compareCharts;
    @FXML
    private Label frequency30db;
    @FXML
    private Label frequency3db;
    @FXML
    private Label countInBase;
    @FXML
    private Label sizeOfBase;
    private ChartCreator chartCreator = new ChartCreator();
```

```
    public void openFileButton(ActionEvent actionEvent) {
        log.debug("Try to open file");
        FileChooser chooser = new FileChooser();
        chooser.setTitle("Open EXCEL-File");
        chooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("EXCEL-file",
"*.xlsx"));
        File file = chooser.showOpenDialog(new Stage());

        log.debug("Starts process file");
        DocumentProcessor documentProcessor = new DocumentProcessor(file);
        HashMap<String, List<Device>> deviceMap = documentProcessor.getDeviceMap();
        int sizeOfDatabase = 0;
        for (Map.Entry<String, List<Device>> device : deviceMap.entrySet()) {
            sizeOfDatabase += device.getValue().size();
        }
        countInBase.setText(String.valueOf(sizeOfDatabase));
        long fileSizeInBytes = file.length();
        long fileSizeInKB = fileSizeInBytes / 1024;
        sizeOfBase.setText(String.valueOf(fileSizeInKB) + " Kb");

        log.debug("Devices was successfully parsed");
        TreeItem<Device> stringTreeItem = new TreeItem<>();
        for (String deviceType : deviceMap.keySet()) {
            Device device1 = new Device();
            device1.setName(deviceType);
            TreeItem<Device> deviceNameTreeItem = new TreeItem<>(device1);
            for (Device device : deviceMap.get(deviceType)) {
                TreeItem<Device> stringTreeItem1 = new TreeItem<>(device);
                deviceNameTreeItem.getChildren().add(stringTreeItem1);
            }
            stringTreeItem.getChildren().add(deviceNameTreeItem);
        }
        devicesTreeView.setRoot(stringTreeItem);
        devicesTreeView.setShowRoot(false);
        devicesTreeView.getRoot().setExpanded(true);
    }

    @Action
    public void mouseClick(MouseEvent mouseEvent) {
        Object selectedItem = devicesTreeView.getSelectionModel().getSelectedItem();
        spectrMaskChart.setAxisSortingPolicy(LineChart.SortingPolicy.NONE);
        if (selectedItem != null && selectedItem instanceof TreeItem && ((TreeItem)
selectedItem).getValue() instanceof Device) {
            Device device = (Device) ((TreeItem) selectedItem).getValue();
```

```
            chartCreator.setSpectrMaskChart(spectrMaskChart);
            chartCreator.createChart(device);
            frequency30db.setText(device.getPrd30db() + " МГц");
            frequency3db.setText(device.getPrd3db() + " МГц");
        }
    }

    public void cleanChart(ActionEvent actionEvent) {
        chartCreator.setSpectrMaskChart(spectrMaskChart);
        chartCreator.cleanChart();
    }

    public void compareCharts(ActionEvent actionEvent) {
        chartCreator.setSpectrMaskChart(spectrMaskChart);
        chartCreator.setCompareCharts(compareCharts.isSelected());
    }

    public void saveFileButton(ActionEvent actionEvent) {
        FileChooser fileChooser = new FileChooser();
        FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("PNG files
(*.png)", "*.png");
        fileChooser.getExtensionFilters().add(extFilter);
        File file = fileChooser.showSaveDialog(new Stage());
        if (file != null) {
            WritableImage image = spectrMaskChart.snapshot(new SnapshotParameters(),
null);
            try {
                ImageIO.write(SwingFXUtils.fromFXImage(image, null), "png", file);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
package dom;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class Coordinates {
    private Double frequency;
```

```
    private Double strip;

    public Coordinates() {
    }

    public Coordinates(Double frequency, Double strip) {
        this.frequency = frequency;
        this.strip = strip;
    }
}
```

```
package dom;

import javafx.scene.chart.XYChart;

import java.util.Comparator;

public class CoordinatesComparator implements Comparator<XYChart.Data<Double,
Double>> {

    @Override
    public int compare(XYChart.Data<Double, Double> o1, XYChart.Data<Double, Double>
o2) {
        if (o1.getXValue().equals(o2.getXValue())) {
            if (o1.getXValue() <= 0) {
                return o1.getYValue().compareTo(o2.getYValue());
            } else {
                return o2.getYValue().compareTo(o1.getYValue());
            }
        } else {
            return o1.getXValue().compareTo(o2.getXValue());
        }
    }
}
```

```
package dom;

import lombok.AccessLevel;
import lombok.Getter;
import lombok.Setter;

import java.util.ArrayList;
import java.util.List;
```

```
@Setter(AccessLevel.PUBLIC)
@Getter
public class Device {
   private Long id;
   private Long count;
   private String name;
   private DeviceType deviceType;
   private Long minFrequency;
   private Long maxFrequency;
   private String radiationClass;
   private String numberOfSignalPositionsInSignalSpace;
   private String code;
   private String modulation;
   private Double prd3db;
   private Double prd30db;
   private String frequencyDeviation;
   private List<Coordinates> measurementsMask = new ArrayList<>();
   private String extraInfo;

   @Override
   public String toString() {
      return this.name;
   }
}
```

```
package dom;

public enum DeviceType {
   A, D;
}
```

```
package excel;

import dom.Device;
import lombok.Getter;
import lombok.extern.log4j.Log4j;
import org.apache.commons.lang3.StringUtils;
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.File;
import java.io.FileInputStream;
```

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;

@Log4j
public class DocumentProcessor {
   @Getter
   private HashMap<String, List<Device>> deviceMap = new HashMap<>();

   public DocumentProcessor(File file) {
      try {
         FileInputStream fileStream = new FileInputStream(file);
         Workbook workbook = new XSSFWorkbook(fileStream);
         Iterator<Sheet> sheetIterator = workbook.sheetIterator();
         while (sheetIterator.hasNext()) {
            processSheet(sheetIterator.next());
         }
         fileStream.close();
      } catch (IOException e) {
         log.warn(e.getMessage(), e);
      }
   }

   private void processSheet(Sheet sheet) {
      deviceMap.putIfAbsent(sheet.getSheetName(), new ArrayList<>());
      Iterator<Row> rowIterator = sheet.rowIterator();
      RowProcessor rowProcessor = null;
      while (rowIterator.hasNext()) {
         Row row = rowIterator.next();
         try {
            if (checkIfRowIsEmpty(row)) {
               if (row.getRowNum() <= 1) {
                  continue;
               } else {
                  break;
               }
            }
            if (rowProcessor == null) {
               rowProcessor = new RowProcessor(row);
               continue;
            }
            Device device = rowProcessor.processRow(row);
            if (!device.getMeasurementsMask().isEmpty()) {
```

```
                List<Device> devices = deviceMap.get(sheet.getSheetName());
                devices.add(device);
            }
        } catch (Exception e) {
            System.out.println("Exception " + e + " in sheet " + sheet.getSheetName());
        }
    }
  }

  private boolean checkIfRowIsEmpty(Row row) {
    if (row == null) {
      return true;
    }
    if (row.getLastCellNum() <= 0) {
      return true;
    }
    for (int cellNum = row.getFirstCellNum(); cellNum < row.getLastCellNum();
cellNum++) {
      Cell cell = row.getCell(cellNum);
      if (cell != null && cell.getCellType() != CellType.BLANK &&
StringUtils.isNotBlank(cell.toString())) {
        return false;
      }
    }
    return true;
  }
}
```

```
package excel;

import dom.Coordinates;
import dom.Device;
import dom.DeviceType;
import lombok.extern.log4j.Log4j;
import org.apache.commons.lang3.StringUtils;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellType;
import org.apache.poi.ss.usermodel.Row;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

@Log4j
```

```
public class RowProcessor {
   private int headId = -1;
   private int headCount = -1;
   private int headName = -1;
   private int headDeviceType = -1;
   private int headMinFrequency = -1;
   private int headMaxFrequency = -1;
   private int headRadiationClass = -1;
   private int headNumberOfSignalPositionsInSignalSpace = -1;
   private int headCode = -1;
   private int headModulation = -1;
   private int headPrd3db = -1;
   private int headPrd30db = -1;
   private int headFrequencyDeviation = -1;
   private List<Integer> headFrequency = new ArrayList<>();
   private List<Integer> headStrip = new ArrayList<>();

   public RowProcessor(Row row) throws IllegalArgumentException {
      Iterator<Cell> cellIterator = row.cellIterator();
      while (cellIterator.hasNext()) {
         Cell cell = cellIterator.next();
         if (isEquals("ID", cell)) {
            headId = cell.getColumnIndex();
         } else if (isEquals("кіл-сть", cell)) {
            headCount = cell.getColumnIndex();
         } else if (isEquals("назва/тип РЕЗ", cell)) {
            headName = cell.getColumnIndex();
         } else if (isEquals("A/D", cell)) {
            headDeviceType = cell.getColumnIndex();
         } else if (isEquals("Мін_част", cell)) {
            headMinFrequency = cell.getColumnIndex();
         } else if (isEquals("Макс_част", cell)) {
            headMaxFrequency = cell.getColumnIndex();
         } else if (isEquals("клас випромінювання", cell) || isEquals("Класвипром", cell))
{
            headRadiationClass = cell.getColumnIndex();
         } else if (isEquals("поз-сть", cell) || isEquals("Поз-ність", cell)) {
            headNumberOfSignalPositionsInSignalSpace = cell.getColumnIndex();
         } else if (isEquals("код", cell)) {
            headCode = cell.getColumnIndex();
         } else if (isEquals("модуляція", cell)) {
            headModulation = cell.getColumnIndex();
         } else if (isEquals("Прд(-3дБ)", cell)) {
            headPrd3db = cell.getColumnIndex();
         } else if (isEquals("Прд(-30дБ)", cell)) {
```

```
            headPrd30db = cell.getColumnIndex();
         } else if (isEquals("відхил частоти", cell)) {
            headFrequencyDeviation = cell.getColumnIndex();
         } else if (isEquals("на рівні (дБ)", cell)) {
            headStrip.add(cell.getColumnIndex());
         } else if (isEquals("смуга (МГц)", cell) || isEquals("смуга 5×CS (МГц)", cell)) {
            headFrequency.add(cell.getColumnIndex());
         } else if (isEquals("№ з/п", cell) || isEquals("№з/п", cell) || isEquals("№\nз/п.",
cell)) continue;
         else if (isEquals("Кол.РЕЗ (qnty) rec", cell)) continue;
         else if (CellType.BLANK.equals(cell.getCellType())) continue;
         else if (isEquals("EN 302.217-2-2014", cell)) continue;
         else {
            throw new IllegalArgumentException("Wrong value in cell " +
cell.getAddress() + " cellType " + cell.getCellType() + " cell value " + getValue(cell));
         }
      }
   }

   public Device processRow(Row row) {
      Device device = new Device();
      if (headId > 0) {
         device.setId(getLongValue(row.getCell(headId)));
      }
      if (headCount > 0) {
         device.setCount(getLongValue(row.getCell(headCount)));
      }
      if (headName > 0) {
         device.setName(getStringValue(row.getCell(headName)));
      }
      if (headDeviceType > 0) {
         String stringValue = getStringValue(row.getCell(headDeviceType));
         DeviceType deviceType = null;
         if (stringValue != null && !stringValue.isEmpty()) {
            deviceType = DeviceType.valueOf(stringValue);
         }
         device.setDeviceType(deviceType);
      }
      if (headMinFrequency > 0) {
         device.setMinFrequency(getLongValue(row.getCell(headMinFrequency)));
      }
      if (headMaxFrequency > 0) {
         device.setMaxFrequency(getLongValue(row.getCell(headMaxFrequency)));
      }
      if (headRadiationClass > 0) {
```

```
            device.setRadiationClass(getStringValue(row.getCell(headRadiationClass)));
        }
        if (headNumberOfSignalPositionsInSignalSpace > 0) {

device.setNumberOfSignalPositionsInSignalSpace(getStringValue(row.getCell(headNumb
erOfSignalPositionsInSignalSpace)));
        }
        if (headCode > 0) {
            device.setCode(getStringValue(row.getCell(headCode)));
        }
        if (headModulation > 0) {
            device.setModulation(getStringValue(row.getCell(headModulation)));
        }
        if (headPrd3db > 0) {
            device.setPrd3db(getDoubleValue(row.getCell(headPrd3db)));
        }
        if (headPrd30db > 0) {
            device.setPrd30db(getDoubleValue(row.getCell(headPrd30db)));
        }
        if (headFrequencyDeviation > 0) {

device.setFrequencyDeviation(getStringValue(row.getCell(headFrequencyDeviation)));
        }
        if (!headFrequency.isEmpty()) {
            List<Coordinates> coordinatesList = new ArrayList<>();
            for (int i : headFrequency) {
                Coordinates coordinates = new Coordinates();
                Double doubleValue = getDoubleValue(row.getCell(i));
                if (doubleValue != null) {
                    coordinates.setFrequency(doubleValue);
                    coordinatesList.add(coordinates);
                }
            }
            int j = 0;
            for (int i : headStrip) {
                Double doubleValue = getDoubleValue(row.getCell(i));
                if (doubleValue != null) {
                    coordinatesList.get(j).setStrip(doubleValue);
                    j++;
                }
            }
            device.setMeasurementsMask(coordinatesList);
        }
        return device;
    }
```

```
public Object getValue(Cell cell) {
   switch (cell.getCellType()) {
      case FORMULA:
         return cell.getCellFormula();
      case BLANK:
         return StringUtils.EMPTY;
      case BOOLEAN:
         return cell.getBooleanCellValue();
      case STRING:
         return cell.getStringCellValue();
      case NUMERIC:
         String cellVallue = String.valueOf(cell.getNumericCellValue());
         if (cellVallue.startsWith("+")) {
            cellVallue = cellVallue.replaceAll("\\+", StringUtils.EMPTY);
         }
         return Double.valueOf(cellVallue);
      default:
         return null;
   }
}

private String getStringValue(Cell cell) {
   return getValue(cell).toString();
}

private Long getLongValue(Cell cell) {
   Long result = null;
   Double doubleValue = getDoubleValue(cell);
   if (doubleValue != null) {
      result = doubleValue.longValue();
   }
   return result;
}

private Double getDoubleValue(Cell cell) {
   Double result = null;
   String s = getValue(cell).toString();
   if (!s.isEmpty()) {
      result = Double.valueOf(s);
   }
   return result;
}
```

```
    private boolean isEquals(String etalon, Cell cell) {
        boolean result = false;
        switch (cell.getCellType()) {
            case STRING:
                etalon = etalon.replaceAll("(\\s+|\\.|,|\\n|\\r|_|-|\\(|\\))", "");
                String cellValue = cell.getStringCellValue().replaceAll("(\\s+|\\.|,|\\n|\\r|_|-
|\\(|\\))", "");
                result = etalon.equalsIgnoreCase(cellValue);
                break;
            case NUMERIC:
                result = etalon.equalsIgnoreCase(String.valueOf(cell.getNumericCellValue()));
                break;
            case BLANK:
            case BOOLEAN:
            case _NONE:
            case ERROR:
            case FORMULA:
            default:
                result = false;
                break;
        }
        return result;
    }
}
```

```
package lineChart;

import dom.Coordinates;
import dom.CoordinatesComparator;
import dom.Device;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.chart.LineChart;
import javafx.scene.chart.XYChart;
import javafx.scene.control.Tooltip;
import javafx.util.Pair;
import lombok.Setter;

import java.util.ArrayList;
import java.util.List;

public class ChartCreator {
    @Setter
    private boolean compareCharts;
```

```
    @Setter
    private LineChart spectrMaskChart;

    public void cleanChart() {
        spectrMaskChart.getData().clear();
    }

    public void createChart(Device device) {
        if (!device.getMeasurementsMask().isEmpty()) {
            if (!compareCharts) {
                cleanChart();
            }
            XYChart.Series<Double, Double> series = new XYChart.Series<>();
            ObservableList<XYChart.Data<Double, Double>> datas =
FXCollections.observableArrayList();

            for (Coordinates coordinates : device.getMeasurementsMask()) {
                double y = coordinates.getStrip();
                double x = coordinates.getFrequency();
                datas.add(new XYChart.Data<>(x, y));
                datas.add(new XYChart.Data<>(-x, y));
            }
            datas.add(findPRD30Db(device));
            findPRD3Db(device);
            datas = datas.sorted(new CoordinatesComparator());
            series.setData(datas);
            StringBuilder chartTitle = new StringBuilder("Рисунок Х Х.Х - " +
device.getName());
            if (device.getMinFrequency() != null && device.getMaxFrequency() != null) {
                chartTitle.append(", f=").append(device.getMinFrequency()).append("-
").append(device.getMaxFrequency()).append("МГц");
            }
            if (device.getRadiationClass() != null && !device.getRadiationClass().isEmpty()) {
                chartTitle.append(", ").append(device.getRadiationClass());
            }
            if (device.getModulation() != null && !device.getModulation().isEmpty()) {
                chartTitle.append(", ").append(device.getModulation());
            }
            series.setName(chartTitle.toString());

            spectrMaskChart.getData().add(series);
            createTooltip();
        }
    }
```

```
    private XYChart.Data<Double, Double> findPRD30Db(Device device) {
        XYChart.Data<Double, Double> less30 = new XYChart.Data<>(0.0, -60.0);
        XYChart.Data<Double, Double> more30 = new XYChart.Data<>(0.0, 10.0);
        for (Coordinates coordinates : device.getMeasurementsMask()) {
            double y = coordinates.getStrip();
            double x = coordinates.getFrequency();
            if (y <= -30 && y > less30.getYValue()) {
                less30.setYValue(y);
                less30.setXValue(x);
            }
            if (y >= -30 && y < more30.getYValue()) {
                more30.setYValue(y);
                more30.setXValue(x);
            }
        }
        double x30db = (more30.getXValue() - less30.getXValue()) * (-30 -
less30.getYValue()) / (more30.getYValue() - less30.getYValue()) + less30.getXValue();
        device.setPrd30db(x30db);
        return new XYChart.Data<>(x30db, -30.0);
    }

    private XYChart.Data<Double, Double> findPRD3Db(Device device) {
        device.getMeasurementsMask().sort((o1, o2) -> {
            return o2.getStrip().compareTo(o1.getStrip());
        });
        Double lowestStrip =
device.getMeasurementsMask().get(device.getMeasurementsMask().size() -
1).getStrip();
        if (lowestStrip < 0) {
            lowestStrip *= -1;
        }
        Double highestStrip = device.getMeasurementsMask().get(0).getStrip();
        Coordinates beginCoordinates = new Coordinates();
        beginCoordinates.setFrequency(0.0);
        Double deltaStrip = lowestStrip + highestStrip;
        beginCoordinates.setStrip(deltaStrip);
        Double area = 0.0;
        List<Double> areaResults = new ArrayList<>();
        for (Coordinates coordinates : device.getMeasurementsMask()) {
            Coordinates coordinatesTemp = new Coordinates();
            coordinatesTemp.setFrequency(coordinates.getFrequency());
            coordinatesTemp.setStrip(coordinates.getStrip() + deltaStrip);
            area += calculateAreaByTwoDots(beginCoordinates, coordinatesTemp);
            areaResults.add(area);
            beginCoordinates = coordinatesTemp;
```

```
        }
        Double neediestArea = area * 0.995;
        Double deltaArea = area - neediestArea;
        Pair<Integer, Integer> indexesOfPointForSearch3Db = null;
        for (int i = 0; i < areaResults.size(); i++) {
            if (areaResults.get(i) > neediestArea) {
                indexesOfPointForSearch3Db = new Pair<>(i - 1, i);
                break;
            }
        }
        Coordinates coordinates =
device.getMeasurementsMask().get(indexesOfPointForSearch3Db.getKey());
        Coordinates firstLocalCoordinate = new Coordinates(coordinates.getFrequency(),
coordinates.getStrip() + deltaStrip);
        Coordinates coordinates2 =
device.getMeasurementsMask().get(indexesOfPointForSearch3Db.getValue());
        Coordinates secondLocalCoordinates = new
Coordinates(coordinates2.getFrequency(), coordinates2.getStrip() + deltaStrip);
        Pair<Double, Double> equation = createEquation(firstLocalCoordinate,
secondLocalCoordinates);
        Double localArea = takeIntegral(equation, firstLocalCoordinate.getFrequency(),
secondLocalCoordinates.getFrequency());
        neediestArea = localArea - deltaArea;
        XYChart.Data<Double, Double> prd3db = null;
        for (double d = secondLocalCoordinates.getFrequency(); d >
firstLocalCoordinate.getFrequency(); d -= 0.01) {
            Double integral = takeIntegral(equation, firstLocalCoordinate.getFrequency(), d);
            if (Math.abs(neediestArea - integral) < 0.1) {
                prd3db = new XYChart.Data<>(d, secondLocalCoordinates.getStrip());
                device.setPrd3db(d);
                break;
            }
        }
        return prd3db;
    }

    private Double calculateAreaByTwoDots(Coordinates firstCoordinate, Coordinates
secondCoordinates) {
        Pair<Double, Double> equation = createEquation(firstCoordinate,
secondCoordinates);
        return takeIntegral(equation, firstCoordinate.getFrequency(),
secondCoordinates.getFrequency());
    }

    private Double takeIntegral(Pair<Double, Double> equation, Double firstLimit, Double
```

```
secondLimit) {
      Double result = 0.0;
      if (equation != null) {
         result = (equation.getKey() * (Math.pow(secondLimit, 2) / 2) +
equation.getValue() * secondLimit) -
               (equation.getKey() * (Math.pow(firstLimit, 2) / 2) + equation.getValue() *
firstLimit);
      }
      return result;
   }

   private Pair<Double, Double> createEquation(Coordinates firstCoordinate,
Coordinates secondCoordinates) {
      Pair result = null;
      if (firstCoordinate.getFrequency().compareTo(secondCoordinates.getFrequency())
!= 0) {
         double firstPart = (secondCoordinates.getStrip() - firstCoordinate.getStrip()) /
(secondCoordinates.getFrequency() - firstCoordinate.getFrequency());
         double secondPart = (secondCoordinates.getFrequency() *
firstCoordinate.getStrip() - firstCoordinate.getFrequency() *
secondCoordinates.getStrip()) / (secondCoordinates.getFrequency() -
firstCoordinate.getFrequency());
         result = new Pair<Double, Double>(firstPart, secondPart);
      }
      return result;
   }

   private void createTooltip() {
      for (Object datum : spectrMaskChart.getData()) {
         for (XYChart.Data<Double, Double> d : ((XYChart.Series<Double, Double>)
datum).getData()) {
            Tooltip.install(d.getNode(), new Tooltip(
                  "Частота сигнала: " + d.getXValue().toString() + "\n" +
                        "Уровень : " + d.getYValue()));
         }
      }
   }
}
```

ДОДАТОК Б

Слайди презентації

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

АТЕСТАЦІЙНА РОБОТА МАГІСТРА

ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМ РАДІОМОНІТОРИНГУ

Виконав:
студент 2 курсу, групи ІКТм-18-1 Пішоха К.С.

Науковий керівник:
д.т.н. проф. Цопа О.І.

2018 р.

Вступ

Мета роботи: створення бази спектральних масок випромінювання РЕЗ основних радіотехнологій, розробка програмного забезпечення для їх побудування і інтеграція їх в аналізатори спектру для вдосконалення автоматизації процесу контролю (моніторингу) радіочастот в Україні.

Об'єкт досліджень: Засоби програмування для створення (побудови) спектральних масок випромінюючих радіоелектронних засобів основних цифрових і аналогових радіотехнологій, які охоплюються системою радіочастотного моніторингу України.

Вступ

Предмет дослідження :	спектральні маски випромінювання РЕЗ основних цифрових і аналогових радіо технологій та можливість їх використання в автоматизованих системах радіомоніторингу.
Методи дослідження:	методи статистичного синтезу й аналізу, аналітичного огляду, об'єктна орієнтованого програмування.

Актуальність атестаційної роботи

- Глобальним напрямком апаратної автоматизації є перехід від автоматизованих режимів роботи засобів радіомоніторингу до напівавтоматичних. Один із таких режимів базується на методі перевірки відповідності параметрів радіовипромінювання передавача на конкретній частоті його спектральній масці (електронному портрету).
- Аналіз, сучасного ринку програмного забезпечення в даній області показав відсутність на вітчизняному ринку автоматичних методів побудови спектральних масок передавачів РЕЗ. Також аналіз показав необхідність їх використання на території нашої держави.

ОРГАНІЗАЦІЙНА СТРУКТУРА ДП «УДЦР»

ДП «УДЦР» - Державне підприємство «Український державний центр радіочастот» - головний регулятор національної **СРЧМ.** Складається з 9 регіональних філій.
СРЧМ – система радіочастотного моніторингу.

АНАЛІЗ НОРМАТИВНИХ ДОКУМЕНТІВ

Згідно з «Планом використання радіочастотного ресурсу України» узагальнені вимоги базових і основних загальних стандартів на технічні параметри передавачів 22 основних цифрових та аналогових радіотехнологій і розроблені спектральні «маски» класів їх радіовипромінювань

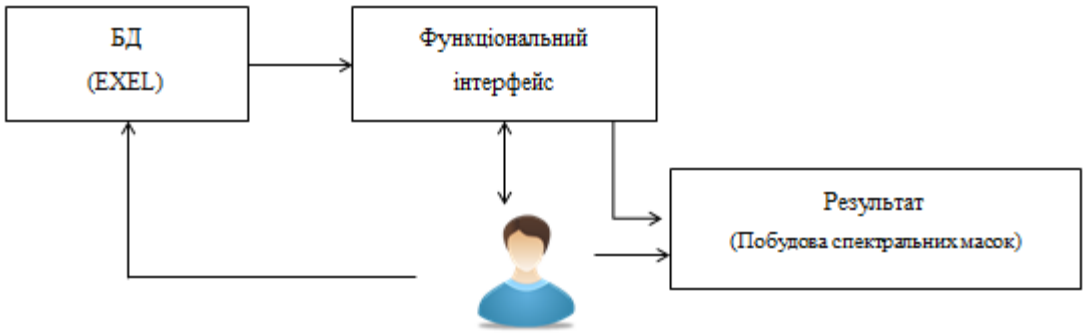
Перелік радіотехнологій
- РЕЗ широкосмугового радіодоступу (ШСРД) в діапазоні частот від 1,35 до 6 ГГц;
- РЕЗ мультисервісного радіодоступу (MMDS) у діапазоні частот від 2,3 до 10,65 ГГц;
- РЕЗ мультимедійного радіодоступу (MVDS) в діапазоні частот від 40,5 до 42,5 ГГц;
- РЕЗ цифрового радіорелейного зв'язку в діапазоні частот від 3,4 ГГц до 40,5 ГГц;
- РЕЗ цифрового стільникового связи CDMA-450 і CDMA-800;
- РЕЗ цифрової бездротової телефонії (DECT);
- РЕЗ цифрового стільникового связи GSM-900 та GSM-1800;
- РЕЗ цифрового стільникового зв'язку IMT-2000 (UMTS);
- РЕЗ цифрового наземного телевізійного мовлення стандартів DVB-T і DVB-T2;
- РЕЗ аналогового звукового мовлення;
- РЕЗ аналогового телевізійного мовлення;
- РЕЗ багатоканального наземного телерадіомовлення;
- РЕЗ аналогового короткохвильового радіозв'язку;
- РЕЗ радіозв'язку передачі даних;
- РЕЗ радіозв'язку берегових та суднових станцій;
- РЕМ радіотелеметрії охоронних і пожежних систем;
- РЕЗ радіолокаційної розвідки і супроводу;
- РЕЗ радіотелеметрії і радіодистанційного управління;
- РЕЗ радіомаяків;
- РЕЗ цифрового та аналогового транкінгового радіозв'язку;

Аналіз мов програмування

Аналіз мов програмування показав, що найкращім інструментом реалізації проекту є мова **Java.**

За вимогою замовника базу даних необхідно реалізувати у вигляді EXEL таблиць для спрощення роботи користувача.

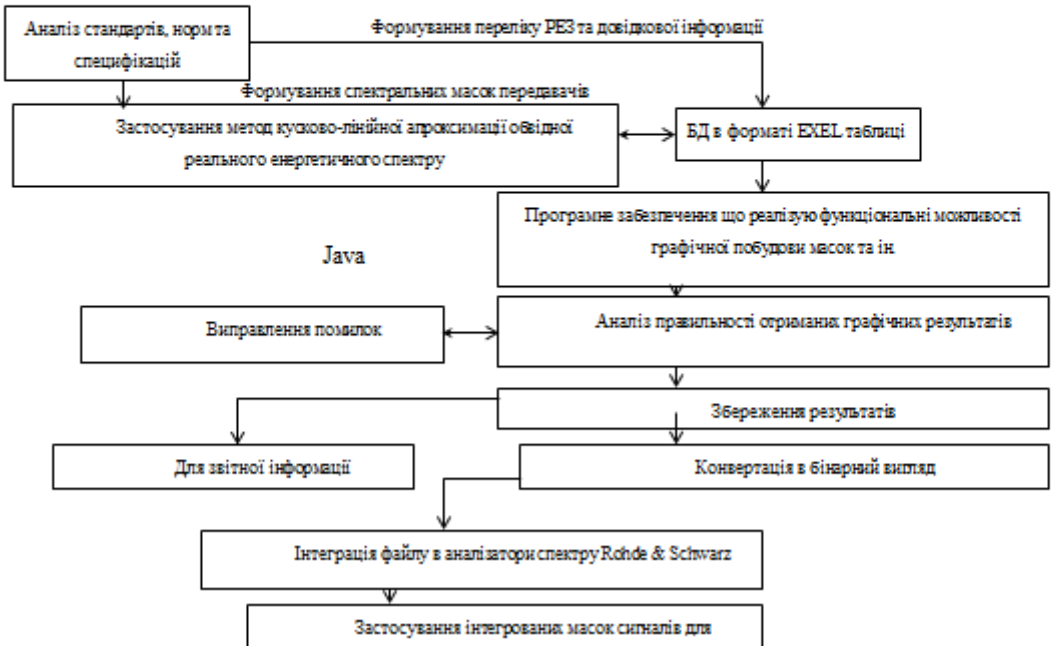
СТРУКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



Вимогами до функціоналу:

- простий інтерфейс для оператора;
- можливість використання таблиць EXEL заданого зразка;
- формування в інтерфейсі каталогу радіотехнологій та РЕЗ в кожній з них;
- відображення спектральних масок передавачів РЕЗ;
- довідкову інформацію про кожну маску під графічним зображенням;
- режим порівняння двох і більше масок;
- збереження отриманих масок у файл для звітної інформації.

АЛГОРИТМ РЕАЛІЗАЦІЇ ПРОЕКТУ



РЕЗУЛЬТАТІВ РОЗРОБКИ БАЗИ ДАНИХ

					Вхідні дані РРС								Нормативні посилання	
№п/п	ID	Кол.РЕЗ (шт/у) тех	Назва/тип РЕЗ	A/D	Мін. част.	Макс. част.	Клас емісії	Пл-ність	Код	Моду-ляція	Пріор.(-ЗдБ)	Пріор.(-30дБ)	EN 302 217-2-2014	Відхил частоти
1	2385	201	9500 MPR 11	D	10700	11700	28M0D7W	4	PSK	QPSK	51	60	B2, 2, >32	±17MHz
2	2386	8	9500 MPR 11	D	10700	11700	28M0G7W	4	PSK	QPSK	51	60	B2, 2, >32	±17MHz
3	2623	1	9500 MPR 15	D	14400	15350	28M0D7W	4	PSK	QPSK	23.Лют	56	D2, 2, >32	±23MHz
4	2571	1	9500 MPR 18	D	17700	19700	13M8D7W	4	PSK	QPSK	8	36	D2, 2, >16	±29MHz
5	2574	39	9500 MPR 18	D	17700	19700	27M5D7W	4	PSK	QPSK	28	36	D2, 2, >32	±29MHz
6	2463	69	9500 MPR 18	D	17700	19700	27M5G7W	4	PSK	QPSK	28	36	D2, 2, >32	±29MHz
7	2664	7	9500 MPR 18	D	17700	19700	55M0D7W	4	PSK	QPSK	60	120	D2, 2, >64	±29MHz
8	2572	34	9500 MPR 38	D	37100	39400	14M0D7W	4	PSK	QPSK	8	40	E2, 2, >16	±55MHz
9	2573	7	9500 MPR 38	D	37100	39400	28M0D7W	4	PSK	QPSK	28	64	E2, 2, >32	±55MHz
10	2361	31	9500 MPR 38	D	37100	39400	56M0D7W	4	PSK	QPSK	51	72	E2, 2, >64	±55MHz
11	2426	0	9500 MPR 8	D	7900	8400	28M0D7W	4	PSK	QPSK	28	36	B2, 2, >32	±12MHz
12	2425	3	9500 MPR 8	D	7900	8400	28M0G7W	4	PSK	QPSK	28	36	B2, 2, >32	±12MHz
13	1137	128	ALCATEL 9413 AWY	D	12750	13250	28M0D7W	128	QAM	128-QAM	26	34	D2, 5H8, >137	±13.5 MHz
14	360	4	ALCATEL 9415 UX	D	14400	15350	7M00D7W	128	QAM	128-QAM	7	12	D2, 5H, >34	±230 KHz

					Маска випромінювання								Додаткова інформація	
Напруга (дБ)	Смуга (МГц)	Напруга (дБ)	Смуга (МГц)	Напруга (дБ)	смуга(МГц)	Напруга(дБ)	смуга(МГц)	Напруга(дБ)	смуга(МГц)	Напруга(дБ)	смуга(МГц)	Напруга(дБ)	смуга (S/N) (МГц)	
+2	25,6	-23	32,8	-23	50	-45	90					-45	140	згідно табл. 2e Image 6a(a)
+2	25,6	-23	32,8	-23	50	-45	90					-45	145	згідно табл. 2e Image 6a(a)
+2	25,6	-23	32,8	-23	50	-45	90					-45	140	згідно табл. 2e Image 6a(a)
+1	13,6	-23	16,8	-23	27,2	-45	48					-45	65	згідно табл. 2d Image 6a(a)
+2	25,6	-23	32,8	-23	50	-45	90					-45	135	згідно табл. 2e Image 6a(a)
+2	25,6	-23	32,8	-23	50	-45	90					-45	135	згідно табл. 2e Image 6a(a)
+2	51,2	-23	65.6	-23	100	-45	180					-45	275	згідно табл. 2f Image 6a(a)
+1	13,6	-23	16,8	-23	27,2	-45	48					-45	70	згідно табл. 2d Image 6a(a)
+2	25,6	-23	32,8	-23	50	-45	90					-45	140	згідно табл. 2e Image 6a(a)
+2	51,2	-23	65.6	-23	100	-45	180					-45	280	згідно табл. 2f Image 6a(a)
+2	25,6	-23	32,8	-23	50	-45	90					-45	140	згідно табл. 2e Image 6a(a)
+2	25,6	-23	32,8	-23	50	-45	90					-45	145	згідно табл. 2e Image 6a(a)
+2	24	-10	29	-32	31	-36	34	-45	80	-55	108	-55	140	згідно табл. 2e Image 6a(e)
+1	6	-10	7,25	-32	7,75	-36	8,5	-45	20	-55	27	-55	35	згідно табл. 2c Image 6a(e)

ФУНКЦІОНАЛЬНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ «SpectrMASK-1.0-SNAPSHOT-jfx»

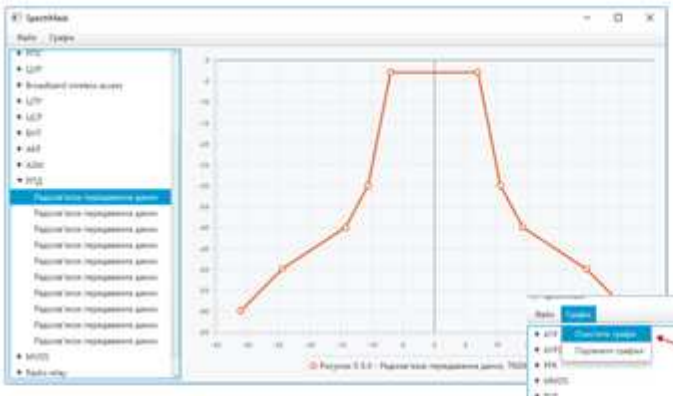
Загальний інтерфейс

Математична модель

$$M(\Delta f) = M_1(\Delta f_i) + M_2 \lg \frac{[\Delta f]}{\Delta f_i}$$

метод кусково-лінійної апроксимації обвідної реального енергетичного спектру

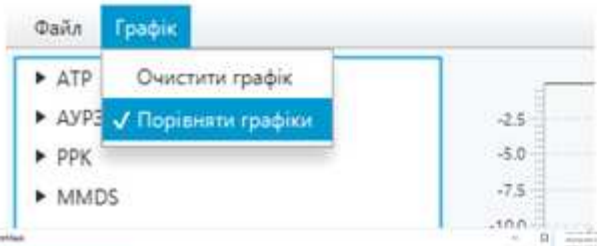
ФУНКЦІОНАЛЬНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ «SpectrMASK-1.0-SNAPSHOT-jfx»



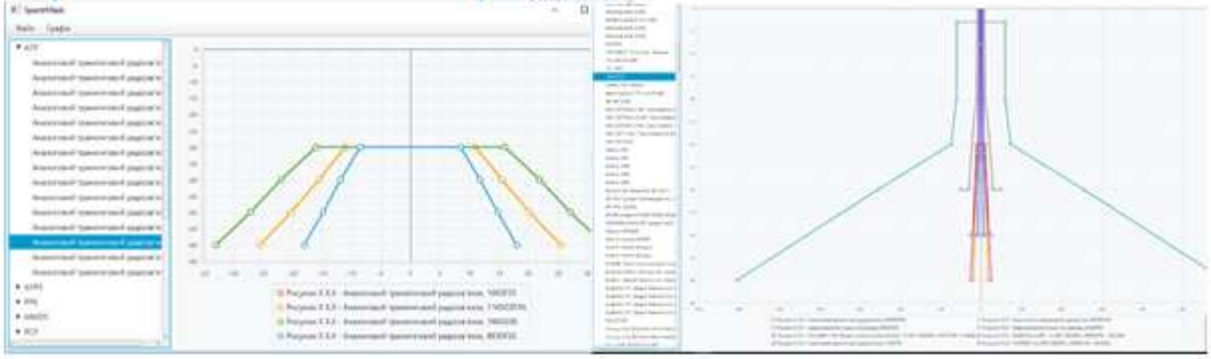
Режим побудови масок з динамічного списку

Режим видалення маски з робочої поверхні графіку

ФУНКЦІОНАЛЬНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ «SpectrMASK-1.0-SNAPSHOT-jfx»



Режим порівняння спектральних масок



транкінговий зв'язок

різні радіотехнології

ІНТЕГРАЦІЯ СПЕКТРАЛЬНИХ МАСОК В FSH-8

ВИСНОВКИ

- Згідно з «Планом використання радіочастотного ресурсу України» узагальнені вимоги базових і основних загальних стандартів на технічні параметри передавачів 22 основних цифрових та аналогових радіотехнологій і розроблені спектральні «маски» класів їх радіовипромінювань.
- У відповідності з розробленими спектральними «масками» класів радіовипромінювань за даними бази частотних присвоєнь ДП «УДЦР» створено у табличному вигляді інформаційний базис з даними щодо параметрів спектральних масок радіовипромінювань і допустимих значень відхилення присвоєних частот по кожному із типів РЕЗ.
- Розроблено базу даних інформаційного базису, алгоритм і спеціалізоване програмне забезпечення автоматизованого формування спектральних масок у графічному вигляді.
- Розроблено алгоритм автоматизованої інтеграції масок передавачів у аналізатори спектру, що використовуються радіочастотними органами України.

ДОДАТОК В

Відомості атестаційної роботи магістра

Позначення	Найменування	Дод. відомості
	Текстові документи	
1 ГЮІК.ХХХХХХ.021 ПЗ	Пояснювальна записка	118 с.
	Графічні документи	
2 ГЮІК.ХХХХХ.001 Д1	Графічна частина	16 слайдів
	Носії інформації	
	Компакт - диск (CD)	1

					ГЮІК. ХХХХХХ .013 Д1			
Змін.	Арк.	Номер докум.	Підп.	Дата				
Розроб.		Пішоха К.С.			Інформаційне забезпечення систем радіомоніторингу		Аркуш	Аркушів
Перевір.		Цопа О.І.						
Н.контр.						ХНУРЕ кафедра РТІКС		
Затв.					Відомість атестаційної роботи магістра			