**Software description**

The main class in the SW Helicopter is Display. It creates main Frame and connects the main classes of the program. Figure 1 shows interconnections between classes. Instance of this class has the JFrame type.



1. **Class PortSettings**

**Variables:**

public static CommPortIdentifier thePortID;

public static CommPort thePort;

Class which includes following **methods:**

1.1. public static void SimpleWrite\_request (byte Request)

* forming the message to request parameters from the helicopter;
* calculating the CRC;
* open and initialisation of comport;
* sending the message to helicopter;

1.2.public static void SimpleWrite\_update()

* getting the parameters of desirable values of control parameters from the textfield of the Frame;
* calculating the CRC;
* open and initialisation of comport;
* sending the message to helicopter;

1.3.public static void SimpleWrite\_Parametr\_request ()

* sending the request for getting the parameters from helicopter;
* calculating the CRC;
* open and initialisation of comport;
* sending the message to helicopter;

1.4.public static void SimpleWrite\_response ()

* was created to check the functionally of this program by connection two computer via Xbee;
* sending the test message with predefined values

1.5.public static String FindPortIdentification()

* identify the comport automatically

1. **class BytesTransformation**

Making all transformation with primitive variables;

Methods:

* 1. public static int UnsignedbyteToInt (byte first)

transforming unsignedbyte to integer

* 1. static byte [] charToByteArray( char c )

transforming char to two bytes array

* 1. public static char CRC(byte []bytes)

calculating the CRC of presented byte array

1. **public class SimpleRead\_response**

implements Runnable, SerialPortEventListener

**variables:**

Grafics graf;

static CommPortIdentifier portId;

static Enumeration portList;

InputStream source;

InputStream inputStream;

SerialPort serialPort;

Thread readThread;

ByteArrayInputStream bais;

JTextArea text= new JTextArea(10,10);

**Methods:**

public static void Start()

* inisialisation
* opening the com port

public SimpleRead\_response()

* retrieves the graphical part of the program
* getting information from the helicopter
* display it in jTextArea as a text

1. **public class Grafics**

extends ApplicationFrame implements ActionListener Grafics

**variables:**

public static final int SUBPLOT\_COUNT = 4;

private TimeSeriesCollection[] datasets;

private double[] lastValue = new double[SUBPLOT\_COUNT];

**method:**

public Grafics(final String title)

creates a graphics. The number of graphics is SUBPLOT\_COUNT

1. **public class Setting**

extends javax.swing.JFrame

creates the form for filling by operator. Saving,getting and reading parameters of helicopter

1. **public class SimpleRead\_parametr\_response**

implements Runnable, SerialPortEventListener

**variables:**

Grafics demo;

static CommPortIdentifier portId;

static Enumeration portList;

static String t;

String FILENAME = "binary.dat";

InputStream source;

InputStream inputStream;

SerialPort serialPort;

Thread readThread;

ByteArrayInputStream bais;

JEditorPane text1= new JEditorPane();

JEditorPane text2= new JEditorPane();

static char v[]=new char [17];

JTextField kpR=new JTextField();

JTextField kiR=new JTextField();

JTextField kdR=new JTextField();

JTextField kpP=new JTextField();

JTextField kiP=new JTextField();

JTextField kdP=new JTextField();

JTextField kpY=new JTextField();

JTextField kiY=new JTextField();

JTextField kdY=new JTextField();

JTextField lpR=new JTextField();

JTextField hpR=new JTextField();

JTextField lpP=new JTextField();

JTextField hpP=new JTextField();

JTextField inertiaX=new JTextField();

JTextField inertiaY=new JTextField();

JTextField inertiaZ=new JTextField();

JTextField lenBoom=new JTextField();

**Methods:**

* 1. **public static void Start()**

initialisation of com port

* 1. **public SimpleRead\_parametr\_response()**

getting static data from the helicopter and create the Frame, where it displays them

1. **public class Utils**

**7.1. public static String getExtension(File f)**

getting extension from the chosen file name

**7.2. public static String getbody(File f)**

getting the body of the chosen file name

1. **public class PropMgr**

**t**his class was created to work with properties file.

**8.1.public static void defaultPropswrite()**

write default properties to the file "defaultProps2**"**

**8.2.** **public static String defaultPropsread(String t)**

Read the default properties to the file "defaultProps2"

**8.3.public static String readProps(String otkuda,String t)**

Read properties from the chosen properties file

**8.4.** **public static void writeProps**

Write properties to the desirable file

1. **public class Objects**

need to create brief structure of the sending or receiving message

**methods**:

9.1.static byte id(byte ID)

9.2. static byte dlc(byte DLC)

9.3. static byte request(byte Request)

9.4.static char CRC(char crc)

9.5. static char accXRaw(char accXRaw){9.6. static char accYRaw(char accYRaw)

9.7.static short accY(short accY)

9.8. static char accZRaw(char accZRaw)

9.9. static short accZ(short accZ)

9.10.static short velZ(short velZ)

9.11.static short posZ(short posZ)

9.12.static char angVelRRaw(char angVelRRaw)

9.13.static short angVelR(short angVelR)

9.14.static short angR(short angR)

9.15.static char angVelPRaw(char angVelPRaw)

9.16.static short angVelP(short angVelP)

9.17.static short angP(short angP)

9.18. static char angVelYRaw(char angVelYRaw)

9.19.static short angVelY(short angVelY)

9.20. static short angY(short angY)

9.21. static int battery(byte battery)

9.22.static char tempRaw(char tempRaw)

9.23.static int temp(byte temp)

9.24. static char batteryRaw(char batteryRaw)

9.25.static char airPressureRaw(char airPressureRaw)

9.26.static int airPressure(byte airPressure)

9.27.static char rpmFront(char rpmFront)

9.28.static char rpmLeft(char rpmLeft)

9.29. static char rpmRear(char rpmRear)

9.30.static char rpmRight(char rpmRight)

9.31.static char forceTotal(char forceTotal)

9.32.static int remoteForceRaw(byte remoteForceRaw)

9.33.static char remoteForce(char remoteForce)

9.34.static int remoteYawRaw(byte remoteYawRaw)

9.35 static char lpP(char lpP)

9.36.static char remoteYaw(char remoteYaw)

9.37.static int remotePitchRaw(byte remotePitchRaw)

9.38.static char remotePitch(char remotePitch)

9.39.static int remoteRollRaw(byte remoteRollRaw)

9.40.static char remoteRoll(char remoteRoll)

9.41. static int error(byte error)

9.42.static char x\_axis\_p(char x\_axis\_p)

9.43.static char x\_axis\_d(char x\_axis\_d)

9.44.static char pitch\_p(char pitch\_p)

9.45.static char pitch\_i(char pitch\_i)

9.46.static char pitch\_d(char pitch\_d)

9.47.static char yaw\_p(char yaw\_p)

9.48.static char yaw\_i(char yaw\_i)

9.49. static char yaw\_d(char yaw\_d)

9.50.static char lpR(char lpR)

9.51. static char hpR(char hpR)

9.52.static char hpP(char hpP)

9.53. static char inertiaX(char inertiaX)

9.54.static char inertiaY(char inertiaY)

9.55.static char inertiaZ(char inertiaZ)

9.56.static char lenBoom(char lenBoom)

**Created classes for testing procedure:**

1. **public class readfromfile\_mod()**

reading from the file

**Variables**

public static DataInputStream inDataStream;

public static int [] v= new int[30];

long de;

static int i=0;

**method:**

**public int[] readfromfile\_mod()**

read from the file put the data to the outcoming stream

**public class writetofile**

**Method:**

Writing some numbers to the file

**public writetofile()**