```
package seafarm;

import java.math.BigInteger;
import java.util.Arrays;


/**
 *
 * @author Jørgen
 */

/*

Data transfer Protocoll:

//////FROM GUI TO RASBERRY PI///////

Byte 0
    - Bit [0] = Stopp
    - Bit [1] = Forward
    - Bit [2] = Reverse
    - Bit [3] = Left
    - Bit [4] = Right
    - Bit [5] = glideLeft
    - Bit [6] = glideRight
    - Bit [7] = **free**

Byte 1
    - light Value (mapped from 0 - 100)

Byte 2
    - thruster Value   (mapped from 0 - 100)


Byte 3
    - Bit [0] = **free**
    - Bit [1] = **free**
    - Bit [2] = Start/stop ROV   (value 1 = Start, value 0 = stop)
    - Bit [3] = **free**
    - Bit [4] = **free**
    - Bit [5] = **free**
    - Bit [6] = **free**
    - Bit [7] = **free**
```

```
Byte 4
    - **RESERVED**


//////FROM RASBERRY TO GUI///////

Receiving protocol:
Byte 0: Pixy x value lowbyte
Byte 1: Pixy x value highbyte
Byte 2: Pixy y value lowbyte
Byte 3: Pixy y value highbyte
Byte 4: Distance sensor
Byte 5: reserved


 */
public class RovDatahandler {

    private byte[] dataFromArduino;
    private byte[] dataFromGui;
    private boolean dataFromArduinoAvaliable = false;
    private boolean dataFromGuiAvailable = false;
    private boolean threadStatus;
    // private int pixyXvalue;
    // private int pixyYvalue;
    private int distanceSensor;
    private byte requestCodeFromArduino;
    private boolean enableROV;


    public RovDatahandler() {
        this.dataFromArduino = new byte[23];
        this.dataFromGui = new byte[6];
        //this.sendData = new SendData();
    }

    //*****************************************************************
    //********** PRIVATE METHODS AREA*********************************
    private byte setBit(byte b, int bit) {
        return b |= 1 << bit;
    }

    private byte releaseBit(byte b, int bit) {
        return b &= ~(1 << bit);
```

```
}

//*****************************************************************
//********************** THREAD STATUS METHODS**********************
public boolean shouldThreadRun() {
    return threadStatus;
}

public void setThreadStatus(boolean threadStatus) {
    this.threadStatus = threadStatus;
}

//*****************************************************************
//************** FROM GUI METHODS*********************************
/**
 * returns the byte array protokoll sendt from GUI
 *
 * @return dataFromGui
 */
public byte[] getDataFromGui() {
    SeaFarm.enumStateEvent = RovSendEventState.FALSE;
    return this.dataFromGui;

}

/**
 * setting the byte from protocoll to high
 */
public void stopROV() {
    dataFromGui[0] = this.setBit(dataFromGui[0], 0);
    //this.fireStateChanged();
    this.sendData();
}

/**
 * setting the byte from protocoll to Low
 */
public void releaseStopROV() {
    dataFromGui[0] = this.releaseBit(dataFromGui[0], 0);
    // this.fireStateChanged();
    this.sendData();
}

/**
```

```
 * setting the byte from protocoll to high
 */
public void goFwd() {
    dataFromGui[0] = this.setBit(dataFromGui[0], 1);
    // this.fireStateChanged();
    this.sendData();
}

/**
 * setting the byte from protocoll to Low
 */
public void releaseGoFwd() {
    dataFromGui[0] = this.releaseBit(dataFromGui[0], 1);
    // this.fireStateChanged();
    this.sendData();
}

/**
 * setting the byte from protocoll to high
 */
public void goRew() {
    dataFromGui[0] = this.setBit(dataFromGui[0], 2);
    // this.fireStateChanged();
    this.sendData();
}

/**
 * setting the byte from protocoll to Low
 */
public void releaseGoRew() {
    dataFromGui[0] = this.releaseBit(dataFromGui[0], 2);
    // this.fireStateChanged();
    this.sendData();
}

/**
 * setting the byte from protocoll to high
 */
public void goLeft() {
    dataFromGui[0] = this.setBit(dataFromGui[0], 3);
    //this.fireStateChanged();
    this.sendData();
}
```

```
/**
 * setting the byte from protocoll to Low
 */
public void releaseGoLeft() {
    dataFromGui[0] = this.releaseBit(dataFromGui[0], 3);
    // this.fireStateChanged();
    this.sendData();
}

/**
 * setting the byte from protocoll to high
 */
public void goRight() {
    dataFromGui[0] = this.setBit(dataFromGui[0], 4);
    //this.fireStateChanged();
    this.sendData();
}

public void releaseGoRight() {
    dataFromGui[0] = this.releaseBit(dataFromGui[0], 4);
    //this.fireStateChanged();
    this.sendData();
}

/**
 * setting the byte from protocoll to high
 */
public void setSlideLeft() {
    dataFromGui[0] = this.setBit(dataFromGui[0], 5);
    //this.fireStateChanged();
    this.sendData();
}

/**
 * setting the byte from protocoll to Low
 */
public void resetSlideLeft() {
    dataFromGui[0] = this.releaseBit(dataFromGui[0], 5);
    //this.fireStateChanged();
    this.sendData();
}

/**
 * setting the byte from protocoll to high
```

```
 */
public void setSlideRight() {
    dataFromGui[0] = this.setBit(dataFromGui[0], 6);
    //this.fireStateChanged();
    this.sendData();
}

/**
 * setting the byte from protocoll to Low
 */
public void resetSlideRight() {
    dataFromGui[0] = this.releaseBit(dataFromGui[0], 6);
    //this.fireStateChanged();
    this.sendData();
}

/**
 * setting the byte from protocoll to high
 */
public void setLightValue(byte sensetivity) {
    dataFromGui[1] = sensetivity;
    //this.fireStateChanged();
    this.sendData();
}

/**
 *
 * @return dataFromGui on byte 1
 */
public byte getLightValue() {
    return dataFromGui[1];
}

/**
 * setting the byte from protocoll to high
 */
public void setThrusterValue(byte sensetivity) {
    dataFromGui[2] = sensetivity;
    //this.fireStateChanged();
    this.sendData();
}

/**
 *
```

```
     * @return dataFromGui on byte 2
     */
    public byte getThrusterValue() {
        return dataFromGui[2];
    }

    public void enableROV() {
        dataFromGui[3] = this.setBit(dataFromGui[3], 3);
        //this.fireStateChanged();
        this.sendData();
    }

    public void disableROV() {
        dataFromGui[3] = this.releaseBit(dataFromGui[3], 3);
        //this.fireStateChanged();
        this.sendData();
    }

    /**
     *
     * @return dataFromGui on byte 5
     */
    public byte getRequestCode() {
        return dataFromGui[5];
    }

    /**
     * increments the byte on array[5] with one for each run
     */
    public void incrementRequestCode() {
        dataFromGui[5]++;
        //this.fireStateChanged();
        this.sendData();
    }

    /**
     * creates a new udp connection and sends the data
     */
    public void sendData() {
        new RovUDPsender().send(SeaFarm.ROVIPADDRESS, dataFromGui,
SeaFarm.ROVSENDPORT);
    }

}
```