```java
package userApplication2;
import java.io.IOException;

import javax.sound.sampled.LineUnavailableException;


public class main {

    public static void main(String[] args) throws
LineUnavailableException,IOException {

//  echo e1 = new echo();
//  e1.echofunction("E3554\r",38020,48020);
//  e1.echofunction("E0000\r",38020,48020);

//  image i1 = new image();
//  i1.imagefunction("M7494 CAM=FIX \r",38020,48020);

    sound s1 = new sound();
//  s1.DCPMsoundfunction("A1120\rF999",38020,48020); //LXX APIΘMOΣ
    s1.AQDCPMsoundfunction("A1120\rAQF999",38020,48020);

//  vehicle v1 = new vehicle();
//  v1.vehiclefunction("V0165 OBD=01 ", 38020, 48020);

//  ithakicopter i1 = new ithakicopter();
//  i1.ithakicopterfunction("Q6175\r", 38020,48078); //48078 is constant!

//  temperature t1 = new temperature();
//  t1.temperaturefunction("E3554 ", 38020, 48020);
    }
}
```

```java
package userApplication2;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.ArrayList;
import java.util.List;
import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.Writer;
```

```
public class echo {

    public void echofunction(String packetInfo,int serverPort,int clientPort )
throws IOException{
        DatagramSocket s = new DatagramSocket();
        byte[] txbuffer = packetInfo.getBytes(); //echo bytes
        byte[] hostIP = { (byte)155,(byte)207,18,(byte)208 };
        InetAddress hostAddress = InetAddress.getByAddress(hostIP);
        DatagramPacket p = new DatagramPacket(txbuffer,txbuffer.length,
        hostAddress,serverPort); //packet to server from client
        s.send(p);
        DatagramSocket r = new DatagramSocket(clientPort);
        r.setSoTimeout(3200);
        byte[] rxbuffer = new byte[2048];
        DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length); //server
to client
        //Managing Packages
        List<Long> deltas=new ArrayList<Long>();
        List<Double> counters=new ArrayList<Double>();
        List<Integer> sums=new ArrayList<Integer>();
        Writer responses = null;
        Writer throughput = null;
        Writer srtt = null;
        Writer sigma = null;
        Writer rto = null;
        if(packetInfo == "E0000\r"){
        responses = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("G1withoutdelay.txt"), "utf-8"));
        throughput = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("G2withoutdelay.txt"), "utf-8"));
        srtt = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("srttwithoutdelay.txt"), "utf-8"));
        sigma = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("sigmawithoutdelay.txt"), "utf-8"));
        rto = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("rtowithoutdelay.txt"), "utf-8"));
        }else{
        responses = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("G1withdelay.txt"), "utf-8"));
        throughput = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("G2withdelay.txt"), "utf-8"));
        srtt = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("srttwithdelay.txt"), "utf-8"));
        sigma = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("sigmawithdelay.txt"), "utf-8"));
```

```
        rto = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("rtowithdelay.txt"), "utf-8"));
        }
        long timeStart=System.currentTimeMillis();

        for (;;) {
        long t1 = System.currentTimeMillis();
        s.send(p);
        try {
        r.receive(q);
        long t2 = System.currentTimeMillis();
        String message = new String(rxbuffer,0,q.getLength());

        long delta = t2-t1;
        System.out.println(delta);
        deltas.add(delta);
        responses.write(delta+",");
        long executionTime = (System.currentTimeMillis()-timeStart)/1000;

        int sum=0;
        double counter = 0;
        for (int i=0;i<deltas.size();i++){
            int j=i;
            while ((sum/1000<8) && (j<deltas.size())){
                sum += deltas.get(j);
                counter++;
                j++;
            }
            counter = counter*0.125;
            counters.add(counter);
            throughput.write(counter+",");
            sums.add(sum);
            counter = 0;
            sum = 0;
        }

        if(executionTime > 240) {
            System.out.println(deltas);
            break;
            }
        } catch (Exception x) {
        System.out.println(x);
            }
        }
        if(packetInfo != "E0000\r"){
```

```
        double alpha = 0.2;
        double beta = 0.5;
        double ce = 1;
        double x_srtt = 0;
        double x_sigma = 0;
        double x_rto = 0;
        for (int j=0; j<deltas.size(); j++) {
                x_srtt = alpha*x_srtt + deltas.get(j)*(1-alpha);
                srtt.write(x_srtt + ",");
                x_sigma = beta*x_sigma + Math.abs(x_srtt-deltas.get(j)*(1-beta));
                sigma.write(x_sigma + ",");
                x_rto = x_srtt + x_sigma * ce;
                rto.write(x_rto + ",");
            }
        }


        s.close();
        r.close();
        srtt.close();
        rto.close();
        sigma.close();
        responses.close();
        throughput.close();
    }



}
```

```
package userApplication2;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.io.*;
import java.awt.*;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.FileOutputStream;
import java.util.ArrayList;
import java.util.*;
public class image {
```

```
    public void imagefunction(String packetInfo,int serverPort,int clientPort)
throws IOException {
        String fileName = "image2.jpg";
        ArrayList<Byte> stream = new ArrayList<Byte>();
        boolean terminated = false;
        DatagramSocket s = new DatagramSocket();
        byte[] txbuffer = packetInfo.getBytes();
        byte[] hostIP = { (byte)155,(byte)207,18,(byte)208 };
        InetAddress hostAddress = InetAddress.getByAddress(hostIP);
        DatagramPacket p = new DatagramPacket(txbuffer,txbuffer.length,
        hostAddress,serverPort);
        s.send(p);
        DatagramSocket r = new DatagramSocket(clientPort);
        r.setSoTimeout(3200);
        byte[] rxbuffer = new byte[128];
        DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);
        s.send(p);
        int counter = 0;
        for (;;) {
            try {
            r.receive(q);
            for(int i=0; i<rxbuffer.length; i++) {
                stream.add(rxbuffer[i]);
                if((counter==0)&&(stream.size()>2)&&(stream.get(stream.size()-
2)==(byte)0xFF)&&(stream.get(stream.size()-1)==(byte)0xD8)) {
                    System.out.println("Image Start Captured");
                    counter=5;
                    stream.clear();
                    stream.add((byte)0xFF);
                    stream.add((byte)0xD8);
                   }
                if((stream.size()>2)&&(stream.get(stream.size()-
2)==(byte)0xFF)&&(stream.get(stream.size()-1)==(byte)0xD9)) {
                    System.out.println("Image End Captured");
                    terminated=true;

                   }
                if(terminated) {
                        break;
                   }
            }
            if(terminated) {
                break;
            }
```

```
            } catch (Exception x) {
            System.out.println(x);
            }
        }

        byte[] byteArray = new byte[stream.size()];
        for (int i=0; i<stream.size(); i++) {
            byteArray[i] = stream.get(i);
        }
        ByteArrayInputStream bis = new ByteArrayInputStream(byteArray);
        BufferedImage bImage = ImageIO.read(bis);
        ImageIO.write(bImage, "jpg", new File(fileName) );

        System.out.println(fileName + " created");
        FileOutputStream fos = new FileOutputStream("C:\\demo\\imagefile.jpg");
        fos.write(byteArray);
        System.out.println(Integer.toHexString(byteArray[0]));
        System.out.println(Integer.toHexString(byteArray[1]));
        System.out.println(Integer.toHexString(byteArray[stream.size()-2]));
        System.out.println(Integer.toHexString(byteArray[stream.size()-1]));
        s.close();
        r.close();
}
}
```

```
package userApplication2;
import java.util.ArrayList;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.Writer;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;


public class ithakicopter {
    public void ithakicopterfunction(String packetInfo,int serverPort,int
clientPort) throws IOException{
        DatagramSocket s = new DatagramSocket();
        byte[] txbuffer = packetInfo.getBytes();
        byte[] hostIP = { (byte)155,(byte)207,18,(byte)208 };
```

```
        InetAddress hostAddress = InetAddress.getByAddress(hostIP);
        ArrayList <Float> lmotors = new ArrayList<Float>();
        ArrayList <Float> rmotors = new ArrayList<Float>();
        ArrayList <Float> altitudes = new ArrayList<Float>();
        ArrayList <Float> temperatures = new ArrayList<Float>();
        ArrayList <Float> pressures = new ArrayList<Float>();
        Writer tlmtr = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("lmotors.txt"), "utf-8"));
        Writer trmtr = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("rmotors.txt"), "utf-8"));
        Writer talt = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("altitudes.txt"), "utf-8"));
        Writer ttemp = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("temperatures.txt"), "utf-8"));
        Writer tpres = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("pressures.txt"), "utf-8"));
        DatagramPacket p = new DatagramPacket(txbuffer,txbuffer.length,
        hostAddress,serverPort);
        DatagramSocket r = new DatagramSocket(clientPort);
        r.setSoTimeout(5000);
        byte[] rxbuffer = new byte[128];
        DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);
        String[] splitted = new String[9];
        float lmotor,rmotor,pressure,altitude,temperature;
        long timeStart=System.currentTimeMillis();
        s.send(p);
        for (;;) {
        try {
        r.receive(q);
        String message = new String(rxbuffer,0,q.getLength());

        splitted = message.split(" " , 0);

        System.out.println(message);

        String unilmotor = splitted[3];
        String[] splittedlmotor = unilmotor.split("=", 0);
        lmotor=Float.parseFloat(splittedlmotor[1]);
        System.out.println(lmotor);
        lmotors.add(lmotor);
        tlmtr.write(lmotor+",");

        String unirmotor = splitted[4];
        String[] splittedrmotor = unirmotor.split("=", 0);
        rmotor=Float.parseFloat(splittedrmotor[1]);
```

```
        System.out.println(rmotor);
        rmotors.add(rmotor);
        trmtr.write(rmotor+",");

        String unialtitude = splitted[5];
        String[] splittedaltitude = unialtitude.split("=", 0);
        altitude=Float.parseFloat(splittedaltitude[1]);
        System.out.println(altitude);
        altitudes.add(altitude);
        talt.write(altitude+",");


        String unitemperature = splitted[6];
        String[] splittedtemperature = unitemperature.split("=", 0);
        temperature=Float.parseFloat(splittedtemperature[1]);
        System.out.println(temperature);
        temperatures.add(temperature);
        ttemp.write(temperature+",");

        String unipressure = splitted[7];
        String[] splittedpressure = unipressure.split("=", 0);
        pressure=Float.parseFloat(splittedpressure[1]);
        System.out.println(pressure);
        pressures.add(pressure);
        tpres.write(pressure+",");
        System.out.println(System.currentTimeMillis()-timeStart);
        if((System.currentTimeMillis()-timeStart)/1000>120){
            break;
        }
        } catch (Exception x) {
        System.out.println(x);

        }

        }
        trmtr.close();
        tlmtr.close();
        ttemp.close();
        talt.close();
        tpres.close();
        s.close();
        r.close();
    }
}
```

```java
package userApplication2;
import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.ArrayList;

import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.SourceDataLine;

public class sound {

    public void DCPMsoundfunction(String packetInfo,int serverPort,int clientPort
) throws LineUnavailableException,IOException{
        int numberOfPackets=999;
        int beta = 3; // mporei kai 4
        Writer txtdiffs = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("DCPMdifferences.txt"), "utf-8"));
        Writer txtsamples = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("DCPMsamples.txt"), "utf-8"));
        DatagramSocket s = new DatagramSocket();
        ArrayList<Integer> diffs = new ArrayList<Integer>();
        ArrayList<Integer> samples = new ArrayList<Integer>();
        byte[] txbuffer = packetInfo.getBytes();
        byte[] hostIP = { (byte)155,(byte)207,18,(byte)208 };
        InetAddress hostAddress = InetAddress.getByAddress(hostIP);
        DatagramPacket p = new DatagramPacket(txbuffer,txbuffer.length,
        hostAddress,serverPort);
        s.send(p);
        DatagramSocket r = new DatagramSocket(clientPort);
        r.setSoTimeout(4000);
        byte[] rxbuffer = new byte[128];
        DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);
        int nible1 = 0;
        int nible2 = 0;
        int d1=0,d2=0,sample1=0,sample2=0;
        s.send(p);
        for(int y=1;y<numberOfPackets;y++) {
        try {
```

```
        r.receive(q);
            for(int i=0;i<rxbuffer.length;i++) {
                nible1 = rxbuffer[i] & 15;
                nible2 =((rxbuffer[i] & 240)>>4);
                d1 = (nible1-8)*beta;
                d2 = (nible2-8)*beta;
                diffs.add(d1);
                diffs.add(d2);
                sample1 = sample2 + d2;
                sample2 = sample1 + d1;
                samples.add(sample1);
                samples.add(sample2);
                }
            }catch (Exception x) {
        System.out.println(x);
            }
        }
        byte[] song = new byte[samples.size()];
        for(int i=0; i<samples.size();i++) {
            song[i]=(byte)(int)samples.get(i);
            txtsamples.write(samples.get(i)+",");
        }
        for(int i=0; i<diffs.size(); i++){
            txtdiffs.write(diffs.get(i)+",");
        }
        AudioFormat decoder = new AudioFormat(8000,8,1,true,false);
        SourceDataLine playsong = AudioSystem.getSourceDataLine(decoder);
        playsong.open(decoder,32000);
        playsong.start();
        playsong.write(song,0,song.length);
        playsong.stop();
        playsong.close();
        s.close();
        r.close();
    }



public void AQDCPMsoundfunction(String packetInfo,int serverPort,int clientPort )
throws LineUnavailableException,IOException{
    int numberOfPackets=999;
    ArrayList <Integer> means = new ArrayList<Integer>();
    ArrayList <Integer> samples = new ArrayList<Integer>();
    ArrayList <Integer> steps = new ArrayList<Integer>();
```

```
    Writer txtsteps = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("aqDCPMsteps1.txt"), "utf-8"));
    Writer txtsamples = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("aqDCPMsamples.txt"), "utf-8"));
    Writer txtmeans = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("aqDCPMmeans1.txt"), "utf-8"));
    DatagramSocket s = new DatagramSocket();
    byte[] txbuffer = packetInfo.getBytes();
    byte[] hostIP = { (byte)155,(byte)207,18,(byte)208 };
    InetAddress hostAddress = InetAddress.getByAddress(hostIP);
    DatagramPacket p = new DatagramPacket(txbuffer,txbuffer.length,
    hostAddress,serverPort);
    s.send(p);
    DatagramSocket r = new DatagramSocket(clientPort);
    r.setSoTimeout(4000);
    byte[] rxbuffer = new byte[132];
    DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);
    int nible1=0, nible2=0, d1=0, d2=0, sample1=0, sample2=0, step=0, mean=0,
tempByte, count=0;

    byte[] song = new byte[4*numberOfPackets*132];
    s.send(p);

    for(int i=1;i<numberOfPackets;i++) {
        try {
        mean = (int) (rxbuffer[0] + Math.pow(2.0, 8.0) * rxbuffer[1]);
        means.add(mean);
        txtmeans.write((float)mean+",");
        step = (int) (rxbuffer[2] + Math.pow(2.0, 8.0) * rxbuffer[3]);
        steps.add(step);
        txtsteps.write((float)step+",");
        r.receive(q);
        for(int j=4;j<rxbuffer.length;j++) {

        tempByte = rxbuffer[j];

        nible1 = (int) (tempByte & 0x0000000F) - 8;
        nible2 = (int) ((tempByte & 0x000000F0)>>4) - 8;

        sample1 = nible2*step + mean;
        sample2 = nible1*step + mean;
        samples.add(sample1);
        txtsamples.write(sample1+",");
        samples.add(sample2);
        txtsamples.write(sample2+",");
```

```
        song[count] = (byte) (sample1 & 0xFF);
        count++;
        song[count] = (byte) ((sample1 >> 8) & 0xFF);
        count++;
        song[count] = (byte) (sample2 & 0xFF);
        count++;
        song[count] = (byte) ((sample2 >> 8) & 0xFF);
        count++;
        }
        }catch (Exception x) {
        System.out.println(x);
            }
        }
        AudioFormat decoder = new AudioFormat(8000,16,1,true,false);
        SourceDataLine playsong = AudioSystem.getSourceDataLine(decoder);
        playsong.open(decoder,32000);
        playsong.start();
        playsong.write(song,0,song.length);
        playsong.stop();
        playsong.close();
        txtsamples.close();
        txtmeans.close();
        txtsteps.close();
        s.close();
        r.close();
}
}


package userApplication2;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class temperature {

    public void temperaturefunction(String packetInfo,int serverPort,int
clientPort ) throws IOException{
        DatagramSocket s = new DatagramSocket();
        byte[] hostIP = { (byte)155,(byte)207,18,(byte)208 };
        InetAddress hostAddress = InetAddress.getByAddress(hostIP);
        DatagramSocket r = new DatagramSocket(clientPort);
        r.setSoTimeout(2000);
        byte[] rxbuffer = new byte[2048];
```

```
        DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);
        for (int i=0;i<10;i++) {
        try {
        byte[] txbuffer = (packetInfo+"T0"+String.valueOf(i)).getBytes();
        DatagramPacket p = new DatagramPacket(txbuffer,txbuffer.length,
        hostAddress,serverPort);
        s.send(p);
        r.receive(q);
        String message = new String(rxbuffer,0,q.getLength());

        if(message.contains("C")){
            String k = new String(rxbuffer, 44, 2);
            System.out.println(k);
            System.out.println(message);
        }
        } catch (Exception x) {
        System.out.println(x);
        }
    }
    for (int i=10;i<100;i++) {
        try {
        byte[] txbuffer = (packetInfo+"T"+String.valueOf(i)).getBytes();
        DatagramPacket p = new DatagramPacket(txbuffer,txbuffer.length,
        hostAddress,serverPort);
        s.send(p);
        r.receive(q);
        String message = new String(rxbuffer,0,q.getLength());
        if(message.contains("C")){
            String k = new String(rxbuffer, 44, 2);
            System.out.println(k);
            System.out.println(message);
        }
        } catch (Exception x) {
        System.out.println(x);
        }
    }

    }



}
```

```
package userApplication2;

import java.io.IOException;
import java.io.Writer;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.BufferedWriter;

public class vehicle {

    public void vehiclefunction(String packetInfo,int serverPort,int clientPort )
throws IOException{
        DatagramSocket s = new DatagramSocket();
        String[] modeId = {"1F", "0F", "11", "0C", "0C", "05"};
        byte[] hostIP = { (byte)155,(byte)207,18,(byte)208 };
        InetAddress hostAddress = InetAddress.getByAddress(hostIP);
        DatagramSocket r = new DatagramSocket(clientPort);
        r.setSoTimeout(2000);
        byte[] rxbuffer = new byte[2048];
        DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);
        String xstring ="",ystring="";
        int x1,y1, engineTime=0,
intakeAirTemp=0,throttle=0,rpm=0,speed=0,coolanttemp=0;
        Writer teng = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("enginetime.txt"), "utf-8"));
        Writer tintake = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("intake.txt"), "utf-8"));
        Writer tthrotle = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("throttle.txt"), "utf-8"));
        Writer trpm = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("rpm.txt"), "utf-8"));
        Writer tspeed = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("speed.txt"), "utf-8"));
        Writer tcoolant = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("coolanttemp.txt"), "utf-8"));
        long timeStart=System.currentTimeMillis();
        while((System.currentTimeMillis()-timeStart)/1000<240){
        for (int i=0;i<6;i++) {
        try {
        byte[] txbuffer = (packetInfo+modeId[i]+"\r").getBytes();
        DatagramPacket p = new DatagramPacket(txbuffer,txbuffer.length,
        hostAddress,serverPort);
```

```
        s.send(p);
        r.receive(q);
        String message = new String(rxbuffer,0,q.getLength());
        switch(i){

        case 0:
            xstring = new String(rxbuffer,6,2);
            ystring = new String(rxbuffer,9,2);
            x1 = Integer.parseInt(xstring,16);
            y1 = Integer.parseInt(ystring,16);
            engineTime = 256*x1*y1/100; // Ta deytera einai oti deixnei *100
            teng.write(String.valueOf(engineTime)+",");

        case 1:
            xstring = new String(rxbuffer,6,2);
            x1 = Integer.parseInt(xstring,16);
            intakeAirTemp = x1-40;
            tintake.write(String.valueOf(intakeAirTemp)+",");

        case 2:
            xstring = new String(rxbuffer,6,2);
            x1 = Integer.parseInt(xstring,16);
            throttle = x1*100/255;
            tthrotle.write(String.valueOf(throttle)+",");

        case 3:
            xstring = new String(rxbuffer,6,2);
            ystring = new String(rxbuffer,9,2);
            x1 = Integer.parseInt(xstring,16);
            y1 = Integer.parseInt(ystring,16);
            rpm = ((x1*256)+y1)/4;
            trpm.write(String.valueOf(rpm)+",");

        case 4:
            xstring = new String(rxbuffer,6,2);
            x1 = Integer.parseInt(xstring,16);
            speed = x1;
            tspeed.write(String.valueOf(speed)+",");

        case 5:
            xstring = new String(rxbuffer,6,2);
            x1 = Integer.parseInt(xstring,16);
            coolanttemp = x1-40;
            tcoolant.write(String.valueOf(coolanttemp)+",");
```

```
        }
        System.out.println("Engine Time is:"+ engineTime);
        System.out.println("Air intake is:"+ intakeAirTemp);
        System.out.println("Throttle is:"+ throttle);
        System.out.println("Rpm is"+rpm);
        System.out.println("Speed is:"+ speed);
        System.out.println("Coolant Temperature is:"+ coolanttemp);
        } catch (Exception x) {
        System.out.println(x);
        }
    }
        }
        s.close();
        r.close();
        teng.close();
        tintake.close();
        trpm.close();
        tcoolant.close();
        tthrotle.close();
        tspeed.close();
    }



}
```