Δίκτυα Υπολογιστών II
Αλέξανδρος Λαδάς
9405

Source

UserApplication.java

```java
package userApplication;

import java.io.IOException;
import java.util.Scanner;

public class UserApplication {

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

        System.out.println("Welcome");

        byte[] serverPublicAdress = {(byte)155, (byte)207, (byte)18, 
(byte)208};
        int clientListeningPort = 48039;
        int serverListeningPort = 38039;
        String echoRequestCode = "E3795";
        String imageRequestCode = "M1708 UDP=1024 CAM=";
        String soundRequestCode = "A9058 F400";
        String vehicleRequestCode = "V2617 OBD=01 ";

        try (Scanner myObj = new Scanner(System.in)) {
            System.out.println("Select application:");
            System.out.println("1: Echo");
            System.out.println("2: Temperature");
            System.out.println("3: Image");
            System.out.println("4: Sound");
            System.out.println("5: Helicopter");
            System.out.println("6: Vehicle");
            String select = myObj.nextLine();

            switch (select){

            case "1":
                System.out.println("Loop time:(in minutes)");
                float loopTime = Float.parseFloat(myObj.nextLine());
                System.out.println("Throughput sample:(8/16/32)");
                int throughputSample = Integer.parseInt(myObj.nextLine());
```

```
                    Echo Echo = new Echo();
                    Echo.echo(clientListeningPort, serverListeningPort, 
echoRequestCode, serverPublicAdress, loopTime, throughputSample);
                    break;

                case "2":
                    Temperature Temperature = new Temperature();
                    Temperature.temperature(clientListeningPort, 
serverListeningPort, echoRequestCode, serverPublicAdress);
                    break;

                case "3":
                    System.out.println("Select Cam:(FIX/PTZ)");
                    imageRequestCode += myObj.nextLine();
                    Image Image = new Image();
                    Image.image(clientListeningPort, serverListeningPort, 
imageRequestCode, serverPublicAdress, "image.jpg");
                    break;

                case "4":
                    System.out.println("Select aq:(true/false)");
                    boolean aq = Boolean.parseBoolean(myObj.nextLine());
                    if (aq) {
                        soundRequestCode = "A9058 AQ F400";
                    }
                    Sound Sound = new Sound();
                    Sound.sound(clientListeningPort, serverListeningPort, 
soundRequestCode, serverPublicAdress, 800, aq);
                    break;

                case "5":
                    IthakiCopter IthakiCopter = new IthakiCopter();
                    IthakiCopter.ithakiCopter();
                    break;

                case "6":
                    System.out.println("Loop time:(in minutes)");
                    float vehicleLoopTime2 = Float.parseFloat(myObj.nextLine());
                    Vehicle Vehicle = new Vehicle();
                    Vehicle.vehicle(clientListeningPort, serverListeningPort, 
vehicleRequestCode, serverPublicAdress, 2*vehicleLoopTime2);
                    break;
```

```
            default:

                System.out.println("Invalid Input");

            }
        }
    }
}
```

Echo.java

```
package userApplication;

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 {

    private DatagramSocket s;
    private DatagramSocket r;

    public void echo(int cp, int sp, String pi, byte[] h, float m, int ts){

        int clientPort = cp;
        int serverPort = sp;
        String packetInfo = pi;
        byte[] hostIP = h;
        int throughputSample = ts;

        Writer responseTimes = null;
        Writer movingAverage = null;
        Writer srtt = null;
        Writer sigma = null;
        Writer rto = null;

        try {

            responseTimes = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("G1.txt"), "utf-8"));
            movingAverage = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("G2.txt"), "utf-8"));
```

```
                   srtt = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("srtt.txt"), "utf-8"));
                   sigma = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("sigma.txt"), "utf-8"));
                 rto = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("rto.txt"),
"utf-8"));
                   responseTimes.write("Echo response times with request code " + pi + "(in
milliseconds)");
                 ((BufferedWriter) responseTimes).newLine();
                 ((BufferedWriter) responseTimes).newLine();
                 movingAverage.write("Echo moving average with request code " + pi + "(in
milliseconds, ");
                 ((BufferedWriter) movingAverage).newLine();
                 ((BufferedWriter) movingAverage).newLine();
                 srtt.write("srtt with request code " + pi );
                 ((BufferedWriter) srtt).newLine();
                 ((BufferedWriter) srtt).newLine();
                 sigma.write("sigma with request code " + pi );
                 ((BufferedWriter) sigma).newLine();
                 ((BufferedWriter) sigma).newLine();
                 rto.write("rto with request code " + pi );
                 ((BufferedWriter) rto).newLine();
                 ((BufferedWriter) rto).newLine();

                   s = new DatagramSocket();
                   byte[] txbuffer = packetInfo.getBytes();
                   InetAddress hostAddress = InetAddress.getByAddress(hostIP);
                   DatagramPacket p = new DatagramPacket(txbuffer, txbuffer.length, hostAddress,
serverPort);

                   r = new DatagramSocket(clientPort);
                   r.setSoTimeout(4000);
                   byte[] rxbuffer = new byte[2048];
                   DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);

                   long loopTime = (long) (m*60000);
                   long curTime = 0;
                   List<Long> packageTime=new ArrayList<Long>();
                   long throughputTime = 0;
                   long startOfLoop = System.currentTimeMillis();
                   long tempAverage = 0;
                   int i=1;

                   do {
                          try {
                                 while(throughputTime < 1000) {

                                 curTime = System.currentTimeMillis();
                                 s.send(p);
                                 r.receive(q);
                                 String message = new String(rxbuffer, 0, q.getLength());
                                 System.out.println(message);
                                 packageTime.add(System.currentTimeMillis() - curTime);

                                 responseTimes.write(packageTime.get(packageTime.size()-1) + "");
                               ((BufferedWriter) responseTimes).newLine();

                                 throughputTime += System.currentTimeMillis() - curTime;

                                 }
                                 throughputTime = 0;

                                 do{
                                        tempAverage += packageTime.get(packageTime.size()-i);
                                        i++;
                                        if (packageTime.size() < i) {
                                              break;
                                        }
                                 } while (tempAverage < throughputSample*1000);

                                 tempAverage /= i;
                                 movingAverage.write((tempAverage*0.256) + "");
                               ((BufferedWriter) movingAverage).newLine();
                                 tempAverage=0;
                                 i=1;

                          } catch (Exception x) {
                                 System.out.println(x);
                          }
                    }while (curTime <= (loopTime+startOfLoop));

                    double a = 0.2;
                    double b = 0.5;
                    double c = 1;
                    double d_srtt = 0;
                    double d_sigma = 0;
                    double d_rto = 0;
                    for (int j=0; j<packageTime.size(); j++) {
                          d_srtt = a*d_srtt + packageTime.get(j)*(1-a);
                          srtt.write(d_srtt + "");
                          ((BufferedWriter) srtt).newLine();
                          d_sigma = b*d_sigma + Math.abs(d_srtt-packageTime.get(j)*(1-b));
                          sigma.write(d_sigma + "");
                        ((BufferedWriter) sigma).newLine();
                          d_rto = d_srtt + d_sigma * c;
                          rto.write(d_rto + "");
                          ((BufferedWriter) rto).newLine();
                    }

                    ((BufferedWriter) responseTimes).newLine();
                  responseTimes.write("Number of packages recieved in " + m + " minutes: " +
packageTime.size());

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

```
                } finally {
                        try {
                                responseTimes.close();
                                movingAverage.close();
                                srtt.close();
                                sigma.close();
                                rto.close();
                        } catch (Exception ex) {}
                }

                System.out.println("files created!");
        }
}
```

Temperature.java

```
package userApplication;

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

public class Temperature {

        private DatagramSocket s;
        private DatagramSocket r;

        public void temperature(int cp, int sp, String pi, byte[] h){

                int clientPort = cp;
                int serverPort = sp;
                String packetInfo = pi;
                byte[] hostIP = h;

                Writer temperatures = null;

                try {

                        temperatures = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("temperatures.txt"), "utf-8"));

                        temperatures.write("temperatures with request code " + pi);
                    ((BufferedWriter) temperatures).newLine();
                    ((BufferedWriter) temperatures).newLine();
```

```
            s = new DatagramSocket();

            InetAddress hostAddress = InetAddress.getByAddress(hostIP);

            r = new DatagramSocket(clientPort);
            r.setSoTimeout(4000);
            byte[] rxbuffer = new byte[2048];
            DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);

            try {
                for (int i=0; i<10; i++) {
                    byte[] txbuffer = (packetInfo + " T0" + 
String.valueOf(i)).getBytes();
                    System.out.println((String)(packetInfo + " T0" + 
String.valueOf(i)));
                    DatagramPacket p = new DatagramPacket(txbuffer, txbuffer.length, 
hostAddress, serverPort);

                    s.send(p);
                    r.receive(q);
                    String message = new String(rxbuffer, 0, q.getLength());
                    System.out.println(message);

                    if(q.getLength() > 40) {
                        temperatures.write(new String(rxbuffer, 27, 21));
                        ((BufferedWriter) temperatures).newLine();
                    }
                }
                for (int i=10; i<99; i++) {
                    byte[] txbuffer = ((packetInfo + " T" + 
String.valueOf(i))).getBytes();
                    System.out.println(packetInfo + " T" + String.valueOf(i));
                    DatagramPacket p = new DatagramPacket(txbuffer, txbuffer.length, 
hostAddress, serverPort);

                    s.send(p);
                    r.receive(q);
                    String message = new String(rxbuffer, 0, q.getLength());
                    System.out.println(message);

                    if(q.getLength() > 40) {
                        temperatures.write(new String(rxbuffer, 27, 21));
                        ((BufferedWriter) temperatures).newLine();
                    }
                }
            } catch (Exception x) {
                System.out.println(x);
            }
        }catch (Exception x) {
            System.out.println(x);
        } finally {
            try {
                temperatures.close();
            } catch (Exception ex) {}
        }

        System.out.println("files created!");
    }
}
```

Image.java

```
package userApplication;

import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.ArrayList;
import javax.imageio.ImageIO;

public class Image {

        private DatagramSocket s;
        private DatagramSocket r;

        public void image(int cp, int sp, String pi, byte[] h, String f) throws IOException {

            int clientPort = cp;
            int serverPort = sp;
            String packetInfo = pi;
            byte[] hostIP = h;
            String filename = f;
            ArrayList<Byte> bytesList = new ArrayList<Byte>();

            try {
                s = new DatagramSocket();
                byte[] txbuffer = packetInfo.getBytes();
                InetAddress hostAddress = InetAddress.getByAddress(hostIP);
                DatagramPacket p = new DatagramPacket(txbuffer, txbuffer.length, hostAddress,
serverPort);

                r = new DatagramSocket(clientPort);
                r.setSoTimeout(2000);
                byte[] rxbuffer = new byte[1024];
                DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);
                s.send(p);

                for (;;) {
                    try {
                        r.receive(q);
                    for (int i=0; i<rxbuffer.length; i++) {
                        bytesList.add(rxbuffer[i]);
                        if ((bytesList.get(bytesList.size()-1) == (byte)0xD8) &&
(bytesList.get(bytesList.size()-2) == (byte)0xFF)) {
                            System.out.println("Let the image begin");
                            bytesList.clear();
                            bytesList.add((byte)0xFF);
                            bytesList.add((byte)0xD8);
                        }else if((bytesList.get(bytesList.size()-1) == (byte)0xD9)
&& (bytesList.get(bytesList.size()-2) == (byte)0xFF)) {
                            System.out.println("Let's get outa here");
                            break;
                        }
                    }
                    if((bytesList.get(bytesList.size()-1) == (byte)0xD9) &&
(bytesList.get(bytesList.size()-2) == (byte)0xFF)) {
                        System.out.println("One more step");
                        break;
                    }
                } catch (Exception x) {
                    System.out.println(x);
                    break;
                }
            }
        }catch (Exception x) {
            System.out.println(x);
        }

        byte[] arrayOfBytes = new byte[bytesList.size()];
        for (int i=0; i<bytesList.size(); i++) {
            arrayOfBytes[i] = bytesList.get(i);
        }

        ByteArrayInputStream bis = new ByteArrayInputStream(arrayOfBytes);
        BufferedImage bImage = ImageIO.read(bis);
        ImageIO.write(bImage, "jpg", new File(filename) );
        System.out.println(filename + " created");

    }
}
```

Sound.java

```
package userApplication;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
```

Δίκτυα Υπολογιστών ΙΙ
Αλέξανδρος Λαδάς
9405

```
package userApplication;

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

public class Sound {

	private DatagramSocket s;
	private DatagramSocket r;

	public void sound(int cp, int sp, String pi, byte[] h, int n, boolean a) throws IOException
{

		int clientPort = cp;
		int serverPort = sp;
		String packetInfo = pi;
		byte[] hostIP = h;
		int numOfPackets = n;
		boolean aq = a;
		int aqAddSize = 0;
		int qs = 8;
		if (aq) {
			aqAddSize = 4;
			qs = 16;
		}

		Writer differences = null;
		Writer samples = null;
		Writer means = null;
		Writer steps = null;
		AudioFormat audioFormat = new AudioFormat(8000, qs, 1, true, false);

		try {
			differences = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("differences.txt"), "utf-8"));
			((BufferedWriter) differences).newLine();
			samples = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("samples.txt"), "utf-8"));
			((BufferedWriter) samples).newLine();
			means = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("means.txt"), "utf-8"));
			((BufferedWriter) means).newLine();
			steps = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("steps.txt"), "utf-8"));
```

```
            ((BufferedWriter) steps).newLine();

            s = new DatagramSocket();
            byte[] txbuffer = packetInfo.getBytes();
            InetAddress hostAddress = InetAddress.getByAddress(hostIP);
            DatagramPacket p = new DatagramPacket(txbuffer, txbuffer.length, hostAddress, 
serverPort);

            r = new DatagramSocket(clientPort);
            r.setSoTimeout(4000);


            byte[] rxbuffer = new byte[128 + aqAddSize];
            DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);
            s.send(p);

            byte[] bytesArray = new byte[numOfPackets*rxbuffer.length*4];
            int s2 = 0;
            int s1 = 0;
            int counter = 0;


            for (int i=0; i<numOfPackets; i++) {
                try {
                    r.receive(q);
                    if (aq) {
                        int mean = (int) (rxbuffer[0] + Math.pow(2.0, 8.0) * 
rxbuffer[1]);
                        int step = (int) (rxbuffer[2] + Math.pow(2.0, 8.0) * 
rxbuffer[3]);
                        means.write(mean + "");
                        ((BufferedWriter) means).newLine();
                        steps.write(step + "");
                        ((BufferedWriter) steps).newLine();
                        for (int j=4; j<132; j++) {
                            int byteForCutting = rxbuffer[j];

                            int firstHalf = (int) (byteForCutting & 0x0000000F) 
- 8;
                            int secondHalf = (int) ((byteForCutting & 
0x000000F0)>>4) - 8;

                            differences.write(firstHalf + "");
                            ((BufferedWriter) differences).newLine();
                            differences.write(secondHalf + "");
                            ((BufferedWriter) differences).newLine();

                            s1 = secondHalf*step + mean;
                            samples.write(s1 + "");
                            ((BufferedWriter) samples).newLine();

                            bytesArray[counter] = (byte) (s1 & 0xFF);
                            counter++;
                            bytesArray[counter] = (byte) ((s1 >> 8) & 0xFF);
                            counter++;

                            s2 = firstHalf*step + mean;
                            samples.write(s2 + "");
                            ((BufferedWriter) samples).newLine();

                            bytesArray[counter] = (byte) (s2 & 0xFF);
                            counter++;
                            bytesArray[counter] = (byte) ((s2 >> 8) & 0xFF);
                            counter++;
                        }
               }else {
                        for (int j=0; j<128; j++) {
                            int byteForCutting = rxbuffer[j];
                            int firstHalf = (15 & byteForCutting) - 8;
                            int secondHalf = ((240 & byteForCutting) >> 4) - 8;

                            differences.write(firstHalf + "");
                            ((BufferedWriter) differences).newLine();
                            differences.write(secondHalf + "");
                            ((BufferedWriter) differences).newLine();

                            s1 = s2 + secondHalf;
                            bytesArray[counter] = (byte)s1;
                            counter++;
                            s2 = s1 + firstHalf;
                            bytesArray[counter] = (byte)s2;
                            counter++;

                            samples.write(s1 + "");
                            ((BufferedWriter) samples).newLine();
                            samples.write(s2 + "");
                            ((BufferedWriter) samples).newLine();
                        }
               }

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

       System.out.println(bytesArray.length);
       SourceDataLine sdl = AudioSystem.getSourceDataLine(audioFormat);
       sdl.open(audioFormat, numOfPackets*rxbuffer.length*2);
       sdl.start();
       sdl.write(bytesArray, 0, numOfPackets*rxbuffer.length*2);

       sdl.close();

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

```java
        }finally {
            try {
                samples.close();
                differences.close();
                means.close();
                steps.close();
            } catch (Exception ex) {}
        }
    }
}
```

IthakiCopter.java

```java
package userApplication;

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

public class IthakiCopter {

    private DatagramSocket r;

    public void ithakiCopter(){

        int clientPort = 48078;
        Writer LRMotor = null;
        Writer Altitude = null;

        try {
            LRMotor = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("LR.txt"), "utf-8"));
            Altitude = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("Altitude.txt"), "utf-8"));
            r = new DatagramSocket(clientPort);
            r.setSoTimeout(4000);
            byte[] rxbuffer = new byte[2048];
            DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);

            String message;
            String lrmotor;
            String altitude;

            for (int i=0; i<70; i++) {
                try {
                    r.receive(q);
                    message = new String(rxbuffer,0,q.getLength());
                    lrmotor = new String(rxbuffer,40,3);
                    altitude = new String(rxbuffer,64,3);
                    System.out.println(message);
                    LRMotor.write(lrmotor + "");
                  ((BufferedWriter) LRMotor).newLine();
                  Altitude.write(altitude + "");
                  ((BufferedWriter) Altitude).newLine();

                } catch (Exception x) {
                    System.out.println(x);
                }
            }
        }catch (Exception x) {
             System.out.println(x);
        } finally {
            try {
                LRMotor.close();
                Altitude.close();
            } catch (Exception ex) {}
        }
    }
}
```

Vehicle.java

```
package userApplication;

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

public class Vehicle {

    private DatagramSocket s;
    private DatagramSocket r;

    public void vehicle(int cp, int sp, String pi, byte[] h, float m){

        int clientPort = cp;
        int serverPort = sp;
        String packetInfo = pi;
        byte[] hostIP = h;
        String message;
        String[] pid = {"1F", "0F", "11", "0C", "0C", "05"};

        long loopTime = (long) (m*60000);
        long startOfLoop = System.currentTimeMillis();
        long curTime = 0;
        Writer engineRunTime = null;
        Writer intakeAirTemp = null;
        Writer throttlePosition = null;
        Writer engineRPM = null;
        Writer vehicleSpeed = null;
        Writer coolantTemperature = null;


        try {
            engineRunTime = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("engineRunTime.txt"), "utf-8"));
            engineRunTime.write("engineRunTime with request code " + pi);
            intakeAirTemp = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("intakeAirTemp.txt"), "utf-8"));
            intakeAirTemp.write("intakeAirTemp with request code " + pi);
            throttlePosition = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("throttlePosition.txt"), "utf-8"));
            throttlePosition.write("throttlePosition with request code " + pi);
            engineRPM = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("engineRPM.txt"), "utf-8"));
            engineRPM.write("engineRPM with request code " + pi);
            vehicleSpeed = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("vehicleSpeed.txt"), "utf-8"));
            vehicleSpeed.write("vehicleSpeed with request code " + pi);
            coolantTemperature = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream("coolantTemperature.txt"), "utf-8"));
            coolantTemperature.write("coolantTemperature with request code " + pi);

            s = new DatagramSocket();
            r = new DatagramSocket(clientPort);
            r.setSoTimeout(10000);
            byte[] rxbuffer = new byte[2048];
            DatagramPacket q = new DatagramPacket(rxbuffer,rxbuffer.length);
            InetAddress hostAddress = InetAddress.getByAddress(hostIP);
            do {
                curTime = System.currentTimeMillis();
                for (int i = 0; i<6; i++) {
                    byte[] txbuffer = (packetInfo+pid[i]).getBytes();

                    DatagramPacket p = new DatagramPacket(txbuffer, txbuffer.length,
hostAddress, serverPort);
                    s.send(p);
                    r.receive(q);
                    message = new String(rxbuffer,0,q.getLength());
                    System.out.println(message);

                    switch(i) {
                        case 0:
                            String xx0 = new String(rxbuffer,6,2);
                            String yy0 = new String(rxbuffer,9,2);
                            int ixx0 = Integer.parseInt(xx0, 16);
                            int iyy0 = Integer.parseInt(yy0, 16);
                          ((BufferedWriter) engineRunTime).newLine();
```

```
                  engineRunTime.write(256*ixx0+iyy0 + "");
                  break;

                case 1:
                    String xx1 = new String(rxbuffer,6,2);
                    int ixx1 = Integer.parseInt(xx1, 16);
                    ((BufferedWriter) intakeAirTemp).newLine();
                    intakeAirTemp.write((ixx1-40) + "");

                    break;

                case 2:
                    String xx2 = new String(rxbuffer,6,2);
                    int ixx2 = Integer.parseInt(xx2, 16);
                    ((BufferedWriter) throttlePosition).newLine();
                    throttlePosition.write((ixx2*100/255) + "");
                    break;

                case 3:
                    String xx3 = new String(rxbuffer,6,2);
                    String yy3 = new String(rxbuffer,9,2);
                    int ixx3 = Integer.parseInt(xx3, 16);
                    int iyy3 = Integer.parseInt(yy3, 16);
                  ((BufferedWriter) engineRPM).newLine();
                  engineRPM.write(((ixx3*256)+iyy3)/4 + "");
                  break;

                case 4:
                    String xx4 = new String(rxbuffer,6,2);
                    int ixx4 = Integer.parseInt(xx4, 16);
                    ((BufferedWriter) vehicleSpeed).newLine();
                    vehicleSpeed.write(ixx4 + "");
                    break;

                case 5:
                    String xx5 = new String(rxbuffer,6,2);
                    int ixx5 = Integer.parseInt(xx5, 16);
                    ((BufferedWriter) coolantTemperature).newLine();
                    coolantTemperature.write((ixx5-40) + "");
                    break;
            }
        }
    }while (curTime <= (loopTime+startOfLoop));
}catch (Exception x) {
    System.out.println(x);
} finally {
    try {
        engineRunTime.close();
        intakeAirTemp.close();
        throttlePosition.close();
        engineRPM.close();
        vehicleSpeed.close();
        coolantTemperature.close();
```

```
            } catch (Exception ex) {}
        }
    }
}
```