Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Πολυτεχνική Σχολή

Δίκτυα Υπολογιστών ΙΙ

Θεόδωρος Κατζάλης
ΑΕΜ:9282
katzalis@auth.gr

December 3, 2020

Contents

1 Δομή του προγράμματος

```
applications
    Copter.java
    Echo.java
    Media.java
    Obd.java
plots
    plot.py
stamps
    audio.txt
    auto.txt
    copter_tcp.txt
    copter.txt
    echo.txt
    https.txt
    image.txt
    obd_tcp.txt
    obd.txt
    temp.txt
    test.txt
    welcome.txt
UserApplication.java
```

- Το αρχείο που βρίσκεται η main είναι το *UserApplication.java*.
- Στον φάκελο *applications*, δημιουργήσαμε ξεχωριστά αρχεία για κάθε εφαρμογή.
- Στον φάκελο *stamps*, βρίσκονται οι έξοδοι του προγράμματος figlet για ascii art λόγους, όπως έχει αναφερθεί και στο report!
- Στον φάκελο *plots*, έχουμε τέλος ένα python αρχείο για να δημιουργήσουμε τα διαγράμματα μας.

2 UserApplication.java

```java
// it is considered in general a bad practise to use asterisks to import all the classes
import java.io.*;
import java.net.*;
import java.nio.charset.StandardCharsets;
import java.lang.System;
import java.awt.Desktop;
import javax.sound.sampled.*;
import java.lang.Math.*;
import java.util.Arrays;
import java.util.Scanner;
import java.time.LocalDateTime;

import applications.*;

class UserApplication {

    // TODO create a script that is scraping from ithaki website the request code and the ports.

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


        // windows users may be not able to view colors on terminal
        final String ANSI_CYAN = "\u001B[36m";
        final String ANSI_RESET = "\u001B[0m";

        // print welcome text
        Scanner input = new Scanner(new File("stamps/welcome.txt"));
        while (input.hasNextLine())
        {
            System.out.print(ANSI_CYAN); // add some color!
            System.out.print(input.nextLine());
            System.out.println(ANSI_RESET);
        }
        System.out.println();
        System.out.print("Press ENTER to continue");
        System.in.read();// pause a little bit to enjoy the view


         // preamble
        byte[] clientIP = { (byte)192, (byte)168,  (byte)1, (byte)20};
        byte[] hostIP = { (byte)155, (byte)207,  (byte)18, (byte)208};
        InetAddress clientAddress = InetAddress.getByAddress(clientIP);
        InetAddress hostAddress = InetAddress.getByAddress(hostIP);
        int serverPort = 38022;
        int clientPort = 48022;
        String requestCodeEcho = "E0818 ";
        String requestCodeImage = "M2685UDP=1024";
        String requestCodeSound = "A7269";
        String requestCodeCopter = "Q2797";
        String requestCodeVehicle = "V4118";

        DatagramSocket socket = new DatagramSocket(clientPort);
        long timeBefore = 0;


        int flag =1; // control user input
        do {
```

```
58      System.out.println("\nPlease enter a number (1-11). Available options are:\n1) Echo
    with delay\n2) Echo no delay\n3) Temperature\n4) Image\n5) Music\n6) Vehicle UDP\n7)
    Ithakicopter UDP\n8) Autopilot\n9) HTTPS TCP\n10) Ithakicopter TCP\n11) Vehicle TCP");
59      String choiceApp = (new Scanner(System.in)).nextLine();
60
61      //checkArguements(args); // check the validity of command line arguement
62
63
64      switch (choiceApp) {
65      case "1":
66
67      /* -------------------------- Echo -------------------------- */
68      input = new Scanner(new File("stamps/echo.txt"));
69      while (input.hasNextLine())
70      {
71          System.out.println(input.nextLine());
72      }
73      Thread.sleep(1500); // pause a little bit to enjoy the view
74
75      File info = new File("logs/echo_info_delay.txt");
76      FileWriter writerInfo = new FileWriter(info);
77      writerInfo.write("Info:\n" + "The request code is " + requestCodeEcho + "\n");
78      writerInfo.write("Tic: " + LocalDateTime.now() + "\n");
79
80      File fileSamples = new File("logs/echo_samples_delay.txt");
81      FileWriter writerSamples = new FileWriter(fileSamples);
82
83      File fileThroughput = new File("logs/echo_throughput_delay.txt");
84      FileWriter writerThroughput = new FileWriter(fileThroughput);
85
86      File fileRto = new File("logs/rto.txt");
87      FileWriter writerRto = new FileWriter(fileRto);
88
89      timeBefore = System.currentTimeMillis();
90      long tic[] = new long[8];
91      //long toc[] = new long[8];
92      int cumsum[] = new int[8];  // cumulative sum
93      float throughput[] = new float[8];  // worst case we need 8 elements to store data
94      int count8sec[] = new int[8]; // keep track how many 8 seconds have passed
95      for(int i = 0; i<8; i++) {
96          tic[i] = timeBefore + i*1000; // move per second
97      }
98
99      double rtts = 0;
100     double rttd = 0;
101     double rto = 0;
102     int isFirst = 1;
103
104     while ((System.currentTimeMillis() - timeBefore) < 60000 * 4){
105         long rtt = Echo.execute(socket, hostAddress, serverPort, requestCodeEcho);
106         writerSamples.write(rtt + "\n");
107
108         // throughput moving average
109
110          for(int i = 0; i<8; i++) {
111              //System.out.println("The element " + i + " has tic: " + tic[i]);
112              long toc = System.currentTimeMillis() - tic[i];
113              System.out.println("The element " + i + " has toc: " + toc);
114              if (toc < 8000 && toc > 0) {
115                  cumsum[i] += 32*8; // assume no timeouts during the measurements
```

```
                    System.out.println("Cumsum: " + cumsum[i]);
                }
                else if(toc > 8000){
                    count8sec[i]++;
                    tic[i] = count8sec[i]*8000 + timeBefore + i*1000;
                    throughput[i] = cumsum[i]/(float)8;
                    System.out.println("I will flush " + cumsum[i] + " cumsum");
                    System.out.println("The throughput is: " + throughput[i]);

                    writerThroughput.write(throughput[i]+ "\n");


                    cumsum[i] = 0; // let's start again for the next 8 seconds
                }
            }


            // Retransmission timeout

            // init values
            if (isFirst == 1) {
                rtts = rtt;
                rttd = rtt/2;
                rto = 1; // according to rfc
                writerRto.write("RTT SRTT RTTd RTO\n");
            }
            double temp = rtts;
            rtts = 0.875*temp + 0.125*rtt;

            temp = rttd;
            rttd = 0.75*temp + 0.25*Math.abs(rtt - rtts);

            rto = rtts + 1.8*rttd;

            writerRto.write(rtt + " " + rtts + " " + rttd + " " + rto + "\n");

            System.out.println();
            isFirst = 0;
        }

        writerInfo.write("Toc: " + LocalDateTime.now());

        writerInfo.close();
        writerSamples.close();
        writerThroughput.close();
        writerRto.close();
        socket.close();
        break;

        case "2":
        input = new Scanner(new File("stamps/echo.txt"));
        while (input.hasNextLine())
        {
            System.out.println(input.nextLine());
        }
        Thread.sleep(1500); // pause a little bit to enjoy the view

        info = new File("logs/echo_info_no_delay.txt");
        writerInfo = new FileWriter(info);
        writerInfo.write("Info:\n" + "The request code is " + requestCodeEcho + "\n");
```

```
        writerInfo.write("Tic: " + LocalDateTime.now() + "\n");

        fileSamples = new File("logs/echo_samples_no_delay.txt");
        writerSamples = new FileWriter(fileSamples);

        fileThroughput = new File("logs/echo_throughput_no_delay.txt");
        writerThroughput = new FileWriter(fileThroughput);

        timeBefore = System.currentTimeMillis();
        tic = new long[8];
        //long toc[] = new long[8];
        cumsum = new int[8];  // cumulative sum
        throughput = new float[8];  // worst case we need 8 elements to store data
        count8sec = new int[8];
        for(int i = 0; i<8; i++) {
            tic[i] = timeBefore + i*1000; // move per second
        }

        while ((System.currentTimeMillis() - timeBefore) < 60000 * 4){
            long value = Echo.execute(socket, hostAddress, serverPort, "E0000");
            writerSamples.write(value + "\n");

             // throughput moving average

             for(int i = 0; i<8; i++) {
                 //System.out.println("The element " + i + " has tic: " + tic[i]);
                 long toc = System.currentTimeMillis() - tic[i];
                 System.out.println("The element " + i + " has toc: " + toc);
                 if (toc < 8000 && toc > 0) {
                     cumsum[i] += 32*8; // assume no timeouts during the measurements
                     System.out.println("Cumsum: " + cumsum[i]);
                 }
                 else if(toc > 8000){
                     count8sec[i]++;
                     tic[i] = count8sec[i]*8000 + timeBefore + i*1000;
                     throughput[i] = cumsum[i]/(float)8;
                     System.out.println("I will flush " + cumsum[i] + " cumsum");
                     System.out.println("The throughput is: " + throughput[i]);

                     writerThroughput.write(throughput[i]+ "\n");

                     cumsum[i] = 0; // let's start again for the next 8 seconds
                 }
             }
             System.out.println();
        }

        writerInfo.write("Toc: " + LocalDateTime.now());

        writerInfo.close();
        writerSamples.close();
        writerThroughput.close();
        socket.close();
        break;



        case "3":
        /* ---------------------------- Temperature ---------------------------- */
        input = new Scanner(new File("stamps/temp.txt"));
```

```
236         while (input.hasNextLine())
237         {
238             System.out.println(input.nextLine());
239         }
240         Thread.sleep(1500); // pause a little bit to enjoy the view
241 
242         FileWriter writerTemp = new FileWriter(new File("logs/temp_info.txt")) ;
243         writerTemp.write("Info Temperature app:\n" + requestCodeEcho + "\n" + LocalDateTime.
    now() + "\n");
244 
245         for (int i = 0; i < 1; i++) {
246              Echo.execute(socket, hostAddress, serverPort, requestCodeEcho + "T00");
247              System.out.println();
248         }
249 
250         writerTemp.write(LocalDateTime.now() + "\n");
251         writerTemp.close();
252         socket.close();
253         break;
254 
255         case "4":
256         /* --------------------------- Image --------------------------- */
257         input = new Scanner(new File("stamps/image.txt"));
258         while (input.hasNextLine())
259         {
260             System.out.println(input.nextLine());
261         }
262         Thread.sleep(1500); // pause a little bit to enjoy the view
263 
264         //for (int i = 0; i<4; i++) Echo.execute(socket, hostAddress, serverPort,
    requestCodeEcho);
265 
266         String encodingImage = "CAM=PTZDIR=R";
267         FileWriter writerImage = new FileWriter(new File("logs/image_info_" + encodingImage));
268         writerImage.write(encodingImage + "\n" + requestCodeImage + "\n" + LocalDateTime.now()
     + "\n");
269         for (int i = 0; i < 1; i++) {
270              Media.image(socket, hostAddress, serverPort, requestCodeImage + encodingImage);
271              System.out.println();
272         }
273         writerImage.write(LocalDateTime.now() + "\n");
274         writerImage.close();
275         socket.close();
276         break;
277 
278         case "5":
279         /* --------------------------- Audio --------------------------- */
280         input = new Scanner(new File("stamps/audio.txt"));
281         while (input.hasNextLine())
282         {
283             System.out.println(input.nextLine());
284         }
285         Thread.sleep(1500);
286 
287         //for (int i = 0; i<4; i++) Echo.execute(socket, hostAddress, serverPort,
    requestCodeEcho);
288 
289 
290         String numAudioPackets = "999";
291         String[] type = {"F", "T"};
```

```
292         String[] encoding = {"AQ", ""};
293         // assuming random choice of track
294         String completeRequest = requestCodeSound + encoding[0] + type[0] + numAudioPackets;
295 
296         File infoMusic = new File("logs/music_info_" + encoding[1] + type[0] + ".txt");
297         FileWriter writerInfoMusic = new FileWriter(infoMusic);
298         writerInfoMusic.write(requestCodeSound + "\nEncoding: " + encoding[1] + "\nType: " +
    type[0] + LocalDateTime.now() + "\n");
299 
300         Media.audio(socket, hostAddress, serverPort, completeRequest);
301         System.out.println();
302 
303         writerInfoMusic.write(LocalDateTime.now() + "\n");
304         writerInfoMusic.close();
305         socket.close();
306         break;
307 
308         case "6":
309         /* --------------------------- Vehicle OBD UDP--------------------------- */
310         input = new Scanner(new File("stamps/obd.txt"));
311         while (input.hasNextLine())
312         {
313             System.out.println(input.nextLine());
314         }
315         Thread.sleep(1500); // pause a little bit to enjoy the view
316 
317         Obd.udpTelemetry(socket, hostAddress, serverPort, requestCodeVehicle);
318         socket.close();
319         break;
320 
321         case "7":
322         /* --------------------------- Ithakicopter UDP--------------------------- */
323         socket = new DatagramSocket(48078);
324         input = new Scanner(new File("stamps/copter.txt"));
325         while (input.hasNextLine())
326         {
327             System.out.println(input.nextLine());
328         }
329         Thread.sleep(1500);
330 
331         System.out.println("For Ithakicopter UDP telemetry you need to open ithakicopter.jar")
    ;
332         System.out.print("Did you open it? If yes press ENTER to continue");
333         System.in.read();
334         Thread.sleep(1000); // pause a bit to catch up with the user
335         System.out.println("Press ENTER to exit");
336         Thread.sleep(1000);
337 
338         FileWriter writerCopter = new FileWriter(new File("logs/copter_info.txt")) ;
339         writerCopter.write("Info Ithakicopter app:\n" + LocalDateTime.now() + "\n");
340         writerCopter.write("MOTOR ALTITUDE TEMPERATURE PRESSURE");
341 
342 
343         for (int i = 0; i<4; i++) Echo.execute(socket, hostAddress, serverPort,
    requestCodeEcho);
344 
345         while (System.in.available() == 0) {
346             Copter.udpTelemetry(socket, hostAddress, serverPort, writerCopter);
347 
348         }
```

```
        writerCopter.write(LocalDateTime.now() + "\n");
        writerCopter.close();
        socket.close();
        break;

        case "8":
        /* --------------------------- Autopilot --------------------------- */
        socket = new DatagramSocket(48078);
        Socket socketAuto = new Socket(hostAddress, 38048);
        input = new Scanner(new File("stamps/auto.txt"));
        while (input.hasNextLine())
        {
            System.out.println(input.nextLine());
        }
        Thread.sleep(1500); // pause a little bit to enjoy the view

        int lowerBound = 160;
        int higherBound = 190;
        Copter.autopilot(socket, hostAddress, serverPort, socketAuto, Math.min(200, Math.max
(150, lowerBound)), Math.min(200, Math.max(150, higherBound)));
        socketAuto.close();
        break;



        case "9":
        /* --------------------------- HTTPS TCP--------------------------- */
        Socket httpsSocket = new Socket(hostAddress, 80);
        input = new Scanner(new File("stamps/https.txt"));
        while (input.hasNextLine())
        {
            System.out.println(input.nextLine());
        }
        Thread.sleep(1500); // pause a little bit to enjoy the view

        https(httpsSocket);
        httpsSocket.close();
        break;

        case "10":
        /* --------------------------- Ithakicopter TCP--------------------------- */
        // can we use 38098? If we open the website we see that this is the port opened
        //Socket socketCopter = new Socket(hostAddress, 38048);
        input = new Scanner(new File("stamps/copter_tcp.txt"));
        while (input.hasNextLine())
        {
            System.out.println(input.nextLine());
        }
        Thread.sleep(1500); // pause a little bit to enjoy the view

        for (int i = 0; i<4; i++) Echo.execute(socket, hostAddress, serverPort,
requestCodeEcho);

        int target = 180;
        for (int i = 0; i<20; i++) {
        System.out.println(new String(Copter.tcpTelemetry(hostAddress, target)));
        }

        //socketCopter.close();
```

```
        break;

        case "11":
        /* -------------------------- Vehicle OBD TCP-------------------------- */
        input = new Scanner(new File("stamps/obd_tcp.txt"));
        while (input.hasNextLine())
        {
            System.out.println(input.nextLine());
        }
        Thread.sleep(1500); // pause a little bit to enjoy the view

        for (int i = 0; i<4; i++) Echo.execute(socket, hostAddress, serverPort,
    requestCodeEcho);

        Socket socketVehicle = new Socket(hostAddress, 29078);

        FileWriter writerVehicleInfo = new FileWriter(new File("logs/car_info.txt")) ;
        writerVehicleInfo.write("Info Vehicle app:\n" + LocalDateTime.now() + "\n");
        FileWriter writerVehicleData = new FileWriter(new File("logs/car_telemetry.txt"));

        timeBefore = System.currentTimeMillis();
        float engineTime = 0;
        while ( engineTime < 60 * 4){
            engineTime = Obd.tcpTelemetry(socketVehicle, writerVehicleData);
            System.out.println("The engine run time is " + engineTime + "\n");
        }

        writerVehicleInfo.write(LocalDateTime.now() + "\n");
        writerVehicleInfo.close();
        writerVehicleData.close();
        socketVehicle.close();
        break;



        /*
         * ************************************************************************
         *            This is a playgorund!! Test whatever you like....
         * ************************************************************************
         *
         */
        case "12":
        Socket foo = new Socket(hostAddress, 38048);
        OutputStream out = foo.getOutputStream();
        InputStream in = foo.getInputStream();
        InputStreamReader isr = new InputStreamReader(in);
        BufferedReader bf = new BufferedReader(isr);
        ByteArrayOutputStream bis = new ByteArrayOutputStream();

        // if use readAllBytes the InputStream is closed
        // actually readNBytes is quite weird to be honest
        out.write("AUTO FLIGHTLEVEL=100 LMOTOR=100 RMOTOR=100 PILOT \r\n".getBytes());
        String data = new String();
        while ((data = bf.readLine()) != null) {
            bis.write((data + "\n").getBytes());
        }
        data = new String(bis.toByteArray(), StandardCharsets.US_ASCII);
        System.out.println(data);
        break;
```

```
466         case "13":
467 
468         // Major difference between this code snippet and the above is the reposnse time!
    Actually it is all about null and ending stream!
469         Socket foo1 = new Socket(hostAddress, 38048);
470         OutputStream out1 = foo1.getOutputStream();
471         InputStream in1 = foo1.getInputStream();
472         InputStreamReader isr1 = new InputStreamReader(in1);
473         BufferedReader bf1 = new BufferedReader(isr1);
474 
475         // if use readAllBytes the InputStream is closed
476         // actually readNBytes is quite weird to be honest
477         for (int i = 0; i < 4; i++) {
478         out1.write("AUTO FLIGHTLEVEL=100 LMOTOR=100 RMOTOR=100 PILOT \r\n".getBytes());
479         //String data1 = new String(in1.readNBytes(427), StandardCharsets.US_ASCII);
480         //System.out.println(data1);
481         //System.out.println(bf1.readLine());
482         //System.out.println(bf1.readLine());
483 
484         //for (int i = 0; i < 40; i++) {
485         //    System.out.println(bf1.readLine());
486         //}
487         // In general we have a lag when reading if we are in the end of the stream
488 
489         //while(bf1.readLine() != null) {System.out.println(bf1.readLine());} // warning bf
    read is called two times
490         // waiting to encounter null isn't good. Too much lag. Waiting if the stream is closed
     or not
491 
492         // why we do this? Ithaki when establishing this connection first send some
    introductory info and then the actual telemetry. So you need to handle streams in a proper
    way!
493         if (i==0) {
494             for (int l = 0; l < 14; l++) { // after some tinkering we have the data
495                 System.out.println(bf1.readLine());
496             }
497         }
498         else {
499             System.out.println(bf1.readLine());
500         }
501         }
502 
503 
504         break;
505         /*
506          * ************************************************************************
507          *                              End of playground
508          * ************************************************************************
509          *
510          */
511 
512 
513 
514 
515         default:
516          System.out.println("Please provide a valid input. If you want to exit then press Control-C
    .\n");
517          flag = 0;
518 
519             } // end switch
```

```
        } while(flag == 0);


        /* ---------------------------- Close UDP sockets ---------------------------- */
        if (!socket.isClosed()) {
            socket.close();
            System.out.println("\nShuting down UDP sockets...");
        }


        System.out.println("\nx--------------------Hooray! Java application finished
    successfully!--------------------x");

    }


    private static void https(Socket socket) {
        try {
            InputStream in = socket.getInputStream(); // what I receive from the server
            OutputStream out = socket.getOutputStream(); // what i send to the server

            long timeBefore = System.currentTimeMillis();
            out.write("GET /netlab/hello.html HTTP/1.0\r\nHost: ithaki.eng.auth.gr:80\r\n\r\n"
    .getBytes());

            byte[] inputBuffer = in.readAllBytes();
            String message = new String(inputBuffer, StandardCharsets.US_ASCII);
            System.out.println("Ithaki responded via TCP with: \n" + message);
            System.out.println("Time response: " + (System.currentTimeMillis() - timeBefore)/(
    float)1000 + " seconds");

            socket.close();
        }
        catch (Exception x) {
            System.out.println(x + "TCP application failed");
        }

    }

    private static void checkArguements (String[] args) {
        if (args.length == 2) {
            System.out.println("Command line arguements: This is the first " + args[0] + " " +
     "and this is the second " + args[1]);
        }
        String[] directionOptions = {"L", "D", "U", "R"};
        int flag = 0;
        for (String i : directionOptions) {
            if (i.equals(args[0])) {
                flag = 1;
                System.out.println("Direction: " + args[0]);
                break;
            }
        }
        if (flag == 0) {
            System.out.println("Try again, wrong direction. Available options are: L, R, U, D"
    );
            return;
        }
        if (Integer.parseInt(args[1])>=1 && Integer.parseInt(args[1])<=100) {
            System.out.println("How many times to repeat the movement of the camera on that
```

```
    direction? " + Integer.parseInt(args[1]));
575         }
576         else {
577             System.out.println("Invalid input. Range should be 1-100");
578             return;
579         }
580     }
581 
582     }
```

3 applications

3.1 Echo.java

```
1  package applications;
2  
3  import java.net.DatagramSocket;
4  import java.net.DatagramPacket;
5  import java.net.InetAddress;
6  import java.nio.charset.StandardCharsets;
7  
8  public class Echo {
9  
10     /*
11      * UDP TX/RX Echo application with delay
12      *
13      * WARNING: It doesn't close the DatagramSocket. You should do it manually if it is
    desired after the call of the function.
14      *
15      * @param requestCode If request code is set to E000 then the execute will have no delay
    for the RX
16      */
17     public static long execute(DatagramSocket socket, InetAddress hostAddress, int serverPort,
     String requestCode) {
18         System.out.println("\n--------------------Echo application--------------------");
19 
20         if (requestCode.equals("E0000")) System.out.println("Delay: OFF");
21         else if (requestCode.length()>5) System.out.println("Mode: Temperature\nDelay: OFF");
22         else System.out.println("Delay: OFF");
23 
24         byte[] txbuffer = requestCode.getBytes();
25         byte[] rxbuffer = new byte[64];
26         long diff = 0;
27         try {
28             socket.setSoTimeout(3000);
29             DatagramPacket sendPacket = new DatagramPacket(txbuffer, txbuffer.length,
    hostAddress, serverPort);
30             DatagramPacket receivePacket= new DatagramPacket(rxbuffer, rxbuffer.length);
31 
32             // ACTION
33             socket.send(sendPacket);
34             System.out.println("The request code is: "+ requestCode + "\nThe destination port
    is: " + serverPort + "\nMy listening port (clientPort): " + socket.getLocalPort());
35             long timeBefore = System.currentTimeMillis();
36             System.out.println("My system time, when the request is sent, is: " + timeBefore);
37 
38             // LISTEN
39             socket.receive(receivePacket);
40             long timeAfter = System.currentTimeMillis();
41             diff = timeAfter - timeBefore;
```

```
            System.out.println("The time required to reveive a packet is: " + diff + "
    milliseconds");
            //System.out.println("The port that opened ithaki to send the request is : " +
    receivePacket.getPort() + " and the address of ithaki is: " + receivePacket.getAddress());
            String message = new String(receivePacket.getData(), StandardCharsets.US_ASCII);
    // convert binary to ASCI
            System.out.println("Ithaki responded with: " + message);
        }
        catch (Exception x) {
            // x.printStackTrace(); // a more detailed diagnostic call
            System.out.println(x);
            System.out.println("Something went wrong about Echo application mode");
        }
        return diff;
    }
}
```

3.2 Media.java

```
package applications;

import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.InetAddress;
import java.io.ByteArrayOutputStream;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileWriter;
import java.io.FileOutputStream;
import java.io.IOException;
import java.awt.Desktop;
import java.util.Arrays;
import javax.sound.sampled.*;

public class Media {

    private static String pathFileImage = "/home/tkatz/repos/ece-networks2/media/image/sandbox
    /ithaki_image.jpg";
    private static String pathFileSound = "/home/tkatz/repos/ece-networks2/media/music/sandbox
    /track.wav";

    public static void image(DatagramSocket socket, InetAddress hostAddress, int serverPort,
    String requestCode) throws IOException {

        //if (numImage != (Integer.parseInt(args[1])-1)){
        //    continue; // readjust the camera as many time is requested via the command line
    arguement. Print only the result, the last request
        //}

        byte[] txbufferImage = requestCode.getBytes();
        byte[] rxbufferImage = new byte[1024];
        int countPackets = 0;
        long timeBefore = System.currentTimeMillis();
        long timeBeforePerPacket = System.currentTimeMillis();
        //System.out.println("My system time, when the request is sent, is: " + timeBefore);

        System.out.println("The request code is " + requestCode);
        DatagramPacket sendPacket = new DatagramPacket(txbufferImage, txbufferImage.length,
    hostAddress, serverPort);
        DatagramPacket receivePacket= new DatagramPacket(rxbufferImage, rxbufferImage.length);

```

```
        // TX
        System.out.println("I am sleeping... Camera needs time to readjust");
        try {
            socket.send(sendPacket);
            Thread.sleep(5000); // sleep in order for the camera to readjust
        }
        catch (Exception x) {
            // x.printStackTrace(); // a more detailed diagnostic call
            System.out.println(x);
            System.out.println("Image application TX failed");
        }

        // RX
        ByteArrayOutputStream bufferImage = new ByteArrayOutputStream();
outerloop:
        try {
            socket.setSoTimeout(3000);
            for (;;) {
                socket.receive(receivePacket); // blocking command
                countPackets++;

                long timeAfterPerPacket = System.currentTimeMillis();
                System.out.println("The time required to reveive a packet is: " + (timeAfterPerPacket - timeBeforePerPacket)/(float)1000 + " seconds");
                timeBeforePerPacket = System.currentTimeMillis();

                System.out.println("Packet No" + countPackets + ". Length of data: " + rxbufferImage.length + ". The received bytes in hexadecimal format are:");
                for (int i = 0; i<rxbufferImage.length; i++) {
                    System.out.print(String.format("%02X", rxbufferImage[i])); // convert bytes to hexa string

                    bufferImage.write(rxbufferImage[i]); // dynamic byte allocation
                    if ((String.format("%02X", rxbufferImage[i]).equals("D9")) && (i!=0)) {
                        if ((String.format("%02X", rxbufferImage[i-1]).equals("FF"))) {
                            break outerloop; // stop writing when EOF (0xFFD9 delimiter)
                        }
                    }
                }
                System.out.println();
            }
        }
        catch (Exception x) {
            // x.printStackTrace(); // a more detailed diagnostic call
            System.out.println(x);
            System.out.println("Image application RX failed");
        }
        long timeAfter = System.currentTimeMillis(); // get the time when the image is received in bytes

        // logs for the received byte content
        System.out.println("\nComplete byte content of the image file in hexadecimal format:");
        byte[] completeDataImage = bufferImage.toByteArray();
        for (byte i : completeDataImage) {
            System.out.print(String.format("%02X", i)); // print hexadecimal the content of the byte array
        }
        System.out.println("\n\nTotal number of packages: " + (countPackets));
        System.out.println("How many Kbytes is the image? " + completeDataImage.length/(float)
```

```
1000);

        // save image to a file
        File imageFile = new File(pathFileImage);
        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream(imageFile);
            fos.write(completeDataImage);
            System.out.println("File has been written successfully");
        }
        catch (Exception x) {
            // x.printStackTrace(); //
            System.out.println("Image application error when writing the file:");
        }

        fos.close(); // close the OutputStream

        // what time is o'clock?
        System.out.println("Total amount of time to receive a frame: " + (timeAfter -
timeBefore)/(float)1000 + " seconds");
        timeAfter = System.currentTimeMillis(); // get the time when the file is ready
        System.out.println("Total amount of time to receive and write a frame in a .jpg file:
" + (timeAfter - timeBefore)/(float)1000 + " seconds");

        // open file image
        Desktop desktop = Desktop.getDesktop();
        if (imageFile.exists()) {
            //desktop.open(imageFile);
        }
    }

    public static void audio(DatagramSocket socket, InetAddress hostAddress, int serverPort,
String requestCode) {

        // parsing the requestCode
        // expecting requestCode: AXXXX + ("AQ"" or "") + ("T" or "F") + numAudioPackets
        String encoding = "";
        String type = "F";
        String numAudioPackets = "";
        if (requestCode.length() == 11) {
            encoding = "AQ";
            type = requestCode.substring(7, 8);
            numAudioPackets = requestCode.substring(8, 11);
        }
        else {
            type = requestCode.substring(5, 6);
            numAudioPackets = requestCode.substring(6, 9);
        }
        System.out.println("Requested: Encoding: " + encoding + ". Type: " + type + ". Number
of packets: " + numAudioPackets);

        // TX
        byte[] txbufferSound = ("L02" + requestCode).getBytes();
        DatagramPacket sendPacket = new DatagramPacket(txbufferSound, txbufferSound.length,
hostAddress, serverPort);
        try {
            socket.send(sendPacket);
        }
        catch (Exception x) {
            // x.printStackTrace(); // a more detailed diagnostic call
```

```
            System.out.println(x);
            System.out.println("Audio application TX failed");
        }

        // RX
        byte[] dataSound = new byte[128];
        DatagramPacket receivePacket= new DatagramPacket(dataSound, dataSound.length);
        ByteArrayOutputStream bufferSound = new ByteArrayOutputStream();
        int countPackets = 0;
        int packetsSize = 0;
        long timeBefore = System.currentTimeMillis();
        long timeBeforePerPacket = System.currentTimeMillis();


        // create files
        File diffSamples = new File("logs/" + encoding + type + "diff_samples.txt") ;
        File fileSamples = new File("logs/" + encoding + type + "samples.txt") ;
        File fileMean = new File("logs/aqdpcm_mean.txt");
        File fileStep = new File("logs/aqdpcm_step.txt");
        FileWriter writerDiffSamples = null;
        FileWriter writerSamples = null;
        FileWriter writerMean = null;
        FileWriter writerStep = null;
        try {
            writerDiffSamples = new FileWriter(diffSamples);
            writerSamples = new FileWriter(fileSamples);
            writerMean = new FileWriter(fileMean);
            writerStep = new FileWriter(fileStep);
        }
        catch (Exception x) {
            System.out.println(x);
            System.out.println("Failed to create a file writer for the DPCM");
        }


        try {
            socket.setSoTimeout(3000);
            for (int l = 0; l < Integer.parseInt(numAudioPackets); l++) {
                timeBeforePerPacket = System.currentTimeMillis();
                socket.receive(receivePacket);
                countPackets++;
                long timeAfterPerPacket = System.currentTimeMillis();
                System.out.println("The time required to reveive a packet is: " + (
timeAfterPerPacket - timeBeforePerPacket)/(float)1000 + " seconds");

                packetsSize += dataSound.length;
                System.out.println("Packet No" + countPackets + ". Length of data: " +
dataSound.length);

                if (encoding.equals("")) {
                    // DPCM
                    bufferSound.write(dpcm(dataSound, writerDiffSamples, writerSamples));
                }
                else if (encoding.equals("AQ")) {
                    // AQ-DPCM

                    bufferSound.write(adpcm(dataSound, writerDiffSamples, writerSamples,
writerMean,
                                            writerStep));
                }
```

```
                else {
                    System.out.println("This is not a valid request code");
                }
                System.out.println();
            }
        }
        catch (Exception x) {
            System.out.println(x);
            System.out.println("Receiving/writing the audio data failed");
        }

        // close files
        try {
            writerSamples.close();
            writerDiffSamples.close();
            writerMean.close();
            writerStep.close();
        }
        catch (Exception x) {
            System.out.println(x);
            System.out.println("Failed to close audio files");
        }

        long timeAfter = System.currentTimeMillis();

        System.out.println("\nComplete byte content of the sound file in hexadecimal format:")
;
        byte[] completeDataSound = bufferSound.toByteArray();
        for (byte i : completeDataSound) {
            String hexa = String.format("%02X", i); // print hexadecimal the content of the
byte array
            System.out.print(hexa);
        }

        System.out.println("\n\nTotal number of packages: " + (countPackets));
        System.out.println("How many Kbytes is the sound? " + completeDataSound.length/(float)
1000 + "\nHow many Kbytes is the data that was actually sent? " + packetsSize/(float)1000);
        System.out.println("Total amount of time to receive sound data: " + (timeAfter -
timeBefore)/(float)1000 + " seconds");

        boolean isBigEndian = false; // only in 16 bit samples does matter. In AQ-DPCM we use
16 bit encoding
        int encodingBits = 8;
        if (encoding.equals("AQ")) {
            isBigEndian = true;
            encodingBits = 16;
        }
        AudioFormat modulationPCM = new AudioFormat(8000, encodingBits, 1, true, isBigEndian);
        // play sound
        try {
            SourceDataLine outputAudio = AudioSystem.getSourceDataLine(modulationPCM);
            //outputAudio.open(modulationPCM, 3200);
            outputAudio.open(modulationPCM);
            outputAudio.start();

            System.out.println("Getting ready to hear some music?");
            Thread.sleep(2000);
            System.out.print("In 3");
            Thread.sleep(1000);
            System.out.print(", 2");
```

```
            Thread.sleep(1000);
            System.out.println(", 1...");
            Thread.sleep(500);
            System.out.println("Listening...");
            Thread.sleep(500);
            outputAudio.write(completeDataSound, 0, completeDataSound.length);
            outputAudio.stop();
            outputAudio.close();
            System.out.println("\nSound application success!");
        }
        catch (Exception x) {
            System.out.println(x);
            System.out.println("Sound playing failed");
        }

        // save music to file
        try{
            ByteArrayInputStream bufferSoundInput = new ByteArrayInputStream(completeDataSound
);
            AudioInputStream streamSoundInput = new AudioInputStream(bufferSoundInput,
modulationPCM, completeDataSound.length / modulationPCM.getFrameSize());
            AudioSystem.write(streamSoundInput, AudioFileFormat.Type.WAVE, new File(
pathFileSound));
            System.out.println("Sound file creation success");
        }
        catch (Exception x) {
            System.out.println(x);
            System.out.println("Sound file creation failed");
        }
    }

    private static byte[] dpcm(byte[] dataSound, FileWriter writerDiffSamples,
                               FileWriter writerSamples) {

        ByteArrayOutputStream bufferSound = new ByteArrayOutputStream();
        int init = 0;
        int step = 1; // Trials: for 100 is pure noise, 4 good, 10 bad. I think below 4 you
are good. In general it shouldn't



        for (int i = 0; i<dataSound.length; i++) {

            String hexa = String.format("%02X", dataSound[i]); // print hexadecimal the
content of the byte array
            System.out.print("Input: decimal: " + dataSound[i] + ", unsigned: " + Byte.
toUnsignedInt(dataSound[i])  + " and the hexa: " + hexa + ", ");
            // get nibbles
            int maskLow = 0x0F;
            int maskHigh = 0xF0;
            int nibbleLow = dataSound[i] & maskLow; // D[i] = x[i] - x[i-1]
            int nibbleHigh = (dataSound[i] & maskHigh)>>4; // D[i-1] = x[i-1] - x[i-2]

            // differences
            int diffHigh = (nibbleHigh - 8)*step;
            int diffLow = (nibbleLow - 8)*step;

            // get samples
            int sampleFirst = init + diffHigh;
            int sampleSecond = sampleFirst + diffLow;
```

```
            System.out.print("Masks high and low: " + maskHigh + ", " + maskLow + ". Masks in
hex: " + String.format("%02X", maskHigh) +", " + String.format("%02X", maskLow) + ". Result
 of mask: " + String.format("%02X", nibbleHigh) + ", " + String.format("%02X", nibbleLow) +
 ". Nibbles high and low: " + nibbleHigh + ", " + nibbleLow + ", so the actual differences
are: " + (nibbleHigh-8) +", " + (nibbleLow-8) + " and samples: " + sampleFirst + ", " +
sampleSecond);
            init = sampleSecond;

            // check range
            int max8 = (int)(Math.pow(2,7)) - 1;
            int min8 = -(int)(Math.pow(2,7));
            int[] samples = {sampleFirst, sampleSecond};
            for (int j=0; j< samples.length; j++) {
                if (samples[j]>max8) samples[j] = max8;
                else if (samples[j]<min8) samples[j] = min8;
            }

            // write data to files
            try {
                writerDiffSamples.write(diffHigh + "\n" + diffLow + "\n");
                writerSamples.write(samples[0] + "\n" + samples[1] + "\n");
            }
            catch (Exception x) {
                System.out.println(x);
                System.out.println("Failed to write data to DPCM file");
            }

            // write to buffer
            byte[] decodedSound = new byte[2];
            decodedSound[0] = (byte)sampleFirst;
            decodedSound[1] = (byte)sampleSecond;
            System.out.println(". Output: " + String.format("%02X", decodedSound[0]) + String.
format("%02X", decodedSound[1]));
            try {
                bufferSound.write(decodedSound);
            }
            catch (Exception x) {
                System.out.println(x);
                System.out.println("Decoding DPCM failed");
            }
        }



        return bufferSound.toByteArray();

    }

    private static byte[] adpcm(byte[] dataSound, FileWriter writerDiffSamples,
                                FileWriter writerSamples, FileWriter writerMean, FileWriter
writerStep) {

        // get the header first
        int mean = (Byte.toUnsignedInt(dataSound[1])<<8 | Byte.toUnsignedInt(dataSound[0]));
// be sure to not preserve the byte sign
        int meanSigned = (dataSound[1]<<8 | dataSound[0]); // this is wrong. Not sure though?
        System.out.println("dataSound[1]: " + String.format("%02X", dataSound[1]) + ",
dataSound[1]<<8: " + String.format("%02X", (Byte.toUnsignedInt(dataSound[1]))<<8));
        System.out.println("The MSB of mean is " + String.format("%02X", dataSound[1]) + " and
 the LSB of the mean is "+ String.format("%02X", dataSound[0]) + ". The mean is " + mean +
```

```
" and signed " + meanSigned + " and in hex unsigned: " + String.format("%02X", mean) + "
and signed " + String.format("%02X", meanSigned));
362      int step = (Byte.toUnsignedInt(dataSound[3])<<8 | Byte.toUnsignedInt(dataSound[2]));
363      System.out.println("The MSB of step is " + String.format("%02X", dataSound[3]) + " and
 the LSB of the step is " + String.format("%02X", dataSound[2]) + ". The step is " + step +
 " and in hex: " + String.format("%02X", step));
364      // step should be unsigned? we are safe since it is int and the value is 2 bytes
maximum. Int preserves sign
365
366      //int mean = 256*dataSound[1] + dataSound[0];
367      //int step = 256*dataSound[3] + dataSound[2];
368      //System.out.print((short)(256*Byte.toUnsignedInt(dataSound[1]) + Byte.toUnsignedInt(
dataSound[0])));
369      //System.out.println(", "+ (short)((256*Byte.toUnsignedInt(dataSound[3]) + Byte.
toUnsignedInt(dataSound[2]))));
370
371      try {
372          writerMean.write(meanSigned + "\n");
373          writerStep.write(step + "\n");
374      }
375      catch (Exception x) {
376          System.out.println(x);
377          System.out.println("Failed to write mean and step files for AQ-DPCM");
378      }
379
380      ByteArrayOutputStream bufferSound = new ByteArrayOutputStream();
381      int init = meanSigned; // in DPCM we don't know the init value, we assume zero. But
here we have data in the header.
382      for (int i = 3; i<dataSound.length; i++) {
383          // the sample may be bigger than byte. So you will need 16 bit encoding and store
each int to 2 bytes.
384          System.out.print("Input: " + String.format("%02X", dataSound[i]) + ", "); // print
 hexadecimal the content of the byte array
385
386          // get nibbles
387          int maskLow = 0x0F;
388          int maskHigh = 0xF0;
389          int nibbleLow = (dataSound[i] & maskLow); // D[i] = x[i] - x[i-1], should be
unsigned
390          int nibbleHigh = (dataSound[i] & maskHigh)>>4; // D[i-1] = x[i-1] - x[i-2], should
 be unsigned
391
392          // differences
393          int diffHigh = (nibbleHigh - 8)*step;
394          int diffLow = (nibbleLow - 8)*step;
395
396          // get samples (implement recursive formula)
397          int sampleFirst = init + diffHigh;
398          int sampleSecond = sampleFirst + diffLow;
399          System.out.print("Masks high and low: " + maskHigh + ", " + maskLow + ". Masks in
hex: " + String.format("%02X", maskHigh) +", " + String.format("%02X", maskLow) + ". Result
 of mask: " + String.format("%02X", nibbleHigh) + ", " + String.format("%02X", nibbleLow) +
 ". Nibbles high and low: " + nibbleHigh + ", " + nibbleLow + ", so the actual differences
are: " + (nibbleHigh-8)*step +", " + (nibbleLow-8)*step + " and samples: " + sampleFirst +
", " + sampleSecond);
400          init = sampleSecond;
401
402          // check range
403          int max16 = (int)(Math.pow(2,15)) - 1;
404          int min16 = -(int)(Math.pow(2,15));
```

```
            int[] samples = {sampleFirst, sampleSecond};
            for (int j=0; j<samples.length; j++) {
                if (samples[j]>max16) samples[j] = max16;
                else if (samples[j]<min16)  samples[j] = min16;
            }
            System.out.print(". The actual samples due to 16-bit restriction are: " + samples[0] + " and " + samples[1] + " and in hex format: " + String.format("%02X", samples[0]) + ", " + String.format("%02X", samples[1]) + ". In short " + (short)samples[0] + ", " + (short)samples[1] + " and in hex format as a short: " + String.format("%02X", (short)samples[0]) + ", " + String.format("%02X", (short)samples[1]));

            // write data to files
            try {
                writerDiffSamples.write(diffHigh + "\n" + diffLow + "\n");
                writerSamples.write(samples[0] + "\n" + samples[1] + "\n");
            }
            catch (Exception x) {
                System.out.println(x);
                System.out.println("Failed to write data to AQ-DPCM file");
            }

            // write to buffer
            byte[] decodedSound = new byte[4];
            decodedSound[0] = (byte)(samples[0]>>8); // MSB of sample 15-8
            decodedSound[1] = (byte)samples[0]; // LSB of sample 7-0
            decodedSound[2] = (byte)(samples[1]>>8);
            decodedSound[3] = (byte)samples[1];
            System.out.println(". Output: First sample " + String.format("%02X", decodedSound[0]) + String.format("%02X", decodedSound[1]) + " second sample: " + String.format("%02X", decodedSound[2]) + String.format("%02X", decodedSound[3]));
            try {
                bufferSound.write(decodedSound);
            }
            catch (Exception x) {
                System.out.println(x);
                System.out.println("Decoding DPCM failed");
            }
        }


        return bufferSound.toByteArray();
    }
}
```

3.3 Obd.java

```
package applications;
import java.io.File;
import java.io.FileWriter;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.Socket;
import java.net.InetAddress;
import java.nio.charset.StandardCharsets;

public class Obd {

```

```
    private static String[] header = {"01 1F", "01 0F", "01 11", "01 0C", "01 0D", "01 05"};


    public static void udpTelemetry(DatagramSocket socket, InetAddress hostAddress, int
serverPort, String requestCode) {

        byte[] rxbuffer = new byte[16];
        DatagramPacket receivePacket = new DatagramPacket(rxbuffer, rxbuffer.length);

        for (int i = 0; i < header.length; i++) {

            // TX
            String completeCode = (requestCode + "OBD=" + header[i]);
            byte[] txbuffer = completeCode.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(txbuffer, txbuffer.length,
hostAddress, serverPort);
            System.out.println("Complete request: " + completeCode);
            try {
                socket.send(sendPacket);
            }
            catch (Exception x) {
                // x.printStackTrace(); // a more detailed diagnostic call
                System.out.println(x);
                System.out.println("OBD vehicle application TX failed");
            }
            long timeBefore = System.currentTimeMillis();

            // RX
            try{
                socket.setSoTimeout(3000);
                socket.receive(receivePacket);
                String message = new String(rxbuffer, StandardCharsets.US_ASCII);
                System.out.println("Ithaki responded via UDP with: " + message);
                System.out.println("Ithaki UDP time response: " + (System.currentTimeMillis()-
timeBefore)/(float)1000 + " seconds");

                int[] values = parser(message);
                formula(values[0], values[1], header[i]);
            }
            catch (Exception x) {
                System.out.println(x);
                System.out.println("RX UDP vehicle failed");
            }
        }

    }

    public static float tcpTelemetry(Socket socket, FileWriter writerVehicle) {

        float engineTime = 0;

        try {
            InputStream in = socket.getInputStream();
            OutputStream out = socket.getOutputStream();
            BufferedReader bf = new BufferedReader(new InputStreamReader(in)); // wrapper on
top of the wrapper as java docs recommends

            for (int i = 0; i < header.length; i++) {
                out.write((header[i] + "\r").getBytes());
                //out.flush();
```

```
                long timeBefore = System.currentTimeMillis();
                System.out.println("Created TCP socket and set output stream... Waiting for response");

                System.out.println("Header: " + header[i]);
                String data = bf.readLine();
                System.out.println("Ithaki responded via TCP with: " + data);
                System.out.println("Ithaki TCP time response: " + (System.currentTimeMillis()-timeBefore)/(float)1000 + " seconds");

                int[] values = parser(data);
                float value = formula(values[0], values[1], header[i]);
                writerVehicle.write(value + " ");
                if (header[i] == "01 1F") engineTime = value;
            }

            writerVehicle.write("\n");
        }
        catch (Exception x) {
            System.out.println(x);
            System.out.println("Oops... Vehicle OBD TCP failed");

        }

        return engineTime;
    }

    private static float formula(int first, int second, String header) {
        float value = 0;
        switch (header) {
            case "01 1F":
                int engineRunTime = first*256 + second;
                System.out.println("Engine run time: " + engineRunTime);
                value = engineRunTime;
                break;

            case "01 0F":
                int intakeAirTemp = first - 40;
                System.out.println("Intake Air Temperature: " + intakeAirTemp);
                value = intakeAirTemp;
                break;

            case "01 11":
                float throttlePos = (first*100)/(float)255;
                System.out.println("Throttle position: " + throttlePos);
                value = throttlePos;
                break;

            case "01 0C":
                float engineRpm = ((first*256) + second)/(float)4;
                System.out.println("Engine RPM: " + engineRpm);
                value = engineRpm;
                break;

            case "01 0D":
                int speed = first;
                System.out.println("Vehicle speed: " + speed);
                value = speed;
                break;

```

```
            case "01 05":
                int coolantTemp = first -40;
                System.out.println("Coolant Temperature: " + coolantTemp);
                value = coolantTemp;
                break;

            default:
                System.out.println("Something went wrong calculating formual for vehicle stats
    ");

        }
        System.out.println();
        return value;
    }

    private static int[] parser(String data) {
        String byte1 = data.substring(6,8);
        // how to convert hexadecimal string to int?
        int first = Integer.parseInt(byte1, 16);
        System.out.print("Parsing the data: 1st byte: " + byte1 + " and as an integer: " +
    first);
        String byte2 = "";
        int second = 0;
        if (data.length()>8) {
            byte2 = data.substring(9,11);
            second = Integer.parseInt(byte2, 16);
            System.out.print(", 2nd byte: " + byte2 + " and as an integer: " + second);
        }
        System.out.println();

        int[] temp = {first, second};
        return temp;
    }
}
```

3.4 Copter.java

```
package applications;

import java.io.FileWriter;
import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.DatagramSocket;
import java.net.DatagramPacket;
import java.net.Socket;
import java.net.InetAddress;
import java.nio.charset.StandardCharsets;
import java.io.ByteArrayOutputStream;

public class Copter {
    public static String udpTelemetry(DatagramSocket socket, InetAddress hostAddress, int
    serverPort, FileWriter writerCopter) {
        // TX
        // open ithakicopter.jar


        //RX only
        byte[] rxbuffer = new byte[128];
```

```
        DatagramPacket receivePacket = new DatagramPacket(rxbuffer, rxbuffer.length);

        long timeBefore = System.currentTimeMillis();
        String telemetry = new String();
        try{
            socket.setSoTimeout(3000);
            socket.receive(receivePacket);
            telemetry = new String(rxbuffer, StandardCharsets.US_ASCII);
            //System.out.print("Time repsonse: " + (System.currentTimeMillis() - timeBefore)/(
    float)1000);
            System.out.println("Received data via UDP: " + telemetry);

            String[] tokensMotor = telemetry.split("LMOTOR=");
            String[] tokensAltitude = telemetry.split("ALTITUDE=");
            String[] tokensTemp = telemetry.split("TEMPERATURE=");
            String[] tokensPress = telemetry.split("PRESSURE=");
            writerCopter.write(tokensMotor[1].substring(0, 3) + " ");
            writerCopter.write(tokensAltitude[1].substring(0, 3) + " ");
            writerCopter.write(tokensTemp[1].substring(1, 6) + " ");
            writerCopter.write(tokensPress[1].substring(0, 7) + "\n");
        }
        catch (Exception x) {
            System.out.println(x);
            System.out.println("RX UDP ithakicopter failed");
        }
        return telemetry;

    }

    public static String tcpTelemetry(InetAddress hostAddress, int target) {
        String telemetry = "";
        Socket socket = new Socket();
        try {
            socket = new Socket(hostAddress, 38048);
            InputStream in = socket.getInputStream();
            OutputStream out= socket.getOutputStream();
            BufferedReader bf = new BufferedReader(new InputStreamReader(in)); // wrapper on
    top of the wrapper as java docs recommends
            ByteArrayOutputStream bos = new ByteArrayOutputStream();

            String command = "AUTO FLIGHTLEVEL=" + target + " LMOTOR=" + target + " RMOTOR=" +
     target +  " PILOT \r\n";
            //System.out.print("Request: " + command);
            out.write(command.getBytes());
            out.flush();

            //in.skipNBytes(427);
            for (int i = 0; i < 14; i++) {
                bos.write((bf.readLine() + "\n").getBytes());
            }
            String data = new String(bos.toByteArray(), StandardCharsets.US_ASCII);
            //System.out.println("Received data via TCP: " + data);

            String[] tokens = data.split("\n");
            // take only the useful data and skip the info ithaki sent
            telemetry = tokens[13];
        }
        catch (Exception x) {
            System.out.println(x);
            System.out.println("Oops... Ithakicopter TCP failed");
```

```
        }
        try {
        socket.close();
        }
        catch (Exception x) {
            System.out.println(x);
            System.out.println("Failed to close socket for ithakicopter TCP");
        }
        return telemetry;
    }


    /*
     * tcpTelemetry function for the TX and udpTelemetry for RX. The way that these two functions are implemented force the autopilot
     * to be used with a combination of these two. We want to send a command only if it is needed and we want to listen all the time
     * to get feedback.
     *
     */
    public static void autopilot(DatagramSocket listen, InetAddress hostAddress, int serverPort, Socket send, int lowerBound, int higherBound) {

        lowerBound = Math.min(lowerBound, higherBound);
        higherBound = Math.max(lowerBound, higherBound);

        int target = (lowerBound + higherBound)/2;
        int motor = -1;

        try {
        System.out.println("AUTOPILOT: ON");
        System.out.println("You need to open ithakicopter.jar. Press ENTER to continue...");
        System.in.read();
        System.out.println("Press Control-C to exit...");
        Thread.sleep(1000);
        for (;;) {
            if ((motor<(lowerBound)) || (motor>(higherBound))) {
                System.out.println("Send packet. Readjust...");
                tcpTelemetry(hostAddress, target);
                }

                String telemetry = Copter.udpTelemetry(listen, hostAddress, serverPort, null);
                String[] tokens = telemetry.split("LMOTOR=");
                motor = Integer.parseInt(tokens[1].substring(0,3)); // get motor values

                System.out.println("Parsed motor values: " + motor);
        }
        }
        catch (Exception x) {
            System.out.println(x);
            System.out.println("AUTOPILOT failed");
        }
    }

}
```

4 Plots

```
from scipy.stats import norm
import matplotlib.pyplot as plt
```

```
import matplotlib.mlab as mlab
import pandas as pd
from numpy import genfromtxt
import numpy as np


# data = pd.read_csv('../logs/echo_samples_delay.csv', sep=',', header=None)
# data.plot(kind='bar')
# plt.ylable('frequency')
# plt.xlabel('Number of packets')
# plt.title('Histogram response time')
# plt.show()

#plt.hist(rtt,histtype = 'bar', bins='auto', density=1, alpha=0.7)
# print("Length of the array is: " + str(len(rtt)))


# Response time diagram
x = genfromtxt('../logs/session2/echo_samples_delay.txt', delimiter=',')
x = genfromtxt('../logs/session2/echo_samples_no_delay.txt', delimiter=',')
x = genfromtxt('../logs/session2/echo_throughput_delay.txt', delimiter=',')
x = genfromtxt('../logs/session2/echo_throughput_no_delay.txt', delimiter=',')
x = genfromtxt('../logs/session1/AQFsamples.txt')
x = genfromtxt('../logs/session2/Fsamples.txt')
x = genfromtxt('../logs/session2/Tsamples.txt')
x = genfromtxt('../logs/session2/second_clip/aqdpcm_mean.txt')
x = genfromtxt('../logs/session2/second_clip/aqdpcm_step.txt')
x = genfromtxt('../logs/second_clip/aqdpcm_step.txt')
 plt.subplot(2,1,1)
 plt.plot(x, 'm')
 plt.xlabel('Number of samples', fontsize=12)
 plt.ylabel('Amplitude', fontsize=12)
 plt.grid(True)

 plt.subplot(2,1,2)
 plt.xlabel('Number of samples', fontsize=12)
 plt.ylabel('Amplitude', fontsize=12)
 plt.grid(True)
 plt.plot(x, 'm')
 plt.xlim(100, 200)
 plt.show()


#Retransmission timeout plot
data = genfromtxt('../logs/session2/rto.txt', delimiter=' ')
rtt = data[1:,0]
srtt = data[1:,1]
rttd = data[1:,2]
rto = data[1:,3]
plt.plot(rtt, label = "RTT")
plt.plot(srtt, label = "SRTT")
plt.plot(rttd, label = "RTTd")
plt.plot(rto, label = "RTO")
plt.xlabel('Number of packets', fontsize=12)
plt.ylabel("Time response", fontsize=12)
plt.legend()
plt.show()

# Copter
data = genfromtxt('../logs/session2/copter_2nd_run/copter_info.txt', delimiter=' ')
```

```
rtt = data[1:,0]
srtt = data[1:,1]
rttd = data[1:,2]
rto = data[1:,3]
plt.plot(rtt, label = "MOTOR")
plt.plot(srtt, label = "ALTITUDE")
plt.plot(rttd, label = "TEMPERATURE")
plt.plot(rto, label = "PRESSURE")
plt.xlabel('Number of packets', fontsize=12, labelpad=10)
plt.ylabel("Data", fontsize=12, labelpad=10)
plt.legend()
plt.grid(True);
plt.yticks(np.arange(0, 1200, 50))
plt.show()

#Vehicle
data = genfromtxt('../logs/car_telemetry.txt', delimiter=' ')
rtt = data[0:,0]
srtt = data[0:,1]
rttd = data[0:,2]
rto = data[0:,3]
s = data[0:, 4]
t = data[0:, 5]
plt.subplot(211)
plt.plot(rtt, label = "Engine run time")
plt.plot(rto, label = "Engine RPM")
plt.xlabel('Number of packets', fontsize=12, labelpad=10)
plt.ylabel("Data", fontsize=12, labelpad=10)
plt.legend()
plt.grid(True);
plt.subplot(212)
plt.plot(srtt, label = "Intake air temperature")
plt.plot(rttd, label = "Throttle position")
plt.plot(s, label = "Vehicle speed")
plt.plot(t, label = "Coolant temperature")
plt.xlabel('Number of packets', fontsize=12, labelpad=10)
plt.ylabel("Data", fontsize=12, labelpad=10)
plt.legend()
plt.grid(True);
#plt.yticks(np.arange(0, 2500, 25))
plt.show()
```