



Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης  
Πολυτεχνική Σχολή

## Δίκτυα Υπολογιστών II

Θεόδωρος Κατζάλης  
ΑΕΜ:9282  
katzalis@auth.gr

December 3, 2020

# Contents

1	Δομή του προγράμματος	2
2	UserApplication.java	3
3	applications	13
3.1	Echo.java . . . . .	13
3.2	Media.java . . . . .	14
3.3	Obd.java . . . . .	22
3.4	Copter.java . . . . .	25
4	Plots	27

# 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

```
1 // it is considered in general a bad practise to use asterisks to import all the classes
2 import java.io.*;
3 import java.net.*;
4 import java.nio.charset.StandardCharsets;
5 import java.lang.System;
6 import java.awt.Desktop;
7 import javax.sound.sampled.*;
8 import java.lang.Math.*;
9 import java.util.Arrays;
10 import java.util.Scanner;
11 import java.time.LocalDateTime;
12
13 import applications.*;
14
15 class UserApplication {
16
17     // TODO create a script that is scraping from ithaki website the request code and the
18     // ports.
19
20     public static void main (String[] args) throws Exception {
21
22         // windows users may be not able to view colors on terminal
23         final String ANSI_CYAN = "\u001B[36m";
24         final String ANSI_RESET = "\u001B[0m";
25
26         // print welcome text
27         Scanner input = new Scanner(new File("stamps/welcome.txt"));
28         while (input.hasNextLine())
29         {
30             System.out.print(ANSI_CYAN); // add some color!
31             System.out.print(input.nextLine());
32             System.out.println(ANSI_RESET);
33         }
34         System.out.println();
35         System.out.print("Press ENTER to continue");
36         System.in.read(); // pause a little bit to enjoy the view
37
38
39         // preamble
40         byte[] clientIP = { (byte)192, (byte)168, (byte)1, (byte)20};
41         byte[] hostIP = { (byte)155, (byte)207, (byte)18, (byte)208};
42         InetAddress clientAddress = InetAddress.getByAddress(clientIP);
43         InetAddress hostAddress = InetAddress.getByAddress(hostIP);
44         int serverPort = 38022;
45         int clientPort = 48022;
46         String requestCodeEcho = "E0818 ";
47         String requestCodeImage = "M2685UDP=1024";
48         String requestCodeSound = "A7269";
49         String requestCodeCopter = "Q2797";
50         String requestCodeVehicle = "V4118";
51
52         DatagramSocket socket = new DatagramSocket(clientPort);
53         long timeBefore = 0;
54
55
56         int flag =1; // control user input
57         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 argument
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

```

```

116         System.out.println("Cumsum: " + cumsum[i]);
117     }
118     else if(toc > 8000){
119         count8sec[i]++;
120         tic[i] = count8sec[i]*8000 + timeBefore + i*1000;
121         throughput[i] = cumsum[i]/(float)8;
122         System.out.println("I will flush " + cumsum[i] + " cumsum");
123         System.out.println("The throughput is: " + throughput[i]);
124
125         writerThroughput.write(throughput[i]+ "\n");
126
127
128         cumsum[i] = 0; // let's start again for the next 8 seconds
129     }
130 }
131
132
133 // Retransmission timeout
134
135 // init values
136 if (isFirst == 1) {
137     rtts = rtt;
138     rtttd = rtt/2;
139     rto = 1; // according to rfc
140     writerRto.write("RTT SRTT RTTd RTO\n");
141 }
142 double temp = rtts;
143 rtts = 0.875*temp + 0.125*rtt;
144
145 temp = rtttd;
146 rtttd = 0.75*temp + 0.25*Math.abs(rtt - rtts);
147
148 rto = rtts + 1.8*rttd;
149
150 writerRto.write(rtt + " " + rtts + " " + rtttd + " " + rto + "\n");
151
152 System.out.println();
153 isFirst = 0;
154 }
155
156 writerInfo.write("Toc: " + LocalDateTime.now());
157
158 writerInfo.close();
159 writerSamples.close();
160 writerThroughput.close();
161 writerRto.close();
162 socket.close();
163 break;
164
165 case "2":
166     input = new Scanner(new File("stamps/echo.txt"));
167     while (input.hasNextLine())
168     {
169         System.out.println(input.nextLine());
170     }
171     Thread.sleep(1500); // pause a little bit to enjoy the view
172
173     info = new File("logs/echo_info_no_delay.txt");
174     writerInfo = new FileWriter(info);
175     writerInfo.write("Info:\n" + "The request code is " + requestCodeEcho + "\n");

```

```

176 writerInfo.write("Tic: " + LocalDateTime.now() + "\n");
177
178 fileSamples = new File("logs/echo_samples_no_delay.txt");
179 writerSamples = new FileWriter(fileSamples);
180
181 fileThroughput = new File("logs/echo_throughput_no_delay.txt");
182 writerThroughput = new FileWriter(fileThroughput);
183
184 timeBefore = System.currentTimeMillis();
185 tic = new long[8];
186 //long toc[] = new long[8];
187 cumsum = new int[8]; // cumulative sum
188 throughput = new float[8]; // worst case we need 8 elements to store data
189 count8sec = new int[8];
190 for(int i = 0; i<8; i++) {
191     tic[i] = timeBefore + i*1000; // move per second
192 }
193
194 while ((System.currentTimeMillis() - timeBefore) < 60000 * 4){
195     long value = Echo.execute(socket, hostAddress, serverPort, "E0000");
196     writerSamples.write(value + "\n");
197
198     // throughput moving average
199
200     for(int i = 0; i<8; i++) {
201         //System.out.println("The element " + i + " has tic: " + tic[i]);
202         long toc = System.currentTimeMillis() - tic[i];
203         System.out.println("The element " + i + " has toc: " + toc);
204         if (toc < 8000 && toc > 0) {
205             cumsum[i] += 32*8; // assume no timeouts during the measurements
206             System.out.println("Cumsum: " + cumsum[i]);
207         }
208         else if(toc > 8000){
209             count8sec[i]++;
210             tic[i] = count8sec[i]*8000 + timeBefore + i*1000;
211             throughput[i] = cumsum[i]/(float)8;
212             System.out.println("I will flush " + cumsum[i] + " cumsum");
213             System.out.println("The throughput is: " + throughput[i]);
214
215             writerThroughput.write(throughput[i]+ "\n");
216
217             cumsum[i] = 0; // let's start again for the next 8 seconds
218         }
219     }
220     System.out.println();
221 }
222
223 writerInfo.write("Toc: " + LocalDateTime.now());
224
225 writerInfo.close();
226 writerSamples.close();
227 writerThroughput.close();
228 socket.close();
229 break;
230
231
232
233 case "3":
234     /* ----- Temperature ----- */
235     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 String numAudioPackets = "999";
290 String[] type = {"F", "T"};
291

```



```

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

```

```

349 writerCopter.write(LocalDateTime.now() + "\n");
350 writerCopter.close();
351 socket.close();
352 break;
353
354 case "8":
355 /* ----- Autopilot ----- */
356 socket = new DatagramSocket(48078);
357 Socket socketAuto = new Socket(hostAddress, 38048);
358 input = new Scanner(new File("stamps/auto.txt"));
359 while (input.hasNextLine())
360 {
361     System.out.println(input.nextLine());
362 }
363 Thread.sleep(1500); // pause a little bit to enjoy the view
364
365 int lowerBound = 160;
366 int higherBound = 190;
367 Copter.autopilot(socket, hostAddress, serverPort, socketAuto, Math.min(200, Math.max
(150, lowerBound)), Math.min(200, Math.max(150, higherBound)));
368 socketAuto.close();
369 break;
370
371
372
373 case "9":
374 /* ----- HTTPS TCP----- */
375 Socket httpsSocket = new Socket(hostAddress, 80);
376 input = new Scanner(new File("stamps/https.txt"));
377 while (input.hasNextLine())
378 {
379     System.out.println(input.nextLine());
380 }
381 Thread.sleep(1500); // pause a little bit to enjoy the view
382
383 https(httpsSocket);
384 httpsSocket.close();
385 break;
386
387 case "10":
388 /* ----- Ithakicopter TCP----- */
389 // can we use 38098? If we open the website we see that this is the port opened
390 //Socket socketCopter = new Socket(hostAddress, 38048);
391 input = new Scanner(new File("stamps/copter_tcp.txt"));
392 while (input.hasNextLine())
393 {
394     System.out.println(input.nextLine());
395 }
396 Thread.sleep(1500); // pause a little bit to enjoy the view
397
398 for (int i = 0; i<4; i++) Echo.execute(socket, hostAddress, serverPort,
requestCodeEcho);
399
400 int target = 180;
401 for (int i = 0; i<20; i++) {
402     System.out.println(new String(Copter.tcpTelemetry(hostAddress, target)));
403 }
404
405 //socketCopter.close();
406 break;

```

```

407 case "11":
408 /* ----- Vehicle OBD TCP----- */
409 input = new Scanner(new File("stamps/obd_tcp.txt"));
410 while (input.hasNextLine())
411 {
412     System.out.println(input.nextLine());
413 }
414 Thread.sleep(1500); // pause a little bit to enjoy the view
415
416 for (int i = 0; i<4; i++) Echo.execute(socket, hostAddress, serverPort,
417 requestCodeEcho);
418
419 Socket socketVehicle = new Socket(hostAddress, 29078);
420
421 FileWriter writerVehicleInfo = new FileWriter(new File("logs/car_info.txt"));
422 writerVehicleInfo.write("Info Vehicle app:\n" + LocalDateTime.now() + "\n");
423 FileWriter writerVehicleData = new FileWriter(new File("logs/car_telemetry.txt"));
424
425 timeBefore = System.currentTimeMillis();
426 float engineTime = 0;
427 while ( engineTime < 60 * 4){
428     engineTime = Obd.tcpTelemetry(socketVehicle, writerVehicleData);
429     System.out.println("The engine run time is " + engineTime + "\n");
430 }
431
432 writerVehicleInfo.write(LocalDateTime.now() + "\n");
433 writerVehicleInfo.close();
434 writerVehicleData.close();
435 socketVehicle.close();
436 break;
437
438
439
440 /*
441 * *****
442 *           This is a playground!! Test whatever you like....
443 * *****
444 *
445 */
446 case "12":
447 Socket foo = new Socket(hostAddress, 38048);
448 OutputStream out = foo.getOutputStream();
449 InputStream in = foo.getInputStream();
450 InputStreamReader isr = new InputStreamReader(in);
451 BufferedReader bf = new BufferedReader(isr);
452 ByteArrayOutputStream bis = new ByteArrayOutputStream();
453
454 // if use readAllBytes the InputStream is closed
455 // actually readNBytes is quite weird to be honest
456 out.write("AUTO FLIGHTLEVEL=100 LMOTOR=100 RMOTOR=100 PILOT \r\n".getBytes());
457 String data = new String();
458 while ((data = bf.readLine()) != null) {
459     bis.write((data + "\n").getBytes());
460 }
461 data = new String(bis.toByteArray(), StandardCharsets.US_ASCII);
462 System.out.println(data);
463 break;
464
465 case "13":

```

```

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

```

```

520
521
522  /* ----- Close UDP sockets ----- */
523  if (!socket.isClosed()) {
524      socket.close();
525      System.out.println("\nShuting down UDP sockets...");
526  }
527
528
529  System.out.println("\nx-----Hooray! Java application finished
530  successfully!-----x");
531
532
533
534  private static void https(Socket socket) {
535      try {
536          InputStream in = socket.getInputStream(); // what I receive from the server
537          OutputStream out = socket.getOutputStream(); // what i send to the server
538
539          long timeBefore = System.currentTimeMillis();
540          out.write("GET /netlab/hello.html HTTP/1.0\r\nHost: ithaki.eng.auth.gr:80\r\n\r\n"
541          .getBytes());
542
543          byte[] inputBuffer = in.readAllBytes();
544          String message = new String(inputBuffer, StandardCharsets.US_ASCII);
545          System.out.println("Ithaki responded via TCP with: \n" + message);
546          System.out.println("Time response: " + (System.currentTimeMillis() - timeBefore)/(
547          float)1000 + " seconds");
548
549          socket.close();
550      }
551      catch (Exception x) {
552          System.out.println(x + "TCP application failed");
553      }
554
555  }
556
557  private static void checkArguements (String[] args) {
558      if (args.length == 2) {
559          System.out.println("Command line arguements: This is the first " + args[0] + " " +
560          "and this is the second " + args[1]);
561      }
562      String[] directionOptions = {"L", "D", "U", "R"};
563      int flag = 0;
564      for (String i : directionOptions) {
565          if (i.equals(args[0])) {
566              flag = 1;
567              System.out.println("Direction: " + args[0]);
568              break;
569          }
570      }
571      if (flag == 0) {
572          System.out.println("Try again, wrong direction. Available options are: L, R, U, D"
573          );
574          return;
575      }
576      if (Integer.parseInt(args[1])>=1 && Integer.parseInt(args[1])<=100) {
577          System.out.println("How many times to repeat the movement of the camera on that
578          direction? " + Integer.parseInt(args[1]));
579      }
580  }

```

```

574     }
575     else {
576         System.out.println("Invalid input. Range should be 1-100");
577         return;
578     }
579 }
580
581 }

```

## 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
14     * desired after the call of the function.
15     *
16     * @param requestCode If request code is set to E000 then the execute will have no delay
17     * for the RX
18     */
19     public static long execute(DatagramSocket socket, InetAddress hostAddress, int serverPort,
20     String requestCode) {
21         System.out.println("\n-----Echo application-----");
22
23         if (requestCode.equals("E0000")) System.out.println("Delay: OFF");
24         else if (requestCode.length()>5) System.out.println("Mode: Temperature\nDelay: OFF");
25         else System.out.println("Delay: OFF");
26
27         byte[] txbuffer = requestCode.getBytes();
28         byte[] rxbuffer = new byte[64];
29         long diff = 0;
30         try {
31             socket.setSoTimeout(3000);
32             DatagramPacket sendPacket = new DatagramPacket(txbuffer, txbuffer.length,
33             hostAddress, serverPort);
34             DatagramPacket receivePacket= new DatagramPacket(rxbuffer, rxbuffer.length);
35
36             // ACTION
37             socket.send(sendPacket);
38             System.out.println("The request code is: " + requestCode + "\nThe destination port
39             is: " + serverPort + "\nMy listening port (clientPort): " + socket.getLocalPort());
40             long timeBefore = System.currentTimeMillis();
41             System.out.println("My system time, when the request is sent, is: " + timeBefore);
42
43             // LISTEN
44             socket.receive(receivePacket);
45             long timeAfter = System.currentTimeMillis();
46             diff = timeAfter - timeBefore;

```

```

42     System.out.println("The time required to receive a packet is: " + diff + "
milliseconds");
43     //System.out.println("The port that opened ithaki to send the request is : " +
receivePacket.getPort() + " and the address of ithaki is: " + receivePacket.getAddress());
44     String message = new String(receivePacket.getData(), StandardCharsets.US_ASCII);
// convert binary to ASCII
45     System.out.println("Ithaki responded with: " + message);
46 }
47 catch (Exception x) {
48     // x.printStackTrace(); // a more detailed diagnostic call
49     System.out.println(x);
50     System.out.println("Something went wrong about Echo application mode");
51 }
52 return diff;
53 }
54 }

```

## 3.2 Media.java

```

1 package applications;
2
3 import java.net.DatagramSocket;
4 import java.net.DatagramPacket;
5 import java.net.InetAddress;
6 import java.io.ByteArrayOutputStream;
7 import java.io.ByteArrayInputStream;
8 import java.io.File;
9 import java.io.FileWriter;
10 import java.io.FileOutputStream;
11 import java.io.IOException;
12 import java.awt.Desktop;
13 import java.util.Arrays;
14 import javax.sound.sampled.*;
15
16 public class Media {
17
18     private static String pathFileImage = "/home/tkatz/repos/ece-networks2/media/image/sandbox
/ithaki_image.jpg";
19     private static String pathFileSound = "/home/tkatz/repos/ece-networks2/media/music/sandbox
/track.wav";
20
21     public static void image(DatagramSocket socket, InetAddress hostAddress, int serverPort,
String requestCode) throws IOException {
22
23         //if (numImage != (Integer.parseInt(args[1])-1)){
24         //    continue; // readjust the camera as many time is requested via the command line
argument. Print only the result, the last request
25         //}
26
27         byte[] txbufferImage = requestCode.getBytes();
28         byte[] rxbufferImage = new byte[1024];
29         int countPackets = 0;
30         long timeBefore = System.currentTimeMillis();
31         long timeBeforePerPacket = System.currentTimeMillis();
32         //System.out.println("My system time, when the request is sent, is: " + timeBefore);
33
34         System.out.println("The request code is " + requestCode);
35         DatagramPacket sendPacket = new DatagramPacket(txbufferImage, txbufferImage.length,
hostAddress, serverPort);
36         DatagramPacket receivePacket= new DatagramPacket(rxbufferImage, rxbufferImage.length);
37

```

```

38 // TX
39 System.out.println("I am sleeping... Camera needs time to readjust");
40 try {
41     socket.send(sendPacket);
42     Thread.sleep(5000); // sleep in order for the camera to readjust
43 }
44 catch (Exception x) {
45     // x.printStackTrace(); // a more detailed diagnostic call
46     System.out.println(x);
47     System.out.println("Image application TX failed");
48 }
49
50 // RX
51 ByteArrayOutputStream bufferImage = new ByteArrayOutputStream();
52 outerloop:
53     try {
54         socket.setSoTimeout(3000);
55         for (;;) {
56             socket.receive(receivePacket); // blocking command
57             countPackets++;
58
59             long timeAfterPerPacket = System.currentTimeMillis();
60             System.out.println("The time required to receive a packet is: " + (
timeAfterPerPacket - timeBeforePerPacket)/(float)1000 + " seconds");
61             timeBeforePerPacket = System.currentTimeMillis();
62
63             System.out.println("Packet No" + countPackets + ". Length of data: " +
rxbufferImage.length + ". The received bytes in hexadecimal format are:");
64             for (int i = 0; i<rxbufferImage.length; i++) {
65                 System.out.print(String.format("%02X", rxbufferImage[i])); // convert
bytes to hexa string
66
67                 bufferImage.write(rxbufferImage[i]); // dynamic byte allocation
68                 if ((String.format("%02X", rxbufferImage[i]).equals("D9")) && (i!=0)) {
69                     if ((String.format("%02X", rxbufferImage[i-1]).equals("FF"))) {
70                         break outerloop; // stop writing when EOF (0xFFD9 delimiter)
71                     }
72                 }
73             }
74             System.out.println();
75         }
76     }
77     catch (Exception x) {
78         // x.printStackTrace(); // a more detailed diagnostic call
79         System.out.println(x);
80         System.out.println("Image application RX failed");
81     }
82     long timeAfter = System.currentTimeMillis(); // get the time when the image is
received in bytes
83
84     // logs for the received byte content
85     System.out.println("\nComplete byte content of the image file in hexadecimal format:");
86
87     ;
88     byte[] completeDataImage = bufferImage.toByteArray();
89     for (byte i : completeDataImage) {
90         System.out.print(String.format("%02X", i)); // print hexadecimal the content of
the byte array
91     }
92     System.out.println("\n\nTotal number of packages: " + (countPackets));
93     System.out.println("How many Kbytes is the image? " + completeDataImage.length/(float)

```



```

1000);
92
93 // save image to a file
94 File imageFile = new File(pathFileImage);
95 FileOutputStream fos = null;
96 try {
97     fos = new FileOutputStream(imageFile);
98     fos.write(completeDataImage);
99     System.out.println("File has been written successfully");
100 }
101 catch (Exception x) {
102     // x.printStackTrace(); //
103     System.out.println("Image application error when writing the file:");
104 }
105
106 fos.close(); // close the OutputStream
107
108 // what time is o'clock?
109 System.out.println("Total amount of time to receive a frame: " + (timeAfter -
timeBefore)/(float)1000 + " seconds");
110 timeAfter = System.currentTimeMillis(); // get the time when the file is ready
111 System.out.println("Total amount of time to receive and write a frame in a .jpg file:
" + (timeAfter - timeBefore)/(float)1000 + " seconds");
112
113 // open file image
114 Desktop desktop = Desktop.getDesktop();
115 if (imageFile.exists()) {
116     //desktop.open(imageFile);
117 }
118 }
119
120 public static void audio(DatagramSocket socket, InetAddress hostAddress, int serverPort,
String requestCode) {
121
122     // parsing the requestCode
123     // expecting requestCode: AXXXX + ("AQ" or "") + ("T" or "F") + numAudioPackets
124     String encoding = "";
125     String type = "F";
126     String numAudioPackets = "";
127     if (requestCode.length() == 11) {
128         encoding = "AQ";
129         type = requestCode.substring(7, 8);
130         numAudioPackets = requestCode.substring(8, 11);
131     }
132     else {
133         type = requestCode.substring(5, 6);
134         numAudioPackets = requestCode.substring(6, 9);
135     }
136     System.out.println("Requested: Encoding: " + encoding + ". Type: " + type + ". Number
of packets: " + numAudioPackets);
137
138     // TX
139     byte[] txbufferSound = ("L02" + requestCode).getBytes();
140     DatagramPacket sendPacket = new DatagramPacket(txbufferSound, txbufferSound.length,
hostAddress, serverPort);
141     try {
142         socket.send(sendPacket);
143     }
144     catch (Exception x) {
145         // x.printStackTrace(); // a more detailed diagnostic call

```

```

146     System.out.println(x);
147     System.out.println("Audio application TX failed");
148 }
149
150 // RX
151 byte[] dataSound = new byte[128];
152 DatagramPacket receivePacket= new DatagramPacket(dataSound, dataSound.length);
153 ByteArrayOutputStream bufferSound = new ByteArrayOutputStream();
154 int countPackets = 0;
155 int packetsSize = 0;
156 long timeBefore = System.currentTimeMillis();
157 long timeBeforePerPacket = System.currentTimeMillis();
158
159
160 // create files
161 File diffSamples = new File("logs/" + encoding + type + "diff_samples.txt") ;
162 File fileSamples = new File("logs/" + encoding + type + "samples.txt") ;
163 File fileMean = new File("logs/aqdpdm_mean.txt");
164 File fileStep = new File("logs/aqdpdm_step.txt");
165 FileWriter writerDiffSamples = null;
166 FileWriter writerSamples = null;
167 FileWriter writerMean = null;
168 FileWriter writerStep = null;
169 try {
170     writerDiffSamples = new FileWriter(diffSamples);
171     writerSamples = new FileWriter(fileSamples);
172     writerMean = new FileWriter(fileMean);
173     writerStep = new FileWriter(fileStep);
174 }
175 catch (Exception x) {
176     System.out.println(x);
177     System.out.println("Failed to create a file writer for the DPCM");
178 }
179
180
181 try {
182     socket.setSoTimeout(3000);
183     for (int l = 0; l < Integer.parseInt(numAudioPackets); l++) {
184         timeBeforePerPacket = System.currentTimeMillis();
185         socket.receive(receivePacket);
186         countPackets++;
187         long timeAfterPerPacket = System.currentTimeMillis();
188         System.out.println("The time required to reveive a packet is: " + (
timeAfterPerPacket - timeBeforePerPacket)/(float)1000 + " seconds");
189
190         packetsSize += dataSound.length;
191         System.out.println("Packet No" + countPackets + ". Length of data: " +
dataSound.length);
192
193         if (encoding.equals("")) {
194             // DPCM
195             bufferSound.write(dpcm(dataSound, writerDiffSamples, writerSamples));
196         }
197         else if (encoding.equals("AQ")) {
198             // AQ-DPCM
199
200             bufferSound.write(adpcm(dataSound, writerDiffSamples, writerSamples,
writerMean,
201                                     writerStep));
202         }

```

```

203         else {
204             System.out.println("This is not a valid request code");
205         }
206         System.out.println();
207     }
208 }
209 catch (Exception x) {
210     System.out.println(x);
211     System.out.println("Receiving/writing the audio data failed");
212 }
213
214 // close files
215 try {
216     writerSamples.close();
217     writerDiffSamples.close();
218     writerMean.close();
219     writerStep.close();
220 }
221 catch (Exception x) {
222     System.out.println(x);
223     System.out.println("Failed to close audio files");
224 }
225
226 long timeAfter = System.currentTimeMillis();
227
228 System.out.println("\nComplete byte content of the sound file in hexadecimal format:");
229 ;
230 byte[] completeDataSound = bufferSound.toByteArray();
231 for (byte i : completeDataSound) {
232     String hexa = String.format("%02X", i); // print hexadecimal the content of the
byte array
233     System.out.print(hexa);
234 }
235
236 System.out.println("\n\nTotal number of packages: " + (countPackets));
237 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);
238 System.out.println("Total amount of time to receive sound data: " + (timeAfter -
timeBefore)/(float)1000 + " seconds");
239
240 boolean isBigEndian = false; // only in 16 bit samples does matter. In AQ-DPCM we use
16 bit encoding
241 int encodingBits = 8;
242 if (encoding.equals("AQ")) {
243     isBigEndian = true;
244     encodingBits = 16;
245 }
246 AudioFormat modulationPCM = new AudioFormat(8000, encodingBits, 1, true, isBigEndian);
247 // play sound
248 try {
249     SourceDataLine outputAudio = AudioSystem.getSourceDataLine(modulationPCM);
250     //outputAudio.open(modulationPCM, 3200);
251     outputAudio.open(modulationPCM);
252     outputAudio.start();
253
254     System.out.println("Getting ready to hear some music?");
255     Thread.sleep(2000);
256     System.out.print("In 3");
257     Thread.sleep(1000);
258     System.out.print(", 2");

```

```

258     Thread.sleep(1000);
259     System.out.println(", 1...");
260     Thread.sleep(500);
261     System.out.println("Listening...");
262     Thread.sleep(500);
263     outputAudio.write(completeDataSound, 0, completeDataSound.length);
264     outputAudio.stop();
265     outputAudio.close();
266     System.out.println("\nSound application success!");
267 }
268 catch (Exception x) {
269     System.out.println(x);
270     System.out.println("Sound playing failed");
271 }
272
273 // save music to file
274 try{
275     ByteArrayInputStream bufferSoundInput = new ByteArrayInputStream(completeDataSound
276 );
277     AudioInputStream streamSoundInput = new AudioInputStream(bufferSoundInput,
278 modulationPCM, completeDataSound.length / modulationPCM.getFrameSize());
279     AudioSystem.write(streamSoundInput, AudioFileFormat.Type.WAVE, new File(
280 pathFileSound));
281     System.out.println("Sound file creation success");
282 }
283 catch (Exception x) {
284     System.out.println(x);
285     System.out.println("Sound file creation failed");
286 }
287 }
288
289 private static byte[] dpcm(byte[] dataSound, FileWriter writerDiffSamples,
290     FileWriter writerSamples) {
291
292     ByteArrayOutputStream bufferSound = new ByteArrayOutputStream();
293     int init = 0;
294     int step = 1; // Trials: for 100 is pure noise, 4 good, 10 bad. I think below 4 you
295 are good. In general it shouldn't
296
297     for (int i = 0; i<dataSound.length; i++) {
298
299         String hexa = String.format("%02X", dataSound[i]); // print hexadecimal the
300 content of the byte array
301         System.out.print("Input: decimal: " + dataSound[i] + ", unsigned: " + Byte.
302 toUnsignedInt(dataSound[i]) + " and the hexa: " + hexa + ", ");
303         // get nibbles
304         int maskLow = 0x0F;
305         int maskHigh = 0xF0;
306         int nibbleLow = dataSound[i] & maskLow; // D[i] = x[i] - x[i-1]
307         int nibbleHigh = (dataSound[i] & maskHigh)>>4; // D[i-1] = x[i-1] - x[i-2]
308
309         // differences
310         int diffHigh = (nibbleHigh - 8)*step;
311         int diffLow = (nibbleLow - 8)*step;
312
313         // get samples
314         int sampleFirst = init + diffHigh;
315         int sampleSecond = sampleFirst + diffLow;

```

```

312     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);
313     init = sampleSecond;
314
315     // check range
316     int max8 = (int)(Math.pow(2,7)) - 1;
317     int min8 = -(int)(Math.pow(2,7));
318     int[] samples = {sampleFirst, sampleSecond};
319     for (int j=0; j< samples.length; j++) {
320         if (samples[j]>max8) samples[j] = max8;
321         else if (samples[j]<min8) samples[j] = min8;
322     }
323
324     // write data to files
325     try {
326         writerDiffSamples.write(diffHigh + "\n" + diffLow + "\n");
327         writerSamples.write(samples[0] + "\n" + samples[1] + "\n");
328     }
329     catch (Exception x) {
330         System.out.println(x);
331         System.out.println("Failed to write data to DPCM file");
332     }
333
334     // write to buffer
335     byte[] decodedSound = new byte[2];
336     decodedSound[0] = (byte)sampleFirst;
337     decodedSound[1] = (byte)sampleSecond;
338     System.out.println(". Output: " + String.format("%02X", decodedSound[0]) + String.
format("%02X", decodedSound[1]));
339     try {
340         bufferSound.write(decodedSound);
341     }
342     catch (Exception x) {
343         System.out.println(x);
344         System.out.println("Decoding DPCM failed");
345     }
346 }
347
348
349
350     return bufferSound.toByteArray();
351
352 }
353
354 private static byte[] adpcm(byte[] dataSound, FileWriter writerDiffSamples,
355                             FileWriter writerSamples, FileWriter writerMean, FileWriter
writerStep) {
356
357     // get the header first
358     int mean = (Byte.toUnsignedInt(dataSound[1])<<8 | Byte.toUnsignedInt(dataSound[0]));
359     // be sure to not preserve the byte sign
360     int meanSigned = (dataSound[1]<<8 | dataSound[0]); // this is wrong. Not sure though?
361     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));
    int step = (Byte.toUnsignedInt(dataSound[3])<<8 | Byte.toUnsignedInt(dataSound[2]));
    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));

    try {
        writerMean.write(meanSigned + "\n");
        writerStep.write(step + "\n");
    }
    catch (Exception x) {
        System.out.println(x);
        System.out.println("Failed to write mean and step files for AQ-DPCM");
    }

    ByteArrayOutputStream bufferSound = new ByteArrayOutputStream();
    int init = meanSigned; // in DPCM we don't know the init value, we assume zero. But
here we have data in the header.
    for (int i = 3; i<dataSound.length; i++) {
        // the sample may be bigger than byte. So you will need 16 bit encoding and store
each int to 2 bytes.
        System.out.print("Input: " + String.format("%02X", dataSound[i]) + ", "); // print
hexadecimal the content of the byte array

        // get nibbles
        int maskLow = 0x0F;
        int maskHigh = 0xF0;
        int nibbleLow = (dataSound[i] & maskLow); // D[i] = x[i] - x[i-1], should be
unsigned
        int nibbleHigh = (dataSound[i] & maskHigh)>>4; // D[i-1] = x[i-1] - x[i-2], should
be unsigned

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

        // get samples (implement recursive formula)
        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)*step + ", " + (nibbleLow-8)*step + " and samples: " + sampleFirst +
", " + sampleSecond);
        init = sampleSecond;

        // check range
        int max16 = (int)(Math.pow(2,15)) - 1;
        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]));
405
406 // write data to files
407 try {
408     writerDiffSamples.write(diffHigh + "\n" + diffLow + "\n");
409     writerSamples.write(samples[0] + "\n" + samples[1] + "\n");
410 }
411 catch (Exception x) {
412     System.out.println(x);
413     System.out.println("Failed to write data to AQ-DPCM file");
414 }
415
416 // write to buffer
417 byte[] decodedSound = new byte[4];
418 decodedSound[0] = (byte)(samples[0]>>8); // MSB of sample 15-8
419 decodedSound[1] = (byte)samples[0]; // LSB of sample 7-0
420 decodedSound[2] = (byte)(samples[1]>>8);
421 decodedSound[3] = (byte)samples[1];
422 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]));
423 try {
424     bufferSound.write(decodedSound);
425 }
426 catch (Exception x) {
427     System.out.println(x);
428     System.out.println("Decoding DPCM failed");
429 }
430 }
431
432
433 return bufferSound.toByteArray();
434 }
435 }

```

### 3.3 Obd.java

```

1 package applications;
2 import java.io.File;
3 import java.io.FileWriter;
4 import java.io.BufferedReader;
5 import java.io.InputStream;
6 import java.io.InputStreamReader;
7 import java.io.OutputStream;
8 import java.net.DatagramSocket;
9 import java.net.DatagramPacket;
10 import java.net.Socket;
11 import java.net.InetAddress;
12 import java.nio.charset.StandardCharsets;
13
14 public class Obd {
15
16     private static String[] header = {"01 1F", "01 0F", "01 11", "01 0C", "01 0D", "01 05"};
17
18
19     public static void udpTelemetry(DatagramSocket socket, InetAddress hostAddress, int
serverPort, String requestCode) {
20
21         byte[] rxbuffer = new byte[16];
22         DatagramPacket receivePacket = new DatagramPacket(rxbuffer, rxbuffer.length);
23

```

```

24     for (int i = 0; i < header.length; i++) {
25
26         // TX
27         String completeCode = (requestCode + "OBD=" + header[i]);
28         byte[] txbuffer = completeCode.getBytes();
29         DatagramPacket sendPacket = new DatagramPacket(txbuffer, txbuffer.length,
hostAddress, serverPort);
30         System.out.println("Complete request: " + completeCode);
31         try {
32             socket.send(sendPacket);
33         }
34         catch (Exception x) {
35             // x.printStackTrace(); // a more detailed diagnostic call
36             System.out.println(x);
37             System.out.println("OBD vehicle application TX failed");
38         }
39         long timeBefore = System.currentTimeMillis();
40
41         // RX
42         try{
43             socket.setSoTimeout(3000);
44             socket.receive(receivePacket);
45             String message = new String(rxbuffer, StandardCharsets.US_ASCII);
46             System.out.println("Ithaki responded via UDP with: " + message);
47             System.out.println("Ithaki UDP time response: " + (System.currentTimeMillis()-
timeBefore)/(float)1000 + " seconds");
48
49             int[] values = parser(message);
50             formula(values[0], values[1], header[i]);
51         }
52         catch (Exception x) {
53             System.out.println(x);
54             System.out.println("RX UDP vehicle failed");
55         }
56     }
57
58 }
59
60 public static float tcpTelemetry(Socket socket, FileWriter writerVehicle) {
61
62     float engineTime = 0;
63
64     try {
65         InputStream in = socket.getInputStream();
66         OutputStream out = socket.getOutputStream();
67         BufferedReader bf = new BufferedReader(new InputStreamReader(in)); // wrapper on
top of the wrapper as java docs recommends
68
69         for (int i = 0; i < header.length; i++) {
70             out.write((header[i] + "\r").getBytes());
71             //out.flush();
72             long timeBefore = System.currentTimeMillis();
73             System.out.println("Created TCP socket and set output stream... Waiting for
response");
74
75             System.out.println("Header: " + header[i]);
76             String data = bf.readLine();
77             System.out.println("Ithaki responded via TCP with: " + data);
78             System.out.println("Ithaki TCP time response: " + (System.currentTimeMillis()-
timeBefore)/(float)1000 + " seconds");

```



```

79         int[] values = parser(data);
80         float value = formula(values[0], values[1], header[i]);
81         writerVehicle.write(value + " ");
82         if (header[i] == "01 1F") engineTime = value;
83     }
84
85     writerVehicle.write("\n");
86 }
87
88 catch (Exception x) {
89     System.out.println(x);
90     System.out.println("Oops... Vehicle OBD TCP failed");
91 }
92
93
94 return engineTime;
95 }
96
97 private static float formula(int first, int second, String header) {
98     float value = 0;
99     switch (header) {
100         case "01 1F":
101             int engineRunTime = first*256 + second;
102             System.out.println("Engine run time: " + engineRunTime);
103             value = engineRunTime;
104             break;
105
106         case "01 0F":
107             int intakeAirTemp = first - 40;
108             System.out.println("Intake Air Temperature: " + intakeAirTemp);
109             value = intakeAirTemp;
110             break;
111
112         case "01 11":
113             float throttlePos = (first*100)/(float)255;
114             System.out.println("Throttle position: " + throttlePos);
115             value = throttlePos;
116             break;
117
118         case "01 0C":
119             float engineRpm = ((first*256) + second)/(float)4;
120             System.out.println("Engine RPM: " + engineRpm);
121             value = engineRpm;
122             break;
123
124         case "01 0D":
125             int speed = first;
126             System.out.println("Vehicle speed: " + speed);
127             value = speed;
128             break;
129
130         case "01 05":
131             int coolantTemp = first - 40;
132             System.out.println("Coolant Temperature: " + coolantTemp);
133             value = coolantTemp;
134             break;
135
136         default:
137             System.out.println("Something went wrong calculating formula for vehicle stats
");

```

```

138     }
139     System.out.println();
140     return value;
141 }
142
143
144 private static int[] parser(String data) {
145     String byte1 = data.substring(6,8);
146     // how to convert hexadecimal string to int?
147     int first = Integer.parseInt(byte1, 16);
148     System.out.print("Parsing the data: 1st byte: " + byte1 + " and as an integer: " +
first);
149     String byte2 = "";
150     int second = 0;
151     if (data.length() > 8) {
152         byte2 = data.substring(9,11);
153         second = Integer.parseInt(byte2, 16);
154         System.out.print(", 2nd byte: " + byte2 + " and as an integer: " + second);
155     }
156     System.out.println();
157
158     int[] temp = {first, second};
159     return temp;
160 }
161 }

```

### 3.4 Copter.java

```

1 package applications;
2
3 import java.io.FileWriter;
4 import java.io.BufferedReader;
5 import java.io.ByteArrayOutputStream;
6 import java.io.InputStream;
7 import java.io.InputStreamReader;
8 import java.io.OutputStream;
9 import java.net.DatagramSocket;
10 import java.net.DatagramPacket;
11 import java.net.Socket;
12 import java.net.InetAddress;
13 import java.nio.charset.StandardCharsets;
14 import java.io.ByteArrayOutputStream;
15
16 public class Copter {
17     public static String udpTelemetry(DatagramSocket socket, InetAddress hostAddress, int
serverPort, FileWriter writerCopter) {
18         // TX
19         // open ithakicopter.jar
20
21
22         //RX only
23         byte[] rxbuffer = new byte[128];
24         DatagramPacket receivePacket = new DatagramPacket(rxbuffer, rxbuffer.length);
25
26         long timeBefore = System.currentTimeMillis();
27         String telemetry = new String();
28         try{
29             socket.setSoTimeout(3000);
30             socket.receive(receivePacket);
31             telemetry = new String(rxbuffer, StandardCharsets.US_ASCII);

```

```

32 //System.out.print("Time repsonse: " + (System.currentTimeMillis() - timeBefore)/(
float)1000);
33 System.out.println("Received data via UDP: " + telemetry);
34
35 String[] tokensMotor = telemetry.split("LMOTOR=");
36 String[] tokensAltitude = telemetry.split("ALTITUDE=");
37 String[] tokensTemp = telemetry.split("TEMPERATURE=");
38 String[] tokensPress = telemetry.split("PRESSURE=");
39 writerCopter.write(tokensMotor[1].substring(0, 3) + " ");
40 writerCopter.write(tokensAltitude[1].substring(0, 3) + " ");
41 writerCopter.write(tokensTemp[1].substring(1, 6) + " ");
42 writerCopter.write(tokensPress[1].substring(0, 7) + "\n");
43 }
44 catch (Exception x) {
45     System.out.println(x);
46     System.out.println("RX UDP ithakicopter failed");
47 }
48 return telemetry;
49
50 }
51
52 public static String tcpTelemetry(InetAddress hostAddress, int target) {
53     String telemetry = "";
54     Socket socket = new Socket();
55     try {
56         socket = new Socket(hostAddress, 38048);
57         InputStream in = socket.getInputStream();
58         OutputStream out= socket.getOutputStream();
59         BufferedReader bf = new BufferedReader(new InputStreamReader(in)); // wrapper on
top of the wrapper as java docs recommends
60         ByteArrayOutputStream bos = new ByteArrayOutputStream();
61
62         String command = "AUTO FLIGHTLEVEL=" + target + " LMOTOR=" + target + " RMOTOR=" +
target + " PILOT \r\n";
63         //System.out.print("Request: " + command);
64         out.write(command.getBytes());
65         out.flush();
66
67         //in.skipNBytes(427);
68         for (int i = 0; i < 14; i++) {
69             bos.write((bf.readLine() + "\n").getBytes());
70         }
71         String data = new String(bos.toByteArray(), StandardCharsets.US_ASCII);
72         //System.out.println("Received data via TCP: " + data);
73
74         String[] tokens = data.split("\n");
75         // take only the useful data and skip the info ithaki sent
76         telemetry = tokens[13];
77     }
78     catch (Exception x) {
79         System.out.println(x);
80         System.out.println("Oops... Ithakicopter TCP failed");
81     }
82     try {
83         socket.close();
84     }
85     catch (Exception x) {
86         System.out.println(x);
87         System.out.println("Failed to close socket for ithakicopter TCP");
88     }
89 }

```

```

89     return telemetry;
90 }
91
92
93 /*
94  * tcpTelemetry function for the TX and udpTelemetry for RX. The way that these two
95  * functions are implemented force the autopilot
96  * to be used with a combination of these two. We want to send a command only if it is
97  * needed and we want to listen all the time
98  * to get feedback.
99  */
100 public static void autopilot(DatagramSocket listen, InetAddress hostAddress, int
serverPort, Socket send, int lowerBound, int higherBound) {
101
102     lowerBound = Math.min(lowerBound, higherBound);
103     higherBound = Math.max(lowerBound, higherBound);
104
105     int target = (lowerBound + higherBound)/2;
106     int motor = -1;
107
108     try {
109         System.out.println("AUTOPILOT: ON");
110         System.out.println("You need to open ithakicopter.jar. Press ENTER to continue...");
111         System.in.read();
112         System.out.println("Press Control-C to exit...");
113         Thread.sleep(1000);
114         for (;;) {
115             if ((motor < (lowerBound)) || (motor > (higherBound))) {
116                 System.out.println("Send packet. Readjust...");
117                 tcpTelemetry(hostAddress, target);
118             }
119
120             String telemetry = Copter.udpTelemetry(listen, hostAddress, serverPort, null);
121             String[] tokens = telemetry.split("LMOTOR=");
122             motor = Integer.parseInt(tokens[1].substring(0,3)); // get motor values
123
124             System.out.println("Parsed motor values: " + motor);
125         }
126     } catch (Exception x) {
127         System.out.println(x);
128         System.out.println("AUTOPILOT failed");
129     }
130 }
131
132 }

```

## 4 Plots

```

1 from scipy.stats import norm
2 import matplotlib.pyplot as plt
3 import matplotlib.mlab as mlab
4 import pandas as pd
5 from numpy import genfromtxt
6 import numpy as np
7
8
9 # data = pd.read_csv('../logs/echo_samples_delay.csv', sep=',', header=None)
10 # data.plot(kind='bar')

```

```

11 # plt.ylabel('frequency')
12 # plt.xlabel('Number of packets')
13 # plt.title('Histogram response time')
14 # plt.show()
15
16 #plt.hist(rtt,histtype = 'bar', bins='auto', density=1, alpha=0.7)
17 # print("Length of the array is: " + str(len(rtt)))
18
19
20 # Response time diagram
21 x = genfromtxt('../logs/session2/echo_samples_delay.txt', delimiter=',')
22 x = genfromtxt('../logs/session2/echo_samples_no_delay.txt', delimiter=',')
23 x = genfromtxt('../logs/session2/echo_throughput_delay.txt', delimiter=',')
24 x = genfromtxt('../logs/session2/echo_throughput_no_delay.txt', delimiter=',')
25 x = genfromtxt('../logs/session1/AQFsamples.txt')
26 x = genfromtxt('../logs/session2/Fsamples.txt')
27 x = genfromtxt('../logs/session2/Tsamples.txt')
28 x = genfromtxt('../logs/session2/second_clip/aqdpcm_mean.txt')
29 x = genfromtxt('../logs/session2/second_clip/aqdpcm_step.txt')
30 x = genfromtxt('../logs/second_clip/aqdpcm_step.txt')
31 plt.subplot(2,1,1)
32 plt.plot(x, 'm')
33 plt.xlabel('Number of samples', fontsize=12)
34 plt.ylabel('Amplitude', fontsize=12)
35 plt.grid(True)
36
37 plt.subplot(2,1,2)
38 plt.xlabel('Number of samples', fontsize=12)
39 plt.ylabel('Amplitude', fontsize=12)
40 plt.grid(True)
41 plt.plot(x, 'm')
42 plt.xlim(100, 200)
43 plt.show()
44
45
46 #Retransmission timeout plot
47 data = genfromtxt('../logs/session2/rto.txt', delimiter=' ')
48 rtt = data[1:,0]
49 srtt = data[1:,1]
50 rttd = data[1:,2]
51 rto = data[1:,3]
52 plt.plot(rtt, label = "RTT")
53 plt.plot(srtt, label = "SRTT")
54 plt.plot(rttd, label = "RTTd")
55 plt.plot(rto, label = "RTO")
56 plt.xlabel('Number of packets', fontsize=12)
57 plt.ylabel("Time response", fontsize=12)
58 plt.legend()
59 plt.show()
60
61 # Copter
62 data = genfromtxt('../logs/session2/copter_2nd_run/copter_info.txt', delimiter=' ')
63 rtt = data[1:,0]
64 srtt = data[1:,1]
65 rttd = data[1:,2]
66 rto = data[1:,3]
67 plt.plot(rtt, label = "MOTOR")
68 plt.plot(srtt, label = "ALTITUDE")
69 plt.plot(rttd, label = "TEMPERATURE")
70 plt.plot(rto, label = "PRESSURE")

```

```

71 plt.xlabel('Number of packets', fontsize=12, labelpad=10)
72 plt.ylabel("Data", fontsize=12, labelpad=10)
73 plt.legend()
74 plt.grid(True);
75 plt.yticks(np.arange(0, 1200, 50))
76 plt.show()
77
78 #Vehicle
79 data = genfromtxt('../logs/car_telemetry.txt', delimiter=' ')
80 rtt = data[0:,0]
81 srtt = data[0:,1]
82 rtttd = data[0:,2]
83 rto = data[0:,3]
84 s = data[0:, 4]
85 t = data[0:, 5]
86 plt.subplot(211)
87 plt.plot(rtt, label = "Engine run time")
88 plt.plot(rto, label = "Engine RPM")
89 plt.xlabel('Number of packets', fontsize=12, labelpad=10)
90 plt.ylabel("Data", fontsize=12, labelpad=10)
91 plt.legend()
92 plt.grid(True);
93 plt.subplot(212)
94 plt.plot(srtt, label = "Intake air temperature")
95 plt.plot(rtttd, label = "Throttle position")
96 plt.plot(s, label = "Vehicle speed")
97 plt.plot(t, label = "Coolant temperature")
98 plt.xlabel('Number of packets', fontsize=12, labelpad=10)
99 plt.ylabel("Data", fontsize=12, labelpad=10)
100 plt.legend()
101 plt.grid(True);
102 #plt.yticks(np.arange(0, 2500, 25))
103 plt.show()

```