Δίκτυα Υπολογιστών 1

. Source Code .

Κωνσταντίνος Σαμαράς-Τσακίρης, 7972

21/12/2015

Όλος ο κώδικας βρίσκεται στο Github:
https://github.com/Oblynx/networksProject

1 userApplication [main]

```
1  import java.util.*;
2
3  public class userApplication {
4    public static void main(String[] param){
5      int serial= 10, echoMsgTime=6*60000, testsSucceeded=0, i, speed= 8000, timeout= 30*60000;
6      VirtualModem vm= new VirtualModem();
7      ArrayList<Packet> echoes;
8      vm.RXsetup(speed, timeout);
9
10     //Get packets for 4 minutes
11     echoes= vm.echoPacketRX("E6996\r", echoMsgTime, serial);
12     for (i=0; i< echoes.size(); i++) if (echoes.get(i).incomplete) break;
13     if (i == echoes.size() && echoes.size() > 0) { testsSucceeded++; System.out.println("Test1
           finish"); }
14     else System.out.println("Test1 INCOMPLETE!");
15     //vm.close(); vm.RXsetup(speed, timeout);
16     if(!vm.imageRX("M4660\r", serial).incomplete) { testsSucceeded++; System.out.println("
           Test2 finish"); }
17     else System.out.println("Test2 INCOMPLETE!");
18     vm.close(); vm.RXsetup(speed, timeout);
19     if(!vm.imageRX("G9539\r", serial).incomplete) { testsSucceeded++; System.out.println("
           Test3 finish"); }
20     else System.out.println("Test3 INCOMPLETE!");
21     vm.close(); vm.RXsetup(speed, timeout);
22     if(!vm.gpsMapRX("P5987R=1004040\r", serial, 10).incomplete) { testsSucceeded++; System.out
           .println("Test4 finish"); }
23     else System.out.println("Test4 INCOMPLETE!");
24     vm.close(); vm.RXsetup(speed, timeout);
25     echoes= vm.arqRX("Q4761\r", "R7581\r", echoMsgTime, serial);
26     for (i=0; i< echoes.size(); i++) if (echoes.get(i).incomplete) break;
27     if (i == echoes.size() && echoes.size() > 0) { testsSucceeded++; System.out.println("Test5
           finish"); }
28     else System.out.println("Test5 INCOMPLETE!");
29
30     System.out.println("\t-->  Tests succedeed: "+testsSucceeded+"/5  <--");
31     vm.close();
32   }
33 }
```

2 Packet

```
1  import java.io.File;
2  import java.io.IOException;
3  import java.nio.file.*;
4  import java.nio.file.StandardOpenOption;
5  import java.util.ArrayList;
6  import java.util.Arrays;
7  import java.util.List;
8
9  //! The main communication data object
10 public class Packet {
11   Packet() { data= new ArrayList<Byte>(); }
12   Packet(ArrayList<Byte> d) { data= new ArrayList<Byte>(d); }
13
14   //! Time between getting the first and last byte of the package
15   public long rxTime() { return startTime-endTime; }
16   //! Log this packet's metadata to file
17   public void log(Path path){
18     List<String> log= Arrays.asList(String.format("%d;%d;%d;%d;%b", startTime,endTime,
         responseTimeMillis,retries,incomplete));
19     StandardOpenOption option= (new File(path.toString()).exists())? StandardOpenOption.APPEND
         : StandardOpenOption.CREATE;
20     try { Files.write(path, log,StandardOpenOption.WRITE, option);
21     } catch (IOException e) { e.printStackTrace(); }
22   }
23
24   public ArrayList<Byte> data;
25   public long startTime=0, endTime=0;
26   //! Time between: before writing code to modem and after receiving first byte back (whether
       that byte belongs to package or not)
27   public long responseTimeMillis=0;
28   //! How many times the FCS check failed
29   public int retries=0;
30   //! Whether the package has been fully received
31   public boolean incomplete= false;
32 }
```

3 VirtualModem

```
1  import ithakimodem.Modem;
2  import java.io.IOException;
3  import java.io.UnsupportedEncodingException;
4  import java.nio.file.*;
5  import java.time.Instant;
6  import java.util.*;
7
8  public class VirtualModem {
9    //! Dial Ithaki
10   public void RXsetup(int speed, int timeout){
11     modem= new Modem();
12     modem.setSpeed(speed);
13     modem.setTimeout(timeout);
14     modem.write("ATd2310ithaki\r".getBytes());
15   }
16   //! Echoes what the modem says to the console
17   public void echoModem(String code){
18     getPacket(code, new ArrayList<Byte>(), new ArrayList<Byte>(), 100);
19   }
20   //! Request echo packages continuously, until durationMillis time has passed.
21   public ArrayList<Packet> echoPacketRX(String code, long durationMillis, int serial){
22     ArrayList<Packet> packets= new ArrayList<Packet>();
23     long startTime= System.currentTimeMillis();
24     while(System.currentTimeMillis()-startTime < durationMillis){
25       Packet packet= getPacket(code, echoStart, echoEnd, 100);
26       processEchoPacket(packet, serial);
27       packets.add(packet);
28     }
29     return packets;
30   }
31   //! Request 1 image from Ithaki and store it to file
32   public Packet imageRX(String code, int serial){
33     System.out.println("Image transfer begun");
34     Packet packet= getPacket(code, jpgStart, jpgEnd, 120*1024);
35     String imgName= (code.charAt(0) == 'M')? "image":(code.charAt(0)=='G')? "noise": "map";
36     if (!packet.incomplete){
37       System.out.println("Image transfer COMPLETE!");
38       processImage(packet, imgName, serial);
39     } else {
40       System.out.println("TIMEOUT! Image transfer FAILED!");
41     }
42     return packet;
43   }
44   //! Request a GPS packet, calculate coordinates and request annotated map
45   public Packet gpsMapRX(String code, Integer imgIdx, int secBetweenPos){
46     System.out.println("GPS receiving");
47     Packet packet= getPacket(code, gpsStart, gpsEnd, 100);
48     if (packet.incomplete) {
49       System.out.println("Error! Packet transfer TIMEDOUT!");
50       throw new RuntimeException();
51     }
52     System.out.println("GPS RECEIVED!");
53     String posCode= positionFromGPS(packet, code, secBetweenPos);
54     System.out.println("Generated code: --> "+posCode+"\nGetting map...");
55     return imageRX(posCode, imgIdx);
56   }
57   //! Implement ARQ mechanism to countermeasure transmission errors
58   public ArrayList<Packet> arqRX(String ack, String nack, long durationMillis, int serial){
59     ArrayList<Packet> packets= new ArrayList<Packet>();
60     long startTime= System.currentTimeMillis();
61     while(System.currentTimeMillis()-startTime < durationMillis){
62       int retry= 0;
63       long start,end;
64       Packet packet= getPacket(ack, echoStart, echoEnd, 100);
65       start= packet.startTime; end= packet.endTime;
66       //If transmission error, request again...
67       while (errorARQ(packet)){
68         packet= getPacket(nack, echoStart, echoEnd, 100);
69         end= packet.endTime;
70         retry++;
71       }
```

```
72        packet.retries= retry;
73        packet.startTime= start; packet.endTime= end;
74        //Got correct package
75        processARQ(packet, retry, serial);
76        packets.add(packet);
77      }
78      return packets;
79    }
80    //! Cleanup resources
81    public void close(){ modem.close(); }
82
83    // $$$$$  PRIVATE  $$$$$
84    private void processEchoPacket(Packet packet, Integer serial){
85      if(!packet.incomplete){
86        StringBuffer output= new StringBuffer();
87        for(byte b: packet.data) output.append((char)b);
88        Path path= Paths.get("./log/echoes"+serial.toString()+".log");
89        packet.log(path);
90        //System.out.println("Echo packet received");
91      }
92    }
93    private void processImage(Packet packet, String imgName, Integer serial){
94      if(!packet.incomplete){
95        Path path= Paths.get("./img/"+imgName+serial.toString()+".jpg");
96        Byte[] log= new Byte[packet.data.size()];
97        log= packet.data.toArray(log);
98        try { Files.write(path, toPrimitives(log), StandardOpenOption.CREATE);
99        } catch (IOException e) { e.printStackTrace(); }
100       System.out.println("Image saved to file#"+serial+". Timestamp: "+Instant.now());
101     }
102   }
103   //! Parse GPS packet and get string of position codes with positions > 4secs apart
104   private String positionFromGPS(Packet packet, String code, int secBetweenPos){
105     StringBuffer sigcode= new StringBuffer(), curr= new StringBuffer();
106     ArrayList<StringBuffer> positionBufs= new ArrayList<StringBuffer>();
107     boolean gettingPos= false;
108     //Get all the GPGGA lines in positionBufs
109     for(byte b: packet.data){
110       if(sigcode.length() < 6) sigcode.append((char)b);
111       else{
112         sigcode.deleteCharAt(0);
113         sigcode.append((char)b);
114         if(sigcode.toString().equals(gpsPosHeader)){
115           gettingPos= true;
116           positionBufs.add(new StringBuffer());
117           curr= positionBufs.get(positionBufs.size()-1);
118         }
119         if(gettingPos) curr.append((char)b);
120         if(sigcode.substring(5).equals("\r")){
121           gettingPos= false;
122         }
123       }
124     }
125   System.out.println("Relevant lines: "+positionBufs.size());
126     //Extract positions from positionBuf lines
127     String[] positions= new String[9];
128     int posIdx=0;
129     boolean firstTime= true;
130     //Timestamp of each GPS signal in seconds
131     int[] time= new int[2];
132     for(StringBuffer buf: positionBufs){
133       if(posIdx >= 9) break;
134       String[] parts= buf.toString().split(",");
135       time[1]= (int)Float.parseFloat(parts[1]);
136       if (firstTime || time[1]-time[0] > secBetweenPos){
137         firstTime= false;
138         time[0]= time[1];
139         String latitude= parts[2], longitude= parts[4];
140         /*float latFrac= Float.parseFloat(latitude), longFrac= Float.parseFloat(longitude);
141       System.out.println(String.format("%.2f", latFrac)+" "+String.format("%.2f", longFrac));
142         latFrac/= 100; longFrac/= 100;.
143         int latDeg= (int)latFrac, longDeg= (int)longFrac;
144         latFrac-= latDeg; longFrac-= longDeg;
```

```
145         latFrac*= 60; longFrac*= 60;
146         int latMin= (int)(latFrac), longMin= (int)(longFrac);
147         latFrac*= 60; longFrac*= 60;
148         int latSec= Math.round(latFrac%60), longSec= Math.round(longFrac%60);*/
149 
150         int latDeg= Integer.parseInt(latitude.substring(0,2)), longDeg= Integer.parseInt(
                longitude.substring(1,3));
151         int latMin= Integer.parseInt(latitude.substring(2,4)), longMin= Integer.parseInt(
                longitude.substring(3,5));
152         int latSec= (int)(Integer.parseInt(latitude.split("\\.")[1].substring(0,2))*0.6);
153         int longSec= (int)(Integer.parseInt(longitude.split("\\.")[1].substring(0,2))*0.6);
154         //Create a position-ful gps_request_code
155         positions[posIdx++]= "T="+String.format("%02d", longDeg)+String.format("%02d", longMin
                )+
156             String.format("%02d", longSec)+String.format("%02d", latDeg)+String.format("%02d",
                    latMin)+
157             String.format("%02d", latSec);//+"\r";
158       }
159     }
160   System.out.println("T-codes: "+posIdx);
161     //Buffer to concatenate all the position codes together: <code> T=... T=... ...\r
162     StringBuffer concatPos= new StringBuffer();
163     concatPos.append(code)/*.deleteCharAt(concatPos.length()-1);*/.delete(5, concatPos.length
            ());
164     for(int i=0; i<posIdx; i++) concatPos.append(positions[i]);
165     concatPos.append("\r");
166     return concatPos.toString();
167   }
168   private boolean errorARQ(Packet packet){
169     StringBuffer fcsBuf= new StringBuffer();
170     byte[] hex= new byte[16];
171     if(packet.data.size() != 58){
172       System.out.println("[errorARQ]: Error! Packet size= "+packet.data.size());
173       return true;
174     } else {
175       //<31bytes>HEX FCS<6bytes>
176       for(int i=31; i< 31+16; i++) hex[i-31]= packet.data.get(i);
177       for(int i=49; i<52; i++) fcsBuf.append( (char) ((byte)packet.data.get(i)) );
178       //Parse fcs
179       int fcs= Integer.parseInt(fcsBuf.toString());
180       //Calculate HEX xor
181       int fcsCheck= fcs(hex);
182       return fcs != fcsCheck;
183     }
184   }
185   public int fcs(byte[] hex){
186     int fcsCheck= (int)hex[0];
187     for(int i=1; i<16; i++) fcsCheck^= (int)hex[i];
188     return fcsCheck;
189   }
190   private void processARQ(Packet packet, int retries, Integer serial){
191     if(!packet.incomplete){
192       StringBuffer output= new StringBuffer();
193       for(byte b: packet.data) output.append((char)b);
194       Path path= Paths.get("./log/arques"+serial.toString()+".log");
195       packet.log(path);
196     }
197   }
198 
199   //! Write and read byte streams from the modem. Calculate response time and package RX time.
200   //!   Special use if (start||end) isEmpty: show all data received to console
201   private Packet getPacket(String code, ArrayList<Byte> start, ArrayList<Byte> end, int
          capacity){
202     ArrayList<Byte> sigStart= new ArrayList<Byte>(), sigEnd= new ArrayList<Byte>();
203     ByteFlag mdk= new ByteFlag();
204     boolean packetStart= false, packetIn= false, packetEnd= false;
205     Packet packet= new Packet();
206     mdk.terminate= false;
207     long sendTime= System.currentTimeMillis();
208     modem.write(code.getBytes());
209     //Loop until the *end delimiter* has been received
210     while(!mdk.terminate){
211       mdk= readByte();
```

```
212         if (packet.responseTimeMillis <= 0) packet.responseTimeMillis= System.currentTimeMillis
                ()-sendTime;
213         if (!mdk.terminate && !(start.isEmpty() || end.isEmpty())){
214           if(start.size() == 0) packet.data.add((byte)mdk.k);
215           else{
216             // Update packet delimiter buffer
217             if(sigStart.size() < start.size()) sigStart.add((byte)mdk.k);
218             else {
219               sigStart.remove(0);
220               sigStart.add((byte)mdk.k);
221             }
222             if(sigEnd.size() < end.size()) sigEnd.add((byte)mdk.k);
223             else {
224               sigEnd.remove(0);
225               sigEnd.add((byte)mdk.k);
226             }
227             // Signal accordingly
228             boolean oldpacketStart= packetStart, oldpacketEnd= packetEnd;
229             packetStart= sigStart.equals(start);
230             packetEnd= sigEnd.equals(end);
231             // On packet start...
232             if (!oldpacketStart && packetStart){
233               packetIn= true;
234               //Only record the first time this packet reaches here (ARQ)
235               if (packet.startTime <= 0) packet.startTime= System.currentTimeMillis();
236               packet.data.ensureCapacity(capacity);
237               for(int i=0; i<start.size()-1; i++) packet.data.add(start.get(i));
238             }
239             // While packet is being transmitted...
240             if (packetIn) packet.data.add((byte)mdk.k);
241             // On packet end
242             if (!oldpacketEnd && packetEnd){
243               packetIn= false;
244               packet.endTime= System.currentTimeMillis();
245               mdk.terminate= true;
246             }
247           }
248         } else if (!mdk.terminate) System.out.print((char)mdk.k);   // start/end isEmpty
249         else packet.incomplete= true;
250       }
251       return packet;
252     }
253
254     // $$$$$  Utils  $$$$$
255     private byte[] toPrimitives(Byte[] oBytes){
256         byte[] bytes = new byte[oBytes.length];
257         for(int i = 0; i < oBytes.length; i++) { bytes[i] = oBytes[i]; }
258         return bytes;
259     }
260     private ByteFlag readByte(){
261       ByteFlag data= new ByteFlag();
262       try{
263         data.k= modem.read();
264         if (data.k==-1) data.terminate= true;
265       } catch(Exception e){ data.terminate= true; }
266       return data;
267     }
268     private static class ByteFlag{
269       public int k= 0;
270       public boolean terminate= false;
271     }
272
273     // $$$$$ Members $$$$$
274     @SuppressWarnings("serial")
275     private ArrayList<Byte> jpgStart= new ArrayList<Byte>() {{add((byte)0xFF); add((byte)0xD8)
            ;}};
276     @SuppressWarnings("serial")
277     private ArrayList<Byte> jpgEnd  = new ArrayList<Byte>() {{add((byte)0xFF); add((byte)0xD9)
            ;}};
278     @SuppressWarnings("serial")
279     private ArrayList<Byte> echoStart= new ArrayList<Byte>() {{ try{ for(byte b: "PSTART".
            getBytes("US-ASCII")) add(b); }
280                                 catch (UnsupportedEncodingException e) {e.printStackTrace();}
```

```
                                        }};
281     @SuppressWarnings("serial")
282     private ArrayList<Byte> echoEnd= new ArrayList<Byte>()   {{ try{ for(byte b: "PSTOP".
            getBytes("US-ASCII")) add(b); }
283                                     catch (UnsupportedEncodingException e) {e.printStackTrace();}
                                        }};
284     @SuppressWarnings("serial")
285     private ArrayList<Byte> gpsStart= new ArrayList<Byte>()
286       {{ try{ for(byte b: "START ITHAKI GPS TRACKING\r\n".getBytes("US-ASCII")) add(b); }
287          catch (UnsupportedEncodingException e) {e.printStackTrace();} }};
288     @SuppressWarnings("serial")
289     private ArrayList<Byte> gpsEnd= new ArrayList<Byte>()
290       {{ try{ for(byte b: "STOP ITHAKI GPS TRACKING\r\n".getBytes("US-ASCII")) add(b); }
291          catch (UnsupportedEncodingException e) {e.printStackTrace();} }};
292     private String gpsPosHeader= "$GPGGA";
293     private Modem modem;
294 }
```