```java
/*
 * Copyright (C) 2016 redxef.
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with this library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
 * MA 02110-1301  USA
 */
package powerspy.client;

import java.awt.event.ActionEvent;
import java.io.IOException;
import javax.swing.Timer;
import powerspy.baselib.*;
import static powerspy.baselib.IODefs.*;

/**
 *
 * @author redxef
 */
public class Controller extends Thread {

        private final Frame f;
        private final Timer update_interval;
        private PSInputStream is;
        private PSOutputStream os;

        private boolean keep_alive;

        private int current;
        private int real_power;
        private int apparent_power;
        private int reactive_power;
        private int raw_current;
        private int offs;
        private int raw_supply;

        /**
         * Constructs a new Controller for interacting with PowerSpy.
         *
         * @param f the Frame to link with
         */
        public Controller(Frame f)
        {
                keep_alive = false;
                this.f = f;
```

```
57                  is = null;
58                  os = null;
59                  update_interval = new Timer(500, (ActionEvent e) -> {
60                          doUpdate();
61                  });
62          }
63
64          /**
65           * Sets a new InputStream to read the data from.
66           *
67           * @param is the new InputStream
68           */
69          public synchronized void setPSInputStream(PSInputStream is)
70          {
71                  this.is = is;
72          }
73
74          /**
75           * Returns the currently installed InputStream
76           *
77           * @return the InputStream
78           */
79          public synchronized PSInputStream getPSInputStream()
80          {
81                  return is;
82          }
83
84          /**
85           * Sets the new OutputStream to write data to.
86           *
87           * @param os the new OutputStream
88           */
89          public synchronized void setPSOutputStream(PSOutputStream os)
90          {
91                  this.os = os;
92          }
93
94          /**
95           * Returns the currently installed OutputStream.
96           *
97           * @return the OutputStream
98           */
99          public synchronized PSOutputStream getPSOutputStream()
100         {
101                 return os;
102         }
103
104         private synchronized void doUpdate()
105         {
106                 System.out.println("curr: " + current);
107                 System.out.println("real: " + real_power);
108                 System.out.println("appa: " + apparent_power);
109                 System.out.println("reac: " + reactive_power);
110                 f.setCurrent((float) current / 1000);
111                 f.setRealPower((float) real_power / 1000);
112                 f.setApparentPower((float) apparent_power / 1000);
113                 f.setReactivePower((float) reactive_power / 1000);
```

```
114                 f.setRawCurrent((float) raw_current);
115                 f.setOffs((float) offs);
116                 f.setRawVoltage((float) raw_supply);
117         }
118
119         @Override
120         public synchronized void start()
121         {
122                 keep_alive = true;
123                 update_interval.start();
124                 super.start();
125         }
126
127         private synchronized char doRun(char read_mode) throws IOException, Packa
128         {
129                 if (is.readPackage()) {
130                         if (read_mode == NONE && is.getDataPacket().isUInt8()) {
131                                 read_mode = (char) is.getDataPacket().readUInt8()
132                         } else if (read_mode != NONE && is.getDataPacket().isInt2
133                                 switch (read_mode) {
134                                         case K_CURRENT:
135                                                 current = is.getDataPacket().read
136                                                 break;
137                                         case K_REALPOWER:
138                                                 real_power = is.getDataPacket().r
139                                                 break;
140                                         case K_APPARENTEPOWER:
141                                                 apparent_power = is.getDataPacket
142                                                 break;
143                                         case K_REACTIVEPOWER:
144                                                 reactive_power = is.getDataPacket
145                                                 break;
146                                         case K_RAWCURRENT:
147                                                 raw_current = is.getDataPacket().
148                                                 break;
149                                         case K_OFFS:
150                                                 offs = is.getDataPacket().readInt
151                                                 break;
152                                         case K_RAWVOLTAGE:
153                                                 raw_supply = is.getDataPacket().r
154                                                 break;
155                                         default:
156                                                 break;
157                                 }
158                                 read_mode = NONE;
159                         }
160                         is.clear();
161                 }
162                 return read_mode;
163         }
164
165         @Override
166         public void run()
167         {
168                 char read_mode = NONE;
169                 while (keep_alive) {
170                         try {
```

```
171                                 read_mode = doRun(read_mode);
172                                 Thread.sleep(20);
173                         } catch (IOException | PackageException | InterruptedExce
174                                 ex.printStackTrace(System.err);
175                                 is.clear();
176                         }
177                 }
178         }
179
180         /**
181          * Terminates this Controller and stops the update interval.
182          */
183         public void terminate()
184         {
185                 keep_alive = false;
186                 update_interval.stop();
187         }
188
189 }
```