```
1  // Annex II
2  // Main window for conducting and analysing the tremor experiments with the
     g.Nautilus accelerometor
3  // Written by Javi Rodriguez and Ben Hesketh
4
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10 using System.Windows;
11 using System.Windows.Controls;
12 using System.Windows.Data;
13 using System.Windows.Documents;
14 using System.Windows.Input;
15 using System.Windows.Media;
16 using System.Windows.Media.Imaging;
17 using System.Windows.Navigation;
18 using System.Windows.Shapes;
19 using Gtec.Gds.Client.API.Wrapper;
20 using Gtec2;
21 using System.Timers;
22 using TFM;
23
24 namespace accelerometer
25 {
26     /// <summary>
27     /// Interaction logic for MainWindow.xaml
28     /// </summary>
29     public partial class MainWindow : Window
30     {
31         AccelAmplifier amp;
32
33         //Vectors created for the required axis
34         List<double> xAccel = new List<double>();
35         List<double> yAccel = new List<double>();
36         List<double> zAccel = new List<double>();
37
38         List<double> addedAxis = new List<double>();//the result of the axis
             added together
39
40         //Vectors created for the fake tremor experiment
41         List<double> noTremorX = new List<double>();
42         List<double> noTremorY = new List<double>();
43         List<double> noTremorZ = new List<double>();
44         List<double> lightTremorX = new List<double>();
45         List<double> lightTremorY = new List<double>();
46         List<double> lightTremorZ = new List<double>();
47         List<double> mediumTremorX = new List<double>();
48         List<double> mediumTremorY = new List<double>();
49         List<double> mediumTremorZ = new List<double>();
50         List<double> heavyTremorX = new List<double>();
51         List<double> heavyTremorY = new List<double>();
52         List<double> heavyTremorZ = new List<double>();
53
54         Timer t = new Timer(200);//timer created
```

```
55
56          /// <summary>
57          /// The main window for the program to run
58          /// </summary>
59          public MainWindow()
60          {
61              //Area for initializing variables
62              InitializeComponent();
63
64              /*
65              for (int i = 0; i < 2000; i++)// optional code to add zeros to the front of the vectors
66              {
67                  xAccel.Add(0);
68                  yAccel.Add(0);
69                  zAccel.Add(0);
70              }*/
71
72              t.Elapsed += (o, e) =>
73              {
74                  try {
75                  System.Windows.Application.Current.Dispatcher.Invoke(() =>
76                  {
77                      xPlotter.Plot(xAccel);
78                      yPlotter.Plot(yAccel);
79                      zPlotter.Plot(zAccel);
80                  });
81                  }catch
82                  {
83
84                  }
85              };
86          }
87
88          /// <summary>
89          /// Actions to be carried out whenever the connect button is clicked. Searches for the accelerometer and amplifier and starts a timer.
90          /// </summary>
91          /// <param name="sender"></param>
92          /// <param name="eArgs"></param>
93          private void connectBTN_Click(object sender, RoutedEventArgs eArgs)
94          {
95              FrameDefinition.softwareEnvironment = FrameDefinition.SoftwareEnvironments.CUSTOM;
96
97              amp = new AccelAmplifier();
98
99              amp.OnNewScopeData += Amp_OnNewScopeData;
100
101             amp.g_searchCompleted += (o, e) =>
102             {
103                 var unfiltered = amp.getUnfilteredConnectedDevices();
104
105                 if (unfiltered.Count == 1 && unfiltered[0] == "simulator")
106                     throw new Exception("no valid amplifier connected");
107
```

```
108                 amp.SelectAmplifier(unfiltered[0]);
109
110                 FrameDefinition.setForCSPWihtNautilus();
111                 PatientManager.workingDirectory =
                      Environment.CurrentDirectory;
112
113                 amp.startAdquiring();//acquistion begins
114                 t.Start();//timer start
115             };
116
117             amp.searchConnectedDevices();
118
119         }
120
121         private void Window_Closing(object sender,
              System.ComponentModel.CancelEventArgs e)
122         {
123             try
124             {
125                 amp.stopAdquiring();
126             } catch//closes the window whenever there is
127             {
128
129             }
130         }
131
132         /// <summary>
133         /// What happens when a new sample is read in from the amp
134         /// </summary>
135         /// <param name="sender"></param>
136         /// <param name="e"></param>
137         private void Amp_OnNewScopeData(object sender, newScopeDataArgs e)
138         {
139             // Optional code to remove the values at time=0 so that the graph
                  remains at the same timescale
140             //xAccel.Add(Math.Max(Math.Min(e.GetData()[0], 1), -1));
141             //xAccel.RemoveAt(0);
142             //yAccel.Add(Math.Max(Math.Min(e.GetData()[1], 1), -1));
143             //yAccel.RemoveAt(0);
144             //zAccel.Add(Math.Max(Math.Min(e.GetData()[2], 1), -1));
145             //zAccel.RemoveAt(0);
146
147             xAccel.Add(e.GetData()[0]);
148             yAccel.Add(e.GetData()[1]);
149             zAccel.Add(e.GetData()[2]);
150         }
151
152         /// <summary>
153         /// Actions to be carrried out whenever the stop button is clicked.
              Typically save to file.
154         /// </summary>
155         /// <param name="sender"></param>
156         /// <param name="e"></param>
157         private void stopBTN_Click(object sender, RoutedEventArgs e)
158         {
159
```

```
160             string filename = Filenametb.Text+".csv";//input from text box.
                  name for each experiment
161
162             string patientfolder = "patient" + Patientnumbertb.Text;// +
                  @"\";//input from text box. patient number. used to create a new
                   directory
163
164             System.IO.Directory.CreateDirectory(@"C:\Users\Bens laptop
                  \Documents\g.tec\acquisition project\" + patientfolder);// new
                  directory created for each patient
165             amp.stopAdquiring();
166
167             //for (int i = 0; i < xAccel.Count; i++) // limit signal
168             //{
169             //    var s = xAccel[i];
170             //    if (s > 0.3) s = 10;
171             //    else s = -10;
172             //    xAccel[i] = s;
173             //}
174
175             for (int i = 0; i<50; i++) //Remove first 50 values to minimize
                  the initial spike
176             {
177                 xAccel.RemoveAt(i);
178                 yAccel.RemoveAt(i);
179                 zAccel.RemoveAt(i);
180             }
181
182             // Save to file within the specified directory for each patient.
                  In the format date, time, axis, experiment name
183             Helper.Savefile(xAccel, @"C:\Users\Bens laptop\Documents\g.tec
                  \acquisition project\" + patientfolder + @"\"+
                  DateTime.Now.ToString("yyyyMMddHHmm") + "acc_x_" + filename);
184             Helper.Savefile(yAccel, @"C:\Users\Bens laptop\Documents\g.tec
                  \acquisition project\" + patientfolder + @"\"+
                  DateTime.Now.ToString("yyyyMMddHHmm") + "acc_y_" + filename);
185             Helper.Savefile(zAccel, @"C:\Users\Bens laptop\Documents\g.tec
                  \acquisition project\" + patientfolder + @"\"+
                  DateTime.Now.ToString("yyyyMMddHHmm") + "acc_z_" + filename);
186         }
187
188         /// <summary>
189         /// Offline post processing functions to be carried out whenever the
              Process button is clicked
190         /// </summary>
191         /// <param name="sender"></param>
192         /// <param name="e"></param>
193         private void process_Click(object sender, RoutedEventArgs e)
194         {
195             /////////////////
196             //// Code for dealing  with fake tremor experiment:
197
198             for (int patientnumber = 101; patientnumber < 106; patientnumber+
                  +)//selection of patient number (101 to 105 represents
                  experiments 1 to 5)
```

```
199             {
200                 string directory = @"C:\Users\Bens laptop\Documents\g.tec
                      \acquisition project\patient" + Convert.ToString
                      (patientnumber) + @"\";
201
202                 List<double> scores = new List<double>();//create a list of
                      scores
203                                                         //int i = 1000;
204
205                 int i = 1000; // initial trimming point set to 1000
                      dataelements
206
207                     //no tremor
208
209                     string keyword = "notremor";
210
211                     List<List<double>> chosenData = SearchAndLoad(directory,
                        keyword);//new list of lists containing the files returned
                         from the search and load function
212
213                     List<double> notremor = Helper.AddAxis(Helper.Sum
                        (Helper.Abs(Helper.Window(chosenData[0])), true, i),
                        Helper.Sum(Helper.Abs(Helper.Window(chosenData[1])), true,
                         i), Helper.Sum(Helper.Abs(Helper.Window(chosenData[2])),
                        true, i));
214                     //^^ choses the X, Y and Z axis (indexed 0, 1, 2) of the
                        desired files based on a keyword. Processes each axis
                        moving ave, abs, sum. Adds each axis. Saves to csv.
                        Creates a new list.
215
216                     double s_notremor = Helper.Score(notremor, false,
                        15000);//calculate scores from 15000 elements (60 secs at
                        fs = 250 Hz)
217
218                     //light tremor
219
220                     string keyword2 = "lighttremor";
221
222                     List<List<double>> chosenData2 = SearchAndLoad(directory,
                        keyword2);//new list of lists containing the files
                        returned from the search and load function
223
224                     List<double> lighttremor = Helper.AddAxis(Helper.Sum
                        (Helper.Abs(Helper.Window(chosenData2[0])), true, i),
                        Helper.Sum(Helper.Abs(Helper.Window(chosenData2[1])),
                        true, i), Helper.Sum(Helper.Abs(Helper.Window(chosenData2
                        [2])), true, i));
225                     //^^ choses the X, Y and Z axis (indexed 0, 1, 2) of the
                        desired files based on a keyword. Processes each axis
                        moving ave, abs, sum. Adds each axis. Saves to csv.
                        Creates a new list.
226
227                     double s_lighttremor = Helper.Score(lighttremor, false,
                        15000);//calculate scores from 15000 elements (60 secs at
                        fs = 250 Hz)
228
```

```
229                     //medium tremor
230
231                     string keyword3 = "mediumtremor";
232
233                     List<List<double>> chosenData3 = SearchAndLoad(directory,
                        keyword3);//new list of lists containing the files
                        returned from the search and load function
234
235                     List<double> mediumtremor = Helper.AddAxis(Helper.Sum
                        (Helper.Abs(Helper.Window(chosenData3[0])), true, i),
                        Helper.Sum(Helper.Abs(Helper.Window(chosenData3[1])),
                        true, i), Helper.Sum(Helper.Abs(Helper.Window(chosenData3
                        [2])), true, i));
236                     //^^ choses the X, Y and Z axis (indexed 0, 1, 2) of the
                        desired files based on a keyword. Processes each axis
                        moving ave, abs, sum. Adds each axis. Saves to csv.
                        Creates a new list.
237
238                     double s_mediumtremor = Helper.Score(mediumtremor, false,
                        15000);//calculate scores from normally 15000 elements (30
                         secs at fs = 250 Hz)
239
240                     //heavytremor
241
242                     string keyword4 = "heavytremor";
243
244                     List<List<double>> chosenData4 = SearchAndLoad(directory,
                        keyword4);//new list of lists containing the files
                        returned from the search and load function
245
246                     List<double> heavytremor = Helper.AddAxis(Helper.Sum
                        (Helper.Abs(Helper.Window(chosenData4[0])), true, i),
                        Helper.Sum(Helper.Abs(Helper.Window(chosenData4[1])),
                        true, i), Helper.Sum(Helper.Abs(Helper.Window(chosenData4
                        [2])), true, i));
247                     //^^ choses the X, Y and Z axis (indexed 0, 1, 2) of the
                        desired files based on a keyword. Processes each axis
                        moving ave, abs, sum. Adds each axis. Saves to csv.
                        Creates a new list.
248
249                     double s_heavytremor = Helper.Score(heavytremor, false,
                        15000);//calculate scores from 15000 elements (60 secs at
                        fs = 250 Hz)
250
251                     //Scoring mechanism adds a score from each experiment to a
                         new list of doubles "scores"
252
253                     scores.Add(patientnumber);
254                     scores.Add(i);
255                     scores.Add(s_notremor);
256                     scores.Add(s_lighttremor);
257                     scores.Add(s_mediumtremor);
258                     scores.Add(s_heavytremor);
259
260                 Helper.Savefile(scores, directory +
                      "trimmedscores1000.csv");//save list of scores in a csv file
```

```
                    in the patient directory
261             }
262
263             ////////////////////
264             //// Code for dealing with patient data
265
266             for (int patientnumber = 1; patientnumber < 6; patientnumber++)//
                  selection of patient number
267             {
268
269                 string directory = @"C:\Users\Bens laptop\Documents\g.tec
                      \acquisition project\patient" + Convert.ToString
                      (patientnumber) + @"\";
270
271                 List<double> scores = new List<double>();//create a list of
                      scores
272
273                 // For loop which incrementally adjusts the initial trimming
                      point of each signal thereby splitting the signals into
                      subsections for internal analysis
274                 for (int i = 1000; i <= 9000; i += 1000)//i = starting point
                      for each measurement subsection
275                 {
276                     //Motion experiment - healthy arm
277
278                     string keyword = "motionh";
279
280                     List<List<double>> chosenData = SearchAndLoad(directory,
                        keyword);//new list of lists containing the files returned
                         from the search and load function
281
282                     List<double> motionh = Helper.AddAxis(Helper.Sum
                        (Helper.Abs(Helper.Window(chosenData[0])), true, i),
                        Helper.Sum(Helper.Abs(Helper.Window(chosenData[1])), true,
                         i), Helper.Sum(Helper.Abs(Helper.Window(chosenData[2])),
                        true, i));
283                     //^^ choses the X, Y and Z axis (indexed 0, 1, 2) of the
                        desired files based on a keyword. Processes each axis
                        moving ave, abs, sum. Adds each axis. Saves to csv.
                        Creates a new list.
284
285                     double s_motionh = Helper.Score(motionh, false, 1000);//
                        calculate scores from x elements, typically set to 7500
                        (30 secs at fs = 250 Hz)
286
287                     //Motion experiment - affected arm
288
289                     string keyword2 = "motiona";
290
291                     List<List<double>> chosenData2 = SearchAndLoad(directory,
                        keyword2);//new list of lists containing the files
                        returned from the search and load function
292
293                     List<double> motiona = Helper.AddAxis(Helper.Sum
                        (Helper.Abs(Helper.Window(chosenData2[0])), true, i),
                        Helper.Sum(Helper.Abs(Helper.Window(chosenData2[1])),
```

```
                    true, i), Helper.Sum(Helper.Abs(Helper.Window(chosenData2
                    [2])), true, i));
294                 //^^ choses the X, Y and Z axis (indexed 0, 1, 2) of the
                    desired files based on a keyword. Processes each axis
                    moving ave, abs, sum. Adds each axis. Saves to csv.
                    Creates a new list.
295
296                 double s_motiona = Helper.Score(motiona, false, 1000);//
                    calculate scores from x elements, typically set to 7500
                    (30 secs at fs = 250 Hz)
297
298                 //Extension experiment - healthy arm
299
300                 string keyword3 = "exth";
301
302                 List<List<double>> chosenData3 = SearchAndLoad(directory,
                    keyword3);//new list of lists containing the files
                    returned from the search and load function
303
304                 List<double> exth = Helper.AddAxis(Helper.Sum(Helper.Abs
                    (Helper.Window(chosenData3[0])), true, i), Helper.Sum
                    (Helper.Abs(Helper.Window(chosenData3[1])), true, i),
                    Helper.Sum(Helper.Abs(Helper.Window(chosenData3[2])),
                    true, i));
305                 //^^ choses the X, Y and Z axis (indexed 0, 1, 2) of the
                    desired files based on a keyword. Processes each axis
                    moving ave, abs, sum. Adds each axis. Saves to csv.
                    Creates a new list.
306
307                 double s_exth = Helper.Score(exth, false, 1000);//
                    calculate scores from x elements, normally 2500 elements
                    (10 secs at fs = 250 Hz)
308
309                 //Extension experiment - affected arm
310
311                 string keyword4 = "exta";
312
313                 List<List<double>> chosenData4 = SearchAndLoad(directory,
                    keyword4);//new list of lists containing the files
                    returned from the search and load function
314
315                 List<double> exta = Helper.AddAxis(Helper.Sum(Helper.Abs
                    (Helper.Window(chosenData4[0])), true, i), Helper.Sum
                    (Helper.Abs(Helper.Window(chosenData4[1])), true, i),
                    Helper.Sum(Helper.Abs(Helper.Window(chosenData4[2])),
                    true, i));
316                 //^^ choses the X, Y and Z axis (indexed 0, 1, 2) of the
                    desired files based on a keyword. Processes each axis
                    moving ave, abs, sum. Adds each axis. Saves to csv.
                    Creates a new list.
317
318                 double s_exta = Helper.Score(exta, false, 1000);//
                    calculate scores from x elements, normally 2500 elements
                    (10 secs at fs = 250 Hz)
319
320                 //Scoring mechanism adds a score from each experiment to a
```

```
                    new list of doubles "scores"
321
322                     scores.Add(patientnumber);
323                     scores.Add(i);
324                     scores.Add(s_motionh);
325                     scores.Add(s_motiona);
326                     scores.Add(s_exth);
327                     scores.Add(s_exta);
328                 }
329                 Helper.Savefile(scores, directory + "trimmedscoresmot.csv");// save scores in a csv file in the patient directory
330             }
331
332         }
333
334         /// <summary>
335         /// Searches through a folder and finds csv files containing the keyword, loads them into vectors
336         /// </summary>
337         /// <param name="basepath"></param>filepath
338         /// <param name="keyword"></param>keyword for the search
339         /// <returns></returns>
340         private List<List<double>> SearchAndLoad(string basepath , string keyword)
341         {
342             var files = System.IO.Directory.GetFiles(basepath);//directory selection
343             var keywordFiles = files.Where(x => x.Contains(keyword)).ToList();//find files containing key words
344             keywordFiles.Sort();//sorted so that its in the order x, y, z - > 0, 1, 2
345             List<List<double>> keywordData = new List<List<double>>();//list of list of doubles
346             keywordData.Add(new List<double>());//one for each axis
347             keywordData.Add(new List<double>());
348             keywordData.Add(new List<double>());
349
350             //for (int fileIdx = 0; fileIdx < keywordFiles.Count(); fileIdx++)// .Count not working properly
351             for (int fileIdx = 0; fileIdx < 3; fileIdx++)//loop hard has been coded to 3 because there is always three axis
352             {
353                 keywordData[fileIdx] = Helper.Loadfile(keywordFiles[fileIdx]);
354             }
355
356             return keywordData;
357         }
358     }
359
360     public class AccelAmplifier : Gtec2.AmpController//base statioon / amplifier communication functions. Sampling rate = 250
361     {
362         private volatile bool searchFinished = false;
363
364         public override GdsClientApiLibraryWrapper.GdsConfiguration createCustomConfiguration(ulong connectionHandle, string
```

```
            serialNumber, FrameDefinition.SoftwareEnvironments environment)
        {
            GNautilusGdsClientApiLibraryWrapper.GdsGNautilusConfiguration
              deviceConfiguration = new
              GNautilusGdsClientApiLibraryWrapper.GdsGNautilusConfiguration();

            deviceConfiguration.ValidationIndicator = false;
            deviceConfiguration.BatteryLevel = false;
            deviceConfiguration.AccelerationData = true;
            deviceConfiguration.NumberOfScans = 0;
            deviceConfiguration.SamplingRate = 250;

            FrameDefinition.Fz = (int)deviceConfiguration.SamplingRate;
            deviceConfiguration.Channels = new
              GNautilusGdsClientApiLibraryWrapper.GdsGNautilusChannelConfigura
              tion[GNautilusGdsClientApiLibraryWrapper.MaxNumberOfChannels];

            for (int i = 0; i < deviceConfiguration.Channels.Length; i++)
            {
                deviceConfiguration.Channels[i].Enabled = false;
                deviceConfiguration.Channels[i].Sensitivity =
                  FrameDefinition.AmpSensitivity;
                deviceConfiguration.Channels[i].BipolarChannel = -1;
            }

            //build and return complete GDS configuration structure
            GdsClientApiLibraryWrapper.GdsConfiguration NAutilusConfig = new
              GdsClientApiLibraryWrapper.GdsConfiguration();

            NAutilusConfig.DeviceConfiguration = deviceConfiguration;
            NAutilusConfig.DeviceInfo = new
              GdsClientApiLibraryWrapper.DeviceInfo();
            NAutilusConfig.DeviceInfo.DeviceType =
              GdsClientApiLibraryWrapper.GdsDeviceType.GNautilus;
            NAutilusConfig.DeviceInfo.Name = serialNumber;

            return NAutilusConfig;
        }
    }
}
```