

Software Prototyping & Visualisierung Projekt



ESD
FH-HAGENBERG

Sommersemester 2015

Name: _____

Abgabetermin: 28.05.2015 23:55

Mat.Nr.: _____

Punkte: _____

Aufwand in h: _____

korrigiert: _____

Dieser Übungszettel ist in Zweiergruppen zu lösen. Es reicht, wenn ein Gruppenmitglied die Arbeit im Moodle abgibt.

Teil 1 (32 Punkte) Projekt - Applikation

Entwickeln Sie eine WPF Anwendung, die Sensordaten empfängt und verarbeitet. Sie haben freie Hand welche Sensoren Sie anbinden wollen. Sie können eine beliebige Schnittstelle verwenden, um auf die Daten zuzugreifen.

Entscheiden Sie sich für eine von 2 Möglichkeiten, was Sie mit den Sensordaten machen:

a) Visualisierung der Sensordaten

Stellen Sie die Sensordaten graphisch dar. Achten Sie je nach Sensor auf eine entsprechend sinnvolle Darstellungsform. Die Sensordaten sollen so zeitnah wie möglich dargestellt werden.

b) Interpretation der Sensordaten

Die Sensordaten sollen in der Form interpretiert werden, dass eine Aktion ausgelöst wird. Liefern sie zusätzlich auch ein Testprogramm, in dem man erkennt, was Sie steuern können/könnten.

Sie dürfen für diese Übung auch externe Libraries (z.B. aus Nuget) und fremden Code verwenden. Ziel ist es, einen funktionierenden Prototyp zu entwickeln.

Teil 2 (16 Punkte) Projekt - Präsentation

Zusätzlich zur Abgabe im Moodle müssen Sie ihr Projekt auch in der Übung, am 21.05.2015 präsentieren. Jede Gruppe hat dafür 10min Zeit. Nutzen Sie diese 10 min nicht nur um die Funktionalität ihrer Applikation zu präsentieren, sondern auch, um die interessantesten Code-Passagen zu erklären. Zum Zeitpunkt der Präsentation muss der Prototyp noch nicht fertig sein, jedoch schon so weit entwickelt, dass Sie (Teil-)Funktionalitäten präsentieren können.

Sie können für ihre Präsentation maximal 16 Punkte erhalten. Es wird die Gruppe bewertet und keine Einzelpersonen.

Allgemeine Hinweise: Legen Sie bei der Erstellung Ihrer Übung großen Wert auf eine **saubere Strukturierung** und auf eine **sorgfältige Ausarbeitung**! **Dokumentieren** Sie alle Schnittstellen und versehen Sie Ihre Algorithmen an entscheidenden Stellen ausführlich mit **Kommentaren**! **Testen** Sie ihre Implementierungen ausführlich! Geben Sie **Lösungsideen** an!

1 Dokumentation

1.1 Server Client Kommunikation

Die Kommunikation basiert auf TCP/IP. Kommuniziert wird zwischen einem Server (WPF App) und einem Client (Android App oder C# Testclient). Die Daten werden in XML-Format geschickt. Das Format wird durch eine Klasse, die die Sensorwerte beinhaltet, bestimmt. Diese Klasse wird verwendet um die Daten in XML zu serialisieren bzw. zu deserialisieren.

1.2 Android App

Die Android App hat folgende Bedienelemente:

- Feld für Server IP Adresse
- Feld für Server Port
- Connect Button
- Disconnect Button
- Feld für Anzeige der Sensorwerte

Mithilfe eines Sensor Managers werden die Sensordaten des Androidgerätes in periodischen Abständen ausgelesen. Welcher Sensor ausgelesen werden soll, kann eingestellt werden. Es können auch mehrere Sensoren ausgelesen werden.

1.3 WPF App

Die WPF App hat folgende Elemente:

- Ip Adresse des Servers
- Port des Servers
- Start Button
- Stop Button
- Anzeige eines 3D Objektes

In einem Backgroundworker läuft ein Server welcher auf Daten von einem Client wartet. Sobald Daten vorhanden sind werden die entsprechenden Properties, welche and die GUI gebunden sind (zum Beispiel Winkelwerte), gesetzt. Die 3D Darstellung des Androidgerätes wurden mit einem ModelVisual3D in einem Viewport realisiert.

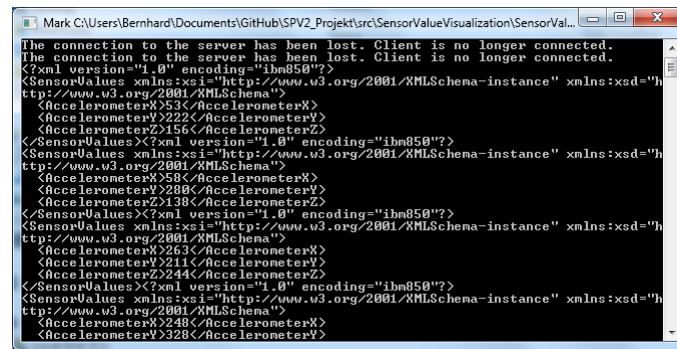
1.4 C# Testclient

Zusätzlich zur Android App wurde für Testzwecke ein C# Testclient entwickelt. Dieser schickt in einem einstellbaren Intervall zufällige Testwerte an den Server.

1.5 Erweiterbarkeit

In der Android App können beliebige Sensorwerte ausgelesen werden. Die Klasse für die Sensordaten muss entsprechend verändert werden. In der WPF Applikation können weitere Tabs hinzugefügt werden in denen dann verschiedene Sensordaten (z.B. Temperatur als Thermometer oder Lichteinfall als Lampe, ...) visualisiert werden können.

2 Test und Screenshots



```
Mark C:\Users\Bernhard\Documents\GitHub\SPV2_Projekt\src\SensorValueVisualization\SensorVal...
The connection to the server has been lost. Client is no longer connected.
The connection to the server has been lost. Client is no longer connected.
<?xml version="1.0" encoding="ibm850"?>
<SensorValues xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="h
ttp://www.w3.org/2001/XMLSchema">
  <AccelerometerX>53</AccelerometerX>
  <AccelerometerY>222</AccelerometerY>
  <AccelerometerZ>156</AccelerometerZ>
</SensorValues><?xml version="1.0" encoding="ibm850"?>
<SensorValues xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="h
ttp://www.w3.org/2001/XMLSchema">
  <AccelerometerX>58</AccelerometerX>
  <AccelerometerY>230</AccelerometerY>
  <AccelerometerZ>138</AccelerometerZ>
</SensorValues><?xml version="1.0" encoding="ibm850"?>
<SensorValues xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="h
ttp://www.w3.org/2001/XMLSchema">
  <AccelerometerX>263</AccelerometerX>
  <AccelerometerY>211</AccelerometerY>
  <AccelerometerZ>244</AccelerometerZ>
</SensorValues><?xml version="1.0" encoding="ibm850"?>
<SensorValues xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="h
ttp://www.w3.org/2001/XMLSchema">
  <AccelerometerX>248</AccelerometerX>
  <AccelerometerY>328</AccelerometerY>
```

Figure 1: C# Testclient

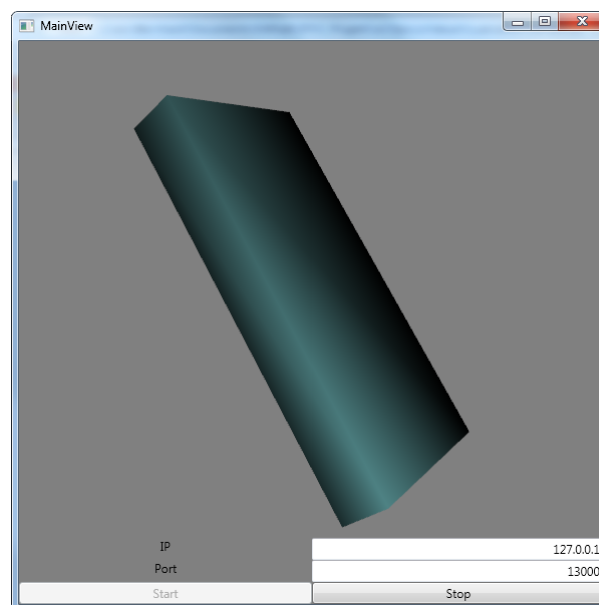


Figure 2: WPF App

3 Source Code

3.1 WPF App

C# Klasse für Sensordaten:

.././src/SensorValueVisualization/SensorValues/SensorValues.cs

```
1  i»¿using System;
2
3  namespace SensorValuesServer
4  {
5      [Serializable]
6      public class SensorValues
7      {
8          public int AccelerometerX { get; set; }
9
10         public int AccelerometerY { get; set; }
11
12         public int AccelerometerZ { get; set; }
13
14         public override string ToString()
15         {
16             return String.Format("AccelerometerX: {0}, AccelerometerY: {1},
17                                   AccelerometerZ: {2}", AccelerometerX,
18                                   AccelerometerY, AccelerometerZ);
19         }
20     }
```

WPF App:

.././src/SensorValueVisualization/SensorValueVisualization/View/MainView.xaml

```
1  i»¿<Window x:Class="SensorValueVisualization.View.MainView"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          Title="MainView" Height="600" Width="600"
5          DataContext="{Binding Main, Source={StaticResource Locator}}">
6      <Grid Background="Gray">
7          <Grid.RowDefinitions>
8              <RowDefinition Height="*" />
9              <RowDefinition Height="Auto" />
10          </Grid.RowDefinitions>
11          <Viewport3D x:Name="Viewport" Grid.Row="0">
12              <Viewport3D.Camera>
13                  <!--<PerspectiveCamera x:Name="MainCamera" Position="6 5 4"
14                      LookDirection="-6 -5 -4" />-->
15                  <PerspectiveCamera x:Name="ZeroCamera" Position="3 2.125
16                      2.5" LookDirection="-1 -1 -1" />
17              </Viewport3D.Camera>
18
19              <ModelVisual3D x:Name="TopModelVisual3D">
20                  <ModelVisual3D.Children>
21                      <ModelVisual3D>
22                          <ModelVisual3D.Content>
23                              <DirectionalLight x:Name="DirLightMain"
24                                  Direction="-1,-1,-1">
25                                  </DirectionalLight>
26                              </ModelVisual3D.Content>
27                          </ModelVisual3D>
28                      </ModelVisual3D.Children>
29                  </ModelVisual3D>
30              </ModelVisual3D>
31          </Viewport3D>
32      </Grid>
33  </Window>
```

```

26         <ModelVisual3D>
27             <ModelVisual3D.Content>
28                 <GeometryModel3D>
29                     <GeometryModel3D.Geometry>
30                         <MeshGeometry3D x:Name="MeshMain"
31                             Positions="0 0 0 2 0 0 0 0.25 0
                                     2 0.25 0 0 0 1 2 0 1 0 0.25 1
                                     2 0.25 1"
32                             TriangleIndices="2 3 1 2 1 0 7 1
                                              3 7 5 1 6 5 7 6 4 5 6 2 0 6
                                              0 4 2 7 3 2 6 7 0 1 5 0 5 4
                                              ">
33                         </MeshGeometry3D>
34                     </GeometryModel3D.Geometry>
35                     <GeometryModel3D.Material>
36                         <DiffuseMaterial>
37                             <DiffuseMaterial.Brush>
38                                 <SolidColorBrush Color="
                                    CadetBlue"/>
39                             </DiffuseMaterial.Brush>
40                         </DiffuseMaterial>
41                     </GeometryModel3D.Material>
42                 </GeometryModel3D>
43             </ModelVisual3D.Content>
44             <ModelVisual3D.Transform>
45                 <Transform3DGroup>
46                     <Transform3DGroup.Children>
47                         <RotateTransform3D CenterX="1" CenterY=
48                             "0.125" CenterZ="0.5">
49                             <RotateTransform3D.Rotation>
50                                 <AxisAngleRotation3D Axis="1 0
51                                     0" Angle="{Binding
52                                         AccelerometerX}" />
53                                 </RotateTransform3D.Rotation>
54                             </RotateTransform3D>
55                         <RotateTransform3D CenterX="1" CenterY=
56                             "0.125" CenterZ="0.5">
57                             <RotateTransform3D.Rotation>
58                                 <AxisAngleRotation3D Axis="0 1
59                                     0" Angle="{Binding
60                                         AccelerometerY}" />
61                                 </RotateTransform3D.Rotation>
62                             </RotateTransform3D>
63                         <RotateTransform3D CenterX="1" CenterY=
64                             "0.125" CenterZ="0.5">
65                             <RotateTransform3D.Rotation>
66                                 <AxisAngleRotation3D Axis="0 0
67                                     1" Angle="{Binding
68                                         AccelerometerZ}" />
69                                 </RotateTransform3D.Rotation>
70                             </RotateTransform3D>
71                         </Transform3DGroup.Children>
72                     </Transform3DGroup>
73                 </ModelVisual3D.Transform>
74             </ModelVisual3D>
75         </ModelVisual3D.Children>
76     </ModelVisual3D>
77 </Viewport3D>
78
79 <Grid Grid.Row="1">

```

```

71         <Grid.ColumnDefinitions>
72             <ColumnDefinition></ColumnDefinition>
73             <ColumnDefinition></ColumnDefinition>
74         </Grid.ColumnDefinitions>
75         <Grid.RowDefinitions>
76             <RowDefinition></RowDefinition>
77             <RowDefinition></RowDefinition>
78             <RowDefinition></RowDefinition>
79         </Grid.RowDefinitions>
80
81         <TextBlock Grid.Column="0" Grid.Row="0" HorizontalAlignment="
            Center">IP</TextBlock>
82         <TextBlock Grid.Column="0" Grid.Row="1" HorizontalAlignment="
            Center">Port</TextBlock>
83         <TextBox Grid.Column="1" Grid.Row="0"
            HorizontalContentAlignment="Right" Text="{Binding IpAdress}"
            ></TextBox>
84         <TextBox Grid.Column="1" Grid.Row="1"
            HorizontalContentAlignment="Right" Text="{Binding Port}"></
            TextBox>
85         <Button Command="{Binding ClickStartCommand}" Grid.Column="0"
            Grid.Row="2" IsEnabled="{Binding IsDisconnected}">Start</
            Button>
86         <Button Command="{Binding ClickStopCommand}" Grid.Column="1"
            Grid.Row="2" IsEnabled="{Binding IsConnected}">Stop</Button>
87     </Grid>
88
89     <!--<StackPanel Grid.Row="0">
90         <Slider x:Name="XAxisSlider" Height="20" HorizontalAlignment="
            Center" Margin="5" Width="300" Maximum="360" />
91         <Slider x:Name="YAxisSlider" Height="20" HorizontalAlignment="
            Center" Margin="5" Width="300" Maximum="360" />
92         <Slider x:Name="ZAxisSlider" Height="20" HorizontalAlignment="
            Center" Margin="5" Width="300" Maximum="360" />
93     </StackPanel>-->
94
95     </Grid>
96 </Window>

```

Viewmodel:

../src/SensorValueVisualization/SensorValueVisualization/ViewModel/MainViewModel.cs

```

1  using System;
2  using System.ComponentModel;
3  using System.Diagnostics;
4  using System.Windows;
5  using GalaSoft.MvvmLight;
6  using GalaSoft.MvvmLight.Command;
7  using SensorValuesServer;
8
9  namespace SensorValueVisualization.ViewModel
10 {
11     /// <summary>
12     /// This class contains properties that the main View can data bind to.
13     /// <para>
14     /// Use the <strong>mvvmminpc</strong> snippet to add bindable
        properties to this ViewModel.
15     /// </para>
16     /// <para>
17     /// You can also use Blend to data bind with the tool's support.
18     /// </para>

```

```

19     /// <para>
20     /// See http://www.galasoft.ch/mvvm
21     /// </para>
22     /// </summary>
23     public class MainViewModel : ViewModelBase
24     {
25         private ChatServer _chatServer;
26         private BackgroundWorker _chatServerWorker;
27         public MainViewModel()
28         {
29             IpAdress = "127.0.0.1";
30             Port = 1234;
31             IsConnected = false;
32         }
33
34         ~MainViewModel()
35         {
36             _chatServer.Stop();
37             IsConnected = false;
38         }
39
40         private string _ipAdress;
41
42         public string IpAdress
43         {
44             get { return _ipAdress; }
45             set
46             {
47                 _ipAdress = value;
48                 RaisePropertyChanged(() => IpAdress);
49             }
50         }
51
52         private int _port;
53
54         public int Port
55         {
56             get { return _port; }
57             set
58             {
59                 _port = value;
60                 RaisePropertyChanged(() => Port);
61             }
62         }
63
64         private void RunChatServer(object sender, DoWorkEventArgs e)
65         {
66             _chatServer.Start();
67         }
68
69         private void ReadSensorValues(object sender,
70                                     ProgressChangedEventArgs e)
71         {
72             SensorValues sensorValues = e.UserState as SensorValues;
73
74             if (sensorValues != null)
75             {
76                 AccelerometerX = sensorValues.AccelerometerX;
77                 AccelerometerY = sensorValues.AccelerometerY;
78                 AccelerometerZ = sensorValues.AccelerometerZ;
79             }
80         }
81     }

```

```

78         }
79     }
80
81     private int _accelerometerX;
82     public int AccelerometerX
83     {
84         get { return _accelerometerX; }
85         set
86         {
87             _accelerometerX = value;
88             RaisePropertyChanged(() => AccelerometerX);
89         }
90     }
91
92     private int _accelerometerY;
93     public int AccelerometerY
94     {
95         get { return _accelerometerY; }
96         set
97         {
98             _accelerometerY = value;
99             RaisePropertyChanged(() => AccelerometerY);
100        }
101    }
102
103     private int _accelerometerZ;
104     public int AccelerometerZ
105     {
106         get { return _accelerometerZ; }
107         set
108         {
109             _accelerometerZ = value;
110             RaisePropertyChanged(() => AccelerometerZ);
111         }
112     }
113
114     private RelayCommand _clickStartCommand;
115
116     public RelayCommand ClickStartCommand
117     {
118         get { return _clickStartCommand ?? (_clickStartCommand = new
119             RelayCommand(OnClickStart)); }
120     }
121
122     private void OnClickStart()
123     {
124         _chatServerWorker = new BackgroundWorker
125         {
126             WorkerReportsProgress = true,
127             WorkerSupportsCancellation = true
128         };
129         _chatServer = new ChatServer(IpAdress, Port, _chatServerWorker)
130         ;
131         try
132         {
133             _chatServerWorker.DoWork += RunChatServer;
134             _chatServerWorker.ProgressChanged += ReadSensorValues;
135         }
136         catch (Exception e)
137         {
138         }
139     }

```



```

136         Debug.WriteLine("Unexpected Exception occured.");
137         Debug.WriteLine(e.ToString());
138     }
139
140     if (!_chatServerWorker.IsBusy)
141     {
142         _chatServerWorker.RunWorkerAsync();
143         IsConnected = true;
144     }
145 }
146
147 private RelayCommand _clickStopCommand;
148
149 public RelayCommand ClickStopCommand
150 {
151     get { return _clickStopCommand ?? (_clickStopCommand = new
152         RelayCommand(OnClickStop)); }
153 }
154
155 private void OnClickStop()
156 {
157     _chatServer.Stop();
158     IsConnected = false;
159 }
160
161 private bool _isConnected;
162
163 public bool IsConnected
164 {
165     get { return _isConnected; }
166     set
167     {
168         IsDisconnected = !value;
169         _isConnected = value;
170         RaisePropertyChanged(() => IsConnected);
171     }
172 }
173
174 private bool _isDisconnected;
175
176 public bool IsDisconnected
177 {
178     get { return _isDisconnected; }
179     set
180     {
181         _isDisconnected = value;
182         RaisePropertyChanged(() => IsDisconnected);
183     }
184 }
185 }

```

Server:

../src/SensorValueVisualization/SensorValueVisualization/ViewModel/ChatServer.cs

```

1  i»using System;
2  using System.ComponentModel;
3  using System.Diagnostics;
4  using System.IO;
5  using System.Net;
6  using System.Net.Sockets;

```

```

7  using System.Runtime.Serialization;
8  using System.Text;
9  using System.Threading;
10 using System.Xml.Serialization;
11 using SensorValuesServer;
12
13 namespace SensorValueVisualization.ViewModel
14 {
15     public class ChatServer
16     {
17         public string IpAdress { get; private set; }
18         public int Port { get; private set; }
19
20         private const int ReadIntervallInMilliseconds = 500;
21
22         private TcpClient _connectedClient;
23
24         private readonly TcpListener _listener;
25         //private readonly IFormatter _formatter;
26         private readonly XmlSerializer _formatter;
27         private bool _isRunning;
28         private BackgroundWorker _backgroundWorker;
29
30         public ChatServer(string ipAddress, int port, BackgroundWorker
            backgroundWorker)
31         {
32             //_formatter = new BinaryFormatter();
33             _formatter = new XmlSerializer(typeof(SensorValues));
34
35             IpAdress = ipAddress;
36             Port = port;
37
38             IPAddress address = IPAddress.Parse(ipAddress);
39             _listener = new TcpListener(address, port);
40
41             _backgroundWorker = backgroundWorker;
42         }
43
44         public void Start()
45         {
46             _isRunning = true;
47
48             _listener.Start();
49             Console.WriteLine("{0} Server started, now listening for
                clients.", DateTime.Now.ToString("G"));
50
51             while (_isRunning)
52             {
53                 if (!_listener.Pending())
54                 {
55                     Thread.Sleep(500);
56                     continue;
57                 }
58
59                 _connectedClient = _listener.AcceptTcpClient();
60
61                 try
62                 {
63                     ThreadPool.QueueUserWorkItem(ReadClientMessages, null);
64                     Debug.WriteLine("Client has connected properly.");

```

```

65         }
66         catch (InvalidCastException e)
67         {
68             Debug.WriteLine("Client has not connected properly.");
69             Debug.WriteLine(e.ToString());
70             _connectedClient.Close();
71         }
72     }
73 }
74
75 public void Stop()
76 {
77     _isRunning = false;
78
79     if (_connectedClient != null)
80     {
81         _connectedClient.Close();
82     }
83
84     _listener.Stop();
85 }
86
87 private void ReadClientMessages(Object obj)
88 {
89     while (_isRunning && _connectedClient.Connected)
90     {
91         if (_connectedClient.Connected)
92         {
93             NetworkStream stream = _connectedClient.GetStream();
94             {
95                 try
96                 {
97                     //SensorValues sensorValues = (SensorValues)
98                     //    _formatter.Deserialize(stream);
99
100                     StringBuilder receivedXml = new StringBuilder(
101                         String.Empty);
102                     using (StreamReader reader = new StreamReader(
103                         stream))
104                     {
105                         string receivedLine;
106
107                         while ((receivedLine = reader.ReadLine())
108                             != null)
109                         {
110                             receivedXml.AppendLine(receivedLine);
111                         }
112
113                         using (Stream xmlStream =
114                             GenerateStreamFromString(receivedXml.
115                                 ToString()))
116                         {
117                             SensorValues sensorValues = (
118                                 SensorValues)_formatter.Deserialize(
119                                     xmlStream);
120                             Debug.WriteLine(sensorValues);
121                             _backgroundWorker.ReportProgress(0,
122                                 sensorValues);
123                         }
124                     }
125                 }
126             }
127         }
128     }
129 }

```

```

116         }
117         catch (IOException)
118         {
119             //Client closed connection
120         }
121         catch (SerializationException)
122         {
123             //currently no new message
124         }
125         catch (InvalidCastException e)
126         {
127             Debug.WriteLine("Could not cast received
128                             message.");
129             Debug.WriteLine(e.ToString());
130         }
131         finally
132         {
133             Thread.Sleep(ReadIntervallInMilliseconds);
134         }
135     }
136 }
137 }
138
139 private Stream GenerateStreamFromString(string s)
140 {
141     MemoryStream stream = new MemoryStream();
142     StreamWriter writer = new StreamWriter(stream);
143     writer.Write(s);
144     writer.Flush();
145     stream.Position = 0;
146     return stream;
147 }
148 }
149 }

```

C# Testclient:

../src/SensorValueVisualization/SensorValuesTestClient/Program.cs

```

1  i»¿using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Net;
5  using System.Net.Sockets;
6  using System.Text;
7  using System.Threading;
8  using System.Threading.Tasks;
9  using System.Xml.Serialization;
10 using SensorValuesServer;
11
12 namespace SensorValuesTestClient
13 {
14     class Program
15     {
16         private static TcpClient _tcpClient;
17         private static XmlSerializer _formatter;
18         private static NetworkStream _networkStream;
19         private static Random _randomGenerator;
20
21         private const int RandomValueMax = 360;
22

```

```

23     static void Main(string[] args)
24     {
25         _tcpClient = new TcpClient();
26         _formatter = new XmlSerializer(typeof(SensorValues));
27         _randomGenerator = new Random();
28
29         while (true)
30         {
31             SendSensorValues(new SensorValues { AccelerometerX =
                _randomGenerator.Next(0, RandomValueMax), AccelerometerY
                = _randomGenerator.Next(0, RandomValueMax),
                AccelerometerZ = _randomGenerator.Next(0, RandomValueMax
                ) });
32             Thread.Sleep(TimeSpan.FromSeconds(1));
33         }
34     }
35
36     private static void SendSensorValues(SensorValues message)
37     {
38         IAsyncResult asyncResult = _tcpClient.BeginConnect(IPAddress.
            Parse("127.0.0.1"), 1234, null, null);
39         if (!asyncResult.AsyncWaitHandle.WaitOne(TimeSpan.FromSeconds
            (5), false))
40         {
41             _tcpClient.Close();
42             throw new TimeoutException();
43         }
44
45         if (_tcpClient.Connected)
46         {
47             _tcpClient.EndConnect(asyncResult);
48             _networkStream = _tcpClient.GetStream();
49             _formatter.Serialize(Console.Out, message);
50             _formatter.Serialize(_networkStream, message);
51             _tcpClient.Client.Shutdown(SocketShutdown.Both);
52             _networkStream.Close();
53         }
54         else
55         {
56             Console.WriteLine("The connection to the server has been
                lost. Client is no longer connected.");
57         }
58
59         _tcpClient.Close();
60         _tcpClient = new TcpClient();
61     }
62 }
63 }

```

3.2 Android App

Sensordaten Klasse:

../src/SensorValuesApp/app/src/main/java/reinhard/sensorvaluesapp/SensorValues.java

```

1 package reinhard.sensorvaluesapp;
2
3 import org.simpleframework.xml.Element;
4 import org.simpleframework.xml.Root;
5

```

```

6  import java.io.Serializable;
7
8  /**
9   * Created by Bernhard on 25.05.2015.
10  */
11
12  @Root
13  public class SensorValues {
14
15      @Element
16      private float AccelerometerX;
17
18      @Element
19      private float AccelerometerY;
20
21      @Element
22      private float AccelerometerZ;
23
24      public float getAccelerometerX() {
25          return AccelerometerX;
26      }
27
28      public void setAccelerometerX(float accelerometerX) {
29          AccelerometerX = accelerometerX;
30      }
31
32      public float getAccelerometerY() {
33          return AccelerometerY;
34      }
35
36      public void setAccelerometerY(float accelerometerY) {
37          AccelerometerY = accelerometerY;
38      }
39
40      public float getAccelerometerZ() {
41          return AccelerometerZ;
42      }
43
44      public void setAccelerometerZ(float accelerometerZ) {
45          AccelerometerZ = accelerometerZ;
46      }
47  }

```

TCP Client:

../src/SensorValuesApp/app/src/main/java/reinhard/sensorvaluesapp/TcpClient.java

```

1  package reinhard.sensorvaluesapp;
2
3  import android.util.Log;
4
5  import org.simpleframework.xml.Attribute;
6  import org.simpleframework.xml.Element;
7  import org.simpleframework.xml.Root;
8  import org.simpleframework.xml.Serializer;
9  import org.simpleframework.xml.core.Persister;
10
11  import java.io.BufferedReader;
12  import java.io.BufferedWriter;
13  import java.io.File;
14  import java.io.InputStreamReader;
15  import java.io.OutputStreamWriter;

```

```

16 import java.io.PrintWriter;
17 import java.net.InetAddress;
18 import java.net.Socket;
19
20 public class TcpClient {
21
22     private String mServerIp;
23     private int mServerPort;
24
25     // message to send to the server
26     private String mServerMessage;
27     // sends message received notifications
28     //private OnMessageReceived mMessageListener = null;
29     // while this is true, the server will continue running
30     private boolean mRun = false;
31     // used to send messages
32     private PrintWriter mBufferOut;
33     // used to read messages from the server
34     private BufferedReader mBufferIn;
35
36     /**
37      * Constructor of the class. OnMessageReceived listens for the
38      * messages received from server
39      */
40     public TcpClient(String serverIp, int serverPort) {
41         //mMessageListener = listener;
42         mServerIp = serverIp;
43         mServerPort = serverPort;
44     }
45
46     // Sends the message entered by client to the server
47     public void sendMessage(SensorValues sensorValues) {
48         if (mBufferOut != null && !mBufferOut.checkError() && sensorValues
49             != null) {
50             Serializer serializer = new Persister();
51             try {
52                 serializer.write(sensorValues, mBufferOut);
53             }
54             catch (Exception e) {
55                 Log.e("Serialization", "S: Error", e);
56             }
57         }
58     }
59
60     /**
61      * Close the connection and release the members
62      */
63     public void stop() {
64         Log.i("Debug", "stop");
65
66         // send message that we are closing the connection
67         //sendMessage(Constants.CLOSED_CONNECTION + "Kazy");
68
69         mRun = false;
70
71         if (mBufferOut != null) {
72             mBufferOut.flush();
73             mBufferOut.close();
74         }
75     }
76 }

```

```

74         //mMessageListener = null;
75         mBufferIn = null;
76         mBufferOut = null;
77         mServerMessage = null;
78     }
79
80     public void run() {
81
82         mRun = true;
83
84         try {
85             //here you must put your computer's IP address.
86             InetAddress serverAddr = InetAddress.getByName(mServerIp);
87
88             Log.i("TCP Client", "C: Connecting...");
89
90             Log.i("Server Ip", serverAddr.getHostAddress());
91             Log.i("Server Port", Integer.toString(mServerPort));
92
93             //create a socket to make the connection with the server
94             Socket socket = new Socket(serverAddr, mServerPort);
95
96             try {
97                 Log.i("Debug", "inside try catch");
98                 //sends the message to the server
99                 mBufferOut = new PrintWriter(new BufferedWriter(new
100                     OutputStreamWriter(socket.getOutputStream()), true);
101
102                 //receives the message which the server sends back
103                 mBufferIn = new BufferedReader(new InputStreamReader(socket
104                     .getInputStream()));
105                 // send login name
106                 //sendMessage(Constants.LOGIN_NAME + PreferencesManager.
107                     getInstance().getUserName());
108                 //sendMessage("Hi");
109                 //in this while the client listens for the messages sent by
110                 //the server
111                 while (mRun) {
112                     mServerMessage = mBufferIn.readLine();
113                     //if (mServerMessage != null && mMessageListener !=
114                         null) {
115                         // //call the method messageReceived from MyActivity
116                         // class
117                         // mMessageListener.messageReceived(mServerMessage);
118                     //}
119                 }
120                 Log.e("RESPONSE FROM SERVER", "S: Received Message: '" +
121                     mServerMessage + "'");
122             } catch (Exception e) {
123
124                 Log.e("TCP", "S: Error", e);
125
126             } finally {
127                 //the socket must be closed. It is not possible to
128                 //reconnect to this socket
129                 // after it is closed, which means a new socket instance
130                 //has to be created.
131                 socket.close();
132             }
133         }
134     }

```



```

125         }
126
127     } catch (Exception e) {
128
129         Log.e("TCP", "C: Error", e);
130
131     }
132
133 }
134
135 //Declare the interface. The method messageReceived(String message)
136 //will must be implemented in the MyActivity
137 //class at on asynckTask doInBackground
138 public interface OnMessageReceived {
139     public void messageReceived(String message);
140 }

```

Android App:

../src/SensorValuesApp/app/src/main/java/reinhard/sensorvaluesapp/SensorValuesActivity.java

```

1 package reinhard.sensorvaluesapp;
2
3 import android.app.Activity;
4 import android.content.Context;
5 import android.hardware.Sensor;
6 import android.hardware.SensorEvent;
7 import android.hardware.SensorEventListener;
8 import android.hardware.SensorManager;
9 import android.os.AsyncTask;
10 import android.os.Bundle;
11 import android.util.Log;
12 import android.view.Menu;
13 import android.view.MenuItem;
14 import android.view.View;
15 import android.widget.Button;
16 import android.widget.EditText;
17 import android.widget.TextView;
18
19
20 public class SensorValuesActivity extends Activity implements
    SensorEventListener {
21
22     private SensorManager senSensorManager;
23     private Sensor senAccelerometer;
24
25     private long lastUpdate = 0;
26     private float last_x, last_y, last_z;
27     private static final int SHAKE_THRESHOLD = 600;
28
29     private Button connectButton;
30     private Button disconnectButton;
31
32     TcpClient tcpClient;
33
34     @Override
35     protected void onCreate(Bundle savedInstanceState) {
36         super.onCreate(savedInstanceState);
37         setContentView(R.layout.activity_sensor_values);
38

```

```

39         senSensorManager = (SensorManager) getSystemService(Context.
            SENSOR_SERVICE);
40         senAccelerometer = senSensorManager.getDefaultSensor(Sensor.
            TYPE_ACCELEROMETER);
41         senSensorManager.registerListener(this, senAccelerometer ,
            SensorManager.SENSOR_DELAY_NORMAL);
42
43         connectButton = (Button) findViewById(R.id.btnConnect);
44         //connectButton.setEnabled(true);
45         disconnectButton = (Button) findViewById(R.id.btnDisconnect);
46         //disconnectButton.setEnabled(false);
47     }
48
49     protected void onPause() {
50         super.onPause();
51         senSensorManager.unregisterListener(this);
52     }
53
54     protected void onResume() {
55         super.onResume();
56         senSensorManager.registerListener(this, senAccelerometer,
            SensorManager.SENSOR_DELAY_NORMAL);
57     }
58
59     @Override
60     public boolean onCreateOptionsMenu(Menu menu) {
61         // Inflate the menu; this adds items to the action bar if it is
            present.
62         getMenuInflater().inflate(R.menu.sensor_values, menu);
63         return true;
64     }
65
66     @Override
67     public boolean onOptionsItemSelected(MenuItem item) {
68         // Handle action bar item clicks here. The action bar will
69         // automatically handle clicks on the Home/Up button, so long
70         // as you specify a parent activity in AndroidManifest.xml.
71         int id = item.getItemId();
72         if (id == R.id.action_settings) {
73             return true;
74         }
75         return super.onOptionsItemSelected(item);
76     }
77
78     @Override
79     public void onSensorChanged(SensorEvent event) {
80         Sensor mySensor = event.sensor;
81
82         if (mySensor.getType() == Sensor.TYPE_ACCELEROMETER) {
83             SensorValues sensorValues = new SensorValues();
84             sensorValues.setAccelerometerX(event.values[0]);
85             sensorValues.setAccelerometerY(event.values[1]);
86             sensorValues.setAccelerometerZ(event.values[2]);
87
88             long curTime = System.currentTimeMillis();
89
90             if ((curTime - lastUpdate) > 100) {
91                 long diffTime = (curTime - lastUpdate);
92                 lastUpdate = curTime;
93

```

```

94         TextView textView = (TextView) findViewById(R.id.
95             lblAccelerometerValues);
96         textView.setText(String.format("X: %s, Y: %s, Z: %s",
97             sensorValues.getAccelerometerX(), sensorValues.
98             getAccelerometerY(), sensorValues.getAccelerometerZ()));
99
100         // check if client is connected
101         if (tcpClient != null) {
102             tcpClient.sendMessage(sensorValues);
103         }
104     }
105
106     @Override
107     public void onAccuracyChanged(Sensor sensor, int accuracy) {
108     }
109
110     public void OnClickConnect(View view) {
111         new ConnectTask().execute("");
112
113         // TODO: check if connected
114
115         disconnectButton.setEnabled(true);
116         connectButton.setEnabled(false);
117     }
118
119     public void OnClickDisconnect(View view) {
120         tcpClient.stop();
121         connectButton.setEnabled(true);
122         disconnectButton.setEnabled(false);
123     }
124
125     public class ConnectTask extends AsyncTask<String, String, TcpClient> {
126
127         @Override
128         protected TcpClient doInBackground(String... message) {
129             EditText serverIp = (EditText) findViewById(R.id.tbServerIp);
130             EditText serverPort = (EditText) findViewById(R.id.tbServerPort
131                 );
132             tcpClient = new TcpClient(serverIp.getText().toString(),
133                 Integer.parseInt(serverPort.getText().toString()));
134             tcpClient.run();
135
136             return null;
137         }
138     }

```

Android Oberfläche:

../src/SensorValuesApp/app/src/main/res/layout/activity_sensor_values.xml

```

1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:paddingLeft="@dimen/activity_horizontal_margin"
6     android:paddingRight="@dimen/activity_horizontal_margin"
7     android:paddingTop="@dimen/activity_vertical_margin"
8     android:paddingBottom="@dimen/activity_vertical_margin"

```

```

9      tools:context=".SensorValuesActivity">
10
11      <LinearLayout
12          android:layout_width="wrap_content"
13          android:layout_height="wrap_content"
14          android:orientation="vertical">
15
16          <TextView
17              android:layout_width="wrap_content"
18              android:layout_height="wrap_content"
19              android:textAppearance="?android:attr/textAppearanceSmall"
20              android:text="Server Ip:"
21              android:id="@+id/lblServerIp" />
22
23          <EditText
24              android:layout_width="wrap_content"
25              android:layout_height="wrap_content"
26              android:inputType="phone"
27              android:ems="10"
28              android:id="@+id/tbServerIp"
29              android:text="10.0.3.2"/>
30
31          <TextView
32              android:layout_width="wrap_content"
33              android:layout_height="wrap_content"
34              android:textAppearance="?android:attr/textAppearanceSmall"
35              android:text="Server Port:"
36              android:id="@+id/lblServerPort" />
37
38          <EditText
39              android:layout_width="wrap_content"
40              android:layout_height="wrap_content"
41              android:inputType="phone"
42              android:ems="10"
43              android:id="@+id/tbServerPort"
44              android:text="1234"/>
45
46          <Space
47              android:layout_width="match_parent"
48              android:layout_height="wrap_content"
49              android:minHeight="10dp" />
50
51          <Button
52              android:layout_width="wrap_content"
53              android:layout_height="wrap_content"
54              android:text="Connect"
55              android:id="@+id/btnConnect"
56              android:onClick="OnClickConnect"/>
57
58          <Button
59              android:layout_width="wrap_content"
60              android:layout_height="wrap_content"
61              android:text="Disconnect"
62              android:id="@+id/btnDisconnect"
63              android:onClick="OnClickDisconnect"/>
64
65          <Space
66              android:layout_width="match_parent"
67              android:layout_height="wrap_content"
68              android:minHeight="25dp" />

```

```
69
70     <TextView
71         android:layout_width="wrap_content"
72         android:layout_height="wrap_content"
73         android:textAppearance="?android:attr/textAppearanceSmall"
74         android:text="Accelerometer Values:"
75         android:id="@+id/lblAccelerometer" />
76
77     <TextView
78         android:layout_width="wrap_content"
79         android:layout_height="wrap_content"
80         android:textAppearance="?android:attr/textAppearanceSmall"
81         android:text="n/a"
82         android:id="@+id/lblAccelerometerValues" />
83
84     </LinearLayout>
85
86 </RelativeLayout>
```