

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.

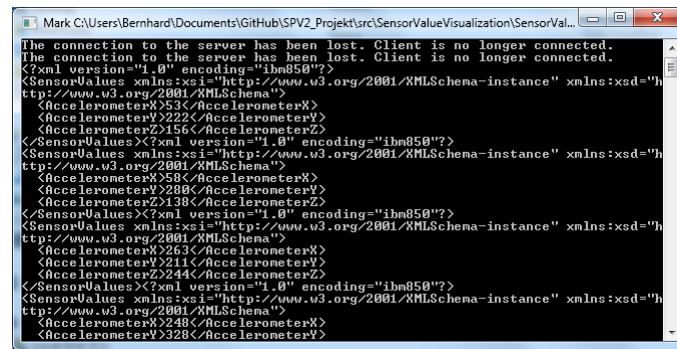
1.4 C# Testclient

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

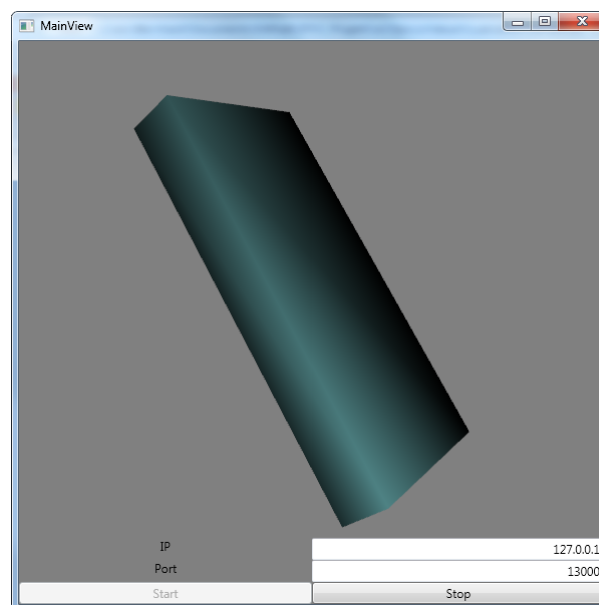


Figure 2: WPF App mit C# Testclient

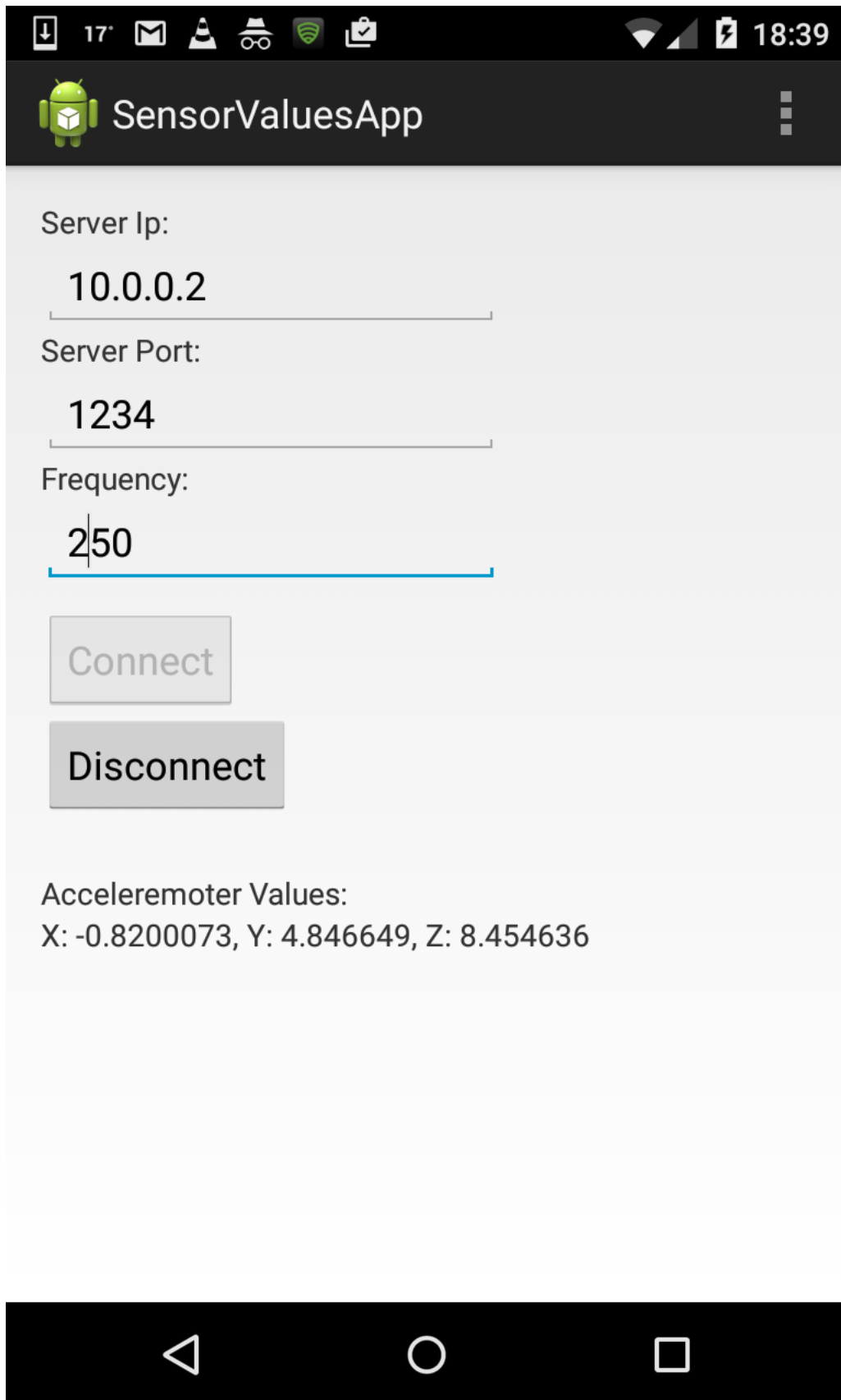


Figure 3: Android App

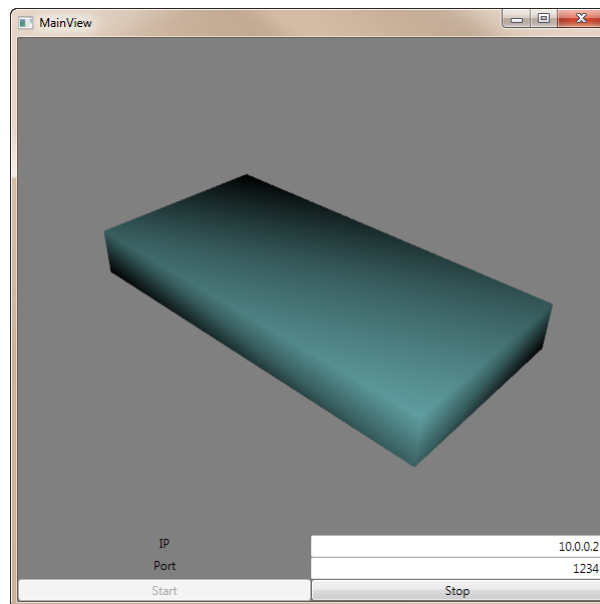


Figure 4: WPF App Test mit Android App (1)

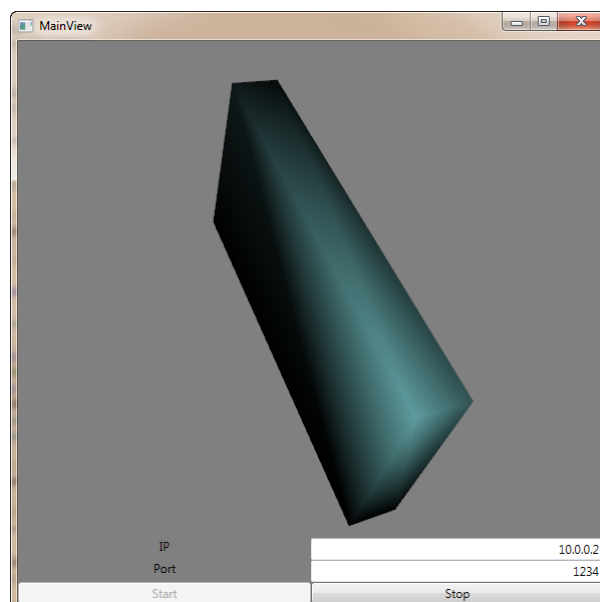


Figure 5: WPF App Test mit Android App (2)

3 Source Code

3.1 WPF App

C# Klasse für Sensordaten:

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

```
1  i»¿using System;
2
3  namespace SensorValues
4  {
5      [Serializable]
6      public class SensorValues
7      {
8          public double AccelerometerX { get; set; }
9
10         public double AccelerometerY { get; set; }
11
12         public double 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              </Viewport3D>
31          </Grid>
32      </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 GalaSoft.MvvmLight;
5  using GalaSoft.MvvmLight.Command;
6
7  namespace SensorValueVisualization.ViewModel
8  {
9      /// <summary>
10     /// This class contains properties that the main View can data bind to.
11     /// <para>
12     /// Use the <strong>mvvmminpc</strong> snippet to add bindable
        properties to this ViewModel.
13     /// </para>
14     /// <para>
15     /// You can also use Blend to data bind with the tool's support.
16     /// </para>
17     /// <para>
18     /// See http://www.galasoft.ch/mvvm

```



```

19  /// </para>
20  /// </summary>
21  public class MainViewModel : ViewModelBase
22  {
23      private const double Gravity = 9.81;
24      private const double Multiplier = 360/(2*Gravity);
25
26      private SensorValuesServer _sensorValuesServer;
27      private BackgroundWorker _chatServerWorker;
28      public MainViewModel()
29      {
30          IpAddress = "10.0.0.2";
31          Port = 1234;
32          IsConnected = false;
33      }
34
35      ~MainViewModel()
36      {
37          _sensorValuesServer.Stop();
38          IsConnected = false;
39      }
40
41      private string _ipAddress;
42
43      public string IpAddress
44      {
45          get { return _ipAddress; }
46          set
47          {
48              _ipAddress = value;
49              RaisePropertyChanged(() => IpAddress);
50          }
51      }
52
53      private int _port;
54
55      public int Port
56      {
57          get { return _port; }
58          set
59          {
60              _port = value;
61              RaisePropertyChanged(() => Port);
62          }
63      }
64
65      private void RunChatServer(object sender, DoWorkEventArgs e)
66      {
67          _sensorValuesServer.Start();
68      }
69
70      private int ConvertFromAccelerometerToAngle(double val)
71      {
72          return Convert.ToInt32(Math.Round(val*Multiplier));
73      }
74
75      private void ReadSensorValues(object sender,
76          ProgressChangedEventArgs e)

```

```

77         SensorValues.SensorValues sensorValues = e.UserState as
           SensorValues.SensorValues;
78
79         if (sensorValues != null)
80         {
81             AccelerometerX = ConvertFromAccelerometerToAngle(
                sensorValues.AccelerometerX);
82             AccelerometerY = ConvertFromAccelerometerToAngle(
                sensorValues.AccelerometerY);
83             AccelerometerZ = ConvertFromAccelerometerToAngle(
                sensorValues.AccelerometerZ);
84         }
85     }
86
87     private int _accelerometerX;
88     public int AccelerometerX
89     {
90         get { return _accelerometerX; }
91         set
92         {
93             _accelerometerX = value;
94             RaisePropertyChanged(() => AccelerometerX);
95         }
96     }
97
98     private int _accelerometerY;
99     public int AccelerometerY
100    {
101        get { return _accelerometerY; }
102        set
103        {
104            _accelerometerY = value;
105            RaisePropertyChanged(() => AccelerometerY);
106        }
107    }
108
109     private int _accelerometerZ;
110     public int AccelerometerZ
111     {
112         get { return _accelerometerZ; }
113         set
114         {
115             _accelerometerZ = value;
116             RaisePropertyChanged(() => AccelerometerZ);
117         }
118     }
119
120     private RelayCommand _clickStartCommand;
121
122     public RelayCommand ClickStartCommand
123     {
124         get { return _clickStartCommand ?? (_clickStartCommand = new
            RelayCommand(OnClickStart)); }
125     }
126
127     private void OnClickStart()
128     {
129         _chatServerWorker = new BackgroundWorker
130         {
131             WorkerReportsProgress = true,

```

```

132         WorkerSupportsCancellation = true
133     };
134     _sensorValuesServer = new SensorValuesServer(IpAddress, Port,
        _chatServerWorker);
135     try
136     {
137         _chatServerWorker.DoWork += RunChatServer;
138         _chatServerWorker.ProgressChanged += ReadSensorValues;
139     }
140     catch (Exception e)
141     {
142         Debug.WriteLine("Unexpected Exception occured.");
143         Debug.WriteLine(e.ToString());
144     }
145
146     if (!_chatServerWorker.IsBusy)
147     {
148         _chatServerWorker.RunWorkerAsync();
149         IsConnected = true;
150     }
151 }
152
153 private RelayCommand _clickStopCommand;
154
155 public RelayCommand ClickStopCommand
156 {
157     get { return _clickStopCommand ?? (_clickStopCommand = new
        RelayCommand(OnClickStop)); }
158 }
159
160 private void OnClickStop()
161 {
162     _sensorValuesServer.Stop();
163     IsConnected = false;
164 }
165
166 private bool _isConnected;
167
168 public bool IsConnected
169 {
170     get { return _isConnected; }
171     set
172     {
173         IsDisconnected = !value;
174         _isConnected = value;
175         RaisePropertyChanged(() => IsConnected);
176     }
177 }
178
179 private bool _isDisconnected;
180
181 public bool IsDisconnected
182 {
183     get { return _isDisconnected; }
184     set
185     {
186         _isDisconnected = value;
187         RaisePropertyChanged(() => IsDisconnected);
188     }
189 }

```

```
190     }
191 }
```

Server:

../src/SensorValueVisualization/SensorValueVisualization/ViewModel/SensorValuesServer.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
12 namespace SensorValueVisualization.ViewModel
13 {
14     public class SensorValuesServer
15     {
16         public string IpAdress { get; private set; }
17         public int Port { get; private set; }
18
19         private const int ReadIntervallInMilliseconds = 300;
20
21         private TcpClient _connectedClient;
22
23         private readonly TcpListener _listener;
24         //private readonly IFormatter _formatter;
25         private readonly XmlSerializer _serializer;
26         private bool _isRunning;
27         private readonly BackgroundWorker _backgroundWorker;
28         private StreamReader _reader;
29
30         public SensorValuesServer(string ipAddress, int port,
31             BackgroundWorker backgroundWorker)
32         {
33             //_formatter = new BinaryFormatter();
34             _serializer = new XmlSerializer(typeof(SensorValues.SensorValues));
35
36             IpAdress = ipAddress;
37             Port = port;
38
39             IPAddress address = IPAddress.Parse(ipAddress);
40             _listener = new TcpListener(address, port);
41
42             _backgroundWorker = backgroundWorker;
43         }
44
45         public void Start()
46         {
47             _isRunning = true;
48
49             _listener.Start();
50             Console.WriteLine("{0} Server started, now listening for
51                 clients.", DateTime.Now.ToString("G"));
52
53             while (_isRunning)
54             {
55 
```

```

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     if (_reader != null)
85     {
86         _reader.Dispose();
87     }
88
89     _listener.Stop();
90 }
91
92 private void ReadClientMessages(Object obj)
93 {
94     while (_isRunning && _connectedClient.Connected)
95     {
96         if (_connectedClient.Connected)
97         {
98             NetworkStream stream = _connectedClient.GetStream();
99             {
100                 try
101                 {
102                     //SensorValues sensorValues = (SensorValues)
103                     //    _formatter.Deserialize(stream);
104
105                     StringBuilder receivedXml = new StringBuilder(
106                         String.Empty);
107                     _reader = new StreamReader(stream);
108
109                     string receivedLine;
110
111                     while ((receivedLine = _reader.ReadLine()) !=
112                         null)

```

```

110         {
111             receivedXml.AppendLine(receivedLine);
112
113             if (receivedLine == "</SensorValues>")
114             {
115                 break;
116             }
117         }
118
119         using (Stream xmlStream =
120             GenerateStreamFromString(receivedXml.
121                 ToString()))
122         {
123             SensorValues.SensorValues sensorValues = (
124                 SensorValues.SensorValues)_formatter.
125                 Deserialize(xmlStream);
126             Debug.WriteLine(sensorValues);
127             _backgroundWorker.ReportProgress(0,
128                 sensorValues);
129         }
130     }
131     catch (IOException)
132     {
133         //Client closed connection
134     }
135     catch (SerializationException)
136     {
137         //currently no new message
138     }
139     catch (InvalidCastException e)
140     {
141         Debug.WriteLine("Could not cast received
142             message.");
143         Debug.WriteLine(e.ToString());
144     }
145     finally
146     {
147         Thread.Sleep(ReadIntervallInMilliseconds);
148     }
149 }
150
151 private Stream GenerateStreamFromString(string s)
152 {
153     MemoryStream stream = new MemoryStream();
154     StreamWriter writer = new StreamWriter(stream);
155     writer.Write(s);
156     writer.Flush();
157     stream.Position = 0;
158     return stream;
159 }

```

C# Testclient:

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

```

1  i»using System;
2  using System.Net;

```

```

3 using System.Net.Sockets;
4 using System.Threading;
5 using System.Xml.Serialization;
6
7 namespace SensorValuesTestClient
8 {
9     class Program
10    {
11        private static TcpClient _tcpClient;
12        private static XmlSerializer _formatter;
13        private static NetworkStream _networkStream;
14        private static Random _randomGenerator;
15
16        private const int RandomValueMax = 360;
17
18        static void Main()
19        {
20            _tcpClient = new TcpClient();
21            _formatter = new XmlSerializer(typeof(SensorValues.SensorValues
22                ));
23            _randomGenerator = new Random();
24
25            while (true)
26            {
27                SendSensorValues(new SensorValues.SensorValues {
28                    AccelerometerX = _randomGenerator.Next(0, RandomValueMax
29                    ), AccelerometerY = _randomGenerator.Next(0,
30                    RandomValueMax), AccelerometerZ = _randomGenerator.Next
31                    (0, RandomValueMax) });
32                Thread.Sleep(TimeSpan.FromSeconds(1));
33            }
34
35            private static void SendSensorValues(SensorValues.SensorValues
36                message)
37            {
38                IAsyncResult asyncResult = _tcpClient.BeginConnect(IPAddress.
39                    Parse("127.0.0.1"), 1234, null, null);
40                if (!asyncResult.AsyncWaitHandle.WaitOne(TimeSpan.FromSeconds
41                    (5), false))
42                {
43                    _tcpClient.Close();
44                    throw new TimeoutException();
45                }
46
47                if (_tcpClient.Connected)
48                {
49                    _tcpClient.EndConnect(asyncResult);
50                    _networkStream = _tcpClient.GetStream();
51                    _formatter.Serialize(Console.Out, message);
52                    _formatter.Serialize(_networkStream, message);
53                    _tcpClient.Client.Shutdown(SocketShutdown.Both);
54                    _networkStream.Close();
55                }
56                else
57                {
58                    Console.WriteLine("The connection to the server has been
59                        lost. Client is no longer connected.");
60                }
61            }
62        }
63    }
64 }

```

```

54         _tcpClient.Close();
55         _tcpClient = new TcpClient();
56     }
57 }
58 }

```

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 java.io.BufferedReader;
6 import java.io.BufferedWriter;
7 import java.io.InputStreamReader;
8 import java.io.OutputStreamWriter;
9 import java.io.PrintWriter;
10 import java.net.InetAddress;
11 import java.net.Socket;
12
13 public class TcpClient {
14
15     private String mServerIp;
16     private int mServerPort;
17
18     private boolean mRun = false;
19     private PrintWriter mBufferOut;
20
21     public TcpClient(String serverIp, int serverPort) {
22         mServerIp = serverIp;
23         mServerPort = serverPort;
24     }
25
26     public void sendMessage(SensorValues sensorValues) {
27         if (mBufferOut != null && !mBufferOut.checkError() && sensorValues
28             != null) {
29             try {
30                 XmlCreator creator = new XmlCreator();
31                 mBufferOut.write(creator.CreateXmlFromSensorValues(
32                     sensorValues));
33             }
34             catch (Exception e) {
35                 Log.e("Serialization", "S: Error", e);
36             }
37         }
38
39         public void stop() {
40             Log.i("Debug", "stop");
41
42             mRun = false;
43
44             if (mBufferOut != null) {
45                 mBufferOut.flush();
46                 mBufferOut.close();
47             }
48
49             mBufferOut = null;
50         }
51
52         public void run() {
53
54             mRun = true;
55
56             try {
57                 InetAddress serverAddr = InetAddress.getByName(mServerIp);
58                 Log.i("TCP Client", "C: Connecting...");
```

```

58         Log.i("Server Ip", serverAddr.getHostAddress());
59         Log.i("Server Port", Integer.toString(mServerPort));
60
61         Socket socket = new Socket(serverAddr, mServerPort);
62
63         try {
64             Log.i("Debug", "inside try catch");
65             mBufferOut = new PrintWriter(new BufferedWriter(new
66                 OutputStreamWriter(socket.getOutputStream()), true);
67             while (mRun) {
68             }
69         } catch (Exception e) {
70             Log.e("TCP", "S: Error", e);
71         } finally {
72             socket.close();
73         }
74     } catch (Exception e) {
75
76         Log.e("TCP", "C: Error", e);
77     }
78 }
79
80 public interface OnMessageReceived {
81     public void messageReceived(String message);
82 }
83 }

```

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
21     SensorEventListener {
22
23     private SensorManager senSensorManager;
24     private Sensor senAccelerometer;
25
26     private long lastUpdate = 0;
27
28     private Button connectButton;
29     private Button disconnectButton;
30 }

```

```

30     TcpClient tcpClient;
31
32     @Override
33     protected void onCreate(Bundle savedInstanceState) {
34         super.onCreate(savedInstanceState);
35         setContentView(R.layout.activity_sensor_values);
36
37         senSensorManager = (SensorManager) getSystemService(Context.
            SENSOR_SERVICE);
38         senAccelerometer = senSensorManager.getDefaultSensor(Sensor.
            TYPE_ACCELEROMETER);
39         senSensorManager.registerListener(this, senAccelerometer ,
            SensorManager.SENSOR_DELAY_NORMAL);
40
41         connectButton = (Button) findViewById(R.id.btnConnect);
42         disconnectButton = (Button) findViewById(R.id.btnDisconnect);
43     }
44
45     protected void onPause() {
46         super.onPause();
47         senSensorManager.unregisterListener(this);
48     }
49
50     protected void onResume() {
51         super.onResume();
52         senSensorManager.registerListener(this, senAccelerometer,
            SensorManager.SENSOR_DELAY_NORMAL);
53     }
54
55     @Override
56     public boolean onCreateOptionsMenu(Menu menu) {
57         getMenuInflater().inflate(R.menu.sensor_values, menu);
58         return true;
59     }
60
61     @Override
62     public boolean onOptionsItemSelected(MenuItem item) {
63         int id = item.getItemId();
64         if (id == R.id.action_settings) {
65             return true;
66         }
67         return super.onOptionsItemSelected(item);
68     }
69
70     @Override
71     public void onSensorChanged(SensorEvent event) {
72         Sensor mySensor = event.sensor;
73
74         if (mySensor.getType() == Sensor.TYPE_ACCELEROMETER) {
75             SensorValues sensorValues = new SensorValues();
76             sensorValues.setAccelerometerX(event.values[0]);
77             sensorValues.setAccelerometerY(event.values[1]);
78             sensorValues.setAccelerometerZ(event.values[2]);
79
80             long curTime = System.currentTimeMillis();
81
82             EditText editText = (EditText) findViewById(R.id.tbFrequency);
83             int frequency = Integer.parseInt(editText.getText().toString())
84                 ;

```

```

85         if ((curTime - lastUpdate) > frequency) {
86             long diffTime = (curTime - lastUpdate);
87             lastUpdate = curTime;
88
89             TextView textView = (TextView) findViewById(R.id.
                lblAccelerometerValues);
90             textView.setText(String.format("X: %s, Y: %s, Z: %s",
                sensorValues.getAccelerometerX(), sensorValues.
                getAccelerometerY(), sensorValues.getAccelerometerZ()));
91
92             // check if client is connected
93             if (tcpClient != null) {
94                 tcpClient.sendMessage(sensorValues);
95             }
96         }
97     }
98 }
99
100 @Override
101 public void onAccuracyChanged(Sensor sensor, int accuracy) {
102 }
103
104
105 public void OnClickConnect(View view) {
106     new ConnectTask().execute("");
107
108     disconnectButton.setEnabled(true);
109     connectButton.setEnabled(false);
110 }
111
112 public void OnClickDisconnect(View view) {
113     tcpClient.stop();
114     connectButton.setEnabled(true);
115     disconnectButton.setEnabled(false);
116 }
117
118 public class ConnectTask extends AsyncTask<String, String, TcpClient> {
119
120     @Override
121     protected TcpClient doInBackground(String... message) {
122         EditText serverIp = (EditText) findViewById(R.id.tbServerIp);
123         EditText serverPort = (EditText) findViewById(R.id.tbServerPort
            );
124         tcpClient = new TcpClient(serverIp.getText().toString(),
            Integer.parseInt(serverPort.getText().toString()));
125         tcpClient.run();
126
127         return null;
128     }
129 }
130 }

```

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      <TextView
47          android:layout_width="wrap_content"
48          android:layout_height="wrap_content"
49          android:textAppearance="?android:attr/textAppearanceSmall"
50          android:text="Frequency:"
51          android:id="@+id/lblFrequency" />
52
53      <EditText
54          android:layout_width="wrap_content"
55          android:layout_height="wrap_content"
56          android:inputType="phone"
57          android:ems="10"
58          android:id="@+id/tbFrequency"
59          android:text="500"/>
60
61      <Space
62          android:layout_width="match_parent"
63          android:layout_height="wrap_content"
64          android:minHeight="10dp" />
65
66      <Button

```

```

67         android:layout_width="wrap_content"
68         android:layout_height="wrap_content"
69         android:text="Connect"
70         android:id="@+id/btnConnect"
71         android:onClick="OnClickConnect"
72         android:enabled="true" />
73
74     <Button
75         android:layout_width="wrap_content"
76         android:layout_height="wrap_content"
77         android:text="Disconnect"
78         android:id="@+id/btnDisconnect"
79         android:onClick="OnClickDisconnect"
80         android:enabled="false" />
81
82     <Space
83         android:layout_width="match_parent"
84         android:layout_height="wrap_content"
85         android:minHeight="25dp" />
86
87     <TextView
88         android:layout_width="wrap_content"
89         android:layout_height="wrap_content"
90         android:textAppearance="?android:attr/textAppearanceSmall"
91         android:text="Accelerometer Values:"
92         android:id="@+id/lblAccelerometer" />
93
94     <TextView
95         android:layout_width="wrap_content"
96         android:layout_height="wrap_content"
97         android:textAppearance="?android:attr/textAppearanceSmall"
98         android:text="n/a"
99         android:id="@+id/lblAccelerometerValues" />
100
101     </LinearLayout>
102
103 </RelativeLayout>

```

XML-Serializer:

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

```

1 package reinhard.sensorvaluesapp;
2
3 /**
4  * Created by Reinhard on 28.05.2015.
5  */
6 public class XmlCreator {
7     private String GetHeader()
8     {
9         return "<?xml version=\"1.0\" encoding=\"ibm850\"?>\r\n";
10    }
11
12    public String CreateTags(String tag, String value)
13    {
14        return String.format("<%s>%s</%s>\r\n", tag, value, tag);
15    }
16
17    public String CreateXmlFromSensorValues(SensorValues values)
18    {
19        String header = GetHeader();

```

```
20     String xmlValues = String.format("  %s  %s  %s", CreateTags("
21         AccelerometerX", Float.toString(values.getAccelerometerX())),
22         CreateTags("AccelerometerY", Float.toString(values.
23             getAccelerometerY()))),
24         CreateTags("AccelerometerZ", Float.toString(values.
25             getAccelerometerZ())));
26     return String.format("%s<SensorValues xmlns:xsi=\"http://www.w3.org
27         /2001/XMLSchema-instance\" xmlns:xsd=\"http://www.w3.org/2001/
28         XMLSchema\">\r\n%s</SensorValues>\r\n", header, xmlValues);
29 }
```