```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 7th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeSenseHAT is a program who use a
 *   Sense HAT, it's an electronic card who can be
 *   mesured value with sensor. This program use
 *   the Sense HAT to mesure the temperature, the
 *   humidity and the pressure.
\*-------------------------------------------------*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Windows.Networking;
using Windows.Networking.Sockets;
using Windows.Storage.Streams;

namespace RaspiHomeSenseHAT
{
    public class CommunicationWithServer
    {
        #region Fields
        #region Constants
        // Default information to connect on the server
        private const int PORT = 54565;
        //// Need to be changed fo each configuration
        private const string IPSERVER = "10.134.97.117";// "192.168.2.8";

        // String format for connection of the client
        private const string FORMATSTRING = "IPRasp={0};Location=
          {1};Component={2}";
        private const string COMMUNICATIONSEPARATOR = "@";

        // Important need to be changed if it's another room!
        private const string LOCATION = "Salon";
        private const string COMPONENT = "Sensor";
        private const string RPINAME = "SenseHAT_" + LOCATION;

        private const int MESSAGE_FULL_LENGHT = 512;
        #endregion

        #region Variables
        private ModelSenseHAT _mSenseHAT;

        private StreamSocket _socket = new StreamSocket();
        private StreamSocketListener _listener = new StreamSocketListener();
        private List<StreamSocket> _connections = new List<StreamSocket>();
        private bool _isConnected = false;
        private bool _connecting = false;
        #endregion
        #endregion
```

```
56
57          #region Properties
58          public ModelSenseHAT MSenseHAT
59          {
60              get
61              {
62                  return _mSenseHAT;
63              }
64
65              set
66              {
67                  _mSenseHAT = value;
68              }
69          }
70
71          public StreamSocket Socket
72          {
73              get
74              {
75                  return _socket;
76              }
77
78              set
79              {
80                  _socket = value;
81              }
82          }
83
84          public StreamSocketListener Listener
85          {
86              get
87              {
88                  return _listener;
89              }
90
91              set
92              {
93                  _listener = value;
94              }
95          }
96
97          public List<StreamSocket> Connections
98          {
99              get
100             {
101                 return _connections;
102             }
103
104             set
105             {
106                 _connections = value;
107             }
108         }
109
110         public bool IsConnected
111         {
```

```
112             get
113             {
114                 return _isConnected;
115             }
116 
117             set
118             {
119                 _isConnected = value;
120             }
121         }
122 
123         public bool Connecting
124         {
125             get
126             {
127                 return _connecting;
128             }
129 
130             set
131             {
132                 _connecting = value;
133             }
134         }
135         #endregion
136 
137         #region Constructors
138         /// <summary>
139         /// Constructor: Initializer
140         /// </summary>
141         /// <param name="paramModel"></param>
142         public CommunicationWithServer(ModelSenseHAT paramModel)
143         {
144             this.MSenseHAT = paramModel;
145 
146             Connect();
147         }
148         #endregion
149 
150         #region Methods
151         /// <summary>
152         /// Connect the raspberry to the server
153         /// </summary>
154         private async void Connect()
155         {
156             try
157             {
158                 this.Connecting = true;
159                 await this.Socket.ConnectAsync(new HostName(IPSERVER),
                      PORT.ToString());
160                 SendForInitialize();
161                 this.Connecting = false;
162                 this.IsConnected = true;
163 
164                 WaitForData(this.Socket);
165             }
166             catch (Exception)
```

```
167             {
168                 this.Connecting = false;
169                 this.IsConnected = false;
170             }
171         }
172 
173         /// <summary>
174         /// Listen the traffic on the port
175         /// </summary>
176         private async void Listen()
177         {
178             this.Listener.ConnectionReceived += listenerConnectionReceived;
179             await this.Listener.BindServiceNameAsync(PORT.ToString());
180         }
181 
182         void listenerConnectionReceived(StreamSocketListener sender,
              StreamSocketListenerConnectionReceivedEventArgs args)
183         {
184             this.Connections.Add(args.Socket);
185 
186             WaitForData(args.Socket);
187         }
188 
189         /// <summary>
190         /// Send the message in input to output
191         /// </summary>
192         /// <param name="socket"> actual stream </param>
193         /// <param name="message"> message to send </param>
194         private async void SendMessage(StreamSocket socket, string message)
195         {
196             DataWriter dataWriter = new DataWriter(socket.OutputStream);
197             var len = dataWriter.MeasureString(message); // Gets the UTF-8
                  string length.
198             dataWriter.WriteInt32((int)len);
199             dataWriter.WriteString(message);
200             var ret = await dataWriter.StoreAsync();
201             dataWriter.DetachStream();
202         }
203 
204         /// <summary>
205         /// Wait data readed if exist
206         /// </summary>
207         /// <param name="socket"></param>
208         private async void WaitForData(StreamSocket socket)
209         {
210             await Task.Delay(TimeSpan.FromMilliseconds(200));
211             DataReader dataReader = new DataReader(socket.InputStream);
212             dataReader.InputStreamOptions = InputStreamOptions.Partial;
213             var msglenght = dataReader.UnconsumedBufferLength;
214             uint stringBytes = msglenght;
215 
216             try
217             {
218                 // Read modification in the stream
219                 stringBytes = await dataReader.LoadAsync(MESSAGE_FULL_LENGHT);
```

```
220
221                 // read message
222                 string msg = dataReader.ReadString(stringBytes);
223
224                 // Send in return if the value exist
225                 if (msg != "")
226                 {
227                     await Task.Delay(TimeSpan.FromMilliseconds(200));
228                     ReplyValues();
229                 }
230             }
231             catch (Exception e)
232             {
233                 string output = e.Message;
234
235                 if (msglenght < 1)
236                     return;
237             }
238
239             WaitForData(socket);
240         }
241
242         /// <summary>
243         /// Send to initialize the raspberry to the server
244         /// </summary>
245         private void SendForInitialize()
246         {
247             // Message send:                                                ↵
                  "@NAME@Connection:IPRASP=x.x.x.x;Location=y;Component=z,z"
248             SendMessage(this.Socket, string.Format(COMMUNICATIONSEPARATOR +  ↵
                  RPINAME + COMMUNICATIONSEPARATOR + "Connection:" + FORMATSTRING, ↵
                   GetHostName(), LOCATION, COMPONENT));
249         }
250
251         /// <summary>
252         /// Send values in reply to the server
253         /// </summary>
254         public void ReplyValues()
255         {
256             // Message receive : "@Reply:TEMP=x;HUMI=y;PRES=z"
257             SendMessage(this.Socket, COMMUNICATIONSEPARATOR + "Reply:" +     ↵
                  this.MSenseHAT.SendValues());
258         }
259
260         /// <summary>
261         /// Get the ip of the raspberry
262         /// </summary>
263         /// <returns>return a string like 192.168.1.2</returns>
264         public string GetHostName()
265         {
266             List<string> IpAddress = new List<string>();
267             var Hosts =                                                      ↵
                  Windows.Networking.Connectivity.NetworkInformation.GetHostNames ↵
                  ().ToList();
268             foreach (var Host in Hosts)
269             {
```

```
270                 string IP = Host.DisplayName;
271                 IpAddress.Add(IP);
272             }
273             return IpAddress.Last();
274         }
275         #endregion
276     }
277 }
278
```