

```
1  /*-----*\
2  * Author    : Salvi Cyril
3  * Date      : 8th juny 2017
4  * Diploma  : RaspiHome
5  * Classroom : T.IS-E2B
6  *
7  * Description:
8  *     RaspiHomeTabletWindows is a program
9  *     compatible with the Windows tablet. It's a
10 *     program that can be use as tactil graphic
11 *     interface to order the component linked with
12 *     the other Raspberry Pi.
13 \*-----*/
14
15 using System;
16 using System.Collections.Generic;
17 using System.Linq;
18 using System.Threading.Tasks;
19 using Windows.Networking;
20 using Windows.Networking.Sockets;
21 using Windows.Storage.Streams;
22 using Windows.UI.Xaml;
23
24 namespace RaspiHomeTabletWindows
25 {
26     public class CommunicationWithServer
27     {
28         #region Fields
29         #region Constants
30         // Default information to connect on the server
31         private const int PORT = 54565;
32         //// Need to be changed fo each configuration
33         private const string IPSEVER = "10.134.97.117";// "192.168.2.8";
34
35         private const string FORMATSTRING = "IPRasp={0};Location={1};Component={2}";
36         private const string COMMUNICATIONSEPARATOR = "@";
37
38         // Important need to be changed if it's another room!
39         private const string LOCATION = "Salon";
40         private const string COMPONENT = "Tablet";
41         private const string RPINAME = "Tablet_" + LOCATION;
42
43         private const int MESSAGE_FULL LENGHT = 512;
44         #endregion
45
46         #region Variables
47         private StreamSocket _socket = new StreamSocket();
48         private StreamSocketListener _listener = new StreamSocketListener();
49         private List<StreamSocket> _connections = new List<StreamSocket>();
50         private bool _isConnected = false;
51         private bool _connecting = false;
52
53         private Windows.Storage.ApplicationDataContainer localSettings =
54             Windows.Storage.ApplicationData.Current.LocalSettings;
55     }
}
```

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

```
112         set
113         {
114             _isConnected = value;
115         }
116     }
117
118     public bool Connecting
119     {
120         get
121         {
122             return _connecting;
123         }
124
125         set
126         {
127             _connecting = value;
128         }
129     }
130
131     public string MessageCommand
132     {
133         get
134         {
135             return _messageCommand;
136         }
137
138         set
139         {
140             _messageCommand = value;
141         }
142     }
143
144     public string NameButtonClicked
145     {
146         get
147         {
148             return _nameButtonClicked;
149         }
150
151         set
152         {
153             _nameButtonClicked = value;
154         }
155     }
156     #endregion
157
158     #region Constructors
159     /// <summary>
160     /// Constructor: Initializer
161     /// </summary>
162     public CommunicationWithServer()
163     {
164         Connect();
165
166         this._dTimer = new DispatcherTimer();
167         this._dTimer.Interval = new TimeSpan(10);
```

```
168         this._dTimer.Tick += _dTimer_Tick;
169
170         this._dTimer.Start();
171     }
172 #endregion
173
174 #region Events
175 private void _dTimer_Tick(object sender, object e)
176 {
177     if (localSettings.Values["SendMessageToServer"] != null)
178     {
179         var messageToSend = localSettings.Values
180             ["SendMessageToServer"];
181         this.SendCommandToServer(messageToSend.ToString());
182         localSettings.Values.Remove("SendMessageToServer");
183     }
184 #endregion
185
186 #region Methods
187 /// <summary>
188 /// Connect the raspberry to the server
189 /// </summary>
190 private async void Connect()
191 {
192     try
193     {
194         this.Connecting = true;
195         await this.Socket.ConnectAsync(new HostName(IPSERVER),
196             PORT.ToString());
197         SendForInitialize();
198         this.Connecting = false;
199         this.IsConnected = true;
200
201         WaitForData(this.Socket);
202     }
203     catch (Exception)
204     {
205         this.Connecting = false;
206         this.IsConnected = false;
207     }
208
209     /// <summary>
210     /// Listen the traffic on the port
211     /// </summary>
212     private async void Listen()
213     {
214         this.Listener.ConnectionReceived += listenerConnectionReceived;
215         await this.Listener.BindServiceNameAsync(PORT.ToString());
216     }
217
218     void listenerConnectionReceived(StreamSocketListener sender,
219         StreamSocketListenerConnectionReceivedEventArgs args)
220     {
221         this.Connections.Add(args.Socket);
222     }
223 }
```

```
221
222     WaitForData(args.Socket);
223 }
224
225 /// <summary>
226 /// Send the message in input to output
227 /// </summary>
228 /// <param name="socket"></param>
229 /// <param name="message"></param>
230 private async void SendMessage(StreamSocket socket, string message)
231 {
232     DataWriter dataWriter = new DataWriter(socket.OutputStream);
233     var len = dataWriter.MeasureString(message); // Gets the UTF-8
234         string length.
235     dataWriter.WriteInt32((int)len);
236     dataWriter.WriteString(message);
237     var ret = await dataWriter.StoreAsync();
238     dataWriter.DetachStream();
239 }
240
241 /// <summary>
242 /// Send to initialize the raspberry to the server
243 /// </summary>
244 private void SendForInitialize()
245 {
246     SendMessage(this.Socket, string.Format(COMMUNICATIONSEPARATOR +
247         RPINAME + COMMUNICATIONSEPARATOR + "Connection:" + FORMATSTRING,
248         GetHostName(), LOCATION, COMPONENT));
249 }
250
251 /// <summary>
252 /// Send the command to the server
253 /// </summary>
254 public void SendCommandToServer(string message)
255 {
256     SendMessage(this.Socket, COMMUNICATIONSEPARATOR + "Send:" +
257         message);
258     this.MessageCommand = message;
259 }
260
261 /// <summary>
262 /// Wait data readed if exist
263 /// </summary>
264 /// <param name="socket"></param>
265 private async void WaitForData(StreamSocket socket)
266 {
267     DataReader dataReader = new DataReader(socket.InputStream);
268     dataReader.InputStreamOptions = InputStreamOptions.Partial;
269     var messageLenght = dataReader.UnconsumedBufferLength;
270     uint stringBytes = messageLenght;
271
272     try
273     {
274         // Read modification in the stream
275         stringBytes = await dataReader.LoadAsync(MESSAGE_FULL_LENGTH);
276     }
277 }
```

```
273         // read message
274         string messageRead = dataReader.ReadString(stringBytes);
275
276         await Task.Delay(TimeSpan.FromMilliseconds(200));
277         // Store value
278         localSettings.Values["ReceiveMessageFromServer"] = messageRead;
279     }
280     catch (Exception e)
281     {
282         string output = e.Message;
283
284         if (messageLenght < 1)
285             return;
286     }
287
288     WaitForData(socket);
289 }
290
291 /// <summary>
292 /// Get the ip of the raspberry
293 /// </summary>
294 /// <returns>return a string like 192.168.1.2</returns>
295 public string GetHostName()
296 {
297     List<string> IPAddress = new List<string>();
298     var Hosts =
299         Windows.Networking.Connectivity.NetworkInformation.GetHostNames
300         ().ToList();
301     foreach (var Host in Hosts)
302     {
303         string IP = Host.DisplayName;
304         IPAddress.Add(IP);
305     }
306     return IPAddress.Last();
307 }
308 #endregion
309 }
```