

# CENTRE DE FORMATION PROFESSIONNELLE TECHNIQUE

## TRAVAIL DE DIPLÔME

---

# RaspiHome

---

## Programmes du projet

*Auteur :*

Salvi Cyril

*Encadrant :*

M. BONVIN PASCAL

T.IS-E2B

2016 - 2017

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using Windows.Media.SpeechRecognition;
5  using Windows.Devices.Spi;
6  using System.Text;
7  using System.Threading;
8  using System.Threading.Tasks;
9  using Windows.Devices.Enumeration;
10
11 namespace RaspiHomeSpeechNSynthetize
12 {
13     public class Speecher
14     {
15         #region Fields
16         #region Constants
17         #endregion
18
19         #region Variable
20         private Synthetizer _rhSynt;
21         private Commands _rhCommands;
22         private CommunicationWithServer _comWithServer;
23
24         private SpeechRecognizer _recoEngine = null;
25         private SpiDevice _mcp3202;
26
27         private bool _isRaspiCalled = false;
28         #endregion
29         #endregion
30
31         #region Properties
32         public Synthetizer RhSynt
33         {
34             get
35             {
36                 return _rhSynt;
37             }
38
39             set
40             {
41                 _rhSynt = value;
42             }
43         }
44
45         public bool IsRaspiCalled
46         {
47             get
48             {
49                 return _isRaspiCalled;
50             }
51
52             set
53             {
54                 _isRaspiCalled = value;
55             }
56         }
57     }
58 }
```

```
57      public CommunicationWithServer ComWithServer
58      {
59          get
60          {
61              return _comWithServer;
62          }
63
64          set
65          {
66              _comWithServer = value;
67          }
68      }
69
70
71      public Commands RhCommands
72      {
73          get
74          {
75              return _rhCommands;
76          }
77
78          set
79          {
80              _rhCommands = value;
81          }
82      }
83 #endregion
84
85 #region Constructor
86 /// <summary>
87 /// Constructor: Initialize
88 /// </summary>
89 public Speecher()
90 {
91     // Initialize the synthetizer
92     this.RhSynt = new Synthesizer(this);
93     this.RhCommands = new Commands();
94
95     // Initialize the communication with the server
96     this.ComWithServer = new CommunicationWithServer(this);
97
98     // Create the speech recognition object
99     this._recoEngine = new SpeechRecognizer();
100
101    // Initialize the recognition
102    InitializeSpeechRecognizer();
103 }
104 #endregion
105
106 #region Methods
107 /// <summary>
108 /// Initialize the Spi communication
109 /// </summary>
110 private async void InitializeSpi()
111 {
112     //using SPI0 on the Pi
```

```
113         var spiSettings = new SpiConnectionSettings(0); //for spi bus index ↗
114             0
115             spiSettings.ClockFrequency = 3600000; //3.6 MHz
116             spiSettings.Mode = SpiMode.Mode0;
117
118             string spiQuery = SpiDevice.GetDeviceSelector("SPI0");
119
120             var deviceInfo = await DeviceInformation.FindAllAsync(spiQuery);
121             if (deviceInfo != null && deviceInfo.Count > 0)
122             {
123                 _mcp3202 = await SpiDevice.FromIdAsync(deviceInfo[0].Id,      ↗
124                     spiSettings);
125             }
126
127             /// <summary>
128             /// Read digital input with SPI
129             /// </summary>
130             /// <returns></returns>
131             private string ReadSpiData()
132             {
133                 byte[] transmitBuffer = new byte[3] { 0x01, 0x80, 0 };
134                 byte[] receiveBuffer = new byte[3];
135
136                 string result = "";
137
138                 _mcp3202.TransferFullDuplex(transmitBuffer, receiveBuffer);
139
140                 return result = Encoding.UTF8.GetString(receiveBuffer);
141             }
142
143             /// <summary>
144             /// (obsolete on UWP) Set the configuration of the speecher
145             /// </summary>
146             private void InitializeSpeechRecognizer()
147             {
148
149             /// <summary>
150             /// (obsolete on UWP) Enable the speech, used when raspi is not      ↗
151                 talking
152             /// </summary>
153             public void EnableSpeech()
154             {
155                 //this._recoEngine.RecognizeAsync(RecognizeMode.Multiple);
156             }
157
158             /// <summary>
159             /// (obsolete on UWP) Disable the speech, used when raspi is talking
160             /// </summary>
161             public void DisableSpeech()
162             {
163                 // this._recoEngine.RecognizeAsyncStop();
164             }
165
```

```
166     /// <summary>
167     /// Used to call raspi
168     /// </summary>
169     /// <param name="nameMentioned"></param>
170     private void CallRaspi(string nameMentioned)
171     {
172         DisableSpeech();
173
174         this.RhSynt.RaspiCalled(nameMentioned);
175
176         EnableSpeech();
177     }
178
179     /// <summary>
180     /// Used to send the command after raspi called
181     /// </summary>
182     /// <param name="brutCommand"></param>
183     public void SendBrutCommand(string brutCommand)
184     {
185         DisableSpeech();
186
187         this.ComWithServer.SendCommandToServer(brutCommand); ↗
188     }
189
190     /// <summary>
191     /// Reply message to the sender (reply message error if command ↗
192     /// unknow)
193     /// </summary>
194     /// <param name="messageReply"></param>
195     /// <param name="messageCommand"></param>
196     public void ReplyForSynthetize(string messageReply, string ↗
197     messageCommand)
198     {
199         if (messageReply == "ERROR_MESSAGE")
200         {
201             this.RhSynt.WrongCommand(); ↗
202         }
203         else
204         {
205             this.RhSynt.RaspiSayInformation
206                 (this.RhSynt.SetProprelyInformations(messageReply, ↗
207                 messageCommand));
208         }
209         this.IsRaspiCalled = false; ↗
210
211         EnableSpeech();
212     }
213 #endregion
214 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Globalization;
4  using System.Linq;
5  using System.Text;
6  using Windows.Media.SpeechSynthesis;
7  using System.Threading;
8  using System.Threading.Tasks;
9  using Windows.UI.Xaml.Controls;
10 using Windows.UI.Core;
11
12 namespace RaspiHomeSpeechNSynthetize
13 {
14     public class Synthetizer
15     {
16         #region Fields
17         #region Constants
18         private const string RASPI_NAME = "raspi";
19         private const char SEPARATOR = ' ';
20         // Change value when new update ("en" to "fr")
21         private const string LANGUAGE_SELECTION = "en";
22         private const int TIME_TO_WAIT = 3;
23         #endregion
24
25         #region Variable
26         private Speecher _rhSpeech;
27
28         private Commands _rhCommands;
29
30         private Random _rnd;
31         private SpeechSynthesizer _voice;
32         private string _commandReceiveStr = "";
33         private string _commandToSend = "";
34
35         private bool _isCalled = false;
36         private bool _isCompleted = false;
37         private List<string> _lstSentenceSplited;
38         #endregion
39         #endregion
40
41         #region Properties
42         public Speecher RhSpeech
43         {
44             get
45             {
46                 return _rhSpeech;
47             }
48
49             set
50             {
51                 _rhSpeech = value;
52             }
53         }
54
55         public string CommandReceiveStr
56         {
```

```
57         get
58     {
59         return _commandReceiveStr;
60     }
61
62     set
63     {
64         _commandReceiveStr = value;
65     }
66 }
67
68     public bool IsCalled
69     {
70         get
71     {
72         return _isCalled;
73     }
74
75         set
76     {
77         _isCalled = value;
78     }
79 }
80
81     public bool IsCompleted
82     {
83         get
84     {
85         return _isCompleted;
86     }
87
88         set
89     {
90         _isCompleted = value;
91     }
92 }
93
94     public Commands RhCommands
95     {
96         get
97     {
98         return _rhCommands;
99     }
100
101         set
102     {
103         _rhCommands = value;
104     }
105 }
106 #endregion
107
108 #region Constructor
109 /// <summary>
110 /// Constructor: Initialize
111 /// </summary>
112 public Synthetizer(Speecher paramSpeecher)
```

```
113     {
114         this.RhSpeech = paramSpeecher;
115         this.RhCommands = new Commands();
116         this._rnd = new Random();
117         this._voice = new SpeechSynthesizer();
118         this._lstSentenceSplited = new List<string>();
119     }
120 #endregion
121
122 #region Methods
123 /// <summary>
124 /// Raspberry processus, wait calling to start communication with      ↗
125 /// </summary>
126 private void RaspiProcessus()
127 {
128     if (this.IsCalled)
129     {
130         SendCommand();
131     }
132 }
133
134 /// <summary>
135 /// Send to the synthetize method the order to reply
136 /// </summary>
137 /// <param name="name"></param>
138 public void RaspiCalled(string name)
139 {
140     string raspiName = RemoveDiacritics(name).ToLower();
141
142     if (raspiName == RASPI_NAME.ToLower())
143     {
144         RaspiCalled(this.RhCommands.WhenCalling);
145         this.RhSpeech.IsRaspiCalled = true;
146         this.IsCalled = true;
147     }
148 }
149
150 /// <summary>
151 /// Send the instruction for the Raspberry
152 /// </summary>
153 /// <returns></returns>
154 public string SendCommand()
155 {
156     this.RhSpeech.IsRaspiCalled = false;
157     this.IsCalled = false;
158
159     this.IsCompleted = true;
160     return this._commandToSend;
161 }
162
163 /// <summary>
164 /// Processus to choose the sentence to say
165 /// </summary>
166 /// <param name="repertory"> List of sentence to say </param>
167 private void RaspiCalled(List<string> repertory)
```

```
168     {
169         string messageToSay = repertory[_rnd.Next(0, repertory.Count - 1)];
170         this.RaspiTalk(messageToSay);
171     }
172 }
173
174 /// <summary>
175 /// Allow the raspi to let her talk
176 /// </summary>
177 /// <param name="messageToSay"> sentence to say </param>
178 private async void RaspiTalk(string messageToSay)
179 {
180     // Get the output element (audio jack)
181     MediaElement mediaElement = new MediaElement();
182     SpeechSynthesizer synth = new SpeechSynthesizer();
183
184     // Set the default language
185     foreach (VoiceInformation vInfo in SpeechSynthesizer.AllVoices)
186     {
187         if (vInfo.Language.Contains(LANGUAGE_SELECTION))
188         {
189             synth.Voice = vInfo;
190             break;
191         }
192         else
193             synth.Voice = vInfo;
194     }
195
196     SpeechSynthesisStream synthStream = await
197         synth.SynthesizeTextToStreamAsync(messageToSay);
198
199     mediaElement.SetSource(synthStream, synthStream.ContentType);
200     // 0 = min / 1 = max
201     mediaElement.Volume = 1;
202     mediaElement.Play();
203
204     // Work like Thread.Sleep(TIME_TO_WAIT)
205     await Task.Delay(TimeSpan.FromSeconds(TIME_TO_WAIT));
206 }
207
208 /// <summary>
209 /// Called when there is any error
210 /// </summary>
211 public void WrongCommand()
212 {
213     // Reach the error request sentences to say
214     this.RaspiCalled(this.RhCommands.SpeicherRespondingRequestError);
215 }
216
217 /// <summary>
218 /// Allow the Raspi, to let her talk with list of information
219 /// </summary>
220 /// <param name="informationsToGive"></param>
221 public void RaspiSayInformation(List<string> informationsToGive)
```

```
222         foreach (string informationToSay in informationsToGive)
223         {
224             if (informationToSay != "")
225             {
226                 this.RaspiTalk(informationToSay);
227             }
228         }
229     }
230 
231     /// <summary>
232     /// Set information to sythetize
233     /// </summary>
234     public List<string> SetProprelyInformations(string messageReply,      ↪
235     string messageCommand)
236     {
237         List<string> result = new List<string>();
238 
239         bool temp = false, humi = false, pres = false;
240 
241         if (messageCommand.Contains("température"))
242             temp = true;
243         if (messageCommand.Contains("humidité"))
244             humi = true;
245         if (messageCommand.Contains("pression"))
246             pres = true;
247         if (messageCommand.Contains("état"))
248         {
249             temp = true;
250             humi = true;
251             pres = true;
252         }
253 
254         string[] informationSplited = messageReply.Split(';');
255         if (informationSplited[0] != "")
256         {
257             foreach (string information in informationSplited)
258             {
259                 switch (information.Split('=').First())
260                 {
261                     case "TEMP":
262                         if (temp)
263                         {
264                             result.Add
265                             (this.RhCommands.SpeecherRespondingEtatRequest[0] +
266                             information.Split('=').Last() +           ↪
267                             this.RhCommands.SpeecherRespondingEtatRequest[1]);           ↪
268                         }
269                         break;
270                     case "HUMI":
271                         if (humi)
272                         {
273                             result.Add
274                             (this.RhCommands.SpeecherRespondingEtatRequest[4] +
275                             information.Split('=').Last() +           ↪
276                             this.RhCommands.SpeecherRespondingEtatRequest[5]);           ↪
277                         }
278                 }
279             }
280         }
281     }
```

```
    this.RhCommands.SpeecherRespondingEstatRequest[5]);
274        }
275        break;
276    case "PRES":
277        if (pres)
278        {
279            result.Add
280                (this.RhCommands.SpeecherRespondingEstatRequest[6] +
281                 information.Split('=').Last() +
282
283            this.RhCommands.SpeecherRespondingEstatRequest[7]);
284        }
285        break;
286    }
287    else
288        result.Add("");
289    return result;
290}
291
292 /// <summary>
293 /// Stack Overflow solution to delete accents in strings
294 /// http://stackoverflow.com/questions/249087/how-do-i-remove-
295     diacritics-accents-from-a-string-in-net
296 /// </summary>
297 /// <param name="str"></param>
298 /// <returns></returns>
299 static string RemoveDiacritics(string str)
300 {
301     var normalizedString = str.Normalize(NormalizationForm.FormD);
302     var stringBuilder = new StringBuilder();
303
304     foreach (var c in normalizedString)
305     {
306         var unicodeCategory = CharUnicodeInfo.GetUnicodeCategory(c);
307         if (unicodeCategory != UnicodeCategory.NonSpacingMark)
308         {
309             stringBuilder.Append(c);
310         }
311     }
312
313     return stringBuilder.ToString().Normalize
314         (NormalizationForm.FormC);
315 }
316 #endregion
317 }
```

```
1  using RaspiHomeSpeechNSynthetize;
2  using System;
3  using System.Collections.Generic;
4  using System.Diagnostics;
5  using System.Linq;
6  using System.Net;
7  using System.Net.Sockets;
8  using System.Text;
9  using System.Threading;
10 using System.Threading.Tasks;
11 using Windows.ApplicationModel.Core;
12 using Windows.Networking;
13 using Windows.Networking.Connectivity;
14 using Windows.Networking.Sockets;
15 using Windows.Storage.Streams;
16
17 namespace RaspiHomeSpeechNSynthetize
18 {
19     public class CommunicationWithServer
20     {
21         #region Fields
22         #region Constants
23             // Default information to connect on the server
24             private const int PORT = 54565;
25             //// Need to be changed fo each configuration
26             private const string IPSERVER = "10.134.97.117";// "192.168.2.8";
27
28             private const string FORMATSTRING = "IPRasp={0};Location=
29                 {1};Component={2}";
30             private const string COMMUNICATIONSEPARATOR = "@";
31
32             // Important need to be changed if it's another room!
33             private const string LOCATION = "Salon";
34             private const string COMPONENT = "Microphone";
35             private const string RPINAME = "Microphone_" + LOCATION;// 
36                 "192.168.2.8";
37
38             private const int MESSAGE_FULL_LENGTH = 512;
39             #endregion
40
41             #region Variables
42             private Speecher _speecher;
43
44             private StreamSocket _socket = new StreamSocket();
45             private StreamSocketListener _listener = new StreamSocketListener();
46             private List<StreamSocket> _connections = new List<StreamSocket>();
47             private bool _isConnected = false;
48             private bool _connecting = false;
49
50             private string _messageCommand = "";
51             #endregion
52             #endregion
53             public Speecher Speecher
54             {
```

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

```
111         set
112     {
113         _isConnected = value;
114     }
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 #endregion
144
145 #region Constructors
146 /// <summary>
147 /// Constructor: Initializer
148 /// </summary>
149 /// <param name="paramModel"></param>
150 public CommunicationWithServer(Speecher paramModel)
151 {
152     this.Speecher = paramModel;
153
154     Connect();
155 }
156 #endregion
157
158 #region Methods
159 /// <summary>
160 /// Connect the raspberry to the server
161 /// </summary>
162 private async void Connect()
163 {
164     try
165     {
166         this.Connecting = true;
```

```
167         await this.Socket.ConnectAsync(new HostName(IPSERVER),  
168             PORT.ToString());  
169         SendForInitialize();  
170         this.Connecting = false;  
171         this.isConnected = true;  
172         WaitForData(this.Socket);  
173     }  
174     catch (Exception)  
175     {  
176         this.Connecting = false;  
177         this.isConnected = false;  
178     }  
179 }  
180  
181     /// <summary>  
182     /// Listen the traffic on the port  
183     /// </summary>  
184     private async void Listen()  
185     {  
186         this.Listener.ConnectionReceived += listenerConnectionReceived;  
187         await this.Listener.BindServiceNameAsync(PORT.ToString());  
188     }  
189  
190     void listenerConnectionReceived(StreamSocketListener sender,  
191         StreamSocketListenerConnectionReceivedEventArgs args)  
192     {  
193         this.Connections.Add(args.Socket);  
194         WaitForData(args.Socket);  
195     }  
196  
197     /// <summary>  
198     /// Send the message in input to output  
199     /// </summary>  
200     /// <param name="socket"></param>  
201     /// <param name="message"></param>  
202     private async void SendMessage(StreamSocket socket, string message)  
203     {  
204         DataWriter dataWriter = new DataWriter(socket.OutputStream);  
205         var len = dataWriter.MeasureString(message); // Gets the UTF-8  
206             string length.  
207         dataWriter.WriteInt32((int)len);  
208         dataWriter.WriteString(message);  
209         var ret = await dataWriter.StoreAsync();  
210         dataWriter.DetachStream();  
211     }  
212  
213     /// <summary>  
214     /// Send to initialize the raspberry to the server  
215     /// </summary>  
216     private void SendForInitialize()  
217     {  
218         SendMessage(this.Socket, string.Format(COMMUNICATIONSEPARATOR +  
219             RPINAME + COMMUNICATIONSEPARATOR + "Connection:" + FORMATSTRING,  
220             GetHostName(), LOCATION, COMPONENT));  
221     }
```

```
218     }
219
220     /// <summary>
221     /// Send the command to the server
222     /// </summary>
223     public void SendCommandToServer(string message)
224     {
225         SendMessage(this.Socket, COMMUNICATIONSEPARATOR + "Send:" +
226                     message);
227         this.MessageCommand = message;
228     }
229
230     /// <summary>
231     /// Wait data readed if exist
232     /// </summary>
233     /// <param name="socket"></param>
234     private async void WaitForData(StreamSocket socket)
235     {
236         DataReader dataReader = new DataReader(socket.InputStream);
237         dataReader.InputStreamOptions = InputStreamOptions.Partial;
238         var messageLenght = dataReader.UnconsumedBufferLength;
239         uint stringBytes = messageLenght;
240
241         try
242         {
243             // Read modification in the stream
244             stringBytes = await dataReader.LoadAsync(MESSAGE_FULL_LENGTH);
245
246             // read message
247             string messageRead = dataReader.ReadString(stringBytes);
248
249             // Send in return if the value exist
250             if (messageRead != "")
251             {
252                 this.Speecher.ReplyForSynthetize(messageRead,
253                                                 this.MessageCommand);
254             }
255
256             messageRead = "";
257         }
258         catch (Exception ) { }
259
260         WaitForData(socket);
261     }
262
263     /// <summary>
264     /// Get the ip of the raspberry
265     /// </summary>
266     /// <returns>return a string like 192.168.1.2</returns>
267     public string GetHostName()
268     {
269         List<string> IpAddress = new List<string>();
270         var Hosts =
271             Windows.Networking.Connectivity.NetworkInformation.GetHostNames
272             ().ToList();
273         foreach (var Host in Hosts)
```

```
270         {
271             string IP = Host.DisplayName;
272             IpAddress.Add(IP);
273         }
274         return IpAddress.Last();
275     }
276     #endregion
277 }
278 }
279 }
```

```
1 <Page
2     x:Class="RaspiHomeSpeechNSynthetize.MainPage"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeSpeechNSynthetize"
6     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8     mc:Ignorable="d">
9
10    <StackPanel Orientation="Horizontal" Background="White"             ↵
11        HorizontalAlignment="Center" VerticalAlignment="Center">
12            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center"   ↵
13                Margin="0,0,50,0">
14                <StackPanel Orientation="Horizontal">
15                    <TextBox x:Name="tbxLightCommand1" Width="250" Text="Allume la    ↵
16                        lumière du salon"/>
17                    <Button x:Name="btnLightCommand1" Width="55" Content="Send"      ↵
18                        Margin="40,0,0,0" Click="btnLightCommand1_Click"/>
19                </StackPanel>
20
21                <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
22                    <TextBox x:Name="tbxLightCommand2" Width="250" Text="Éteins la    ↵
23                        lumière du salon"/>
24                    <Button x:Name="btnLightCommand2" Width="55" Content="Send"      ↵
25                        Margin="40,0,0,0" Click="btnLightCommand2_Click"/>
26                </StackPanel>
27
28                <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
29                    <TextBox x:Name="tbxState" Width="250" Text="Quel est la    ↵
30                        température du salon"/>
31                    <Button x:Name="btnState" Width="55" Content="Send"          ↵
32                        Margin="40,0,0,0" Click="btnState_Click"/>
33                </StackPanel>
34
35                <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center"   ↵
36                    Margin="0,0,50,0">
37                    <StackPanel Orientation="Horizontal">
38                        <TextBox x:Name="tbxStore1" Width="250" Text="Monte le store du    ↵
39                            salon"/>
40                        <Button x:Name="btnSendCommand" Width="55" Content="Send"      ↵
41                            Margin="40,0,0,0" Click="btnStore1_Click"/>
42                    </StackPanel>
43
44                    <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
45                        <TextBox x:Name="tbxStore2" Width="250" Text="Descends le store    ↵
46                            du salon"/>
47                        <Button x:Name="btnStore2" Width="55" Content="Send"          ↵
48                            Margin="40,0,0,0" Click="btnStore2_Click"/>
49                    </StackPanel>
50
51                    <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
52                        <TextBox x:Name="tbxStore3" Width="250" Text="Stop le store du    ↵
53                            salon"/>
54                        <Button x:Name="btnStore3" Width="55" Content="Send"          ↵
55                            Margin="40,0,0,0" Click="btnStore3_Click"/>
56                    </StackPanel>
57
```

```
42      </StackPanel>
43      </StackPanel>
44    </StackPanel>
45  </Page>
46
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Runtime.InteropServices.WindowsRuntime;
6  using Windows.Foundation;
7  using Windows.Foundation.Collections;
8  using Windows.UI.Xaml;
9  using Windows.UI.Xaml.Controls;
10 using Windows.UI.Xaml.Controls.Primitives;
11 using Windows.UI.Xaml.Data;
12 using Windows.UI.Xaml.Input;
13 using Windows.UI.Xaml.Media;
14 using Windows.UI.Xaml.Navigation;
15
16 // Pour plus d'informations sur le modèle d'élément Page vierge, consultez la ↵
17 // page http://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409
18
19 namespace RaspiHomeSpeechNSynthetize
20 {
21     /// <summary>
22     /// Une page vide peut être utilisée seule ou constituer une page de destination au sein d'un frame.
23     /// </summary>
24     public sealed partial class MainPage : Page
25     {
26         #region Fields
27         private Speecher _speecher;
28         #endregion
29
30         #region Constuctor
31         public MainPage()
32         {
33             this.InitializeComponent();
34
35             this._speecher = new Speecher();
36         }
37         #endregion
38
39         #region Event
40         private void btnLightCommand1_Click(object sender, RoutedEventArgs e)
41         {
42             if (this.tbxLightCommand1.Text != "")
43                 this._speecher.SendBrutCommand(this.tbxLightCommand1.Text);
44             else
45                 this._speecher.SendBrutCommand("Allumer lumiere");
46         }
47
48         private void btnLightCommand2_Click(object sender, RoutedEventArgs e)
49         {
50             if (this.tbxLightCommand2.Text != "")
51                 this._speecher.SendBrutCommand(this.tbxLightCommand2.Text);
52             else
53                 this._speecher.SendBrutCommand("Eteindre lumiere");
54         }
55 }
```

```
55     private void btnState_Click(object sender, RoutedEventArgs e)
56     {
57         if (this.tbxState.Text != "")
58             this._speicher.SendBrutCommand(this.tbxState.Text);
59         else
60             this._speicher.SendBrutCommand("Temperature du salon");
61     }
62
63     private void btnStore1_Click(object sender, RoutedEventArgs e)
64     {
65         if (this.tbxStore1.Text != "")
66             this._speicher.SendBrutCommand(this.tbxStore1.Text);
67         else
68             this._speicher.SendBrutCommand("Monter store");
69     }
70
71     private void btnStore2_Click(object sender, RoutedEventArgs e)
72     {
73         if (this.tbxStore2.Text != "")
74             this._speicher.SendBrutCommand(this.tbxStore2.Text);
75         else
76             this._speicher.SendBrutCommand("Descendre store");
77     }
78
79     private void btnStore3_Click(object sender, RoutedEventArgs e)
80     {
81         if (this.tbxStore3.Text != "")
82             this._speicher.SendBrutCommand(this.tbxStore3.Text);
83         else
84             this._speicher.SendBrutCommand("Stopper store");
85     }
86     #endregion
87 }
88 }
```

```
1  using System.Collections.Generic;
2
3  namespace RaspiHomeSpeechNSynthetize
4  {
5      public class Commands
6      {
7          #region Commands variable
8          private List<string> _whenCalling = new List<string>()
9          {
10              "Oui, que puis-je faire pour vous?", "Comment puis-je vous
11                  servir?", "Je suis toute ouïe"
12          };
13
14          private List<string> _whenCallingError = new List<string>()
15          {
16              "Je ne m'appelle pas comme ça!", "Je crois que vous vous êtes trompé ↵
17                  de personne!"
18          };
19
20          private List<string> _speecherRespondingRequest = new List<string>()
21          {
22              "Tout de suite", "Je m'y mets de ce pas!", "Ne quittez pas!"
23          };
24
25          private List<string> _speecherRespondingRequestError = new
26              List<string>()
27          {
28              "Je n'ai pas compris ce que vous avez demandé"
29          };
30
31          private List<string> _raspiHomeObjectKnown = new List<string>()
32          {
33              "lumiere", "lumieres",
34              "store", "stores",
35              "television", "televisions",
36              "porte", "portes",
37              "fenetre", "fenetres",
38          };
39
40          private List<string> _raspiHomeActionKnown = new List<string>()
41          {
42              "allumer", "allume",
43              "eteindre", "eteins",
44              "monter", "monte",
45              "descendre", "descends",
46              "stopper", "stop",
47              "ouvrir", "ouvre",
48              "fermer", "ferme",
49          };
50
51          private List<string> _speecherRespondingEstatRequest = new List<string>() ↵
52          ()
53          {
54              "Il fait , , degrés Celsius , , degrés Farad", " , degrés Kelvin", "Le ↵
55                  taux d'humidité est de , , pourcent", "La pression est ↵
56                  actuellement de , , hecato Pascal"
57          };
58      }
59  }
```

```
51         };
52
53         private List<string> _raspiHomeActionWithoutObjectKnown = new
54             List<string>()
55         {
56             "temperature", "humidite", "pression", "etat"
57         };
58
59         private List<string> _raspiHomeLocationKnown = new List<string>()
60             {
61                 "maison", "salon", "cuisine", "parent", "enfant", "bureau", "toilette", "bain"
62             };
63
64         private List<string> _raspiHomeCommandUselessConnecter = new
65             List<string>()
66         {
67             "le", "la", "les", "un", "une", "des", "mon", "ma", "mes", "son", "sa", "ses", "ce", "cet",
68             "cette", "ces", "cel", "celle", "du", "de"
69         };
70
71         private List<string> _raspiHomeGrammarCommand = new List<string>()
72         {
73             "allume la lumière", "allume les lumières", "allumer la
74                 lumière", "allumer les lumières",
75             "éteint la lumière", "éteint les lumières", "éteindre la
76                 lumière", "éteindre les lumières",
77             "monte le store", "monte les stores", "monter le store", "monter les
78                 stores",
79             "descend le store", "descend les stores", "descendre le
80                 store", "descendre les stores",
81             "ouvre le store", "ouvre les stores", "ouvrir le store", "ouvrir
82                 les stores",
83             "ferme le store", "ferme les stores", "fermer le store", "fermer
84                 les stores",
85             "quelle est la température", "quelle est l'humidité", "quelle est
86                 la pression", "quelle est l'état",
87             "de la maison", "de la cuisine", "du salon", "de la salle de
88                 bain", "de la chambre", "de la pièce",
89         };
90     #endregion
91
92     #region Properties
93     public List<string> WhenCalling
94     {
95         get
96         {
97             return _whenCalling;
98         }
99
100        set
101        {
102            _whenCalling = value;
103        }
104    }
```

```
94      public List<string> WhenCallingError
95      {
96          get
97          {
98              return _whenCallingError;
99          }
100         set
101         {
102             _whenCallingError = value;
103         }
104     }
105
106
107
108     public List<string> SpeecherRespondingRequest
109     {
110         get
111         {
112             return _speecherRespondingRequest;
113         }
114         set
115         {
116             _speecherRespondingRequest = value;
117         }
118     }
119 }
120
121     public List<string> SpeecherRespondingRequestError
122     {
123         get
124         {
125             return _speecherRespondingRequestError;
126         }
127         set
128         {
129             _speecherRespondingRequestError = value;
130         }
131     }
132 }
133
134     public List<string> RaspiHomeObjectKnown
135     {
136         get
137         {
138             return _raspiHomeObjectKnown;
139         }
140         set
141         {
142             _raspiHomeObjectKnown = value;
143         }
144     }
145 }
146
147     public List<string> RaspiHomeActionKnown
148     {
149         get
```

```
150         {
151             return _raspiHomeActionKnown;
152         }
153
154         set
155         {
156             _raspiHomeActionKnown = value;
157         }
158     }
159
160     public List<string> SpeecherRespondingEtatRequest
161     {
162         get
163         {
164             return _speecherRespondingEtatRequest;
165         }
166
167         set
168         {
169             _speecherRespondingEtatRequest = value;
170         }
171     }
172
173     public List<string> RaspiHomeActionWithoutObjectKnown
174     {
175         get
176         {
177             return _raspiHomeActionWithoutObjectKnown;
178         }
179
180         set
181         {
182             _raspiHomeActionWithoutObjectKnown = value;
183         }
184     }
185
186     public List<string> RaspiHomeLocationKnown
187     {
188         get
189         {
190             return _raspiHomeLocationKnown;
191         }
192
193         set
194         {
195             _raspiHomeLocationKnown = value;
196         }
197     }
198
199     public List<string> RaspiHomeCommandUselessConnecter
200     {
201         get
202         {
203             return _raspiHomeCommandUselessConnecter;
204         }
205     }
```

```
206         set
207         {
208             _raspiHomeCommandUselessConnector = value;
209         }
210     }
211
212     public List<string> RaspiHomeGrammarCommand
213     {
214         get
215         {
216             return _raspiHomeGrammarCommand;
217         }
218
219         set
220         {
221             _raspiHomeGrammarCommand = value;
222         }
223     }
224 #endregion
225 }
226 }
227 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using Windows.Media.SpeechRecognition;
5  using Windows.Devices.Spi;
6  using System.Text;
7  using System.Threading;
8  using System.Threading.Tasks;
9  using Windows.Devices.Enumeration;
10
11 namespace RaspiHomeSpeechNSynthetize
12 {
13     public class Speecher
14     {
15         #region Fields
16         #region Constants
17         #endregion
18
19         #region Variable
20         private Synthetizer _rhSynt;
21         private Commands _rhCommands;
22         private CommunicationWithServer _comWithServer;
23
24         private SpeechRecognizer _recoEngine = null;
25         private SpiDevice _mcp3202;
26
27         private bool _isRaspiCalled = false;
28         #endregion
29         #endregion
30
31         #region Properties
32         public Synthetizer RhSynt
33         {
34             get
35             {
36                 return _rhSynt;
37             }
38
39             set
40             {
41                 _rhSynt = value;
42             }
43         }
44
45         public bool IsRaspiCalled
46         {
47             get
48             {
49                 return _isRaspiCalled;
50             }
51
52             set
53             {
54                 _isRaspiCalled = value;
55             }
56         }
57     }
58 }
```

```
57      public CommunicationWithServer ComWithServer
58      {
59          get
60          {
61              return _comWithServer;
62          }
63
64          set
65          {
66              _comWithServer = value;
67          }
68      }
69
70
71      public Commands RhCommands
72      {
73          get
74          {
75              return _rhCommands;
76          }
77
78          set
79          {
80              _rhCommands = value;
81          }
82      }
83 #endregion
84
85 #region Constructor
86 /// <summary>
87 /// Constructor: Initialize
88 /// </summary>
89 public Speecher()
90 {
91     // Initialize the synthetizer
92     this.RhSynt = new Synthesizer(this);
93     this.RhCommands = new Commands();
94
95     // Initialize the communication with the server
96     this.ComWithServer = new CommunicationWithServer(this);
97
98     // Create the speech recognition object
99     this._recoEngine = new SpeechRecognizer();
100
101    // Initialize the recognition
102    InitializeSpeechRecognizer();
103 }
104 #endregion
105
106 #region Methods
107 /// <summary>
108 /// Initialize the Spi communication
109 /// </summary>
110 private async void InitializeSpi()
111 {
112     //using SPI0 on the Pi
```

```
113         var spiSettings = new SpiConnectionSettings(0); //for spi bus index ↗
114             0
115             spiSettings.ClockFrequency = 3600000; //3.6 MHz
116             spiSettings.Mode = SpiMode.Mode0;
117
118             string spiQuery = SpiDevice.GetDeviceSelector("SPI0");
119
120             var deviceInfo = await DeviceInformation.FindAllAsync(spiQuery);
121             if (deviceInfo != null && deviceInfo.Count > 0)
122             {
123                 _mcp3202 = await SpiDevice.FromIdAsync(deviceInfo[0].Id,      ↗
124                     spiSettings);
125             }
126
127             /// <summary>
128             /// Read digital input with SPI
129             /// </summary>
130             /// <returns></returns>
131             private string ReadSpiData()
132             {
133                 byte[] transmitBuffer = new byte[3] { 0x01, 0x80, 0 };
134                 byte[] receiveBuffer = new byte[3];
135
136                 string result = "";
137
138                 _mcp3202.TransferFullDuplex(transmitBuffer, receiveBuffer);
139
140                 return result = Encoding.UTF8.GetString(receiveBuffer);
141             }
142
143             /// <summary>
144             /// (obsolete on UWP) Set the configuration of the speecher
145             /// </summary>
146             private void InitializeSpeechRecognizer()
147             {
148
149             /// <summary>
150             /// (obsolete on UWP) Enable the speech, used when raspi is not      ↗
151                 talking
152             /// </summary>
153             public void EnableSpeech()
154             {
155                 //this._recoEngine.RecognizeAsync(RecognizeMode.Multiple);
156             }
157
158             /// <summary>
159             /// (obsolete on UWP) Disable the speech, used when raspi is talking
160             /// </summary>
161             public void DisableSpeech()
162             {
163                 // this._recoEngine.RecognizeAsyncStop();
164             }
165
```

```
166     /// <summary>
167     /// Used to call raspi
168     /// </summary>
169     /// <param name="nameMentioned"></param>
170     private void CallRaspi(string nameMentioned)
171     {
172         DisableSpeech();
173
174         this.RhSynt.RaspiCalled(nameMentioned);
175
176         EnableSpeech();
177     }
178
179     /// <summary>
180     /// Used to send the command after raspi called
181     /// </summary>
182     /// <param name="brutCommand"></param>
183     public void SendBrutCommand(string brutCommand)
184     {
185         DisableSpeech();
186
187         this.ComWithServer.SendCommandToServer(brutCommand); ↗
188     }
189
190     /// <summary>
191     /// Reply message to the sender (reply message error if command ↗
192     /// unknow)
193     /// </summary>
194     /// <param name="messageReply"></param>
195     /// <param name="messageCommand"></param>
196     public void ReplyForSynthetize(string messageReply, string ↗
197     messageCommand)
198     {
199         if (messageReply == "ERROR_MESSAGE")
200         {
201             this.RhSynt.WrongCommand(); ↗
202         }
203         else
204         {
205             this.RhSynt.RaspiSayInformation
206                 (this.RhSynt.SetProprelyInformations(messageReply, ↗
207                 messageCommand));
208         }
209         this.IsRaspiCalled = false; ↗
210
211         EnableSpeech();
212     }
213 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Globalization;
4  using System.Linq;
5  using System.Text;
6  using Windows.Media.SpeechSynthesis;
7  using System.Threading;
8  using System.Threading.Tasks;
9  using Windows.UI.Xaml.Controls;
10 using Windows.UI.Core;
11
12 namespace RaspiHomeSpeechNSynthetize
13 {
14     public class Synthetizer
15     {
16         #region Fields
17         #region Constants
18         private const string RASPI_NAME = "raspi";
19         private const char SEPARATOR = ' ';
20         // Change value when new update ("en" to "fr")
21         private const string LANGUAGE_SELECTION = "en";
22         private const int TIME_TO_WAIT = 3;
23         #endregion
24
25         #region Variable
26         private Speecher _rhSpeech;
27
28         private Commands _rhCommands;
29
30         private Random _rnd;
31         private SpeechSynthesizer _voice;
32         private string _commandReceiveStr = "";
33         private string _commandToSend = "";
34
35         private bool _isCalled = false;
36         private bool _isCompleted = false;
37         private List<string> _lstSentenceSplited;
38         #endregion
39         #endregion
40
41         #region Properties
42         public Speecher RhSpeech
43         {
44             get
45             {
46                 return _rhSpeech;
47             }
48
49             set
50             {
51                 _rhSpeech = value;
52             }
53         }
54
55         public string CommandReceiveStr
56         {
```

```
57         get
58     {
59         return _commandReceiveStr;
60     }
61
62     set
63     {
64         _commandReceiveStr = value;
65     }
66 }
67
68     public bool IsCalled
69     {
70         get
71     {
72         return _isCalled;
73     }
74
75         set
76     {
77         _isCalled = value;
78     }
79 }
80
81     public bool IsCompleted
82     {
83         get
84     {
85         return _isCompleted;
86     }
87
88         set
89     {
90         _isCompleted = value;
91     }
92 }
93
94     public Commands RhCommands
95     {
96         get
97     {
98         return _rhCommands;
99     }
100
101         set
102     {
103         _rhCommands = value;
104     }
105 }
106 #endregion
107
108 #region Constructor
109 /// <summary>
110 /// Constructor: Initialize
111 /// </summary>
112 public Synthetizer(Speecher paramSpeecher)
```

```
113     {
114         this.RhSpeech = paramSpeecher;
115         this.RhCommands = new Commands();
116         this._rnd = new Random();
117         this._voice = new SpeechSynthesizer();
118         this._lstSentenceSplited = new List<string>();
119     }
120 #endregion
121
122 #region Methods
123 /// <summary>
124 /// Raspberry processus, wait calling to start communication with      ↗
125 /// </summary>
126 private void RaspiProcessus()
127 {
128     if (this.IsCalled)
129     {
130         SendCommand();
131     }
132 }
133
134 /// <summary>
135 /// Send to the synthetize method the order to reply
136 /// </summary>
137 /// <param name="name"></param>
138 public void RaspiCalled(string name)
139 {
140     string raspiName = RemoveDiacritics(name).ToLower();
141
142     if (raspiName == RASPI_NAME.ToLower())
143     {
144         RaspiCalled(this.RhCommands.WhenCalling);
145         this.RhSpeech.IsRaspiCalled = true;
146         this.IsCalled = true;
147     }
148 }
149
150 /// <summary>
151 /// Send the instruction for the Raspberry
152 /// </summary>
153 /// <returns></returns>
154 public string SendCommand()
155 {
156     this.RhSpeech.IsRaspiCalled = false;
157     this.IsCalled = false;
158
159     this.IsCompleted = true;
160     return this._commandToSend;
161 }
162
163 /// <summary>
164 /// Processus to choose the sentence to say
165 /// </summary>
166 /// <param name="repertory"> List of sentence to say </param>
167 private void RaspiCalled(List<string> repertory)
```

```
168     {
169         string messageToSay = repertory[_rnd.Next(0, repertory.Count - 1)];
170         this.RaspiTalk(messageToSay);
171     }
172 }
173
174 /// <summary>
175 /// Allow the raspi to let her talk
176 /// </summary>
177 /// <param name="messageToSay"> sentence to say </param>
178 private async void RaspiTalk(string messageToSay)
179 {
180     // Get the output element (audio jack)
181     MediaElement mediaElement = new MediaElement();
182     SpeechSynthesizer synth = new SpeechSynthesizer();
183
184     // Set the default language
185     foreach (VoiceInformation vInfo in SpeechSynthesizer.AllVoices)
186     {
187         if (vInfo.Language.Contains(LANGUAGE_SELECTION))
188         {
189             synth.Voice = vInfo;
190             break;
191         }
192         else
193             synth.Voice = vInfo;
194     }
195
196     SpeechSynthesisStream synthStream = await
197         synth.SynthesizeTextToStreamAsync(messageToSay);
198
199     mediaElement.SetSource(synthStream, synthStream.ContentType);
200     // 0 = min / 1 = max
201     mediaElement.Volume = 1;
202     mediaElement.Play();
203
204     // Work like Thread.Sleep(TIME_TO_WAIT)
205     await Task.Delay(TimeSpan.FromSeconds(TIME_TO_WAIT));
206 }
207
208 /// <summary>
209 /// Called when there is any error
210 /// </summary>
211 public void WrongCommand()
212 {
213     // Reach the error request sentences to say
214     this.RaspiCalled(this.RhCommands.SpeicherRespondingRequestError);
215 }
216
217 /// <summary>
218 /// Allow the Raspi, to let her talk with list of information
219 /// </summary>
220 /// <param name="informationsToGive"></param>
221 public void RaspiSayInformation(List<string> informationsToGive)
```



```
    this.RhCommands.SpeecherRespondingEstatRequest[5]);
274        }
275        break;
276    case "PRES":
277        if (pres)
278        {
279            result.Add
280                (this.RhCommands.SpeecherRespondingEstatRequest[6] +
281                 information.Split('=').Last() +
282
283            this.RhCommands.SpeecherRespondingEstatRequest[7]);
284        }
285        break;
286    }
287    else
288        result.Add("");
289    return result;
290}
291
292 /// <summary>
293 /// Stack Overflow solution to delete accents in strings
294 /// http://stackoverflow.com/questions/249087/how-do-i-remove-
295     diacritics-accents-from-a-string-in-net
296 /// </summary>
297 /// <param name="str"></param>
298 /// <returns></returns>
299 static string RemoveDiacritics(string str)
300 {
301     var normalizedString = str.Normalize(NormalizationForm.FormD);
302     var stringBuilder = new StringBuilder();
303
304     foreach (var c in normalizedString)
305     {
306         var unicodeCategory = CharUnicodeInfo.GetUnicodeCategory(c);
307         if (unicodeCategory != UnicodeCategory.NonSpacingMark)
308         {
309             stringBuilder.Append(c);
310         }
311     }
312
313     return stringBuilder.ToString().Normalize
314         (NormalizationForm.FormC);
315 }
316 #endregion
317 }
```

```
1  using RaspiHomeSpeechNSynthetize;
2  using System;
3  using System.Collections.Generic;
4  using System.Diagnostics;
5  using System.Linq;
6  using System.Net;
7  using System.Net.Sockets;
8  using System.Text;
9  using System.Threading;
10 using System.Threading.Tasks;
11 using Windows.ApplicationModel.Core;
12 using Windows.Networking;
13 using Windows.Networking.Connectivity;
14 using Windows.Networking.Sockets;
15 using Windows.Storage.Streams;
16
17 namespace RaspiHomeSpeechNSynthetize
18 {
19     public class CommunicationWithServer
20     {
21         #region Fields
22         #region Constants
23             // Default information to connect on the server
24             private const int PORT = 54565;
25             //// Need to be changed fo each configuration
26             private const string IPSERVER = "10.134.97.117";// "192.168.2.8";
27
28             private const string FORMATSTRING = "IPRasp={0};Location=
29                 {1};Component={2}";
30             private const string COMMUNICATIONSEPARATOR = "@";
31
32             // Important need to be changed if it's another room!
33             private const string LOCATION = "Salon";
34             private const string COMPONENT = "Microphone";
35             private const string RPINAME = "Microphone_" + LOCATION;// 
36                 "192.168.2.8";
37
38             private const int MESSAGE_FULL_LENGTH = 512;
39             #endregion
40
41             #region Variables
42             private Speecher _speecher;
43
44             private StreamSocket _socket = new StreamSocket();
45             private StreamSocketListener _listener = new StreamSocketListener();
46             private List<StreamSocket> _connections = new List<StreamSocket>();
47             private bool _isConnected = false;
48             private bool _connecting = false;
49
50             private string _messageCommand = "";
51             #endregion
52             #endregion
53             public Speecher Speecher
54             {
```

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

```
111         set
112     {
113         _isConnected = value;
114     }
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 #endregion
144
145 #region Constructors
146 /// <summary>
147 /// Constructor: Initializer
148 /// </summary>
149 /// <param name="paramModel"></param>
150 public CommunicationWithServer(Speecher paramModel)
151 {
152     this.Speecher = paramModel;
153
154     Connect();
155 }
156 #endregion
157
158 #region Methods
159 /// <summary>
160 /// Connect the raspberry to the server
161 /// </summary>
162 private async void Connect()
163 {
164     try
165     {
166         this.Connecting = true;
```

```
167         await this.Socket.ConnectAsync(new HostName(IPSERVER),  
168             PORT.ToString());  
169         SendForInitialize();  
170         this.Connecting = false;  
171         this.isConnected = true;  
172         WaitForData(this.Socket);  
173     }  
174     catch (Exception)  
175     {  
176         this.Connecting = false;  
177         this.isConnected = false;  
178     }  
179 }  
180  
181     /// <summary>  
182     /// Listen the traffic on the port  
183     /// </summary>  
184     private async void Listen()  
185     {  
186         this.Listener.ConnectionReceived += listenerConnectionReceived;  
187         await this.Listener.BindServiceNameAsync(PORT.ToString());  
188     }  
189  
190     void listenerConnectionReceived(StreamSocketListener sender,  
191         StreamSocketListenerConnectionReceivedEventArgs args)  
192     {  
193         this.Connections.Add(args.Socket);  
194         WaitForData(args.Socket);  
195     }  
196  
197     /// <summary>  
198     /// Send the message in input to output  
199     /// </summary>  
200     /// <param name="socket"></param>  
201     /// <param name="message"></param>  
202     private async void SendMessage(StreamSocket socket, string message)  
203     {  
204         DataWriter dataWriter = new DataWriter(socket.OutputStream);  
205         var len = dataWriter.MeasureString(message); // Gets the UTF-8  
206             string length.  
207         dataWriter.WriteInt32((int)len);  
208         dataWriter.WriteString(message);  
209         var ret = await dataWriter.StoreAsync();  
210         dataWriter.DetachStream();  
211     }  
212  
213     /// <summary>  
214     /// Send to initialize the raspberry to the server  
215     /// </summary>  
216     private void SendForInitialize()  
217     {  
218         SendMessage(this.Socket, string.Format(COMMUNICATIONSEPARATOR +  
219             RPINAME + COMMUNICATIONSEPARATOR + "Connection:" + FORMATSTRING,  
220             GetHostName(), LOCATION, COMPONENT));  
221     }
```

```
218     }
219
220     /// <summary>
221     /// Send the command to the server
222     /// </summary>
223     public void SendCommandToServer(string message)
224     {
225         SendMessage(this.Socket, COMMUNICATIONSEPARATOR + "Send:" +
226                     message);
227         this.MessageCommand = message;
228     }
229
230     /// <summary>
231     /// Wait data readed if exist
232     /// </summary>
233     /// <param name="socket"></param>
234     private async void WaitForData(StreamSocket socket)
235     {
236         DataReader dataReader = new DataReader(socket.InputStream);
237         dataReader.InputStreamOptions = InputStreamOptions.Partial;
238         var messageLenght = dataReader.UnconsumedBufferLength;
239         uint stringBytes = messageLenght;
240
241         try
242         {
243             // Read modification in the stream
244             stringBytes = await dataReader.LoadAsync(MESSAGE_FULL_LENGTH);
245
246             // read message
247             string messageRead = dataReader.ReadString(stringBytes);
248
249             // Send in return if the value exist
250             if (messageRead != "")
251             {
252                 this.Speecher.ReplyForSynthetize(messageRead,
253                                                 this.MessageCommand);
254             }
255
256             messageRead = "";
257         }
258         catch (Exception ) { }
259
260         WaitForData(socket);
261     }
262
263     /// <summary>
264     /// Get the ip of the raspberry
265     /// </summary>
266     /// <returns>return a string like 192.168.1.2</returns>
267     public string GetHostName()
268     {
269         List<string> IpAddress = new List<string>();
270         var Hosts =
271             Windows.Networking.Connectivity.NetworkInformation.GetHostNames
272             ().ToList();
273         foreach (var Host in Hosts)
```

```
270         {
271             string IP = Host.DisplayName;
272             IpAddress.Add(IP);
273         }
274         return IpAddress.Last();
275     }
276     #endregion
277 }
278 }
279 }
```

```
1 <Page
2     x:Class="RaspiHomeSpeechNSynthetize.MainPage"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeSpeechNSynthetize"
6     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8     mc:Ignorable="d">
9
10    <StackPanel Orientation="Horizontal" Background="White"             ↵
11        HorizontalAlignment="Center" VerticalAlignment="Center">
12            <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center"   ↵
13                Margin="0,0,50,0">
14                <StackPanel Orientation="Horizontal">
15                    <TextBox x:Name="tbxLightCommand1" Width="250" Text="Allume la    ↵
16                        lumière du salon"/>
17                    <Button x:Name="btnLightCommand1" Width="55" Content="Send"      ↵
18                        Margin="40,0,0,0" Click="btnLightCommand1_Click"/>
19                </StackPanel>
20
21                <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
22                    <TextBox x:Name="tbxLightCommand2" Width="250" Text="Éteins la    ↵
23                        lumière du salon"/>
24                    <Button x:Name="btnLightCommand2" Width="55" Content="Send"      ↵
25                        Margin="40,0,0,0" Click="btnLightCommand2_Click"/>
26                </StackPanel>
27
28                <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
29                    <TextBox x:Name="tbxState" Width="250" Text="Quel est la    ↵
30                        température du salon"/>
31                    <Button x:Name="btnState" Width="55" Content="Send"          ↵
32                        Margin="40,0,0,0" Click="btnState_Click"/>
33                </StackPanel>
34
35                <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center"   ↵
36                    Margin="0,0,50,0">
37                    <StackPanel Orientation="Horizontal">
38                        <TextBox x:Name="tbxStore1" Width="250" Text="Monte le store du    ↵
39                            salon"/>
40                        <Button x:Name="btnSendCommand" Width="55" Content="Send"      ↵
41                            Margin="40,0,0,0" Click="btnStore1_Click"/>
42                    </StackPanel>
43
44                    <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
45                        <TextBox x:Name="tbxStore2" Width="250" Text="Descends le store    ↵
46                            du salon"/>
47                        <Button x:Name="btnStore2" Width="55" Content="Send"          ↵
48                            Margin="40,0,0,0" Click="btnStore2_Click"/>
49                    </StackPanel>
50
51                    <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
52                        <TextBox x:Name="tbxStore3" Width="250" Text="Stop le store du    ↵
53                            salon"/>
54                        <Button x:Name="btnStore3" Width="55" Content="Send"          ↵
55                            Margin="40,0,0,0" Click="btnStore3_Click"/>
56                    </StackPanel>
57
```

```
42      </StackPanel>
43      </StackPanel>
44    </StackPanel>
45  </Page>
46
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Runtime.InteropServices.WindowsRuntime;
6  using Windows.Foundation;
7  using Windows.Foundation.Collections;
8  using Windows.UI.Xaml;
9  using Windows.UI.Xaml.Controls;
10 using Windows.UI.Xaml.Controls.Primitives;
11 using Windows.UI.Xaml.Data;
12 using Windows.UI.Xaml.Input;
13 using Windows.UI.Xaml.Media;
14 using Windows.UI.Xaml.Navigation;
15
16 // Pour plus d'informations sur le modèle d'élément Page vierge, consultez la ↵
17 // page http://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409
18
19 namespace RaspiHomeSpeechNSynthetize
20 {
21     /// <summary>
22     /// Une page vide peut être utilisée seule ou constituer une page de destination au sein d'un frame.
23     /// </summary>
24     public sealed partial class MainPage : Page
25     {
26         #region Fields
27         private Speecher _speecher;
28         #endregion
29
30         #region Constuctor
31         public MainPage()
32         {
33             this.InitializeComponent();
34
35             this._speecher = new Speecher();
36         }
37         #endregion
38
39         #region Event
40         private void btnLightCommand1_Click(object sender, RoutedEventArgs e)
41         {
42             if (this.tbxLightCommand1.Text != "")
43                 this._speecher.SendBrutCommand(this.tbxLightCommand1.Text);
44             else
45                 this._speecher.SendBrutCommand("Allumer lumiere");
46         }
47
48         private void btnLightCommand2_Click(object sender, RoutedEventArgs e)
49         {
50             if (this.tbxLightCommand2.Text != "")
51                 this._speecher.SendBrutCommand(this.tbxLightCommand2.Text);
52             else
53                 this._speecher.SendBrutCommand("Eteindre lumiere");
54         }
55 }
```

```
55     private void btnState_Click(object sender, RoutedEventArgs e)
56     {
57         if (this.tbxState.Text != "")
58             this._speicher.SendBrutCommand(this.tbxState.Text);
59         else
60             this._speicher.SendBrutCommand("Temperature du salon");
61     }
62
63     private void btnStore1_Click(object sender, RoutedEventArgs e)
64     {
65         if (this.tbxStore1.Text != "")
66             this._speicher.SendBrutCommand(this.tbxStore1.Text);
67         else
68             this._speicher.SendBrutCommand("Monter store");
69     }
70
71     private void btnStore2_Click(object sender, RoutedEventArgs e)
72     {
73         if (this.tbxStore2.Text != "")
74             this._speicher.SendBrutCommand(this.tbxStore2.Text);
75         else
76             this._speicher.SendBrutCommand("Descendre store");
77     }
78
79     private void btnStore3_Click(object sender, RoutedEventArgs e)
80     {
81         if (this.tbxStore3.Text != "")
82             this._speicher.SendBrutCommand(this.tbxStore3.Text);
83         else
84             this._speicher.SendBrutCommand("Stopper store");
85     }
86     #endregion
87 }
88 }
```

```
1  using System.Collections.Generic;
2
3  namespace RaspiHomeSpeechNSynthetize
4  {
5      public class Commands
6      {
7          #region Commands variable
8          private List<string> _whenCalling = new List<string>()
9          {
10              "Oui, que puis-je faire pour vous?", "Comment puis-je vous
11                  servir?", "Je suis toute ouïe"
12          };
13
14          private List<string> _whenCallingError = new List<string>()
15          {
16              "Je ne m'appelle pas comme ça!", "Je crois que vous vous êtes trompé ↵
17                  de personne!"
18          };
19
20          private List<string> _speecherRespondingRequest = new List<string>()
21          {
22              "Tout de suite", "Je m'y mets de ce pas!", "Ne quittez pas!"
23          };
24
25          private List<string> _speecherRespondingRequestError = new
26              List<string>()
27          {
28              "Je n'ai pas compris ce que vous avez demandé"
29          };
30
31          private List<string> _raspiHomeObjectKnown = new List<string>()
32          {
33              "lumiere", "lumieres",
34              "store", "stores",
35              "television", "televisions",
36              "porte", "portes",
37              "fenetre", "fenetres",
38          };
39
40          private List<string> _raspiHomeActionKnown = new List<string>()
41          {
42              "allumer", "allume",
43              "eteindre", "eteins",
44              "monter", "monte",
45              "descendre", "descends",
46              "stopper", "stop",
47              "ouvrir", "ouvre",
48              "fermer", "ferme",
49          };
50
51          private List<string> _speecherRespondingEstatRequest = new List<string>() ↵
52          ()
53          {
54              "Il fait , , degrés Celsius , , degrés Farad", " , degrés Kelvin", "Le ↵
55                  taux d'humidité est de , , pourcent", "La pression est ↵
56                  actuellement de , , hecato Pascal"
57          };
58
59      }
60
61  }
```

```
51         };
52
53         private List<string> _raspiHomeActionWithoutObjectKnown = new
54             List<string>()
55         {
56             "temperature", "humidite", "pression", "etat"
57         };
58
59         private List<string> _raspiHomeLocationKnown = new List<string>()
60             {
61                 "maison", "salon", "cuisine", "parent", "enfant", "bureau", "toilette", "bain"
62             };
63
64         private List<string> _raspiHomeCommandUselessConnecter = new
65             List<string>()
66         {
67             "le", "la", "les", "un", "une", "des", "mon", "ma", "mes", "son", "sa", "ses", "ce", "cet",
68             "cette", "ces", "cel", "celle", "du", "de"
69         };
70
71         private List<string> _raspiHomeGrammarCommand = new List<string>()
72         {
73             "allume la lumière", "allume les lumières", "allumer la
74                 lumière", "allumer les lumières",
75             "éteint la lumière", "éteint les lumières", "éteindre la
76                 lumière", "éteindre les lumières",
77             "monte le store", "monte les stores", "monter le store", "monter les
78                 stores",
79             "descend le store", "descend les stores", "descendre le
80                 store", "descendre les stores",
81             "ouvre le store", "ouvre les stores", "ouvrir le store", "ouvrir
82                 les stores",
83             "ferme le store", "ferme les stores", "fermer le store", "fermer
84                 les stores",
85             "quelle est la température", "quelle est l'humidité", "quelle est
86                 la pression", "quelle est l'état",
87             "de la maison", "de la cuisine", "du salon", "de la salle de
88                 bain", "de la chambre", "de la pièce",
89         };
90     #endregion
91
92     #region Properties
93     public List<string> WhenCalling
94     {
95         get
96         {
97             return _whenCalling;
98         }
99
100        set
101        {
102            _whenCalling = value;
103        }
104    }
```

```
94      public List<string> WhenCallingError
95      {
96          get
97          {
98              return _whenCallingError;
99          }
100         set
101         {
102             _whenCallingError = value;
103         }
104     }
105
106
107
108     public List<string> SpeecherRespondingRequest
109     {
110         get
111         {
112             return _speecherRespondingRequest;
113         }
114         set
115         {
116             _speecherRespondingRequest = value;
117         }
118     }
119 }
120
121     public List<string> SpeecherRespondingRequestError
122     {
123         get
124         {
125             return _speecherRespondingRequestError;
126         }
127         set
128         {
129             _speecherRespondingRequestError = value;
130         }
131     }
132 }
133
134     public List<string> RaspiHomeObjectKnown
135     {
136         get
137         {
138             return _raspiHomeObjectKnown;
139         }
140         set
141         {
142             _raspiHomeObjectKnown = value;
143         }
144     }
145 }
146
147     public List<string> RaspiHomeActionKnown
148     {
149         get
```

```
150         {
151             return _raspiHomeActionKnown;
152         }
153
154         set
155         {
156             _raspiHomeActionKnown = value;
157         }
158     }
159
160     public List<string> SpeecherRespondingEtatRequest
161     {
162         get
163         {
164             return _speecherRespondingEtatRequest;
165         }
166
167         set
168         {
169             _speecherRespondingEtatRequest = value;
170         }
171     }
172
173     public List<string> RaspiHomeActionWithoutObjectKnown
174     {
175         get
176         {
177             return _raspiHomeActionWithoutObjectKnown;
178         }
179
180         set
181         {
182             _raspiHomeActionWithoutObjectKnown = value;
183         }
184     }
185
186     public List<string> RaspiHomeLocationKnown
187     {
188         get
189         {
190             return _raspiHomeLocationKnown;
191         }
192
193         set
194         {
195             _raspiHomeLocationKnown = value;
196         }
197     }
198
199     public List<string> RaspiHomeCommandUselessConnecter
200     {
201         get
202         {
203             return _raspiHomeCommandUselessConnecter;
204         }
205     }
```

```
206         set
207         {
208             _raspiHomeCommandUselessConnector = value;
209         }
210     }
211
212     public List<string> RaspiHomeGrammarCommand
213     {
214         get
215         {
216             return _raspiHomeGrammarCommand;
217         }
218
219         set
220         {
221             _raspiHomeGrammarCommand = value;
222         }
223     }
224 #endregion
225 }
226 }
```

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <grammar version="1.0" xml:lang="fr-FR" mode="voice" root="toplevel"
3 tag-format="semantics/1.0"
4 xmlns="http://www.w3.org/2001/06/grammar"
5 xmlns:sapi="http://schemas.microsoft.com/Speech/2002/06/SRGSExtensions">
6 <rule id="toplevel" scope="public">
7   <one-of>
8     <item>
9       <ruleref uri="#Raspi"/>
10    </item>
11    <item>
12      <ruleref uri="#Meteo"/>
13    </item>
14    <item>
15      <ruleref uri="#Action"/>
16    </item>
17  </one-of>
18 </rule>
19
20 <rule id="Raspi">
21   <example> raspi </example>
22
23   <item> raspi </item>
24 </rule>
25
26
27 <rule id="Meteo">
28   <example> quel est la temperature </example>
29   <example> quel est le taux d'humidite </example>
30   <example> quel est le niveau de pression </example>
31   <example> quel est l'etat</example>
32
33   <one-of>
34     <item> temperature </item>
35     <item> humidite </item>
36     <item> pression </item>
37     <item> etat </item>
38   </one-of>
39   <ruleref uri="#Localisation"/>
40 </rule>
41
42 <rule id="Action">
43   <example> allume la lumiere</example>
44   <example> eteint la lumiere</example>
45   <example> monte le store</example>
46   <example> descend le store</example>
47   <example> ouvre le store</example>
48   <example> ferme le store</example>
49
50   <one-of>
51     <item> allume </item>
52     <item> eteint </item>
53     <item> monte </item>
54     <item> descend </item>
55     <item> ouvre </item>
56     <item> ferme </item>
```

```
57      </one-of>
58
59      <ruleref uri="#Composant"/>
60  </rule>
61
62  <rule id="Composant">
63      <example> la lumiere </example>
64      <example> les lumieres </example>
65      <example> le store </example>
66      <example> les stores </example>
67
68      <one-of>
69          <item> lumiere </item>
70          <item> store </item>
71      </one-of>
72      <ruleref uri="#Localisation"/>
73  </rule>
74
75  <rule id="Localisation">
76      <example> du salon </example>
77      <example> de la cuisine </example>
78      <example> du bureau </example>
79      <example> de la chambre </example>
80      <example> des toilettes </example>
81      <example> de la salle de bain </example>
82      <example> de la maison </example>
83
84      <item repeat="0-1"> salon </item>
85      <item repeat="0-1"> cuisine </item>
86      <item repeat="0-1"> bureau </item>
87      <item repeat="0-1"> chambre </item>
88      <item repeat="0-1"> toilette </item>
89      <item repeat="0-1"> salle de bain </item>
90      <item repeat="0-1"> maison </item>
91  </rule>
92 </grammar>
```

```
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 namespace RaspiHomeTabletWindows.Menu
16 {
17     public class MenuModel : PropertyChangedBase
18     {
19         #region Fields
20         #region Constants
21         #endregion
22
23         #region Variables
24         private MenuView _view = null;
25
26         private CommunicationWithServer _comWithServer = null;
27
28         // Use to store data in the cache
29         private Windows.Storage.ApplicationDataContainer localSettings =
30             Windows.Storage.ApplicationData.Current.LocalSettings;
31         #endregion
32         #endregion
33
34         #region Properties
35         public MenuView View
36         {
37             get
38             {
39                 return _view;
40             }
41
42             set
43             {
44                 _view = value;
45             }
46         }
47
48         public CommunicationWithServer ComWithServer
49         {
50             get
51             {
52                 return _comWithServer;
53             }
54
55             set
56             {
```

```
56             _comWithServer = value;
57         }
58     }
59 #endregion
60
61     #region Constructor
62     /// <summary>
63     /// Constructor: Initializer
64     /// </summary>
65     public MenuModel(MenuView paramView)
66     {
67         // Communication like Model-View
68         this.View = paramView;
69
70         // Initialize communication with server
71         this.ComWithServer = new CommunicationWithServer();
72     }
73 #endregion
74
75     #region Methods
76     #endregion
77 }
78 }
79 }
```

```
1 <Page
2     x:Class="RaspiHomeTabletWindows.Menu.MenuView"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeTabletWindows.Menu"
6     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8     mc:Ignorable="d">
9     <Grid>
10         <Grid.ColumnDefinitions>
11             <ColumnDefinition Width="10"/>
12             <ColumnDefinition/>
13             <ColumnDefinition Width="10"/>
14         </Grid.ColumnDefinitions>
15         <Grid.RowDefinitions>
16             <RowDefinition Height="10"/>
17             <RowDefinition/>
18             <RowDefinition Height="10"/>
19         </Grid.RowDefinitions>
20
21         <Grid Grid.Column="1" Grid.Row="1">
22             <Grid.RowDefinitions>
23                 <RowDefinition Height="*" MinHeight="160"/>
24                 <RowDefinition Height="5*"/>
25             </Grid.RowDefinitions>
26             <!--TOOLBAR-->
27             <Rectangle Fill="WhiteSmoke" Grid.Row="0"/>
28             <Grid>
29                 <Grid.ColumnDefinitions>
30                     <ColumnDefinition/>
31                     <ColumnDefinition Width="200"/>
32                 </Grid.ColumnDefinitions>
33                 <!--BUTTON TOOLBAR-->
34                 <StackPanel x:Name="stkMenuToolbar" Orientation="Horizontal"      ↴
35                     Grid.Column="0"/>
36                 <!--SOME INFORMATION-->
37                 <Grid Grid.Column="1">
38                     <TextBlock Text="RaspiHome" HorizontalAlignment="Left"      ↴
39                         VerticalAlignment="Center" FontSize="30" Margin="10,0,0,0"/>
40                     </TextBlock>
41                 </Grid>
42             </Grid>
43
44             <!--FRAME EVENT WITH TOOLBAR BUTTON-->
45             <Grid x:Name="grdFrame" Grid.Row="1" Background="White">
46                 <Frame x:Name="frmMenu"/>
47             </Grid>
48         </Grid>
49     </Page>
```

```
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 RaspiHomeTabletWindows.Menu.MenuToolbar;
16 using RaspiHomeTabletWindows.Modules.GlobalSetup;
17 using RaspiHomeTabletWindows.Modules.Home;
18 using RaspiHomeTabletWindows.Modules.Information;
19 using RaspiHomeTabletWindows.Modules.Settings;
20 using System;
21 using System.Collections.Generic;
22 using System.Linq;
23 using Windows.UI.Xaml;
24 using Windows.UI.Xaml.Controls;
25
26 // The User Control item template is documented at http://go.microsoft.com/ ↵
27 // fwlink/?LinkId=234236
28
29 namespace RaspiHomeTabletWindows.Menu
30 {
31     public sealed partial class MenuView : Page
32     {
33         #region Fields
34         #region Constants
35         #endregion
36         #region Variables
37         private MenuModel _model = null;
38
39         private ToolbarButtonView _btnToolbarView = null;
40
41         private string _frameAlreadyChoose = "";
42         private Dictionary<string, Dictionary<string, string>> ↵
43             _buttonInformation = new Dictionary<string, Dictionary<string, string>,> ↵
44             string>>() {
45             { "Home", new Dictionary<string, string>() { { "Retourner à" ↵
46                 "l'accueil" , "Home.png" } } },
47             { "Global setup", new Dictionary<string, string>() { { "Visualiser" ↵
48                 "l'ensemble des modules", "GlobalSetup.png" } } },
49             { "Information", new Dictionary<string, string>() { { "Regarder les" ↵
50                 "information du système", "Information.png" } } },
51             { "Setting", new Dictionary<string, string>() { { "Paramétrage de" ↵
52                 "l'application", "Setting.png" } } }
53         };
54
55         private List<string> _listChoise = null;
56     }
57 }
```

```
50
51     private List<ToolbarButtonData> _lstToolbarButtonData = null;
52     private List<ToolbarButtonView> _lstToolbarButton = null;
53     #endregion
54     #endregion
55
56     #region Properties
57     public MenuModel Model
58     {
59         get
60         {
61             return _model;
62         }
63
64         set
65         {
66             _model = value;
67         }
68     }
69
70     public string FrameAlreadyChoose
71     {
72         get
73         {
74             return _frameAlreadyChoose;
75         }
76
77         set
78         {
79             _frameAlreadyChoose = value;
80         }
81     }
82
83     public List<string> LstChoise
84     {
85         get
86         {
87             return _listChoise;
88         }
89
90         set
91         {
92             _listChoise = value;
93         }
94     }
95
96     public List<ToolbarButtonData> LstToolbarButtonData
97     {
98         get
99         {
100             return _lstToolbarButtonData;
101         }
102
103         set
104         {
105             _lstToolbarButtonData = value;
```

```
106         }
107     }
108
109     public List<ToolbarButtonView> LstToolbarButton
110     {
111         get
112         {
113             return _lstToolbarButton;
114         }
115
116         set
117         {
118             _lstToolbarButton = value;
119         }
120     }
121 #endregion
122
123 #region Constructor
124 /// <summary>
125 /// Constructor: Initializer
126 /// </summary>
127 public MenuView()
128 {
129     this.InitializeComponent();
130
131     this.Loaded += UserControl_Loaded;
132
133     this.Model = new MenuModel(this);
134
135     InitializeToolbarButton();
136 }
137 #endregion
138
139 #region Event
140 private void UserControl_Loaded(object sender, RoutedEventArgs e)
141 {
142     UpdateMenuToolbar();
143 }
144
145 private void MenuToolbarButton_Click(object sender, EventArgs e)
146 {
147     foreach (var toolbarButton in this.LstToolbarButton)
148     {
149         toolbarButton.IsSelected = false;
150     }
151
152     switch (((ToolbarButtonView)sender).WhoseButtonClicked)
153     {
154         case "Home":
155             this.frmMenu.Content = null;
156             this.frmMenu.Navigate(typeof(HomeView));
157             break;
158         case "Global setup":
159             this.frmMenu.Content = null;
160             this.frmMenu.Navigate(typeof(GlobalSetupView));
161             break;
162     }
163 }
```

```
162             case "Information":
163                 this.frmMenu.Content = null;
164                 this.frmMenu.Navigate(typeof(InformationView));
165                 break;
166             case "Setting":
167                 this.frmMenu.Content = null;
168                 this.frmMenu.Navigate(typeof(SettingView));
169                 break;
170         }
171 
172         ((ToolbarButtonView)sender).IsSelected = true;
173     }
174 #endregion
175 
176 #region Methods
177     private void InitializeToolBarButton()
178     {
179         this.LstToolBarButtonData = new List<ToolBarButtonData>();
180         this.LstToolBarButton = new List<ToolBarButtonView>();
181         this.LstChoise = new List<string>();
182 
183         foreach (var keyInfo in this._buttonInformation.Keys)
184         {
185             this.LstToolBarButtonData.Add(new ToolBarButtonData(keyInfo,
186                 this._buttonInformation[keyInfo].Keys.FirstOrDefault(),
187                 this._buttonInformation[keyInfo][this._buttonInformation
188                     [keyInfo].Keys.FirstOrDefault()]));
189             this.LstChoise.Add(keyInfo);
190         }
191     }
192 
193     private void UpdateMenuToolbar()
194     {
195         //foreach (ToolBarButtonView t in this.stkMenuToolbar.Children)
196         //    t._click -= MenuToolbarButton_Click;
197         this.stkMenuToolbar.Children.Clear();
198         this.LstToolBarButton.Clear();
199         foreach (ToolBarButtonData t in this.LstToolBarButtonData)
200         {
201             this._btnToolbarView = new ToolbarButtonView(t.FrameChoose,
202                 t.Description, t.IconLink);
203             this._btnToolbarView.Tag = t;
204             this._btnToolbarView._click += MenuToolbarButton_Click;
205             this.stkMenuToolbar.Children.Add(this._btnToolbarView);
206             this.LstToolBarButton.Add(this._btnToolbarView);
207         }
208     }
209 }
210 }
```

```
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 IPSERVER = "10.134.97.117";// "192.168.2.8";
34
35             private const string FORMATSTRING = "IPRasp={0};Location=
36                 {1};Component={2}";
37             private const string COMMUNICATIONSEPARATOR = "@";
38
39             // Important need to be changed if it's another room!
40             private const string LOCATION = "Salon";
41             private const string COMPONENT = "Tablet";
42             private const string RPINAME = "Tablet_" + LOCATION;
43
44             private const int MESSAGE_FULL_LENGTH = 512;
45             #endregion
46
47             #region Variables
48             private StreamSocket _socket = new StreamSocket();
49             private StreamSocketListener _listener = new StreamSocketListener();
50             private List<StreamSocket> _connections = new List<StreamSocket>();
51             private bool _isConnected = false;
52             private bool _connecting = false;
53
54             private Windows.Storage.ApplicationDataContainer localSettings =
55             Windows.Storage.ApplicationData.Current.LocalSettings;
```

```
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
73         set
74         {
75             _socket = value;
76         }
77     }
78
79     public StreamSocketListener Listener
80     {
81         get
82         {
83             return _listener;
84         }
85
86         set
87         {
88             _listener = value;
89         }
90     }
91
92     public List<StreamSocket> Connections
93     {
94         get
95         {
96             return _connections;
97         }
98
99         set
100        {
101            _connections = value;
102        }
103    }
104
105    public bool IsConnected
106    {
107        get
108        {
109            return _isConnected;
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
210 /// <summary>
211 /// Listen the traffic on the port
212 /// </summary>
213 private async void Listen()
214 {
215     this.Listener.ConnectionReceived += listenerConnectionReceived;
216     await this.Listener.BindServiceNameAsync(PORT.ToString());
217 }
218
219 void listenerConnectionReceived(StreamSocketListener sender,
220                               StreamSocketListenerConnectionReceivedEventArgs args)
221 {
222     this.Connections.Add(args.Socket);
```

```
221             WaitForData(args.Socket);
222         }
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);
```

```
273             // read message
274             string messageRead = dataReader.ReadString(stringBytes);
275
276             await Task.Delay(TimeSpan.FromMilliseconds(200));
277             // Store value
278             localSettings.Values["ReceiveMessageFromServer"] =
279                 messageRead;
280         }
281         catch (Exception e)
282         {
283             string output = e.Message;
284
285             if (messageLenght < 1)
286                 return;
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 }
309 #endregion
```

```
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 Windows.UI.Xaml.Media.Imaging;
16
17 namespace RaspiHomeTabletWindows.Menu.MenuToolbar
18 {
19     public class ToolbarButtonModel : PropertyChangedBase
20     {
21         #region Fields
22         #region Constants
23         #endregion
24
25         #region Variables
26         private ToolbarButtonView _view = null;
27
28         private string _description = "";
29         private string _folderIconName = "";
30         private string _iconLink = null;
31         private BitmapImage imgSource = null;
32         #endregion
33         #endregion
34
35         #region Properties
36         public ToolbarButtonView View
37         {
38             get
39             {
40                 return _view;
41             }
42
43             set
44             {
45                 _view = value;
46             }
47         }
48
49         public string Description
50         {
51             get
52             {
53                 return _description;
54             }
55
56             set
```

```
57             {
58                 _description = value;
59             }
60         }
61
62     public string FolderIconName
63     {
64         get
65         {
66             return _folderIconName;
67         }
68
69         set
70         {
71             _folderIconName = value;
72         }
73     }
74
75     public string IconLink
76     {
77         get
78         {
79             return _iconLink;
80         }
81
82         set
83         {
84             _iconLink = value;
85         }
86     }
87
88     public string IconPath { get; set; }
89
90     public BitmapImage ImgSource
91     {
92         get
93         {
94             return imgSource;
95         }
96
97         set
98         {
99             imgSource = value;
100        }
101    }
102 #endregion
103
104 #region Constructor
105 /// <summary>
106 /// Constructor: Initializer
107 /// </summary>
108 public ToolbarButtonModel(ToolbarButtonView paramView)
109 {
110     this.View = paramView;
111 }
112 #endregion
```

```
113
114      #region Events
115      #endregion
116
117      #region Methods
118      /// <summary>
119      /// Set informations for the button
120      /// </summary>
121      /// <param name="description"></param>
122      /// <param name="iconLink"></param>
123      public void SetInformation(string description, string iconLink)
124      {
125          this.Description = description;
126          this.IconLink = iconLink;
127
128          if (iconLink != "")
129              this.IconPath = "ms-appx:///Icon/" + _iconLink;
130          else
131              this.IconPath = "";
132
133          ChangeIcon();
134      }
135
136      /// <summary>
137      /// Change the icon of the button
138      /// </summary>
139      private void ChangeIcon()
140      {
141          this.View.ChangeIcon(this.IconPath);
142      }
143      #endregion
144  }
145 }
146 }
```

```
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 namespace RaspiHomeTabletWindows.Menu.MenuToolbar
16 {
17     public class ToolbarButtonData
18     {
19         #region Variables
20         private string _frameChoose;
21         private string _description;
22         private string _iconLink;
23         private bool _isSelected = false;
24         #endregion
25
26         #region Properties
27         public string FrameChoose
28         {
29             get
30             {
31                 return _frameChoose;
32             }
33
34             set
35             {
36                 _frameChoose = value;
37             }
38         }
39
40         public string Description
41         {
42             get
43             {
44                 return _description;
45             }
46
47             set
48             {
49                 _description = value;
50             }
51         }
52
53         public string IconLink
54         {
55             get
56             {
```

```
57             return _iconLink;
58         }
59
60         set
61     {
62         _iconLink = value;
63     }
64 }
65
66 public bool IsSelected
67 {
68     get
69     {
70         return _isSelected;
71     }
72
73     set
74     {
75         _isSelected = value;
76         this.isSelected = value;
77     }
78 }
79 #endregion
80
81 #region Constructor
82 /// <summary>
83 /// Constructor: Initializer
84 /// </summary>
85 public ToolbarButtonData(string frameChoose, string description, string iconLink)
86 {
87     this.FrameChoose = frameChoose;
88     this.Description = description;
89     this.IconLink = iconLink;
90 }
91 #endregion
92 }
93 }
94 }
```

```
1 <UserControl
2     x:Class="RaspiHomeTabletWindows.Menu.MenuToolbar.ToolbarButtonView"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeTabletWindows.Menu.MenuToolbar"
6     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8     mc:Ignorable="d"
9     Height="150" Width="180">
10    <UserControl.Resources>
11
12    </UserControl.Resources>
13
14    <Button x:Name="btnToolbar" Height="150" Margin="10,0,20,0"          ↗
15        Style="{StaticResource styleRoundButton}" Background="Gray"      ↗
16        ToolTipService.ToolTip="{Binding Path=ToolTipMessage}" Width="{Binding   ↗
17            Height, ElementName=btnToolbar}" Click="btnToolbar_Click">
18        <Image x:Name="imgMenuToolbarButton" Height="100" Width="100"/>
19    </Button>
20 </UserControl>
21
```

```
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 Windows.UI;
17 using Windows.UI.Xaml;
18 using Windows.UI.Xaml.Controls;
19 using Windows.UI.Xaml.Media;
20 using Windows.UI.Xaml.Media.Imaging;
21
22 // The User Control item template is documented at http://go.microsoft.com/ →
23 // fwlink/?LinkId=234236
24
25 namespace RaspiHomeTabletWindows.Menu.MenuToolbar
26 {
27     public sealed partial class ToolbarButtonView : UserControl
28     {
29         #region Fields
30         #region Constants
31         #endregion
32         #region Variables
33         private ToolbarButtonModel _model = null;
34
35         public event EventHandler _click;
36
37         private bool _isSelected = false;
38         private bool _isPressed = false;
39
40         private string _whoseButtonClicked = "";
41         #endregion
42         #endregion
43
44         #region Properties
45         public ToolbarButtonModel Model
46         {
47             get
48             {
49                 return _model;
50             }
51
52             set
53             {
54                 _model = value;
55             }
56         }
57     }
58 }
```

```
56     }
57
58     public bool IsSelected
59     {
60         get
61         {
62             return _isSelected;
63         }
64
65         set
66         {
67             _isSelected = value;
68
69             btnToolbar.IsTabStop = _isSelected;
70
71             if (value)
72             {
73                 btnToolbar.BorderThickness = new Thickness(6.5);
74                 btnToolbar.BorderBrush = new SolidColorBrush
75                     (Color.FromArgb(255, 73, 130, 5));
76             }
77             else
78             {
79                 btnToolbar.BorderThickness = new Thickness(5);
80                 btnToolbar.BorderBrush = new SolidColorBrush
81                     (Color.FromArgb(255, 76, 74, 75));
82             }
83         }
84     }
85
86     public bool IsPressed
87     {
88         get
89         {
90             return _isPressed;
91         }
92
93         set
94         {
95             _isPressed = value;
96         }
97     }
98
99     public string WhoseButtonClicked
100    {
101        get
102        {
103            return _whoseButtonClicked;
104        }
105
106        set
107        {
108            _whoseButtonClicked = value;
109        }
109 #endregion
```

```
110
111     #region Constructor
112     /// <summary>
113     /// Constructor: Initializer
114     /// </summary>
115     public ToolbarButtonView(string frameChoose, string description,
116                             string iconLink)      ↵
116     {
117         InitializeComponent();
118
119         this.Model = new ToolbarButtonModel(this);
120
121         this.WhoseButtonClicked = frameChoose;
122
123         SetInformation(description, iconLink);
124     }
125 #endregion
126
127 #region Events
128     /// <summary>
129     /// Event on the button
130     /// </summary>
131     /// <param name="sender"></param>
132     /// <param name="e"></param>
133     private void btnToolbar_Click(object sender, RoutedEventArgs e)
134     {
135         if (this._click != null)
136             this._click(this, null);
137         this.IsPressed = true;
138     }
139 #endregion
140
141 #region Methods
142     /// <summary>
143     /// Set information of the button on the toolbar
144     /// </summary>
145     /// <param name="description"></param>
146     /// <param name="iconLink"></param>
147     private void SetInformation(string description, string iconLink)
148     {
149         this.Model.SetInformation(description, iconLink);
150     }
151
152     /// <summary>
153     /// Change the Icon of the button on the toolbar
154     /// </summary>
155     /// <param name="iconPath"></param>
156     public void ChangeIcon(string iconPath)
157     {
158         if (iconPath != "")
159             this.imgMenuToolbarButton.Source = new BitmapImage(new Uri      ↵
160                                         (iconPath));
161     }
162 #endregion
163 }
```

```
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 namespace RaspiHomeTabletWindows.Modules.Home
16 {
17     public class HomeModel
18     {
19         #region Fields
20         #region Constants
21         #endregion
22
23         #region Varaibles
24         private HomeView _view = null;
25         #endregion
26         #endregion
27
28         #region Properties
29         public HomeView View
30         {
31             get
32             {
33                 return _view;
34             }
35
36             set
37             {
38                 _view = value;
39             }
40         }
41         #endregion
42
43         #region Constructor
44         /// <summary>
45         /// Constructor: Initializer
46         /// </summary>
47         public HomeModel(HomeView paramView)
48         {
49             this.View = paramView;
50         }
51         #endregion
52
53         #region Methods
54         #endregion
55     }
56 }
```

```
1 <Page
2     x:Class="RaspiHomeTabletWindows.Modules.Home.HomeView"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeTabletWindows.Modules.Home"
6     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8     mc:Ignorable="d">
9
10    <Grid>
11        <Grid.RowDefinitions>
12            <RowDefinition Height="50"/>
13            <RowDefinition/>
14        </Grid.RowDefinitions>
15        <StackPanel x:Name="stkLocationButton" Orientation="Horizontal"      ↴
16            Grid.Row="0"/>
17            <Frame x:Name="frmHome" Grid.Row="1"/>
18    </Grid>
19 </Page>
```

```
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 RaspiHomeTabletWindows.Menu.LocationButton;
16 using System;
17 using System.Collections.Generic;
18 using Windows.UI.Xaml;
19 using Windows.UI.Xaml.Controls;
20
21 // The User Control item template is documented at http://go.microsoft.com/ ↵
22 // fwlink/?LinkId=234236
23
24 namespace RaspiHomeTabletWindows.Modules.Home
25 {
26     public sealed partial class HomeView : Page
27     {
28         #region Fields
29         #region Constants
30         #endregion
31
32         #region Varaibles
33         private HomeModel _model = null;
34
35         private LocationButtonView _btnLocationButtonView = null;
36
37         private string _frameAlreadyChoose = "";
38         private List<string> _buttonInformation = new List<string>() {
39             "Maison", "Salon", "Cuisine",
40         };
41
42         private List<string> _listChoise = null;
43
44         private List<LocationButtonData> _lstLocationButtonData = null;
45         private List<LocationButtonView> _lstLocationButton = null;
46
47         private Windows.Storage.ApplicationDataContainer localSettings = ↵
48             Windows.Storage.ApplicationData.Current.LocalSettings;
49
50         #endregion
51
52         #region Properties
53         public HomeModel Model
54         {
```

```
55         get
56     {
57         return _model;
58     }
59
60     set
61     {
62         _model = value;
63     }
64 }
65
66     public string FrameAlreadyChoose
67     {
68         get
69     {
70         return _frameAlreadyChoose;
71     }
72
73         set
74     {
75         _frameAlreadyChoose = value;
76     }
77 }
78
79     public List<string> LstChoise
80     {
81         get
82     {
83         return _listChoise;
84     }
85
86         set
87     {
88         _listChoise = value;
89     }
90 }
91
92     public List<LocationButtonData> LstToolbarButtonData
93     {
94         get
95     {
96         return _lstLocationButtonData;
97     }
98
99         set
100    {
101        _lstLocationButtonData = value;
102    }
103 }
104
105    public List<LocationButtonView> LstToolbarButton
106    {
107        get
108    {
109        return _lstLocationButton;
110    }
111 }
```

```
111         set
112     {
113         _lstLocationButton = value;
114     }
115 }
116 }
117 #endregion
118
119 #region Constructor
120 /// <summary>
121 /// Constructor: Initializer
122 /// </summary>
123 public HomeView()
124 {
125     this.InitializeComponent();
126
127     this.Loaded += HomeView_Loaded;
128
129     this.Model = new HomeModel(this);
130
131     InitializeLocationButton();
132 }
133 #endregion
134
135 #region Events
136 private void HomeView_Loaded(object sender, RoutedEventArgs e)
137 {
138     UpdateMenuToolbar();
139 }
140
141 /// <summary>
142 /// Check the button clicked
143 /// </summary>
144 /// <param name="sender"></param>
145 /// <param name="e"></param>
146 private void _btnToolbarView_click(object sender, EventArgs e)
147 {
148     foreach (var locationButton in this.LstToolbarButton)
149     {
150         locationButton.IsSelected = false;
151     }
152
153     string buttonClicked = ((LocationButtonView)
154         sender).WhoseButtonClicked;
155
156     localSettings.Values["NameButtonClicked"] = buttonClicked;
157
158     var actualFrameChoose = localSettings.Values["NameButtonClicked"];
159
160     switch (buttonClicked)
161     {
162         case "Maison":
163             if ((actualFrameChoose.ToString() != "Maison") ||
164                 (actualFrameChoose == null))
165                 localSettings.Values.Remove("NameButtonClicked");
166     }
167 }
```

```
165             localSettings.Values["NameButtonClicked"] = buttonClicked;
166             this.frmHome.Content = null;
167             this.frmHome.Navigate(typeof(Location.House.RoomView));
168             break;
169         case "Salon":
170             if ((actualFrameChoose.ToString() != "Salon") ||
171                 (actualFrameChoose == null))
172                 localSettings.Values.Remove("NameButtonClicked");
173
174             localSettings.Values["NameButtonClicked"] = buttonClicked;
175             this.frmHome.Content = null;
176             this.frmHome.Navigate(typeof(
177                 (Location.OtherRoom.RoomView)));
178             break;
179         case "Cuisine":
180             if ((actualFrameChoose.ToString() != "Cuisine") ||
181                 (actualFrameChoose == null))
182                 localSettings.Values.Remove("NameButtonClicked");
183
184             localSettings.Values["NameButtonClicked"] = buttonClicked;
185             this.frmHome.Content = null;
186             this.frmHome.Navigate(typeof(
187                 (Location.OtherRoom.RoomView)));
188             break;
189         case "Bureau":
190             if ((actualFrameChoose.ToString() != "Bureau") ||
191                 (actualFrameChoose == null))
192                 localSettings.Values.Remove("NameButtonClicked");
193
194             ((LocationButtonView)sender).IsSelected = true;
195         }
196     #endregion
197
198     #region Methods
199     /// <summary>
200     /// Initialize the button on the toolbar
201     /// </summary>
202     private void InitializeLocationButton()
203     {
204         this.LstToolbarButtonData = new List<LocationButtonData>();
205         this.LstToolbarButton = new List<LocationButtonView>();
206         this.LstChoise = new List<string>();
207
208         foreach (var buttonName in this._buttonInformation)
209         {
210             this.LstToolbarButtonData.Add(new LocationButtonData
211                 (buttonName, buttonName));
212             this.LstChoise.Add(buttonName);
213         }
```

```
214     }
215
216     /// <summary>
217     /// Update the toolbar item
218     /// </summary>
219     private void UpdateMenuToolbar()
220     {
221         this.stkLocationButton.Children.Clear();
222         this.LstToolbarButton.Clear();
223         foreach (LocationButtonData t in this.LstToolbarButtonData)
224         {
225             this._btnLocationButtonView = new LocationButtonView
226                 (t.FrameChoose, t.Description);
227             this._btnLocationButtonView.Tag = t;
228             this._btnLocationButtonView._click += _btnToolbarView__click;
229             this.stkLocationButton.Children.Add
230                 (this._btnLocationButtonView);
231             this.LstToolbarButton.Add(this._btnLocationButtonView);
232         }
233     }
234 }
235 }
```

```
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 namespace RaspiHomeTabletWindows.Menu.LocationButton
16 {
17     public class LocationButtonModel : PropertyChangedBase
18     {
19         #region Fields
20         #region Constants
21         #endregion
22
23         #region Variables
24         private LocationButtonView _view = null;
25
26         private string _description = "";
27         private string _folderProjectName = "";
28         #endregion
29         #endregion
30
31         #region Properties
32
33         public LocationButtonView View
34         {
35             get
36             {
37                 return _view;
38             }
39
40             set
41             {
42                 _view = value;
43             }
44         }
45
46         public string FolderProjectName
47         {
48             get
49             {
50                 return _folderProjectName;
51             }
52
53             set
54             {
55                 _folderProjectName = value;
56             }
57         }
58     }
59 }
```

```
57      }
58
59      public string LocationName
60      {
61          get
62          {
63              return _description;
64          }
65
66          set
67          {
68              _description = value;
69              OnPropertyChanged("LocationName");
70          }
71      }
72 #endregion
73
74 #region Constructor
75 /// <summary>
76 /// Constructor: Initializer
77 /// </summary>
78 public LocationButtonModel(LocationButtonView paramView)
79 {
80     this.View = paramView;
81 }
82 #endregion
83
84 #region Methods
85 /// <summary>
86 /// Set information name
87 /// </summary>
88 /// <param name="description"></param>
89 public void SetInformation(string description)
90 {
91     this.LocationName = description;
92 }
93 #endregion
94 }
95 }
96 }
```

```
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 namespace RaspiHomeTabletWindows.Menu.LocationButton
16 {
17     public class LocationButtonData
18     {
19         #region Fields
20         #region Constants
21         #endregion
22
23         #region Variables
24         private string _frameChoose;
25         private string _description;
26         private bool _isSelected = false;
27         #endregion
28         #endregion
29
30         #region Properties
31
32         public string FrameChoose
33         {
34             get
35             {
36                 return _frameChoose;
37             }
38
39             set
40             {
41                 _frameChoose = value;
42             }
43         }
44
45         public string Description
46         {
47             get
48             {
49                 return _description;
50             }
51
52             set
53             {
54                 _description = value;
55             }
56     }
```

```
57     public bool IsSelected
58     {
59         get
60         {
61             return _isSelected;
62         }
63     }
64     set
65     {
66         _isSelected = value;
67         this.IsSelected = value;
68     }
69 }
70 #endregion
71
72 #region Constructor
73 /// <summary>
74 /// Constructor: Initializer
75 /// </summary>
76 public LocationButtonData(string frameChoose, string description)
77 {
78     this.FrameChoose = frameChoose;
79     this.Description = description;
80 }
81 #endregion
82
83 #region Methods
84 #endregion
85
86     }
87 }
88 }
```

```
1 <UserControl
2     x:Class="RaspiHomeTabletWindows.Menu.LocationButton.LocationButtonView"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeTabletWindows.Menu.LocationButton"
6     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8     mc:Ignorable="d"
9     d:DesignHeight="50" d:DesignWidth="150">
10
11     <Button x:Name="btnButtonLocation" Height="50" Width="120"
12         Margin="10,0,20,0" Background="{StaticResource DefaultBackgroundColor}" Click="btnButtonLocation_Click">
13         <TextBlock x:Name="tblLocationName" Text="{Binding Path=LocationName}" Foreground="White" FontSize="28"/>
14     </Button>
15 </UserControl>
```

```
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 Windows.UI;
17 using Windows.UI.Xaml;
18 using Windows.UI.Xaml.Controls;
19 using Windows.UI.Xaml.Media;
20
21 // The User Control item template is documented at http://go.microsoft.com/ ↵
22 // fwlink/?LinkId=234236
23
24 namespace RaspiHomeTabletWindows.Menu.LocationButton
25 {
26     public sealed partial class LocationButtonView : UserControl
27     {
28         #region Fields
29         #region Constants
30         #endregion
31         #region Variables
32         private LocationButtonModel _model = null;
33
34         public event EventHandler _click;
35
36         private bool _isSelected = false;
37         private bool _isPressed = false;
38
39         private string _whoseButtonClicked = "";
40         #endregion
41         #endregion
42
43         #region Properties
44
45         public LocationButtonModel Model
46         {
47             get
48             {
49                 return _model;
50             }
51
52             set
53             {
54                 _model = value;
55             }
56         }
57     }
58 }
```

```
56     }
57
58     public bool IsSelected
59     {
60         get
61         {
62             return _isSelected;
63         }
64
65         set
66         {
67             _isSelected = value;
68
69             btnButtonLocation.IsTabStop = _isSelected;
70
71             if (value)
72             {
73                 btnButtonLocation.BorderThickness = new Thickness(3);
74                 btnButtonLocation.BorderBrush = new SolidColorBrush
75                     (Color.FromArgb(255, 73, 130, 5));
76                 tblLocationName.Foreground = new SolidColorBrush
77                     (Color.FromArgb(255, 73, 130, 5));
78             }
79             else
80             {
81                 btnButtonLocation.BorderThickness = new Thickness(1);
82                 btnButtonLocation.BorderBrush = new SolidColorBrush
83                     (Color.FromArgb(255, 76, 74, 75));
84                 tblLocationName.Foreground = new SolidColorBrush
85                     (Color.FromArgb(255, 255, 255, 255));
86             }
87         }
88
89         public bool IsPressed
90         {
91             get
92             {
93                 return _isPressed;
94             }
95             set
96             {
97                 _isPressed = value;
98             }
99
100        public string WhoseButtonClicked
101        {
102            get
103            {
104                return _whoseButtonClicked;
105            }
106            set
107            {
```

```
108             _whoseButtonClicked = value;
109         }
110     }
111 #endregion
112
113 #region Constructor
114 /// <summary>
115 /// Constructor: Initializer
116 /// </summary>
117 public LocationButtonView(string frameChoose, string description)
118 {
119     this.InitializeComponent();
120
121     this.Model = new LocationButtonModel(this);
122
123     this.WhoseButtonClicked = frameChoose;
124
125     SetInformation(description);
126
127     this.tblLocationName.Text = description;
128 }
129 #endregion
130
131 #region Events
132 /// <summary>
133 /// Event on the button
134 /// </summary>
135 /// <param name="sender"></param>
136 /// <param name="e"></param>
137 private void btnButtonLocation_Click(object sender, RoutedEventArgs e)
138 {
139     if (this._click != null)
140         this._click(this, null);
141     this.IsPressed = true;
142 }
143 #endregion
144
145 #region Methods
146 /// <summary>
147 /// Set the Text of the button
148 /// </summary>
149 /// <param name="description"></param>
150 private void SetInformation(string description)
151 {
152     this.Model.SetInformation(description);
153 }
154 #endregion
155
156 }
157 }
158 }
```

```
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 namespace RaspiHomeTabletWindows.Modules.Home.Location.House
16 {
17     public class RoomModel
18     {
19         #region Fields
20         #region Constants
21         #endregion
22
23         #region Varaibles
24         private RoomView _view = null;
25
26         Windows.Storage.ApplicationDataContainer localSettings =
27             Windows.Storage.ApplicationData.Current.LocalSettings;
28         #endregion
29         #endregion
30
31         #region Properties
32         public RoomView View
33         {
34             get
35             {
36                 return _view;
37             }
38
39             set
40             {
41                 _view = value;
42             }
43         }
44         #endregion
45
46         #region Constructor
47         /// <summary>
48         /// Constructor: Initializer
49         /// </summary>
50         public RoomModel(RoomView paramView)
51         {
52             this.View = paramView;
53         }
54         #endregion
55         #region Methods
```

```
56     /// <summary>
57     /// Save message to be send
58     /// </summary>
59     /// <param name="action"></param>
60     /// <param name="component"></param>
61     public void SendMessage(string action, string component)
62     {
63         localSettings.Values["SendMessageToServer"] = action + " " +
64             component;
65     }
66 }
67 }
68 }
```

```
1 <Page
2     x:Class="RaspiHomeTabletWindows.Modules.Home.Location.House.RoomView"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeTabletWindows.Modules.Home.Location.House"
6     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8     mc:Ignorable="d">
9
10    <RelativePanel x:Name="rpnldisplay" Margin="10">
11        <!--LIGHT BUTTON-->
12        <Rectangle x:Name="rect1" Width="17"/>
13        <TextBlock x:Name="tblLight" Text="Lumière" RelativePanel.RightOf="rect1" RelativePanel.AlignTopWithPanel="True" FontSize="26"/>
14        <Button x:Name="btnLightOnOff" RelativePanel.Below="tblLight" Height="120" Width="120" Background="{StaticResource DefaultBackgroundColor}" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Click="btnLightOnOff_Click">
15            <Image x:Name="imgLightButton" Source="ms-appx:///Icon/bulb.png"/>
16        </Button>
17
18        <!--STORE BUTTON-->
19        <!--UP-->
20        <Rectangle x:Name="rect2" Width="20" RelativePanel.AlignRightWithPanel="True"/>
21        <TextBlock x:Name="tblUp" Text="Monter" RelativePanel.LeftOf="rect2" RelativePanel.AlignTopWithPanel="True" FontSize="26"/>
22        <Button x:Name="btnStoreUp" RelativePanel.AlignRightWithPanel="True" RelativePanel.Below="tblUp" Height="120" Width="120" Background="{StaticResource DefaultBackgroundColor}" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Click="btnStoreUp_Click">
23            <Image Source="ms-appx:///Icon/arrowUp.png"/>
24        </Button>
25
26        <Rectangle x:Name="rect3" Height="20" Width="120" RelativePanel.AlignRightWithPanel="True" RelativePanel.Below="btnStoreUp"/>
27
28        <!--DOWN-->
29        <Button x:Name="btnStoreDown" RelativePanel.AlignRightWithPanel="True" RelativePanel.Below="rect3" Height="120" Width="120" Background="{StaticResource DefaultBackgroundColor}" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Click="btnStoreDown_Click">
30            <Image Source="ms-appx:///Icon/arrowDown.png"/>
31        </Button>
32        <TextBlock x:Name="tblDown" Text="Descendre" RelativePanel.AlignHorizontalCenterWith="tblUp" RelativePanel.Below="btnStoreDown" FontSize="26"/>
33
34        <!--OPEN-->
35        <TextBlock x:Name="tblOpen" Text="Ouvrir" RelativePanel.AlignHorizontalCenterWith="tblUp" RelativePanel.Above="btnStoreOpen" FontSize="26"/>
```

```
36      <Button x:Name="btnStoreOpen" RelativePanel.AlignRightWithPanel="True" ↵
37          RelativePanel.Above="rect4" Height="120" Width="120" ↵
38          Background="{StaticResource DefaultBackgroundColor}" ↵
39          HorizontalContentAlignment="Center" VerticalContentAlignment="Center" ↵
40          Click="btnStoreOpen_Click">
41          <Image Source="ms-appx:///Icon/arrowUp.png"/>
42      </Button>
43
44      <Rectangle x:Name="rect4" Height="20" Width="150" ↵
45          RelativePanel.AlignRightWithPanel="True" ↵
46          RelativePanel.Above="btnStoreClose"/>
47
48      <!--CLOSE-->
49      <Button x:Name="btnStoreClose" RelativePanel.AlignRightWithPanel="True" ↵
50          RelativePanel.Above="tblClose" Height="120" Width="120" ↵
51          Background="{StaticResource DefaultBackgroundColor}" ↵
52          HorizontalContentAlignment="Center" VerticalContentAlignment="Center" ↵
53          Click="btnStoreClose_Click">
54          <Image Source="ms-appx:///Icon/arrowDown.png"/>
55      </Button>
56      <TextBlock x:Name="tblClose" Text="Fermer" ↵
57          RelativePanel.AlignHorizontalCenterWith="tblUp" ↵
58          RelativePanel.AlignBottomWithPanel="True" FontSize="26"/>
59
60  </RelativePanel>
61 </Page>
62
```

```
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 Windows.UI.Xaml;
17 using Windows.UI.Xaml.Controls;
18 using Windows.UI.Xaml.Media.Imaging;
19
20 // The User Control item template is documented at http://go.microsoft.com/ →
21 //fwlink/?LinkId=234236
22
23 namespace RaspiHomeTabletWindows.Modules.Home.Location.House
24 {
25     public sealed partial class RoomView : Page
26     {
27         #region Fields
28         #region Constants
29         #endregion
30
31         #region Varaibles
32         private RoomModel _model = null;
33
34         private bool _isOn = false;
35         private bool _isUp = false;
36         private bool _isDown = false;
37         private bool _isOpen = false;
38         private bool _isClose = false;
39         #endregion
40         #endregion
41
42         #region Properties
43         public RoomModel Model
44         {
45             get
46             {
47                 return _model;
48             }
49             set
50             {
51                 _model = value;
52             }
53         }
54
55         public bool IsOn
```

```
56         {
57             get
58             {
59                 return _isOn;
60             }
61         }
62         set
63         {
64             _isOn = value;
65
66             if (value)
67             {
68                 // Send message to save
69                 this.Model.SendMessage("allumer", "lumiere");
70                 //Change the picture
71                 this.imgLightButton.Source = new BitmapImage(new Uri("ms- ↵
72 appx:///Icon/bulbLighting.png"));
73             }
74             else
75             {
76                 // Send message to save
77                 this.Model.SendMessage("eteindre", "lumiere");
78                 // Change the picture
79                 this.imgLightButton.Source = new BitmapImage(new Uri("ms- ↵
80 appx:///Icon/bulb.png"));
81             }
82         }
83     public bool IsUp
84     {
85         get
86         {
87             return _isUp;
88         }
89
90         set
91         {
92             _isUp = value;
93         }
94     }
95
96     public bool IsDown
97     {
98         get
99         {
100             return _isDown;
101         }
102
103         set
104         {
105             _isDown = value;
106         }
107     }
108
109    public bool IsOpen
```

```
110         {
111             get
112             {
113                 return _isOpen;
114             }
115
116             set
117             {
118                 _isOpen = value;
119             }
120         }
121
122     public bool IsClose
123     {
124         get
125         {
126             return _isClose;
127         }
128
129         set
130         {
131             _isClose = value;
132         }
133     }
134 #endregion
135
136 #region Constructor
137 /// <summary>
138 /// Constructor: Initializer
139 /// </summary>
140 public RoomView()
141 {
142     this.InitializeComponent();
143
144     this.Model = new RoomModel(this);
145 }
146 #endregion
147
148 #region Events
149 /// <summary>
150 /// Light control
151 /// </summary>
152 private void btnLightOnOff_Click(object sender, RoutedEventArgs e)
153 {
154     this.IsOn = !this.IsOn;
155 }
156 /**
157
158 /// <summary>
159 /// Store control
160 /// </summary>
161 private void btnStoreUp_Click(object sender, RoutedEventArgs e)
162 {
163     this.IsUp = true;
164     this.Model.SendMessage("monter", "store");
165 }
```

```
166
167      private void btnStoreDown_Click(object sender, RoutedEventArgs e)
168      {
169          this.IsDown = true;
170          this.Model.SendMessage("descendre", "store");
171      }
172
173      private void btnStoreOpen_Click(object sender, RoutedEventArgs e)
174      {
175          this.isOpen = true;
176          this.Model.SendMessage("ouvrir", "store");
177      }
178
179      private void btnStoreClose_Click(object sender, RoutedEventArgs e)
180      {
181          this.isClose = true;
182          this.Model.SendMessage("fermer", "store");
183      }
184      #endregion
185
186      #region Methods
187      #endregion
188  }
189 }
190 }
```

```
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.Linq;
17 using System.Threading.Tasks;
18 using Windows.UI.Xaml;
19
20 namespace RaspiHomeTabletWindows.Modules.Home.Location.OtherRoom
21 {
22     public class RoomModel
23     {
24         #region Fields
25         #region Constants
26         #endregion
27
28         #region Variables
29         private RoomView _view = null;
30
31         private string _messageReaded = "";
32
33         DispatcherTimer _dTimer = null;
34
35         Windows.Storage.ApplicationDataContainer localSettings =
36             Windows.Storage.ApplicationData.Current.LocalSettings;
37         #endregion
38         #endregion
39
40         #region Properties
41         public RoomView View
42         {
43             get
44             {
45                 return _view;
46             }
47
48             set
49             {
50                 _view = value;
51             }
52
53         public string MessageReceive
54         {
55             get
```

```
56         {
57             return _messageReaded;
58         }
59
60         set
61     {
62         _messageReaded = value;
63     }
64 }
65 #endregion
66
67 #region Constructor
68 /// <summary>
69 /// Constructor: Initializer
70 /// </summary>
71 public RoomModel(RoomView paramView)
72 {
73     this.View = paramView;
74
75     this._dTimer = new DispatcherTimer();
76     this._dTimer.Interval = new TimeSpan(200);
77     this._dTimer.Tick += _dTimer_Tick; ;
78
79     this._dTimer.Start();
80
81     this.InitializeState();
82 }
83
84 private void _dTimer_Tick(object sender, object e)
85 {
86     if (localSettings.Values["ReceiveMessageFromServer"] != null) ➔
87     {
88         var messageToSend = localSettings.Values
89             ["ReceiveMessageFromServer"];
90         this.MessageReceive = messageToSend.ToString();
91         UpDateView();
92         localSettings.Values.Remove("ReceiveMessageFromServer");
93
94         this._dTimer.Stop();
95     }
96 }
97 #endregion
98
99 #region Events
100 /// <summary>
101 /// Initialize at the start (check if the sense hat exist)
102 /// </summary>
103 private void InitializeState()
104 {
105     // Update state room
106     var locationName = localSettings.Values["NameButtonClicked"];
107
108     if (locationName != null) ➔
109         localSettings.Values["SendMessageToServer"] = "etat " +
110             locationName;
111     else
```

```
110             localSettings.Values["SendMessageToServer"] = "etat salon";
111         }
112     #endregion
113
114     #region Methods
115     /// <summary>
116     /// Save value to be send
117     /// </summary>
118     /// <param name="action"></param>
119     /// <param name="component"></param>
120     public void SendMessage(string action, string component)
121     {
122         var locationName = localSettings.Values["NameButtonClicked"];
123
124         if (locationName != null)
125             localSettings.Values["SendMessageToServer"] = action + " " + ↵
126                                         component + " " + locationName;
127     }
128
129     /// <summary>
130     /// Update state values on the view
131     /// </summary>
132     private async void UpDateView()
133     {
134         await Task.Delay(TimeSpan.FromMilliseconds(200));
135
136         this.View.EnableDisplayState();
137
138         var informations = this.MessageReceive.Split(';');
139         foreach (var info in informations)
140         {
141             switch (info.Split('=').First())
142             {
143                 case "TEMP":
144                     this.View.StrTemp = info.Split('=').Last();
145                     break;
146                 case "HUMI":
147                     this.View.StrHumi = info.Split('=').Last();
148                     break;
149                 case "PRES":
150                     this.View.StrPres = info.Split('=').Last();
151                     break;
152             }
153         }
154     #endregion
155 }
156 }
```

```
1 <Page
2     x:Class="RaspiHomeTabletWindows.Modules.Home.Location.OtherRoom.RoomView"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeTabletWindows.Modules.Home.Location.OtherRoom"
6     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8     mc:Ignorable="d">
9
10    <RelativePanel x:Name="rpnldDisplay" Margin="10">
11        <!--LIGHT BUTTON-->
12        <Rectangle x:Name="rectTextLight" Width="17"/>
13        <TextBlock x:Name="tblLight" Text="Lumière"
14            RelativePanel.RightOf="rectTextLight"
15            RelativePanel.AlignTopWithPanel="True" FontSize="26"/>
16        <Button x:Name="btnLightOnOff" RelativePanel.Below="tblLight"
17            Height="120" Width="120" Background="{StaticResource
18                DefaultBackgroundColor}" HorizontalContentAlignment="Center"
19                VerticalContentAlignment="Center" Click="btnLightOnOff_Click">
20            <Image x:Name="imgLightButton" Source="ms-appx:///Icon/bulb.png"/>
21        </Button>
22
23        <!--STATE OF THE ROOM-->
24        <!--HUMIDITY-->
25        <Rectangle x:Name="rectHumid" RelativePanel.AlignBottomWithPanel="True" Height="34"/>
26        <Rectangle x:Name="rectInformation" RelativePanel.Above="rectHumid" Width="75"/>
27        <Image x:Name="imgHumidity" Visibility="Collapsed" Source="ms-appx:///Icon/Humidity.png" RelativePanel.AlignLeftWithPanel="True" RelativePanel.Above="rectHumid" Height="75"/>
28
29        <!--TEMPERATURE-->
30        <Rectangle x:Name="rectThermo" RelativePanel.Above="imgHumidity" Height="34"/>
31        <Image x:Name="imgThermometer" Visibility="Collapsed" Source="ms-appx:///Icon/Termometer.png" RelativePanel.AlignHorizontalCenterWith="imgHumidity" RelativePanel.Above="rectThermo" Height="75"/>
32
33        <!--PRESSURE-->
34        <Image x:Name="imgBarometer" Visibility="Collapsed" Source="ms-appx:///Icon/Pressure.png" RelativePanel.RightOf="rectPressure" RelativePanel.AlignVerticalCenterWith="imgThermometer" Height="65"/>
35        <TextBlock x:Name="tblTemperature"
36            RelativePanel.RightOf="rectInformation"
37            RelativePanel.AlignVerticalCenterWith="imgThermometer" FontSize="26"/>
38        <TextBlock x:Name="tblHumidity" RelativePanel.RightOf="rectInformation" RelativePanel.AlignVerticalCenterWith="imgHumidity" FontSize="26"/>
```

```
36     <TextBlock x:Name="tblPressure"                                     ↵
          RelativePanel.RightOf="rectInformation2"                         ↵
          RelativePanel.AlignVerticalCenterWith="imgBarometer" FontSize="26"/>
37
38
39
40     <!--STORE BUTTON-->
41     <!--UP-->
42     <Rectangle x:Name="rect2" Width="20"                                ↵
          RelativePanel.AlignRightWithPanel="True"/>
43     <TextBlock x:Name="tblUp" Text="Monter" RelativePanel.LeftOf="rect2"   ↵
          RelativePanel.AlignTopWithPanel="True" FontSize="26"/>
44     <Button x:Name="btnStoreUp" RelativePanel.AlignRightWithPanel="True"  ↵
          RelativePanel.Below="tblUp" Height="120" Width="120"                ↵
          Background="{StaticResource DefaultBackgroundColor}"                  ↵
          HorizontalContentAlignment="Center" VerticalContentAlignment="Center" ↵
          Click="btnStoreUp_Click">
45         <Image Source="ms-appx:///Icon/arrowUp.png"/>
46     </Button>
47
48     <Rectangle x:Name="rect3" Height="20" Width="120"                      ↵
          RelativePanel.AlignRightWithPanel="True"                            ↵
          RelativePanel.Below="btnStoreUp"/>
49
50     <!--DOWN-->
51     <Button x:Name="btnStoreDown" RelativePanel.AlignRightWithPanel="True" ↵
          RelativePanel.Below="rect3" Height="120" Width="120"                ↵
          Background="{StaticResource DefaultBackgroundColor}"                  ↵
          HorizontalContentAlignment="Center" VerticalContentAlignment="Center" ↵
          Click="btnStoreDown_Click">
52         <Image Source="ms-appx:///Icon/arrowDown.png"/>
53     </Button>
54     <TextBlock x:Name="tblDown" Text="Descendre"                           ↵
          RelativePanel.AlignHorizontalCenterWith="tblUp"                   ↵
          RelativePanel.Below="btnStoreDown" FontSize="26"/>
55
56     <!--OPEN-->
57     <TextBlock x:Name="tblOpen" Text="Ouvrir"                               ↵
          RelativePanel.AlignHorizontalCenterWith="tblUp"                   ↵
          RelativePanel.Above="btnStoreOpen" FontSize="26"/>
58     <Button x:Name="btnStoreOpen" RelativePanel.AlignRightWithPanel="True" ↵
          RelativePanel.Above="rect4" Height="120" Width="120"                ↵
          Background="{StaticResource DefaultBackgroundColor}"                  ↵
          HorizontalContentAlignment="Center" VerticalContentAlignment="Center" ↵
          Click="btnStoreOpen_Click">
59         <Image Source="ms-appx:///Icon/arrowUp.png"/>
60     </Button>
61
62     <Rectangle x:Name="rect4" Height="20" Width="150"                      ↵
          RelativePanel.AlignRightWithPanel="True"                            ↵
          RelativePanel.Above="btnStoreClose"/>
63
64     <!--CLOSE-->
65     <Button x:Name="btnStoreClose" RelativePanel.AlignRightWithPanel="True" ↵
          RelativePanel.Above="tblClose" Height="120" Width="120"                ↵
          Background="{StaticResource DefaultBackgroundColor}"                  ↵
```

```
    HorizontalContentAlignment="Center" VerticalContentAlignment="Center" ↵
    Click="btnStoreClose_Click">
66        <Image Source="ms-appx:///Icon/arrowDown.png"/>
67    </Button>
68    <TextBlock x:Name="tblClose" Text="Fermer" ↵
        RelativePanel.AlignHorizontalCenterWith="tblUp" ↵
        RelativePanel.AlignBottomWithPanel="True" FontSize="26"/>
69    </RelativePanel>
70 </Page>
71
```

```
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 Windows.UI.Xaml;
17 using Windows.UI.Xaml.Controls;
18 using Windows.UI.Xaml.Media.Imaging;
19
20 // The User Control item template is documented at http://go.microsoft.com/ →
21 //fwlink/?LinkId=234236
22
23 namespace RaspiHomeTabletWindows.Modules.Home.Location.OtherRoom
24 {
25     public sealed partial class RoomView : Page
26     {
27         #region Fields
28         #region Constants
29         #endregion
30         #region Varaibles
31         private RoomModel _model = null;
32
33         private string _strTemp = "";
34         private string _strHumi = "";
35         private string _strPres = "";
36
37         private bool _isOn = false;
38         private bool _isUp = false;
39         private bool _isDown = false;
40         private bool _isOpen = false;
41         private bool _isClose = false;
42         #endregion
43         #endregion
44
45         #region Properties
46         public RoomModel Model
47         {
48             get
49             {
50                 return _model;
51             }
52
53             set
54             {
55                 _model = value;
```

```
56         }
57     }
58
59     public string StrTemp
60     {
61         get
62         {
63             return _strTemp;
64         }
65
66         set
67         {
68             _strTemp = value;
69
70             this.tblTemperature.Text = value + " °C";
71         }
72     }
73
74     public string StrHumi
75     {
76         get
77         {
78             return _strHumi;
79         }
80
81         set
82         {
83             _strHumi = value;
84
85             this.tblHumidity.Text = value + " %";
86         }
87     }
88
89     public string StrPres
90     {
91         get
92         {
93             return _strPres;
94         }
95
96         set
97         {
98             _strPres = value;
99
100            this.tblPressure.Text = value + " [hPa]";
101        }
102    }
103
104    public bool IsOn
105    {
106        get
107        {
108            return _isOn;
109        }
110
111        set
```

```
112         {
113             _isOn = value;
114
115             if (value)
116             {
117                 // Send message to save
118                 this.Model.SendMessage("allumer", "lumiere");
119                 // Change the picture
120                 this.imgLightButton.Source = new BitmapImage(new Uri("ms- ↵
121                                         appx:///Icon/bulbLighting.png"));
122             }
123             else
124             {
125                 // Save message to save
126                 this.Model.SendMessage("eteindre", "lumiere");
127                 // Change the picture
128                 this.imgLightButton.Source = new BitmapImage(new Uri("ms- ↵
129                                         appx:///Icon/bulb.png"));
130             }
131         }
132
133         public bool IsUp
134     {
135         get
136         {
137             return _isUp;
138         }
139
140         set
141         {
142             _isUp = value;
143         }
144     }
145
146         public bool IsDown
147     {
148         get
149         {
150             return _isDown;
151         }
152
153         set
154         {
155             _isDown = value;
156         }
157     }
158
159         public bool IsOpen
160     {
161         get
162         {
163             return _isOpen;
164         }
165
166         set
```

```
166         {
167             _isOpen = value;
168         }
169     }
170
171     public bool IsClose
172     {
173         get
174         {
175             return _isClose;
176         }
177
178         set
179         {
180             _isClose = value;
181         }
182     }
183 #endregion
184
185 #region Constructor
186 /// <summary>
187 /// Constructor: Initializer
188 /// </summary>
189 public RoomView()
190 {
191     this.InitializeComponent();
192
193     this.Model = new RoomModel(this);
194 }
195 #endregion
196
197 #region Events
198 /// <summary>
199 /// Light control
200 /// </summary>
201 private void btnLightOnOff_Click(object sender, RoutedEventArgs e)
202 {
203     this.isOn = !this.isOn;
204 }
205
206 /// <summary>
207 /// Store control
208 /// </summary>
209 private void btnStoreUp_Click(object sender, RoutedEventArgs e)
210 {
211     this.isUp = true;
212     this.Model.SendMessage("monter", "store");
213 }
214
215 private void btnStoreDown_Click(object sender, RoutedEventArgs e)
216 {
217     this.isDown = true;
218     this.Model.SendMessage("descendre", "store");
219 }
220
221 private void btnStoreOpen_Click(object sender, RoutedEventArgs e)
```

```
222     {
223         this.IsOpen = true;
224         this.Model.SendMessage("ouvrir", "store");
225     }
226
227     private void btnStoreClose_Click(object sender, RoutedEventArgs e)
228     {
229         this.Close = true;
230         this.Model.SendMessage("fermer", "store");
231     }
232 #endregion
233
234 #region Methods
235 /// <summary>
236 /// Set to visible the state values if exist
237 /// </summary>
238 public void EnableDisplayState()
239 {
240     this.imgThermometer.Visibility = Visibility.Visible;
241     this.imgHumidity.Visibility = Visibility.Visible;
242     this.imgBarometer.Visibility = Visibility.Visible;
243 }
244 #endregion
245 }
246 }
247 }
```

```
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 namespace RaspiHomeTabletWindows.Modules.GlobalSetup
16 {
17     public class GlobalSetupModel : PropertyChangedBase
18     {
19         #region Fields
20         #region Constants
21         #endregion
22
23         #region Varaibles
24         private GlobalSetupView _view = null;
25         #endregion
26         #endregion
27
28         #region Properties
29         public GlobalSetupView View
30         {
31             get
32             {
33                 return _view;
34             }
35
36             set
37             {
38                 _view = value;
39             }
40         }
41         #endregion
42
43         #region Constructor
44         public GlobalSetupModel(GlobalSetupView paramView)
45         {
46             this.View = paramView;
47         }
48         #endregion
49
50         #region Methods
51         #endregion
52     }
53 }
```

```
1 <Page
2     x:Class="RaspiHomeTabletWindows.Modules.GlobalSetup.GlobalSetupView"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeTabletWindows.Modules.GlobalSetup"
6     xmlns:d="http://schemas.microsoft.com/expressionblend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8     mc:Ignorable="d">
9
10    <Grid>
11        <Image Source="ms-appx:///Icon/GlobalSetup.png"/>
12    </Grid>
13 </Page>
14
```

```
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 Windows.UI.Xaml.Controls;
16
17 // The User Control item template is documented at http://go.microsoft.com/
18 // fwlink/?LinkId=234236
19
20 namespace RaspiHomeTabletWindows.Modules.GlobalSetup
21 {
22     public sealed partial class GlobalSetupView : Page
23     {
24         #region Fields
25         #region Constants
26         #endregion
27         #region Varaibles
28         private GlobalSetupModel _model = null;
29         #endregion
30         #endregion
31
32         #region Properties
33         public GlobalSetupModel Model
34         {
35             get
36             {
37                 return _model;
38             }
39
40             set
41             {
42                 _model = value;
43             }
44         }
45         #endregion
46
47         #region Constructor
48         /// <summary>
49         /// Constructor: Initializer
50         /// </summary>
51         public GlobalSetupView()
52         {
53             this.InitializeComponent();
54
55             this.Model = new GlobalSetupModel(this);
```

```
56     }
57     #endregion
58
59     #region Events
60     #endregion
61
62     #region Methods
63     #endregion
64 }
65 }
66
```

```
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 namespace RaspiHomeTabletWindows.Modules.Information
16 {
17     public class InformationModel : PropertyChangedBase
18     {
19         #region Fields
20         #region Constants
21         #endregion
22
23         #region Varaibles
24         private InformationView _view = null;
25         #endregion
26         #endregion
27
28         #region Properties
29         public InformationView View
30         {
31             get
32             {
33                 return _view;
34             }
35
36             set
37             {
38                 _view = value;
39             }
40         }
41         #endregion
42
43         #region Constructor
44         /// <summary>
45         /// Constructor: Initializer
46         /// </summary>
47         public InformationModel(InformationView paramView)
48         {
49             this.View = paramView;
50         }
51         #endregion
52
53         #region Methods
54         #endregion
55     }
56 }
```

```
1 <Page
2     x:Class="RaspiHomeTabletWindows.Modules.Information.InformationView"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeTabletWindows.Modules.Information"
6     xmlns:d="http://schemas.microsoft.com/expressionblend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8     mc:Ignorable="d">
9
10    <Grid>
11        <Image Source="ms-appx:///Icon/Information.png" />
12    </Grid>
13 </Page>
14
```

```
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 Windows.UI.Xaml.Controls;
16
17 // The User Control item template is documented at http://go.microsoft.com/fwlink/?LinkId=234236
18
19 namespace RaspiHomeTabletWindows.Modules.Information
20 {
21     public sealed partial class InformationView : Page
22     {
23         #region Fields
24         #region Constants
25         #endregion
26
27         #region Varaibles
28         private InformationModel _model = null;
29         #endregion
30         #endregion
31
32         #region Properties
33         public InformationModel Model
34         {
35             get
36             {
37                 return _model;
38             }
39
40             set
41             {
42                 _model = value;
43             }
44         }
45         #endregion
46
47         #region Constructor
48         /// <summary>
49         /// Constructor: Initializer
50         /// </summary>
51         public InformationView()
52         {
53             this.InitializeComponent();
54
55             this.Model = new InformationModel(this);
```

```
56     }
57     #endregion
58
59     #region Events
60     #endregion
61
62     #region Methods
63     #endregion
64 }
65 }
66
```

```
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 namespace RaspiHomeTabletWindows.Modules.Settings
16 {
17     public class SettingModel : PropertyChangedBase
18     {
19         #region Fields
20         #region Constants
21         #endregion
22
23         #region Variables
24         private SettingView _view = null;
25         #endregion
26         #endregion
27
28         #region Properties
29         public SettingView View
30         {
31             get
32             {
33                 return _view;
34             }
35
36             set
37             {
38                 _view = value;
39             }
40         }
41         #endregion
42
43         #region Constructor
44         /// <summary>
45         /// Constructor: Initializer
46         /// </summary>
47         public SettingModel(SettingView paramView)
48         {
49             this.View = paramView;
50         }
51         #endregion
52
53         #region Methods
54         #endregion
55     }
56 }
```

```
1 <Page
2     x:Class="RaspiHomeTabletWindows.Modules.Settings.SettingsView"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeTabletWindows.Modules.Settings"
6     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8     mc:Ignorable="d">
9
10    <Grid>
11        <Image Source="ms-appx:///Icon/Setting.png" />
12    </Grid>
13 </Page>
14
```

```
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 Windows.UI.Xaml.Controls;
16
17 // The User Control item template is documented at http://go.microsoft.com/
18 // fwlink/?LinkId=234236
19
20 namespace RaspiHomeTabletWindows.Modules.Settings
21 {
22     public sealed partial class SettingView : Page
23     {
24         #region Fields
25         #region Constants
26         #endregion
27         #region Variables
28         private SettingModel _model = null;
29         #endregion
30         #endregion
31
32         #region Properties
33         public SettingModel Model
34         {
35             get
36             {
37                 return _model;
38             }
39
40             set
41             {
42                 _model = value;
43             }
44         }
45         #endregion
46
47         #region Constructor
48         /// <summary>
49         /// Constructor: Initializer
50         /// </summary>
51         public SettingView()
52         {
53             this.InitializeComponent();
54
55             this.Model = new SettingModel(this);
```

```
56     }
57     #endregion
58
59     #region Events
60     #endregion
61
62     #region Methods
63     #endregion
64 }
65 }
66
```

```
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.ComponentModel;
16
17 namespace RaspiHomeTabletWindows
18 {
19     public class PropertyChangedBase
20     {
21         public event PropertyChangedEventHandler PropertyChanged;
22
23         protected virtual void OnPropertyChanged(string propertyName)
24         {
25             //Raise the PropertyChanged event on the UI Thread, with the
26             //relevant propertyName parameter:
27             if (PropertyChanged != null)
28             {
29                 PropertyChanged(this, new PropertyChangedEventArgs
30                     (propertyName));
31             }
32         }
33     }
```

```
1 <Application
2     x:Class="RaspiHomeTabletWindows.App"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeTabletWindows"
6     RequestedTheme="Light">
7     <Application.Resources>
8         <!--| COLOR |-->
9         <Color x:Key="White">#FFFDFDFD</Color>
10        <Color x:Key="LightGray">#FFEEEEEE</Color>
11        <Color x:Key="Gray">#FFBCBCBC</Color>
12        <Color x:Key="DarkGray">#FF2C2C2C</Color>
13        <Color x:Key="ShadowGray">#FF4C4A4B</Color>
14        <Color x:Key="GreenForest">#FF498205</Color>
15
16        <!--Event color-->
17        <Color x:Key="WhiteOnEvent">#80F5F5F5</Color>
18        <Color x:Key="DarkGrayOnEvent">#802C2C2C</Color>
19        <Color x:Key="BlueAzureOnEvent">#80498205</Color>
20
21        <!--| BRUSH |-->
22        <SolidColorBrush x:Key="DefaultBackColor"
23             Color="{StaticResource White}"/>
24
25        <SolidColorBrush x:Key="DefaultBackgroundColor"
26             Color="{StaticResource ShadowGray}"/>
27
28        <SolidColorBrush x:Key="DefaultBorderColor"
29             Color="{StaticResource DarkGray}"/>
30
31        <SolidColorBrush x:Key="OverColor"
32             Color="{StaticResource WhiteOnEvent}"/>
33
34        <SolidColorBrush x:Key="PressedColor"
35             Color="{StaticResource DarkGrayOnEvent}"/>
36        <!--| STYLE |-->
37        <Style x:Name="styleRoundButton" TargetType="Button">
38            <Setter Property="Background" Value="{StaticResource
39                DefaultBackColor}"/>
40            <Setter Property="BorderBrush" Value="{StaticResource
41                DefaultBackgroundColor}"/>
42            <Setter Property="BorderThickness" Value="5"/>
43            <Setter Property="Template">
44                <Setter.Value>
45                    <ControlTemplate TargetType="Button">
46                        <Border x:Name="Border"
47                            Background="{TemplateBinding Background}"
48                            BorderBrush="{TemplateBinding BorderBrush}"
49                            BorderThickness="{TemplateBinding BorderThickness}"
50                            CornerRadius="70">
51                            <ContentPresenter
52                                HorizontalAlignment="{TemplateBinding
53                                    HorizontalContentAlignment}"
54                                VerticalAlignment="{TemplateBinding
55                                    VerticalContentAlignment}"/>
56                        </Border>
```

```
52          </ControlTemplate>
53          </Setter.Value>
54      </Setter>
55      </Style>
56  </Application.Resources>
57 </Application>
58
```

```
1  /*-----*/
2  * Author    : Salvi Cyril
3  * Date      : 7th juny 2017
4  * Diploma   : RaspiHome
5  * Classroom : T.IS-E2B
6  *
7  * Description:
8  *      RaspiHomeSenseHAT is a program who use a
9  *      Sense HAT, it's an electronic card who can be
10 *      mesured value with sensor. This program use
11 *      the Sense HAT to mesure the temperature, the
12 *      humidity and the pressure.
13 \*-----*/
14
15 using System;
16 using Emmellsoft.IoT.Rpi.SenseHat;
17 using Windows.UI;
18 using Windows.UI.Xaml;
19
20 namespace RaspiHomeSenseHAT
21 {
22     public class ModelSenseHAT
23     {
24         #region Fields
25         #region Constants
26         #endregion
27
28         #region Variables
29         private ViewSenseHAT _vSenseHAT;
30         private CommunicationWithServer _comWithServer;
31
32         // Sense HAT librairy
33         private ISenseHat _senseHat;
34         private ISenseHatDisplay _senseHatDisplay;
35         private SenseHatData _data;
36
37         // Set default color matrix to OFF
38         private Color _uiColor = Color.FromArgb(0, 0, 0, 0);
39         #endregion
40         #endregion
41
42         #region Properties
43         public ViewSenseHAT VSenseHAT
44         {
45             get
46             {
47                 return _vSenseHAT;
48             }
49
50             set
51             {
52                 _vSenseHAT = value;
53             }
54         }
55
56         public SenseHatData Data
```

```
57         {
58             get
59             {
60                 return _data;
61             }
62
63             set
64             {
65                 _data = value;
66             }
67         }
68
69     public CommunicationWithServer ComWithServer
70     {
71         get
72         {
73             return _comWithServer;
74         }
75
76         set
77         {
78             _comWithServer = value;
79         }
80     }
81 #endregion
82
83 #region Constructors
84 /// <summary>
85 /// Constructor: Initializer
86 /// </summary>
87 /// <param name="paramView"></param>
88 public ModelSenseHAT(ViewSenseHAT paramView)
89 {
90     // Communication like Model-View
91     this.VSenseHAT = paramView;
92
93     // Initialize the Sense HAT (don't need to be initialized before the communication start because it's only a sensor) ↵
94     InitializeSenseHat();
95
96     // Initialize the communication with the server
97     this.ComWithServer = new CommunicationWithServer(this); ↵
98 }
99 #endregion
100
101 #region Methods
102 /// <summary>
103 /// Initialize the Sense HAT
104 /// </summary>
105 public async void InitializeSenseHat()
106 {
107     this._senseHat = await SenseHatFactory.GetSenseHat();
108     this._senseHatDisplay = this._senseHat.Display;
109     this._senseHatDisplay.Fill(_uiColor);
110 }
```

```
111         SetValue();  
112     }  
113  
114     /// <summary>  
115     /// Set the value get with sensor  
116     /// </summary>  
117     public void SetValue()  
118     {  
119         // Update values  
120         this._senseHat.Sensors.HumiditySensor.Update();  
121         this._senseHat.Sensors.PressureSensor.Update();  
122         this._senseHatDisplay.Update();  
123  
124         // Set values  
125         this.Data = new SenseHatData();  
126         this.Data.Temperature = this._senseHat.Sensors.Temperature;  
127         this.Data.Humidity = this._senseHat.Sensors.Humidity;  
128         this.Data.Pressure = this._senseHat.Sensors.Pressure;  
129     }  
130  
131     /// <summary>  
132     /// Send the values with a special format:"TEMP=x;HUMI=y;PRES=z"  
133     /// /// Values are rounded  
134     /// </summary>  
135     /// <returns></returns>  
136     public string SendValues()  
137     {  
138         // Update values of sensors  
139         SetValue();  
140  
141         return "TEMP=" + Math.Round((decimal)this.Data.Temperature) + ";" +  
142             + "HUMI=" + Math.Round((decimal)this.Data.Humidity) + ";" +  
143             + "PRES=" + Math.Round((decimal)this.Data.Pressure);  
144     }  
145     #endregion  
146 }
```

```
1  /*-----*\n2   * Author      : Salvi Cyril\n3   * Date        : 7th juny 2017\n4   * Diploma     : RaspiHome\n5   * Classroom   : T.IS-E2B\n6   *\n7   * Description:\n8   *           RaspiHomeSenseHAT is a program who use a\n9   *           Sense HAT, it's an electronic card who can be\n10  *           mesured value with sensor. This program use\n11  *           the Sense HAT to mesure the temperature, the\n12  *           humidity and the pressure.\n13 \/*-----*/\n14\n15 namespace RaspiHomeSenseHAT\n16 {\n17     public class SenseHatData\n18     {\n19         public double? Humidity { get; set; }\n20         public double? Pressure { get; set; }\n21         public double? Temperature { get; set; }\n22     }\n23 }\n24
```

```
1  /*-----*/
2  * Author      : Salvi Cyril
3  * Date        : 7th juny 2017
4  * Diploma     : RaspiHome
5  * Classroom   : T.IS-E2B
6  *
7  * Description:
8  *      RaspiHomeSenseHAT is a program who use a
9  *      Sense HAT, it's an electronic card who can be
10 *      mesured value with sensor. This program use
11 *      the Sense HAT to mesure the temperature, the
12 *      humidity and the pressure.
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
23 namespace RaspiHomeSenseHAT
24 {
25     public class CommunicationWithServer
26     {
27         #region Fields
28         #region Constants
29             // Default information to connect on the server
30             private const int PORT = 54565;
31             //// Need to be changed fo each configuration
32             private const string IPSERVER = "10.134.97.117";// "192.168.2.8";
33
34             // String format for connection of the client
35             private const string FORMATSTRING = "IPRasp={0};Location=
36                 {1};Component={2}";
37             private const string COMMUNICATIONSEPARATOR = "@";
38
39             // Important need to be changed if it's another room!
40             private const string LOCATION = "Salon";
41             private const string COMPONENT = "Sensor";
42             private const string RPINAME = "SenseHAT_" + LOCATION;
43
44             private const int MESSAGE_FULL_LENGTH = 512;
45             #endregion
46
47             #region Variables
48             private ModelSenseHAT _mSenseHAT;
49
50             private StreamSocket _socket = new StreamSocket();
51             private StreamSocketListener _listener = new StreamSocketListener();
52             private List<StreamSocket> _connections = new List<StreamSocket>();
53             private bool _isConnected = false;
54             private bool _connecting = false;
55             #endregion
56             #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),      ↴
160                 PORT.ToString());
161             SendForInitialize();
162             this.Connecting = false;
163             this.IsConnected = true;
164
165             WaitForData(this.Socket);
166         }
167         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,      ↗
183     StreamSocketListenerConnectionReceivedEventArgs args)
184 {
185     this.Connections.Add(args.Socket);
186
187     WaitForData(args.Socket);
188 }
189
190 /// <summary>
191 /// Send the message in input to output
192 /// </summary>
193 /// <param name="socket"> actual stream </param>
194 /// <param name="message"> message to send </param>
195 private async void SendMessage(StreamSocket socket, string message)
196 {
197     DataWriter dataWriter = new DataWriter(socket.OutputStream);
198     var len = dataWriter.MeasureString(message); // Gets the UTF-8      ↗
199     string length.
200     dataWriter.WriteInt32((int)len);
201     dataWriter.WriteString(message);
202     var ret = await dataWriter.StoreAsync();
203     dataWriter.DetachStream();
204 }
205
206 /// <summary>
207 /// Wait data readed if exist
208 /// </summary>
209 /// <param name="socket"></param>
210 private async void WaitForData(StreamSocket socket)
211 {
212     await Task.Delay(TimeSpan.FromMilliseconds(200));
213     DataReader dataReader = new DataReader(socket.InputStream);
214     dataReader.InputStreamOptions = InputStreamOptions.Partial;
215     var msglength = dataReader.UnconsumedBufferLength;
216     uint stringBytes = msglength;
217
218     try
219     {
220         // Read modification in the stream
221         stringBytes = await dataReader.LoadAsync(MESSAGE_FULL_LENGTH); ↗
```

```
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 (msglength < 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:
248         // "@NAME@Connection:IPRASP=x.x.x.x;Location=y;Component=z,z"
249         SendMessage(this.Socket, string.Format(COMMUNICATIONSEPARATOR +
250                                         RPINAME + COMMUNICATIONSEPARATOR + "Connection:" + FORMATSTRING,
251                                         GetHostName(), LOCATION, COMPONENT));
252
253     /// <summary>
254     /// Send values in reply to the server
255     /// </summary>
256     public void ReplyValues()
257     {
258         // Message receive : "@Reply:TEMP=x;HUMI=y;PRES=z"
259         SendMessage(this.Socket, COMMUNICATIONSEPARATOR + "Reply:" +
260                     this.MSenseHAT.SendValues());
261
262     /// <summary>
263     /// Get the ip of the raspberry
264     /// </summary>
265     /// <returns>return a string like 192.168.1.2</returns>
266     public string GetHostName()
267     {
268         List<string> IpAddress = new List<string>();
269         var Hosts =
270             Windows.Networking.Connectivity.NetworkInformation.GetHostNames
271             ().ToList();
272         foreach (var Host in Hosts)
273         {
```

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

```
1 <Page
2     x:Class="RaspiHomeSenseHAT.ViewSenseHAT"
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5     xmlns:local="using:RaspiHomeSenseHAT"
6     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8     mc:Ignorable="d">
9 </Page>
10
```

```
1  /*-----*\n2   * Author      : Salvi Cyril\n3   * Date        : 7th juny 2017\n4   * Diploma     : RaspiHome\n5   * Classroom   : T.IS-E2B\n6   *\n7   * Description:\n8   *           RaspiHomeSenseHAT is a program who use a\n9   *           Sense HAT, it's an electronic card who can be\n10  *          mesured value with sensor. This program use\n11  *          the Sense HAT to mesure the temperature, the\n12  *          humidity and the pressure.\n13 \/*-----*/\n14\n15 using Windows.UI.Xaml.Controls;\n16\n17 // Pour plus d'informations sur le modèle d'élément Page vierge, consultez la ↵\n18 // page http://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409\n19 namespace RaspiHomeSenseHAT\n20 {\n21     /// <summary>\n22     /// Une page vide peut être utilisée seule ou constituer une page de      ↵\n23     /// destination au sein d'un frame.\n24     public sealed partial class ViewSenseHAT : Page\n25     {\n26         #region Fields\n27         #region Constants\n28         #endregion\n29\n30         #region Variables\n31         private ModelSenseHAT _mSenseHAT;\n32         #endregion\n33         #endregion\n34\n35         #region Properties\n36         public ModelSenseHAT MSenseHAT\n37         {\n38             get\n39             {\n40                 return _mSenseHAT;\n41             }\n42\n43             set\n44             {\n45                 _mSenseHAT = value;\n46             }\n47         }\n48         #endregion\n49\n50         #region Constructors\n51         /// <summary>\n52         /// Constructor: Initializer\n53         /// </summary>\n54         public ViewSenseHAT()
```

```
55         {
56             this.InitializeComponent();
57
58             this.MSenseHAT = new ModelSenseHAT(this);
59         }
60     #endregion
61 }
62 }
63
```

```
1  /*-----*/
2  * Author    : Salvi Cyril
3  * Date      : 7th juny 2017
4  * Diploma   : RaspiHome
5  * Classroom : T.IS-E2B
6  *
7  * Description:
8  *      RaspiHomePiFaceDigital2 is a program who use
9  *      a PiFace Digital 2, it's an electronic card who
10 *      can be use to plug electronic component. This
11 *      program use the PiFace Digital 2 to activate
12 *      light and store.
13 \*-----*/
14
15 using System;
16 using System.Collections.Generic;
17 using System.Diagnostics;
18 using System.Globalization;
19 using System.Linq;
20 using System.Reflection;
21 using System.Text;
22
23 namespace RaspiHomePiFaceDigital2
24 {
25     public class ModelPiFaceDigital2
26     {
27         #region Fields
28         #region Constants
29         #endregion
30
31         #region Variables
32         private ViewPiFaceDigital2 _vPiFace;
33
34         private List<Component> _components;
35         private CommunicationWithServer _comWithServer;
36
37         // Command to know
38         private List<string> _raspiHomeComponentKnown = new List<string>()
39         {
40             "lumiere", "lumieres",
41             "store", "stores",
42             "television", "televisions",
43             "porte", "portes",
44             "fenetre", "fenetres",
45         };
46
47         private List<string> _raspiHomeActionKnown = new List<string>()
48         {
49             "allumer", "allume",
50             "eteindre", "eteins",
51             "monter", "monte",
52             "descendre", "descends",
53             "stopper", "stop",
54             "ouvrir", "ouvre",
55             "fermer", "ferme",
56             "stopper", "stop",
```

```
57         };
58
59         // Word translation
60         private Dictionary<string, string> _raspiLanguageTranslation = new Dictionary<string, string>()
61     {
62         { "lumiere", "Light" }, { "lumieres", "Light" },
63         { "store", "Store" }, { "stores", "Store" },
64     };
65
66         // KEY=[ACTION NAME], VALUE[KEY=[PROPERTY NAME], VALUE=[VALUE TO SET THEPROPERTY]]
67         private Dictionary<string, Dictionary<string, bool>>
68             _raspiBooleanCommandTranslation = new Dictionary<string, Dictionary<string, bool>>()
69     {
70         { "allume", new Dictionary<string, bool> { { "IsOn", true } } },
71         { "allumer", new Dictionary<string, bool> { { "IsOn", true } } },
72         { "eteins", new Dictionary<string, bool> { { "IsOn", false } } },
73         { "eteindre", new Dictionary<string, bool> { { "IsOn", false } } },
74         { "monte", new Dictionary<string, bool> { { "IsUp", true } } },
75         { "monter", new Dictionary<string, bool> { { "IsUp", true } } },
76         { "descends", new Dictionary<string, bool> { { "IsDown", true } } },
77         { "descendre", new Dictionary<string, bool> { { "IsDown", true } } },
78         { "stop", new Dictionary<string, bool> { { "IsStop", true } } },
79         { "stopper", new Dictionary<string, bool> { { "IsStop", true } } },
80     };
81     #endregion
82     #endregion
83
84     #region Properties
85     public ViewPiFaceDigital2 VPiFace
86     {
87         get
88         {
89             return _vPiFace;
90         }
91
92         public List<Component> Components
93     {
94         get
95         {
96             return _components;
97         }
98
99         set
100        {
```

```
101             _components = value;
102         }
103     }
104
105     public CommunicationWithServer ComWithServer
106     {
107         get
108         {
109             return _comWithServer;
110         }
111
112         set
113         {
114             _comWithServer = value;
115         }
116     }
117 #endregion
118
119 #region Constructors
120 /// <summary>
121 /// Constructor: Initializer
122 /// </summary>
123 /// <param name="paramView"></param>
124 public ModelPiFaceDigital2(ViewPiFaceDigital2 paramView)
125 {
126     // Communication like Model-View
127     this.VPiFace = paramView;
128
129     // Initialize the components and add the components linked with the Raspberry
130     this.Components = new List<Component>();
131     this.Components.Add(new Light());
132     this.Components.Add(new Store());
133
134     // Initialize the PiFace Digital 2
135     InitializePiFace();
136
137     // Initialize the server communication
138     this.ComWithServer = new CommunicationWithServer(this);
139 }
140 #endregion
141
142 #region Methods
143 /// <summary>
144 /// Initialize the PiFace Digital 2
145 /// </summary>
146 private async void InitializePiFace()
147 {
148     try
149     {
150         await MCP23S17.InitializeSPI();
151
152         MCP23S17.InitializeMCP23S17();
153         MCP23S17.SetPinMode(0x00FF); // 0x0000 = all outputs, 0xffff=all inputs, 0x00FF is PIface Default
154         MCP23S17.PullupMode(0x00FF); // 0x0000 = no pullups,
```

```
155             0xffff=all pullups, 0x00FF is PIFace Default
156             MCP23S17.WriteByte(0x0000); // 0x0000 = no pullups, 0xffff=all ↵
157             pullups, 0x00FF is PIFace Default
158         }
159         catch (Exception ex)
160     {
161         Debug.WriteLine(ex.Message);
162     }
163 
164     /// <summary>
165     /// Set the value to be writed on the PiFace
166     /// </summary>
167     /// <param name="messageRead"> message read from the server </param>
168     public void SetValue(string messageRead)
169     {
170         // Initialize the message value
171         string sentence = this.RemoveDiacritics(messageRead);
172         string action = this.GetActionFromSentence(sentence);
173         string actionValue = this.ReadValueOfSelectedComponent(action);
174         string component = this.GetComponentFromSentence(sentence);
175         Type componentType = this.GetComponentType(component);
176 
177         foreach (Component itemType in this.Components)
178         {
179             if (itemType.GetType() == componentType)
180             {
181                 this.writeValue(itemType, action, itemType.GetType()
182                               .GetProperty(actionValue));
183             }
184         }
185 
186     /// <summary>
187     /// Find location exist
188     /// </summary>
189     /// <param name="sentence"> sentence order</param>
190     /// <returns> return the action linked to the action word </returns>
191     private string GetActionFromSentence(string sentence)
192     {
193         string result = "";
194         string[] words = sentence.ToLower().Split(' ');
195 
196         foreach (var word in words)
197         {
198             if (this._raspiHomeActionKnown.Contains(word))
199             {
200                 result = word;
201                 break;
202             }
203         }
204 
205         return result;
206     }
207 
208     /// <summary>
```

```
208     /// Get the component called
209     /// </summary>
210     /// <param name="sentence"> sentence order </param>
211     /// <returns> return the component linked to the component word </>
212     /// returns
213     private string GetComponentFromSentence(string sentence)
214     {
215         string result = "";
216         string[] words = sentence.ToLower().Split(' ');
217
218         foreach (var word in words)
219         {
220             if (this._raspiHomeComponentKnown.Contains(word))
221             {
222                 result = word;
223                 break;
224             }
225         }
226
227         return result;
228     }
229
230     /// <summary>
231     /// Find all client who have the object in the sentence
232     /// </summary>
233     /// <param name="componentName"></param>
234     /// <returns>the object type</returns>
235     private Type GetComponentType(string componentName)
236     {
237         Type result = null;
238         Type[] types = typeof(Component).GetTypeInfo().Assembly.GetTypes()();
239
240         foreach (var typeOfComonent in types)
241         {
242             if (typeOfComonent.Name == this._raspiLanguageTranslation[componentName])
243             {
244                 result = typeOfComonent;
245                 break;
246             }
247         }
248
249         return result;
250     }
251
252     /// <summary>
253     /// Read properties value of classes
254     /// </summary>
255     /// <param name="actionName"> name used to change the good property </>
256     /// <returns> return the name of the property to change the value </>
257     /// returns
258     private string ReadValueOfSelectedComponent(string actionName)
259     {
260         string result = "";
```

```
259
260         foreach (var actionKeys in
261             this._raspiBooleanCommandTranslation.Keys)
262             if (actionKeys == actionName)
263             {
264                 // Find the Value of the dictionary trough the inner
265                 // dictionary to get the first value
266                 result = this._raspiBooleanCommandTranslation
267                 [actionName].First().Key;
268                 break;
269             }
270
271         return result;
272     }
273
274     /// <summary>
275     /// Search the val to change
276     /// </summary>
277     /// <param name="component"> the component to write value </param>
278     /// <param name="action"> the action (ON/OFF) </param>
279     /// <param name="typeVariable"> the property to change value </param>
280     private void WriteValue(Component component, string action,
281                             PropertyInfo typeVariable)
282     {
283         switch (typeVariable.PropertyType.Name)
284         {
285             case "Boolean":
286                 // Set the new value dynamaly with value registered in
287                 // an boolean dictionary
288                 typeVariable.SetValue(component,
289                     this._raspiBooleanCommandTranslation[action]
290                     [typeVariable.Name]);
291                 break;
292             case "Double":
293                 break;
294             case "Int16":
295             case "Int32":
296             case "Int64":
297                 break;
298         }
299
300     }
301
302     /// <summary>
303     /// Stack Overflow solution to delete accents in strings
304     /// http://stackoverflow.com/questions/249087/how-do-i-remove-
305     /// diacritics-accents-from-a-string-in-net
306     /// </summary>
307     /// <param name="sentence"> sentence with diacritics to remove </
308     /// param>
309     /// <returns> same sentence without diacritics </returns>
310     private string RemoveDiacritics(string sentence)
311     {
312         var normalizedString = sentence.Normalize
313             (NormalizationForm.FormD);
314         var stringBuilder = new StringBuilder();
```

```
305         foreach (var c in normalizedString)
306         {
307             var unicodeCategory = CharUnicodeInfo.GetUnicodeCategory(c);
308             if (unicodeCategory != UnicodeCategory.NonSpacingMark)
309             {
310                 stringBuilder.Append(c);
311             }
312         }
313
314         return stringBuilder.ToString().Normalize
315             (NormalizationForm.FormC);
316     #endregion
317 }
318 }
319 }
```

```
1  /*-----*/
2  * Author      : Salvi Cyril
3  * Date        : 7th juny 2017
4  * Diploma     : RaspiHome
5  * Classroom   : T.IS-E2B
6  *
7  * Description:
8  *      RaspiHomePiFaceDigital2 is a program who use
9  *      a PiFace Digital 2, it's an electronic card who
10 *      can be use to plug electronic component. This
11 *      program use the PiFace Digital 2 to activate
12 *      light and store.
13 \*-----*/
14
15 using System;
16 using System.Collections.Generic;
17 using System.Linq;
18 using Windows.Networking;
19 using Windows.Networking.Sockets;
20 using Windows.Storage.Streams;
21
22 namespace RaspiHomePiFaceDigital2
23 {
24     public class CommunicationWithServer
25     {
26         #region Fields
27         #region Constants
28             // Default information to connect on the server
29             private const int PORT = 54565;
30             //// Need to be changed fo each configuration
31             private const string IPSERVER = "10.134.97.117";// "192.168.2.8";
32
33             // String format for connection of the client
34             private const string FORMATSTRING = "Connection:IPRasp={0};Location=    ↴
35             {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 RPINAME = "PiFace_" + LOCATION;
41
42             private const int MESSAGE_FULL_LENGTH = 512;
43         #endregion
44
45         #region Variables
46             private ModelPiFaceDigital2 _mPiFace;
47
48             // Connection's variable
49             private StreamSocket _socket = new StreamSocket();
50             private StreamSocketListener _listener = new StreamSocketListener();
51             private List<StreamSocket> _connections = new List<StreamSocket>();
52             private bool _isConnected = false;
53             private bool _connecting = false;
54         #endregion
55         #endregion
56     }
57 }
```

```
56     #region Properties
57     public ModelPiFaceDigital2 MPiFace
58     {
59         get
60         {
61             return _mPiFace;
62         }
63     }
64     set
65     {
66         _mPiFace = value;
67     }
68 }
69
70     public StreamSocket Socket
71     {
72         get
73         {
74             return _socket;
75         }
76     }
77     set
78     {
79         _socket = value;
80     }
81 }
82
83     public StreamSocketListener Listener
84     {
85         get
86         {
87             return _listener;
88         }
89     }
90     set
91     {
92         _listener = value;
93     }
94 }
95
96     public List<StreamSocket> Connections
97     {
98         get
99         {
100             return _connections;
101         }
102     }
103     set
104     {
105         _connections = value;
106     }
107 }
108
109     public bool IsConnected
110     {
111         get
```

```
112         {
113             return _isConnected;
114         }
115
116         set
117     {
118         _isConnected = value;
119     }
120 }
121
122     public bool Connecting
123 {
124     get
125     {
126         return _connecting;
127     }
128
129     set
130     {
131         _connecting = value;
132     }
133 }
134 #endregion
135
136 #region Constructors
137 /// <summary>
138 /// Constructor: Initializer
139 /// </summary>
140 /// <param name="paramModel"></param>
141     public CommunicationWithServer(ModelPiFaceDigital2 paramModel)
142 {
143     this.MPiFace = paramModel;
144
145     Connect();
146 }
147 #endregion
148
149 #region Methods
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         // wait a confirmation from the server
160         await this.Socket.ConnectAsync(new HostName(IPSERVER),
161                                         PORT.ToString());
162         SendForInitialize();
163         this.Connecting = false;
164         this.IsConnected = true;
165
166         WaitForData(this.Socket);
167     }
168 }
```

```
167         catch (Exception)
168     {
169         this.Connecting = false;
170         this.IsConnected = false;
171     }
172 }
173
174 /// <summary>
175 /// Listen the traffic on the port
176 /// </summary>
177 private async void Listen()
178 {
179     this.Listener.ConnectionReceived += listenerConnectionReceived;
180     await this.Listener.BindServiceNameAsync(PORT.ToString());
181 }
182
183 void listenerConnectionReceived(StreamSocketListener sender,             ↪
184     StreamSocketListenerConnectionReceivedEventArgs args)
185 {
186     this.Connections.Add(args.Socket);
187
188     WaitForData(args.Socket);
189 }
190
191 /// <summary>
192 /// Send the message in input to output
193 /// </summary>
194 /// <param name="socket"> actual stream </param>
195 /// <param name="message"> message to send </param>
196 private async void SendMessage(StreamSocket socket, string message)
197 {
198     DataWriter dataWriter = new DataWriter(socket.OutputStream);
199     var len = dataWriter.MeasureString(message); // Gets the UTF-8      ↪
200         string length.
201     dataWriter.WriteInt32((int)len);
202     dataWriter.WriteString(message);
203     var ret = await dataWriter.StoreAsync();
204     dataWriter.DetachStream();
205 }
206
207 /// <summary>
208 /// Send to initialize the raspberry to the server
209 /// </summary>
210 private void SendForInitialize()
211 {
212     // Message send:
213     // "@NAME@Connection:IPRASP=x.x.x.x;Location=y;Component=z,z"
214     SendMessage(this.Socket, string.Format(COMMUNICATIONSEPARATOR +      ↪
215         RPINAME + COMMUNICATIONSEPARATOR + FORMATSTRING, GetHostName(),      ↪
216         LOCATION, GetComponent()));
217 }
```

```
218     private async void WaitForData(StreamSocket socket)
219     {
220         DataReader dataReader = new DataReader(socket.InputStream);
221         dataReader.InputStreamOptions = InputStreamOptions.Partial;
222         var msglength = dataReader.UnconsumedBufferLength;
223         uint stringBytes = msglength;
224
225
226         try
227         {
228             // Read modification in the stream
229             stringBytes = await dataReader.LoadAsync(MESSAGE_FULL_LENGTH);
230
231             // read message
232             string msg = dataReader.ReadString(stringBytes);
233
234             // Send in return if the value exist
235             if (msg != "")
236             {
237                 this.MPiFace.SetValue(msg);
238             }
239         }
240         catch (Exception e)
241         {
242             string output = e.Message;
243
244             if (msglength < 1)
245                 return;
246         }
247
248         // Restart loop to wait data
249         WaitForData(socket);
250     }
251
252     /// <summary>
253     /// Get the ip of the raspberry
254     /// </summary>
255     /// <returns>return a string like 192.168.1.2</returns>
256     private string GetHostName()
257     {
258         List<string> IpAddress = new List<string>();
259         var Hosts =
260             Windows.Networking.Connectivity.NetworkInformation.GetHostNames
261             ().ToList();
262         foreach (var Host in Hosts)
263         {
264             string IP = Host.DisplayName;
265             IpAddress.Add(IP);
266         }
267         return IpAddress.Last();
268     }
269
270     /// <summary>
271     /// Get component in the list of components
272     /// </summary>
273     /// <returns> return a usable string for the connection on the
```

```
    server</returns>
272     private string GetComponent()
273     {
274         string result = "";
275         int cnt = 0;
276         foreach (var component in this.MPiFace.Components)
277         {
278             // Get the name of the class
279             result += component.ToString().Split('.').Last();
280             cnt++;
281             // Add the component separator for the string format
282             if (cnt < this.MPiFace.Components.Count)
283                 result += ",";
284         }
285     }
286
287     return result;
288 }
289 }
290 #endregion
291 #endregion
292 }
293
294
```

```
1  /*-----*/
2  * Author      : Salvi Cyril
3  * Date        : 7th juny 2017
4  * Diploma     : RaspiHome
5  * Classroom   : T.IS-E2B
6  *
7  * Description:
8  *             RaspiHomePiFaceDigital2 is a program who use
9  *             a PiFace Digital 2, it's an electronic card who
10 *             can be use to plug electronic component. This
11 *             program use the PiFace Digital 2 to activate
12 *             light and store.
13 \*-----*/
14
15 namespace RaspiHomePiFaceDigital2
16 {
17     public abstract class Component{}
18 }
19
```

```
1  /*-----*/
2  * Author    : Salvi Cyril
3  * Date      : 7th juny 2017
4  * Diploma   : RaspiHome
5  * Classroom : T.IS-E2B
6  *
7  * Description:
8  *      RaspiHomePiFaceDigital2 is a program who use
9  *      a PiFace Digital 2, it's an electronic card who
10 *      can be use to plug electronic component. This
11 *      program use the PiFace Digital 2 to activate
12 *      light and store.
13 \*-----*/
14
15 namespace RaspiHomePiFaceDigital2
16 {
17     public class Light : Component
18     {
19         #region Fields
20         #region Constant
21         // PiFace output
22         private const byte RELAIA = PiFaceDigital2.RelayA;
23         private const byte RELAIB = PiFaceDigital2.RelayB;
24
25         // PiFace State
26         private const byte OFF = MCP23S17.Off;
27         private const byte ON = MCP23S17.On;
28         #endregion
29
30         #region Variable
31         private bool _isOn = false;
32         private bool _isOnA = false;
33         private bool _isOnB = false;
34         #endregion
35         #endregion
36
37         #region Properties
38         public bool IsOn
39         {
40             get
41             {
42                 return _isOn;
43             }
44
45             set
46             {
47                 _isOn = value;
48                 this.IsOnA = value;
49                 this.IsOnB = value;
50             }
51         }
52
53         public bool IsOnA
54         {
55             get
56             {
```

```
57             return _isOnA;
58         }
59
60         set
61     {
62             _isOnA = value;
63             if (value)
64             {
65                 // Turn ON the light
66                 MCP23S17.WritePin(RELAIA, ON);
67             }
68             else
69             {
70                 // Turn OFF the light
71                 MCP23S17.WritePin(RELAIA, OFF);
72             }
73         }
74     }
75
76     public bool IsOnB
77     {
78         get
79     {
80         return _isOnB;
81     }
82
83         set
84     {
85         _isOnB = value;
86         if (value)
87         {
88             // Turn ON the light
89             MCP23S17.WritePin(RELAIB, ON);
90         }
91         else
92         {
93             // Turn OFF the light
94             MCP23S17.WritePin(RELAIB, OFF);
95         }
96     }
97 }
98 #endregion
99
100    #region Constructor
101    #endregion
102
103    #region Methods
104    #endregion
105 }
106 }
107 }
```

```
1  /*-----*/
2  * Author    : Salvi Cyril
3  * Date      : 7th juny 2017
4  * Diploma   : RaspiHome
5  * Classroom : T.IS-E2B
6  *
7  * Description:
8  *      RaspiHomePiFaceDigital2 is a program who use
9  *      a PiFace Digital 2, it's an electronic card who
10 *      can be use to plug electronic component. This
11 *      program use the PiFace Digital 2 to activate
12 *      light and store.
13 \*-----*/
14
15 using System;
16 using System.Threading.Tasks;
17 using Windows.UI.Xaml;
18
19 namespace RaspiHomePiFaceDigital2
20 {
21     public class Store : Component
22     {
23         #region Fields
24         #region Constant
25             // PiFace output for motor
26             private const byte UP = PiFaceDigital2.LED4;
27             private const byte DOWN = PiFaceDigital2.LED3;
28
29             // PiFace State
30             private const byte OFF = MCP23S17.Off;
31             private const byte ON = MCP23S17.On;
32
33             // Max value for store (totally open)
34             private const int MAX_LEVEL = 200; // Time span total = 19seconds ↗
35             // raspberry latency)
36             // Min value for store (totally close)
37             private const int MIN_LEVEL = 0;
38
39             // Tick for timer
40             private const int TICKS = 10;
41             private const int TICK_SECOND = 1;
42             #endregion
43
44             #region Variable
45             private DispatcherTimer _dTimerUp = new DispatcherTimer();
46             private DispatcherTimer _dTimerDown = new DispatcherTimer();
47
48             private bool _isUp = false;
49             private bool _isDown = false;
50             private bool _isOpen = false;
51             private bool _isClose = false;
52             private bool _isStop = false;
53
54             private int _counterStopped = 0;
55             #endregion
56             #endregion
```

```
56
57     #region Properties
58     public bool IsUp
59     {
60         get
61         {
62             return _isUp;
63         }
64
65         set
66         {
67             _isUp = value;
68
69             // Maximum level
70             if (value && this.CounterStopped < MAX_LEVEL)
71             {
72                 this.SetLevel("IsUp");
73             }
74         }
75     }
76
77     public bool IsDown
78     {
79         get
80         {
81             return _isDown;
82         }
83
84         set
85         {
86             _isDown = value;
87
88             // Minimum level
89             if (value && this.CounterStopped > MIN_LEVEL)
90             {
91                 this.SetLevel("IsDown");
92             }
93         }
94     }
95
96     public bool IsOpen
97     {
98         get
99         {
100            return _isOpen;
101        }
102
103        set
104        {
105            _isOpen = value;
106
107            if (value)
108            {
109                this.SetLevel("IsOpen");
110            }
111        }
112    }
113}
```

```
112     }
113
114     public bool IsClose
115     {
116         get
117         {
118             return _isClose;
119         }
120
121         set
122         {
123             _isClose = value;
124
125             if (value)
126             {
127                 this.SetLevel("IsClose");
128             }
129         }
130     }
131
132     public bool IsStop
133     {
134         get
135         {
136             return _isStop;
137         }
138
139         set
140         {
141             _isStop = value;
142
143             // Stop everything
144             if (value)
145             {
146                 this._dTimerUp.Stop();
147                 this._dTimerDown.Stop();
148                 SetLevel("IsStop");
149                 this.IsStop = false;
150             }
151         }
152     }
153
154     public int CounterStopped
155     {
156         get
157         {
158             return _counterStopped;
159         }
160
161         set
162         {
163             _counterStopped = value;
164
165             // Store manager
166             if (value == MAX_LEVEL)
167             {
```

```
168             this._dTimerUp.Stop();
169             SetLevel("IsStop");
170             _counterStopped = MAX_LEVEL;
171         }
172         else if (value == MIN_LEVEL)
173     {
174             this._dTimerDown.Stop();
175             SetLevel("IsStop");
176             _counterStopped = MIN_LEVEL;
177         }
178     }
179 }
180 #endregion
181
182 #region Constructor
183 public Store()
184 {
185     this._dTimerUp.Interval = new TimeSpan(TICKS);
186     this._dTimerUp.Tick += _dTimerUp_Tick;
187
188     this._dTimerDown.Interval = new TimeSpan(TICKS);
189     this._dTimerDown.Tick += _dTimerDown_Tick;
190 }
191
192 private void _dTimerUp_Tick(object sender, object e)
193 {
194     this.CounterStopped++;
195 }
196
197 private void _dTimerDown_Tick(object sender, object e)
198 {
199     this.CounterStopped--;
200 }
201 #endregion
202
203 #region Methods
204 /// <summary>
205 /// Set the level
206 /// </summary>
207 /// <param name="propertyName"></param>
208 private async void SetLevel(string propertyName)
209 {
210     switch (propertyName)
211     {
212         case "IsUp":
213             this.IsDown = false;
214
215             MCP23S17.WritePin(DOWN, OFF);
216             MCP23S17.WritePin(UP, ON);
217
218             this.SetLevelUp();
219             break;
220         case "IsDown":
221             this.IsUp = false;
222
223             MCP23S17.WritePin(UP, OFF);
```

```
224             MCP23S17.WritePin(DOWN, ON);
225
226             this.SetLevelDown();
227             break;
228         case "IsOpen":
229             this.Close = false;
230
231             this.SetLevel("IsUp");
232             await Task.Delay(TimeSpan.FromSeconds(TICK_SECOND));      ↵
233
234             this.SetLevel("IsStop");
235             break;
236         case "IsClose":
237             this.Open = false;
238
239             this.SetLevel("IsDown");
240             await Task.Delay(TimeSpan.FromSeconds(TICK_SECOND));
241             this.SetLevel("IsStop");
242             break;
243         case "IsStop":
244             this.Up = false;
245             this.Down = false;
246             this.Open = false;
247             this.Close = false;
248
249             MCP23S17.WritePin(UP, OFF);
250             MCP23S17.WritePin(DOWN, OFF);
251             break;
252     }
253
254     /// <summary>
255     /// Set upper the level of the store
256     /// </summary>
257     private void SetLevelUp()
258     {
259         this._dTimmerDown.Stop();
260         this._dTimmerUp.Start();
261     }
262
263     /// <summary>
264     /// Set downer the level of the store
265     /// </summary>
266     private void SetLevelDown()
267     {
268         this._dTimmerUp.Stop();
269         this._dTimmerDown.Start();
270     }
271     #endregion
272 }
273 }
274 }
```

```
1 <Page  
2     x:Class="RaspiHomePiFaceDigital2.ViewPiFaceDigital2"  
3     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
4     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
5     xmlns:local="using:RaspiHomePiFaceDigital2"  
6     xmlns:d="http://schemas.microsoft.com/expressionblend/2008"  
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  
8     mc:Ignorable="d">  
9 </Page>  
10
```

```
1  /*-----*/
2  * Author      : Salvi Cyril
3  * Date        : 7th juny 2017
4  * Diploma     : RaspiHome
5  * Classroom   : T.IS-E2B
6  *
7  * Description:
8  *             RaspiHomePiFaceDigital2 is a program who use
9  *             a PiFace Digital 2, it's an electronic card who
10 *             can be use to plug electronic component. This
11 *             program use the PiFace Digital 2 to activate
12 *             light and store.
13 \*-----*/
14
15 using Windows.UI.Xaml.Controls;
16
17 // Pour plus d'informations sur le modèle d'élément Page vierge, consultez la ↵
18 // page http://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409
19 namespace RaspiHomePiFaceDigital2
20 {
21     /// <summary>
22     /// Une page vide peut être utilisée seule ou constituer une page de      ↵
23     /// destination au sein d'un frame.
24     public sealed partial class ViewPiFaceDigital2 : Page
25     {
26         #region Fields
27         #region Constants
28         #endregion
29
30         #region Variables
31         private ModelPiFaceDigital2 _mPiFace;
32         #endregion
33         #endregion
34
35         #region Properties
36         public ModelPiFaceDigital2 MPiFace
37         {
38             get
39             {
40                 return _mPiFace;
41             }
42
43             set
44             {
45                 _mPiFace = value;
46             }
47         }
48         #endregion
49
50         #region Constructors
51         /// <summary>
52         /// Construcor: Initializer
53         /// </summary>
54         public ViewPiFaceDigital2()
```

```
55         {
56             this.InitializeComponent();
57
58             this.MPiFace = new ModelPiFaceDigital2(this);
59         }
60         #endregion
61
62         #region Methods
63         #endregion
64     }
65 }
66
```

```

1  /*-----*/
2  * Author      : Salvi Cyril
3  * Date        : 7th juny 2017
4  * Diploma     : RaspiHome
5  * Classroom   : T.IS-E2B
6  *
7  * Description:
8  *      RaspiHomePiFaceDigital2 is a program who use
9  *      a PiFace Digital 2, it's an electronic card who
10 *      can be use to plug electronic component. This
11 *      program use the PiFace Digital 2 to activate
12 *      light and store.
13 \*-----*/
14
15 namespace RaspiHomePiFaceDigital2
16 {
17     public class PiFaceDigital2
18     {
19         // Output
20         public const byte LED0 = 0x08;          // I/O Direction Register
21         public const byte LED1 = 0x09;          // 1 = Input (default), 0 = Output
22         public const byte LED2 = 0x0A;          // MCP23x17 Input Polarity Register
23         public const byte LED3 = 0x0B;          // 0 = Normal (default)(low reads as ↵
24             0), 1 = Inverted (low reads as 1)
25         public const byte LED4 = 0x0C;          // MCP23x17 Interrupt on Change Pin ↵
26             Assignements
27         public const byte LED5 = 0x0D;          // 1 = Input (default), 0 = Output
28         public const byte LED6 = 0x0E;          // MCP23x17 Input Polarity Register
29         public const byte LED7 = 0x0F;          // 0 = Normal (default)(low reads as ↵
30             0), 1 = Inverted (low reads as 1)
31
32         // Input
33         public const byte IN0 = 0x00;           // I/O Direction Register
34         public const byte IN1 = 0x01;           // 1 = Input (default), 0 = Output
35         public const byte IN2 = 0x02;           // MCP23x17 Input Polarity Register
36         public const byte IN3 = 0x03;           // 0 = Normal (default)(low reads as ↵
37             0), 1 = Inverted (low reads as 1)
38         public const byte IN4 = 0x04;           // MCP23x17 Interrupt on Change Pin ↵
39             Assignements
40         public const byte IN5 = 0x05;           // 1 = Input (default), 0 = Output
41         public const byte IN6 = 0x06;           // MCP23x17 Input Polarity Register
42         public const byte IN7 = 0x07;           // 0 = Normal (default)(low reads as ↵
43             0), 1 = Inverted (low reads as 1)
44
45         // Switch / Button
46         public const byte Sw0 = IN0;           // I/O Direction Register
47         public const byte Sw1 = IN1;           // 1 = Input (default), 0 = Output
48         public const byte Sw2 = IN2;           // MCP23x17 Input Polarity Register
49         public const byte Sw3 = IN3;           // 0 = Normal (default)(low reads as ↵
50             0), 1 = Inverted (low reads as 1)
51
52         // Relay
53         public const byte RelayA = LED1;       // MCP23x17 Input Polarity
54             Register
55         public const byte RelayB = LED0;       // 0 = Normal (default)(low reads ↵
56             as 0), 1 = Inverted (low reads as 1)

```

```
48     }
49 }
50
```

```
1  /*-----*/
2  * Author    : Salvi Cyril
3  * Date      : 7th juny 2017
4  * Diploma   : RaspiHome
5  * Classroom : T.IS-E2B
6  *
7  * Description:
8  *      RaspiHomePiFaceDigital2 is a program who use
9  *      a PiFace Digital 2, it's an electronic card who
10 *      can be use to plug electronic component. This
11 *      program use the PiFace Digital 2 to activate
12 *      light and store.
13 \*-----*/
14
15 using System;
16 using System.Diagnostics;
17 using System.Threading.Tasks;
18 using Windows.Devices.Enumeration;
19 using Windows.Devices.Spi;
20
21 namespace RaspiHomePiFaceDigital2
22 {
23     public class MCP23S17
24     {
25
26         private const byte IODIRA = 0x00;      // I/O Direction Register
27         private const byte IODIRB = 0x01;      // 1 = Input (default), 0 = Output
28         private const byte IPOLA = 0x02;        // MCP23x17 Input Polarity Register
29         private const byte IPOLB = 0x03;        // 0 = Normal (default)(low reads as 0), 1 = Inverted (low reads as 1)
30         private const byte GPINTENA = 0x04;      // MCP23x17 Interrupt on Change Pin Assignements
31         private const byte GPINTENB = 0x05;      // 0 = No Interrupt on Change (default), 1 = Interrupt on Change
32         private const byte DEFVALA = 0x06;        // MCP23x17 Default Compare Register for Interrupt on Change
33         private const byte DEFVALB = 0x07;        // Opposite of what is here will trigger an interrupt (default = 0)
34         private const byte INTCONA = 0x08;        // MCP23x17 Interrupt on Change Control Register
35         private const byte INTCONB = 0x09;        // 1 = pin is compared to DEFVAL, 0 = pin is compared to previous state (default)
36         private const byte IOCONA = 0x0A;        // MCP23x17 Configuration Register
37         private const byte IOCONB = 0x0B;        // Also Configuration Register
38         private const byte GPPUA = 0x0C;        // MCP23x17 Weak Pull-Up Resistor Register
39         private const byte GPPUB = 0x0D;        // INPUT ONLY: 0 = No Internal 100k Pull-Up (default) 1 = Internal 100k Pull-Up
40         private const byte INTFA = 0x0E;        // MCP23x17 Interrupt Flag Register
41         private const byte INTFB = 0x0F;        // READ ONLY: 1 = This Pin Triggered the Interrupt
```

```

...\\RaspiHomePiFaceDigital2\\PiFace initializer\\MCP23S17.cs 2
42     private const byte INTCAPA = 0x10;      // MCP23x17 Interrupt Captured    ↵
        Value for Port Register
43     private const byte INTCAPB = 0x11;      // READ ONLY: State of the Pin    ↵
        at the Time the Interrupt Occurred
44     private const byte GPIOA = 0x12;       // MCP23x17 GPIO Port Register
45     private const byte GPIOB = 0x13;       // Value on the Port - Writing    ↵
        Sets Bits in the Output Latch
46     private const byte OLATA = 0x14;       // MCP23x17 Output Latch          ↵
        Register
47     private const byte OLATB = 0x15;       // 1 = Latch High, 0 = Latch Low    ↵
        (default) Reading Returns Latch State, Not Port Value!
48
49     public const byte On = 1;
50     public const byte Off = 0;
51     public const byte Output = 0;
52     public const byte Input = 1;
53
54     private const byte Address = 0x00;      // offset address if hardware    ↵
        addressing is on and is 0 - 7 (A0 - A2)
55     private const byte BaseAddW = 0x40;     // MCP23S17 Write base address
56     private const byte BaseAddR = 0x41;     // MCP23S17 Read Base Address
57     private const byte HAEN = 0x08;        // IOCON register for MCP23S17, x08    ↵
        enables hardware address so sent address must match hardware pins    ↵
        A0-A2
58
59
60     private static UInt16 PinMode = 0xFFFF;   // default Pinmode for the    ↵
        MXP23S17 set to inputs
61     private static UInt16 PullUpMode = 0xFFFF;  // default pullups for    ↵
        the MXP23S17 set to weak pullup
62     private static UInt16 InversionMode = 0X0000; // default invert to    ↵
        normal
63     private static UInt16 PinState = 0X0000;   // default pinstate to    ↵
        all 0's
64
65     /*RaspBerry Pi2 Parameters*/
66     private const string SPI_CONTROLLER_NAME = "SPI0"; /* For Raspberry    ↵
        Pi 2, use SPI0
67     private const Int32 SPI_CHIP_SELECT_LINE = 0;        /* Line 0 maps to    ↵
        physical pin number 24 on the Rpi2, line 1 to pin 26
68
69     private static byte[] readBuffer3 = new byte[3]; /*this is defined to    ↵
        hold the output data*/
70     private static byte[] readBuffer4 = new byte[4]; /*this is defined to    ↵
        hold the output data*/
71     private static byte[] writeBuffer3 = new byte[3];//register, then 16    ↵
        bit value
72     private static byte[] writeBuffer4 = new byte[4];//register, then 16    ↵
        bit value
73
74     private static SpiDevice SpiGPIO;
75     public static async Task InitializeSPI()
76     {
77         try
78         {
79             var settings = new SpiConnectionSettings

```

```
    (SPI_CHIP_SELECT_LINE);
80    settings.ClockFrequency = 1000000; // 10000000;
81    settings.Mode = SpiMode.Mode0; //Mode0,1,2,3; MCP23S17 needs ↵
     mode 0
82
83    string spiAqs = SpiDevice.GetDeviceSelector ↵
     (SPI_CONTROLLER_NAME);
84    var deviceInfo = await DeviceInformation.FindAllAsync(spiAqs);
85    SpiGPIO = await SpiDevice.FromIdAsync(deviceInfo[0].Id, ↵
     settings);
86}
87
88/* If initialization fails, display the exception and stop running ↵
 */
89catch (Exception ex)
90{
91    Debug.WriteLine(ex.Message);
92    //statusText.Text = "\nSPI Initialization Failed";
93}
94}
95
96public static void InitializeMCP23S17()
97{
98    WriteRegister8(IOCONA, HAEN); // enable the ↵
     hardware address incase there is more than one chip
99    WriteRegister16(IODIRA, PinMode); // Set the ↵
     default or current pin mode
100}
101
102public static void WriteRegister8(byte register, byte value)
103{
104    // Direct port manipulation speeds taking Slave Select LOW before ↵
     SPI action
105    writeBuffer3[0] = (BaseAddW | (Address << 1));
106    writeBuffer3[1] = register;
107    writeBuffer3[2] = value;
108    try
109    {
110        SpiGPIO.Write(writeBuffer3);
111    }
112
113    /* If initialization fails, display the exception and stop running ↵
 */
114    catch (Exception ex)
115    {
116        Debug.WriteLine(ex.Message);
117        //statusText.Text = "\nFailed to Wrie to DAC";
118    } // Send the byte
119}
120
121public static void WriteRegister16(byte register, UInt16 value)
122{
123    writeBuffer4[0] = (BaseAddW | (Address << 1));
124    writeBuffer4[1] = register;
125    writeBuffer4[2] = (byte)(value >> 8);
126    writeBuffer4[3] = (byte)(value & 0xFF);
127    try
```

```
127         {
128             SpiGPIO.Write(writeBuffer4);
129         }
130
131         /* If initialization fails, display the exception and stop running ↵
132            */
133         catch (Exception ex)
134         {
135             Debug.WriteLine(ex.Message);
136             //statusText.Text = "\nFailed to Wrie to DAC";
137         }
138     }
139
140     // Set the pin mode a pin at a time or all 16 in one go
141     // any value other than Input is taken as output
142     public static void setPinMode(byte pin, byte mode)
143     {
144         if (pin > 15) return;           // only a 16bit port so do a    ↵
145             bounds check, it cant be less than zero as this is a byte value
146         if (mode == Input)
147         {
148             PinMode |= (UInt16)(1 << (pin));           // update the    ↵
149                 pinMode register with new direction
150         }
151         else
152         {
153             PinMode &= (UInt16)(~(1 << (pin)));        // update the    ↵
154                 pinMode register with new direction
155         }
156         WriteRegister16(IODIRA, PinMode);           // Call the      ↵
157             generic word writer with start register and the mode cache
158     }
159
160     // Set the pullup a pin at a time or all 16 in one go
161     // any value other than On is taken as off
162     public static void pullupMode(byte pin, byte mode)
163     {
164         if (pin > 15) return;
165         if (mode == On)
166         {
167             PullUpMode |= (UInt16)(1 << (pin));
168         }
169         else
170         {
171             PullUpMode &= (UInt16)(~(1 << (pin)));
172         }
173         WriteRegister16(GPPUA, PullUpMode);
174     }
175     public static void PullupMode(UInt16 mode)
176     {
177         WriteRegister16(GPPUA, mode);
```

```
178         PullUpMode = mode;
179     }
180
181     // Set the inversion a pin at a time or all 16 in one go
182     public static void InvertMode(byte pin, byte mode)
183     {
184         if (pin > 15) return;
185         if (mode == On)
186         {
187             InversionMode |= (UInt16)(1 << (pin - 1));
188         }
189         else
190         {
191             InversionMode &= (UInt16)(~(1 << (pin - 1)));
192         }
193         WriteRegister16(IPOLA, InversionMode);
194     }
195     public static void InvertMode(UInt16 mode)
196     {
197         WriteRegister16(IPOLA, mode);
198         InversionMode = mode;
199     }
200
201     // WRITE FUNCTIONS - BY WORD AND BY PIN
202
203     public static void WritePin(byte pin, byte value)
204     {
205         if (pin > 15) return;
206         if (value > 1) return;
207         if (value == 1)
208         {
209             PinState |= (UInt16)(1 << pin);
210         }
211         else
212         {
213             PinState &= (UInt16)(~(1 << pin));
214         }
215         WriteRegister16(GPIOA, PinState);
216     }
217     public static void WriteWord(UInt16 value)
218     {
219         WriteRegister16(GPIOA, value);
220         PinState = value;
221     }
222
223     // READ FUNCTIONS - BY WORD, BYTE AND BY PIN
224     public static UInt16 ReadRegister16()
225     {
226         writeBuffer4[0] = (BaseAddR | (Address << 1));
227         writeBuffer4[1] = GPIOA;
228         writeBuffer4[2] = 0;
229         writeBuffer4[3] = 0;
230         SpiGPIO.TransferFullDuplex(writeBuffer4, readBuffer4);
231         return convertToInt(readBuffer4);                                // ↗
232         Return the constructed word, the format is 0x(register value)
233     }
```

```
233     public static byte ReadRegister8(byte register)
234     {           // This function will read a single register, and return it
235         writeBuffer3[0] = (BaseAddrR | (Address << 1)); // Send the      ↵
236             MCP23S17 opcode, chip address, and read bit
237         writeBuffer3[1] = register;
238         SpiGPIO.TransferFullDuplex(writeBuffer3, readBuffer3);
239         return readBuffer4[2]; // convertToInt
240             (readBuffer);           // Return the
241                 constructed word, the format is 0x(register value)
242     }
243     public static UInt16 ReadPin(byte pin)
244     {
245         if (pin > 15) return 0x00;           // If the pin value is ↵
246             not valid (1-16) return, do nothing and return
247         UInt16 value = ReadRegister16();           //      ↵
248             Initialize a variable to hold the read values to be returned
249         UInt16 pinmask = (UInt16)(1 << pin);           //      ↵
250             Initialize a variable to hold the read values to be returned
251         return ((value & pinmask) > 0) ? On : Off; // Call the word      ↵
252             reading function, extract HIGH/LOW information from the
253                 requested pin
254     }
255
256     private static UInt16 convertToInt(byte[] data)
257     {
258         // byte[0] = command, byte[1] register, byte[2] = data high, byte ↵
259             [3] = data low
260         UInt16 result = (UInt16)(data[2] & 0xFF);
261         result <= 8;
262         result += data[3];
263         return result;
264     }
265 }
```