```
 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          public CommunicationWithServer ComWithServer
59          {
60              get
61              {
62                  return _comWithServer;
63              }
64
65              set
66              {
67                  _comWithServer = value;
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 Synthetizer(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         /// Initialze 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 0
114             spiSettings.ClockFrequency = 3600000; //3.6 MHz
115             spiSettings.Mode = SpiMode.Mode0;
116
117             string spiQuery = SpiDevice.GetDeviceSelector("SPI0");
118
119             var deviceInfo = await DeviceInformation.FindAllAsync(spiQuery);
120             if (deviceInfo != null && deviceInfo.Count > 0)
121             {
122                 _mcp3202 = await SpiDevice.FromIdAsync(deviceInfo[0].Id, spiSettings);
123             }
124         }
125
126         /// <summary>
127         /// Read digital input with SPI
128         /// </summary>
129         /// <returns></returns>
130         private string ReadSpiData()
131         {
132             byte[] transmitBuffer = new byte[3] { 0x01, 0x80, 0 };
133             byte[] receiveBuffer = new byte[3];
134
135             string result = "";
136
137             _mcp3202.TransferFullDuplex(transmitBuffer, receiveBuffer);
138
139             return result = Encoding.UTF8.GetString(receiveBuffer);
140         }
141
142         /// <summary>
143         /// (obsolete on UWP) Set the configuration of the speecher
144         /// </summary>
145         private void InitializeSpeechRecognizer()
146         {
147
148         }
149
150         /// <summary>
151         /// (obsolete on UWP) Enable the speech, used when raspi is not 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
              unknow)
192         /// </summary>
193         /// <param name="messageReply"></param>
194         /// <param name="messageCommand"></param>
195         public void ReplyForSynthetize(string messageReply, string
              messageCommand)
196         {
197             if (messageReply == "ERROR_MESSAGE")
198             {
199                 this.RhSynt.WrongCommand();
200             }
201             else
202             {
203                 this.RhSynt.RaspiSayInformation
                      (this.RhSynt.SetProprelyInformations(messageReply,
                      messageCommand));
204             }
205             this.IsRaspiCalled = false;
206 
207             EnableSpeech();
208         }
209         #endregion
210     }
211 }
212 
```

```
 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 server
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
171             this.RaspiTalk(messageToSay);
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
                  synth.SynthesizeTextToStreamAsync(messageToSay);
197
198             mediaElement.SetSource(synthStream, synthStream.ContentType);
199             // 0 = min / 1 = max
200             mediaElement.Volume = 1;
201             mediaElement.Play();
202
203             // Work like Thread.Sleep(TIME_TO_WAIT)
204             await Task.Delay(TimeSpan.FromSeconds(TIME_TO_WAIT));
205         }
206
207         /// <summary>
208         /// Called when there is any error
209         /// </summary>
210         public void WrongCommand()
211         {
212             // Reach the error resquest sentences to say
213             this.RaspiCalled(this.RhCommands.SpeecherRespondingRequestError);
214         }
215
216         /// <summary>
217         /// Allow the Raspi, to let her talk with list of information
218         /// </summary>
219         /// <param name="informationsToGive"></param>
220         public void RaspiSayInformation(List<string> informationsToGive)
221         {
```

```
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,
              string messageCommand)
235         {
236             List<string> result = new List<string>();
237 
238             bool temp = false, humi = false, pres = false;
239 
240             if (messageCommand.Contains("température"))
241                 temp = true;
242             if (messageCommand.Contains("humidité"))
243                 humi = true;
244             if (messageCommand.Contains("pression"))
245                 pres = true;
246             if (messageCommand.Contains("état"))
247             {
248                 temp = true;
249                 humi = true;
250                 pres = true;
251             }
252 
253             string[] informationSplited = messageReply.Split(';');
254             if (informationSplited[0] != "")
255             {
256                 foreach (string information in informationSplited)
257                 {
258                     switch (information.Split('=').First())
259                     {
260                         case "TEMP":
261                             if (temp)
262                             {
263                                 result.Add
                        (this.RhCommands.SpeecherRespondingEtatRequest[0] +
264                                     information.Split('=').Last() +
265 
    this.RhCommands.SpeecherRespondingEtatRequest[1]);
266                             }
267                             break;
268                         case "HUMI":
269                             if (humi)
270                             {
271                                 result.Add
                        (this.RhCommands.SpeecherRespondingEtatRequest[4] +
272                                     information.Split('=').Last() +
273 
```

```
    this.RhCommands.SpeecherRespondingEtatRequest[5]);
                            }
                            break;
                        case "PRES":
                            if (pres)
                            {
                                result.Add
                    (this.RhCommands.SpeecherRespondingEtatRequest[6] +
                                    information.Split('=').Last() +
    this.RhCommands.SpeecherRespondingEtatRequest[7]);
                            }
                            break;
                    }
                }
            }
            else
                result.Add("");
            return result;
        }

        /// <summary>
        /// Stack Overflow solution to delete accents in strings
        /// http://stackoverflow.com/questions/249087/how-do-i-remove-
          diacritics-accents-from-a-string-in-net
        /// </summary>
        /// <param name="str"></param>
        /// <returns></returns>
        static string RemoveDiacritics(string str)
        {
            var normalizedString = str.Normalize(NormalizationForm.FormD);
            var stringBuilder = new StringBuilder();

            foreach (var c in normalizedString)
            {
                var unicodeCategory = CharUnicodeInfo.GetUnicodeCategory(c);
                if (unicodeCategory != UnicodeCategory.NonSpacingMark)
                {
                    stringBuilder.Append(c);
                }
            }

            return stringBuilder.ToString().Normalize
              (NormalizationForm.FormC);
        }
        #endregion
    }
}

```

```
using RaspiHomeSpeechNSynthetize;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using Windows.ApplicationModel.Core;
using Windows.Networking;
using Windows.Networking.Connectivity;
using Windows.Networking.Sockets;
using Windows.Storage.Streams;

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

        private const string FORMATSTRING = "IPRasp={0};Location=
          {1};Component={2}";
        private const string COMMUNICATIONSEPARATOR = "@";

        // Important need to be changed if it's another room!
        private const string LOCATION = "Salon";
        private const string COMPONENT = "Microphone";
        private const string RPINAME = "Microphone_" + LOCATION;//
          "192.168.2.8";

        private const int MESSAGE_FULL_LENGHT = 512;
        #endregion

        #region Variables
        private Speecher _speecher;

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

        private string _messageCommand = "";
        #endregion
        #endregion

        #region Properties
        public Speecher Speecher
        {
```

```
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
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         #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),
                      PORT.ToString());
168                 SendForInitialize();
169                 this.Connecting = false;
170                 this.IsConnected = true;
171
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,
              StreamSocketListenerConnectionReceivedEventArgs args)
191         {
192             this.Connections.Add(args.Socket);
193
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
                  string length.
206             dataWriter.WriteInt32((int)len);
207             dataWriter.WriteString(message);
208             var ret = await dataWriter.StoreAsync();
209             dataWriter.DetachStream();
210         }
211
212         /// <summary>
213         /// Send to initialize the raspberry to the server
214         /// </summary>
215         private void SendForInitialize()
216         {
217             SendMessage(this.Socket, string.Format(COMMUNICATIONSEPARATOR +
                    RPINAME + COMMUNICATIONSEPARATOR + "Connection:" + FORMATSTRING,
                     GetHostName(), LOCATION, COMPONENT));
```

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

```
<Page
    x:Class="RaspiHomeSpeechNSynthetize.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomeSpeechNSynthetize"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <StackPanel Orientation="Horizontal" Background="White"
      HorizontalAlignment="Center" VerticalAlignment="Center">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center"
          Margin="0,0,50,0">
            <StackPanel Orientation="Horizontal">
                <TextBox x:Name="tbxLightCommand1" Width="250" Text="Allume la
                  lumière du salon"/>
                <Button x:Name="btnLightCommand1" Width="55" Content="Send"
                  Margin="40,0,0,0" Click="btnLightCommand1_Click"/>
            </StackPanel>

            <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
                <TextBox x:Name="tbxLightCommand2" Width="250" Text="Éteins la
                  lumière du salon"/>
                <Button x:Name="btnLightCommand2" Width="55" Content="Send"
                  Margin="40,0,0,0" Click="btnLightCommand2_Click"/>
            </StackPanel>

            <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
                <TextBox x:Name="tbxState" Width="250" Text="Quel est la
                  température du salon"/>
                <Button x:Name="btnState" Width="55" Content="Send"
                  Margin="40,0,0,0" Click="btnState_Click"/>
            </StackPanel>
        </StackPanel>

        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center"
          Margin="0,0,50,0">
            <StackPanel Orientation="Horizontal">
                <TextBox x:Name="tbxStore1" Width="250" Text="Monte le store du
                   salon"/>
                <Button x:Name="btnSendCommand" Width="55" Content="Send"
                  Margin="40,0,0,0" Click="btnStore1_Click"/>
            </StackPanel>

            <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
                <TextBox x:Name="tbxStore2" Width="250" Text="Descends le store
                   du salon"/>
                <Button x:Name="btnStore2" Width="55" Content="Send"
                  Margin="40,0,0,0" Click="btnStore2_Click"/>
            </StackPanel>

            <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
                <TextBox x:Name="tbxStore3" Width="250" Text="Stop le store du
                  salon"/>
                <Button x:Name="btnStore3" Width="55" Content="Send"
                  Margin="40,0,0,0" Click="btnStore3_Click"/>
```

```
            </StackPanel>
        </StackPanel>
    </StackPanel>
</Page>

```

```
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
     page http://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409
17
18 namespace RaspiHomeSpeechNSynthetize
19 {
20     /// <summary>
21     /// Une page vide peut être utilisée seule ou constituer une page de
        destination au sein d'un frame.
22     /// </summary>
23     public sealed partial class MainPage : Page
24     {
25         #region Fields
26         private Speecher _speecher;
27         #endregion
28
29         #region Constuctor
30         public MainPage()
31         {
32             this.InitializeComponent();
33
34             this._speecher = new Speecher();
35         }
36         #endregion
37
38         #region Event
39         private void btnLightCommand1_Click(object sender, RoutedEventArgs e)
40         {
41             if (this.tbxLightCommand1.Text != "")
42                 this._speecher.SendBrutCommand(this.tbxLightCommand1.Text);
43             else
44                 this._speecher.SendBrutCommand("Allumer lumiere");
45         }
46
47         private void btnLightCommand2_Click(object sender, RoutedEventArgs e)
48         {
49             if (this.tbxLightCommand2.Text != "")
50                 this._speecher.SendBrutCommand(this.tbxLightCommand2.Text);
51             else
52                 this._speecher.SendBrutCommand("Eteindre lumiere");
53         }
54
```

```
55          private void btnState_Click(object sender, RoutedEventArgs e)
56          {
57              if (this.tbxState.Text != "")
58                  this._speecher.SendBrutCommand(this.tbxState.Text);
59              else
60                  this._speecher.SendBrutCommand("Temperature du salon");
61          }
62
63          private void btnStore1_Click(object sender, RoutedEventArgs e)
64          {
65              if (this.tbxStore1.Text != "")
66                  this._speecher.SendBrutCommand(this.tbxStore1.Text);
67              else
68                  this._speecher.SendBrutCommand("Monter store");
69          }
70
71          private void btnStore2_Click(object sender, RoutedEventArgs e)
72          {
73              if (this.tbxStore2.Text != "")
74                  this._speecher.SendBrutCommand(this.tbxStore2.Text);
75              else
76                  this._speecher.SendBrutCommand("Descendre store");
77          }
78
79          private void btnStore3_Click(object sender, RoutedEventArgs e)
80          {
81              if (this.tbxStore3.Text != "")
82                  this._speecher.SendBrutCommand(this.tbxStore3.Text);
83              else
84                  this._speecher.SendBrutCommand("Stopper store");
85          }
86          #endregion
87      }
88  }
89
```

```
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
                 servir?","Je suis toute ouïe"
11         };
12
13         private List<string> _whenCallingError = new List<string>()
14         {
15             "Je ne m'appel pas comme ça!","Je crois que vous vous êtes trompé
                 de personne!"
16         };
17
18         private List<string> _speecherRespondingRequest = new List<string>()
19         {
20             "Tout de suite","Je m'y mets de ce pas!","Ne quittez pas!"
21         };
22
23         private List<string> _speecherRespondingRequestError = new
             List<string>()
24         {
25             "Je n'ai pas compris ce que vous avez demandé"
26         };
27
28         private List<string> _raspiHomeObjectKnown = new List<string>()
29         {
30             "lumiere","lumieres",
31             "store","stores",
32             "television","televisions",
33             "porte","portes",
34             "fenetre","fenetres",
35         };
36
37         private List<string> _raspiHomeActionKnown = new List<string>()
38         {
39             "allumer","allume",
40             "eteindre","eteins",
41             "monter","monte",
42             "descendre","descends",
43             "stopper","stop",
44             "ouvrir","ouvre",
45             "fermer","ferme",
46         };
47
48         private List<string> _speecherRespondingEtatRequest = new List<string>
             ()
49         {
50             "Il fait "," degrés Celsius "," degrés Farad"," degrés Kelvin","Le
                 taux d'humidité est de "," pourcent","La pression est
                 actuellement de "," hecto Pascal"
```

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

```
 94
 95          public List<string> WhenCallingError
 96          {
 97              get
 98              {
 99                  return _whenCallingError;
100              }
101
102              set
103              {
104                  _whenCallingError = value;
105              }
106          }
107
108          public List<string> SpeecherRespondingRequest
109          {
110              get
111              {
112                  return _speecherRespondingRequest;
113              }
114
115              set
116              {
117                  _speecherRespondingRequest = value;
118              }
119          }
120
121          public List<string> SpeecherRespondingRequestError
122          {
123              get
124              {
125                  return _speecherRespondingRequestError;
126              }
127
128              set
129              {
130                  _speecherRespondingRequestError = value;
131              }
132          }
133
134          public List<string> RaspiHomeObjectKnown
135          {
136              get
137              {
138                  return _raspiHomeObjectKnown;
139              }
140
141              set
142              {
143                  _raspiHomeObjectKnown = value;
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                 _raspiHomeCommandUselessConnecter = 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          public CommunicationWithServer ComWithServer
59          {
60              get
61              {
62                  return _comWithServer;
63              }
64
65              set
66              {
67                  _comWithServer = value;
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 Synthetizer(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         /// Initialze 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 0
114             spiSettings.ClockFrequency = 3600000; //3.6 MHz
115             spiSettings.Mode = SpiMode.Mode0;
116
117             string spiQuery = SpiDevice.GetDeviceSelector("SPI0");
118
119             var deviceInfo = await DeviceInformation.FindAllAsync(spiQuery);
120             if (deviceInfo != null && deviceInfo.Count > 0)
121             {
122                 _mcp3202 = await SpiDevice.FromIdAsync(deviceInfo[0].Id, spiSettings);
123             }
124         }
125
126         /// <summary>
127         /// Read digital input with SPI
128         /// </summary>
129         /// <returns></returns>
130         private string ReadSpiData()
131         {
132             byte[] transmitBuffer = new byte[3] { 0x01, 0x80, 0 };
133             byte[] receiveBuffer = new byte[3];
134
135             string result = "";
136
137             _mcp3202.TransferFullDuplex(transmitBuffer, receiveBuffer);
138
139             return result = Encoding.UTF8.GetString(receiveBuffer);
140         }
141
142         /// <summary>
143         /// (obsolete on UWP) Set the configuration of the speecher
144         /// </summary>
145         private void InitializeSpeechRecognizer()
146         {
147
148         }
149
150         /// <summary>
151         /// (obsolete on UWP) Enable the speech, used when raspi is not 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
              unknow)
192         /// </summary>
193         /// <param name="messageReply"></param>
194         /// <param name="messageCommand"></param>
195         public void ReplyForSynthetize(string messageReply, string
              messageCommand)
196         {
197             if (messageReply == "ERROR_MESSAGE")
198             {
199                 this.RhSynt.WrongCommand();
200             }
201             else
202             {
203                 this.RhSynt.RaspiSayInformation
                      (this.RhSynt.SetProprelyInformations(messageReply,
                      messageCommand));
204             }
205             this.IsRaspiCalled = false;
206 
207             EnableSpeech();
208         }
209         #endregion
210     }
211 }
212 
```

```
 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 server
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
171             this.RaspiTalk(messageToSay);
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
                  synth.SynthesizeTextToStreamAsync(messageToSay);
197
198             mediaElement.SetSource(synthStream, synthStream.ContentType);
199             // 0 = min / 1 = max
200             mediaElement.Volume = 1;
201             mediaElement.Play();
202
203             // Work like Thread.Sleep(TIME_TO_WAIT)
204             await Task.Delay(TimeSpan.FromSeconds(TIME_TO_WAIT));
205         }
206
207         /// <summary>
208         /// Called when there is any error
209         /// </summary>
210         public void WrongCommand()
211         {
212             // Reach the error resquest sentences to say
213             this.RaspiCalled(this.RhCommands.SpeecherRespondingRequestError);
214         }
215
216         /// <summary>
217         /// Allow the Raspi, to let her talk with list of information
218         /// </summary>
219         /// <param name="informationsToGive"></param>
220         public void RaspiSayInformation(List<string> informationsToGive)
221         {
```

```
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,
              string messageCommand)
235         {
236             List<string> result = new List<string>();
237 
238             bool temp = false, humi = false, pres = false;
239 
240             if (messageCommand.Contains("température"))
241                 temp = true;
242             if (messageCommand.Contains("humidité"))
243                 humi = true;
244             if (messageCommand.Contains("pression"))
245                 pres = true;
246             if (messageCommand.Contains("état"))
247             {
248                 temp = true;
249                 humi = true;
250                 pres = true;
251             }
252 
253             string[] informationSplited = messageReply.Split(';');
254             if (informationSplited[0] != "")
255             {
256                 foreach (string information in informationSplited)
257                 {
258                     switch (information.Split('=').First())
259                     {
260                         case "TEMP":
261                             if (temp)
262                             {
263                                 result.Add
                        (this.RhCommands.SpeecherRespondingEtatRequest[0] +
264                                     information.Split('=').Last() +
265 
    this.RhCommands.SpeecherRespondingEtatRequest[1]);
266                             }
267                             break;
268                         case "HUMI":
269                             if (humi)
270                             {
271                                 result.Add
                        (this.RhCommands.SpeecherRespondingEtatRequest[4] +
272                                     information.Split('=').Last() +
273 
```

```
    this.RhCommands.SpeecherRespondingEtatRequest[5]);
                            }
                            break;
                        case "PRES":
                            if (pres)
                            {
                                result.Add
                    (this.RhCommands.SpeecherRespondingEtatRequest[6] +
                                    information.Split('=').Last() +
    this.RhCommands.SpeecherRespondingEtatRequest[7]);
                            }
                            break;
                    }
                }
            }
            else
                result.Add("");
            return result;
        }

        /// <summary>
        /// Stack Overflow solution to delete accents in strings
        /// http://stackoverflow.com/questions/249087/how-do-i-remove-
          diacritics-accents-from-a-string-in-net
        /// </summary>
        /// <param name="str"></param>
        /// <returns></returns>
        static string RemoveDiacritics(string str)
        {
            var normalizedString = str.Normalize(NormalizationForm.FormD);
            var stringBuilder = new StringBuilder();

            foreach (var c in normalizedString)
            {
                var unicodeCategory = CharUnicodeInfo.GetUnicodeCategory(c);
                if (unicodeCategory != UnicodeCategory.NonSpacingMark)
                {
                    stringBuilder.Append(c);
                }
            }

            return stringBuilder.ToString().Normalize
              (NormalizationForm.FormC);
        }
        #endregion
    }
}

```

```
using RaspiHomeSpeechNSynthetize;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using Windows.ApplicationModel.Core;
using Windows.Networking;
using Windows.Networking.Connectivity;
using Windows.Networking.Sockets;
using Windows.Storage.Streams;

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

        private const string FORMATSTRING = "IPRasp={0};Location=
          {1};Component={2}";
        private const string COMMUNICATIONSEPARATOR = "@";

        // Important need to be changed if it's another room!
        private const string LOCATION = "Salon";
        private const string COMPONENT = "Microphone";
        private const string RPINAME = "Microphone_" + LOCATION;//
          "192.168.2.8";

        private const int MESSAGE_FULL_LENGHT = 512;
        #endregion

        #region Variables
        private Speecher _speecher;

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

        private string _messageCommand = "";
        #endregion
        #endregion

        #region Properties
        public Speecher Speecher
        {
```

```
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
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         #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),
                      PORT.ToString());
168                 SendForInitialize();
169                 this.Connecting = false;
170                 this.IsConnected = true;
171
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,
              StreamSocketListenerConnectionReceivedEventArgs args)
191         {
192             this.Connections.Add(args.Socket);
193
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
                  string length.
206             dataWriter.WriteInt32((int)len);
207             dataWriter.WriteString(message);
208             var ret = await dataWriter.StoreAsync();
209             dataWriter.DetachStream();
210         }
211
212         /// <summary>
213         /// Send to initialize the raspberry to the server
214         /// </summary>
215         private void SendForInitialize()
216         {
217             SendMessage(this.Socket, string.Format(COMMUNICATIONSEPARATOR +
                    RPINAME + COMMUNICATIONSEPARATOR + "Connection:" + FORMATSTRING,
                     GetHostName(), LOCATION, COMPONENT));
```

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

```
<Page
    x:Class="RaspiHomeSpeechNSynthetize.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomeSpeechNSynthetize"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <StackPanel Orientation="Horizontal" Background="White"
      HorizontalAlignment="Center" VerticalAlignment="Center">
        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center"
          Margin="0,0,50,0">
            <StackPanel Orientation="Horizontal">
                <TextBox x:Name="tbxLightCommand1" Width="250" Text="Allume la
                  lumière du salon"/>
                <Button x:Name="btnLightCommand1" Width="55" Content="Send"
                  Margin="40,0,0,0" Click="btnLightCommand1_Click"/>
            </StackPanel>

            <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
                <TextBox x:Name="tbxLightCommand2" Width="250" Text="Éteins la
                  lumière du salon"/>
                <Button x:Name="btnLightCommand2" Width="55" Content="Send"
                  Margin="40,0,0,0" Click="btnLightCommand2_Click"/>
            </StackPanel>

            <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
                <TextBox x:Name="tbxState" Width="250" Text="Quel est la
                  température du salon"/>
                <Button x:Name="btnState" Width="55" Content="Send"
                  Margin="40,0,0,0" Click="btnState_Click"/>
            </StackPanel>
        </StackPanel>

        <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center"
          Margin="0,0,50,0">
            <StackPanel Orientation="Horizontal">
                <TextBox x:Name="tbxStore1" Width="250" Text="Monte le store du
                   salon"/>
                <Button x:Name="btnSendCommand" Width="55" Content="Send"
                  Margin="40,0,0,0" Click="btnStore1_Click"/>
            </StackPanel>

            <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
                <TextBox x:Name="tbxStore2" Width="250" Text="Descends le store
                   du salon"/>
                <Button x:Name="btnStore2" Width="55" Content="Send"
                  Margin="40,0,0,0" Click="btnStore2_Click"/>
            </StackPanel>

            <StackPanel Orientation="Horizontal" Margin="0,40,0,0">
                <TextBox x:Name="tbxStore3" Width="250" Text="Stop le store du
                  salon"/>
                <Button x:Name="btnStore3" Width="55" Content="Send"
                  Margin="40,0,0,0" Click="btnStore3_Click"/>
```

```
            </StackPanel>
        </StackPanel>
    </StackPanel>
</Page>

```

```
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
     page http://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409
17
18 namespace RaspiHomeSpeechNSynthetize
19 {
20     /// <summary>
21     /// Une page vide peut être utilisée seule ou constituer une page de
        destination au sein d'un frame.
22     /// </summary>
23     public sealed partial class MainPage : Page
24     {
25         #region Fields
26         private Speecher _speecher;
27         #endregion
28
29         #region Constuctor
30         public MainPage()
31         {
32             this.InitializeComponent();
33
34             this._speecher = new Speecher();
35         }
36         #endregion
37
38         #region Event
39         private void btnLightCommand1_Click(object sender, RoutedEventArgs e)
40         {
41             if (this.tbxLightCommand1.Text != "")
42                 this._speecher.SendBrutCommand(this.tbxLightCommand1.Text);
43             else
44                 this._speecher.SendBrutCommand("Allumer lumiere");
45         }
46
47         private void btnLightCommand2_Click(object sender, RoutedEventArgs e)
48         {
49             if (this.tbxLightCommand2.Text != "")
50                 this._speecher.SendBrutCommand(this.tbxLightCommand2.Text);
51             else
52                 this._speecher.SendBrutCommand("Eteindre lumiere");
53         }
54
```

```
55          private void btnState_Click(object sender, RoutedEventArgs e)
56          {
57              if (this.tbxState.Text != "")
58                  this._speecher.SendBrutCommand(this.tbxState.Text);
59              else
60                  this._speecher.SendBrutCommand("Temperature du salon");
61          }
62
63          private void btnStore1_Click(object sender, RoutedEventArgs e)
64          {
65              if (this.tbxStore1.Text != "")
66                  this._speecher.SendBrutCommand(this.tbxStore1.Text);
67              else
68                  this._speecher.SendBrutCommand("Monter store");
69          }
70
71          private void btnStore2_Click(object sender, RoutedEventArgs e)
72          {
73              if (this.tbxStore2.Text != "")
74                  this._speecher.SendBrutCommand(this.tbxStore2.Text);
75              else
76                  this._speecher.SendBrutCommand("Descendre store");
77          }
78
79          private void btnStore3_Click(object sender, RoutedEventArgs e)
80          {
81              if (this.tbxStore3.Text != "")
82                  this._speecher.SendBrutCommand(this.tbxStore3.Text);
83              else
84                  this._speecher.SendBrutCommand("Stopper store");
85          }
86          #endregion
87      }
88  }
89
```

```
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
                 servir?","Je suis toute ouïe"
11         };
12
13         private List<string> _whenCallingError = new List<string>()
14         {
15             "Je ne m'appel pas comme ça!","Je crois que vous vous êtes trompé
                 de personne!"
16         };
17
18         private List<string> _speecherRespondingRequest = new List<string>()
19         {
20             "Tout de suite","Je m'y mets de ce pas!","Ne quittez pas!"
21         };
22
23         private List<string> _speecherRespondingRequestError = new
             List<string>()
24         {
25             "Je n'ai pas compris ce que vous avez demandé"
26         };
27
28         private List<string> _raspiHomeObjectKnown = new List<string>()
29         {
30             "lumiere","lumieres",
31             "store","stores",
32             "television","televisions",
33             "porte","portes",
34             "fenetre","fenetres",
35         };
36
37         private List<string> _raspiHomeActionKnown = new List<string>()
38         {
39             "allumer","allume",
40             "eteindre","eteins",
41             "monter","monte",
42             "descendre","descends",
43             "stopper","stop",
44             "ouvrir","ouvre",
45             "fermer","ferme",
46         };
47
48         private List<string> _speecherRespondingEtatRequest = new List<string>
             ()
49         {
50             "Il fait "," degrés Celsius "," degrés Farad"," degrés Kelvin","Le
                 taux d'humidité est de "," pourcent","La pression est
                 actuellement de "," hecto Pascal"
```

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

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

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar version="1.0" xml:lang="fr-FR" mode="voice" root="toplevel"
tag-format="semantics/1.0"
  xml:base="http://www.contoso.com/"
  xmlns="http://www.w3.org/2001/06/grammar"
  xmlns:sapi="http://schemas.microsoft.com/Speech/2002/06/SRGSExtensions">
  <rule id="toplevel" scope="public">
    <one-of>
      <item>
        <ruleref uri="#Raspi"/>
      </item>
      <item>
        <ruleref uri="#Meteo"/>
      </item>
      <item>
        <ruleref uri="#Action"/>
      </item>
    </one-of>
  </rule>

  <rule id="Raspi">
    <example> raspi </example>

    <item> raspi </item>
  </rule>

  <rule id="Meteo">
    <example> quel est la temperature </example>
    <example> quel est le taux d'humidite </example>
    <example> quel est le niveau de pression </example>
    <example> quel est l'etat</example>

    <one-of>
      <item> temperature </item>
      <item> humidite </item>
      <item> pression </item>
      <item> etat </item>
    </one-of>
    <ruleref uri="#Localisation"/>
  </rule>

  <rule id="Action">
    <example> allume la lumiere</example>
    <example> eteint la lumiere</example>
    <example> monte le store</example>
    <example> descend le store</example>
    <example> ouvre le store</example>
    <example> ferme le store</example>

    <one-of>
      <item> allume </item>
      <item> eteint </item>
      <item> monte </item>
      <item> descend </item>
      <item> ouvre </item>
      <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 =
              Windows.Storage.ApplicationData.Current.LocalSettings;
30          #endregion
31          #endregion
32
33          #region Properties
34          public MenuView View
35          {
36              get
37              {
38                  return _view;
39              }
40
41              set
42              {
43                  _view = value;
44              }
45          }
46
47          public CommunicationWithServer ComWithServer
48          {
49              get
50              {
51                  return _comWithServer;
52              }
53
54              set
55              {
```

```
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
```

```
<Page
    x:Class="RaspiHomeTabletWindows.Menu.MenuView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomeTabletWindows.Menu"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="10"/>
            <ColumnDefinition/>
            <ColumnDefinition Width="10"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="10"/>
            <RowDefinition/>
            <RowDefinition Height="10"/>
        </Grid.RowDefinitions>

        <Grid Grid.Column="1" Grid.Row="1">
            <Grid.RowDefinitions>
                <RowDefinition Height="*" MinHeight="160"/>
                <RowDefinition Height="5*"/>
            </Grid.RowDefinitions>
            <!--TOOLBAR-->
            <Rectangle Fill="WhiteSmoke" Grid.Row="0"/>
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                    <ColumnDefinition Width="200"/>
                </Grid.ColumnDefinitions>
                <!--BUTTON TOOLBAR-->
                <StackPanel x:Name="stkMenuToolbar" Orientation="Horizontal"
                  Grid.Column="0"/>
                <!--SOME INFORMATION-->
                <Grid Grid.Column="1">
                    <TextBlock Text="RaspiHome" HorizontalAlignment="Left"
                    VerticalAlignment="Center" FontSize="30" Margin="10,0,0,0"/
                    >
                </Grid>
            </Grid>

            <!--FRAME EVENT WITH TOOLBAR BUTTON-->
            <Grid x:Name="grdFrame" Grid.Row="1" Background="White">
                <Frame x:Name="frmMenu"/>
            </Grid>
        </Grid>
    </Grid>
</Page>

```

```
/*-----------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-----------------------------------------------*/

using RaspiHomeTabletWindows.Menu.MenuToolbar;
using RaspiHomeTabletWindows.Modules.GlobalSetup;
using RaspiHomeTabletWindows.Modules.Home;
using RaspiHomeTabletWindows.Modules.Information;
using RaspiHomeTabletWindows.Modules.Setting;
using System;
using System.Collections.Generic;
using System.Linq;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

// The User Control item template is documented at http://go.microsoft.com/
  fwlink/?LinkId=234236

namespace RaspiHomeTabletWindows.Menu
{
    public sealed partial class MenuView : Page
    {
        #region Fields
        #region Constants
        #endregion

        #region Variables
        private MenuModel _model = null;

        private ToolbarButtonView _btnToolbarView = null;

        private string _frameAlreadyChoose = "";
        private Dictionary<string, Dictionary<string, string>>
          _buttonInformation = new Dictionary<string, Dictionary<string,
          string>>() {
            { "Home", new Dictionary<string,string>() { { "Retourner à
              l'accueil" , "Home.png" } } },
            { "Global setup", new Dictionary<string,string>() { { "Visualiser
              l'ensemble des modules", "GlobalSetup.png" } } },
            { "Information", new Dictionary<string,string>() { { "Regarder les
               information du système", "Information.png" } } },
            { "Setting", new Dictionary<string,string>() { { "Parametrage de
              l'application", "Setting.png" } } }
        };

        private List<string> _listChoise = null;
```

```
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                 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
                    [keyInfo].Keys.FirstOrDefault()]));
188                 this.LstChoise.Add(keyInfo);
189             }
190         }
191 
192         private void UpdateMenuToolbar()
193         {
194             //foreach (ToolbarButtonView t in this.stkMenuToolbar.Children)
195             //    t._click -= MenuToolbarButton_Click;
196             this.stkMenuToolbar.Children.Clear();
197             this.LstToolbarButton.Clear();
198             foreach (ToolbarButtonData t in this.LstToolbarButtonData)
199             {
200                 this._btnToolbarView = new ToolbarButtonView(t.FrameChoose,
                  t.Description, t.IconLink);
201                 this._btnToolbarView.Tag = t;
202                 this._btnToolbarView._click += MenuToolbarButton_Click;
203                 this.stkMenuToolbar.Children.Add(this._btnToolbarView);
204                 this.LstToolbarButton.Add(this._btnToolbarView);
205             }
206         }
207         #endregion
208 
209     }
210 }
211 
```

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-------------------------------------------------*/

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

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

        private const string FORMATSTRING = "IPRasp={0};Location=
          {1};Component={2}";
        private const string COMMUNICATIONSEPARATOR = "@";

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

        private const int MESSAGE_FULL_LENGHT = 512;
        #endregion

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

        private Windows.Storage.ApplicationDataContainer localSettings =
    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
                      ["SendMessageToServer"];
180                 this.SendCommandToServer(messageToSend.ToString());
181                 localSettings.Values.Remove("SendMessageToServer");
182             }
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),
                      PORT.ToString());
196                 SendForInitialize();
197                 this.Connecting = false;
198                 this.IsConnected = true;
199
200                 WaitForData(this.Socket);
201             }
202             catch (Exception)
203             {
204                 this.Connecting = false;
205                 this.IsConnected = false;
206             }
207         }
208
209         /// <summary>
210         /// Listen the traffic on the port
211         /// </summary>
212         private async void Listen()
213         {
214             this.Listener.ConnectionReceived += listenerConnectionReceived;
215             await this.Listener.BindServiceNameAsync(PORT.ToString());
216         }
217
218         void listenerConnectionReceived(StreamSocketListener sender,
              StreamSocketListenerConnectionReceivedEventArgs args)
219         {
220             this.Connections.Add(args.Socket);
```

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

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

```
/*-----------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-----------------------------------------------*/

using Windows.UI.Xaml.Media.Imaging;

namespace RaspiHomeTabletWindows.Menu.MenuToolbar
{
    public class ToolbarButtonModel : PropertyChangedBase
    {
        #region Fields
        #region Constants
        #endregion

        #region Variables
        private ToolbarButtonView _view = null;

        private string _description = "";
        private string _folderIconName = "";
        private string _iconLink = null;
        private BitmapImage imgSource = null;
        #endregion
        #endregion

        #region Properties
        public ToolbarButtonView View
        {
            get
            {
                return _view;
            }

            set
            {
                _view = value;
            }
        }

        public string Description
        {
            get
            {
                return _description;
            }

            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
```

```
<UserControl
    x:Class="RaspiHomeTabletWindows.Menu.MenuToolbar.ToolbarButtonView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomeTabletWindows.Menu.MenuToolbar"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Height="150" Width="180">
    <UserControl.Resources>

    </UserControl.Resources>

    <Button x:Name="btnToolbar" Height="150" Margin="10,0,20,0"
      Style="{StaticResource styleRoundButton}" Background="Gray"
      ToolTipService.ToolTip="{Binding Path=ToolTipMessage}" Width="{Binding
      Height, ElementName=btnToolbar}" Click="btnToolbar_Click">
        <Image x:Name="imgMenuToolbarButton" Height="100" Width="100"/>
    </Button>
</UserControl>

```

```
/*-----------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-----------------------------------------------*/

using System;
using Windows.UI;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Media.Imaging;

// The User Control item template is documented at http://go.microsoft.com/
  fwlink/?LinkId=234236

namespace RaspiHomeTabletWindows.Menu.MenuToolbar
{
    public sealed partial class ToolbarButtonView : UserControl
    {
        #region Fields
        #region Constants
        #endregion

        #region Variables
        private ToolbarButtonModel _model = null;

        public event EventHandler _click;

        private bool _isSelected = false;
        private bool _isPressed = false;

        private string _whoseButtonClicked = "";
        #endregion
        #endregion

        #region Properties
        public ToolbarButtonModel Model
        {
            get
            {
                return _model;
            }

            set
            {
                _model = value;
            }
```

```
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
                        (Color.FromArgb(255, 73, 130, 5));
75                  }
76                  else
77                  {
78                      btnToolbar.BorderThickness = new Thickness(5);
79                      btnToolbar.BorderBrush = new SolidColorBrush
                        (Color.FromArgb(255, 76, 74, 75));
80                  }
81              }
82          }
83
84          public bool IsPressed
85          {
86              get
87              {
88                  return _isPressed;
89              }
90
91              set
92              {
93                  _isPressed = value;
94              }
95          }
96
97          public string WhoseButtonClicked
98          {
99              get
100             {
101                 return _whoseButtonClicked;
102             }
103
104             set
105             {
106                 _whoseButtonClicked = value;
107             }
108         }
109         #endregion
```

```
110
111         #region Constructor
112         /// <summary>
113         /// Constructor: Initializer
114         /// </summary>
115         public ToolbarButtonView(string frameChoose, string description,
              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
                      (iconPath));
160         }
161         #endregion
162     }
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  }
```

```
<Page
    x:Class="RaspiHomeTabletWindows.Modules.Home.HomeView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomeTabletWindows.Modules.Home"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="50"/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <StackPanel x:Name="stkLocationButton" Orientation="Horizontal"
          Grid.Row="0"/>
        <Frame x:Name="frmHome" Grid.Row="1"/>
    </Grid>
</Page>

```

```
/*-----------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-----------------------------------------------*/

using RaspiHomeTabletWindows.Menu.LocationButton;
using System;
using System.Collections.Generic;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

// The User Control item template is documented at http://go.microsoft.com/
  fwlink/?LinkId=234236

namespace RaspiHomeTabletWindows.Modules.Home
{
    public sealed partial class HomeView : Page
    {

        #region Fields
        #region Constants
        #endregion

        #region Varaibles
        private HomeModel _model = null;

        private LocationButtonView _btnLocationButtonView = null;

        private string _frameAlreadyChoose = "";
        private List<string> _buttonInformation = new List<string>() {
            "Maison", "Salon", "Cuisine",
        };

        private List<string> _listChoise = null;

        private List<LocationButtonData> _lstLocationButtonData = null;
        private List<LocationButtonView> _lstLocationButton = null;

        private Windows.Storage.ApplicationDataContainer localSettings =
          Windows.Storage.ApplicationData.Current.LocalSettings;

        #endregion
        #endregion

        #region Properties
        public HomeModel Model
        {
```

```
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
112             set
113             {
114                 _lstLocationButton = value;
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)
                  sender).WhoseButtonClicked;
154
155             localSettings.Values["NameButtonClicked"] = buttonClicked;
156
157             var actualFrameChoose = localSettings.Values["NameButtonClicked"];
158
159             switch (buttonClicked)
160             {
161                 case "Maison":
162                     if ((actualFrameChoose.ToString() != "Maison") ||
                        (actualFrameChoose == null))
163                         localSettings.Values.Remove("NameButtonClicked");
164
```

```
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") ||
                        (actualFrameChoose == null))
171                         localSettings.Values.Remove("NameButtonClicked");
172 
173                     localSettings.Values["NameButtonClicked"] = buttonClicked;
174                     this.frmHome.Content = null;
175                     this.frmHome.Navigate(typeof
                        (Location.OtherRoom.RoomView));
176                     break;
177                 case "Cuisine":
178                     if ((actualFrameChoose.ToString() != "Cuisine") ||
                        (actualFrameChoose == null))
179                         localSettings.Values.Remove("NameButtonClicked");
180 
181                     localSettings.Values["NameButtonClicked"] = buttonClicked;
182                     this.frmHome.Content = null;
183                     this.frmHome.Navigate(typeof
                        (Location.OtherRoom.RoomView));
184                     break;
185                 case "Bureau":
186                     if ((actualFrameChoose.ToString() != "Bureau") ||
                        (actualFrameChoose == null))
187                         localSettings.Values.Remove("NameButtonClicked");
188 
189                     localSettings.Values["NameButtonClicked"] = buttonClicked;
190                     this.frmHome.Content = null;
191                     this.frmHome.Navigate(typeof
                        (Location.OtherRoom.RoomView));
192                     break;
193             }
194 
195             ((LocationButtonView)sender).IsSelected = true;
196         }
197         #endregion
198 
199         #region Methods
200         /// <summary>
201         /// Initialize the button on the toolbar
202         /// </summary>
203         private void InitializeLocationButton()
204         {
205             this.LstToolbarButtonData = new List<LocationButtonData>();
206             this.LstToolbarButton = new List<LocationButtonView>();
207             this.LstChoise = new List<string>();
208 
209             foreach (var buttonName in this._buttonInformation)
210             {
211                 this.LstToolbarButtonData.Add(new LocationButtonData
                      (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
                      (t.FrameChoose, t.Description);
226                 this._btnLocationButtonView.Tag = t;
227                 this._btnLocationButtonView._click += _btnToolbarView__click;
228                 this.stkLocationButton.Children.Add
                      (this._btnLocationButtonView);
229                 this.LstToolbarButton.Add(this._btnLocationButtonView);
230             }
231         }
232         #endregion
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          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
58          public bool IsSelected
59          {
60              get
61              {
62                  return _isSelected;
63              }
64
65              set
66              {
67                  _isSelected = value;
68                  this.IsSelected = value;
69              }
70          }
71          #endregion
72
73          #region Constructor
74          /// <summary>
75          /// Constructor: Initializer
76          /// </summary>
77          public LocationButtonData(string frameChoose, string description)
78          {
79              this.FrameChoose = frameChoose;
80              this.Description = description;
81          }
82          #endregion
83
84          #region Methods
85          #endregion
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"
         Margin="10,0,20,0" Background="{StaticResource DefaultBackgroundColor}"
         Click="btnButtonLocation_Click">
12         <TextBlock x:Name="tblLocationName" Text="{Binding Path=LocationName}"
             Foreground="White" FontSize="28"/>
13     </Button>
14 </UserControl>
15
```

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-------------------------------------------------*/

using System;
using Windows.UI;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Media;

// The User Control item template is documented at http://go.microsoft.com/
  fwlink/?LinkId=234236

namespace RaspiHomeTabletWindows.Menu.LocationButton
{
    public sealed partial class LocationButtonView : UserControl
    {
        #region Fields
        #region Constants
        #endregion

        #region Variables
        private LocationButtonModel _model = null;

        public event EventHandler _click;

        private bool _isSelected = false;
        private bool _isPressed = false;

        private string _whoseButtonClicked = "";
        #endregion
        #endregion

        #region Properties

        public LocationButtonModel Model
        {
            get
            {
                return _model;
            }

            set
            {
                _model = value;
            }
```

```
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
                        (Color.FromArgb(255, 73, 130, 5));
75                      tblLocationName.Foreground = new SolidColorBrush
                        (Color.FromArgb(255, 73, 130, 5));
76                  }
77                  else
78                  {
79                      btnButtonLocation.BorderThickness = new Thickness(1);
80                      btnButtonLocation.BorderBrush = new SolidColorBrush
                        (Color.FromArgb(255, 76, 74, 75));
81                      tblLocationName.Foreground = new SolidColorBrush
                        (Color.FromArgb(255, 255, 255, 255));
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                 }
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 
```

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-------------------------------------------------*/

namespace RaspiHomeTabletWindows.Modules.Home.Location.House
{
    public class RoomModel
    {
        #region Fields
        #region Constants
        #endregion

        #region Varaibles
        private RoomView _view = null;

        Windows.Storage.ApplicationDataContainer localSettings =
          Windows.Storage.ApplicationData.Current.LocalSettings;
        #endregion
        #endregion

        #region Properties
        public RoomView View
        {
            get
            {
                return _view;
            }

            set
            {
                _view = value;
            }
        }
        #endregion

        #region Constructor
        /// <summary>
        /// Constructor: Initializer
        /// </summary>
        public RoomModel(RoomView paramView)
        {
            this.View = paramView;
        }
        #endregion

        #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 + " " +
                  component;
64          }
65          #endregion
66      }
67  }
68
```

```
<Page
    x:Class="RaspiHomeTabletWindows.Modules.Home.Location.House.RoomView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomeTabletWindows.Modules.Home.Location.House"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <RelativePanel x:Name="rpnlDisplay" Margin="10">
        <!--LIGHT BUTTON-->
        <Rectangle x:Name="rect1" Width="17"/>
        <TextBlock x:Name="tblLight" Text="Lumière"
          RelativePanel.RightOf="rect1"  RelativePanel.AlignTopWithPanel="True"
           FontSize="26"/>
        <Button x:Name="btnLightOnOff" RelativePanel.Below="tblLight"
          Height="120" Width="120"  Background="{StaticResource
          DefaultBackgroundColor}" HorizontalContentAlignment="Center"
          VerticalContentAlignment="Center" Click="btnLightOnOff_Click">
            <Image x:Name="imgLightButton" Source="ms-appx:///Icon/bulb.png"/>
        </Button>

        <!--STORE BUTTON-->
        <!--UP-->
        <Rectangle x:Name="rect2" Width="20"
          RelativePanel.AlignRightWithPanel="True"/>
        <TextBlock x:Name="tblUp" Text="Monter" RelativePanel.LeftOf="rect2"
          RelativePanel.AlignTopWithPanel="True" FontSize="26"/>
        <Button x:Name="btnStoreUp" RelativePanel.AlignRightWithPanel="True"
          RelativePanel.Below="tblUp" Height="120" Width="120"
          Background="{StaticResource DefaultBackgroundColor}"
          HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
           Click="btnStoreUp_Click">
            <Image Source="ms-appx:///Icon/arrowUp.png"/>
        </Button>

        <Rectangle x:Name="rect3" Height="20" Width="120"
          RelativePanel.AlignRightWithPanel="True"
          RelativePanel.Below="btnStoreUp"/>

        <!--DOWN-->
        <Button x:Name="btnStoreDown" RelativePanel.AlignRightWithPanel="True"
          RelativePanel.Below="rect3" Height="120" Width="120"
          Background="{StaticResource DefaultBackgroundColor}"
          HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
           Click="btnStoreDown_Click">
            <Image Source="ms-appx:///Icon/arrowDown.png"/>
        </Button>
        <TextBlock x:Name="tblDown" Text="Descendre"
          RelativePanel.AlignHorizontalCenterWith="tblUp"
          RelativePanel.Below="btnStoreDown" FontSize="26"/>

        <!--OPEN-->
        <TextBlock x:Name="tblOpen" Text="Ouvrir"
          RelativePanel.AlignHorizontalCenterWith="tblUp"
          RelativePanel.Above="btnStoreOpen" FontSize="26"/>
```

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

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-------------------------------------------------*/

using System;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Media.Imaging;

// The User Control item template is documented at http://go.microsoft.com/
  fwlink/?LinkId=234236

namespace RaspiHomeTabletWindows.Modules.Home.Location.House
{
    public sealed partial class RoomView : Page
    {
        #region Fields
        #region Constants
        #endregion

        #region Varaibles
        private RoomModel _model = null;

        private bool _isOn = false;
        private bool _isUp = false;
        private bool _isDown = false;
        private bool _isOpen = false;
        private bool _isClose = false;
        #endregion
        #endregion

        #region Properties
        public RoomModel Model
        {
            get
            {
                return _model;
            }

            set
            {
                _model = value;
            }
        }

        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-
                        appx:///Icon/bulbLighting.png"));
72                  }
73                  else
74                  {
75                      // Send message to save
76                      this.Model.SendMessage("eteindre", "lumiere");
77                      // Change the picture
78                      this.imgLightButton.Source = new BitmapImage(new Uri("ms-
                        appx:///Icon/bulb.png"));
79                  }
80              }
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
```

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-------------------------------------------------*/

using System;
using System.Linq;
using System.Threading.Tasks;
using Windows.UI.Xaml;

namespace RaspiHomeTabletWindows.Modules.Home.Location.OtherRoom
{
    public class RoomModel
    {
        #region Fields
        #region Constants
        #endregion

        #region Varaibles
        private RoomView _view = null;

        private string _messageReaded = "";

        DispatcherTimer _dTimer = null;

        Windows.Storage.ApplicationDataContainer localSettings =
          Windows.Storage.ApplicationData.Current.LocalSettings;
        #endregion
        #endregion

        #region Properties
        public RoomView View
        {
            get
            {
                return _view;
            }

            set
            {
                _view = value;
            }
        }

        public string MessageReceive
        {
            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
                      ["ReceiveMessageFromServer"];
89                  this.MessageReceive = messageToSend.ToString();
90                  UpDateView();
91                  localSettings.Values.Remove("ReceiveMessageFromServer");
92
93                  this._dTimer.Stop();
94              }
95          }
96          #endregion
97
98          #region Events
99          /// <summary>
100         /// Initialize at the start (check if the sense hat exist)
101         /// </summary>
102         private void InitializeState()
103         {
104             // Update state room
105             var locationName = localSettings.Values["NameButtonClicked"];
106
107             if (locationName != null)
108                 localSettings.Values["SendMessageToServer"] = "etat " +
                      locationName;
109             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 + " " +
                      component + " " + locationName;
126         }
127
128         /// <summary>
129         /// Update state values on the view
130         /// </summary>
131         private async void UpDateView()
132         {
133             await Task.Delay(TimeSpan.FromMilliseconds(200));
134
135             this.View.EnableDisplayState();
136
137             var informations = this.MessageReceive.Split(';');
138             foreach (var info in informations)
139             {
140                 switch (info.Split('=').First())
141                 {
142                     case "TEMP":
143                         this.View.StrTemp = info.Split('=').Last();
144                         break;
145                     case "HUMI":
146                         this.View.StrHumi = info.Split('=').Last();
147                         break;
148                     case "PRES":
149                         this.View.StrPres = info.Split('=').Last();
150                         break;
151                 }
152             }
153         }
154         #endregion
155     }
156 }
157
```

```
<Page
    x:Class="RaspiHomeTabletWindows.Modules.Home.Location.OtherRoom.RoomView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomeTabletWindows.Modules.Home.Location.OtherRoom"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <RelativePanel x:Name="rpnlDisplay" Margin="10">
        <!--LIGHT BUTTON-->
        <Rectangle x:Name="rectTextLight" Width="17"/>
        <TextBlock x:Name="tblLight" Text="Lumière"
          RelativePanel.RightOf="rectTextLight"
          RelativePanel.AlignTopWithPanel="True" FontSize="26"/>
        <Button x:Name="btnLightOnOff" RelativePanel.Below="tblLight"
          Height="120" Width="120"  Background="{StaticResource
          DefaultBackgroundColor}" HorizontalContentAlignment="Center"
          VerticalContentAlignment="Center" Click="btnLightOnOff_Click">
            <Image x:Name="imgLightButton" Source="ms-appx:///Icon/bulb.png"/>
        </Button>

        <!--STATE OF THE ROOM-->
        <!--HUMIDITY-->
        <Rectangle x:Name="rectHumid" RelativePanel.AlignBottomWithPanel="True"
           Height="34"/>
        <Rectangle x:Name="rectInformation" RelativePanel.Above="rectHumid"
          Width="75"/>
        <Image x:Name="imgHumidity" Visibility="Collapsed" Source="ms-appx:///
          Icon/Humidity.png " RelativePanel.AlignLeftWithPanel="True"
          RelativePanel.Above="rectHumid" Height="75"/>

        <!--TEMPERATURE-->
        <Rectangle x:Name="rectThermo" RelativePanel.Above="imgHumidity"
          Height="34"/>
        <Image x:Name="imgThermometer" Visibility="Collapsed" Source="ms-
          appx:///Icon/Thermometer.png "
          RelativePanel.AlignHorizontalCenterWith="imgHumidity"
          RelativePanel.Above="rectThermo" Height="75"/>

        <Rectangle x:Name="rectPressure" RelativePanel.Above="rectHumid"
          Width="250"/>
        <Rectangle x:Name="rectInformation2" RelativePanel.Above="rectHumid"
          Width="350"/>

        <!--PRESSURE-->
        <Image x:Name="imgBarometer" Visibility="Collapsed" Source="ms-appx:///
          Icon/Pressure.png " RelativePanel.RightOf="rectPressure"
          RelativePanel.AlignVerticalCenterWith="imgThermometer" Height="65"/>

        <TextBlock x:Name="tblTemperature"
          RelativePanel.RightOf="rectInformation"
          RelativePanel.AlignVerticalCenterWith="imgThermometer" FontSize="26"/
          >
        <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
```

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-------------------------------------------------*/

using System;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Media.Imaging;

// The User Control item template is documented at http://go.microsoft.com/
  fwlink/?LinkId=234236

namespace RaspiHomeTabletWindows.Modules.Home.Location.OtherRoom
{
    public sealed partial class RoomView : Page
    {
        #region Fields
        #region Constants
        #endregion

        #region Varaibles
        private RoomModel _model = null;

        private string _strTemp = "";
        private string _strHumi = "";
        private string _strPres = "";

        private bool _isOn = false;
        private bool _isUp = false;
        private bool _isDown = false;
        private bool _isOpen = false;
        private bool _isClose = false;
        #endregion
        #endregion

        #region Properties
        public RoomModel Model
        {
            get
            {
                return _model;
            }

            set
            {
                _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-
                    appx:///Icon/bulbLighting.png"));
121                 }
122                 else
123                 {
124                     // Save message to save
125                     this.Model.SendMessage("eteindre", "lumiere");
126                     // Change the picture
127                     this.imgLightButton.Source = new BitmapImage(new Uri("ms-
                    appx:///Icon/bulb.png"));
128                 }
129             }
130         }
131 
132         public bool IsUp
133         {
134             get
135             {
136                 return _isUp;
137             }
138 
139             set
140             {
141                 _isUp = value;
142             }
143         }
144 
145         public bool IsDown
146         {
147             get
148             {
149                 return _isDown;
150             }
151 
152             set
153             {
154                 _isDown = value;
155             }
156         }
157 
158         public bool IsOpen
159         {
160             get
161             {
162                 return _isOpen;
163             }
164 
165             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.IsClose = 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  }
54
```

```xaml
<Page
    x:Class="RaspiHomeTabletWindows.Modules.GlobalSetup.GlobalSetupView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomeTabletWindows.Modules.GlobalSetup"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid>
        <Image Source="ms-appx:///Icon/GlobalSetup.png"/>
    </Grid>
</Page>

```

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-------------------------------------------------*/

using Windows.UI.Xaml.Controls;

// The User Control item template is documented at http://go.microsoft.com/
  fwlink/?LinkId=234236

namespace RaspiHomeTabletWindows.Modules.GlobalSetup
{
    public sealed partial class GlobalSetupView : Page
    {
        #region Fields
        #region Constants
        #endregion

        #region Varaibles
        private GlobalSetupModel _model = null;
        #endregion
        #endregion

        #region Properties
        public GlobalSetupModel Model
        {
            get
            {
                return _model;
            }

            set
            {
                _model = value;
            }
        }
        #endregion

        #region Constructor
        /// <summary>
        /// Constructor: Initializer
        /// </summary>
        public GlobalSetupView()
        {
            this.InitializeComponent();

            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  }
```

```
<Page
    x:Class="RaspiHomeTabletWindows.Modules.Information.InformationView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomeTabletWindows.Modules.Information"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid>
        <Image Source="ms-appx:///Icon/Information.png" />
    </Grid>
</Page>

```

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-------------------------------------------------*/

using Windows.UI.Xaml.Controls;

// The User Control item template is documented at http://go.microsoft.com/fwlink/?LinkId=234236

namespace RaspiHomeTabletWindows.Modules.Information
{
    public sealed partial class InformationView : Page
    {
        #region Fields
        #region Constants
        #endregion

        #region Varaibles
        private InformationModel _model = null;
        #endregion
        #endregion

        #region Properties
        public InformationModel Model
        {
            get
            {
                return _model;
            }

            set
            {
                _model = value;
            }
        }
        #endregion

        #region Constructor
        /// <summary>
        /// Constructor: Initializer
        /// </summary>
        public InformationView()
        {
            this.InitializeComponent();

            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.Setting
16  {
17      public class SettingModel : PropertyChangedBase
18      {
19          #region Fields
20          #region Constants
21          #endregion
22
23          #region Varaibles
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  }
```

```xaml
<Page
    x:Class="RaspiHomeTabletWindows.Modules.Setting.SettingView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomeTabletWindows.Modules.Setting"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid>
        <Image Source="ms-appx:///Icon/Setting.png" />
    </Grid>
</Page>

```

```
/*-----------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*-----------------------------------------------*/

using Windows.UI.Xaml.Controls;

// The User Control item template is documented at http://go.microsoft.com/
  fwlink/?LinkId=234236

namespace RaspiHomeTabletWindows.Modules.Setting
{
    public sealed partial class SettingView : Page
    {
        #region Fields
        #region Constants
        #endregion

        #region Varaibles
        private SettingModel _model = null;
        #endregion
        #endregion

        #region Properties
        public SettingModel Model
        {
            get
            {
                return _model;
            }

            set
            {
                _model = value;
            }
        }
        #endregion

        #region Constructor
        /// <summary>
        /// Constructor: Initializer
        /// </summary>
        public SettingView()
        {
            this.InitializeComponent();

            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
                  relevant propertyName parameter:
26              if (PropertyChanged != null)
27              {
28                  PropertyChanged(this, new PropertyChangedEventArgs
                      (propertyName));
29              }
30          }
31      }
32  }
33
```

```
<Application
    x:Class="RaspiHomeTabletWindows.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomeTabletWindows"
    RequestedTheme="Light">
    <Application.Resources>
        <!--|   COLOR   |-->
        <Color x:Key="White">#FFFDFDFD</Color>
        <Color x:Key="LightGray">#FFEEEEEE</Color>
        <Color x:Key="Gray">#FFBCBCBC</Color>
        <Color x:Key="DarkGray">#FF2C2C2C</Color>
        <Color x:Key="ShadowGray">#FF4C4A4B</Color>
        <Color x:Key="GreenForest">#FF498205</Color>

        <!--Event color-->
        <Color x:Key="WhiteOnEvent">#80F5F5F5</Color>
        <Color x:Key="DarkGrayOnEvent">#802C2C2C</Color>
        <Color x:Key="BlueAzureOnEvent">#80498205</Color>

        <!--|   BRUSH   |-->
        <SolidColorBrush x:Key="DefaultBackColor"
                         Color="{StaticResource White}"/>

        <SolidColorBrush x:Key="DefaultBackgroundColor"
                         Color="{StaticResource ShadowGray}"/>

        <SolidColorBrush x:Key="DefaultBorderColor"
                         Color="{StaticResource DarkGray}"/>

        <SolidColorBrush x:Key="OverColor"
                         Color="{StaticResource WhiteOnEvent}"/>

        <SolidColorBrush x:Key="PressedColor"
                         Color="{StaticResource DarkGrayOnEvent}"/>
        <!--|   STYLE   |-->
        <Style x:Name="styleRoundButton" TargetType="Button">
            <Setter Property="Background" Value="{StaticResource
              DefaultBackColor}"/>
            <Setter Property="BorderBrush" Value="{StaticResource
              DefaultBackgroundColor}"/>
            <Setter Property="BorderThickness" Value="5"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border x:Name="Border"
                            Background="{TemplateBinding Background}"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}"
                            CornerRadius="70">
                            <ContentPresenter
                        HorizontalAlignment="{TemplateBinding
                        HorizontalContentAlignment}"
                                          VerticalAlignment="{TemplateBinding
                        VerticalContentAlignment}"/>
                        </Border>
```

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

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

using System;
using Emmellsoft.IoT.Rpi.SenseHat;
using Windows.UI;
using Windows.UI.Xaml;

namespace RaspiHomeSenseHAT
{
    public class ModelSenseHAT
    {
        #region Fields
        #region Constants
        #endregion

        #region Variables
        private ViewSenseHAT _vSenseHAT;
        private CommunicationWithServer _comWithServer;

        // Sense HAT librairy
        private ISenseHat _senseHat;
        private ISenseHatDisplay _senseHatDisplay;
        private SenseHatData _data;

        // Set default color matrix to OFF
        private Color _uiColor = Color.FromArgb(0, 0, 0, 0);
        #endregion
        #endregion

        #region Properties
        public ViewSenseHAT VSenseHAT
        {
            get
            {
                return _vSenseHAT;
            }

            set
            {
                _vSenseHAT = value;
            }
        }

        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              // Initilize the Sense HAT (don't need to be initialized before
                  the communication start because it's only a sensor)
94              InitializeSenseHat();
95
96              // Initilize 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) + ";"
                  + "HUMI=" + Math.Round((decimal)this.Data.Humidity) + ";" +
                  "PRES=" + Math.Round((decimal)this.Data.Pressure);
142         }
143         #endregion
144     }
145 }
146
```

```
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  namespace RaspiHomeSenseHAT
16  {
17      public class SenseHatData
18      {
19          public double? Humidity { get; set; }
20          public double? Pressure { get; set; }
21          public double? Temperature { get; set; }
22      }
23  }
24
```

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

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

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

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

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

        private const int MESSAGE_FULL_LENGHT = 512;
        #endregion

        #region Variables
        private ModelSenseHAT _mSenseHAT;

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

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

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

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

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

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

```
<Page
    x:Class="RaspiHomeSenseHAT.ViewSenseHAT"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomeSenseHAT"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
</Page>

```

```
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 Windows.UI.Xaml.Controls;
16
17  // Pour plus d'informations sur le modèle d'élément Page vierge, consultez la
      page http://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409
18
19  namespace RaspiHomeSenseHAT
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 ViewSenseHAT : Page
25      {
26          #region Fields
27          #region Constants
28          #endregion
29
30          #region Variables
31          private ModelSenseHAT _mSenseHAT;
32          #endregion
33          #endregion
34
35          #region Properties
36          public ModelSenseHAT MSenseHAT
37          {
38              get
39              {
40                  return _mSenseHAT;
41              }
42
43              set
44              {
45                  _mSenseHAT = value;
46              }
47          }
48          #endregion
49
50          #region Constructors
51          /// <summary>
52          /// Constructor: Initializer
53          /// </summary>
54          public ViewSenseHAT()
```

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

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 7th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomePiFaceDigital2 is a program who use
 *   a PiFace Digital 2, it's an electronic card who
 *   can be use to plug electronic component. This
 *   program use the PiFace Digital 2 to activate
 *   light and store.
\*-------------------------------------------------*/

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Globalization;
using System.Linq;
using System.Reflection;
using System.Text;

namespace RaspiHomePiFaceDigital2
{
    public class ModelPiFaceDigital2
    {
        #region Fields
        #region Constants
        #endregion

        #region Variables
        private ViewPiFaceDigital2 _vPiFace;

        private List<Component> _components;
        private CommunicationWithServer _comWithServer;

        // Command to know
        private List<string> _raspiHomeComponentKnown = new List<string>()
        {
            "lumiere","lumieres",
            "store","stores",
            "television","televisions",
            "porte","portes",
            "fenetre","fenetres",
        };

        private List<string> _raspiHomeActionKnown = new List<string>()
        {
            "allumer","allume",
            "eteindre","eteins",
            "monter","monte",
            "descendre","descends",
            "stopper","stop",
            "ouvrir","ouvre",
            "fermer","ferme",
            "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>>
              _raspiBooleanCommandTranslation = new Dictionary<string,
              Dictionary<string, bool>>()
68          {
69              { "allume", new Dictionary<string, bool> { { "IsOn", true } } },
                { "allumer", new Dictionary<string, bool> { { "IsOn",
                true } } },
70              { "eteins", new Dictionary<string, bool> { { "IsOn", false } } },
                { "eteindre", new Dictionary<string, bool> { { "IsOn",
                false } } },
71              { "monte", new Dictionary<string, bool> { { "IsUp", true } } },
                { "monter", new Dictionary<string, bool> { { "IsUp", true } } },
72              { "descends", new Dictionary<string, bool> { { "IsDown",
                true } } }, { "descendre", new Dictionary<string, bool>
                { { "IsDown", true } } },
73              { "stop",new Dictionary<string, bool> { {"IsStop",true } } },
                {"stopper",new Dictionary<string, bool> { {"IsStop",true } } },
74          };
75          #endregion
76          #endregion
77
78          #region Properties
79          public ViewPiFaceDigital2 VPiFace
80          {
81              get
82              {
83                  return _vPiFace;
84              }
85
86              set
87              {
88                  _vPiFace = value;
89              }
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             // Initilize 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.InitilizeSPI();
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,
```

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

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

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

```
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
                  (NormalizationForm.FormC);
315         }
316         #endregion
317     }
318 }
319 
```

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 7th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomePiFaceDigital2 is a program who use
 *   a PiFace Digital 2, it's an electronic card who
 *   can be use to plug electronic component. This
 *   program use the PiFace Digital 2 to activate
 *   light and store.
\*-------------------------------------------------*/

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

namespace RaspiHomePiFaceDigital2
{
    public class CommunicationWithServer
    {
        #region Fields
        #region Constants
        // Default information to connect on the server
        private const int PORT = 54565;
        //// Need to be changed fo each configuration
        private const string IPSERVER = "10.134.97.117";// "192.168.2.8";

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

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

        private const int MESSAGE_FULL_LENGHT = 512;
        #endregion

        #region Variables
        private ModelPiFaceDigital2 _mPiFace;

        // Connection's variable
        private StreamSocket _socket = new StreamSocket();
        private StreamSocketListener _listener = new StreamSocketListener();
        private List<StreamSocket> _connections = new List<StreamSocket>();
        private bool _isConnected = false;
        private bool _connecting = false;
        #endregion
        #endregion

```

```
56          #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),
                      PORT.ToString());
161                 SendForInitialize();
162                 this.Connecting = false;
163                 this.IsConnected = true;
164 
165                 WaitForData(this.Socket);
166             }
```

```
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,
              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
                  string length.
199             dataWriter.WriteInt32((int)len);
200             dataWriter.WriteString(message);
201             var ret = await dataWriter.StoreAsync();
202             dataWriter.DetachStream();
203         }
204 
205         /// <summary>
206         /// Send to initialize the raspberry to the server
207         /// </summary>
208         private void SendForInitialize()
209         {
210             // Message send:
                  "@NAME@Connection:IPRASP=x.x.x.x;Location=y;Component=z,z"
211             SendMessage(this.Socket, string.Format(COMMUNICATIONSEPARATOR +
                  RPINAME + COMMUNICATIONSEPARATOR + FORMATSTRING, GetHostName(),
                  LOCATION, GetComponent()));
212         }
213 
214         /// <summary>
215         /// Wait data readed if exist
216         /// </summary>
217         /// <param name="socket"></param>
```

```
218         private async void WaitForData(StreamSocket socket)
219         {
220             DataReader dataReader = new DataReader(socket.InputStream);
221             dataReader.InputStreamOptions = InputStreamOptions.Partial;
222             var msglenght = dataReader.UnconsumedBufferLength;
223             uint stringBytes = msglenght;
224 
225 
226             try
227             {
228                 // Read modification in the stream
229                 stringBytes = await dataReader.LoadAsync(MESSAGE_FULL_LENGHT);
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 (msglenght < 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 =                                                    ↵
                Windows.Networking.Connectivity.NetworkInformation.GetHostNames  ↵
                ().ToList();
260             foreach (var Host in Hosts)
261             {
262                 string IP = Host.DisplayName;
263                 IpAddress.Add(IP);
264             }
265             return IpAddress.Last();
266         }
267 
268         /// <summary>
269         /// Get component in the list of components
270         /// </summary>
271         /// <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
```

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 7th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomePiFaceDigital2 is a program who use
 *   a PiFace Digital 2, it's an electronic card who
 *   can be use to plug electronic component. This
 *   program use the PiFace Digital 2 to activate
 *   light and store.
\*-------------------------------------------------*/

namespace RaspiHomePiFaceDigital2
{
    public class Light : Component
    {
        #region Fields
        #region Constant
        // PiFace output
        private const byte RELAIA = PiFaceDigital2.RelayA;
        private const byte RELAIB = PiFaceDigital2.RelayB;

        // PiFace State
        private const byte OFF = MCP23S17.Off;
        private const byte ON = MCP23S17.On;
        #endregion

        #region Variable
        private bool _isOn = false;
        private bool _isOnA = false;
        private bool _isOnB = false;
        #endregion
        #endregion

        #region Properties
        public bool IsOn
        {
            get
            {
                return _isOn;
            }

            set
            {
                _isOn = value;
                this.IsOnA = value;
                this.IsOnB = value;
            }
        }

        public bool IsOnA
        {
            get
            {
```

```
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
```

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 7th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomePiFaceDigital2 is a program who use
 *   a PiFace Digital 2, it's an electronic card who
 *   can be use to plug electronic component. This
 *   program use the PiFace Digital 2 to activate
 *   light and store.
\*-------------------------------------------------*/

using System;
using System.Threading.Tasks;
using Windows.UI.Xaml;

namespace RaspiHomePiFaceDigital2
{
    public class Store : Component
    {
        #region Fields
        #region Constant
        // PiFace output for motor
        private const byte UP = PiFaceDigital2.LED4;
        private const byte DOWN = PiFaceDigital2.LED3;

        // PiFace State
        private const byte OFF = MCP23S17.Off;
        private const byte ON = MCP23S17.On;

        // Max value for store (totaly open)
        private const int MAX_LEVEL = 200; // Time span total = 19seconds
          (raspberry latency)
        // Min value for store (totaly close)
        private const int MIN_LEVEL = 0;

        // Tick for timer
        private const int TICKS = 10;
        private const int TICK_SECOND = 1;
        #endregion

        #region Variable
        private DispatcherTimer _dTimerUp = new DispatcherTimer();
        private DispatcherTimer _dTimerDown = new DispatcherTimer();

        private bool _isUp = false;
        private bool _isDown = false;
        private bool _isOpen = false;
        private bool _isClose = false;
        private bool _isStop = false;

        private int _counterStopped = 0;
        #endregion
        #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
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.IsClose = false;
230
231                     this.SetLevel("IsUp");
232                     await Task.Delay(TimeSpan.FromSeconds(TICK_SECOND));
233                     this.SetLevel("IsStop");
234                     break;
235                 case "IsClose":
236                     this.IsOpen = false;
237
238                     this.SetLevel("IsDown");
239                     await Task.Delay(TimeSpan.FromSeconds(TICK_SECOND));
240                     this.SetLevel("IsStop");
241                     break;
242                 case "IsStop":
243                     this.IsUp = false;
244                     this.IsDown = false;
245                     this.IsOpen = false;
246                     this.IsClose = false;
247
248                     MCP23S17.WritePin(UP, OFF);
249                     MCP23S17.WritePin(DOWN, OFF);
250                     break;
251             }
252         }
253
254         /// <summary>
255         /// Set upper the level of the store
256         /// </summary>
257         private void SetLevelUp()
258         {
259             this._dTimerDown.Stop();
260             this._dTimerUp.Start();
261         }
262
263         /// <summary>
264         /// Set downer the level of the store
265         /// </summary>
266         private void SetLevelDown()
267         {
268             this._dTimerUp.Stop();
269             this._dTimerDown.Start();
270         }
271         #endregion
272     }
273 }
274
```

```
<Page
    x:Class="RaspiHomePiFaceDigital2.ViewPiFaceDigital2"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RaspiHomePiFaceDigital2"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
</Page>

```

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 7th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomePiFaceDigital2 is a program who use
 *   a PiFace Digital 2, it's an electronic card who
 *   can be use to plug electronic component. This
 *   program use the PiFace Digital 2 to activate
 *   light and store.
\*-------------------------------------------------*/

using Windows.UI.Xaml.Controls;

// Pour plus d'informations sur le modèle d'élément Page vierge, consultez la page http://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409

namespace RaspiHomePiFaceDigital2
{
    /// <summary>
    /// Une page vide peut être utilisée seule ou constituer une page de destination au sein d'un frame.
    /// </summary>
    public sealed partial class ViewPiFaceDigital2 : Page
    {
        #region Fields
        #region Constants
        #endregion

        #region Variables
        private ModelPiFaceDigital2 _mPiFace;
        #endregion
        #endregion

        #region Properties
        public ModelPiFaceDigital2 MPiFace
        {
            get
            {
                return _mPiFace;
            }

            set
            {
                _mPiFace = value;
            }
        }
        #endregion

        #region Constructors
        /// <summary>
        /// Construcor: Initializer
        /// </summary>
        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
               0), 1 = Inverted (low reads as 1)
24          public const byte LED4 = 0x0C;     // MCP23x17 Interrupt on Change Pin
              Assignements
25          public const byte LED5 = 0x0D;      // 1 = Input (default), 0 = Output
26          public const byte LED6 = 0x0E;     // MCP23x17 Input Polarity Register
27          public const byte LED7 = 0x0F;     // 0 = Normal (default)(low reads as
               0), 1 = Inverted (low reads as 1)
28
29          // Input
30          public const byte IN0 = 0x00;     // I/O Direction Register
31          public const byte IN1 = 0x01;      // 1 = Input (default), 0 = Output
32          public const byte IN2 = 0x02;     // MCP23x17 Input Polarity Register
33          public const byte IN3 = 0x03;     // 0 = Normal (default)(low reads as
              0), 1 = Inverted (low reads as 1)
34          public const byte IN4 = 0x04;     // MCP23x17 Interrupt on Change Pin
              Assignements
35          public const byte IN5 = 0x05;      // 1 = Input (default), 0 = Output
36          public const byte IN6 = 0x06;     // MCP23x17 Input Polarity Register
37          public const byte IN7 = 0x07;     // 0 = Normal (default)(low reads as
              0), 1 = Inverted (low reads as 1)
38
39          // Switch / Button
40          public const byte Sw0 = IN0;     // I/O Direction Register
41          public const byte Sw1 = IN1;      // 1 = Input (default), 0 = Output
42          public const byte Sw2 = IN2;     // MCP23x17 Input Polarity Register
43          public const byte Sw3 = IN3;     // 0 = Normal (default)(low reads as
              0), 1 = Inverted (low reads as 1)
44
45          // Relay
46          public const byte RelayA = LED1;     // MCP23x17 Input Polarity
              Register
47          public const byte RelayB = LED0;     // 0 = Normal (default)(low reads
              as 0), 1 = Inverted (low reads as 1)
```

```
48      }
49  }
50
```

```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 7th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomePiFaceDigital2 is a program who use
 *   a PiFace Digital 2, it's an electronic card who
 *   can be use to plug electronic component. This
 *   program use the PiFace Digital 2 to activate
 *   light and store.
\*-------------------------------------------------*/

using System;
using System.Diagnostics;
using System.Threading.Tasks;
using Windows.Devices.Enumeration;
using Windows.Devices.Spi;

namespace RaspiHomePiFaceDigital2
{
    public class MCP23S17
    {

        private const byte IODIRA = 0x00;     // I/O Direction Register
        private const byte IODIRB = 0x01;      // 1 = Input (default), 0 =
          Output
        private const byte IPOLA = 0x02;     // MCP23x17 Input Polarity
          Register
        private const byte IPOLB = 0x03;     // 0 = Normal (default)(low reads
           as 0), 1 = Inverted (low reads as 1)
        private const byte GPINTENA = 0x04;     // MCP23x17 Interrupt on
          Change Pin Assignements
        private const byte GPINTENB = 0x05;     // 0 = No Interrupt on Change
          (default), 1 = Interrupt on Change
        private const byte DEFVALA = 0x06;    // MCP23x17 Default Compare
          Register for Interrupt on Change
        private const byte DEFVALB = 0x07;     // Opposite of what is here
          will trigger an interrupt (default = 0)
        private const byte INTCONA = 0x08;     // MCP23x17 Interrupt on Change
           Control Register
        private const byte INTCONB = 0x09;     // 1 = pin is compared to
          DEFVAL, 0 = pin is compared to previous state (default)
        private const byte IOCONA = 0x0A;     // MCP23x17 Configuration
          Register
        private const byte IOCONB = 0x0B;    //     Also Configuration
          Register
        private const byte GPPUA = 0x0C;     // MCP23x17 Weak Pull-Up Resistor
           Register
        private const byte GPPUB = 0x0D;     // INPUT ONLY: 0 = No Internal
          100k Pull-Up (default) 1 = Internal 100k Pull-Up
        private const byte INTFA = 0x0E;     // MCP23x17 Interrupt Flag
          Register
        private const byte INTFB = 0x0F;    // READ ONLY: 1 = This Pin
          Triggered the Interrupt
```

```
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 InitilizeSPI()
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
                   */
89              catch (Exception ex)
90              {
91                  Debug.WriteLine(ex.Message);
92                  //statusText.Text = "\nSPI Initialization Failed";
93              }
94          }
95
96          public 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         }
102         public 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         public static void WriteRegister16(byte register, UInt16 value)
121         {
122             writeBuffer4[0] = (BaseAddW | (Address << 1));
123             writeBuffer4[1] = register;
124             writeBuffer4[2] = (byte)(value >> 8);
125             writeBuffer4[3] = (byte)(value & 0XFF);
126             try
```

```
127             {
128                 SpiGPIO.Write(writeBuffer4);
129             }
130 
131             /* If initialization fails, display the exception and stop running
                   */
132             catch (Exception ex)
133             {
134                 Debug.WriteLine(ex.Message);
135                 //statusText.Text = "\nFailed to Wrie to DAC";
136             }
137         }
138 
139         // Set the pin mode a pin at a time or all 16 in one go
140         // any value other then Input is taken as output
141         public static void setPinMode(byte pin, byte mode)
142         {
143             if (pin > 15) return;               // only a 16bit port so do a
                  bounds check, it cant be less than zero as this is a byte value
144             if (mode == Input)
145             {
146                 PinMode |= (UInt16)(1 << (pin));               // update the
                    pinMode register with new direction
147             }
148             else
149             {
150                 PinMode &= (UInt16)(~(1 << (pin)));            // update the
                    pinMode register with new direction
151             }
152             WriteRegister16(IODIRA, PinMode);                // Call the
                  generic word writer with start register and the mode cache
153         }
154         public static void SetPinMode(UInt16 mode)
155         {
156             WriteRegister16(IODIRA, mode);
157             PinMode = mode;
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);                                   //  ↵
                  Return the constructed word, the format is 0x(register value)
232         }
```

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