

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

```

57     };
58
59     // Word translation
60     private Dictionary<string, string> _raspiLanguageTranslation = new Dictionary<string, string>()
61     {
62         { "lumiere", "Light"}, { "lumières", "Light"},
63         { "store", "Store"}, { "stores", "Store"},
64     };
65
66     // KEY=[ACTION NAME], VALUE[KEY=[PROPERTY NAME], VALUE=[VALUE TO SET THEPROPERTY]]
67     private Dictionary<string, Dictionary<string, bool>>
68     _raspiBooleanCommandTranslation = new Dictionary<string, Dictionary<string, bool>>()
69     {
70         { "allume", new Dictionary<string, bool> { { "IsOn", true } } },
71         { "allumer", new Dictionary<string, bool> { { "IsOn", true } } },
72         { "eteins", new Dictionary<string, bool> { { "IsOn", false } } },
73         { "eteindre", new Dictionary<string, bool> { { "IsOn", false } } },
74         { "monte", new Dictionary<string, bool> { { "IsUp", true } } },
75         { "monter", new Dictionary<string, bool> { { "IsUp", true } } },
76         { "descends", new Dictionary<string, bool> { { "IsDown", true } } },
77         { "descendre", new Dictionary<string, bool> { { "IsDown", true } } },
78         { "stop", new Dictionary<string, bool> { { "IsStop", true } } },
79         { "stopper", new Dictionary<string, bool> { { "IsStop", true } } },
80     };
81     #endregion
82     #endregion
83
84     #region Properties
85     public ViewPiFaceDigital2 VPiFace
86     {
87         get
88         {
89             return _vPiFace;
90         }
91         set
92         {
93             _vPiFace = value;
94         }
95     }
96
97     public List<Component> Components
98     {
99         get
100         {
101             return _components;
102         }
103         set
104         {

```

```

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
130     // the Raspberry
131     this.Components = new List<Component>();
132     this.Components.Add(new Light());
133     this.Components.Add(new Store());
134
135     // Initilize the PiFace Digital 2
136     InitializePiFace();
137
138     // Initialize the server communication
139     this.ComWithServer = new CommunicationWithServer(this);
140 }
141 #endregion
142
143 #region Methods
144 /// <summary>
145 /// Initialize the PiFace Digital 2
146 /// </summary>
147 private async void InitializePiFace()
148 {
149     try
150     {
151         await MCP23S17.InitilizeSPI();
152
153         MCP23S17.InitializeMCP23S17();
154         MCP23S17.SetPinMode(0x00FF); // 0x0000 = all outputs,
155                                     // 0xffff=all inputs, 0x00FF is PIFace Default
156         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 component 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                                     ↗
261         this._raspiBooleanCommandTranslation.Keys)
262     {
263         if (actionKeys == actionName)
264         {
265             // Find the Value of the dictionary trough the inner
266             dictionary to get the first value
267             result = this._raspiBooleanCommandTranslation
268             [actionName].First().Key;
269             break;
270         }
271     }
272     return result;
273 }
274
275 /// <summary>
276 /// Search the val to change
277 /// </summary>
278 /// <param name="component"> the component to write value </param>
279 /// <param name="action"> the action (ON/OFF) </param>
280 /// <param name="typeVariable"> the property to change value </param>
281 private void WriteValue(Component component, string action,
282     PropertyInfo typeVariable)
283 {
284     switch (typeVariable.PropertyType.Name)
285     {
286     case "Boolean":
287         // Set the new value dynamicaly with value registered in
288         an boolean dictionary
289         typeVariable.SetValue(component,
290             this._raspiBooleanCommandTranslation[action]
291             [typeVariable.Name]);
292         break;
293     case "Double":
294         break;
295     case "Int16":
296     case "Int32":
297     case "Int64":
298         break;
299     }
300 }
301
302 /// <summary>
303 /// Stack Overflow solution to delete accents in strings
304 /// http://stackoverflow.com/questions/249087/how-do-i-remove-
305 /// diacritics-accents-from-a-string-in-net
306 /// </summary>
307 /// <param name="sentence"> sentence with diacritics to remove </
308 param>
309 /// <returns> same sentence without diacritics </returns>
310 private string RemoveDiacritics(string sentence)
311 {
312     var normalizedString = sentence.Normalize
313     (NormalizationForm.FormD);
314     var stringBuilder = new StringBuilder();
```

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

```
1  /*-----*\
2  * Author    : Salvi Cyril
3  * Date      : 7th june 2017
4  * Diploma  : RaspiHome
5  * Classroom : T.IS-E2B
6  *
7  * Description:
8  *     RaspiHomePiFaceDigital2 is a program who use
9  *     a PiFace Digital 2, it's an electronic card who
10 *     can be use to plug electronic component. This
11 *     program use the PiFace Digital 2 to activate
12 *     light and store.
13 \*-----*/
14
15 using System;
16 using System.Collections.Generic;
17 using System.Linq;
18 using Windows.Networking;
19 using Windows.Networking.Sockets;
20 using Windows.Storage.Streams;
21
22 namespace RaspiHomePiFaceDigital2
23 {
24     public class CommunicationWithServer
25     {
26         #region Fields
27         #region Constants
28         // Default information to connect on the server
29         private const int PORT = 54565;
30         //// Need to be changed fo each configuration
31         private const string IPSEVER = "10.134.97.117";// "192.168.2.8";
32
33         // String format for connection of the client
34         private const string FORMATSTRING = "Connection:IPRasp={0};Location= 7
35             {1};Component={2}";
36         private const string COMMUNICATIONSEPARATOR = "@";
37
38         // Important need to be changed if it's another room!
39         private const string LOCATION = "Salon";
40         private const string RPINAME = "PiFace_" + LOCATION;
41
42         private const int MESSAGE_FULL_LENIGHT = 512;
43         #endregion
44
45         #region Variables
46         private ModelPiFaceDigital2 _mPiFace;
47
48         // Connection's variable
49         private StreamSocket _socket = new StreamSocket();
50         private StreamSocketListener _listener = new StreamSocketListener();
51         private List<StreamSocket> _connections = new List<StreamSocket>();
52         private bool _isConnected = false;
53         private bool _connecting = false;
54         #endregion
55         #endregion
```



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

```
112         {
113             return _isConnected;
114         }
115
116         set
117         {
118             _isConnected = value;
119         }
120     }
121
122     public bool Connecting
123     {
124         get
125         {
126             return _connecting;
127         }
128
129         set
130         {
131             _connecting = value;
132         }
133     }
134 #endregion
135
136 #region Constructors
137 /// <summary>
138 /// Constructor: Initializer
139 /// </summary>
140 /// <param name="paramModel"></param>
141 public CommunicationWithServer(ModelPiFaceDigital2 paramModel)
142 {
143     this.MPiFace = paramModel;
144
145     Connect();
146 }
147 #endregion
148
149 #region Methods
150 #region Methods
151 /// <summary>
152 /// Connect the raspberry to the server
153 /// </summary>
154 private async void Connect()
155 {
156     try
157     {
158         this.Connecting = true;
159         // wait a confirmation from the server
160         await this.Socket.ConnectAsync(new HostName(IPSERVER),
161             PORT.ToString());
162         SendForInitialize();
163         this.Connecting = false;
164         this.IsConnected = true;
165
166         WaitForData(this.Socket);
167     }
```

```

167         catch (Exception)
168         {
169             this.Connecting = false;
170             this.IsConnected = false;
171         }
172     }
173
174     /// <summary>
175     /// Listen the traffic on the port
176     /// </summary>
177     private async void Listen()
178     {
179         this.Listener.ConnectionReceived += listenerConnectionReceived;
180         await this.Listener.BindServiceNameAsync(PORT.ToString());
181     }
182
183     void listenerConnectionReceived(StreamSocketListener sender,           ↗
184         StreamSocketListenerConnectionReceivedEventArgs args)
185     {
186         this.Connections.Add(args.Socket);
187
188         WaitForData(args.Socket);
189     }
190
191     /// <summary>
192     /// Send the message in input to output
193     /// </summary>
194     /// <param name="socket"> actual stream </param>
195     /// <param name="message"> message to send </param>
196     private async void SendMessage(StreamSocket socket, string message)
197     {
198         DataWriter dataWriter = new DataWriter(socket.OutputStream);
199         var len = dataWriter.MeasureString(message); // Gets the UTF-8      ↗
200             string length.
201         dataWriter.WriteInt32((int)len);
202         dataWriter.WriteString(message);
203         var ret = await dataWriter.StoreAsync();
204         dataWriter.DetachStream();
205     }
206
207     /// <summary>
208     /// Send to initialize the raspberry to the server
209     /// </summary>
210     private void SendForInitialize()
211     {
212         // Message send:           ↗
213         "@NAME@Connection:IPRASP=x.x.x.x;Location=y;Component=z,z"
214         SendMessage(this.Socket, string.Format(COMMUNICATIONSEPARATOR +    ↗
215             RPINAME + COMMUNICATIONSEPARATOR + FORMATSTRING, GetHostName(), ↗
216             LOCATION, GetComponent()));
217     }
218
219     /// <summary>
220     /// Wait data readed if exist
221     /// </summary>
222     /// <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 msglength = dataReader.UnconsumedBufferLength;
223         uint stringBytes = msglength;
224
225
226         try
227         {
228             // Read modification in the stream
229             stringBytes = await dataReader.LoadAsync(MESSAGE_FULL_LENGTH);
230
231             // read message
232             string msg = dataReader.ReadString(stringBytes);
233
234             // Send in return if the value exist
235             if (msg != "")
236             {
237                 this.MPiFace.SetValue(msg);
238             }
239         }
240         catch (Exception e)
241         {
242             string output = e.Message;
243
244             if (msglength < 1)
245                 return;
246         }
247
248         // Restart loop to wait data
249         WaitForData(socket);
250     }
251
252     /// <summary>
253     /// Get the ip of the raspberry
254     /// </summary>
255     /// <returns>return a string like 192.168.1.2</returns>
256     private string GetHostName()
257     {
258         List<string> IpAddress = new List<string>();
259         var Hosts =
260             Windows.Networking.Connectivity.NetworkInformation.GetHostNames
261             ().ToList();
262         foreach (var Host in Hosts)
263         {
264             string IP = Host.DisplayName;
265             IpAddress.Add(IP);
266         }
267         return IpAddress.Last();
268
269     }
270
271     /// <summary>
272     /// Get component in the list of components
273     /// </summary>
274     /// <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         return result;
287     }
288 }
289 #endregion
290 #endregion
291 }
292 }
293
294
```

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

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

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



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

```
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
234         this.SetLevel("IsStop");
235         break;
236     case "IsClose":
237         this.IsOpen = false;
238
239         this.SetLevel("IsDown");
240         await Task.Delay(TimeSpan.FromSeconds(TICK_SECOND));
241         this.SetLevel("IsStop");
242         break;
243     case "IsStop":
244         this.IsUp = false;
245         this.IsDown = false;
246         this.IsOpen = false;
247         this.IsClose = false;
248
249         MCP23S17.WritePin(UP, OFF);
250         MCP23S17.WritePin(DOWN, OFF);
251         break;
252     }
253
254     /// <summary>
255     /// Set upper the level of the store
256     /// </summary>
257     private void SetLevelUp()
258     {
259         this._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
```

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

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

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



```
1  /*-----*\
2  * Author    : Salvi Cyril
3  * Date      : 7th june 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

```

1  /*-----*\
2  * Author    : Salvi Cyril
3  * Date      : 7th june 2017
4  * Diploma  : RaspiHome
5  * Classroom : T.IS-E2B
6  *
7  * Description:
8  *     RaspiHomePiFaceDigital2 is a program who use
9  *     a PiFace Digital 2, it's an electronic card who
10 *     can be use to plug electronic component. This
11 *     program use the PiFace Digital 2 to activate
12 *     light and store.
13 \*-----*/
14
15 using System;
16 using System.Diagnostics;
17 using System.Threading.Tasks;
18 using Windows.Devices.Enumeration;
19 using Windows.Devices.Spi;
20
21 namespace RaspiHomePiFaceDigital2
22 {
23     public class MCP23S17
24     {
25
26         private const byte IODIRA = 0x00;    // I/O Direction Register
27         private const byte IODIRB = 0x01;    // 1 = Input (default), 0 =  ➤
28         Output
29         private const byte IPOLA = 0x02;    // MCP23x17 Input Polarity      ➤
30         Register
31         private const byte IPOLB = 0x03;    // 0 = Normal (default)(low reads ➤
32         as 0), 1 = Inverted (low reads as 1)
33         private const byte GPINTENA = 0x04; // MCP23x17 Interrupt on      ➤
34         Change Pin Assignements
35         private const byte GPINTENB = 0x05; // 0 = No Interrupt on Change ➤
36         (default), 1 = Interrupt on Change
37         private const byte DEFVALA = 0x06; // MCP23x17 Default Compare ➤
38         Register for Interrupt on Change
39         private const byte DEFVALB = 0x07; // Opposite of what is here ➤
40         will trigger an interrupt (default = 0)
41         private const byte INTCONA = 0x08; // MCP23x17 Interrupt on Change ➤
42         Control Register
43         private const byte INTCONB = 0x09; // 1 = pin is compared to ➤
44         DEFVAL, 0 = pin is compared to previous state (default)
45         private const byte IOCONA = 0x0A; // MCP23x17 Configuration ➤
46         Register
47         private const byte IOCONB = 0x0B; // Also Configuration ➤
48         Register
49         private const byte GPPUA = 0x0C; // MCP23x17 Weak Pull-Up Resistor ➤
50         Register
51         private const byte GPPUB = 0x0D; // INPUT ONLY: 0 = No Internal ➤
52         100k Pull-Up (default) 1 = Internal 100k Pull-Up
53         private const byte INTFA = 0x0E; // MCP23x17 Interrupt Flag ➤
54         Register
55         private const byte INTFB = 0x0F; // READ ONLY: 1 = This Pin ➤
56         Triggered the Interrupt

```

```

...\\RaspiHomePiFaceDigital2\\PiFace initializer\\MCP23S17.cs 2
42     private const byte INTCAPA = 0x10;    // MCP23x17 Interrupt Captured  ↗
        Value for Port Register
43     private const byte INTCAPB = 0x11;    // READ ONLY: State of the Pin  ↗
        at the Time the Interrupt Occurred
44     private const byte GPIOA = 0x12;      // MCP23x17 GPIO Port Register
45     private const byte GPIOB = 0x13;      // Value on the Port - Writing  ↗
        Sets Bits in the Output Latch
46     private const byte OLATA = 0x14;      // MCP23x17 Output Latch  ↗
        Register
47     private const byte OLATB = 0x15;      // 1 = Latch High, 0 = Latch Low ↗
        (default) Reading Returns Latch State, Not Port Value!
48
49     public const byte On = 1;
50     public const byte Off = 0;
51     public const byte Output = 0;
52     public const byte Input = 1;
53
54     private const byte Address = 0x00;    // offset address if hardware  ↗
        addressing is on and is 0 - 7 (A0 - A2)
55     private const byte BaseAddW = 0x40;   // MCP23S17 Write base address
56     private const byte BaseAddR = 0x41;   // MCP23S17 Read Base Address
57     private const byte HAEN = 0x08;       // IOCON register for MCP23S17, x08 ↗
        enables hardware address so sent address must match hardware pins  ↗
        A0-A2
58
59
60     private static UInt16 PinMode = 0xFFFF;    // default Pinmode for the ↗
        MXP23S17 set to inputs
61     private static UInt16 PullUpMode = 0xFFFF;    // default pullups for ↗
        the MXP23S17 set to weak pullup
62     private static UInt16 InversionMode = 0x0000;    // default invert to ↗
        normal
63     private static UInt16 PinState = 0x0000;    // default pinstate to  ↗
        all 0's
64
65     /*RaspBerry Pi2 Parameters*/
66     private const string SPI_CONTROLLER_NAME = "SPI0"; /* For Raspberry ↗
        Pi 2, use SPI0 */
67     private const Int32 SPI_CHIP_SELECT_LINE = 0;    /* Line 0 maps to ↗
        physical pin number 24 on the Rpi2, line 1 to pin 26 */
68
69     private static byte[] readBuffer3 = new byte[3]; /*this is defined to ↗
        hold the output data*/
70     private static byte[] readBuffer4 = new byte[4]; /*this is defined to ↗
        hold the output data*/
71     private static byte[] writeBuffer3 = new byte[3]; //register, then 16 ↗
        bit value
72     private static byte[] writeBuffer4 = new byte[4]; //register, then 16 ↗
        bit value
73
74     private static SpiDevice SpiGPIO;
75     public static async Task 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(IOCNA, 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            */
133         catch (Exception ex)
134         {
135             Debug.WriteLine(ex.Message);
136             //statusText.Text = "\nFailed to Wrie to DAC";
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 ↗
144             // bounds check, it cant be less than zero as this is a byte value
145             if (mode == Input)
146             {
147                 PinMode |= (UInt16)(1 << (pin));           // update the ↗
148                 // pinMode register with new direction
149             }
150             else
151             {
152                 PinMode &= (UInt16)(~(1 << (pin)));           // update the ↗
153                 // pinMode register with new direction
154             }
155             WriteRegister16(IODIRA, PinMode);           // Call the ↗
156             // generic word writer with start register and the mode cache
157         }
158         public static void SetPinMode(UInt16 mode)
159         {
160             WriteRegister16(IODIRA, mode);
161             PinMode = mode;
162         }
163
164         // Set the pullup a pin at a time or all 16 in one go
165         // any value other than On is taken as off
166         public static void pullupMode(byte pin, byte mode)
167         {
168             if (pin > 15) return;
169             if (mode == On)
170             {
171                 PullUpMode |= (UInt16)(1 << (pin));
172             }
173             else
174             {
175                 PullUpMode &= (UInt16)(~(1 << (pin)));
176             }
177             WriteRegister16(GPPUA, PullUpMode);
178         }
179         public static void PullupMode(UInt16 mode)
180         {
181             WriteRegister16(GPPUA, mode);
182         }

```

```

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

```

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