```
/*-------------------------------------------------*\
 * 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
```