using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Windows.Threading;

using System.Windows.Forms;

using System.Threading;

namespace ovencontrol

{

/// <summary>

/// Interaction logic for UserControl1.xaml

/// </summary>

public partial class four : UserControl

{

public bool fin\_c = false;

public string loc = System.Reflection.Assembly.GetExecutingAssembly().Location;

public static System.Media.SoundPlayer playover = new System.Media.SoundPlayer();

public static System.Media.SoundPlayer playend = new System.Media.SoundPlayer();

public DispatcherTimer overrun = new DispatcherTimer();

public DispatcherTimer ending = new DispatcherTimer();

public DispatcherTimer ending1 = new DispatcherTimer();

TimeSpan time\_acheive;

DateTime start\_at;

public byte a = 0, b = 0, c = 0;

public string s, name, setpoint, state, mesure, finishat, maxoven, minoven, remaining;

public DateTime tim = new DateTime();

public DateTime end = new DateTime();

public int idoven, max = 20, min = 20, sp;

public string lien;

bool transitoir = true;

Color on\_color = new Color();

Color over\_color = new Color();

Color end\_color = new Color();

public four()

{

InitializeComponent();

s = loc.Substring(0, loc.Length - 15);

playover.SoundLocation = s + "alarm1.wav";

playend.SoundLocation = s + "alarm.wav";

lien = loc;

overrun.Tick += new EventHandler(overrun\_Tick);

overrun.Interval = TimeSpan.FromMilliseconds(200);

ending.Tick += new EventHandler(ending\_Tick);

ending.Interval = TimeSpan.FromMilliseconds(200);

ending1.Tick += new EventHandler(ending1\_Tick);

ending1.Interval = TimeSpan.FromMilliseconds(200);

get\_default\_colors();

mesure4.setchar('c');

ovennc();

}

void get\_default\_colors()

{

on\_color = on.ColorOff;

over\_color = over.ColorOff;

end\_color = fin.ColorOff;

}

public void alignement(double height)

{

grid.Height = height;

}

private void UserControl\_Loaded(object sender, RoutedEventArgs e)

{

}

private void image3\_ImageFailed(object sender, ExceptionRoutedEventArgs e)

{

}

private void led1\_Loaded(object sender, RoutedEventArgs e)

{

}

// Clignotement

public void overrun\_Tick(object sender, EventArgs e)

{

if (a < 4)

{

Visibility = System.Windows.Visibility.Visible;

redison();

greenisoff();

orangeisoff();

segover();

MyGauge.Indicators[0].Background = Brushes.Red;

MyGauge.Indicators[3].Background = Brushes.Red;

a++;

}

else

{

Visibility = System.Windows.Visibility.Hidden;

a = 0;

}

}

public void ending\_Tick(object sender, EventArgs e)

{

if (a < 2)

{

orangeison();

redisoff();

greenisoff();

segcli();

a++;

}

else

{

orangeisoff();

segnc();

a = 0;

}

}

public void ending1\_Tick(object sender, EventArgs e)

{

if (a < 2)

{

orangeisoff();

redison();

greenisoff();

segcli();

a++;

}

else

{

redisoff();

segnc();

a = 0;

}

}

public void setName(string nom)

{

for (int i = 0; i < nom.Length; i++)

{

label1.Content += nom.ElementAt(i) + "\n";

}

}

// led's stats

public void greenison()

{

on.IsActive = true;

on.ColorOn = Colors.LawnGreen;

}

public void greenisoff()

{

on.IsActive = false;

on.ColorOff = on\_color;

}

public void redison()

{

over.IsActive = true;

over.ColorOn = Colors.Red;

}

public void redisoff()

{

over.IsActive = false;

over.ColorOff = over\_color;

}

public void orangeison()

{

fin.IsActive = true;

fin.ColorOn = Colors.Orange;

}

public void orangeisoff()

{

fin.IsActive = false;

fin.ColorOff = end\_color;

}

public void ledsnc()

{

fin.IsActive = null;

on.IsActive = null;

over.IsActive = null;

fin.ColorNull = Colors.Gray;

fin.IsEnabled = false;

over.ColorNull = Colors.Gray;

over.IsEnabled = false;

on.ColorNull = Colors.Gray;

on.IsEnabled = false;

}

public void ledColorOn(Color col, Color co, Color c)

{

on.ColorOn = col;

over.ColorOn = co;

fin.ColorOn = c;

}

public void ledColorNot(Color col)

{

on.ColorOn = col;

over.ColorOn = col;

fin.ColorOn = col;

over.ColorOff = col;

fin.ColorOff = col;

on.ColorOff = col;

}

public void ledColorOff(Color col, Color co, Color c)

{

on.ColorOff = col;

over.ColorOff = co;

fin.ColorOff = c;

}

public void ledColorNc(Color col, Color co, Color c)

{

on.ColorNull = col;

over.ColorNull = co;

fin.ColorNull = c;

}

// Seven Segments

public void sevenSegments(string s)

{

switch (s)

{

case "on": segon(); break;

case "nc": segnc(); break;

case "over": segover(); break;

case "fin": segcli(); break;

case "off": segoff(); break;

}

}

public void segon()

{

mesure4.setchar('c');

mesurecolor(Brushes.Black);

timecolor(Brushes.Black);

}

public void segon2()

{

mesure4.setchar('c');

mesurecolor(Brushes.Red);

timecolor(Brushes.Red);

}

public void segnc()

{

mesure4.setchar('c');

mesure4.ColonColor = Brushes.LightGray;

mesure1.NumberColor = Brushes.LightGray;

mesure2.NumberColor = Brushes.LightGray;

mesure3.NumberColor = Brushes.LightGray;

mesure4.NumberColor = Brushes.LightGray;

time1.NumberColor = Brushes.LightGray;

time2.NumberColor = Brushes.LightGray;

time3.NumberColor = Brushes.LightGray;

time3.ColonColor = Brushes.LightGray;

time4.NumberColor = Brushes.LightGray;

mesure1.desactive();

mesure2.desactive();

mesure3.desactive();

mesure4.desactive();

time1.desactive();

time2.desactive();

time3.desactive();

time4.desactive();

}

public void segover()

{

mesure4.setchar('c');

mesure4.ColonColor = Brushes.Red;

mesure1.NumberColor = Brushes.Red;

mesure2.NumberColor = Brushes.Red;

mesure3.NumberColor = Brushes.Red;

mesure4.NumberColor = Brushes.Red;

time1.NumberColor = Brushes.Red;

time2.NumberColor = Brushes.Red;

time3.NumberColor = Brushes.Red;

time3.ColonColor = Brushes.Red;

time4.NumberColor = Brushes.Red;

}

public void segcli()

{

mesure4.setchar('c');

mesure4.ColonColor = Brushes.Black;

mesure1.NumberColor = Brushes.Black;

mesure2.NumberColor = Brushes.Black;

mesure3.NumberColor = Brushes.Black;

mesure4.NumberColor = Brushes.Black;

time1.NumberColor = Brushes.Black;

time2.NumberColor = Brushes.Black;

time3.NumberColor = Brushes.Black;

time3.ColonColor = Brushes.Black;

time4.NumberColor = Brushes.Black;

}

public void segoff()

{

mesure4.setchar('c');

mesure4.ColonColor = Brushes.DarkGreen;

mesure1.NumberColor = Brushes.DarkGreen;

mesure2.NumberColor = Brushes.DarkGreen;

mesure3.NumberColor = Brushes.DarkGreen;

mesure4.NumberColor = Brushes.DarkGreen;

timecolor(Brushes.Silver);

time1.desactive();

time2.desactive();

time3.desactive();

time4.desactive();

}

public void setmesure(string m)

{

mesure1.SetNumber(int.Parse("" + (m.ElementAt(0))));

mesure2.SetNumber(int.Parse("" + (m.ElementAt(1))));

mesure3.SetNumber(int.Parse("" + (m.ElementAt(2))));

}

public void settime(string t)

{

time1.SetNumber(int.Parse("" + (t.ElementAt(4))));

time2.SetNumber(int.Parse("" + (t.ElementAt(3))));

time3.SetNumber(int.Parse("" + (t.ElementAt(1))));

time4.SetNumber(int.Parse("" + (t.ElementAt(0))));

}

public void timecolor(SolidColorBrush col)

{

time1.NumberColor = col;

time2.NumberColor = col;

time3.NumberColor = col;

time3.ColonColor = col;

time4.NumberColor = col;

}

public void mesurecolor(SolidColorBrush col)

{

mesure1.NumberColor = col;

mesure2.NumberColor = col;

mesure3.NumberColor = col;

mesure4.NumberColor = col;

mesure4.ColonColor = col;

}

// Gauge Temperature

public void barvalue(int val)

{

MyGauge.Indicators[0].Value = val;

MyGauge.Indicators[3].Value = val;

}

public void markermax(int mx)

{

MyGauge.Indicators[1].Value = mx;

}

public void markermin(int mn)

{

MyGauge.Indicators[2].Value = mn;

}

public void barcolor(SolidColorBrush col, SolidColorBrush co)

{

MyGauge.Indicators[0].Background = col;

MyGauge.Indicators[3].Background = co;

}

// oven stat

public void ovennc()

{

settime("00:00");

mesure4.setchar('c');

barvalue(24);

barcolor(Brushes.LightGray, Brushes.LightGray);

ledsnc();

segnc();

}

public void ovenoff()

{

settime("00:00");

fin\_c = false;

mesure4.setchar('c');

barcolor(Brushes.Red, Brushes.LawnGreen);

greenisoff();

redisoff();

orangeisoff();

segoff();

}

public void ovenon()

{

barcolor(Brushes.Orange, Brushes.Orange);

greenisoff();

redisoff();

orangeison();

segon();

}

public void ovenon2()

{

barcolor(Brushes.Red, Brushes.Red);

greenisoff();

redison();

orangeisoff();

segon2();

}

public void ovenon1()

{

barcolor(Brushes.DarkGreen, Brushes.DarkGreen);

greenison();

redisoff();

orangeisoff();

segon();

}

// initialisation

public void init(string start)

{

if (start.Length == 27)

{

idoven = int.Parse(start.Substring(0, 2));

name = start.Substring(2, 8);

setpoint = start.Substring(10, 3);

minoven = start.Substring(13, 3);

maxoven = start.Substring(16, 3);

time\_acheive = TimeSpan.Parse(start.Substring(19, 8));

markermax(int.Parse(setpoint));

setName(name);

}

}

//update

public int update(string msg)

{

if (msg.Length == 46)

{

idoven = int.Parse(msg.Substring(0, 2));

state = msg.Substring(2,3);

mesure = msg.Substring(5, 3);

finishat = msg.Substring(8, 19);

start\_at = DateTime.Parse(msg.Substring(27, 19));

tim = Convert.ToDateTime(finishat);

}

remaining = (tim.AddMinutes(1) - DateTime.Now.TimeOfDay).ToString().Substring(11, 5);

switch (state)

{

case "ON0": ovenon(); setmesure(mesure); settime(remaining); barvalue(int.Parse(mesure)); break;

case "ON1": ovenon(); setmesure(mesure); settime(remaining); barvalue(int.Parse(mesure)); break;

case "ON2": ovenon(); setmesure(mesure); settime(remaining); barvalue(int.Parse(mesure)); break;

case "ON3": ovenon(); setmesure(mesure); settime(remaining); barvalue(int.Parse(mesure)); break;

case "OFF": ovenoff();setmesure(mesure); barvalue(int.Parse(mesure)); ending.Stop(); ending1.Stop(); overrun.Stop(); Visibility = System.Windows.Visibility.Visible; break;

case "END": ovenoff();setmesure(mesure); barvalue(int.Parse(mesure)); ending.Stop(); ending1.Stop(); overrun.Stop(); Visibility = System.Windows.Visibility.Visible; break;

case "NC ": ovennc(); ending.Stop(); ending1.Stop(); overrun.Stop(); Visibility = System.Windows.Visibility.Visible; break;

}

if (state == "ON1" || state == "ON3")

{

int som = cli() + depasser() + not\_acheive();

return som;

}

return 0;

}

public int cli()

{

if (Convert.ToDateTime(remaining) < Convert.ToDateTime("00:02"))

{

if (int.Parse(mesure) > max || int.Parse(mesure) < min)

{

ending1.Start();

ending.Stop();

}

}

if (Convert.ToDateTime(remaining) == Convert.ToDateTime("00:01"))

{

return 8;

}

return 0;

}

public int depasser()

{

if (int.Parse(mesure) <= int.Parse(minoven) && (transitoir == true))

{

if (int.Parse(mesure) > max)

max = int.Parse(mesure);

if (int.Parse(mesure) <= max - 10) // chute

{

overrun.Start();

return 2;

}

}

else

{

transitoir = false;

max = int.Parse(maxoven);

min = int.Parse(minoven);

// sp = int.Parse(setpoint);

if (int.Parse(mesure) > max )

{

ovenon2();

overrun.Stop();

Visibility = System.Windows.Visibility.Visible;

}

else if (int.Parse(mesure) < min)

{

ovenon2();

overrun.Stop();

Visibility = System.Windows.Visibility.Visible;

}

else if (int.Parse(mesure) <= max && int.Parse(mesure) >= min)

{

if (Convert.ToDateTime(remaining) < Convert.ToDateTime("00:02"))

{

ending.Start();

ending1.Stop();

}

else

{

ovenon1();

overrun.Stop();

Visibility = System.Windows.Visibility.Visible;

}

}

}

return 0;

}

public int not\_acheive()

{

if ((start\_at + time\_acheive) <= DateTime.Now && transitoir)

{

if (int.Parse(mesure) < int.Parse(minoven))

{

overrun.Start();

return 4;

}

}

return 0;

}

}

}