using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO.Ports;

using System.Linq;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace UpperDemo

{

public partial class Form1 : Form

{

// 当前连接套接字

private Socket currentSocket;

// 当前串口

private SerialPort currentSerial;

// socket是否连接

bool isConnect = false;

// 串口是否打开

bool isOpen = false;

public Form1()

{

InitializeComponent();

// 禁用开关风扇按钮

btnControl.Enabled = false;

// 取消跨线程检查

Control.CheckForIllegalCrossThreadCalls = false;

}

// 连接socket服务器

private void btnConnect\_Click(object sender, EventArgs e)

{

txtData.AppendText("连接服务器...\r\n");

try

{

// 创建一个负责通信的Socket

currentSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

// 获得要连接的远程应用程序IP地址和端口号

IPAddress ip = IPAddress.Parse(txtIp.Text);

IPEndPoint point = new IPEndPoint(ip, Int32.Parse(txtPort.Text));

// 连接

currentSocket.Connect(point);

txtData.AppendText("连接服务器成功！\r\n");

isConnect = true;

if (isConnect || isOpen)

{

btnControl.Enabled = true;

}

// 开启一个新线程

// 用于接收服务器发送过来的消息

Thread th = new Thread(Recive);

th.IsBackground = true;

th.Start();

btnConnect.Enabled = false;

}

catch (Exception)

{

//

}

}

// 打开串口

private void btnOpen\_Click(object sender, EventArgs e)

{

txtData.AppendText("打开串口...\r\n");

try

{

currentSerial = new SerialPort();

// 波特率

currentSerial.BaudRate = Int32.Parse(txtBaudRate.Text);

// 串口名

currentSerial.PortName = txtCom.Text;

// 数据位

currentSerial.DataBits = 8;

// 打开

currentSerial.Open();

txtData.AppendText("打开串口成功!\r\n");

isOpen = true;

if (isOpen || isConnect)

{

btnControl.Enabled = true;

}

// 订阅读取事件

currentSerial.DataReceived += new SerialDataReceivedEventHandler(readSerial);

btnOpen.Enabled = false;

}

catch (Exception)

{

//

}

}

// 开关风扇

private void btnControl\_Click(object sender, EventArgs e)

{

if (btnControl.Text == "开风扇")

{

txtData.AppendText("PC开风扇...\r\n向串口发送开风扇指令...\r\n向服务器发送风扇状态...\r\n");

// 向串口发送指令

if (currentSerial.IsOpen)

currentSerial.Write("1");

// 向服务器发送风扇状态

if (currentSocket != null)

currentSocket.Send(Encoding.ASCII.GetBytes("01"));

btnControl.Text = "关风扇";

}

else

{

txtData.AppendText("PC关风扇...\r\n向串口发送关风扇指令...\r\n向服务器发送风扇状态...\r\n");

// 向串口发送指令

if (currentSerial.IsOpen)

currentSerial.Write("0");

// 向服务器发送风扇状态

if (currentSocket != null)

currentSocket.Send(Encoding.ASCII.GetBytes("00"));

btnControl.Text = "开风扇";

}

txtData.AppendText("处理成功!\r\n");

}

// 循环接受服务器的消息

void Recive()

{

while (true)

{

try

{

byte[] buffer = new byte[1];

int res = currentSocket.Receive(buffer);

if (res != 1)

{

break;

}

txtData.AppendText("收到服务器命令...\r\n");

if (buffer[0] == 1)

{

txtData.AppendText("服务器开风扇...\r\n");

btnControl.Text = "关风扇";

// 开风扇

if (currentSerial.IsOpen)

{

currentSerial.Write("1");

txtData.AppendText("处理成功!\r\n");

}

else

{

txtData.AppendText("处理失败!串口未打开.\r\n");

}

}

else if(buffer[0] == 0)

{

txtData.AppendText("服务器关风扇...\r\n");

btnControl.Text = "开风扇";

// 关风扇

if (currentSerial.IsOpen)

{

currentSerial.Write("0");

txtData.AppendText("处理成功!\r\n");

}

else

{

txtData.AppendText("处理失败!串口未打开.\r\n");

}

}

else

{

txtData.AppendText("错误命令：" + buffer[0] + "\r\n");

}

}

catch (Exception)

{

//

}

}

}

// 循环接受串口的数据

void readSerial(object sender, SerialDataReceivedEventArgs e)

{

if (currentSerial.IsOpen)

{

byte[] readbuffer = new byte[currentSerial.BytesToRead];

try

{

// 接收温度数据

currentSerial.Read(readbuffer, 0, readbuffer.Length);

currentSerial.DiscardInBuffer();

labTemp.Text = Encoding.ASCII.GetString(readbuffer);

txtData.AppendText("收到串口温度数据：" + labTemp.Text + "\r\n");

// 发送温度数据到服务器

if (currentSocket != null)

{

currentSocket.Send(Encoding.ASCII.GetBytes("1" + labTemp.Text));

txtData.AppendText("温度数据成功发送到服务器!\r\n");

}

//TODO:自动控制温度

}

catch (TimeoutException)

{

//

}

}

else

{

Thread.Sleep(100);

}

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

// 窗口关闭时关闭连接

if (currentSocket != null)

{

currentSocket.Shutdown(SocketShutdown.Both);

currentSocket.Close();

}

}

}

}