Project Skynet

Software Code : Project_Skynet/FormControl.cs

분당중학교 3학년 김태욱

(2015.07~09)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;

namespace Project_Skynet
{
    public partial class FormControl : Form
    {
        FormMain frmM = null;
        int AmmoLeft = 400;
        public FormControl(FormMain frmM)
        {
            this.frmM = frmM; //form main control
            this.Hide();
            InitializeComponent();
        }

        private void FormControl_Load(object sender, EventArgs e)
        {
            try
            {

                prg_Ammo.Maximum = 400;
                prg_Ammo.Value = 400;
                prg_Overheat.Maximum = 35;
                btn_LowAmmo.Visible = false;

                    serialPort1.PortName = this.frmM.Text;
                    serialPort1.BaudRate = 57600;
                    serialPort1.Open();


                txt_Status.Text = "Status = StandBy";

                if (serialPort1.IsOpen)
                {
                    this.Show();
                }
                else //when serial is not connected
                {
                    MessageBox.Show("Cannot Connect");
                    this.Show();
                }
            }
            catch (Exception ex)
            {
```

```
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }
    #region Normal Events

    private void txt_CMD_KeyDown(object sender, KeyEventArgs e)
    {
        try
        {
            if (e.KeyCode == Keys.Up)
            {

                Write("u", "");
                btn_Up.ForeColor = Color.Lime;
            }
            if (e.KeyCode == Keys.Down)
            {
                Write("d", "");
                btn_Down.ForeColor = Color.Lime;
            }
            if (e.KeyCode == Keys.Left)
            {
                Write("e", "");
                btn_Left.ForeColor = Color.Lime;
            }
            if (e.KeyCode == Keys.Right)
            {
                Write("t", "");
                btn_Right.ForeColor = Color.Lime;
            }
            if (e.KeyCode == Keys.Space)
            {
                Fire();
            }
            ////////////////////
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }
    #endregion

    private void txt_CMD_KeyUp(object sender, KeyEventArgs e)
    {
        try
        {
            if (e.KeyCode == Keys.Up || e.KeyCode == Keys.Down || e.KeyCode ==
              Keys.Left || e.KeyCode == Keys.Right)
            {
                KeyReset();
```

```
            }
            else if (e.KeyCode == Keys.Space)
            {
                Write("f", "");
                timer_Ammo.Stop();
                prg_Overheat.Value = 0;
                if (AmmoLeft <= 30)
                {
                    txt_CMD.AppendText(Environment.NewLine + "Ammo Low");
                }
            }
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }


    private void btn_ScanOn_Click(object sender, EventArgs e)
    {
        try
        {
            Write("S", "Scan Mode Activated");
            btn_ScanOff.BackColor = Color.DarkGray;
            btn_ScanOn.BackColor = Color.Lime;
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }

    private void btn_ScanOff_Click(object sender, EventArgs e)
    {
        try
        {
            Write("s", "Scan Mode Deactivated");
            btn_ScanOn.BackColor = Color.DarkGray;
            btn_ScanOff.BackColor = Color.Lime;
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }

    private void btn_LaserOn_Click(object sender, EventArgs e)
    {
        try
```

```
        {
            Write("Z", "Laser : ON");
            btn_LaserOff.BackColor = Color.DarkGray;
            btn_LaserOn.BackColor = Color.Lime;
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }

    private void btn_LaserOff_Click(object sender, EventArgs e)
    {
        try
        {
            Write("z", "Laser = OFF");
            btn_LaserOn.BackColor = Color.DarkGray;
            btn_LaserOff.BackColor = Color.Lime;
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }

    private void btn_LightOn_Click(object sender, EventArgs e)
    {
        try
        {
            Write("L", "Lights : ON");
            btn_LightOff.BackColor = Color.DarkGray;
            btn_LightOn.BackColor = Color.Lime;
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }

    private void btn_LightOff_Click(object sender, EventArgs e)
    {
        try
        {
            Write("l", "Lights : OFF");
            btn_LightOn.BackColor = Color.DarkGray;
            btn_LightOff.BackColor = Color.Lime;
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
```

```
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }

    private void btn_ReloadOn_Click(object sender, EventArgs e)
    {
        try
        {
            Write("R", "Reload Mode Activated");
            btn_ReloadOff.BackColor = Color.DarkGray;
            btn_ReloadOn.BackColor = Color.Lime;
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }

    private void btn_ReloadOff_Click(object sender, EventArgs e)
    {
        try
        {
            Write("r", "Reload Mode Deactivated");
            btn_ReloadOn.BackColor = Color.DarkGray;
            btn_ReloadOff.BackColor = Color.Lime;
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }

    private void btn_Manual_Click(object sender, EventArgs e)
    {
        try
        {
            Write("M", "Firing Mode Set To : MANUAL");
            btn_Auto.BackColor = Color.DarkGray;
            btn_Manual.BackColor = Color.Lime;
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }

    private void btn_Auto_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            Write("A", "Firing Mode Set To : AUTO");
            btn_Manual.BackColor = Color.DarkGray;
            btn_Auto.BackColor = Color.Lime;
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }

    private void Fire()
    {
        try
        {
            Write("F", "Firing");
            txt_Status.ForeColor = Color.Red;
            txt_Status.Text = "Status = FIRING";
            timer_Ammo.Interval = 200;
            timer_Ammo.Start();
        }
        catch (Exception ex)
        {
            this.frmM.redscreen();
            this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);
        }
    }

    private void KeyReset() //sets all keys to darkgray
    {
        btn_Up.ForeColor = Color.DarkGray;
        btn_Down.ForeColor = Color.DarkGray;
        btn_Left.ForeColor = Color.DarkGray;
        btn_Right.ForeColor = Color.DarkGray;
    }

    public void Write(string a, string b) //sends char by serial and prints string ↵
      to txt_CMD
    {
        try
        {
            if (this.frmM.CheckPrgMode() == 1)
            {
                serialPort1.Write(a);
                this.frmM.WriteSerial(a);
                txt_Serial.AppendText(Environment.NewLine + DateTime.Now + ">>" + ↵
                  b);

                if (b != "")
                {
                    txt_CMD.AppendText(Environment.NewLine + b);
                }
                else //if b == "" ignore   -->for up,down,left,right btns.
```

```
                {

                }
            }
            else
            {
                //ignore
            }
        }
        catch (Exception ex)
        {
            if (this.frmM.z == 1) //Programming Mode On
            {
                //ignore
                }
            else if (this.frmM.z == 0) //Programming Mode Off
            {
                this.frmM.redscreen();

                this.frmM.WriteCMD(Environment.NewLine + "ERROR >> " + ex.Message);

            }
        }
    }

    public void Disconn() //FormMain.Disconnect button
    {
        serialPort1.Close();
        this.frmM.WriteCMD("Serial Disconnected");
    }

    private void timer_Ammo_Tick(object sender, EventArgs e)
    {
        try
        {
            prg_Ammo.Value--;
            prg_Overheat.Value += 3;
            AmmoLeft--;
            txt_Ammo.Text = AmmoLeft.ToString();
            if (AmmoLeft <= 0)
            {
                btn_LowAmmo.Visible = true;
            }
            if (prg_Overheat.Value >= 30)
            {
                Write("f", "");
                txt_CMD.AppendText(Environment.NewLine + "Turret Overheated >>
                  Cooling. . .");
                timer_Ammo.Stop();

            }
        }
        catch (Exception Ex)
```

```
            {
                txt_CMD.AppendText(Environment.NewLine + Ex.Message);
            }
        }
    }
}
```