**Додатки**

**Додаток А**

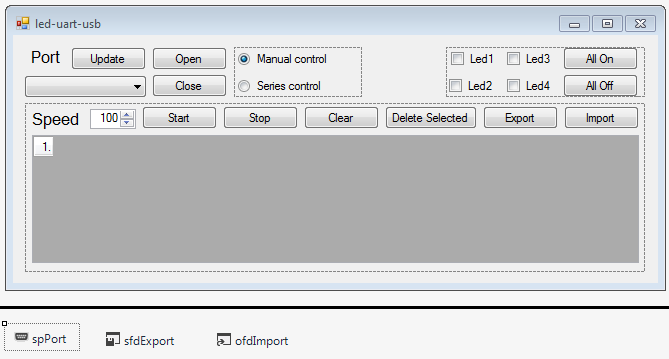


Рис. 8. Form1.cs

Код програми:

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;

using System.Threading;

using System.IO;

namespace led\_uart\_usb

{

public partial class Form1 : Form

{

private bool IsChangedAll { get; set; }

private Thread RunLineStatus { get; set; }

private Thread SearchPorts { get; set; }

private bool IsOpen { get; set; }

private bool IsSleepLineStatus { get; set; }

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

spPort.NewLine = "\r";

rbStatus.Checked = true;

dgvLineStatus.RowCount = 4;

IsSleepLineStatus = true;

IsOpen = false;

RunLineStatus = new Thread(new ThreadStart(Start));

RunLineStatus.Start();

SearchPorts = new Thread(new ThreadStart(Search));

SearchPorts.Start();

Clear();

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

try { RunLineStatus.Abort(); }

catch (Exception ex) { }

try { SearchPorts.Abort(); }

catch (Exception ex) { }

try

{

IsOpen = false;

spPort.Close();

}

catch (Exception ex) { }

}

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

{

spPort.PortName = (string)ddlPorts.SelectedValue;

try

{

if (spPort.IsOpen)

{

IsOpen = false;

spPort.Close();

}

}

catch (Exception ex) { }

try

{

IsOpen = true;

spPort.Open();

btnAllOff\_Click(sender, e);

btnUpdate.Enabled = false;

ddlPorts.Enabled = false;

btnOpen.Enabled = false;

btnClose.Enabled = true;

pControlMode.Enabled = true;

pStatus.Enabled = rbStatus.Checked;

pLineStatus.Enabled = rbLineStatus.Checked;

}

catch (Exception ex)

{

MessageBox.Show("Port unreachable!!!");

Clear();

}

}

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

{

Clear();

}

private void Clear()

{

try

{

btnUpdate.Enabled = true;

ddlPorts.Enabled = true;

btnOpen.Enabled = true;

btnClose.Enabled = false;

pControlMode.Enabled = false;

pStatus.Enabled = false;

pLineStatus.Enabled = false;

IsChangedAll = false;

IsSleepLineStatus = true;

try {

IsOpen = false;

spPort.Close();

}

catch (Exception ex) { }

btnStart.Enabled = true;

btnStop.Enabled = false;

btnClear.Enabled = true;

btnDeleteSelected.Enabled = true;

btnExport.Enabled = true;

btnImport.Enabled = true;

var ports = SerialPort.GetPortNames();

if (ports.Length == 0)

{

btnOpen.Enabled = false;

ports = new string[1] { "Not found" };

}

ddlPorts.DataSource = ports;

cbLed1.Checked = false;

cbLed2.Checked = false;

cbLed3.Checked = false;

cbLed4.Checked = false;

}

catch (Exception ex) { }

}

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

{

IsChangedAll = true;

cbLed1.Checked = true;

cbLed2.Checked = true;

cbLed3.Checked = true;

cbLed4.Checked = true;

Send("AllOn");

IsChangedAll = false;

}

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

{

IsChangedAll = true;

cbLed1.Checked = false;

cbLed2.Checked = false;

cbLed3.Checked = false;

cbLed4.Checked = false;

Send("AllOff");

IsChangedAll = false;

}

private void Send(string send)

{

try

{

Thread.Sleep(25);

spPort.WriteLine(send);

}

catch (Exception ex)

{

if (ex is ArgumentException || ex is InvalidOperationException || ex is TimeoutException)

{

MessageBox.Show("Serial port is not available!!!");

IsSleepLineStatus = true;

Clear();

}

else

{

throw;

}

}

}

private void cbLed1\_CheckedChanged(object sender, EventArgs e)

{

if (!IsChangedAll)

if (cbLed1.Checked)

Send("Port1=1");

else

Send("Port1=0");

}

private void cbLed2\_CheckedChanged(object sender, EventArgs e)

{

if (!IsChangedAll)

if (cbLed2.Checked)

Send("Port2=1");

else

Send("Port2=0");

}

private void cbLed3\_CheckedChanged(object sender, EventArgs e)

{

if (!IsChangedAll)

if (cbLed3.Checked)

Send("Port4=1");

else

Send("Port4=0");

}

private void cbLed4\_CheckedChanged(object sender, EventArgs e)

{

if (!IsChangedAll)

if (cbLed4.Checked)

Send("Port3=1");

else

Send("Port3=0");

}

private void dgvLineStatus\_SelectionChanged(object sender, EventArgs e)

{

if (dgvLineStatus.CurrentCell.ColumnIndex + 1 == dgvLineStatus.ColumnCount)

dgvLineStatus.Columns.Add(new DataGridViewCheckBoxColumn() { HeaderText = (dgvLineStatus.ColumnCount + 1).ToString() });

}

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

{

btnStart.Enabled = false;

btnStop.Enabled = true;

btnClear.Enabled = false;

btnDeleteSelected.Enabled = false;

btnExport.Enabled = false;

btnImport.Enabled = false;

for (int i = 0; i < dgvLineStatus.ColumnCount; i++)

dgvLineStatus.Columns[i].Selected = false;

IsSleepLineStatus = false;

dgvLineStatus.Enabled = false;

}

private void Start()

{

int n;

while (true)

{

for (int i = 0; i < dgvLineStatus.ColumnCount - 1; i++)

{

while (true) if (!IsSleepLineStatus) break;

dgvLineStatus.Rows[0].Cells[i].Selected = true;

for (int j = 0; j < dgvLineStatus.RowCount; j++)

{

n = j + 1;

if (n == 3) n = 4; else if (n == 4) n = 3;

Send(string.Format("Port{0}={1}", n, (bool)(dgvLineStatus.Rows[j].Cells[i].Value ?? false) ? 1 : 0));

}

Thread.Sleep((int)nudSpeed.Value);

dgvLineStatus.Rows[0].Cells[i].Selected = false;

}

}

}

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

{

IsSleepLineStatus = true;

btnStart.Enabled = true;

btnStop.Enabled = false;

btnClear.Enabled = true;

btnDeleteSelected.Enabled = true;

btnExport.Enabled = true;

btnImport.Enabled = true;

dgvLineStatus.Enabled = true;

}

private void rbStatus\_CheckedChanged(object sender, EventArgs e)

{

pStatus.Enabled = true;

pLineStatus.Enabled = false;

cbLed1.Checked = cbLed2.Checked = cbLed3.Checked = cbLed4.Checked = false;

btnAllOff\_Click(sender, e);

IsSleepLineStatus = true;

btnStart.Enabled = true;

btnStop.Enabled = false;

btnClear.Enabled = true;

btnDeleteSelected.Enabled = true;

btnExport.Enabled = true;

btnImport.Enabled = true;

dgvLineStatus.Enabled = true;

}

private void rbLineStatus\_CheckedChanged(object sender, EventArgs e)

{

pStatus.Enabled = false;

pLineStatus.Enabled = true;

cbLed1.Checked = cbLed2.Checked = cbLed3.Checked = cbLed4.Checked = false;

btnAllOff\_Click(sender, e);

IsSleepLineStatus = true;

btnStart.Enabled = true;

btnStop.Enabled = false;

btnClear.Enabled = true;

btnDeleteSelected.Enabled = true;

btnExport.Enabled = true;

btnImport.Enabled = true;

dgvLineStatus.Enabled = true;

}

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

{

if (sfdExport.ShowDialog() == DialogResult.OK)

{

using (StreamWriter sw = new StreamWriter(sfdExport.FileName))

{

string buffer;

for (int i = 0; i < dgvLineStatus.ColumnCount - 1; i++)

{

buffer = string.Empty;

for (int j = 0; j < dgvLineStatus.RowCount; j++)

{

buffer += (bool)(dgvLineStatus.Rows[j].Cells[i].Value ?? false) ? 1 : 0;

if (j + 1 != dgvLineStatus.RowCount) buffer += " ";

}

sw.WriteLine(buffer);

}

sw.Flush();

sw.Close();

}

}

}

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

{

if (ofdImport.ShowDialog() == DialogResult.OK)

{

try

{

using (StreamReader sr = new StreamReader(ofdImport.FileName))

{

dgvLineStatus.Columns.Clear();

dgvLineStatus.Columns.Add(new DataGridViewCheckBoxColumn() { HeaderText = (dgvLineStatus.ColumnCount + 1).ToString() });

dgvLineStatus.RowCount = 4;

dgvLineStatus.Columns.RemoveAt(1);

while (!sr.EndOfStream)

{

var buffer = sr.ReadLine().Split(new Char[1] { ' ' });

if (buffer.Length == 4)

{

for (int i = 0; i < dgvLineStatus.RowCount; i++)

{

dgvLineStatus.Rows[i].Cells[dgvLineStatus.ColumnCount - 1].Value = buffer[i] == "0" ? false : true;

}

dgvLineStatus.Columns.Add(new DataGridViewCheckBoxColumn() { HeaderText = (dgvLineStatus.ColumnCount + 1).ToString() });

}

}

if (dgvLineStatus.ColumnCount == 1)

dgvLineStatus.Columns.Add(new DataGridViewCheckBoxColumn() { HeaderText = (dgvLineStatus.ColumnCount + 1).ToString() });

sr.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Error: Could not read file from disk. Original error: " + ex.Message);

}

}

}

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

{

dgvLineStatus.Columns.Clear();

dgvLineStatus.Columns.Add(new DataGridViewCheckBoxColumn() { HeaderText = (dgvLineStatus.ColumnCount + 1).ToString() });

dgvLineStatus.RowCount = 4;

}

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

{

for (int i = dgvLineStatus.ColumnCount - 1; i >= 0; i--)

if (dgvLineStatus.Columns[i].Selected)

dgvLineStatus.Columns.RemoveAt(i);

for (int i = 0; i < dgvLineStatus.ColumnCount; i++)

{

dgvLineStatus.Columns[i].Selected = false;

dgvLineStatus.Columns[i].HeaderText = (i + 1).ToString();

}

}

private void Search ()

{

while (true)

{

if (!IsOpen)

{

var ports = SerialPort.GetPortNames();

if (ports.Length == 0)

{

btnOpen.Enabled = false;

ports = new string[1] { "Not found" };

}

else

btnOpen.Enabled = true;

ddlPorts.DataSource = ports;

}

Thread.Sleep(10000);

}

}

}

}

**Додаток Б**

Прошивка 1

$regfile = "attiny13.dat"

$crystal = 9600000

$hwstack = 16

$swstack = 16

$framesize = 16

Declare Sub Alloff()

Declare Sub Allon()

Config Timer0 = Pwm , Prescale = 8 , Compare B Pwm = Clear Down , Compare A Pwm = Clear Down

Start Timer0

Open "comb.2:4800,8,n,1" For Input As #2

Dim Pwmint As Byte

Dim Dat As String \* 8

Config Portb.0 = Output

Config Portb.3 = Output

Config Portb.4 = Output

Config Portb.1 = Output

Do

Input #2 , Dat

If Dat = "Port1PWM" Then

Input #2 , Pwmint

Pwm0b = Pwmint

End If

If Dat = "Port2PWM" Then

Input #2 , Pwmint

Pwm0a = Pwmint

End If

If Dat = "AllOff" Then Call Alloff()

If Dat = "AllOn" Then Call Allon()

If Dat = "Port1=1" Then Pwm0b = 255

If Dat = "Port1=0" Then Pwm0b = 0

If Dat = "Port2=1" Then Pwm0a = 255

If Dat = "Port2=0" Then Pwm0a = 0

If Dat = "Port3=1" Then Portb.3 = 1

If Dat = "Port3=0" Then Portb.3 = 0

If Dat = "Port4=1" Then Portb.4 = 1

If Dat = "Port4=0" Then Portb.4 = 0

Loop

End

Sub Allon()

Pwm0a = 255

Portb.3 = 1

Portb.4 = 1

Pwm0b = 255

End Sub

Sub Alloff()

Pwm0a = 0

Portb.3 = 0

Portb.4 = 0

Pwm0b = 0

End Sub

**Додаток В**

Прошивка 2

$regfile = "attiny13.dat"

$crystal = 9600000

$hwstack = 16

$swstack = 16

$framesize = 16

Open "comb.2:4800,8,n,1" For Input As #2

Dim Dat As Byte , M As Byte

Config Portb.0 = Output

Config Portb.3 = Output

Config Portb.4 = Output

Config Portb.1 = Output

Do

Input #2 , Dat

M = Dat Mod 2

Portb.0 = M

Dat = Dat / 2

M = Dat Mod 2

Portb.1 = M

Dat = Dat / 2

M = Dat Mod 2

Portb.3 = M

Dat = Dat / 2

M = Dat Mod 2

Portb.4 = M

Loop

End