Міністерство освіти та науки України

Національний університет “Львівська політехніка”



Лабораторна робота №2  
з курсу «Мікроконтролери Ч.1»

ПРОСТЕ КЕРУВАННЯ МІКРОКОНТРОЛЕРОМ ЗА

ДОПОМОГОЮ КЛІЄНТСЬКОЇ ПРОГРАМИ НА КОМП’ЮТЕРІ

Варіант 9

**Виконав:**

студент групи ІР-21

Касараба В.В.

**Прийняла:**

ст.викладач кафедри КСА

Лагун І. І.

Львів 2021

## Завдання

1. Згідно варіанту завдання (таблиця 1) зібрати у пакеті симуляції

Proteus схему на основі МК ATmega2560 та написати програму

мовою C++ в Arduino IDE для реалізації вказаного завдання.

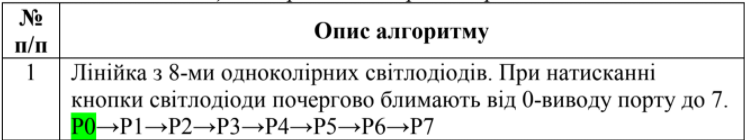
2. Розробити клієнтську програму для керування МК.

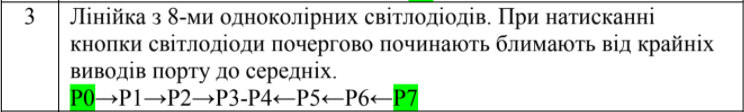
3. Протестувати роботу клієнтської програми зі схемою в Proteus.

4. Залити програму в Arduino-Kit.

5. Отримані результати представити викладачу.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | **Завдання**  1. Для зв’язку клієнтської програми з Proteus необхідно  встановити пару віртуальних COM-портів.  2. Запуск роботи алгоритмів зі світлодіодами виконується  шляхом натиску відповідних кнопок на клієнтській формі.  3. При натисканні кнопок на МК запускаються відповідні  алгоритми блимання іконок на клієнтській формі. | | | |
|  | **Кнопка** | **Світлодіоди** | **Затримка** | **Алгоритм** |
| 9 | PL0, PL2 | port-K | 0,8 сек. | 1, 3 |

****

****

**Відеопрезентація**

<https://drive.google.com/file/d/1oLUt2bMwcZo12ixZ9YrdRd9T1dEMEWSo/view?usp=sharing>

## Робочий процес

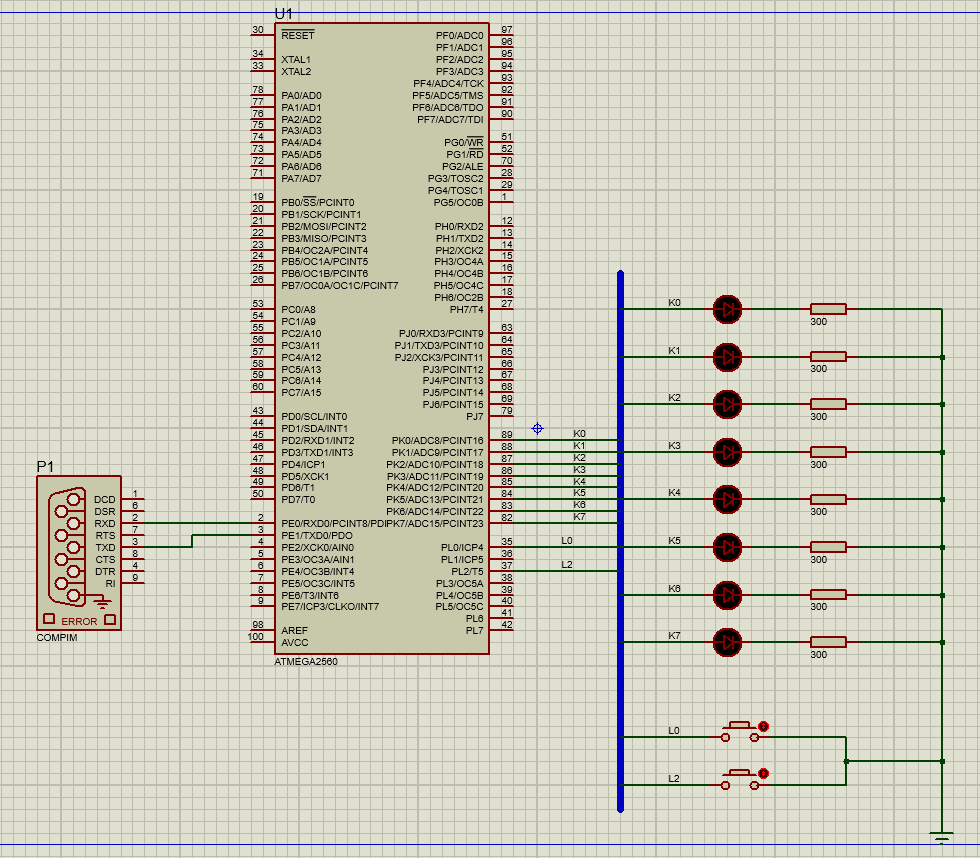


Рис. 1. Схема у пакеті модуляції Proteus

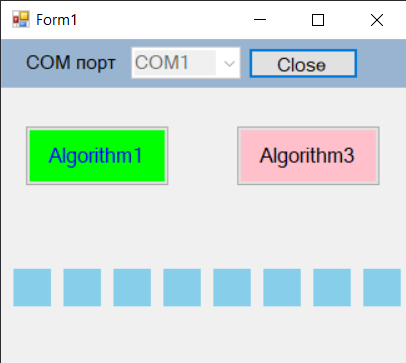


Рис. 2. Клієнська програма

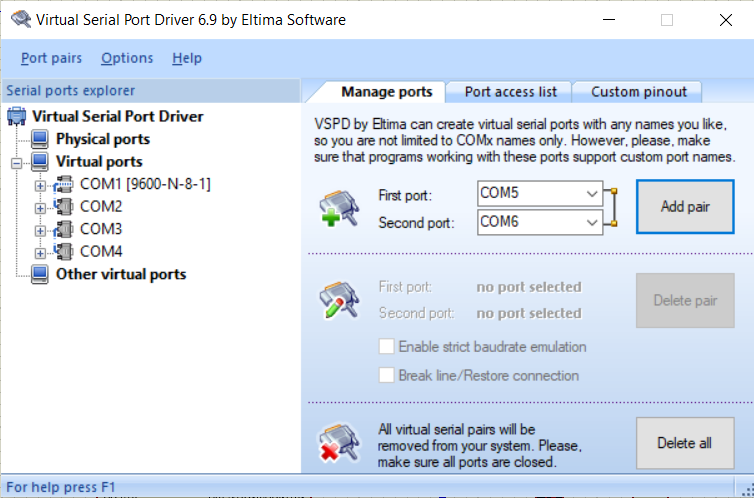


Рис. 3. Налаштування COM портів у Virtual Serial Port

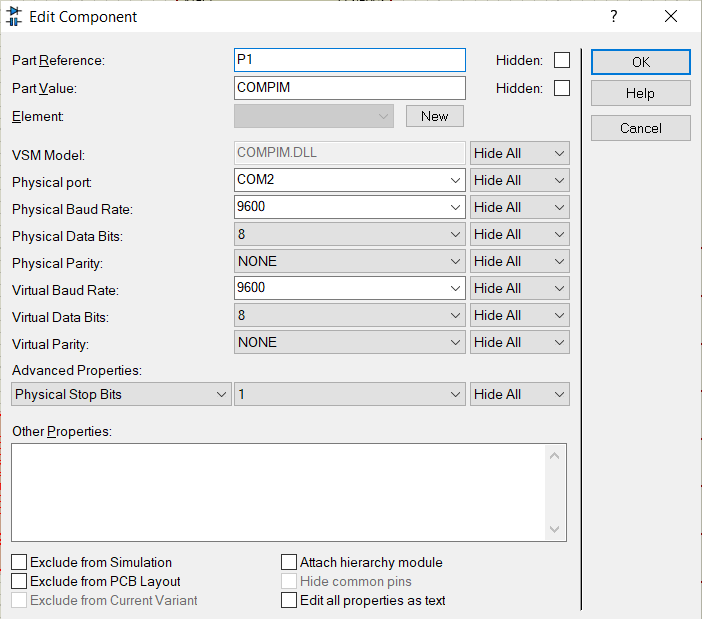
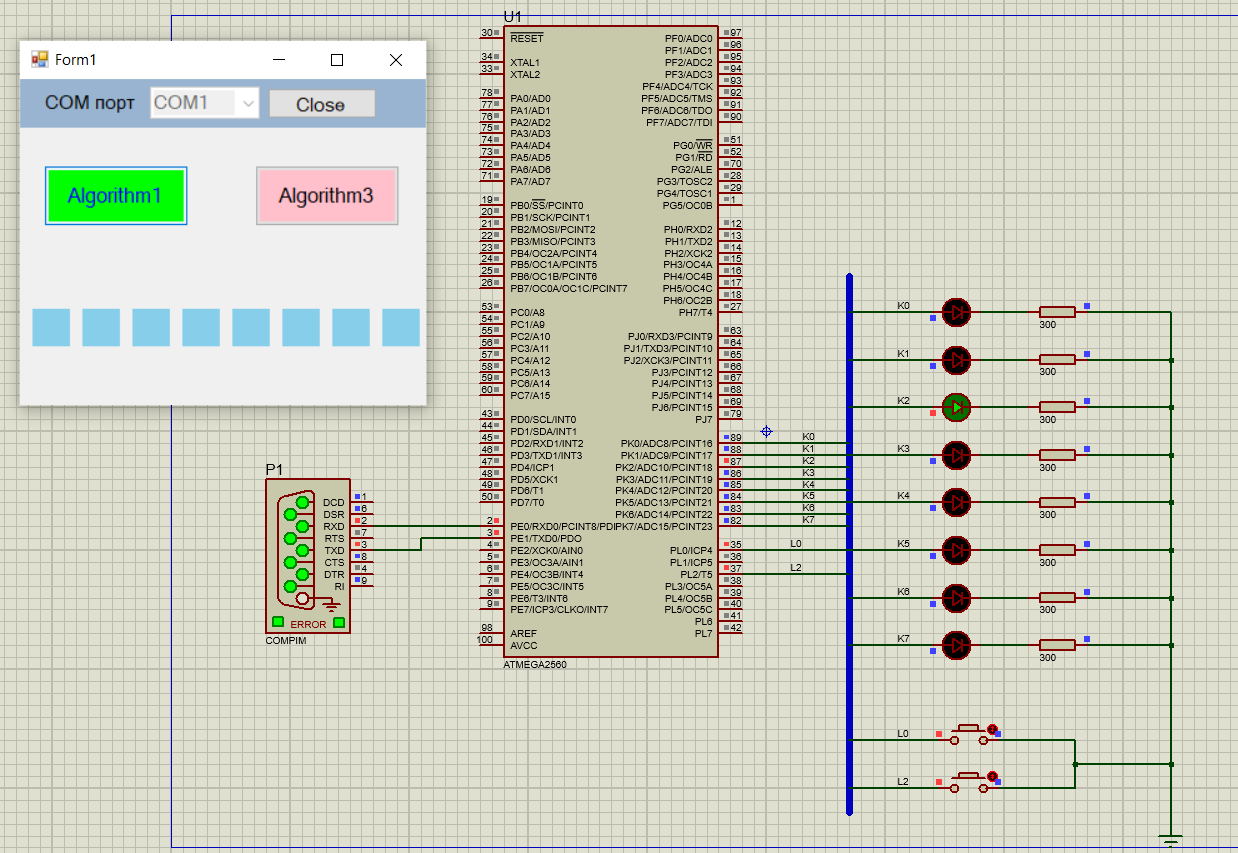
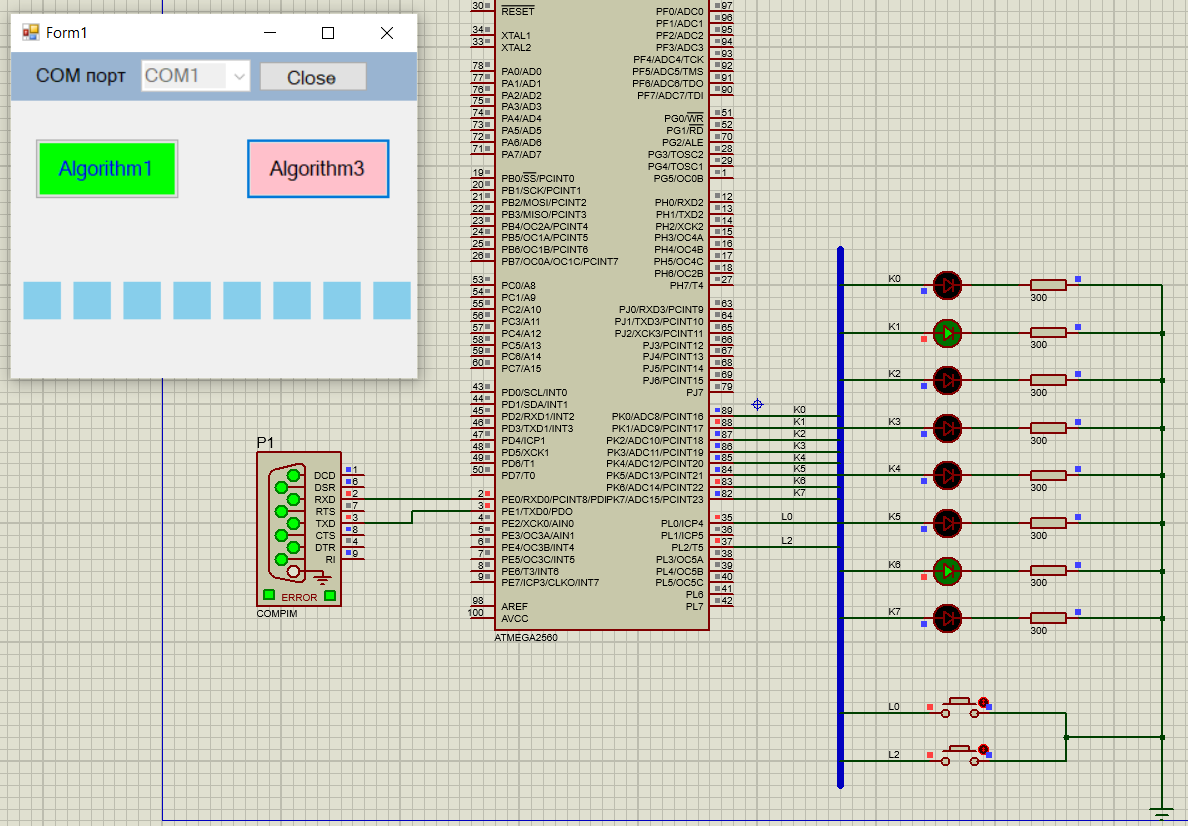


Рис. 4. Параметри протоколу обміну

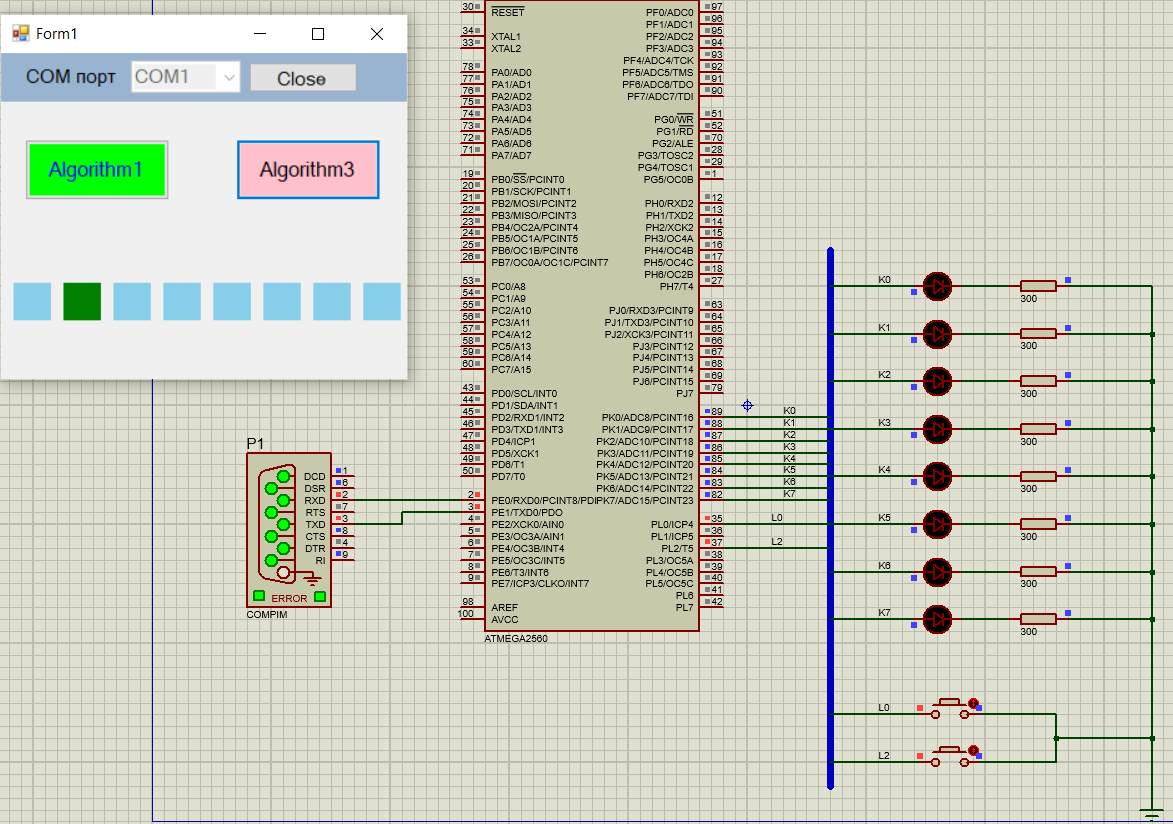
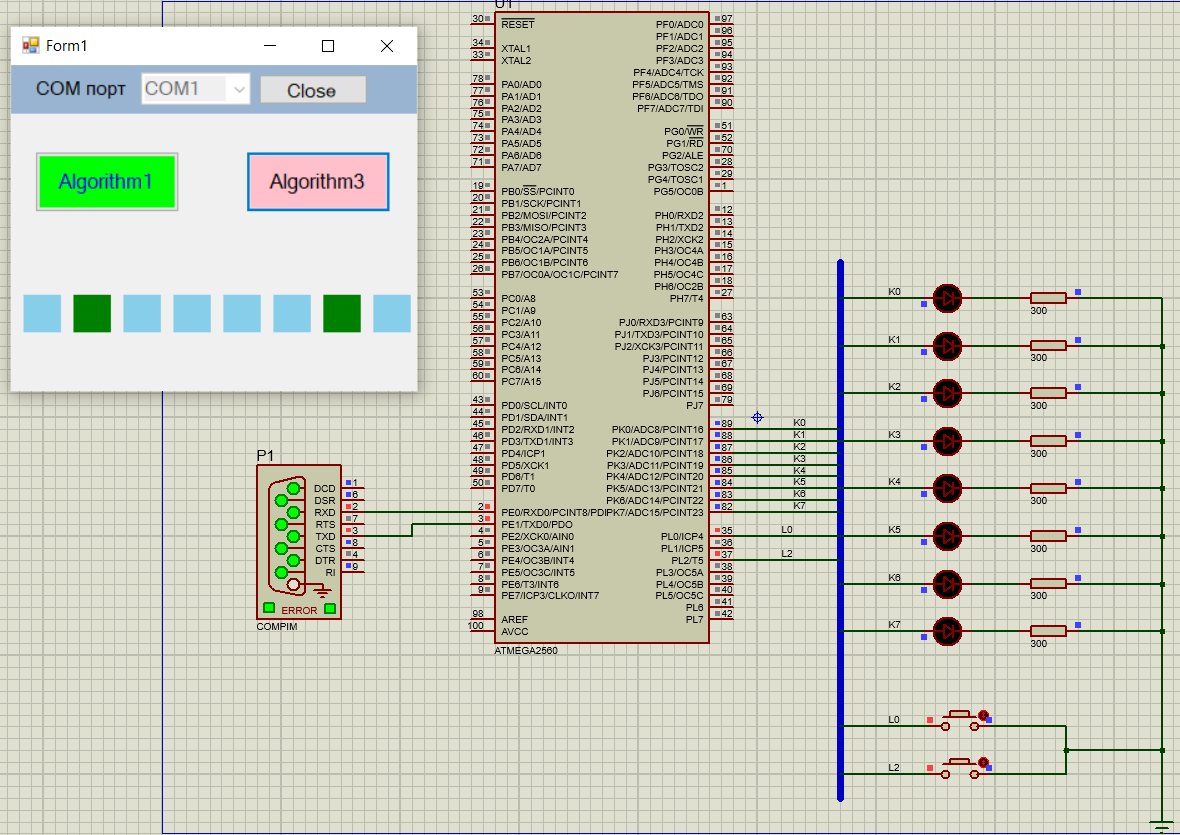
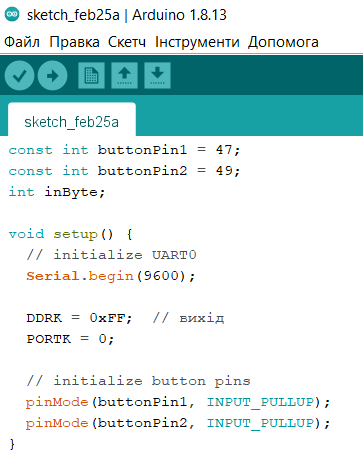
 

Рис. 5. Демонстрація роботи алгоритмів



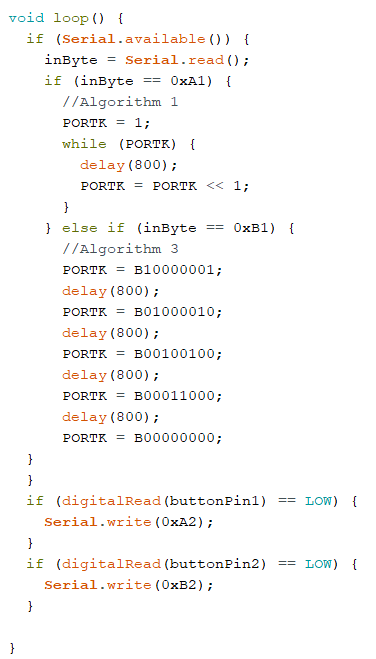


Рис. 6. Код програми мікроконтролера

Частина коду клієнської програми:

using System;

using System.Data;

using System.Linq;

using System.Windows.Forms;

using System.IO.Ports;

using System.Drawing;

using System.Threading;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

private int algorithmCount1 = 0;

private int algorithmCount2 = 0;

private int algorithmCount3 = 0;

private int algorithmNumber;

public Form1()

{

InitializeComponent();

}

private void comboBox1\_Click(object sender, EventArgs e)

{

int num;

comboBox1.Items.Clear();

string[] ports = SerialPort.GetPortNames().OrderBy(a => a.Length > 3 && int.TryParse(a.Substring(3), out num) ? num : 0).ToArray();

comboBox1.Items.AddRange(ports);

}

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

{

if (!serialPort1.IsOpen)

try

{

serialPort1.PortName = comboBox1.Text;

serialPort1.Open();

buttonOpenPort.Text = "Close";

comboBox1.Enabled = false;

button1.Visible = true;

button2.Visible = true;

}

catch

{

MessageBox.Show("Port " + comboBox1.Text + " is invalid!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

serialPort1.Close();

buttonOpenPort.Text = "Open";

comboBox1.Enabled = true;

button1.Visible = false;

button2.Visible = false;

}

}

private void button1\_Click(object sender, EventArgs e)

{

byte[] b1 = new byte[1];

b1[0] = 0xA1;

serialPort1.Write(b1, 0, 1);

}

private void button2\_Click(object sender, EventArgs e)

{

byte[] b1 = new byte[1];

b1[0] = 0xB1;

serialPort1.Write(b1, 0, 1);

}

private void clearAllLed()

{

panel1.BackColor = Color.SkyBlue;

panel2.BackColor = Color.SkyBlue;

panel3.BackColor = Color.SkyBlue;

panel4.BackColor = Color.SkyBlue;

panel5.BackColor = Color.SkyBlue;

panel6.BackColor = Color.SkyBlue;

panel7.BackColor = Color.SkyBlue;

panel8.BackColor = Color.SkyBlue;

}

private void startTimer1()

{

timer2.Stop();

timer1.Start();

}

private void startTimer2()

{

timer1.Stop();

timer2.Start();

}

private void timer1\_Tick(object sender, EventArgs e)

{

Panel[] panels = { panel1, panel2, panel3, panel4, panel5, panel6, panel7, panel8 };

clearAllLed();

if(algorithmCount1 < 8)

{

panels[algorithmCount1].BackColor = Color.Green;

}

algorithmCount1++;

}

private void timer2\_Tick(object sender, EventArgs e)

{

Panel[] panels = { panel1, panel2, panel3, panel4, panel5, panel6, panel7, panel8 };

clearAllLed();

if (algorithmCount2 < 4)

{

panels[algorithmCount2].BackColor = Color.Green;

panels[7-algorithmCount2].BackColor = Color.Green;

}

algorithmCount2++;

}

private void serialPort1\_DataReceived(object sender, SerialDataReceivedEventArgs e)

{

byte commandFromArduino = (byte)serialPort1.ReadByte();

if (commandFromArduino == 0xA2)

{

algorithmCount1 = 0;

this.BeginInvoke(new ThreadStart(startTimer1));

}

if (commandFromArduino == 0xB2)

{

algorithmCount2 = 0;

algorithmCount3 = 7;

this.BeginInvoke(new ThreadStart(startTimer2));

}

}

}

}

## Висновок

На цій лабораторній роботі я навчився керувати мікроконтролером з допомогою клієнської аплікації на комп’ютері.

Спочатку у програмі Proteus я збудував схему на базі мікроконтролера Arduino ATmega2560. До мікроконтролера я підключив LED діоди, кнопки, та COMPIM, використавши порти, вказані у можму варіанті.

Клієнська аплікація реалізована на мові C#, на графічний інтерфейс я добавив кнопки для запуску алгоритмів та панелі, що засвічуються відповідно до алгоритмів.

Використавши додаток Arduino, я написав код алгоритмів та описав зв’язок між мікроконтролером та графічним інтерфейсом на мові С та експортував вихідний бінарник у директорію з побудованою схемою.

Для зв’язку клієнтської програми з Proteus я встановив пару віртуальних COM-портів з допомогою програми Configure Virtual Serial Port.