**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Дисциплина «Программирование и алгоритмизация на языках выскокого уровня»

Отчет о лабораторной работе №5

«Использование графических возможностей приложении на C#»

Вариант № 6

Работу выполнил\_и

студент\_ка 3 курса

очного отделения

<ФИО>

Научный руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

<ФИО>

Москва, 2019

**Задание**

Тема: Использование графических возможностей приложении на C#.

Разработать приложение по обработке данных и визуализации данных пользователя.

Подготовить приложение на языке C# (Windows Forms) по работе с файлами и базой данных (через ADO) для хранения и обработки данных измерений. Реализовать хранение собранных данных в таблице БД и вывод (экспорт) данных за указанный период в текстовый файл.

Реализовать отрисовку данных за указанный период на форме.

Диапазон для данных (множители амплитуды и времени) должны указываться через текстовые поля. Реализовать генератор данных изменений (через датчик случайных чисел) через отдельную форму с указанием различных параметров, вроде амплитуды, периодичности сигнала, длительности изменений. Введенные пользователем данные нужно проверять на допустимость.

**Индивидуальное задание**

L05\_6. Создайте систему мониторинга токовой нагрузки на предприятии

1. Разработать приложение для мониторинга потребления электроэнергии на предприятии:

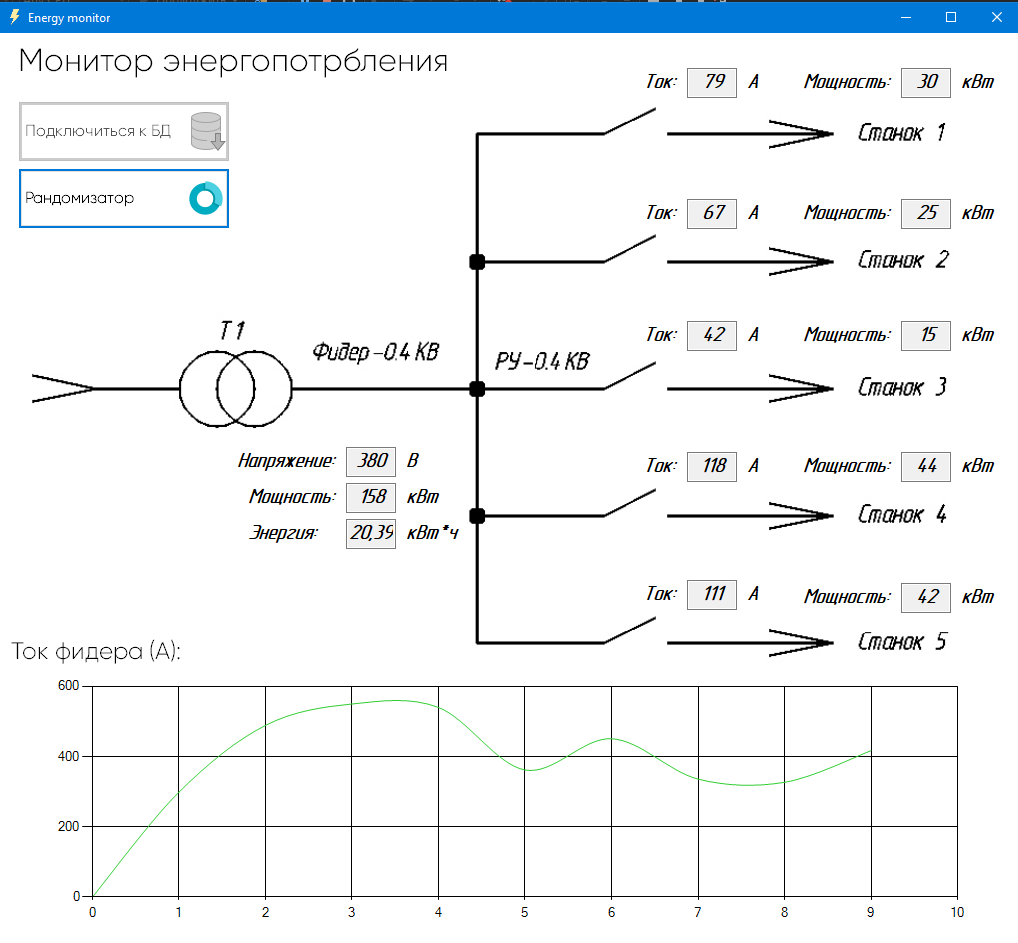
* Сбор данных о токе, протекающем через распределительную систему (5 станков с потреблением 10-50 кВт, 380 В, поминутный мониторинг);
* Вычисление мгновенной мощности и интегрирование для расчёта энергопотребления;
* Расчет падения напряжения на входном фидере при превышении порога потребления (200 кВт).

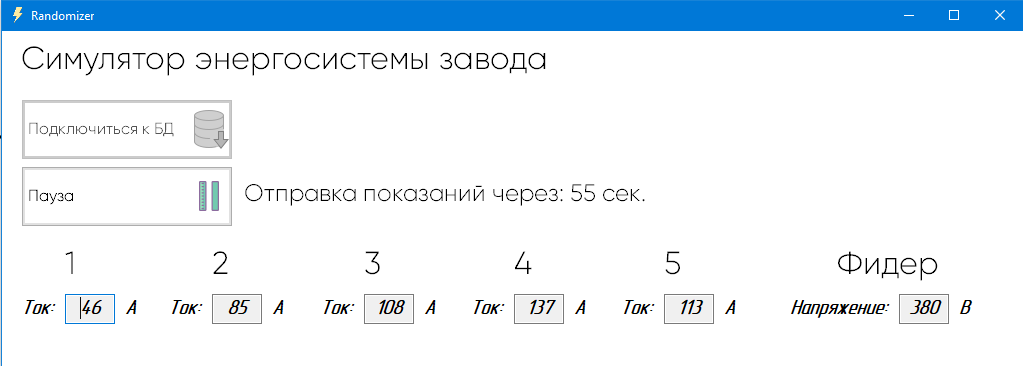
2. Реализовать сбор данных об энергопотреблении с сохранением в файл или БД.

3. (Дополнительно): Реализовать алгоритм интегрального изменения потребляемого станками тока (при генерации) и отображения на графике нештатных условий работы оборудования (выделить красным).

**Ход выполнения работы**

Версия *.NET Framework*: **4.5.2.**

Было написано GUI-приложение по индивидуальному заданию на языке C#.



Для хранения данных, была создана база данных с 1 таблицей. Колонки: токи с 5 станков, текущее напряжение на фидере.

В качестве отправителя, была написана 2 форма (симулятор энергосистемы завода). Форма, согласно заданию, раз в минуту генерирует 5 случайных токов в указанном диапазоне, и, на основе задания вычисляет условное напряжение (падение напряжения) на фидере. Все 6 значений, а также, текущее время (для вычисления энергопотребления (кВт\*ч)) записываются в базу данных в виде стека, растущего вниз. Отправка данных, работа таймера и генерация случайных значений выполняются в отдельном потоке для предотвращения «замораживания» формы.

Основная форма, также, с помощью фонового потока, через определенный промежуток времени, работает с базой данных. Считывается количество строк в базе, если оно увеличилось (прибавилась строка, «стек вырос вниз»), происходит сброс таймера и вычисление значений.

Из базы данных считываются 5 токов, каждый ток умножается на напряжение на фидере для вычисления мгновенной мощности.

Для вычисления энергопотребления (Квт\*ч), используется следующий алгоритм: из базы данных извлекается время, вычисляется промежуток времени в часах, умножается на мгновенную общую мощность и прибавляется к предыдущему энергопотреблению. Данный принцип используется в электросчётчиках.

Так же, при появлении новой строки, дорисовывается график общего тока. Также, согласно заданию, если один из станков стал потреблять выше заявленного тока, график становится красным. По оси X откладывается «счётчик» обновления данных, по оси Y – общий ток через фидер в (А).

**Использованные SQL команды**

* SELECT \* FROM <таблица> -Выбор всей таблицы
* SELECT COUNT(\*) FROM <таблица> -Используется для подсчёта количества строк в таблицe
* INSERT INTO <таблица> (<столбец 1>, <столбец 2>) VALUES (<значение 1>, <значение 1>) -Добавление строки в таблицу

**Исходной код основной формы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace CSharp\_LR5

{

public partial class Form1 : Form

{

/// <summary>

/// Переменные

/// </summary>

private OleDbConnection connection; //Подключение к БД

private long index = 0; //Индекс последней (нижней) строки БД

private long feeder\_current = 0; //Общий ток (фидера)

private Thread t; //Фоновый поток

private DateTime DTstarted; //Время предыдущей регистрации параметров

private bool timeStarted = false; //"Зещёлка", начали ли мы считать время?

private double HoursDiff = 0; //Разница между показаниями в часах

private double energyCounter = 0; //Квт\*ч

private long chartCounter = 0; //Счётчик для графика

private int cycleDelay = 4000; //Задержка в цикле

private int maxCurrent = 130; //Максимальный "штатный" ток для 1 станка

/// <summary>

/// Компоненты формы

/// </summary>

public Form1()

{

InitializeComponent();

}

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

{

//

//Инициализируем график при загрузке формы

//

/\*for (int i = 0; i < 6; i++)

{

chart1.Series[i].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[i].Points.AddXY(0, 0);

}\*/

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[0].Points.AddXY(0, 0);

chart1.ChartAreas[0].AxisX.Minimum = 0;

}

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

{

//

//При закрытии формы, убиваем процесс и отключаемся от БД

//

if (t != null && t.IsAlive)

t.Abort();

if (connection != null)

connection.Close();

Environment.Exit(0); //Убиваем все фоновые процессы

}

private void db\_btn\_Click(object sender, EventArgs e)

{

//

//Кнопка "Подключиться к БД"

//

connection = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; Data Source =" + Path.Combine(AppDomain.CurrentDomain.BaseDirectory, @"..\..\LR05.mdb"));

try

{

this.connection.Open(); //Открываем соединение

table1\_load(); //Пробуем прочитать таблицу

index = countRows(); //Ищем низ БД (последняя запись)

t = new Thread(new ThreadStart(ThreadProc)); //Запускам фоновый процесс

t.Start();

MessageBox.Show("Подключено!");

db\_btn.Enabled = false; //Делаем кнопку "подключиться" некликабельной

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message); //Сообщение об ошибке

}

}

private void random\_btn\_Click(object sender, EventArgs e)

{

//

//Форма генератора случайных чисел

//

if (!CheckOpened("Randomizer")) //Проверяем, не открыта ли форма

{

Randomizer randomizer = new Randomizer();

randomizer.Show(); //Если нет, открываем

}

}

/// <summary>

/// Остальные финкции

/// </summary>

/// <returns></returns>

private bool CheckOpened(string name)

{

//

//Проверка "формы на открытость"

//

FormCollection fc = Application.OpenForms;

foreach (Form frm in fc)

{

if (frm.Text == name)

{

return true;

}

}

return false;

}

private void ThreadProc()

{

//

//Фоновый процесс

//

try

{

while (true) //Бесконечный цекл

{

long curIndex = countRows(); //Проверяем, не выросла ли БД

if (curIndex > index) //Если строк стало больше

{

var table\_1 = table1\_load(); //Выгружаем таблицу

long k = 0;

while (k < curIndex) //Доходим до последней строки

{

k++;

table\_1.Read();

}

if (!timeStarted) //"Защёлка" времени для энергосчётчика

{

timeStarted = true;

DTstarted = (DateTime)table\_1[7]; //Время начала отсчёта

}

//Считаем разницу во времени (в часах) (от последнего времени отнимаем предыдущее)

DateTime DTcurrent = (DateTime)table\_1[7];

HoursDiff = DTcurrent.Subtract(DTstarted).TotalHours;

DTstarted = DTcurrent;

//Вычисляем общий ток

feeder\_current = 0;

for (int i = 1; i < 6; i++)

{

feeder\_current += (int)table\_1[i];

}

fillFields(table\_1); //Заполняем основные поля

fillChart(table\_1); //Чертим график

index = curIndex; //Запоминаем последнюю строку

Thread.Sleep(cycleDelay); //Ждём

}

}

}

catch (ThreadAbortException)

{

//

//Процесс остановлен

//

}

}

private void fillChart(OleDbDataReader table\_1)

{

//

//Отрисовка графика

//

BeginInvoke(new Action(() =>

{

/\*for (int i = 0; i < 5; i++) //Основные токи

chart1.Series[i].Points.AddXY(chartCounter, (int)table\_1[i + 1]);\*/

//Общий ток

chart1.Series[0].Points.AddXY(chartCounter, (int)feeder\_current);

bool error = false; //Проверяем на превышение токов

for (int i = 1; i < 6; i++)

if ((int)table\_1[i] > maxCurrent)

error = true;

if (error) //Если хоть где-то больше maxCurrent, красим красным

chart1.Series[0].Color = Color.Red;

else

chart1.Series[0].Color = Color.LimeGreen;

}));

chartCounter++;

}

private void fillFields(OleDbDataReader table\_1)

{

//

//Заполнение полей

//

BeginInvoke(new Action(() => { //Токи

for (int i = 1; i < 6; i++)

{

System.Windows.Forms.TextBox tb = this.Controls.Find("amps\_" + i.ToString(), true).FirstOrDefault() as System.Windows.Forms.TextBox;

tb.Text = table\_1[i].ToString();

}

}));

BeginInvoke(new Action(() => { //Мгновенные мощности

for (int i = 1; i < 6; i++)

{

System.Windows.Forms.TextBox tb = this.Controls.Find("power\_" + i.ToString(), true).FirstOrDefault() as System.Windows.Forms.TextBox;

tb.Text = (((int)table\_1[i] \* (int)table\_1[6]) / 1000).ToString();

}

}));

//Напряжение на фидере

string volts = table\_1[6].ToString();

if (volts\_feeder.InvokeRequired)

volts\_feeder.BeginInvoke(new Action(() => { volts\_feeder.Text = volts; }));

else

volts\_feeder.Text = volts;

//Мощность на фидере

string fPower = ((feeder\_current \* (int)table\_1[6]) / 1000).ToString();

if (power\_feeder.InvokeRequired)

power\_feeder.BeginInvoke(new Action(() => { power\_feeder.Text = fPower; }));

else

power\_feeder.Text = fPower;

//кВт\*ч

if (timeStarted)

{

energyCounter += ((feeder\_current \* (int)table\_1[6]) / 1000.0) \* HoursDiff;

string energy = new String(energyCounter.ToString().ToCharArray());

if (energy\_feeder.InvokeRequired)

energy\_feeder.BeginInvoke(new Action(() => { energy\_feeder.Text = energy; }));

else

energy\_feeder.Text = energy;

}

}

private OleDbDataReader table1\_load()

{

//

//Чтение всей таблицы

//

OleDbCommand cmd;

cmd = new OleDbCommand(

"SELECT \* FROM table\_1", this.connection

);

return cmd.ExecuteReader();

}

private long countRows()

{

//

//Подсчёт строк

//

OleDbCommand cmd;

cmd = new OleDbCommand(

"SELECT COUNT(\*) FROM table\_1", this.connection

);

return (int)cmd.ExecuteScalar();

}

}

}

**Исходной код генератора случайных значений**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CSharp\_LR5

{

public partial class Randomizer : Form

{

/// <summary>

/// Переменные

/// </summary>

private OleDbConnection connection; //Подключение к БД

private Thread t; //Фоновый поток

private int seconds = 0; //Таймер на отправку

private int[] values = { 0, 0, 0, 0, 0, 0}; //Рандомные значения

private int sendingTime = 60; //Интервал отправки (сек)

/// <summary>

/// Компоненты формы

/// </summary>

public Randomizer()

{

InitializeComponent();

}

private void start\_btn\_Click(object sender, EventArgs e)

{

//

//Кнопка "старт"

//

start\_btn.Visible = false; //Убираем кнопку с глаз долой

pause\_btn.Visible = true; //Включаем кнопку паузы

t = new Thread(new ThreadStart(ThreadProc)); //Запускам фоновый процесс

t.Start();

}

private void pause\_btn\_Click(object sender, EventArgs e)

{

//

//Кнопка "пауза"

//

start\_btn.Visible = true;

pause\_btn.Visible = false; //Прячем кнопки

t.Abort(); //Убиваем процесс

}

private void db\_conn\_btn\_Click(object sender, EventArgs e)

{

//

//Кнопка "подключиться к БД"

//

connection = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; Data Source =" + Path.Combine(AppDomain.CurrentDomain.BaseDirectory, @"..\..\LR05.mdb"));

try

{

this.connection.Open(); //Открываем подключение

table1\_load(); //Пробуем выгрузить таблицу

start\_btn.Visible = true; //Делаем кнопку старта видимой

MessageBox.Show("Подключено!"); //Выводим сообщение

db\_conn\_btn.Enabled = false; //Выключаем кнопку подключения

}

catch (Exception ex)

{

MessageBox.Show("Error while loading database: " + ex.Message);

}

}

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

{

//

//При загрузки формы генерируем первые случайные числа

//

updateRandom();

}

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

{

//

//При закрытии формы, убиваем процесс и отключаемся от БД

//

if (t != null && t.IsAlive)

t.Abort();

if (connection != null)

connection.Close();

}

/// <summary>

/// Остальные финкции

/// </summary>

/// <returns></returns>

private OleDbDataReader table1\_load()

{

//

//Чтение всей таблицы

//

OleDbCommand cmd;

cmd = new OleDbCommand(

"SELECT \* FROM table\_1", this.connection

);

return cmd.ExecuteReader();

}

private void ThreadProc()

{

//

//Фоновый процесс

//

try

{

while (true) //Бесконечный цикл

{

if (seconds == 0) //Если время вышло, отправляем

{

//

//Генерируем запрос

//

string sql = "INSERT INTO table\_1 (machine\_1, machine\_2, machine\_3, machine\_4, machine\_5, feeder, DT)" +

" VALUES (?, ?, ?, ?, ?, ?, ?);";

OleDbCommand cmd = new OleDbCommand();

cmd.CommandText = sql;

cmd.Connection = this.connection;

for (int i = 0; i < 5; i++) //Токи

{

cmd.Parameters.Add("machine\_" + i.ToString(), OleDbType.Integer).Value = values[i];

}

//Напряжение на фидере

cmd.Parameters.Add("feeder", OleDbType.Integer).Value = values[5];

//текущее время

cmd.Parameters.Add("DT", OleDbType.Date).Value = DateTime.Now;

cmd.ExecuteNonQuery();

seconds = sendingTime; //Ставим таймер

updateRandom(); //Обновляем рандомные значения

}

if (timer.InvokeRequired) //Вывод текста тамера (обратный отсчёт)

timer.BeginInvoke(new Action(() => { timer.Text = "Отправка показаний через: " + seconds.ToString() + " сек."; }));

else

timer.Text = "Отправка показаний через: " + seconds.ToString() + " сек.";

Thread.Sleep(1000); //Каждую секунду выводим текст и проверяем таймер

seconds--; //Вычитаем таймер

}

}

catch (ThreadAbortException)

{

//

//Процесс остановлен

//

}

}

private void updateRandom()

{

//

//Генератор рандрмных значений

//

Random rnd = new Random();

for (int i = 0; i < 5; i++) //Токи

values[i] = rnd.Next(26, 140);

//Вычисляем уловное напряжение на фидере

int totalCurrent = 0;

for (int i = 0; i < 5; i++) //Общий ток

totalCurrent += values[i];

long totalPower = (totalCurrent \* 380) / 1000; //Условная общая мощность

//Считаем падение напряжения

if (totalPower > 200)

values[5] = (int)map(totalCurrent, 200, 500, 380, 300);

else

values[5] = 380;

BeginInvoke(new Action(() =>

{ //Выодим в поля

for (int i = 0; i < 5; i++)

{

System.Windows.Forms.TextBox tb = this.Controls.Find("amps\_" + (i + 1).ToString(), true).FirstOrDefault() as System.Windows.Forms.TextBox;

tb.Text = new String(values[i].ToString().ToCharArray());

}

}));

string volts = values[5].ToString(); //Выводим в поле напряжение на фидере

if (voltage\_feeder.InvokeRequired)

voltage\_feeder.BeginInvoke(new Action(() => { voltage\_feeder.Text = volts; }));

else

voltage\_feeder.Text = volts;

}

private long map(long x, long in\_min, long in\_max, long out\_min, long out\_max)

{

//

//Ардуиновская функция map

//

return (x - in\_min) \* (out\_max - out\_min) / (in\_max - in\_min) + out\_min;

}

}

}