AKADEMIA MORSKA W SZCZECINIE

JEDNOSTKA ORGANIZACYJNA:
WYDZIAŁ NAWIGACYJNY, CENTRUM INŻYNIERII RUCHU MORSKIEGO

PRZEWODNIK METODYCZNY

Symulatory Morskie

Laboratorium

Opracował:	mgr inż. Bilewski Mateusz
Zatwierdził:	dr inż. Gucma Maciej
Obowiązuje od: 2012/2013	

RAMOWY SPIS TREŚCI

1. CEL I ZAKRES PRZEDMIOTU

Przekazanie wiedzy z zakresu morskich systemów zintegrowanych (MSZ), w tym systemów mostka zintegrowanego (ang. Integrated Bridge System) – IBS.

2. PROGRAM PRZEDMIOTU

2.1. Program zajęć, regulaminy: BHP, P.POŻ., laboratorium, przykładowy program w języku C#;
2.2. Klasy, metody, funkcje - elementy symulatora;
2.3. Dziedziczenie - przechowywanie danych jednostek pływających;
2.4. Wielowątkowość - problematyka wielu jednostek w jednym symulatorze;
2.5. Obsługa wyjątków - błąd jako zaplanowane rozszerzenie logiki symulatora;
2.6. Komunikacja UDP – zastosowanie w symulatorze;
2.7. Komunikacja TCP/IP – zastosowanie w symulatorze;
2.8. Interpretacja podstawowych modeli hydrodynamicznych;
2.9. Obsługa zdarzeń – interfejs symulatora;
2.10. Delegaty i zdarzenia – logika dla aplikacji okienkowych;
2.11. Generowanie wiadomości NMEA 0183;
2.12. Parser NMEA 0183;
2.13. Aplikacja mapowa typu ECDIS;
2.14. Zaliczenie praktyczne, protokół Modbus;
2.15. Poprawa zaliczenia praktycznego, własne kontrolki .

3. PRZEBIEG ĆWICZEŃ

3.1. Zajęcia nr 1

PROGRAM KOMPUTEROWY:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

namespace Symulator
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            while (true)
            {
                Console.WriteLine(i++);
                Thread.Sleep(100);
            }
        }
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Przeanalizuj program.

Zmień częstotliwość wyświetlania kolejnych wartości.

Zmień program tak, aby wyświetlał czas w sekundach od uruchomienia programu z dokładnością do 100ms.

ZADANIA (ROK III - poziom rozszerzony):

Zmodyfikuj program tak, aby wyświetlał bieżącą godzinę oraz czas od uruchomienia programu.

3.2. Zajęcia nr 2

PROGRAM KOMPUTEROWY:

##################PLIK:vector.cs###############

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;

namespace Symulator
{
    class Vector
    {
        #region Zmienne lokalne
        private eMode mode;
        private Color color;
        private int width;
        private double length;
        private double tempLength;
        private double course;
        private Point position;

        public Vector(Color _color,int _width,double _length, double _course, Point _position)
        {

            color = _color;
            width = _width;
            length = _length;
            course = _course;
            position = _position;
        }
        #endregion Zmienne lokalne

        #region Zmiana parametrów wprost
        /// <summary>
        /// Return display mode.
        /// </summary>
        /// <returns></returns>
        public eMode Mode()
        {
            return mode;
        }

        /// <summary>
        /// Change display mode.
        /// </summary>
        /// <param name="_mode">New mode.</param>
        public void Mode(eMode _mode)
        {
            mode = _mode;
        }

        public void SaveLength()
        {
            tempLength = length;
        }

        public void LoadLength()
        {
            length = tempLength;
        }

        public void ChangeLength(double _length)
        {
            length = _length;
        }

        public double Length()
        {
            return length;
        }
```

```
        public double TempLength()
        {
            return tempLength;
        }

        public Point Position()
        {
            return position;
        }

        public double Course()
        {
            return course;
        }

        public void ChangeCourse(Double _course, bool _delta)
        {
            if (_delta) course = (course + _course) % (2 * Math.PI);
            else course = _course % (2 * Math.PI);
        }

        #endregion Zmiana parametrów wprost

        #region Różne funkcje

        public static double CalcDistancePow(Point A, Point B)
        {
            return (A.X - B.X) * (A.X - B.X) + (A.Y - B.Y) * (A.Y - B.Y);
        }
        #endregion Różne funkcje
    }
}
```

##################PLIK:ship.cs##############

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;

namespace Symulator
{
    enum eMode { normal, changePosition, changeCourse, other, selected};
    class Ship
    {

        #region Zmienne lokalne
        private Point position;        //pozycja statku w pikselach
        private Double course;         //kierunek dziobu względem północy <0-2pi)
        private PointList waterline;   //zbiór punktów wodnicy w pikselach
        private eMode mode;            //tryb
        private static int countid = 0;
        private int id;
        private String name="No name";
        #endregion

        #region Constructors
        /// <summary>
        /// Create ship.
        /// </summary>
        /// <param name="_tgis">Reference to tgis-window.</param>
        /// <param name="_waterline">Waterline in pointlist.</param>
        /// <param name="_position">Starting position.</param>
        /// <param name="_course">Starting course.</param>
        public Ship(PointList _waterline, Point _position, Double _course)
        {
            id = countid++;
            position = _position;
            course = _course;
            waterline = _waterline;
            mode = eMode.normal;
        }
        #endregion

        #region Zmiana parametrów wprost
```

```
/// <summary>
/// Return position.
/// </summary>
/// <returns></returns>
public Point Position()
{
    return position;
}

/// <summary>
/// Return course in radians.
/// </summary>
/// <returns></returns>
public Double Course()
{
    return course;
}

/// <summary>
/// Return display mode.
/// </summary>
/// <returns></returns>
public eMode Mode()
{
    return mode;
}

/// <summary>
/// Change display mode.
/// </summary>
/// <param name="_mode">New mode.</param>
public void Mode(eMode _mode)
{
    mode = _mode;
}

/// <summary>
/// Return ID.
/// </summary>
/// <returns></returns>
public int Id()
{
    return id;
}
#endregion

#region Funkcje zmieniające parametry
/// <summary>
/// Change course of the ship.
/// </summary>
/// <param name="_Course">Angle.</param>
/// <param name="_delta">Add to old course.</param>
public void Rotate(Double _course, bool _delta)
{
    if(_delta) course=(course+_course)%(2*Math.PI);
    else course = _course % (2 * Math.PI);
}

/// <summary>
/// Change position of the ship.
/// </summary>
/// <param name="_position">Position or distance.</param>
/// <param name="_delta">Add to old position.</param>
public void Move(Point _position, bool _delta)
{
    if (_delta)
    {
        position.X += _position.X;
        position.Y += _position.Y;
    }
    else position = _position;
}
```

```
        /// <summary>
        /// Change position of the ship.
        /// </summary>
        /// <param name="_meters">O ile metrów przesunać</param>
        public void Move(double _meters)
        {
            position.X += _meters * Math.Sin(course);
            position.Y += _meters * Math.Cos(course);
        }
        #endregion
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

W programie zamieszczono dwa pliki z klasami odpowiedzialnymi za przechowywanie danych o wektorach oraz jednostkach.
Zaimplementuj dwie jednostki i dwa wektory.
Utwórz funkcje, które wyświetlą bieżącą wartość pozycji.

ZADANIA (ROK III - poziom rozszerzony):

Zaimplementuj zmianę pozycji zgodnie z wybranym modelem hydrodynamicznym.

3.3. Zajęcia nr 3

PROGRAM KOMPUTEROWY:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

namespace Symulator
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            while (true)
            {
                Console.WriteLine(i++);
                Thread.Sleep(100);
            }
        }
    }

    class Ship
    {
        public string Name;
        public Ship(string Name)
        {
            this.Name = Name;
        }
    }
    class Transporter : Ship
    {
        private int Capacity;
        public Transporter(string Name, int Capacity)
            : base(Name)
        {
            this.Name = Name;
            this.Capacity = Capacity;
        }
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Rozszerz program o dwie kolejne podklasy.
Wypisz potrzebne klasy i zależności między nimi.
Zaprogramuj powyższe klasy.

ZADANIA (ROK III - poziom rozszerzony):

Wypisz potrzebne klasy, parametry oraz zależności między nimi dla Twojej części symulatora.

3.4. Zajęcia nr 4

PROGRAM KOMPUTEROWY:

```
using System;
using System.ComponentModel;
using System.Threading;

namespace backgroundworker
{
    class generator
    {
        private BackgroundWorker bw;

        public generator()
        {
            bw = new BackgroundWorker();
            bw.DoWork += new DoWorkEventHandler(bw_DoWork);
        }

        public void Start()
        {
            bw.RunWorkerAsync();
        }

        private void bw_DoWork(object sender, DoWorkEventArgs e)
        {
        }
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Rozszerzyć klasę generator o:

a. wypisywanie id wątku co określony czas (rzędu 100 ms),
b. wypisanie informacji o początku wykonywania rozkazów,
c. zatrzymywanie wątku z zewnątrz + informacja.

W programie głównym:

a. stworzyć 6 obiektów generator na liście (List<>) ,
b. po 10 sekundach zatrzymać wątki o parzystych id,
c. po 15 sekundach uruchomić jeden z zatrzymanych wcześniej wątków.

ZADANIA (ROK III - poziom rozszerzony):

Zastąpić listę słownikiem (Dictionary<>).
Napisać funkcję, która sprawdzi wszystkie obiekty czy są używane i w przypadku, odpowiedzi negatywnej usunie je z list lub słownika.
Zastąpić BackgroundWorker np. poprzez Thread.

3.5. Zajęcia nr 5

PROGRAM KOMPUTEROWY:

```
class Program
{
    static void Main(string[] args)
    {
        double[] tab = new double[10];
        try
        {
            for (int i = 0; i < 10; i++)
            {
                Console.WriteLine(1.0 / (3 - i));   //A
                tab[i] = 1;                         //B
                                                    //C
            }
        }
        catch (IndexOutOfRangeException e1)
        {
            Console.WriteLine("wyjatek 1: {0}", e1.Message);
        }
        catch (DivideByZeroException e2)
        {
            Console.WriteLine("wyjatek 2: {0}", e2.Message);
        }
        catch (Exception e3)
        {
            Console.WriteLine("wyjatek 3: {0}", e3.Message);
        }
        finally
        {
            Console.WriteLine("Zakonczenie");
        }
        Console.Read();
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Zmienić w linii A: „1.0” na „1”.
Zmienić w linii B: „i” na „2*i”.
Dopisać w linii C: „throw new Exception("Ups :)");”.
Sprawdzić kiedy jest wywoływany blok finally i czy w ogóle jest potrzebny?

ZADANIA (ROK III - poziom rozszerzony):

Sprawdzić jakie informacje można uzyskać z wyjątków oprócz parametru „Message”.
Napisać osobną klasę, która będzie obsługiwała wyjątek i wymusić na obiekcie tej klasy obsłużenie takiego wyjątku.

3.6. Zajęcia nr 6

PROGRAM KOMPUTEROWY:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net.Sockets;
using System.Net;

namespace udp_komunikator
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            CheckForIllegalCrossThreadCalls = false;
            bw = new BackgroundWorker();
            bw.DoWork += new DoWorkEventHandler(bw_DoWork);
            bw.RunWorkerAsync();
        }
        private BackgroundWorker bw;
        private void button1_Click(object sender, EventArgs e)
        {
            byte[] buffer = System.Text.Encoding.ASCII.GetBytes(String.Format("{0}:{1}", textBox1.Text,
textBox2.Text));
            UdpClient udpClient = new UdpClient();
            udpClient.Send(buffer, buffer.Length, new IPEndPoint(IPAddress.Broadcast, 23456));
            udpClient.Close();
            textBox2.Text = "";
        }
        private void bw_DoWork(object sender, DoWorkEventArgs ea)
        {
            UdpClient udpClient = new UdpClient(23456);
            while (true)
            {
                IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Any, 0);
                byte[] buffer = udpClient.Receive(ref RemoteIpEndPoint);
                if (buffer != null)
                {
                    richTextBox1.Text += String.Format("{0}\n",
System.Text.Encoding.ASCII.GetString(buffer));
                }
            }
        }
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Program przesyła wiadomości pomiędzy urządzeniami w jednej sieci.
Zmodyfikuj program tak, aby przesyłał wiadomości tylko do konkretnych odbiorców.

ZADANIA (ROK III - poziom rozszerzony):

Program przesyła wiadomości pomiędzy urządzeniami w jednej sieci.
Zmodyfikuj program tak, aby przesyłał konkretne parametry do konkretnych odbiorców.

3.7. Zajęcia nr 7

PROGRAM KOMPUTEROWY:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net.Sockets;
using System.Net;

namespace tcp_ip_komunikator
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            CheckForIllegalCrossThreadCalls = false;
            bw = new BackgroundWorker();
            bw.DoWork += new DoWorkEventHandler(bw_DoWork);
            bw.RunWorkerAsync();
        }
        private BackgroundWorker bw;
        private void button1_Click(object sender, EventArgs e)
        {
            byte[] buffer = System.Text.Encoding.ASCII.GetBytes(String.Format("{0}:{1}", textBox1.Text,
textBox2.Text));
            UdpClient udpClient = new UdpClient();
            udpClient.Send(buffer, buffer.Length, new IPEndPoint(IPAddress.Broadcast, 23456));
            udpClient.Close();
            textBox2.Text = "";
        }
        private void bw_DoWork(object sender, DoWorkEventArgs ea)
        {
            UdpClient udpClient = new UdpClient(23456);
            while (true)
            {
                IPEndPoint RemoteIpEndPoint = new IPEndPoint(IPAddress.Any, 0);
                byte[] buffer = udpClient.Receive(ref RemoteIpEndPoint);
                if (buffer != null)
                {
                    richTextBox1.Text += String.Format("{0}\n",
System.Text.Encoding.ASCII.GetString(buffer));
                }
            }
        }
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Zmodyfikuj program tak, aby transmisja odbywała się poprzez TCP/IP.
Zmodyfikuj program tak, aby przesyłał wiadomości tylko do konkretnych odbiorców.

ZADANIA (ROK III - poziom rozszerzony):

Zmodyfikuj program tak, aby transmisja odbywała się poprzez TCP/IP.
Zmodyfikuj program tak, aby przesyłał konkretne parametry do konkretnych odbiorców.

3.8. Zajęcia nr 8

PROGRAM KOMPUTEROWY:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

namespace Symulator
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            double x = 0;
            double y = 0;
            double speed = 10;
            double course = 0;
            double rudder = 30;
            while (true)
            {
                course += rudder / 10000;
                x += 0.001 * speed * Math.Cos(course / 57.3);
                y += 0.001 * speed * Math.Sin(course / 57.3);
                Console.WriteLine("Pozycja: [{0},{1}]",x,y);
                Thread.Sleep(100);
            }
        }
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Zmodyfikuj program tak, aby wyliczał nową pozycję z wybranego modelu hydrodynamicznego.

ZADANIA (ROK III - poziom rozszerzony):

Zmodyfikuj program tak, aby wyliczał nową pozycję z dowolnego modelu hydrodynamicznego.

Zaimplementuj obsługę dwóch przycisków, tak aby móc zmieniać wychylenie płetwy sterowej.

3.9. Zajęcia nr 9

PROGRAM KOMPUTEROWY:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Zdarzenia
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.Text = "Load";
        }

        private void Form1_MouseClick(object sender, MouseEventArgs e)
        {
            this.Text = "MouseClick";
        }

        private void Form1_MouseLeave(object sender, EventArgs e)
        {
            this.Text = "MouseLeave";
        }

        private void Form1_MouseMove(object sender, MouseEventArgs e)
        {
            this.Text = "MouseMove";
        }

        private void Form1_MouseUp(object sender, MouseEventArgs e)
        {
            this.Text = "MouseUp";
        }

        private void Form1_MouseDown(object sender, MouseEventArgs e)
        {
            this.Text = "MouseDown";
        }
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Przeanalizować program i rozszerzyć o więcej zdarzeń.

ZADANIA (ROK III - poziom rozszerzony):

Przeanalizować program i rozszerzyć o zdarzenia – dla elementu button.

3.10. Zajęcia nr 10

PROGRAM KOMPUTEROWY:

```
delegate void SetTextViaInvoke(string t);

public void SetText(string t)
{
    if (this.richTextBox1.InvokeRequired)
    {

        this.Invoke(new SetTextViaInvoke(SetText), new object[] { t });
    }
    else
    {
        this.richTextBox1.Text = t;
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Powyższy wycinek programu przedstawia metodę oraz delegatę, dzięki której można zmienić parametr Text dla obiektu utworzonego przez inny wątek.
Zaproponuj wykorzystanie powyższego programu.

ZADANIA (ROK III - poziom rozszerzony):

Zastosuj powyższy wycinek programu do zmiany parametru i rozszerz go o drugi parametr.

3.11. Zajęcia nr 11

PROGRAM KOMPUTEROWY:

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

namespace generator_nmea_001
{
    public partial class Form1 : Form
    {
        string par1 = "test";
        string par2 = "test";
        string par3 = "test";
        string par4 = "test";
        public Form1()
        {
            InitializeComponent();
        }
        delegate void SetTextViaInvoke(string t);

        public void SetText(string t)
        {
            if (this.richTextBox1.InvokeRequired)
            {

                this.Invoke(new SetTextViaInvoke(SetText), new object[] { t });
            }
            else
            {
                this.richTextBox1.Text += t;
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            SetText(String.Format("{0},{1},{2},{3}\n", par1, par2, par3, par4));
        }
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Utwórz dowolnie wybraną ramkę NMEA 0183 i uzupełnij losowymi parametrami.

ZADANIA (ROK III - poziom rozszerzony):

Utwórz potrzebne ramki NMEA 0183 i uzupełnij wszystkimi parametrami.

3.12. Zajęcia nr 12

PROGRAM KOMPUTEROWY:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace parsowanie
{
    class Program
    {
        static void Main(string[] args)
        {
            string w = "$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47";
            string suma = "";
            string x = NMEA.Ramka(w, ref suma);
            Console.WriteLine(suma);
            Console.WriteLine(NMEA.SumaKontrolna(x));
            Console.WriteLine(NMEA.SumaKontrolna(suma,NMEA.SumaKontrolna(x)));
            NMEA.Rozdziel(x);
            Console.ReadLine();
        }
    }
    class NMEA
    {
        static public string Ramka(string x,ref string sk)
        {
            string[] t1=x.Split('$');
            if(t1.Length==2)
            {
                string[] t2=t1[1].Split('*');
                if (t1.Length == 2)
                {
                    if (t2[1].Length == 2)
                    {
                        sk = t2[1];
                        return t2[0];
                    }
                }
            }
            return "";
        }
        static public string SumaKontrolna(string x)
        {
            return null;

        }
        static public void Rozdziel(string x)
        {
            string[] t = x.Split(',');
            Console.WriteLine("Wiadomość zawiera {0} pól:",t.Length);
            for (int i = 0; i < t.Length;i++ ) Console.WriteLine("Pole {0} = {1}",i,t[i]);
        }

    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Rozdziel informację z ramki GGA.

ZADANIA (ROK III - poziom rozszerzony):

Dodaj pięć dowolnych ramek NMEA 0183 i je rozparsuj.

3.13. Zajęcia nr 13

PROGRAM KOMPUTEROWY:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

namespace Symulator
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            double x = 0;
            double y = 0;
            double speed = 10;
            double course = 0;
            double rudder = 30;
            while (true)
            {
                course += rudder / 10000;
                x += 0.001 * speed * Math.Cos(course / 57.3);
                y += 0.001 * speed * Math.Sin(course / 57.3);
                Console.WriteLine("Pozycja: [{0},{1}]",x,y);
                Thread.Sleep(100);
            }
        }
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Korzystając z dowolnej biblioteki graficznej zaznacz pozycję jednostki co 0.5 sekundy.

ZADANIA (ROK III - poziom rozszerzony):

Korzystając z dowolnej biblioteki graficznej zaznacz pozycję i wodnicę jednostki co 0.5 sekundy.

3.14. Zajęcia nr 14

PROGRAM KOMPUTEROWY:

```
#define debug
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Sockets;
using System.Net;
using System.ComponentModel;

namespace Modbus
{
    public class Modbus
    {
        private IPAddress ip;
        private TcpListener server = null;
        private Byte[] bytes = new Byte[256];//bufor odbiorczy
        private Byte[] bytes2 = new Byte[256];//bufor nadawczy
        private Rejestry[] rej;
        private Bity[] bit;
        private BackgroundWorker bw;
        public Modbus(string pc_ip,ref Bity[] bitowe,ref Rejestry[] rejestry)
        {
            this.ip = IPAddress.Parse(pc_ip);
            bit = bitowe;       //Coils
            rej = rejestry;     //Registers
        }

        public void Odbieranie()
        {
            bw = new BackgroundWorker();
            bw.DoWork += new DoWorkEventHandler(bw_DoWork);
            bw.RunWorkerAsync();
        }

        private void bw_DoWork(object sender, DoWorkEventArgs ea)
        {
            /*
             * ramka modbus:
             * 0-1 : identyfikator transakcji
             * to pole identyfikatora jest używane do
             * sparowania transakcji gdy zostało wysłanych kolejno wiele wiadomości w jednym
             * połączeniu TCP przez klienta bez czekania na uprzednią odpowiedź.
             *
             * 2-3 : identyfikator protokołu
             * to pole jest zawsze równe 0 dla komunikatów
             * Modbus i innych wartości są zarezerwowane dla przyszłego rozszerzenia
             *
             * 4-5 : pole długości
             * to pole zawiera liczebność kolejnych pól włącznie z identyfikatorem jednostki, kodem
funkcji i polem danych.
             *
             * 6   : identyfikator jednostki
             * to pole używa się w celu identyfikacji zdalnego
             * serwera sieci nie TCP/IP (dal szeregowego mostkowania). W typowym serwerze
             * aplikacji Modbus TCP/IP, identyfikator jednostki jest ustawiony na 00 lub FF,
             * ignorowane przez serwer i po prostu odpowiadane z powrotem w odpowiedzi.
             *
             * 7   : kod funkcji
             *
             * 8-? : dane
             */

            server = new TcpListener(ip, 502);
            server.Start();
            while (true)
            {
                Console.WriteLine("MODBUS: Oczekiwanie na polaczenie");
                TcpClient client = server.AcceptTcpClient();
                Console.WriteLine("MODBUS: Polaczono z " + client.Client.RemoteEndPoint);
                NetworkStream stream = client.GetStream();
                int str;
```

```
                int rozmiar;
                byte bajts;
                while ((str = stream.Read(bytes, 0, bytes.Length)) != 0)
                {
                    try
                    {
                        string data = System.Text.Encoding.ASCII.GetString(bytes, 0, str);
                        rozmiar = data.Length;

                        for (int j = 0; j < 7; j++) bytes2[j] = bytes[j];
                        switch (bytes[7])
                        {
                            case 1:         //read coils?????

                                bajts = (byte)(((int)bytes[11] - 1) / 8 + 1);
                                bytes2[7] = 1;                                  //nr rozkazu
                                bytes2[8] = bajts;                              //ilość bajtów
                                for (int i = 0; i < (int)bajts; i++)
                                    bytes2[9 + i] = bit[i].read();              //bity
                                rozmiar = 9 + bajts;
                                break;
                            case 5:         //write Coil????
#if debug
                                if (bytes[10] == 0) Console.WriteLine("MODBUS: przycisk {0}:'OFF'",
bytes[9]);
                                else Console.WriteLine("MODBUS: przycisk {0}:'ON'", bytes[9]);
#endif
                                bytes2[7] = 5;                                  //nr rozkazu
                                bytes2[8] = bytes[8];                           //address H
                                bytes2[9] = bytes[9];                           //adres L
                                bytes2[10] = bytes[10];                         //wartość - hh-on,00-
off
                                bytes2[11] = bytes[11];                         //wartosc - musi byc 0
                                rozmiar = 12;
                                try
                                {
                                    if (bytes[10] == 255) bit[(bytes[9] - 1) / 8].write((bytes[9] - 1)
% 8, true);//zapis do odpowiedniego miejsca
                                    else bit[(bytes[9] - 1) / 8].write((bytes[9] - 1) % 8, false);
                                }
                                catch (Exception e)
                                {
                                    Console.WriteLine("MODBUS: Próba zapisu poza zakres\n{0}", e);
                                }
                                break;
                            case 3:         //read registers

                                bajts = (byte)((int)bytes[11] * 2);
                                bytes2[7] = 3;                                  //nr rozkazu
                                bytes2[8] = bajts;                              //ilosc bajtow
                                for (int i = 0; i < (int)bytes[11]; i++)
                                {
                                    bytes2[9 + 2 * i] = rej[i].readhigh();
                                    bytes2[10 + 2 * i] = rej[i].readlow();
                                }
                                rozmiar = 9 + bajts;

                                break;
                            case 6:         //write register
#if debug
                                //if (bytes[11] > 127) Console.WriteLine("MODBUS: suwak {0}:'{1}'",
bytes[9], ((int)(bytes[11])) );//zmieniam256 na 128
                                //else
                                Console.WriteLine("MODBUS: suwak {0}:'{1}'", bytes[9], bytes[11]);
#endif
                                bytes2[7] = 6;              //nr rozkazu
                                bytes2[8] = bytes[8];       //adres H
                                bytes2[9] = bytes[9];       //adres L
                                bytes2[10] = bytes[10];     //wartosc H
                                bytes2[11] = bytes[11];     //wartosc L
                                rozmiar = 12;
                                for (int i = 0; i < rej.Length; i++)
                                    if ((rej[i].readadres() == bytes[9]) && rej[i].mastercanchange())
rej[i].write(bytes[10], bytes[11]);
```

```
                                //zapis do odpowiedniego miejsca jeśli zapisywalne
                                break;
                            default:

                                Console.WriteLine("MODBUS: nieznane polecenie:");
                                for (int j = 0; j < rozmiar; j++) Console.WriteLine("MODBUS: bajt
{0}:'{1}'", j, bytes[j]);//Obsługa nieznanego polecenia
                                break;
                        }

                        bytes2[4] = (byte)((rozmiar - 6) / 256);    //poprawka długości ramki TCP
                        bytes2[5] = (byte)((rozmiar - 6) % 256);    //poprawka długości ramki TCP
                        stream.Write(bytes2, 0, rozmiar);           //Wysłanie odpowiedzi
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine("MODBUS: nienznany blad: {0}",ex.Message);
                    }
                }
                client.Close();
            }
        }
    }

    public class Rejestry
    {
        private volatile uint wart;
        private volatile byte high;
        private volatile byte low;
        private volatile byte adres;
        private volatile bool can;
        public Rejestry(byte adres)
        {
            this.adres = adres;
            wart = 0;
            high = 0;
            low = 0;
            can = false;
        }
        public byte readhigh()
        {
            return high;
        }
        public byte readadres()
        {
            return adres;
        }
        public byte readlow()
        {
            return low;
        }
        public uint read()
        {
            return wart;
        }
        public bool mastercanchange()
        {
            return can;
        }
        public void write(byte x, byte y)
        {
            high = x;
            low = y;
            wart = (uint)x * 256 + (uint)y;
        }
        public void write(uint x)
        {
            wart = x;
            high = (byte)(x / 256);
            low = (byte)(x % 256);
        }
        public void mastercanchange(bool can)
        {
            this.can=can;
```

```
        }
    }

    public class Bity
    {
        private volatile byte bajt;
        private volatile bool[] bit = new bool[8];
        public Bity()
        {
            bajt = 0;
            for (int i = 0; i < 8; i++) bit[i] = false;
        }
        public void write(int adres, bool wart)
        {
            bit[adres] = wart;
            for (int i = 0; i < 8; i++)
                if (adres == i)
                {
                    if (wart) bajt |= (byte)(1 << i);
                    else bajt &= (byte)(~(1 << i));
                }
        }
        public void write(byte wart)
        {
            bajt = wart;
            for (int i = 0; i < 8; i++)
            {
                bit[i] = false;
                if ((wart & (128 >> i)) > 0) bit[i] = true;
            }
        }
        public bool read(int adres)
        {
            return bit[adres];
        }
        public byte read()
        {
            return bajt;
        }
    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Zainicjować połączenie za pomocą protokołu MODBUS z panelem dotykowym.

ZADANIA (ROK III - poziom rozszerzony):

Napisz bibliotekę do komunikacji za pomocą dowolnego protokołu przemysłowego (taki jak: Modbus, Profibus, Profinet) i nawiąż połączenie z dowolnym elementem symulatora.

3.15. Zajęcia nr 15

PROGRAM KOMPUTEROWY:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace MBControls
{
    public partial class MBRotate : UserControl
    {
        private int value = 0;
        public MBRotate()
        {
            InitializeComponent();

            label1.Text = "000°";
            label2.Text = "060°";
            label3.Text = "120°";
            label4.Text = "180°";
            label5.Text = "240°";
            label6.Text = "300°";
        }

        [Browsable(true), Category("Dane")]
        public int Value
        {
            get { return value; }
            set
            {
                this.value = value;
                while (this.value < 0) this.value += 360;
                this.value %= 360;
                Refreshes();
            }
        }

        private void MBRotate_Paint(object sender, PaintEventArgs e)
        {
            Refreshes();
        }
        private void Refreshes()
        {
            Pen pen = new System.Drawing.Pen(Color.FromArgb(0, 0, 115), 1);
            Pen pen2 = new System.Drawing.Pen(Color.FromArgb(0, 0, 115), 3);
            SolidBrush brush = new SolidBrush(Color.FromArgb(0, 0, 115));
            int min = (int)(Math.Min(this.Width, this.Height) / 1.28);
            Point r = new Point((this.Width - min) / 2 + 1, (this.Height - min) / 2 + 1);
            Point p = new Point(r.X + min / 2, r.Y + min / 2);
            Graphics g1 = this.CreateGraphics();
            g1.Clear(this.BackColor);
            Point[] pkt = { new Point(r.X + (int)(min * (0.5 + 0.5 * Math.Sin(Math.PI * value / 180))),
r.Y + (int)(min * (0.5 - 0.5 * Math.Cos(Math.PI * value / 180)))), new Point(r.X + (int)(min * (0.5 +
0.05 * Math.Sin(Math.PI * (value - 90) / 180))), r.Y + (int)(min * (0.5 - 0.05 * Math.Cos(Math.PI *
(value - 90) / 180)))), new Point(r.X + (int)(min * (0.5 + 0.05 * Math.Sin(Math.PI * (value + 90) /
180))), r.Y + (int)(min * (0.5 - 0.05 * Math.Cos(Math.PI * (value + 90) / 180)))) };
            g1.FillPolygon(brush, pkt);
            g1.DrawEllipse(pen2, new Rectangle(r.X, r.Y, min - 2, min - 2));
            min = min / 2 - 1;
            for (int i = 0; i < 36; i++)
                if (i % 6 == 0) g1.DrawLine(pen2, p.X + (int)(min * Math.Sin(Math.PI * i / 18)), p.Y +
(int)(min * Math.Cos(Math.PI * i / 18)), p.X + (int)(min * 0.8 * Math.Sin(Math.PI * i / 18)), p.Y +
(int)(min * 0.8 * Math.Cos(Math.PI * i / 18)));
                else g1.DrawLine(pen, p.X + (int)(min * Math.Sin(Math.PI * i / 18)), p.Y + (int)(min *
Math.Cos(Math.PI * i / 18)), p.X + (int)(min * 0.9 * Math.Sin(Math.PI * i / 18)), p.Y + (int)(min * 0.9
* Math.Cos(Math.PI * i / 18)));
            min = (int)(min * 1.35);
            label1.Font = label2.Font = label3.Font = label4.Font = label5.Font = label6.Font = new
Font(FontFamily.GenericSansSerif, min / 12);
```

```
            label1.Location = new Point(p.X - label1.Width / 2, p.Y - label1.Height / 2 - (int)(min *
0.89));
            label2.Location = new Point(p.X - label2.Width / 2 + (int)(min * Math.Sin(Math.PI * 1 /
3)), p.Y - label2.Height / 2 - (int)(min * Math.Cos(Math.PI * 1 / 3)));
            label3.Location = new Point(p.X - label3.Width / 2 + (int)(min * Math.Sin(Math.PI * 2 /
3)), p.Y - label3.Height / 2 - (int)(min * Math.Cos(Math.PI * 2 / 3)));
            label4.Location = new Point(p.X - label4.Width / 2, p.Y - label4.Height / 2 + (int)(min *
0.89));
            label5.Location = new Point(p.X - label5.Width / 2 + (int)(min * Math.Sin(Math.PI * 4 /
3)), p.Y - label5.Height / 2 - (int)(min * Math.Cos(Math.PI * 4 / 3)));
            label6.Location = new Point(p.X - label6.Width / 2 + (int)(min * Math.Sin(Math.PI * 5 /
3)), p.Y - label6.Height / 2 - (int)(min * Math.Cos(Math.PI * 5 / 3)));

        }

    }
}
```

ZADANIA (ROK II - poziom podstawowy):

Stwórz własny element User Control: button.

ZADANIA (ROK III - poziom rozszerzony):

Stwórz własny element User Control: scroll.

4. WARUNKI ZALICZENIA PRZEDMIOTU

Zaliczenia praktyczne

5. EFEKTY KSZTAŁCENIA

Efekty kształcenia semestr IV		Kierunkowe
EK1	Definiuje algorytm działania symulatora w tym: modelu hydrodynamicznego, układu sterownia, Algorytmizuje metody wizualizacji, instrumentów nawigacyjnych, układów siłowni, holowników i innych elementów symulatora.	K_W01; K_W14; K_U15
EK2	Opisuje i metody generowania scenerii 2D i 3D, Funkcjonalnie przedstawia metody używania map elektronicznych.	K_W15; K_U23
EK3	Charakteryzuje zasady funkcjonowania warstw sprzętowych komunikacji sprzętu, Potrafi używać klas obsługi sprzętu w symulacji, Umie zaprogramować własne klasy związane z komunikacja i metodami numerycznymi w programowaniu symulatorów.	K_W04; K_U09

6. INFORMACJE DODATKOWE

7. LITERATURA

1. Babiuch M., *Autocad 2007*, Helion 2007.
2. Baron B., *Algorytmy numeryczne w Delphi*, Helion, 2007.
3. David M., *Fizyka dla programistów gier*, Helion 2003.
4. Gucma L., *Modelowanie czynników ryzyka zderzenia jednostek pływających z konstrukcjami portowymi i pełnomorski-mi*, AM Szczecin 2005.
5. Pang T., *Metody obliczeniowe w fizyce*, PWN 2001.
6. Perry S., *C# I .NET*, Helion 2007.
7. Rosłoniec S., *Wybrane metody numeryczne*, Oficyna PW 2002.
8. Sharp J., *Visual c#2005*, Microsoft 2006.
9. Podręcznik Multigen (multimedia)
10. Gucma S., *Inżynieria Ruchu Morskiego*, Okrętownictwo i Żegluga 2001.

8. FORMULARZE, ZAŁĄCZNIKI