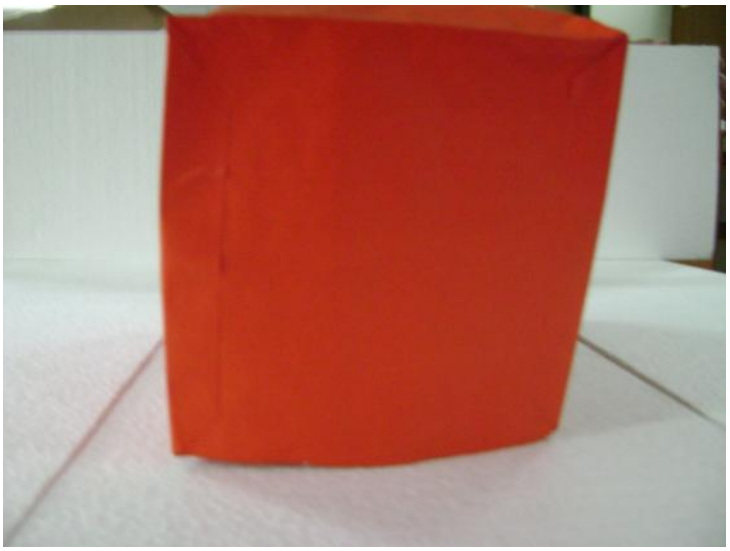
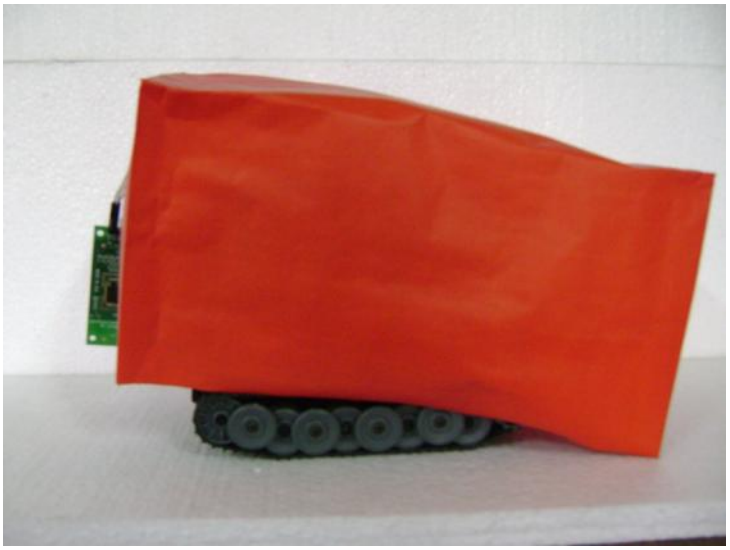
LAMPIRAN A
FOTO ROBOT MOBIL TANK

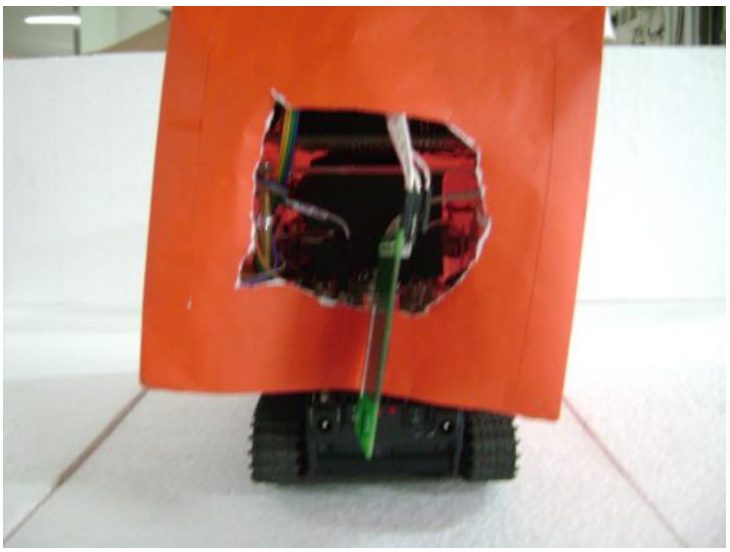
Gambar A.1. Foto Robot Mobil Dengan PenutupDilihat Dari Atas



Gambar A.2 Foto Robot Mobil Dengan Penutup Dilihat Dari depan



Gambar A.3 Foto Robot Mobil Dengan PenutupDilihat Dari samping



Gambar A.4 Foto Robot Mobil Dengan PenutupDilihat Dari Belakang

LAMPIRAN B

PROGRAM PADA PENGONTROL MIKRO

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.3 Professional
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 7/23/2009
Author  : Lab Instrumentasi
Company : UKM
Comments:


Chip type           : ATmega16
Program type        : Application
Clock frequency     : 11.059200 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

#include <mega16.h>

// Standard Input/Output functions
#include <stdio.h>
#include <delay.h>
// Declare your global variables here
void maju()
{      PORTD.2=0;
       PORTD.3=1;
       PORTD.4=1;
       PORTD.5=1;
       PORTD.6=0;
       PORTD.7=1;

}

void kiri()
{      PORTD.2=0;
       PORTD.3=1;
       PORTD.4=1;
       PORTD.5=1;
       PORTD.6=1;
       PORTD.7=0;
       delay_ms(450);
       PORTD.4=0;
       PORTD.5=0;
}
void kanan()
{      PORTD.2=1;
       PORTD.3=0;
       PORTD.4=1;
       PORTD.5=1;
       PORTD.6=0;
       PORTD.7=1;
       delay_ms(450);
```

```
      PORTD.4=0;
      PORTD.5=0;
}
void stop()
{
PORTD.4=0;
PORTD.5=0;
}
void main(void)
{char a;
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x3F;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x3F;

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=In Func0=In
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=T State0=T
PORTD=0x00;
DDRD=0xFC;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
```

```
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x47;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

while (1)
     {
     // Place your code here
       a=getchar();
      if (a=='w')
      {maju();}
      if (a=='q')
      {kiri();}
      if (a=='e')
```

```
      {kanan();}
      if(a=='s')
      {stop();}
     };
}
```

LAMPIRAN C
PROGRAM PADA VISUAL C#

```
using System.IO;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Drawing.Imaging;
using Mehroz;
using System.IO.Ports;
using System.Threading;


namespace MazeSolverDemo
{
	/// <summary>
	/// Summary description for MazeSolver
	/// Version 1.2
	/// Developed by: Syed Mehroz Alam
	/// Email: smehrozalam@yahoo.com
	/// URL: Programming Home "http://www.geocities.com/smehrozalam/"
	///
	///
	/// New in verson 1.1
	///		* Extended the program so that it can work on rectangular maze
(pervious version
	///			worked for only square maze)
	///
	/// New in verson 1.2
	///		* The class can now handle diagonal paths also
	/// </summary>
	public class frmMain : System.Windows.Forms.Form
	{	Bitmap _image;
		Mehroz.MazeSolver m_Maze;
    MazeSolver.StringA stringA;
    MazeSolver.StringT stringT;
		int[,] m_iMaze;
		int m_iSize=20;
		int m_iRowDimensions=7;
		int m_iColDimensions=7;
    string RxString;


		private System.Windows.Forms.PictureBox pictureBox1;
		private System.Windows.Forms.Button cmdReset;
		private System.Windows.Forms.Button cmdExit;
		private System.Windows.Forms.Button button2;
		private System.Windows.Forms.OpenFileDialog OP;
		private System.Windows.Forms.Button button3;
    private System.Windows.Forms.Button button4;
    private SerialPort serialPort1;
    private TextBox Xbox;
    private Label label1;
```

```
private Label label2;
private TextBox Ybox;
private Button Start;
private TextBox textBox1;
private Button Stop;
private TextBox xx;
private TextBox yy;
private Button run;
private TextBox posisi;
private TextBox aa;
private IContainer components;

        public frmMain()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();

            //
            // TODO: Add any constructor code after InitializeComponent call
            //
        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if (components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }

        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
    this.components = new System.ComponentModel.Container();
    this.pictureBox1 = new System.Windows.Forms.PictureBox();
    this.cmdReset = new System.Windows.Forms.Button();
    this.cmdExit = new System.Windows.Forms.Button();
    this.button2 = new System.Windows.Forms.Button();
    this.OP = new System.Windows.Forms.OpenFileDialog();
    this.button3 = new System.Windows.Forms.Button();
    this.button4 = new System.Windows.Forms.Button();
    this.serialPort1 = new System.IO.Ports.SerialPort(this.components);
```

```
            this.Xbox = new System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.Ybox = new System.Windows.Forms.TextBox();
            this.Start = new System.Windows.Forms.Button();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.Stop = new System.Windows.Forms.Button();
            this.xx = new System.Windows.Forms.TextBox();
            this.yy = new System.Windows.Forms.TextBox();
            this.run = new System.Windows.Forms.Button();
            this.posisi = new System.Windows.Forms.TextBox();
            this.aa = new System.Windows.Forms.TextBox();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
            this.SuspendLayout();
            //
            // pictureBox1
            //
            this.pictureBox1.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.pictureBox1.Location = new System.Drawing.Point(24, 16);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(184, 168);
            this.pictureBox1.TabIndex = 0;
            this.pictureBox1.TabStop = false;
            this.pictureBox1.Paint += new
System.Windows.Forms.PaintEventHandler(this.pictureBox1_Paint);
            //
            // cmdReset
            //
            this.cmdReset.Location = new System.Drawing.Point(328, 136);
            this.cmdReset.Name = "cmdReset";
            this.cmdReset.Size = new System.Drawing.Size(104, 24);
            this.cmdReset.TabIndex = 6;
            this.cmdReset.Text = "&Reset Maze";
            this.cmdReset.Click += new System.EventHandler(this.cmdReset_Click);
            //
            // cmdExit
            //
            this.cmdExit.Location = new System.Drawing.Point(328, 176);
            this.cmdExit.Name = "cmdExit";
            this.cmdExit.Size = new System.Drawing.Size(104, 24);
            this.cmdExit.TabIndex = 8;
            this.cmdExit.Text = "E&xit";
            this.cmdExit.Click += new System.EventHandler(this.cmdExit_Click);
            //
            // button2
            //
            this.button2.Location = new System.Drawing.Point(328, 16);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(104, 24);
            this.button2.TabIndex = 11;
            this.button2.Text = "Open";
            this.button2.Click += new System.EventHandler(this.button2_Click);
            //
            // button3
            //
```

```
            this.button3.Location = new System.Drawing.Point(328, 96);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(104, 23);
            this.button3.TabIndex = 12;
            this.button3.Text = "Solve";
            this.button3.Click += new System.EventHandler(this.button3_Click);
            //
            // button4
            //
            this.button4.Location = new System.Drawing.Point(328, 56);
            this.button4.Name = "button4";
            this.button4.Size = new System.Drawing.Size(104, 23);
            this.button4.TabIndex = 15;
            this.button4.Text = "Wall";
            this.button4.Click += new System.EventHandler(this.button4_Click);
            //
            // serialPort1
            //
            this.serialPort1.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(this.serialPort1_DataReceived);
            //
            // Xbox
            //
            this.Xbox.Location = new System.Drawing.Point(232, 255);
            this.Xbox.Name = "Xbox";
            this.Xbox.Size = new System.Drawing.Size(40, 20);
            this.Xbox.TabIndex = 18;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(244, 239);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(14, 13);
            this.label1.TabIndex = 19;
            this.label1.Text = "X";
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(325, 239);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(14, 13);
            this.label2.TabIndex = 20;
            this.label2.Text = "Y";
            //
            // Ybox
            //
            this.Ybox.Location = new System.Drawing.Point(310, 255);
            this.Ybox.Name = "Ybox";
            this.Ybox.Size = new System.Drawing.Size(41, 20);
            this.Ybox.TabIndex = 21;
            //
            // Start
            //
```

```
            this.Start.Location = new System.Drawing.Point(232, 294);
            this.Start.Name = "Start";
            this.Start.Size = new System.Drawing.Size(75, 23);
            this.Start.TabIndex = 22;
            this.Start.Text = "Start";
            this.Start.UseVisualStyleBackColor = true;
            this.Start.Click += new System.EventHandler(this.button1_Click);
            //
            // textBox1
            //
            this.textBox1.Location = new System.Drawing.Point(232, 216);
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(100, 20);
            this.textBox1.TabIndex = 23;
            this.textBox1.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.textBox1_KeyPress);
            //
            // Stop
            //
            this.Stop.Location = new System.Drawing.Point(348, 294);
            this.Stop.Name = "Stop";
            this.Stop.Size = new System.Drawing.Size(75, 23);
            this.Stop.TabIndex = 24;
            this.Stop.Text = "Stop";
            this.Stop.UseVisualStyleBackColor = true;
            this.Stop.Click += new System.EventHandler(this.Stop_Click);
            //
            // xx
            //
            this.xx.Location = new System.Drawing.Point(236, 334);
            this.xx.Name = "xx";
            this.xx.Size = new System.Drawing.Size(55, 20);
            this.xx.TabIndex = 25;
            //
            // yy
            //
            this.yy.Location = new System.Drawing.Point(310, 334);
            this.yy.Name = "yy";
            this.yy.Size = new System.Drawing.Size(51, 20);
            this.yy.TabIndex = 26;
            //
            // run
            //
            this.run.Location = new System.Drawing.Point(74, 255);
            this.run.Name = "run";
            this.run.Size = new System.Drawing.Size(75, 23);
            this.run.TabIndex = 27;
            this.run.Text = "run";
            this.run.UseVisualStyleBackColor = true;
            this.run.Click += new System.EventHandler(this.run_Click);
            //
            // posisi
            //
            this.posisi.Location = new System.Drawing.Point(63, 216);
            this.posisi.Name = "posisi";
```

```
        this.posisi.Size = new System.Drawing.Size(100, 20);
        this.posisi.TabIndex = 28;
        //
        // aa
        //
        this.aa.Location = new System.Drawing.Point(63, 297);
        this.aa.Name = "aa";
        this.aa.Size = new System.Drawing.Size(100, 20);
        this.aa.TabIndex = 30;
        //
        // frmMain
        //
        this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
        this.ClientSize = new System.Drawing.Size(472, 382);
        this.Controls.Add(this.aa);
        this.Controls.Add(this.posisi);
        this.Controls.Add(this.run);
        this.Controls.Add(this.yy);
        this.Controls.Add(this.xx);
        this.Controls.Add(this.Stop);
        this.Controls.Add(this.textBox1);
        this.Controls.Add(this.Start);
        this.Controls.Add(this.Ybox);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.Xbox);
        this.Controls.Add(this.button4);
        this.Controls.Add(this.button3);
        this.Controls.Add(this.button2);
        this.Controls.Add(this.cmdExit);
        this.Controls.Add(this.cmdReset);
        this.Controls.Add(this.pictureBox1);
        this.Name = "frmMain";
        this.Text = "Maze Solver by Syed Mehroz Alam";
        this.Load += new System.EventHandler(this.Form1_Load);
        this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.frmMain_FormClosing);
        ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
        this.ResumeLayout(false);
        this.PerformLayout();

                }
                #endregion

                /// <summary>
                /// The main entry point for the application.
                /// </summary>
                [STAThread]
                static void Main()
                {
                        try
                        {
                                Application.Run(new frmMain());
                        }
                        catch(Exception exp)
```

```
                {
                        System.Windows.Forms.MessageBox.Show("An unhandled
exception "'+exp.Message+"' has occurred. \nPlease tell me at smehrozalam@yahoo.com so that I
can fix this bug. Thank you", "Error");
                }
        }

        private void Form1_Load(object sender, System.EventArgs e)
        {
                m_Maze = new Mehroz.MazeSolver(m_iRowDimensions,
m_iColDimensions);
                this.pictureBox1.Size=new
System.Drawing.Size(m_iColDimensions*m_iSize+3, m_iRowDimensions*m_iSize+3);
                m_iMaze=m_Maze.GetMaze;
        }

        private void pictureBox1_Paint(object sender,
System.Windows.Forms.PaintEventArgs e)
        {
                Graphics myGraphics = e.Graphics;
                for (int i=0;i<m_iRowDimensions;i++)
                        for (int j=0;j<m_iColDimensions;j++)
                        {
                                // print grids
            myGraphics.DrawRectangle(new Pen(Color.Black) , j*m_iSize, i*m_iSize,
m_iSize, m_iSize);
                                // print walls
                                if ( m_iMaze[i,j]==1 )
                                        myGraphics.FillRectangle(new
SolidBrush(Color.DarkGray) , j*m_iSize+1, i*m_iSize+1, m_iSize-1, m_iSize-1);
                                //print path
                                if ( m_iMaze[i,j]==100 )
                                        myGraphics.FillRectangle(new
SolidBrush(Color.Cyan) , j*m_iSize+1, i*m_iSize+1, m_iSize-1, m_iSize-1);
            if (m_iMaze[i, j] == 10)
               myGraphics.FillRectangle(new SolidBrush(Color.Red), j * m_iSize + 1, i *
m_iSize + 1, m_iSize - 1, m_iSize - 1);
          }
        }



        private void cmdReset_Click(object sender, System.EventArgs e)
        {
                m_Maze = new Mehroz.MazeSolver(m_iRowDimensions,
m_iColDimensions);
                m_iMaze=m_Maze.GetMaze;
                this.Refresh();
        }

        private void cmdExit_Click(object sender, System.EventArgs e)
        {
                this.Close();
        }
```

```
		private void button2_Click(object sender, System.EventArgs e)
		{

			OP.InitialDirectory="C:";
			OP.Title="insert";
			OP.FileName="";
			OP.ShowDialog();
			//pictureBox2.Visible=true;
			//increase contrast
			_image=(Bitmap)Image.FromFile(OP.FileName,true);
			Image oldImage=this.pictureBox1.Image;
			Bitmap bm=new Bitmap(this._image.Width, this._image.Height);
			Graphics g=Graphics.FromImage(bm);
			ImageAttributes ia=new ImageAttributes();
			ColorMatrix cm=new ColorMatrix(new float[][]{  new
float[]{2f,0f,0f,0f,0f},

				new float[]{0f,2f,0f,0f,0f},

				new float[]{0f,0f,2f,0f,0f},

				new float[]{0f,0f,0f,2f,0f},

				new float[]{0.001f,0.001f,0.001f,0f,1f}});
			ia.SetColorMatrix(cm);
			g.DrawImage(_image,new
Rectangle(0,0,_image.Width,_image.Height),0,0,_image.Width,_image.Height,GraphicsUnit.Pixel
,ia);
			g.Dispose();
			ia.Dispose();
			this.pictureBox1.Image=bm;
			if(oldImage!=null)
				oldImage.Dispose();
		}

		private void button3_Click(object sender, System.EventArgs e)
		{

				int[,] iSolvedMaze=m_Maze.FindPath(6,1, 1,6);
				if (iSolvedMaze!=null)
				{
					m_iMaze=iSolvedMaze;
				}

				this.Refresh();
		}

		private void pictureBox2_Click(object sender, System.EventArgs e)
		{
		}

		private void button4_Click(object sender, System.EventArgs e)
		{
			m_iMaze=m_Maze.GetMaze;
			for(int i=0;i<m_iRowDimensions;i++)
```

```
                {
                    for (int j=0;j<m_iColDimensions;j++)
                    {
                        int wall=0;
                        for(int a=0;a<m_iSize;a++)
                        {
                            for(int b=0;b<m_iSize;b++)
                            {
                                string c = ((pictureBox1.Image as
Bitmap).GetPixel(a+m_iSize*j,b+m_iSize*i) ).ToString();
                                if (c=="Color [A=255, R=255,
G=255, B=255]")
                                {wall ++;}
                            }
                        }
                        if (wall>=15)
                        {m_iMaze[i,j]=1;}
                    }
                }
                this.Refresh();

            }

    public void maju()
    {
       SerialPort port = new SerialPort("Com14", 9600, Parity.None, 8, StopBits.One);
       port.Open();
       port.Write("w");
       port.Close();
    }
    public void kanan()
    {
       SerialPort port = new SerialPort("Com14", 9600, Parity.None, 8, StopBits.One);
       port.Open();
       port.Write("e");
       port.Close();
    }
    public void kiri()
    {
       SerialPort port = new SerialPort("Com14", 9600, Parity.None, 8, StopBits.One);
       port.Open();
       port.Write("q");
       port.Close();
    }
    public void stop()
    {
       SerialPort port = new SerialPort("Com14", 9600, Parity.None, 8, StopBits.One);
       port.Open();
       port.Write("s");
       port.Close();
    }


    private void button1_Click(object sender, EventArgs e)
    {
```

```
    serialPort1.PortName = "COM1";
    serialPort1.BaudRate = 115200;

    serialPort1.Open();
    if (serialPort1.IsOpen)
    {
        Start.Enabled = false;
        Stop.Enabled = true;
        textBox1.ReadOnly = false;
    }

}

private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    RxString = serialPort1.ReadExisting();
    this.Invoke(new EventHandler(DisplayText));
}

private void frmMain_FormClosing(object sender, FormClosingEventArgs e)
{
   if (serialPort1.IsOpen) serialPort1.Close();

}

private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    // If the port is closed, don't try to send a character.

    if (!serialPort1.IsOpen) return;

    // If the port is Open, declare a char[] array with one element.
    char[] buff = new char[1];

    // Load element 0 with the key character.

    buff[0] = e.KeyChar;

    // Send the one character buffer.
    serialPort1.Write(buff, 0, 1);

    // Set the KeyPress event as handled so the character won't
    // display locally. If you want it to display, omit the next line.
    e.Handled = true;

}

private void DisplayText(object sender, EventArgs e)
{
    string l = char.ConvertFromUtf32(13);
    textBox1.Text = "";
    textBox1.AppendText(RxString);
    stringT = new MazeSolver.StringT(RxString);
    stringA = new MazeSolver.StringA(RxString);
```

```
        Ybox.Text = stringA.nextElement();
        Xbox.Text = stringT.nextElement();
        string c = Xbox.Text;
        string d = Convert.ToString(Ybox.Text);
        string i, j;
        if (c == "8" || c == "9" || c == "7" || c == "10" || c == "6" || c == "11" || c == "5")
        { i = "0"; }
        else if (c == "18" || c == "19" || c == "17" || c == "20" || c == "16" || c == "21" || c == "15")
        { i = "1"; }
        else if (c == "28" || c == "29" || c == "27" || c == "30" || c == "26" || c == "31" || c == "25")
        { i = "2"; }
        else if (c == "38" || c == "39" || c == "37" || c == "40" || c == "36" || c == "41" || c == "35")
        { i = "3"; }
        else if (c == "48" || c == "49" || c == "47" || c == "50" || c == "46" || c == "51" || c == "45")
        { i = "4"; }
        else if (c == "58" || c == "59" || c == "57" || c == "60" || c == "56" || c == "61" || c == "55")
        { i = "5"; }
        else if (c == "68" || c == "69" || c == "67" || c == "70" || c == "66" || c == "71" || c == "65")
        { i = "6"; }
        else { i = " "; };

        if (d == "8" + l || d == "7" + l || d == "9" + l || d == "10" + l || d == "6" + l || d == "5" + l || d
== "11" + l || d == "4" + l || d == "12" + l || d == "3" + l || d == "13" + l || d == "2" + l || d == "14" +
l || d == "1" + l || d == "15" + l)
        { j = "0"; }
        else if (d == "28" + l || d == "27" + l || d == "29" + l || d == "30" + l || d == "26" + l || d ==
"25" + l || d == "31" + l || d == "24" + l || d == "32" + l || d == "23" + l || d == "33" + l || d == "22"
+ l || d == "34" + l || d == "21" + l || d == "35" + l)
        { j = "1"; }
        else if (d == "48" + l || d == "47" + l || d == "49" + l || d == "50" + l || d == "46" + l || d ==
"45" + l || d == "51" + l || d == "44" + l || d == "52" + l || d == "43" + l || d == "53" + l || d == "42"
+ l || d == "54" + l || d == "41" + l || d == "55" + l)
        { j = "2"; }
        else if (d == "68" + l || d == "67" + l || d == "69" + l || d == "70" + l || d == "66" + l || d ==
"65" + l || d == "71" + l || d == "64" + l || d == "72" + l || d == "63" + l || d == "73" + l || d == "6" +
l || d == "74" + l || d == "61" + l || d == "75" + l)
        { j = "3"; }
        else if (d == "88" + l || d == "87" + l || d == "89" + l || d == "90" + l || d == "86" + l || d ==
"85" + l || d == "91" + l || d == "84" + l || d == "92" + l || d == "83" + l || d == "93" + l || d == "82"
+ l || d == "94" + l || d == "81" + l || d == "95" + l)
        { j = "4"; }
        else if (d == "108" + l || d == "107" + l || d == "109" + l || d == "110" + l || d == "106" + l ||
d == "105" + l || d == "111" + l || d == "104" + l || d == "112" + l || d == "103" + l || d == "113" + l
|| d == "102" + l || d == "114" + l || d == "101" + l || d == "115" + l)
        { j = "5"; }
        else if (d == "128" + l || d == "127" + l || d == "129" + l || d == "130" + l || d == "126" + l ||
d == "125" + l || d == "131" + l || d == "124" + l || d == "132" + l || d == "123" + l || d == "133" + l
|| d == "122" + l || d == "134" + l || d == "121" + l || d == "135" + l)
        { j = "6"; }
        else { j = " "; };
        yy.Text = j;
        xx.Text = i;
```

```
}

private void Stop_Click(object sender, EventArgs e)
{
   if (serialPort1.IsOpen)
   {
      serialPort1.Close();
      Start.Enabled = true;
      Stop.Enabled = false;
      textBox1.ReadOnly = true;
   }
}

private void run_Click(object sender, EventArgs e)
{
string a = yy.Text; string b = xx.Text;

   if (a == "1" || a == "2" || a == "3" || a == "4" || a == "5" || a == "6" || a == "0" )
      if (b == "1" || b == "2" || b == "3" || b == "4" || b == "5" || b == "6" || b == "0")
      {
         int j = Convert.ToInt32(b);
         int i = Convert.ToInt32(a);
         if (i == 1 && j == 6)
         { posisi.Text = "selesai"; }
         else if (m_iMaze[i - 1, j] == 100 && posisi.Text == "1")
         {
            maju();
           Thread.Sleep(1000);
            stop();
            aa.Text = "maju";
         }
         else if (m_iMaze[i, j + 1] == 100 && posisi.Text == "1")
         {
            kanan();
            aa.Text = "kanan";
            posisi.Text = "2";
         }
         else if (m_iMaze[i, j - 1] == 100 && posisi.Text == "1")
         {
            kiri();
            aa.Text = "kiri";
            posisi.Text = "4";
         }
         else if (m_iMaze[i, j + 1] == 100 && posisi.Text == "2")
         {
            maju();
            Thread.Sleep(1000);
            stop();
            aa.Text = "maju";
         }
         else if (m_iMaze[i + 1, j] == 100 && posisi.Text == "2")
         {
            kanan();
            aa.Text = "kanan";
            posisi.Text = "3";
```

```
            }
            else if (m_iMaze[i - 1, j] == 100 && posisi.Text == "2")
            {
               kiri();
               aa.Text = "kiri";
               posisi.Text = "1";
            }
            else if (m_iMaze[i + 1, j] == 100 && posisi.Text == "3")
            {
               maju();
               Thread.Sleep(1000);
               stop();
               aa.Text = "maju";
            }
            else if (m_iMaze[i, j - 1] == 100 && posisi.Text == "3")
            {
               kanan();
               aa.Text = "kanan";
               posisi.Text = "4";
            }
            else if (m_iMaze[i, j + 1] == 100 && posisi.Text == "3")
            {
               kiri();
               aa.Text = "kiri";
               posisi.Text = "2";
            }
            else if (m_iMaze[i, j - 1] == 100 && posisi.Text == "4")
            {
               maju();
               Thread.Sleep(1000);
               stop();
               aa.Text = "maju";
            }
            else if (m_iMaze[i + 1, j] == 100 && posisi.Text == "4")
            {
               kanan();
               aa.Text = "kanan";
               posisi.Text = "1";
            }
            else if (m_iMaze[i - 1, j] == 100 && posisi.Text == "4")
            {
               kiri();
               aa.Text = "kiri";
               posisi.Text = "3";
            }

         }



       }

}
```

```
    }
```

MazeSolver.cs

```
using System;
using System.Threading;
namespace Mehroz
{
        /// <summary>
        /// Summary description for MazeSolver.
        /// Class name: MazeSolver
        /// Developed by: Syed Mehroz Alam
        /// Email: smehrozalam@yahoo.com
        /// URL: Programming Home "http://www.geocities.com/smehrozalam/"
        ///
        /// Constructors:
        ///         ( int[,] ): takes 2D integer array
        ///         ( int Rows, int Cols )      initializes the dimensions, indexers may be used
        ///                                                                 to set individual elements'
values
        ///
        /// Properties:
        ///         Rows: returns the no. of rows in the current maze
        ///         Cols: returns the no. of columns in the current maze
        ///         Maze: returns the current maze as a 2D array
        ///         PathCharacter: to get/set the value of path tracing character
        ///         AllowDiagonal: whether diagonal paths are allowed
        ///
        /// Indexers:
        ///         [i,j] = used to s eet/getlements of maze
        ///
        /// Public Methods (Description is given with respective methods' definitions)
        ///                 int[,] FindPath(int iFromY, int iFromX, int iToY, int iToX)
        ///
        /// Private Methods
        ///                 void GetNodeContents(int[,] iMaze, int iNodeNo)
        ///                 void ChangeNodeContents(int[,] iMaze, int iNodeNo, int iNewValue)
        ///                 int[,] Search(int iBeginningNode, int iEndingNode)
        ///
        /// </summary>
        delegate void MazeChangedHandler(int iChanged, int jChanged);
        class MazeSolver
        {

                /// <summary>
                /// Class attributes/members
                /// </summary>
                int[,] m_iMaze;
                int m_iRows;
                int m_iCols;
                int iPath=100;
                bool diagonal=false;
                public event MazeChangedHandler OnMazeChangedEvent;

                /// <summary>
                /// Constructor 1: takes a 2D integer array
```

```
        /// </summary>
        public MazeSolver(int[,] iMaze)
        {
                m_iMaze=iMaze;
                m_iRows=iMaze.GetLength(0);
                m_iCols=iMaze.GetLength(1);
        }

        /// <summary>
        /// Constructor 2: initializes the dimensions of maze,
        /// later, indexers may be used to set individual elements' values
        /// </summary>
        public MazeSolver(int iRows, int iCols)
        {
                m_iMaze=new int[iRows, iCols];
                m_iRows=iRows;
                m_iCols=iCols;
        }

        /// <summary>
        /// Properites:
        /// </summary>
        public int Rows
        {
                get { return m_iRows; }
        }

        public int Cols
        {
                get { return m_iCols; }
        }
        public int[,] GetMaze
        {
                get { return m_iMaze; }
        }
        public int PathCharacter
        {
                get { return iPath; }
                set
                {
                        if (value==0)
                                throw new Exception("Invalid path character
specified");
                        else
                                iPath=value;
                }
        }


        /// <summary>
        /// Indexer
        /// </summary>
        public int this[int iRow, int iCol]
        {
                get { return m_iMaze[iRow,iCol]; }
```

```
            set
            {
                m_iMaze[iRow,iCol]=value;
                if (this.OnMazeChangedEvent!=null)          // trigger event
                    this.OnMazeChangedEvent(iRow, iCol);
            }
        }

        /// <summary>
        /// The function is used to get the contents of a given node in a given maze,
        ///  specified by its node no.
        /// </summary>
        private int GetNodeContents(int[,] iMaze, int iNodeNo)
        {
            int iCols=iMaze.GetLength(1);
            return iMaze[iNodeNo/iCols,iNodeNo-iNodeNo/iCols*iCols];
        }

        /// <summary>
        /// The function is used to change the contents of a given node in a given maze,
        ///  specified by its node no.
        /// </summary>
        private void ChangeNodeContents(int[,] iMaze, int iNodeNo, int iNewValue)
        {
            int iCols=iMaze.GetLength(1);
            iMaze[iNodeNo/iCols,iNodeNo-iNodeNo/iCols*iCols]=iNewValue;
        }

        /// <summary>
        /// This public function finds the shortest path between two points
        /// in the maze and return the solution as an array with the path traced
        /// by "iPath" (can be changed using property "PathCharacter")
        /// if no path exists, the function returns null
        /// </summary>
        public int[,] FindPath(int iFromY, int iFromX, int iToY, int iToX)
        {
            int iBeginningNode = iFromY*this.Cols + iFromX;
            int iEndingNode = iToY*this.Cols + iToX;
            return ( Search(iBeginningNode, iEndingNode) ) ;
        }


        /// <summary>
        /// Internal function for that finds the shortest path using a technique
        /// similar to breadth-first search.
        /// It assigns a node no. to each node(2D array element) and applies the
algorithm
        /// </summary>
        private enum Status
        {       Ready,    Waiting, Processed          }
        private int[,] Search(int iStart, int iStop)
        {
            const int empty=0;

            int iRows=m_iRows;
```

```
                        int iCols=m_iCols;
                        int iMax=iRows*iCols;
                        int[] Queue=new int[iMax];
                        int[] Origin=new int[iMax];
                        int iFront=0, iRear=0;

                        //check if starting and ending points are valid (open)
                        if ( GetNodeContents(m_iMaze, iStart)!=empty ||
GetNodeContents(m_iMaze, iStop)!=empty )
                        {
                                return null;
                        }

                        //create dummy array for storing status
                        int[,] iMazeStatus=new int[iRows,iCols];
                        //initially all nodes are ready
                        for(int i=0;i<iRows;i++)
                                for(int j=0;j<iCols;j++)
                                        iMazeStatus[i,j]=(int)Status.Ready;

                        //add starting node to Q
                        Queue[iRear]=iStart;
                        Origin[iRear]=-1;
                        iRear++;
                        int iCurrent, iLeft, iRight, iTop, iDown, iRightUp, iRightDown,
iLeftUp, iLeftDown;
                        while(iFront!=iRear)        // while Q not empty
                        {
                                if (Queue[iFront]==iStop)            // maze is solved
                                        break;

                                iCurrent=Queue[iFront];

                                iLeft=iCurrent-1;
                                if (iLeft>=0 && iLeft/iCols==iCurrent/iCols)          //if left
node exists
                                        if ( GetNodeContents(m_iMaze, iLeft)==empty )
        //if left node is open(a path exists)
                                                if (GetNodeContents(iMazeStatus, iLeft) ==
(int)Status.Ready)          //if left node is ready
                                                {
                                                        Queue[iRear]=iLeft; //add to Q
                                                        Origin[iRear]=iCurrent;
                                                        ChangeNodeContents(iMazeStatus,
iLeft, (int)Status.Waiting); //change status to waiting
                                                        iRear++;
                                                }

                                iRight=iCurrent+1;
                                if (iRight<iMax && iRight/iCols==iCurrent/iCols)    //if right
node exists
                                        if ( GetNodeContents(m_iMaze, iRight)==empty )
        //if right node is open(a path exists)
                                                if (GetNodeContents(iMazeStatus, iRight)
== (int)Status.Ready)       //if right node is ready
```

```
                                        {
                                                Queue[iRear]=iRight; //add to Q
                                                Origin[iRear]=iCurrent;
                                                ChangeNodeContents(iMazeStatus,
iRight, (int)Status.Waiting); //change status to waiting
                                                iRear++;
                                        }

                                iTop=iCurrent-iCols;
                                if (iTop>=0 )       //if top node exists
                                        if ( GetNodeContents(m_iMaze, iTop)==empty )
        //if top node is open(a path exists)
                                        if (GetNodeContents(iMazeStatus, iTop) ==
(int)Status.Ready)          //if top node is ready
                                        {
                                                Queue[iRear]=iTop; //add to Q
                                                Origin[iRear]=iCurrent;
                                                ChangeNodeContents(iMazeStatus,
iTop, (int)Status.Waiting ); //change status to waiting
                                                iRear++;
                                        }

                                iDown=iCurrent+iCols;
                                if (iDown<iMax )            //if bottom node exists
                                        if ( GetNodeContents(m_iMaze, iDown)==empty )
        //if bottom node is open(a path exists)
                                        if (GetNodeContents(iMazeStatus, iDown)
== (int)Status.Ready)       //if bottom node is ready
                                        {
                                                Queue[iRear]=iDown; //add to Q
                                                Origin[iRear]=iCurrent;
                                                ChangeNodeContents(iMazeStatus,
iDown, (int)Status.Waiting); //change status to waiting
                                                iRear++;
                                        }



                                //change status of current node to processed
                                ChangeNodeContents(iMazeStatus, iCurrent,
(int)Status.Processed);
                                iFront++;

                        }

                        //create an array(maze) for solution
                        int[,] iMazeSolved=new int[iRows,iCols];
                        for(int i=0;i<iRows;i++)
                                for(int j=0;j<iCols;j++)
                                        iMazeSolved[i,j]=m_iMaze[i,j];

                        //make a path in the Solved Maze
                        iCurrent=iStop;
                        ChangeNodeContents(iMazeSolved, iCurrent, iPath);
                        for(int i=iFront; i>=0; i--)
```

```
                    {
                        if (Queue[i]==iCurrent)
                        {
                            iCurrent=Origin[i];
                            if (iCurrent==-1)       // maze is solved
                                return ( iMazeSolved );
                            ChangeNodeContents(iMazeSolved, iCurrent, iPath);
                        }
                    }

                    //no path exists
                    return null;
                }
            }

}
```

StringA.cs

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

namespace MazeSolver
{
   public class StringA
   {
      private string data, delimiter;
      private string[] token;
      private int index;

      public StringA(string dataLine)
      { init(dataLine, " "); }
      private void init(String dataLine, string delim)
      {
         delimiter = delim;
         data = dataLine;
         token = data.Split(delimiter.ToCharArray());
         index = 2;
      }
      public StringA(string dataLine, string delim)
      { init(dataLine, delim); }
      public string nextElement()
      {
         if (index < token.Length)
            return token[index++];
         else
            return "";
      }

   }
}
```

StringT.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Text;

namespace MazeSolver
{
   public class StringT
   {
      private string data, delimiter;
      private string[] token;
      private int index;


      public StringT(string dataLine)
      {
         init(dataLine, " ");
      }
      private void init(String dataLine, string delim)
      {
         delimiter = delim;
         data = dataLine;
         token = data.Split(delimiter.ToCharArray());
         index = 1;
      }
      public StringT(string dataLine, string delim)
      { init(dataLine, delim); }



      public string nextElement()
      {
         if (index < token.Length)
            return token[index++];
         else
            return "";
      }
      //
      // TODO: Add constructor logic here
      //
   }
}
```