LAMPIRAN

MainForm.cs

```
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.IO.Ports;
using Emgu.CV.UI;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;


namespace Face_Recognition
{
    public partial class MainForm : Form
    {
        private SerialPort myport;
        #region variables
        Image<Bgr, Byte> currentFrame; //display image dari webcam
        Image<Gray, byte> result, TrainedFace = null; //used to store the
result image and trained face
        Image<Gray, byte> gray_frame = null; //grayscale current image
aquired from webcam for processing

        Capture grabber; //This is our capture variable

        public CascadeClassifier Face = null;
        MCvFont font;//Our fount for writing within the frame

        int NumLabels;

        //Classifier with default training location
        ClassifierTrain Eigen_Recog = new ClassifierTrain();

        #endregion

        public MainForm()
        {
            InitializeComponent();
            init();
        }
        //membuka data training dr data sebelumnya
        private void init()
        {
            myport = new SerialPort();
            myport.BaudRate = 9600;
            myport.PortName = "COM4";
            myport.Open();
```

```
                        facesDetected[i].Y += (int)(facesDetected[i].Width
* 0.14);
                        facesDetected[i].Height -=
(int)(facesDetected[i].Height * 0.1);
                        facesDetected[i].Width -=
(int)(facesDetected[i].Width * 0.15);


                        result =
currentFrame.Copy(facesDetected[i]).Convert<Gray, byte>().Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                        result._EqualizeHist();
                        //draw the face detected in the 0th (gray) channel
with blue color
                        currentFrame.Draw(facesDetected[i], new
Bgr(Color.Red), 2);

                        if (Eigen_Recog.IsTrained)
                        {
                            string name = Eigen_Recog.Recognise(result);
                            int match_value =
(int)Eigen_Recog.Get_Eigen_Distance;

                            //Draw the label for each face detected and
recognized
                            currentFrame.Draw(name + " ", ref font, new
Point(facesDetected[i].X - 2, facesDetected[i].Y - 2), new
Bgr(Color.LightGreen));
                            label4.Text = name;


                            if (name == "harif")
                                myport.WriteLine("A");

                            if (name == "helen")
                                myport.WriteLine("A");

                            if (name == "sinar")
                                myport.WriteLine("A");

                            if (name == "noone")
                              myport.WriteLine("B");

                        }


                    }
                    catch
                    {

                    }
                });

                image_PICBX.Image = currentFrame.ToBitmap();
```

```
            }
        }

        //ADD Picture box and label to a panel for each face
        int faces_count = 0;
        int faces_panel_Y = 0;
        int faces_panel_X = 0;
        void if_label()
        {
            label4.Text = "sinar";
            myport.WriteLine("A");
        }



        void Clear_label4()
        {
            label4.Text = " ";
        }

        void Clear_Faces_Found()
        {
            this.Faces_Found_Panel.Controls.Clear();
            faces_count = 0;
            faces_panel_Y = 0;
            faces_panel_X = 0;

        }
        void ADD_Face_Found(Image<Gray, Byte> img_found, string
name_person, int match_value)
        {
            PictureBox PI = new PictureBox();
            PI.Location = new Point(faces_panel_X, faces_panel_Y);
            PI.Height = 80;
            PI.Width = 80;
            PI.SizeMode = PictureBoxSizeMode.StretchImage;
            PI.Image = img_found.ToBitmap();
            Label LB = new Label( );
            LB.Text = name_person + " " + match_value.ToString();

            LB.Location = new Point(faces_panel_X, faces_panel_Y + 80);
            //LB.Width = 80;
            LB.Height = 15;

            this.Faces_Found_Panel.Controls.Add(PI);
            this.Faces_Found_Panel.Controls.Add(LB);
            faces_count++;
            if (faces_count == 2)
            {
                faces_panel_X = 0;
                faces_panel_Y += 100;
                faces_count = 0;
            }
            else faces_panel_X += 85;

            if (Faces_Found_Panel.Controls.Count > 10)
```

```
            {
                Clear_Faces_Found();

            }

        }

        //Menu Opeartions
        private void exitToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            this.Dispose();
        }

        //Unknow Eigen face calibration
        private void Eigne_threshold_txtbxChanged(object sender, EventArgs
e)
        {
            try
            {
                Eigen_Recog.Set_Eigen_Threshold =
Math.Abs(Convert.ToInt32(Eigne_threshold_txtbx.Text));
                message_bar.Text = "Eigen Threshold Set";
            }
            catch
            {
                message_bar.Text = "Error in Threshold input please use
int";
            }
        }

        private void MainForm_Load(object sender, EventArgs e)
        {

        }

        private void Faces_Found_Panel_Paint(object sender, PaintEventArgs
e)
        {

        }

        private void image_PICBX_Click(object sender, EventArgs e)
        {

        }


    }
}
```

TrainingForm.cs

```
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 Emgu.CV.UI;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;

using System.IO;
using System.Drawing.Imaging;
using System.Xml;
using System.Threading;


namespace Face_Recognition
{
    public partial class TrainingForm : Form
    {
        #region Variables
        //Camera specific
        Capture grabber;

        //Images for finding face
        Image<Bgr, Byte> currentFrame;
        Image<Gray, byte> result = null;
        Image<Gray, byte> gray_frame = null;

        //Classifier
        CascadeClassifier Face;

        //For aquiring 10 images in a row
        List<Image<Gray, byte>> resultImages = new List<Image<Gray,
byte>>();
        int results_list_pos = 0;
        int num_faces_to_aquire = 10;
        bool RECORD = false;

        //Saving Jpg
        List<Image<Gray, byte>> ImagestoWrite = new List<Image<Gray,
byte>>();
        EncoderParameters ENC_Parameters = new EncoderParameters(1);
        EncoderParameter ENC = new
EncoderParameter(System.Drawing.Imaging.Encoder.Quality, 100);
        ImageCodecInfo Image_Encoder_JPG;

        //Saving XAML Data file
        List<string> NamestoWrite = new List<string>();
        List<string> NamesforFile = new List<string>();
        XmlDocument docu = new XmlDocument();

        //Variables
        MainForm Parent;
        #endregion

        public TrainingForm(MainForm parent)
```

```
        {
            InitializeComponent();
            Parent = parent;
            Face = new CascadeClassifier(Application.StartupPath +
"/Cascades/haarcascade_frontalface_default.xml");
            ENC_Parameters.Param[0] = ENC;
            Image_Encoder_JPG = GetEncoder(ImageFormat.Jpeg);
            initialise_capture();
        }

        private void Training_Form_FormClosing(object sender,
FormClosingEventArgs e)
        {
            stop_capture();
            Parent.retrain();
            Parent.initialise_capture();
        }

        //Camera Start Stop
        public void initialise_capture()
        {
            grabber = new Capture();
            grabber.QueryFrame();
            //Initialize the FrameGraber event
            Application.Idle += new EventHandler(FrameGrabber);
        }
        private void stop_capture()
        {
            Application.Idle -= new EventHandler(FrameGrabber);
            if (grabber != null)
            {
                grabber.Dispose();
            }
            //Initialize the FrameGraber event
        }

        //Process Frame
        void FrameGrabber(object sender, EventArgs e)
        {
            //Get the current frame form capture device
            currentFrame = grabber.QueryFrame().Resize(320, 240,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);

            //Convert it to Grayscale
            if (currentFrame != null)
            {
                gray_frame = currentFrame.Convert<Gray, Byte>();

                //Face Detector
                //MCvAvgComp[][] facesDetected =
gray_frame.DetectHaarCascade(Face, 1.2, 10,
Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING, new Size(20, 20));
//old method
                Rectangle[] facesDetected =
Face.DetectMultiScale(gray_frame, 1.2, 10, new Size(50, 50), Size.Empty);

                //Action for each element detected
```

```
                for(int i = 0; i< facesDetected.Length; i++)// (Rectangle
face_found in facesDetected)
                {
                    //This will focus in on the face from the haar results
its not perfect but it will remove a majoriy
                    //of the background noise

                    facesDetected[i].X += (int)(facesDetected[i].Height *
0.1);
                    facesDetected[i].Y += (int)(facesDetected[i].Width *
0.14);
                    facesDetected[i].Height -=
(int)(facesDetected[i].Height * 0.1);
                    facesDetected[i].Width -= (int)(facesDetected[i].Width
* 0.15);

                    result =
currentFrame.Copy(facesDetected[i]).Convert<Gray, byte>().Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                   // result._EqualizeHist();
                    face_PICBX.Image = result.ToBitmap();
                    //draw the face detected in the 0th (gray) channel with
blue color
                    currentFrame.Draw(facesDetected[i], new Bgr(Color.Red),
2);

                }
                if (RECORD && facesDetected.Length > 0 &&
resultImages.Count < num_faces_to_aquire)
                {
                    resultImages.Add(result);
                    count_lbl.Text = "Count: " +
resultImages.Count.ToString();
                    if (resultImages.Count == num_faces_to_aquire)
                    {
                        ADD_BTN.Enabled = true;
                        NEXT_BTN.Visible = true;
                        PREV_btn.Visible = true;
                        count_lbl.Visible = false;
                        Single_btn.Visible = true;
                        ADD_ALL.Visible = true;
                        RECORD = false;
                        Application.Idle -= new EventHandler(FrameGrabber);
                    }
                }

                image_PICBX.Image = currentFrame.ToBitmap();
            }
        }

        //Saving The Data
        private bool save_training_data(Image face_data)
        {
            try
            {
                Random rand = new Random();
                bool file_create = true;
```

```
                string facename = "face_" + NAME_PERSON.Text + "_" +
rand.Next().ToString() + ".jpg";
                while (file_create)
                {

                    if (!File.Exists(Application.StartupPath +
"/TrainedFaces/" + facename))
                    {
                        file_create = false;
                    }
                    else
                    {
                        facename = "face_" + NAME_PERSON.Text + "_" +
rand.Next().ToString() + ".jpg";
                    }
                }


                if(Directory.Exists(Application.StartupPath +
"/TrainedFaces/"))
                {
                    face_data.Save(Application.StartupPath +
"/TrainedFaces/" + facename, ImageFormat.Jpeg);
                }
                else
                {
                    Directory.CreateDirectory(Application.StartupPath +
"/TrainedFaces/");
                    face_data.Save(Application.StartupPath +
"/TrainedFaces/" + facename, ImageFormat.Jpeg);
                }
                if (File.Exists(Application.StartupPath +
"/TrainedFaces/TrainedLabels.xml"))
                {
                    //File.AppendAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt", NAME_PERSON.Text + "\n\r");
                    bool loading = true;
                    while (loading)
                    {
                        try
                        {
                            docu.Load(Application.StartupPath +
"/TrainedFaces/TrainedLabels.xml");
                            loading = false;
                        }
                        catch
                        {
                            docu = null;
                            docu = new XmlDocument();
                            Thread.Sleep(10);
                        }
                    }

                    //Get the root element
                    XmlElement root = docu.DocumentElement;

                    XmlElement face_D = docu.CreateElement("FACE");
```

```
                    XmlElement name_D = docu.CreateElement("NAME");
                    XmlElement file_D = docu.CreateElement("FILE");

                    //Add the values for each nodes
                    //name.Value = textBoxName.Text;
                    //age.InnerText = textBoxAge.Text;
                    //gender.InnerText = textBoxGender.Text;
                    name_D.InnerText = NAME_PERSON.Text;
                    file_D.InnerText = facename;

                    //Construct the Person element
                    //person.Attributes.Append(name);
                    face_D.AppendChild(name_D);
                    face_D.AppendChild(file_D);

                    //Add the New person element to the end of the root
element
                    root.AppendChild(face_D);

                    //Save the document
                    docu.Save(Application.StartupPath +
"/TrainedFaces/TrainedLabels.xml");
                    //XmlElement child_element =
docu.CreateElement("FACE");
                    //docu.AppendChild(child_element);
                    //docu.Save("TrainedLabels.xml");
                }
                else
                {
                    FileStream FS_Face =
File.OpenWrite(Application.StartupPath +
"/TrainedFaces/TrainedLabels.xml");
                    using (XmlWriter writer = XmlWriter.Create(FS_Face))
                    {
                        writer.WriteStartDocument();
                        writer.WriteStartElement("Faces_For_Training");

                        writer.WriteStartElement("FACE");
                        writer.WriteElementString("NAME",
NAME_PERSON.Text);
                        writer.WriteElementString("FILE", facename);
                        writer.WriteEndElement();

                        writer.WriteEndElement();
                        writer.WriteEndDocument();
                    }
                    FS_Face.Close();
                }

                return true;
            }
            catch (Exception ex)
            {
                return false;
            }

        }
```

```
        private ImageCodecInfo GetEncoder(ImageFormat format)
        {
            ImageCodecInfo[] codecs = ImageCodecInfo.GetImageDecoders();
            foreach (ImageCodecInfo codec in codecs)
            {
                if (codec.FormatID == format.Guid)
                {
                    return codec;
                }
            }
            return null;
        }

        //Delete all the old training data by simply deleting the folder
        private void Delete_Data_BTN_Click(object sender, EventArgs e)
        {
            if (Directory.Exists(Application.StartupPath +
"/TrainedFaces/"))
            {
                Directory.Delete(Application.StartupPath +
"/TrainedFaces/", true);
                Directory.CreateDirectory(Application.StartupPath +
"/TrainedFaces/");
            }
        }

        //Add the image to training data
        private void ADD_BTN_Click(object sender, EventArgs e)
        {
            if (resultImages.Count == num_faces_to_aquire)
            {
                if (!save_training_data(face_PICBX.Image))
MessageBox.Show("Error", "Error in saving file info. Training data not
saved", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else
            {
                stop_capture();
                if (!save_training_data(face_PICBX.Image))
MessageBox.Show("Error", "Error in saving file info. Training data not
saved", MessageBoxButtons.OK, MessageBoxIcon.Error);
                initialise_capture();
            }
        }
        private void Single_btn_Click(object sender, EventArgs e)
        {
            RECORD = false;
            resultImages.Clear();
            NEXT_BTN.Visible = false;
            PREV_btn.Visible = false;
            Application.Idle += new EventHandler(FrameGrabber);
            Single_btn.Visible = false;
            count_lbl.Text = "Count: 0";
            count_lbl.Visible = true;
        }
        //Get 10 image to train
        private void RECORD_BTN_Click(object sender, EventArgs e)
```

```
        {
            if (NAME_PERSON.Text != "")
            {
                if (RECORD)
                {
                    RECORD = false;
                }
                else
                {
                    if (resultImages.Count == 10)
                    {
                        resultImages.Clear();
                        Application.Idle += new EventHandler(FrameGrabber);
                    }
                    RECORD = true;
                    ADD_BTN.Enabled = false;
                }
            }
            else
            {
                MessageBox.Show("Write Person Name First!! ");
            }

        }
        private void NEXT_BTN_Click(object sender, EventArgs e)
        {
            if (results_list_pos < resultImages.Count - 1)
            {
                face_PICBX.Image =
resultImages[results_list_pos].ToBitmap();
                results_list_pos++;
                PREV_btn.Enabled = true;
            }
            else
            {
                NEXT_BTN.Enabled = false;
            }
        }
        private void PREV_btn_Click(object sender, EventArgs e)
        {
            if (results_list_pos > 0)
            {
                results_list_pos--;
                face_PICBX.Image =
resultImages[results_list_pos].ToBitmap();
                NEXT_BTN.Enabled = true;
            }
            else
            {
                PREV_btn.Enabled = false;
            }
        }
        private void ADD_ALL_Click(object sender, EventArgs e)
        {
            for(int i = 0; i<resultImages.Count;i++)
            {
                face_PICBX.Image = resultImages[i].ToBitmap();
```

```
                if (!save_training_data(face_PICBX.Image))
MessageBox.Show("Error", "Error in saving file info. Training data not
saved", MessageBoxButtons.OK, MessageBoxIcon.Error);
                Thread.Sleep(100);
            }
            ADD_ALL.Visible = false;
            //restart single face detection
            Single_btn_Click(null, null);
        }

        private void TrainingForm_Load(object sender, EventArgs e)
        {

        }

        private void image_PICBX_Click(object sender, EventArgs e)
        {

        }

    }
}
```

Servo

```
//control servo motor with serial monitering window - by ujash patel

  //for more go to www.uu-machinetool.blogspot.com

#include <Servo.h>


Servo myservo;  // create servo object to control a servo

              // a maximum of eight servo objects can be created


int pos = 0;

int servodata;


void setup()

{

 Serial.begin(9600);
```

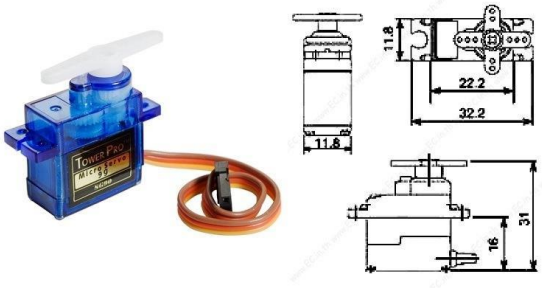
```
Serial.println("Wisuda");
Serial.println("Sinar Monika ");
Serial.println("0613 4035 1630");
Serial.println("8 TEB");
  myservo.attach(9);

}

void loop()
{
  if (Serial.available() > 0)
  {
   servodata = Serial.read();

    if(servodata == 'A') // Single Quote! This is a character.
    {
      Serial.println("speed 1 is selected");
      Serial.println("DELAY 15MS");
                                // in steps of 1 degAAree
       // goes from 0 degrees to 180 degrees
    // goes from 0 degrees to 180 degrees
                               // in steps of 1 degree
                               // in steps of 1 degree
    myservo.write(0);              // tell servo to go to position in variable 'pos'
    delay(15);                     // waits 15ms for the servo to reach the position
```

```
                    // waits 15ms for the servo to reach the position
  }
    // waits 15ms for the servo to reach the position



    }
    if(servodata== 'B')
    {
      Serial.println("No speed");
      Serial.println(" Stop ");
       // goes from 0 degrees to 180 degrees                    // in steps of 1
degree
                              // in steps of 1 degAAree
    myservo.write(40);
    delay(15);
 }
}
```

Datasheet

Servo

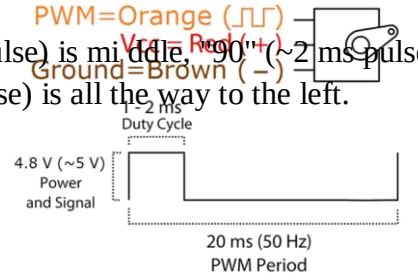
SG90 9 g Micro Servo



Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but *SMALLER*. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

Specifications

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 µs
- Temperature range: 0 ºC – 55 º



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.