```
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;
using System.IO.Ports;
namespace VibrationScopev1._2_VSc1._2_
{
    public partial class Form1 : Form
    {
        Form2 child1 = new Form2(); //this form is for setting serial properties and
                                    graphic properties
        Form3 child2 = new Form3(); //plotting and displaying data in graph
        Form4 child3 = new Form4(); //displaying data in text
        Form5 child4 = new Form5(); //Sending Serial parameter
        public SerialPort VCP = new SerialPort();
        private static string port, baud="38400";
        private static byte receive = 0;
        private static int samples=16384;
        bool connection = false;
        bool coba = false;
        static int accelparametervalue;
        int cnt,cnt2,state=0;
        static List<byte> bytebuffer = new List<byte>();
        List<byte> bytecounter = new List<byte>();
        static byte[] bytebufferarray = new byte[2*samples];
        int[] xstreambuffer = new int[samples];
        int[] ystreambuffer = new int[samples];
        int[] ystreambufferlow = new int[samples / 2];
        int[] ystreambufferhigh = new int[samples / 2];
        int[] zstreambuffer = new int[samples];
        int[] zstreambufferlow = new int[samples / 2];
        int[] zstreambufferhigh = new int[samples / 2];
        int[] streambuffer2 = new int[11] { -110, -20, 30, 100, 150, -70, 50, 20, 200,
                                            30, -190 };
        static int[] dataX = new int[4096];
        static int[] dataY = new int[4096];
        static int[] dataZ = new int[samples];
        int[] AccelData = new int[2048];
        byte[] bytetosendtoWSS = new byte[1];
        #region Custom Events
        public delegate void sendserialtosmartsensorEvemt(byte code);
        public event sendserialtosmartsensorEvemt sendserialtosmartsensor;
        #endregion
        public Form1()
        {
            InitializeComponent();
            child1.MdiParent = this;
            child2.MdiParent = this;
            child3.MdiParent = this;
            child4.MdiParent = this;
            VCP.DataReceived += new SerialDataReceivedEventHandler(ReceiveSerialData);
            child4.sendtoWSS += new Form5.SendToWSS(receiveactivate);
            child4.AccelChannel += new AccelChannelChoice(child4_AccelChannel);
        }
```

```
private void Form1_Load(object sender, EventArgs e)
{
    child1.Show();
}
#region function
private void receivefrchild1(string PORT, string BAUD, string YM, string XM,
                             string YINC, string XINC)
{
    PORT = port;
    BAUD = baud;

}
public void displayontextbox(string mess)
{

    textBox1.AppendText(mess+Environment.NewLine);
}
private string ConvertToHEXstring(byte[]data)
{
    StringBuilder sb = new StringBuilder(data.Length * 3);
    foreach (byte b in data)
        sb.Append(Convert.ToString(b, 16).PadLeft(2, '0').PadRight(3, ' '));
    return sb.ToString().ToUpper();
}
private string ConvertDataAccelToDecimalString(int[] data)
{
    StringBuilder sb = new StringBuilder(data.Length * 3);
    foreach (int b in data)
        sb.Append((Convert.ToInt16(b)).ToString().PadLeft(4, ' ').PadRight(5,
                  ' '));
    return sb.ToString();
}
private void publishtext(string message,string message2,string message3)
{
    publishtotextonform4 pub = new
                         publishtotextonform4(child3.publishtotextbox);
    pub.Invoke(message, message2, message3);
}
private void publishtext2(string message)
{
    publishtotextboxform5 pub=new
                         publishtotextboxform5(child4.publishtotextbox);
    pub.Invoke(message);
}
private void receiveactivate(object sender, SerialEventArgs e)
{
    if (VCP.IsOpen)
    {
        bytetosendtoWSS[0] = e.COMMAND;
        VCP.Write(bytetosendtoWSS, 0, 1);
    }
}
void child4_AccelChannel(object sender, ChoiceEventArgs e)
{
    accelparametervalue = e.CHOICE;
}
private void stopserial()
{
    try
    {
        VCP.Close();
        child2.CONNECTED = false;
    }
```

```
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
#endregion
#region property for receiving data
public string PORTNAME
{
    set { port=value;}
}
public string BAUDRATE
{
    set { baud = value; }
}
public int SAMPLES
{
    set { samples = value; }
}
public byte SERIALRECEIVE
{
    set
    {
        receive = value;
        if (sendserialtosmartsensor != null)
        { sendserialtosmartsensor(value); }
    }

}
#endregion
#region sending data&parameters from delegates
private void SendDataX(int[] datax)
{
    SENDZDATATOPLOT SX = new SENDZDATATOPLOT(child2.ReceiveDataX);
    SX.Invoke(datax);
}
private void SendDataY(int[] datay)
{
    SENDZDATATOPLOT SY = new SENDZDATATOPLOT(child2.ReceiveDataY);
    SY.Invoke(datay);
}
private void SendDataZ(int[] dataz)
{
    SENDZDATATOPLOT SZ = new SENDZDATATOPLOT(child2.ReceiveDataZ);
    SZ.Invoke(dataz);
}
#endregion
#region toolstrip buttons
////toolstrip connect serial////
private void toolStripButton1_Click(object sender, EventArgs e)
{
    connection = !connection;
    if (connection)
    {
        VCP.PortName = port;
        VCP.BaudRate = int.Parse(baud);
        try
        {
            VCP.Open();
            child2.CONNECTED = true;
        }
```

```
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        else
        {
            Thread threadstopserial = new Thread(stopserial);
            threadstopserial.Start();
            Thread.Sleep(1000);
            if (!threadstopserial.IsAlive) threadstopserial.Abort();
            child2.CONNECTED = false;
        }
    }
    ////toolstrip save////
    private void toolStripButton2_Click(object sender, EventArgs e)
    {
        //textBox1.Text = receive.ToString();
        timer1.Enabled=false;
    }
    ////toolstrip settings////
    private void toolStripButton3_Click(object sender, EventArgs e)
    {
        child1.Show();
        child2.Hide();
        child1.Location = new Point(5, 5);
    }
    ////toolstrip graphics////
    private void toolStripButton4_Click(object sender, EventArgs e)
    {
        child1.Hide();
        child2.HIDE = false;
        child2.Show();
        child2.Location = new Point(5, 10);
        child2.Focus();
    }
    ////displaying text////
    private void toolStripButton5_Click(object sender, EventArgs e)
    {
        child3.Show();
        timer1.Enabled = true;
    }
    ////Send Serial Data to WSS////
    private void toolStripButton6_Click(object sender, EventArgs e)
    {
        connection = !connection;
            if (connection)
            {
                VCP.PortName = port;
                VCP.BaudRate = int.Parse(baud);
                try
                {
                    VCP.Open();
                    child2.CONNECTED = true;
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                child4.Show();
            }
```

```
        else
        {
            Thread threadstopserial = new Thread(stopserial);
            threadstopserial.Start();
            Thread.Sleep(1000);
            if (!threadstopserial.IsAlive) threadstopserial.Abort();
            child2.CONNECTED = false;
        }
}
#endregion
#region receiving data from serialport
private void ReceiveSerialData(object objek, SerialDataReceivedEventArgs e)
{
    int totalbytes = VCP.BytesToRead;
    byte[] buffer = new byte[totalbytes];

    if (VCP.IsOpen)
    {
        #region receiving data packet
        //receiving data packet byte//
        VCP.Read(buffer, 0, totalbytes);
        for (int i = 0; i < totalbytes; i++)
        {
            bytebuffer.Add(buffer[i]);
            bytecounter.Add(buffer[i]);
        }
        //cnt = bytebuffer.Count;
        //bytebuffer.Clear();
        #endregion
    }
}
#endregion
#region timer for looping continously
private void timer1_Tick(object sender, EventArgs e)
{
    if (VCP.IsOpen)
    {
        cnt = bytecounter.Count;
        switch (accelparametervalue)
        {
            case 1:
                #region X-SELECTED
                {
                    ////////X SELECTED//////////////
                    if (bytebuffer.Count > samples)
                    {
                        for (int i = 0; i < samples; i++)
                        {
                            bytebufferarray[i] = bytebuffer[i];
                        }
                        for (int i = 0; i < (int)samples / 2; i++)
                        {
                            dataX[i] = Convert.ToInt32((bytebufferarray[2 * i]
                                       << 8 | bytebufferarray[1 + 2 * i]) -
                                       32768);
                        }
                        for (int i = 0; i < (int)samples / 2; i++)
                        {
                            if (dataX[i] < 512 && dataX[i] > -512)
                            {
                                publishtext(dataX[i].ToString(), "", "");
                                xstreambuffer[i]=dataX[i];
                            }
```

```
                }
                textBox1.AppendText(cnt.ToString() +
                        Environment.NewLine);
                SendDataX(xstreambuffer);
                publishtext2(cnt.ToString());
                bytebuffer.Clear();
            }
        }
        #endregion
        break;
    case 2:
        #region Y-SELECTED
        {
            ////////Y SELECTED//////////////
            if (bytebuffer.Count > samples)
            {
                for (int i = 0; i < samples; i++)
                {
                    bytebufferarray[i] = bytebuffer[i];
                }
                for (int i = 0; i < (int)samples / 2; i++)
                {
                    dataY[i] = Convert.ToInt32((bytebufferarray[2 * i]
                               << 8 | bytebufferarray[1 + 2 * i]) -
                               32768);
                }
                for (int i = 0; i < (int)samples / 2; i++)
                {
                    if (dataY[i] < 512 && dataY[i] > -512)
                    {
                        publishtext(dataY[i].ToString(), "", "");
                        ystreambuffer[i] = dataY[i];
                    }
                }
                //child2.RECEIVEYDATA = ystreambuffer;
                SendDataY(ystreambuffer);
                publishtext2(cnt.ToString());
                bytebuffer.Clear();
            }
        }
        #endregion
        break;
    case 3:
        #region Z-SELECTED
        {
            cnt = bytecounter.Count;
            ////////Z SELECTED//////////////
            if (bytebuffer.Count > samples)
            {
                for (int i = 0; i < samples; i++)
                {
                    bytebufferarray[i] = bytebuffer[i];
                }
                for (int i = 0; i < (int)samples / 2; i++)
                {
                    dataZ[i] = Convert.ToInt32((bytebufferarray[2 * i]
                               << 8 | bytebufferarray[1 + 2 * i]) -
                               32768);
                }
                for (int i = 0; i < (int)samples / 2; i++)
                {
                    if (dataZ[i] < 512 && dataZ[i] > -512)
                    {
```

```
                    publishtext(dataZ[i].ToString(), "", "");
                    zstreambuffer[i] = dataZ[i];
                }
            }
            SendDataZ(zstreambuffer);
            publishtext2(cnt.ToString());
            bytebuffer.Clear();
        }
    }
    #endregion
    break;
case 4:
    #region XY-SELECTED
    {
        ////////XY SELECTED//////////////
        if (bytebuffer.Count > samples)
        {
            for (int i = 0; i < samples; i++)
            {
                bytebufferarray[i] = bytebuffer[i];
            }
            for (int i = 0; i < (samples / 4); i++)
            {
                dataX[i] = Convert.ToInt32((bytebufferarray[4 * i]
                          << 8 | bytebufferarray[1 + 4 * i]) -
                          32768);
                dataY[i] = Convert.ToInt32((bytebufferarray[2 + 4
                          * i] << 8 | bytebufferarray[3 + 4 * i])
                          - 32768);
            }
            for (int i = 0; i < (int)samples / 4; i++)
            {
                if ((dataX[i] < 512 && dataX[i] > -512) &&
                   (dataY[i] < 512 && dataY[i] > -512))
                {
                    publishtext(dataX[i].ToString(),
                                dataY[i].ToString(), "");
                }
            }
            textBox1.AppendText(cnt.ToString() + "\t" +
                     samples.ToString() + Environment.NewLine);
            bytebuffer.Clear();
        }
    }
    #endregion
    break;
case 5:
    #region YZ-SELECTED
    {
        ////////YZ SELECTED//////////////
        if (bytebuffer.Count > 2*samples)
        {
            for (int i = 0; i < 2*samples; i++)
            {
                bytebufferarray[i] = bytebuffer[i];
            }
            for (int i = 0; i < (samples / 2); i++)
            {
                dataY[i] = Convert.ToInt32((bytebufferarray[4 * i]
                          << 8 | bytebufferarray[1 + 4 * i]) -
                          32768);
```

```
                    dataZ[i] = Convert.ToInt32((bytebufferarray[2 + 4
                               * i] << 8 | bytebufferarray[3 + 4 * i])
                               - 32768);
                }

                for (int i = 0; i < (int)samples / 2; i++)
                {
                    if ((dataY[i] < 512 && dataY[i] > -512) &&
                        (dataZ[i] < 512 && dataZ[i] > -512))
                    {
                        publishtext("", dataY[i].ToString(),
                                    dataZ[i].ToString());
                        ystreambuffer[i] = dataY[i];
                        zstreambuffer[i] = dataZ[i];
                    }
                }
                    SendDataY(ystreambuffer);
                    SendDataZ(zstreambuffer);
                    bytebuffer.Clear();
            }
        }
        #endregion
        break;
    case 6:
        #region XZ-SELECTED
        {
            ////////XZ SELECTED//////////////
            if (bytebuffer.Count > samples)
            {
                for (int i = 0; i < samples; i++)
                {
                    bytebufferarray[i] = bytebuffer[i];
                }
                for (int i = 0; i < (samples / 4); i++)
                {
                    dataX[i] = Convert.ToInt32((bytebufferarray[4 * i]
                               << 8 | bytebufferarray[1 + 4 * i]) -
                               32768);
                    dataZ[i] = Convert.ToInt32((bytebufferarray[2 + 4
                               * i] << 8 | bytebufferarray[3 + 4 * i])
                               - 32768);
                }
                for (int i = 0; i < (int)samples / 4; i++)
                {
                    if ((dataX[i] < 512 && dataX[i] > -512) &&
                        (dataZ[i] < 512 && dataZ[i] > -512))
                    {
                        publishtext(dataX[i].ToString(), "",
                                    dataZ[i].ToString());
                    }
                }
                textBox1.AppendText(cnt.ToString() + "\t" +
                         samples.ToString() + Environment.NewLine);
                bytebuffer.Clear();
            }
        }
        #endregion
        break;
```

```
        case 7:
            #region XYZ-SELECTED
            {
                ////////XYZ SELECTED//////////////
                if (bytebuffer.Count > 3*samples)
                {
                    for (int i = 0; i < 3*samples; i++)
                    {
                        bytebufferarray[i] = bytebuffer[i];
                    }
                    for (int i = 0; i < (int)samples / 2; i++)
                    {
                        dataX[i] = Convert.ToInt32((bytebufferarray[6 * i]
                                  << 8 | bytebufferarray[1 + 6 * i]) -
                                  32768);
                        dataY[i] = Convert.ToInt32((bytebufferarray[6 * i
                                  + 2] << 8 | bytebufferarray[3 + 6 * i])
                                  - 32768);
                        dataZ[i] = Convert.ToInt32((bytebufferarray[4 + 6
                                  * i] << 8 | bytebufferarray[5 + 6 * i])
                                  - 32768);
                    }
                    for (int i = 0; i < (int)samples / 2; i++)
                    {
                        if ((dataX[i] < 512 && dataX[i] > -512) &&
                            (dataY[i] < 512 && dataY[i] > -512) &&
                            (dataZ[i] < 512 && dataZ[i] > -512))
                        {
                            publishtext(dataX[i].ToString(),
                                dataY[i].ToString(), dataZ[i].ToString());
                            xstreambuffer[i] = dataX[i];
                            ystreambuffer[i] = dataY[i];
                            zstreambuffer[i] = dataZ[i];
                        }
                    }
                    SendDataX(xstreambuffer);
                    SendDataY(ystreambuffer);
                    SendDataZ(zstreambuffer);
                    bytebuffer.Clear();
                }
            }
            #endregion
            break;
        case 8:
            #region check wireless
            {
                if (bytebuffer.Count > 7)
                {
                    for (int i = 2; i < 8; i++)
                    {
                        bytebufferarray[i] = bytebuffer[i];
                    }
                    publishtext2(cnt.ToString());
                }
            }
            #endregion
            break;
        default:
            {
                publishtext("Cannot Display Data", "", "");
            }
            break;
    }
```

```
            }
        }
        #endregion
    }

    public class CustomEventArgs : System.EventArgs
    {
        private static byte receiveddata;
        public byte RECEIVEDATA
        {
            set { receiveddata = value; }
        }
    }
    #region Define delegate & Events
    public delegate void ParameterFromChild2(string port, string baud, string Ymaks,
                                             string Xmax, string Yincrement, string
                                             Xincrement);
    public delegate void ReceiveFromChildren(object ob, CustomEventArgs e);
    public delegate void ReceiveFromParent(object ohj, SerialDataReceivedEventArgs e);
    public delegate void SendSerialParameter(string port,string baud);
    public delegate void SENDXDATATOPLOT(int[] datax);
    public delegate void SENDYDATATOPLOT(int[] datay);
    public delegate void SENDZDATATOPLOT(int[] dataz);
    public delegate void SendSerialtoGraph(int[] data);
    public delegate void publishtotextonform4(string message,string message2,string
                                              message3);
    #endregion
}


public partial class Form2 : Form
    {
        string grid;
        public Form2()
        {
            InitializeComponent();
            fillbox();
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            BaudBox.SelectedIndex = 3;
            checkBox1.Checked = true;
        }
        #region custom event handler
        public event ReceiveFromChildren receivefromchildren;

        #endregion
        #region Function
        private void fillbox()
        {
            string[] Portname = System.IO.Ports.SerialPort.GetPortNames();
            int[] Baud = { 2400, 4800, 9600, 19200, 38400, 57600, 115200 };
            PortBox.DataSource = Portname;
            BaudBox.DataSource = Baud;
        }
        #endregion

        #region method
        private void pass()
        {
            Form1 parent = new Form1();
            Form3 FormGraph = new Form3();
```

```
            parent.PORTNAME = PortBox.SelectedItem.ToString();
            parent.BAUDRATE = BaudBox.SelectedItem.ToString();
            parent.SAMPLES = (int)numericUpDown1.Value;
            FormGraph.Y_MAKS = (float)YnumericUpDown.Value;
            FormGraph.X_MAKS = (float)XnumericUpDown.Value;
            FormGraph.Y_INCREMENT = (int)YnumericUpDown2.Value;
            FormGraph.X_INCREMENT = (int)XnumericUpDown2.Value;
            FormGraph.SAMPLES = (int)numericUpDown1.Value;
            FormGraph.GRID = grid;
            FormGraph.Activate();
        }
        private void PassviaDelegate()
        {
            SendSerialParameter Send = new
                                SendSerialParameter(passingSerialParameter);
        }
        private void passingSerialParameter(string par1, string par2)
        {

        }
        #endregion
        #region button clicked
        ////Default Button///
        private void button1_Click(object sender, EventArgs e)
        {
            BaudBox.SelectedIndex = 4;
        }
        ////OK Button////
        private void button2_Click(object sender, EventArgs e)
        {
            pass();
            this.Hide();
        }
        ////Cancel Button////
        private void button3_Click(object sender, EventArgs e)
        {
            this.Hide();
        }
        #endregion
        #region Form Closing
        private void Form2_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (e.CloseReason == CloseReason.UserClosing)
            {
                e.Cancel = true;
                this.Hide();
            }
        }
        #endregion

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox1.Checked)
            {
                grid = "on";
            }
            else
            {
                grid = "off";
            }
        }

    }
```

```
public partial class Form3 : Form
    {
        public delegate void ReceiveSerialDatatoBePlot(object sender,PlotEventArgs e);
        public event ReceiveSerialDatatoBePlot DataPlot;
        enum warna { putih, hitam };
        static int[] xgraphicdata = new int[samples];
        static int[] ygraphicdata = new int[samples];
        static int[] zgraphicdata=new int[samples];
        static bool connected = false;
        bool hide = true;
        static bool xselect = false;
        static bool yselect = false;
        static bool zselect = false;
        #region fitting the graph
        int YORIGIN = 200;
        #endregion
        object yobjek = null;
        object zobjek = null;
        static string newgrid, gridon;
        //string newgrid,gridon;
        float Xborder = 400, Yborder = 200;
        float XhalfLength = 350, YhalfLength = 150;
        float xlegend = 880, ylegnd = 87.5f;
        float xhalflegend = 100, yhalflegend = 37.5f;
        float Xmaks = 1, Ymaks = 1;
        int Xincrement = 25, Yincrement = 10;
        static float newYmaks=1, newXmaks=1;
        static int newXincrement=25, newYincrement=10;
        static int samples = 18000;
        private Color[] clr = { Color.White, Color.Black };
        PointF[] p = new PointF[6];
        static PointF[] Xpointtodraw = new PointF[samples * 2];
        static PointF[] Ypointtodraw = new PointF[samples * 2];
        static PointF[] Zpointtodraw = new PointF[samples*2];

        public Form3()
        {
            InitializeComponent();
            DataPlot += new ReceiveSerialDatatoBePlot(UpdateDataforGraphic);
        }

        private void Form3_Load(object sender, EventArgs e)
        {
            hide = false;
        }
        #region property
        public float Y_MAKS
        {
            set { newYmaks = value; }
        }
        public float X_MAKS
        {
            set { newXmaks = value; }
        }
        public int Y_INCREMENT
        {
            set { newYincrement = value; }
        }
        public int X_INCREMENT
        {
            set { newXincrement = value; }
        }
```

```
public int SAMPLES
{
    set { samples = value; }
}
public string GRID
{
    set { newgrid = value; }
}
public bool HIDE
{
    set { hide = value; }
}
public bool XSELECT
{
    set { xselect = value; }
}
public bool YSELECT
{
    set { yselect = value; }
}
public bool ZSELECT
{
    set { zselect = value; }
}
public int[] RECEIVEXDATA
{
    get { return xgraphicdata; }
    set
    {
        xgraphicdata = value;
        if (DataPlot != null)
        {
            PlotEventArgs PARGS = new PlotEventArgs(RECEIVEXDATA, null, null);
            DataPlot(this, PARGS);
            this.Refresh();
        }
    }
}
public int[] RECEIVEYDATA
{
    get { return ygraphicdata; }
    set
    {
        ygraphicdata = value;
        if (DataPlot != null)
        {
            PlotEventArgs PARGS = new PlotEventArgs(null, RECEIVEYDATA, null);
            DataPlot(this, PARGS);
            this.Refresh();
        }
    }
}
public int[] ZDATA
{
    set { zgraphicdata = value; }
}
public bool CONNECTED
{
    set { connected = value; }
}

#endregion
```

```
#region function
private float XScalingGraphic()
{
    float XPoint;
    XPoint=(float)700/samples;
    return (float)XPoint; ;
}
private float YScalingGraphic(int data)
{
    float YPoint;
    YPoint=(float)((float)data/Ymaks)*150;
    return (float)YPoint;
}
#region Drawing X Datalines
public void XReceiveSerialDataforGraphic(int[] acceldata)
{

    for (int i = 0; i < samples / 2; i++)
    {
        Xpointtodraw[i] = new PointF(XScalingGraphic() * 2 * i + 50,
                          (float)YORIGIN - YScalingGraphic(acceldata[i]));
    }
}
private void DrawDatalinesX(Graphics gr)
{

    GraphicsPath gpline = new GraphicsPath();
    Pen PD = new Pen(Color.Blue, 2);
    Pen pengraph = new Pen(Color.DarkGreen);
    int pointlength = Ypointtodraw.Length;
    Rectangle RECT = new Rectangle(50, 50, 700, 300);
    gr.IntersectClip(RECT);
    for (int i = 0; i < samples / 2 - 1; i++)
    {
        gr.DrawLine(pengraph, Xpointtodraw[i], Xpointtodraw[i + 1]);
    }
    pengraph.Dispose();
    PD.Dispose();
    gr.Dispose();
}
#endregion
#region Drawing Y Datalines
public void YReceiveSerialDataforGraphic(int[] acceldata)
{
    for (int i = 0; i < samples / 2; i++)
    {
        Ypointtodraw[i] = new PointF(XScalingGraphic() * 2 * i + 50,
                          (float)YORIGIN - YScalingGraphic(acceldata[i]));
    }
}
private void DrawDatalinesY(Graphics gr)
{

    GraphicsPath gpline = new GraphicsPath();
    Pen PD = new Pen(Color.Blue, 2);
    Pen pengraph = new Pen(Color.Red);
    int pointlength = Ypointtodraw.Length;
    Rectangle RECT = new Rectangle(50, 50, 700, 300);
    gr.IntersectClip(RECT);
    for (int i = 0; i < samples / 2 - 1; i++)
    {
        gr.DrawLine(pengraph, Ypointtodraw[i], Ypointtodraw[i + 1]);
    }
```

```
    pengraph.Dispose();
    PD.Dispose();
    gr.Dispose();
}
#endregion
#region Drawing Z Datalines
public void ZReceiveSerialDataforGraphic(int[] acceldata)
{
    for (int i = 0; i < samples/2; i++)
    {
        Zpointtodraw[i] = new PointF(XScalingGraphic() * 2*i + 50,
                          (float)YORIGIN - YScalingGraphic(acceldata[i])  );
    }
}
private void DrawDatalinesZ(Graphics gr)
{

    GraphicsPath gpline = new GraphicsPath();
    Pen PD = new Pen(Color.Blue, 2);
    Pen pengraph = new Pen(Color.Blue);
    int pointlength = Zpointtodraw.Length;
    Rectangle RECT = new Rectangle(50, 50, 700, 300);
    gr.IntersectClip(RECT);
    for (int i = 0; i < samples/2-1; i++)
    {
        gr.DrawLine(pengraph, Zpointtodraw[i], Zpointtodraw[i + 1]);
    }
    pengraph.Dispose();
    PD.Dispose();
    gr.Dispose();
}
#endregion
public void UpdateDataforGraphic(object sender, PlotEventArgs e)
{
}
#endregion
#region retrieving parameter from delegates
public void ReceiveDataX(int[] datax)
{
    XReceiveSerialDataforGraphic(datax);
    this.Refresh();
}
public void ReceiveDataY(int[] datay)
{
    YReceiveSerialDataforGraphic(datay);
}
public void ReceiveDataZ(int[] dataz)
{
    ZReceiveSerialDataforGraphic(dataz);
}

#endregion
#region Painting Graphic
private void Form3_Paint(object sender, PaintEventArgs e)
{
    float XaxisScale = Xmaks / Xincrement;
    float YaxisScale = Ymaks / Yincrement;
    Bitmap BMP1 = new Bitmap(this.Width,this.Height);
    Bitmap BMP2 = new Bitmap(this.Width, this.Height);
    Bitmap BMP3 = new Bitmap(this.Width, this.Height);
    Bitmap BMP4 = new Bitmap(this.Width, this.Height);
    Graphics g1 = e.Graphics;
    Graphics g2 = this.CreateGraphics();
```

```
Graphics G3 = Graphics.FromImage(BMP1);
Graphics G4 = Graphics.FromImage(BMP2);
Graphics G5 = Graphics.FromImage(BMP3);
Graphics G6 = Graphics.FromImage(BMP4);
Graphics UPLOADTOSCREEN = this.CreateGraphics();
#region initialize pen & drawing properties
Pen p1 = new Pen(Color.Black, 2);
Pen p2 = new Pen(Color.Orange);
Pen p3=new Pen(Color.Black);
Pen pdataX = new Pen(Color.DarkBlue);
Pen pdataY = new Pen(Color.Red);
Pen pdataZ = new Pen(Color.DarkGreen);
//Pen pengraph = new Pen(Color.Blue,3);
GraphicsPath gp1 = new GraphicsPath();
StringFormat sf=new StringFormat();
StringFormat sf2 = new StringFormat(StringFormatFlags.DirectionVertical);
sf.Alignment = StringAlignment.Far;
Font f = new Font("Arial Rounded MT Black", 12,FontStyle.Bold);
Font f2 = new Font("Arial Rounded MT Black", 11, FontStyle.Bold);
Font f3 = new Font("Arial Rounded MT Black", 8, FontStyle.Bold);
p2.DashStyle = DashStyle.Dot;
#endregion
#region With Double Buffer
    #region creating canvas
    G3.FillRectangle(new SolidBrush(clr[0]), this.ClientRectangle);
    G4.Clear(Color.Transparent);
    G5.Clear(Color.Transparent);
    G6.Clear(Color.Transparent);
    G3.Clear(Color.White);

    #endregion

    #region drawing data graphic border and axes
    Point[] border1 = new Point[4]
    { new Point((int)(Xborder - XhalfLength), (int)(Yborder -
      YhalfLength)),
    new Point((int)(Xborder+XhalfLength),(int)(Yborder-YhalfLength)),
    new Point((int)(Xborder+XhalfLength),(int)(Yborder+YhalfLength)),
    new Point((int)(Xborder-XhalfLength),(int)(Yborder+YhalfLength))};
    gp1.AddLines(border1);
    gp1.CloseFigure();
    PointF[] toborder = new PointF[2] { new PointF(Xborder - XhalfLength -
                        10, Yborder - YhalfLength - 5), new PointF(Xborder
                        + XhalfLength + 10, Yborder - YhalfLength - 5) };
    PointF[] leftborder = new PointF[4] { new PointF(Xborder - XhalfLength
                          - 10, Yborder - YhalfLength), new PointF(Xborder
                          - XhalfLength - 5, Yborder - YhalfLength), new
                          PointF(Xborder - XhalfLength - 5, Yborder +
                          YhalfLength), new PointF(Xborder - XhalfLength -
                          10, Yborder + YhalfLength) };
    PointF[] bottomborder = new PointF[2] { new PointF(Xborder -
                            XhalfLength - 10, Yborder + YhalfLength + 5),
                            new PointF(Xborder + XhalfLength + 10, Yborder
                            + YhalfLength + 5) };
    PointF[] rightborder = new PointF[4] { new PointF(Xborder +
                           XhalfLength + 10, Yborder - YhalfLength), new
                           PointF(Xborder + XhalfLength + 5, Yborder -
                           YhalfLength), new PointF(Xborder + XhalfLength
                           + 5, Yborder + YhalfLength), new PointF(Xborder
                           + XhalfLength + 10, Yborder + YhalfLength) };

    G3.DrawLines(p1, toborder);
    G3.DrawLines(p1, leftborder);
```

```
G3.DrawLines(p1, rightborder);
G3.DrawLines(p1, bottomborder);
G3.DrawLine(p1, Xborder - XhalfLength, Yborder, Xborder + XhalfLength,
           Yborder);
#endregion

#region drawing legend & Datalines
Point[] legend1 = new Point[4]
{ new Point((int)(xlegend-xhalflegend), (int)(ylegnd - yhalflegend))
,new Point((int)(xlegend+xhalflegend),(int)(ylegnd-yhalflegend))
,new Point((int)(xlegend+xhalflegend),(int)(ylegnd+yhalflegend))
,new Point((int)(xlegend-xhalflegend),(int)(ylegnd+yhalflegend))
};
gp1.AddLines(legend1);
gp1.CloseFigure();
G3.DrawString("Legend", f, Brushes.Black, 781f, 50f); ;

G3.DrawString("DataX", f2, Brushes.DarkGreen, 800f, 70f);
G3.FillRectangle(Brushes.DarkGreen, 785, 75, 10, 10);
G3.DrawString("DataY", f2, Brushes.Red, 800f, 85f);
G3.FillRectangle(Brushes.Red, 785, 90, 10, 10);
G3.DrawString("DataZ", f2, Brushes.DarkBlue, 800f, 100f);
G3.FillRectangle(Brushes.DarkBlue, 785, 105, 10, 10);
if (xselect)
{
    G3.FillRectangle(Brushes.DarkGreen, 855, 78.5f, 55, 3);
    DrawDatalinesX(G4);
}
if (yselect)
{
    G3.FillRectangle(Brushes.Red, 855, 93.5f, 55, 3);
    DrawDatalinesY(G5);
}
if (zselect)
{
    G3.FillRectangle(Brushes.DarkBlue, 855, 108.5f, 55, 3);
    DrawDatalinesZ(G6);
}
G3.DrawPath(p2, gp1);
#endregion

#region drawing grid & scale
for (int i = -Yincrement; i <= Yincrement; i++)
{
    if (gridon == "on")
    {
        G3.DrawLine(p2, Xborder - XhalfLength, (Yborder +
                    ((YhalfLength / Yincrement) * i)), Xborder +
                    XhalfLength, (Yborder + ((YhalfLength /
                    Yincrement) * i)));
    }
    G3.DrawLine(p3, Xborder - XhalfLength - 9, (Yborder +
               ((YhalfLength / Yincrement) * i)), Xborder -
               XhalfLength - 5, (Yborder + ((YhalfLength / Yincrement)
               * i)));
    G3.DrawLine(p3, Xborder + XhalfLength + 9, (Yborder +
               ((YhalfLength / Yincrement) * i)), Xborder +
               XhalfLength + 5, (Yborder + ((YhalfLength / Yincrement)
               * i)));
    G3.DrawString((-YaxisScale * i).ToString(), f3, Brushes.Black,
                  Xborder - XhalfLength - 11, (Yborder - 6 +
                  ((YhalfLength / Yincrement) * i)), sf);
}
```

```
            for (int i = 0; i <= Xincrement; i++)
            {
                if (gridon == "on")
                {
                    G3.DrawLine(p2, (Xborder - XhalfLength + ((2 * XhalfLength /
                                Xincrement) * i)), (Yborder - YhalfLength),
                                (Xborder - XhalfLength + ((2 * XhalfLength /
                                Xincrement) * i)), Yborder + YhalfLength);
                }
                G3.DrawLine(p3, (Xborder - XhalfLength + ((2 * XhalfLength /
                            Xincrement) * i)), Yborder - YhalfLength - 9, (Xborder
                            - XhalfLength + ((2 * XhalfLength / Xincrement) * i)),
                            Yborder - YhalfLength - 5);
                G3.DrawLine(p3, (Xborder - XhalfLength + ((2 * XhalfLength /
                            Xincrement) * i)), Yborder + YhalfLength + 9, (Xborder
                            - XhalfLength + ((2 * XhalfLength / Xincrement) * i)),
                            Yborder + YhalfLength + 5);
                G3.DrawString((XaxisScale * i).ToString(), f3, Brushes.Black,
                              (Xborder - XhalfLength + ((2 * XhalfLength /
                              Xincrement) * i)) - 8, Yborder + YhalfLength + 9,
                              sf2);
            }

            #endregion
            UPLOADTOSCREEN.DrawImage(BMP1, 0, 0);
            UPLOADTOSCREEN.DrawImage(BMP2, 0, 0);
            UPLOADTOSCREEN.DrawImage(BMP3, 0, 0);
            UPLOADTOSCREEN.DrawImage(BMP4, 0, 0);
        #endregion
        G3.Dispose();
        G4.Dispose();
        BMP1.Dispose();
        BMP2.Dispose();
        UPLOADTOSCREEN.Dispose();
        gp1.Dispose();
        p1.Dispose();
        g1.Dispose();
        g2.Dispose();
    }
    #endregion
    #region form closing event
    private void Form3_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (e.CloseReason == CloseReason.UserClosing)
        {
            e.Cancel = true;
            this.Hide();
            hide = true;
        }
    }
    #endregion
    #region another event
    private void Form3_Activated(object sender, EventArgs e)
    {
        Ymaks = newYmaks;
        Xmaks = newXmaks;
        Yincrement = newYincrement;
        Xincrement = newXincrement;
        gridon = newgrid;
        //textBox1.Text = newgrid;
    }
```

```
        private void Form3_Click(object sender, EventArgs e)
        {
            Xmaks = newXmaks;
            Yincrement = newYincrement;
            Xincrement = newXincrement;
            gridon = newgrid;
        }

        private void Form3_MouseMove(object sender, MouseEventArgs e)
        {
            label1.Text= "X :"+ e.X.ToString();
            label2.Text = "Y :" + e.Y.ToString();
        }
        #endregion

        private void button1_Click(object sender, EventArgs e)
        {
            timer1.Enabled = false;
            this.Refresh();
        }



        public class PlotEventArgs : System.EventArgs
        {
            private int[] xdatatemp = new int[2048];
            private int[] ydatatemp = new int[2048];
            private int[] zdatatemp = new int[2048];
            public PlotEventArgs(int[] XPlot,int[] YPlot,int[]ZPlot)
            {
                this.xdatatemp = XPlot;
                this.ydatatemp = YPlot;
                this.zdatatemp = ZPlot;
            }
            public int[] RECEIVEDATA
            {
                get { return xdatatemp; }
            }
            public int[] YRECEIVEDATA
            {
                get { return ydatatemp; }
            }
            public int[] ZRECEIVEDATA
            {
                get { return zdatatemp; }
            }
        }
    }

public partial class Form4 : Form
    {
        public delegate void DisplayTextonChild3(string message);
        public event StopSerialFromChild stopfromchild3;
        public Form4()
        {
            InitializeComponent();
        }
        private void Form4_Load(object sender, EventArgs e)
        {

        }
```

```
        #region function
        public void publishtotextbox(string message,string message2,string message3)
        {
            richTextBox1.Invoke(new EventHandler(delegate
            {

                richTextBox1.AppendText(message+"\t"+message2+"\t"+message3+Environmen
                t.NewLine);
                richTextBox1.SelectionColor = Color.Blue;
                richTextBox1.ScrollToCaret();
            }
            ));
        }
        #endregion
        #region formclosing event
        private void Form4_FormClosing(object sender, FormClosingEventArgs e)
        {
            Form1 parent = new Form1();

            if (e.CloseReason == CloseReason.UserClosing)
            {
                e.Cancel = true;
                this.Hide();
            }
        }
        #endregion
    }

public delegate void publishtotextboxform5(string mess);
    public partial class Form5 : Form
    {
        public delegate void SendToWSS(object sender, SerialEventArgs e);
        public event SendToWSS sendtoWSS;
        public event AccelChannelChoice AccelChannel;
        byte[] bytetosend = new byte[28]
                                    {0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,
                                    0x0A,0x0B,0x0C,0x0D,0x0E,0x0F,0x10,0x11,0x12,0
                                    x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1A,0x1B,
                                    0x1C};
        byte[] testsend = new byte[2] { 0x08, 0x1C };
        public Form5()
        {
            InitializeComponent();
        }

        private void Form5_Load(object sender, EventArgs e)
        {
            radioButton1.Checked = true;
            radioButton2.Checked = false;
            radioButton3.Checked = false;
            radioButton4.Checked = false;
            checkBox4.Checked = true;
        }

        private void Form5_FormClosing(object sender, FormClosingEventArgs e)
        {
            Form1 parent = new Form1();
            if (e.CloseReason == CloseReason.UserClosing)
            {
                e.Cancel = true;
                Thread.Sleep(250);
                parent.VCP.Close();
                this.Hide();
```

```
    }
}

private void button2_Click(object sender, EventArgs e)
{
    radioButton1.Checked = true;
    radioButton2.Checked = false;
    radioButton3.Checked = false;
    radioButton4.Checked = false;
    checkBox4.Checked = true;
    checkBox3.Checked = false;
    checkBox2.Checked = false;
    checkBox1.Checked = false;
}

public void publishtotextbox(string message)
{
    textBox1.Invoke(new EventHandler(delegate
    {
        textBox1.AppendText(message + Environment.NewLine);
    }
    ));
}

private void button1_Click(object sender, EventArgs e)
{
    Form1 parent = new Form1();
    Form3 FormGraph = new Form3();
    byte senddata = 0x01;
    int channeldigitrepresentation = 1;
    //byte senddata = Convert.ToByte(textBox1.Text);
    #region statements
    if (radioButton1.Checked && checkBox2.Checked)
    { senddata = 0x01; channeldigitrepresentation = 1; FormGraph.XSELECT =
      true; FormGraph.YSELECT = false; FormGraph.ZSELECT = false; }   //0x01
    if (radioButton1.Checked && checkBox3.Checked)
    { senddata = 0x02; channeldigitrepresentation = 2; FormGraph.XSELECT =
      false; FormGraph.YSELECT = true; FormGraph.ZSELECT = false; }
    if (radioButton1.Checked && checkBox4.Checked)
    { senddata = 0x03; channeldigitrepresentation = 3; FormGraph.XSELECT =
      false; FormGraph.YSELECT = false; FormGraph.ZSELECT = true; }
    if (radioButton1.Checked && checkBox2.Checked && checkBox3.Checked)
    { senddata = 0x04; channeldigitrepresentation = 4; FormGraph.XSELECT =
      true; FormGraph.YSELECT = true; FormGraph.ZSELECT = false; }
    if (radioButton1.Checked && checkBox3.Checked && checkBox4.Checked)
    { senddata = 0x05; channeldigitrepresentation = 5; FormGraph.XSELECT =
      false; FormGraph.YSELECT = true; FormGraph.ZSELECT = true; }
    if (radioButton1.Checked && checkBox2.Checked && checkBox4.Checked)
    { senddata = 0x06; channeldigitrepresentation = 6; FormGraph.XSELECT =
      true; FormGraph.YSELECT = false; FormGraph.ZSELECT = true; }
    if (radioButton1.Checked && checkBox2.Checked && checkBox3.Checked &&
        checkBox4.Checked)
    { senddata = 0x07; channeldigitrepresentation = 7; FormGraph.XSELECT =
      true; FormGraph.YSELECT = true; FormGraph.ZSELECT = true; }

    //////////////4-G///////
    if (radioButton2.Checked && checkBox2.Checked)
    { senddata = 0x08; channeldigitrepresentation = 1; FormGraph.XSELECT =
      true; FormGraph.YSELECT = false; FormGraph.ZSELECT = false; }   //0x01
    if (radioButton2.Checked && checkBox3.Checked)
    { senddata = 0x09; channeldigitrepresentation = 2; FormGraph.XSELECT =
      false; FormGraph.YSELECT = true; FormGraph.ZSELECT = false; }
    if (radioButton2.Checked && checkBox4.Checked)
```

```
{ senddata = 0x0A; channeldigitrepresentation = 3; FormGraph.XSELECT =
  false; FormGraph.YSELECT = false; FormGraph.ZSELECT = true; }
if (radioButton2.Checked && checkBox2.Checked && checkBox3.Checked)
{ senddata = 0x0B; channeldigitrepresentation = 4; FormGraph.XSELECT =
  true; FormGraph.YSELECT = true; FormGraph.ZSELECT = false; }
if (radioButton2.Checked && checkBox3.Checked && checkBox4.Checked)
{ senddata = 0x0C; channeldigitrepresentation = 5; FormGraph.XSELECT =
  false; FormGraph.YSELECT = true; FormGraph.ZSELECT = true; }
if (radioButton2.Checked && checkBox2.Checked && checkBox4.Checked)
{ senddata = 0x0D; channeldigitrepresentation = 6; FormGraph.XSELECT =
  true; FormGraph.YSELECT = false; FormGraph.ZSELECT = true; }
if (radioButton2.Checked && checkBox2.Checked && checkBox3.Checked &&
    checkBox4.Checked)
{ senddata = 0x0E; channeldigitrepresentation = 7; FormGraph.XSELECT =
  true; FormGraph.YSELECT = true; FormGraph.ZSELECT = true; }

//////////////8-G///////
if (radioButton3.Checked && checkBox2.Checked)
{ senddata = 0x0F; channeldigitrepresentation = 1; FormGraph.XSELECT =
  true; FormGraph.YSELECT = false; FormGraph.ZSELECT = false; }   //0x01
if (radioButton3.Checked && checkBox3.Checked)
{ senddata = 0x10; channeldigitrepresentation = 2; FormGraph.XSELECT =
  false; FormGraph.YSELECT = true; FormGraph.ZSELECT = false; }
if (radioButton3.Checked && checkBox4.Checked)
{ senddata = 0x11; channeldigitrepresentation = 3; FormGraph.XSELECT =
  false; FormGraph.YSELECT = false; FormGraph.ZSELECT = true; }
if (radioButton3.Checked && checkBox2.Checked && checkBox3.Checked)
{ senddata = 0x12; channeldigitrepresentation = 4; FormGraph.XSELECT =
  true; FormGraph.YSELECT = true; FormGraph.ZSELECT = false; }
if (radioButton3.Checked && checkBox3.Checked && checkBox4.Checked)
{ senddata = 0x13; channeldigitrepresentation = 5; FormGraph.XSELECT =
  false; FormGraph.YSELECT = true; FormGraph.ZSELECT = true; }
if (radioButton3.Checked && checkBox2.Checked && checkBox4.Checked)
{ senddata = 0x14; channeldigitrepresentation = 6; FormGraph.XSELECT =
  true; FormGraph.YSELECT = false; FormGraph.ZSELECT = true; }
if (radioButton3.Checked && checkBox2.Checked && checkBox3.Checked &&
   checkBox4.Checked)
{ senddata = 0x15; channeldigitrepresentation = 7; FormGraph.XSELECT =
  true; FormGraph.YSELECT = true; FormGraph.ZSELECT = true; }

//////////////16-G///////
if (radioButton4.Checked && checkBox2.Checked)
{ senddata = 0x16; channeldigitrepresentation = 1; FormGraph.XSELECT =
  true; FormGraph.YSELECT = false; FormGraph.ZSELECT = false; }   //0x01
if (radioButton4.Checked && checkBox3.Checked)
{ senddata = 0x17; channeldigitrepresentation = 2; FormGraph.XSELECT =
  false; FormGraph.YSELECT = true; FormGraph.ZSELECT = false; }
if (radioButton4.Checked && checkBox4.Checked)
{ senddata = 0x18; channeldigitrepresentation = 3; FormGraph.XSELECT =
  false; FormGraph.YSELECT = false; FormGraph.ZSELECT = true; }
if (radioButton4.Checked && checkBox2.Checked && checkBox3.Checked)
{ senddata = 0x19; channeldigitrepresentation = 4; FormGraph.XSELECT =
  true; FormGraph.YSELECT = true; FormGraph.ZSELECT = false; }
if (radioButton4.Checked && checkBox3.Checked && checkBox4.Checked)
{ senddata = 0x1A; channeldigitrepresentation = 5; FormGraph.XSELECT =
  false; FormGraph.YSELECT = true; FormGraph.ZSELECT = true; }
if (radioButton4.Checked && checkBox2.Checked && checkBox4.Checked)
{ senddata = 0x1B; channeldigitrepresentation = 6; FormGraph.XSELECT =
  true; FormGraph.YSELECT = false; FormGraph.ZSELECT = true; }
if (radioButton4.Checked && checkBox2.Checked && checkBox3.Checked &&
    checkBox4.Checked)
{ senddata = 0x1B; channeldigitrepresentation = 7; FormGraph.XSELECT =
  true; FormGraph.YSELECT = true; FormGraph.ZSELECT = true; }
```

```
            else
            {

            }
            #endregion
            if (checkBox1.Checked)
            {
                channeldigitrepresentation = 8;
            }
            #endregion
            SerialEventArgs SArgs = new SerialEventArgs(senddata);
            this.sendtoWSS(this, SArgs);
            ChoiceEventArgs CArgs = new ChoiceEventArgs(channeldigitrepresentation);
            this.AccelChannel(this, CArgs);
        }
    }
```