Câu chuyện bắt đầu từ một cậu bé,
và một ý tưởng
có thể
làm thay đổi thế giới...

PAY IT FORWARD

Đó là khi bạn giúp đỡ 3 người bạn không quen biết,
dù là bằng thời gian,
hay công sức,
hay kinh nghiệm,
hay kiến thức,
hay tiền bạc, ...
của mình.

Mà không chờ đợi một sự báo ân nào.

Chỉ cần mỗi người trong 3 người đó,
lại đem những gì mình có, mà người khác cần,
tiếp tục giúp đỡ thêm 3 người nữa.

Chính những người-giúp-đỡ, và người-được-giúp-đỡ,
sẽ là những người góp phần thay đổi thế giới...

Một thế giới sẻ chia kiến thức - và yêu thương ...

PAY IT FORWARD ...

Chúng tôi không sáng tạo ra câu nói này.

Pay it forward...

Hãy tri ân người giúp mình bằng cách giúp đỡ người khác
Cho đi không phải để nhận lại.

payitforward.edu.vn

HCM City - University of Technology
Faculty of Electrical and Electronics Engineering
The Science-Study Club - *PayItForward*

Graphic User Interface Programming

Implemented on:

Microsoft Visual Studio C#

DECEMBER 9TH 2012

By: TME and Bros

Contents

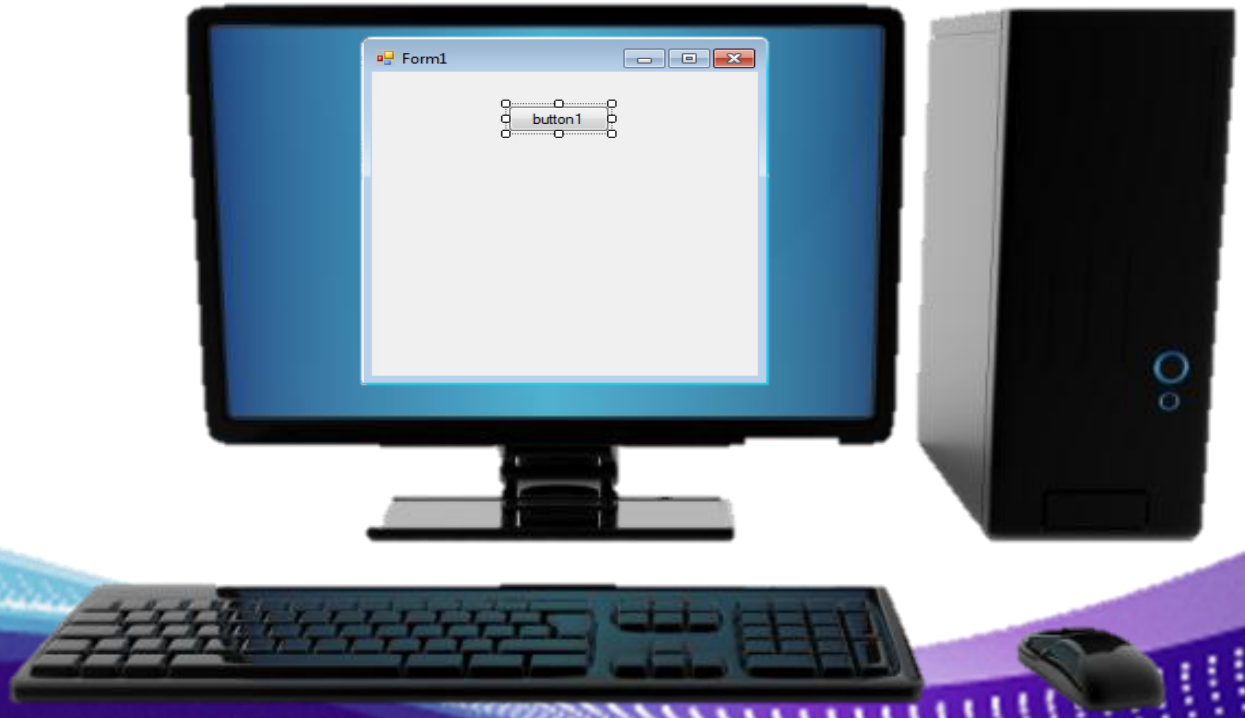
- Interface and GUI definition.
- Object-oriented Programming.
- Visual Studio C# - The Basics
  - Form, Buttons, TextBoxes.
  - Timer, Serial Ports, delegates, cross-thread.
  - Graphing:
  - Packaging:

Graphic User Interface?

Interface is the tool that helps interact, configure and supervise a system.

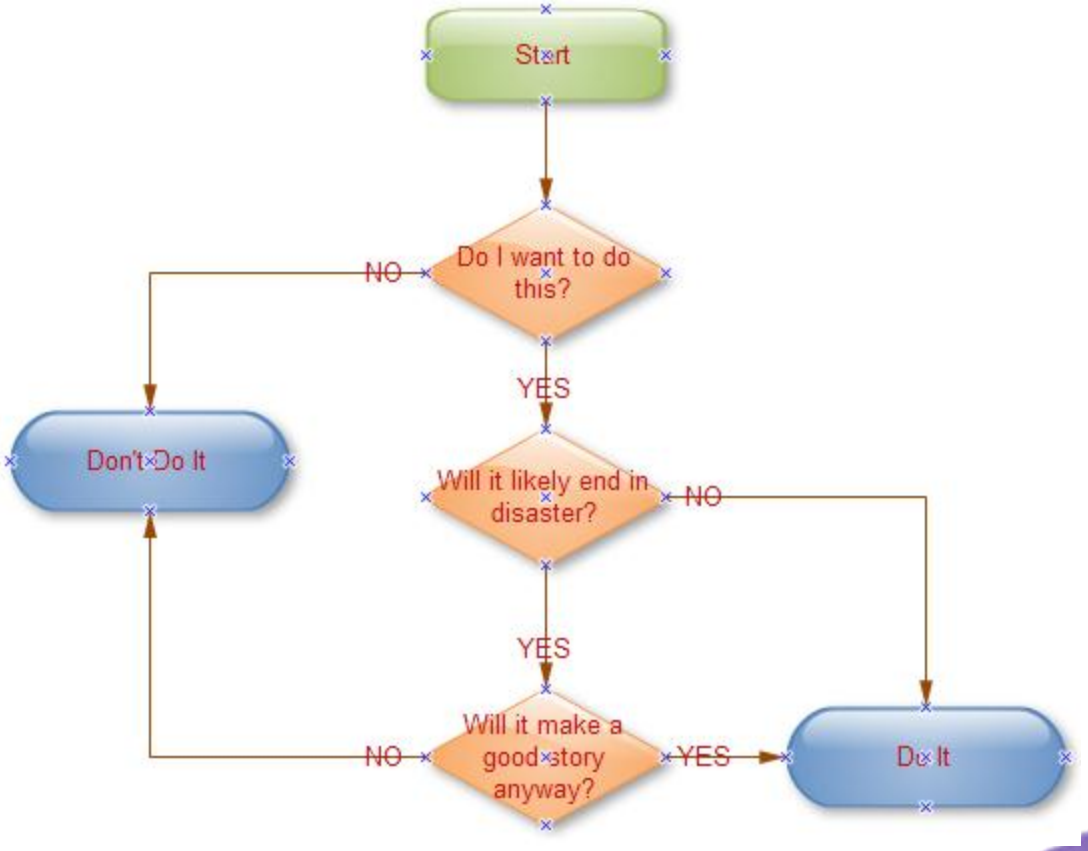
Graphic User Interface:

An interface that helps interact with electronic devices using images on screens.



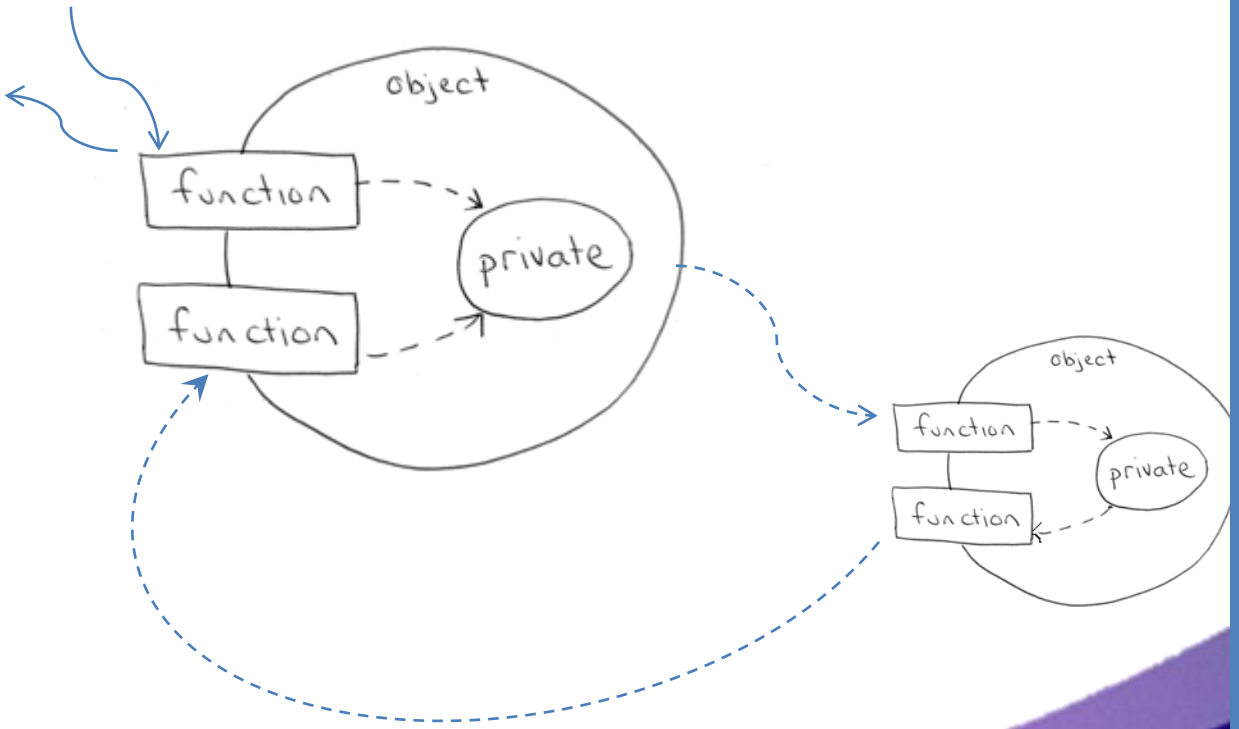
Object-oriented Programming?

In Structured Programming, program is a list of subroutines processed one after another, selected by conditions. User's interactions require latency. Generally speaking, everything's written in this way.



Object-oriented Programming:

- **OOP defines *objects* as instances of *classes*. User interacts with objects by their associated functions termed as *methods*. Methods manipulate the values of objects' *properties* and return the result.**



Object-oriented Programming:

BK Student
public string Name
public short Year
private double GPA
private boolean Single
…
string WriteAGUI();
string SingASong ();

A class definition

PIF's BK Student		
string Name: "Britsk"		
short Year: 2009		
double GPA: 8.xx		
boolean Single: true		
boolean KnowMSP: true		
string WriteAGUI();	→	**"Good Job!"**
string SingASong();	→	**"Hello!!"**

An object

Visual Studio C# - The Basics: Starting A Form

Microsoft Visual Studio (Administrator)

New Project

Recent Templates

Installed Templates

- Visual C#
  - Windows
  - Web
  - Office
  - Cloud
  - Reporting
  - SharePoint
  - Silverlight
  - Test
  - WCF
  - Workflow
- Other Languages
- Other Project Types
- Database
- Modeling Projects
- Test Projects

Online Templates

.NET Framework 4 | Sort by: Default | Search Installed Templates

Windows Forms Application	Visual C#
WPF Appli	Visual C#
Console Application	Visual C#
Class Library	Visual C#
WPF Browser Application	Visual C#
Empty Project	Visual C#
Windows Service	Visual C#
WPF Custom Control Library	Visual C#
WPF User Control Library	Visual C#

Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name: WindowsFormsApplication1

Location: E:\Learning Projects\Visual Studio 2010\Projects\EPPVB

Solution name: WindowsFormsApplication1

Browse...

☑ Create directory for solution

☐ Add to source control

OK

Visual Studio C# - The Basics: Simple *Controls*

Toolbox

- All Windows Forms
- Common Controls
  - Pointer
  - Button
  - CheckBox
  - CheckedListBox
  - ComboBox
  - DateTimePicker
  - Label
  - LinkLabel
  - ListBox
  - ListView
  - MaskedTextBox
  - MonthCalendar
  - NotifyIcon
  - NumericUpDown
  - PictureBox
  - ProgressBar
  - RadioButton
  - RichTextBox
  - TextBox
  - ToolTip
  - TreeView
  - WebBrowser
- Containers
  - Pointer
  - FlowLayoutPanel
  - GroupBox

Double-click

Yr code here!

ntrol Settings

ntrol

Sele Mode

ConfigurationForm.cs | ConfigurationForm.cs [Design]

motor_ctrl.ConfigurationForm

```
private void butt_track_mode_Click(object sender, EventArgs e)
{
    GroupBoxTrackControl.Enabled = true;
    GroupBoxTrackControl.Visible = true;
    reset_track_mode();
    Com.Write("t\0"); //"t" in "tracking"

    Timer.Interval = 100;
    Timer.Enabled=true;

    GroupBoxSelectPort.Visible = false;
    GroupBoxSelectPort.Enabled = false;
    GroupBoxOptions.Visible = false;
    GroupBoxOptions.Enabled = false;
    textbox_temp_text = butt_track_intro;

    //this is neccessarily put here
    ButtonGraphing.Enabled = true;
    if (GraphingForm != null)
    {
        GraphingForm.Close();
        GraphingForm = new GraphForm(this);
        GraphingForm.Enabled = true;
        GraphingForm.Show();
    }
}
```

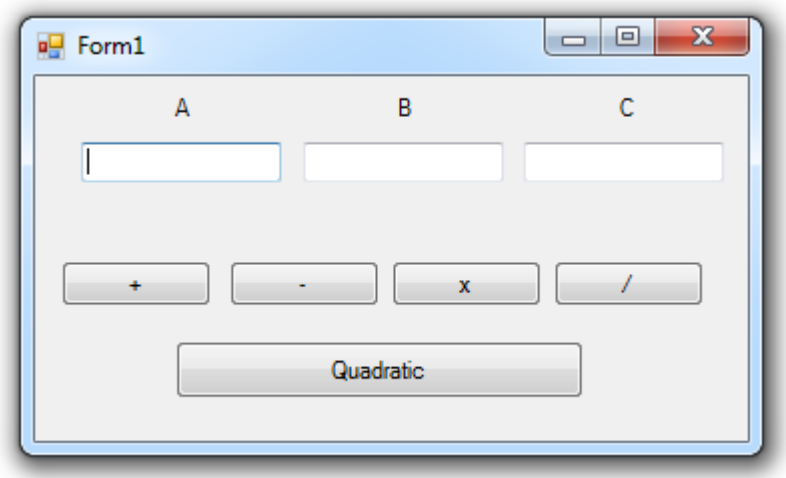
100 %

Visual Studio C# - The Basics: Simple *Controls*

- Problem:
  - Make GUI with **3 textboxes**: txtboxNumA, txtboxNumB, txtboxResult; **5 buttons**: buttonSum, buttonSub, buttonMul, buttonDiv, buttonQuad.
  - Input the numbers to txtboxNumA and txtboxNumB. Click any of the first 4 button to have the respective **result displayed in txtboxResult**. Click buttonQuad to get the **solutions of a quadratic equation** with a, b, c in the three txtboxes, displayed the **solutions in string form on txtboxResult**.

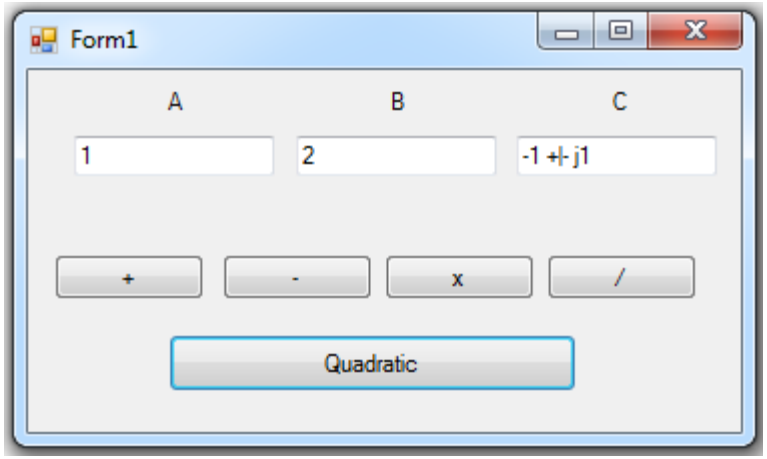
Visual Studio C# - The Basics: Simple *Controls*

```
private void buttonDiv_Click(object sender, EventArgs e)
        {
            double a, b;
            if (txtboxNumA.Text == "" || txtboxNumB.Text == "")
                MessageBox.Show("Missing Input!");
            else
            {
                a = Convert.ToDouble(txtboxNumA.Text);
                b = Convert.ToDouble(txtboxNumB.Text);
                txtboxResult.Text = Convert.ToString(a / b);
            }
        }
```



In this function we have a sample of event response, data check, type conversion and notification.

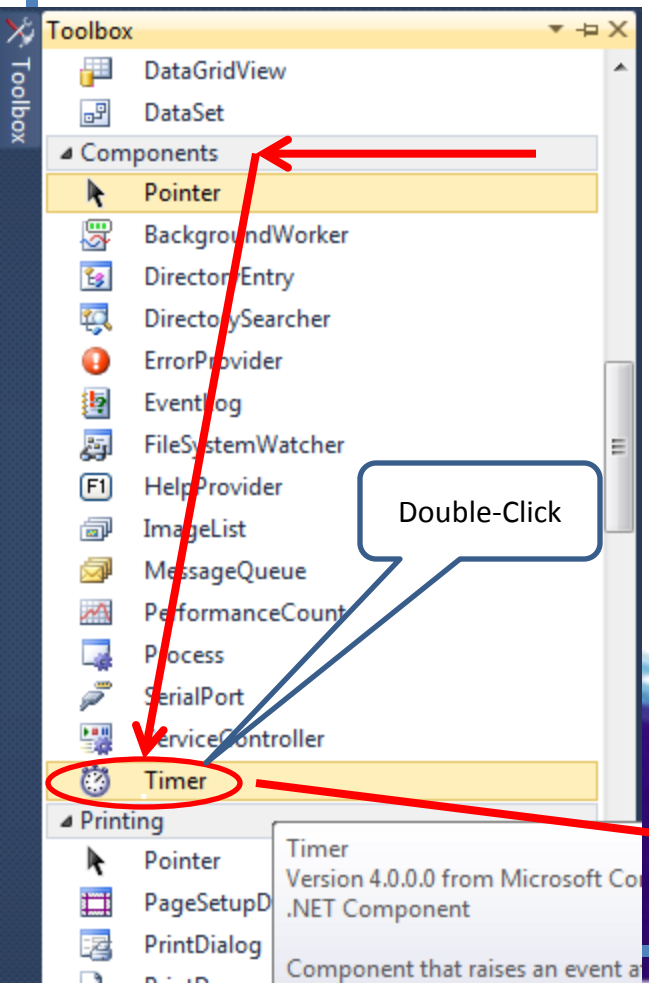
```
private void button1_Click(object sender, EventArgs e)
{
    double a, b, c, delta_sqrt, x1, x2;
    if (txtboxNumA.Text == "" || txtboxNumB.Text == "" || txtboxResult.Text == "")
        MessageBox.Show("Missing Input!");


    else
    {
        a = Convert.ToDouble(txtboxNumA.Text);
        b = Convert.ToDouble(txtboxNumB.Text);
        c = Convert.ToDouble(txtboxResult.Text);
        if ((b * b - 4 * a * c) >= 0)
        {
            delta_sqrt = Math.Sqrt((b * b - 4 * a * c));
            x1 = (-b + delta_sqrt) / 2 / a;
            x2 = (-b - delta_sqrt) / 2 / a;
            txtboxResult.Text = Convert.ToString(x1) + " ; " + Convert.ToString(x2);
        }
        else
        {
            delta_sqrt = Math.Sqrt(-(b * b - 4 * a * c));
            txtboxResult.Text = Convert.ToString(-b/2/a) + " +|- j" + Convert.ToString(delta_sqrt/2/a);
        }
    }
```

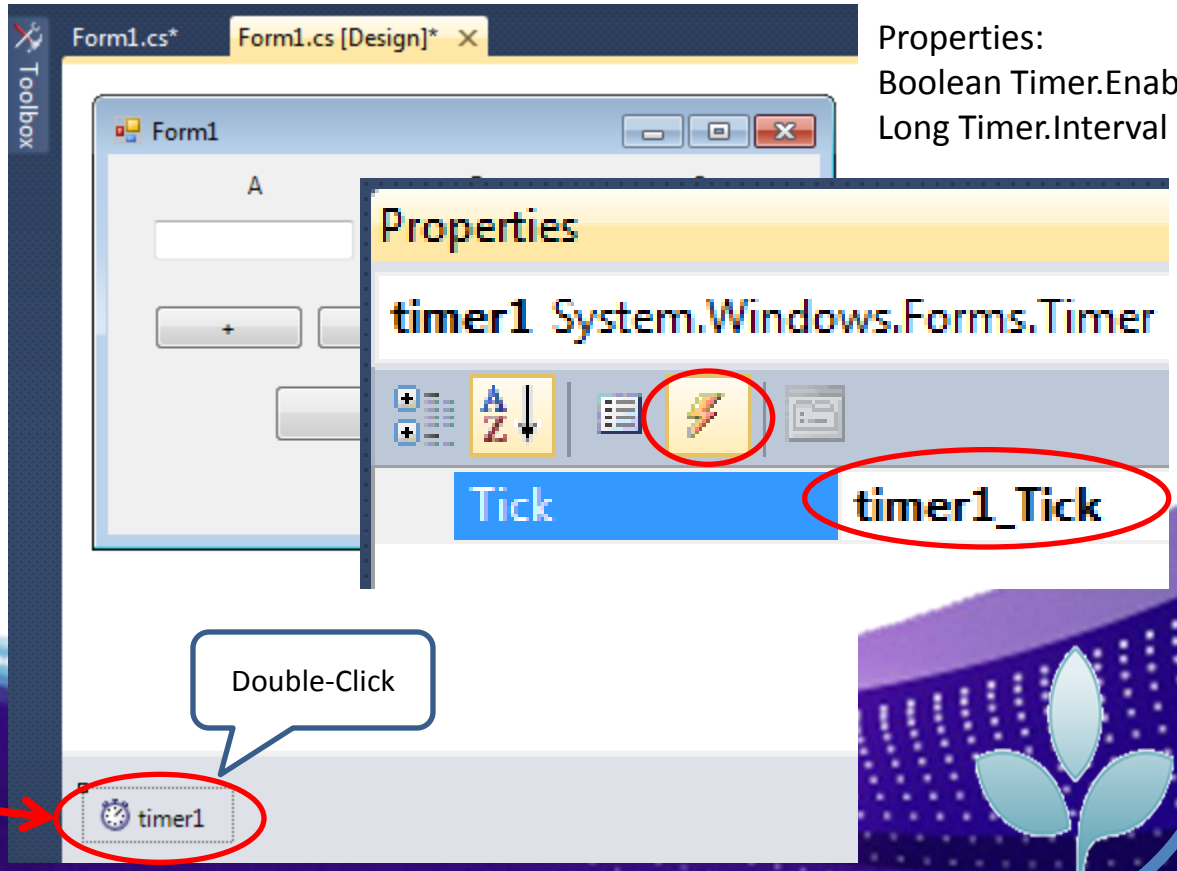


Sometimes, we must use the system's methods, help can be found from MSDN forum.

Visual Studio C# - The Basics: Timer

- **Timer** is a control to generate a periodical event (timer_tick) that provides some service.
- **ms** is the smallest fraction of time **visual studio** can distinguish, and this is a common convention for high-level programming**.**

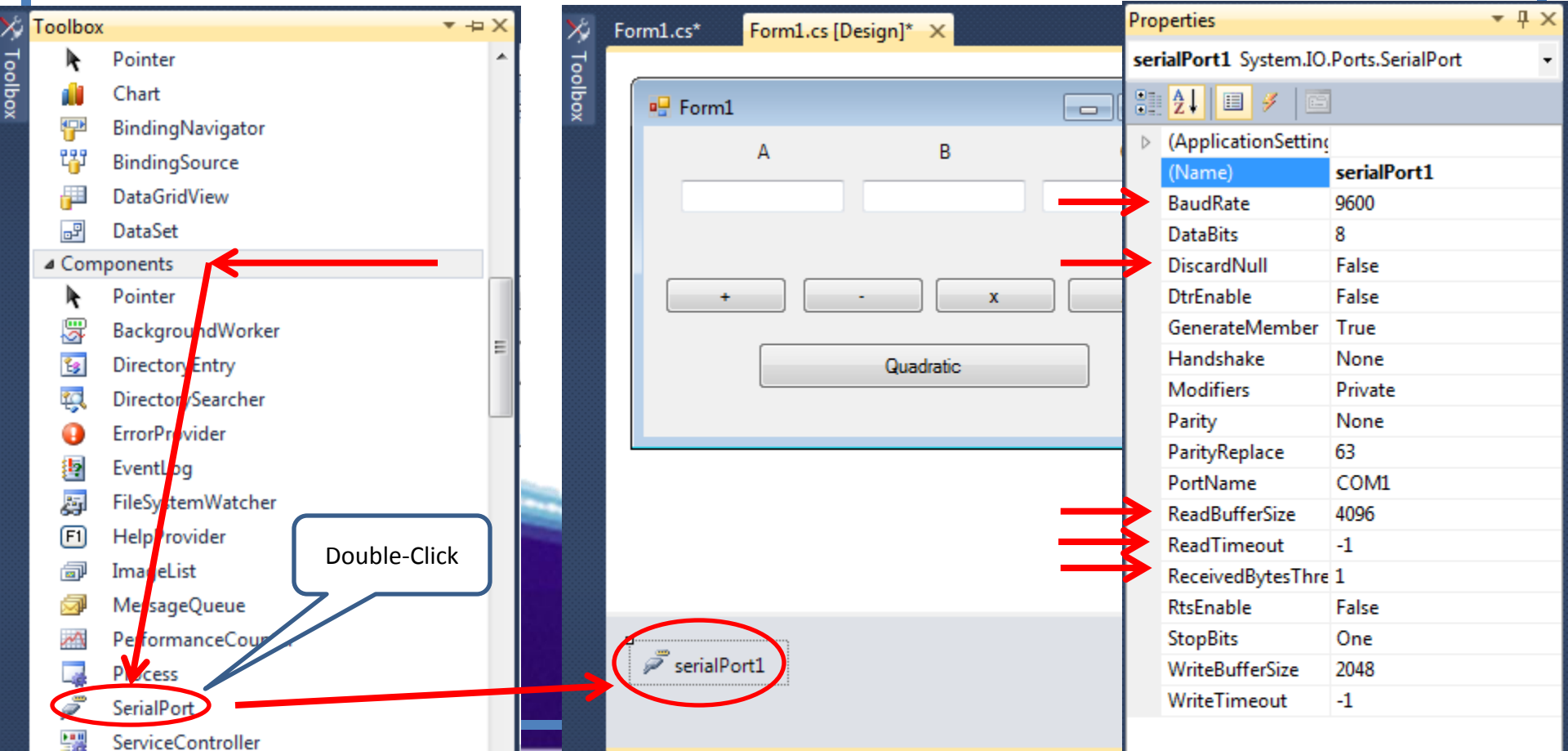




Properties:
Boolean Timer.Enabled
Long Timer.Interval

Visual Studio C# - The Basics: Serial Port

- **Serial Port** is a control much like button. It's useful for communication with the MCU.
- Default properties are usually the same among devices.
- Some **properties** need modification according to applications.



Visual Studio C# - The Basics: Serial Port

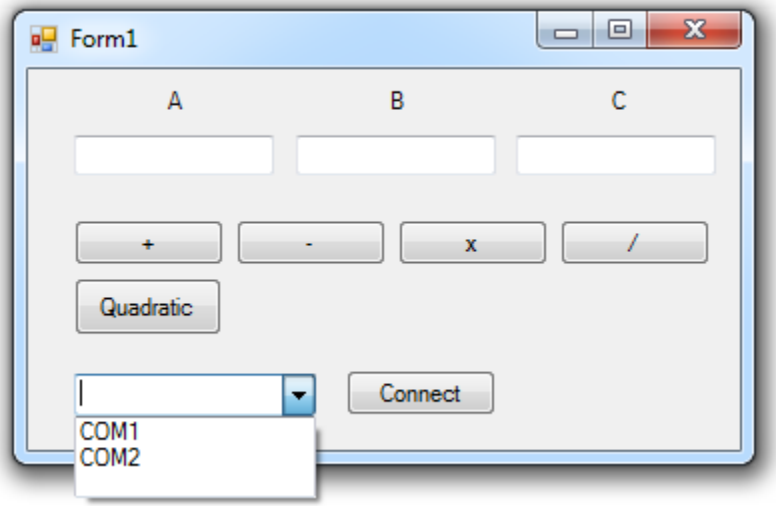
```
using System.IO.Ports;
 …
private void timer1_Tick(object sender, EventArgs e)
{
    int PortNumber = 0;
    string[] ports = SerialPort.GetPortNames();
    if (PortNumber != ports.Length)
    {
        PortNumber = Convert.ToUInt16(ports.Length);
        comboBox1.Items.Clear();
        for (int j = 0; j < PortNumber; j++)
        {
            comboBox1.Items.Add(ports[j]);
        }
        comboBox1.Items.Add("");
    }
}
```

In these codes, for every timer interrupt event we scan for the hardware ports connected to PC and display them in a comboBox *control*

Visual Studio C# - The Basics: Serial Port

- Problem:
  - On last Form of Quadratic Equation, **add some serial port**, **scan for hardware** ports and **display in a comboBox**. A buttonConnect to make connection to a serial port chosen.
  - Receive the values from MCU in IEEE real-number format. Display those on the textboxes, perform calculations.

```
private void buttonConnect_Click(object sender, EventArgs e)
{
    if (comboBox1.Text == "")
    {
        MessageBox.Show("Please select Port!");
    }
    else
    {
        if (buttonConnect.Text == "Connect")
        {
            SerialPort1.PortName = comboBox1.Text;
            try
            {
                SerialPort1.Open();
                buttonConnect.Text = "Disconnect";
                comboBox1.Enabled = false;
            }
            catch
            {
                MessageBox.Show("Please select another Port!");
            }
        }
        else
        {
            buttonConnect.Text = "Connect";        //change button's text to Connect
            comboBox1.Enabled = true;              //enable combo box for selecting port
            SerialPort1.Close();                   //close COM port

        }
    }
}
```



Visual Studio C# - The Basics: Serial Port

```
priv
{
    private void SerialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
    {
        byte txtbox_choice, i;
        byte[] BytesFromMCU = new byte[4];
        txtbox_choice = (byte)SerialPort1.ReadByte();
        float Value = 0;
        for (i = 0; i <= 3; i++)
            BytesFromMCU[i] = (byte)SerialPort1.ReadByte();

        Value = BitConverter.ToSingle(BytesFromMCU, 0);
    switch((char)txtbox_choice)
    {
        case 'a':
            txtboxNumA.Text = Convert.ToString(Value);
            break;
        case 'b':
            txtboxNumA.Text = Convert.ToString(Value);
            break;
        case 'c':
            txtboxNumA.Text = Convert.ToString(Value);
            break;
    }
}
```

InvalidOperationException was unhandled

Cross-thread operation not valid: Control 'txtboxNumA' accessed from a t
other than the thread it was created on.

Troubleshooting tips:

How to make cross-thread calls to Windows Forms controls

Get general help for this exception.

Search for more Help Online...

Actions:

View Detail...

Copy exception detail to the clipboard

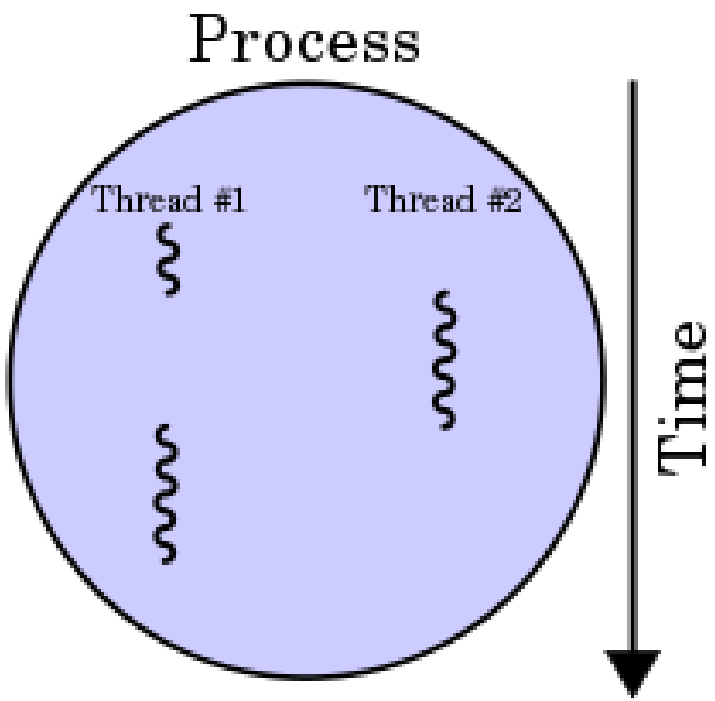
```
    break;
case 'c':
    txtboxNumA.Text = Convert.ToString(Value);
    break;
}
}
```

Will this just work?

There's something wrong...

Visual Studio C# - The Basics: Serial Port

Thread is a sequence of instructions in a process which the hardware is processing at the moment. Some threads may have reference to objects of another, some have not. This brings us back to the fact that all programs are literally written in structured programming method.



Visual Studio C# - The Basics: Delegate and cross-thread reference

- In C# the “dangerous” notion of C’s *pointer* is no longer used: Array’s length needs no declaration, addressing is dynamic and objects exchange values through methods. A ***delegate*** is roughly ***a function pointer*** in traditional C. In this sense, we have a parameter to call for an function not yet determined.
- When a control is not on the current thread, cast a delegate pointing to the function manipulating that control. Use an ***Invoke*** method with that delegate as a parameter to ask the thread containing that control to process the function.

Visual Studio C# - The Basics: Delegate and cross-thread reference

```
private void SerialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
        byte txtbox_choice;
        byte[] BytesFromMCU = new byte[4];
        txtbox_choice = (byte)SerialPort1.ReadByte();
        float Value = 0;
        for (int i = 3; i >= 0; i--)
            BytesFromMCU[i] = (byte)SerialPort1.ReadByte();

        Value = BitConverter.ToSingle(BytesFromMCU, 0);
        display_value((char)txtbox_choice, Value);
}
```

In this code block display_value() is a method that can refer to controls of other thread

```
private delegate void DeleOfdisplay_value(char txtbox_choice, Single Value);
private void display_value(char txtbox_choice, Single Value)
{
    if (txtboxNumA.InvokeRequired)
    {
    DeleOfdisplay_value CrossThreadOfdisplay_value = new DeleOfdisplay_value(display_value);
    txtboxNumA.Invoke(CrossThreadOfdisplay_value, new object[] {txtbox_choice, Value});
    }
    else
    {
        switch ((char)txtbox_choice)
        {
            case 'a':
                txtboxNumA.Text = Convert.ToString(Value);
                break;
            case 'b':
                txtboxNumB.Text = Convert.ToString(Value);
                break;
            case 'c':
                txtboxResult.Text = Convert.ToString(Value);
                break;
        }
    }
}
```

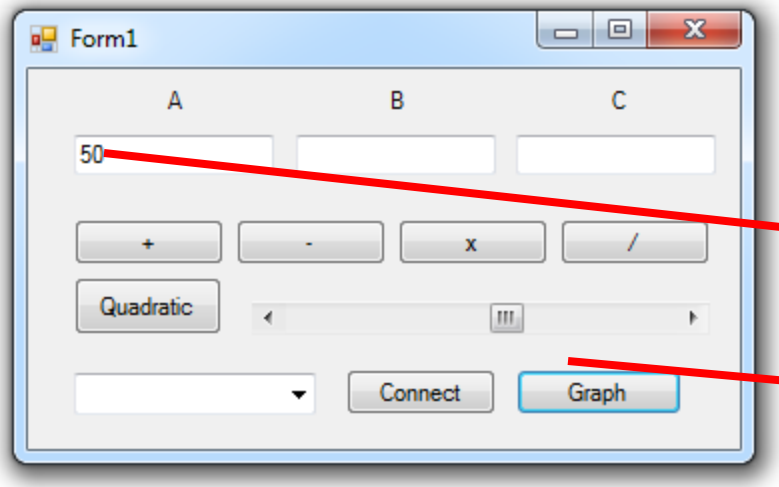
Visual Studio C# - The Basics: Graphing

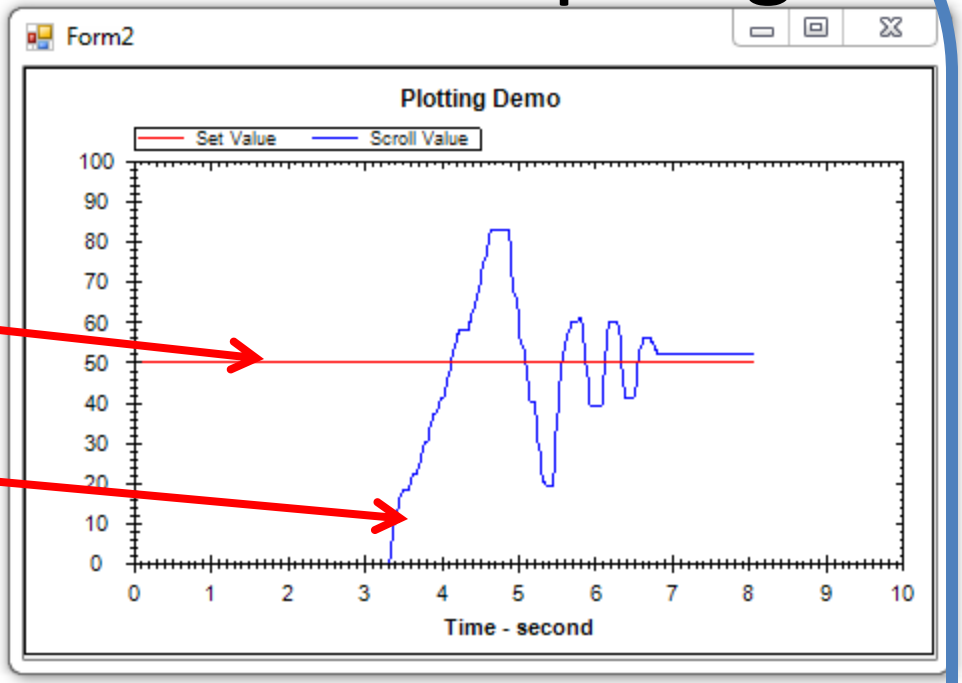
- Graph is a control that is not built-in in VS, we must install a dynamic link library for this control.
- (go to http://www.payitforward.edu.vn/forum/threads/42/ for directions of how to manipulate ZedGraph).

Visual Studio C# - The Basics: Graphing

- Problem:
  - Add a new GraphForm in your project containing a zedGraphControl.
  - On the QuadraticForm add a slider, a buttonGraph which calls GraphForm. The GraphForm plots the value of txtBoxNumA when it's called and the real-time value of the slider.

Visual Studio C# - The Basics: Graphing





Visual Studio C# - The Basics: Graphing

```
namespace GUICalDemo
{
    public partial class GraphForm : Form
    {
        Form1 CalForm;
        double setValue;
        public GraphForm(Form1 sender, double tbA)
        {
            CalForm = sender;
            setValue = tbA;
            InitializeComponent();
        }

        private void GraphForm_FormClosed(object sender, FormClosedEventArgs e)
        {
            CalForm.GraphForm = null;
        }
//some stuff omitted here!
}
```

```
  public partial class GraphForm : Form
  {
   public GraphForm GraphForm = null;
   private void buttonGraph_Click(object sender, EventArgs e)
   {
       if (GraphForm != null)
           GraphForm.Close();
       GraphForm = new GraphForm(this, Convert.ToDouble(txtboxNumA.Text));
       GraphForm.Enabled = true;
       GraphForm.Show();
   })
```

Mutual recognition between the 2 forms.

Visual Studio C# - The Basics: Graphing

- Initial the zedGraphControl (within the namespace of GraphForm)

```
int TickStart;
private void GraphForm_Load(object sender, EventArgs e)
{
    RollingPointPairList ReferenceValue = new RollingPointPairList(10000);
    RollingPointPairList ResponseValue = new RollingPointPairList(10000);

    LineItem RefCurve = zedGraphControl1.GraphPane.AddCurve("Set Value", ReferenceValue, Color.Red, SymbolType.None ;
    LineItem ResCurve = zedGraphControl1.GraphPane.AddCurve("Scroll Value", ResponseValue, Color.Blue, SymbolType.None

    zedGraphControl1.GraphPane.Title.Text = "Plotting Demo";
    zedGraphControl1.GraphPane.XAxis.Title.Text = "Time - second";
    zedGraphControl1.GraphPane.YAxis.Title.Text = " ";

    TickStart = Environment.TickCount;
    zedGraphControl1.GraphPane.XAxis.Scale.Min = 0;
    zedGraphControl1.GraphPane.XAxis.Scale.Max = 10;
    zedGraphControl1.GraphPane.XAxis.Scale.MinorStep = 0.1;
    zedGraphControl1.GraphPane.XAxis.Scale.MajorStep = 1;
}
```

Visual Studio C# - The Basics: Graphing

- Add a timer2 in the GraphForm with a sampling interval. Sample and rescale the Graph in each tick event!

```
private void timer2_Tick(object sender, EventArgs e)
{
    double scrollValue;

    scrollValue = Convert.ToDouble(CalForm.hScrollBar1.Value);
    if (zedGraphControl1.GraphPane.CurveList.Count <= 0)
        return;

    LineItem RefCurve = zedGraphControl1.GraphPane.CurveList[0] as LineItem;
    LineItem ResCurve = zedGraphControl1.GraphPane.CurveList[1] as LineItem;

    if (RefCurve == null || ResCurve == null)
        return;

    IPointListEdit RefPoints = RefCurve.Points as IPointListEdit;
    IPointListEdit ResPoints = ResCurve.Points as IPointListEdit;

    if (RefPoints == null || ResPoints == null)
        return;

    double time = (Environment.TickCount - TickStart)/1000.0;

    RefPoints.Add(time, setValue);
    ResPoints.Add(time, scrollValue);

    Scale XScale = zedGraphControl1.GraphPane.XAxis.Scale;

    if (time > XScale.Max - XScale.MajorStep)
    {
        XScale.Max = time + XScale.MajorStep;
        XScale.Min = 0;
    }
    zedGraphControl1.AxisChange();
    zedGraphControl1.Invalidate();

}
```

Visual Studio C# - The Basics: Packaging

- To provide an install-and-use package that users would not be concerned of having VS or particular controls employed in the project.
- (go to http://www.payitforward.edu.vn/forum/threads/42/ for directions of how to package your stuff).

Thank You!