

InstrumentLab 7.5

.NET Quick Start



www.openwire.org
www.mitov.com

Copyright Boian Mitov 2004 - 2014

Index

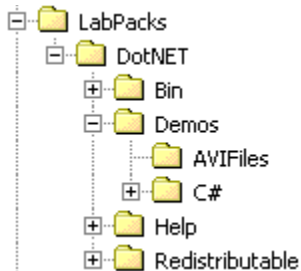
Installation.....	3
Where is InstrumentLab?.....	3
Creating a new InstrumentLab project in Visual C#.....	3
Installing the components on the Toolbox.....	5
Adding the necessary assembly references to your application.....	7
Creating application with angular gauge.....	8
Adding component elements to a gauge.....	10
Distributing your application.....	12
Deploying your 32 bit application with the IPP DLLs.....	13
Deploying your 64 bit application.....	13

Installation

InstrumentLab comes with an installation program. Just start the installation by double-clicking on the Setup.exe file and follow the installation instructions.

Where is InstrumentLab?

After the installation InstrumentLab is located under a single root directory. The default location is C:\Program Files\LabPacks or C:\Program Files (x86)\LabPacks on 64 bit systems. During the installation the user has the option to select alternative directory. Here is how the directory structure should look like after the installation:



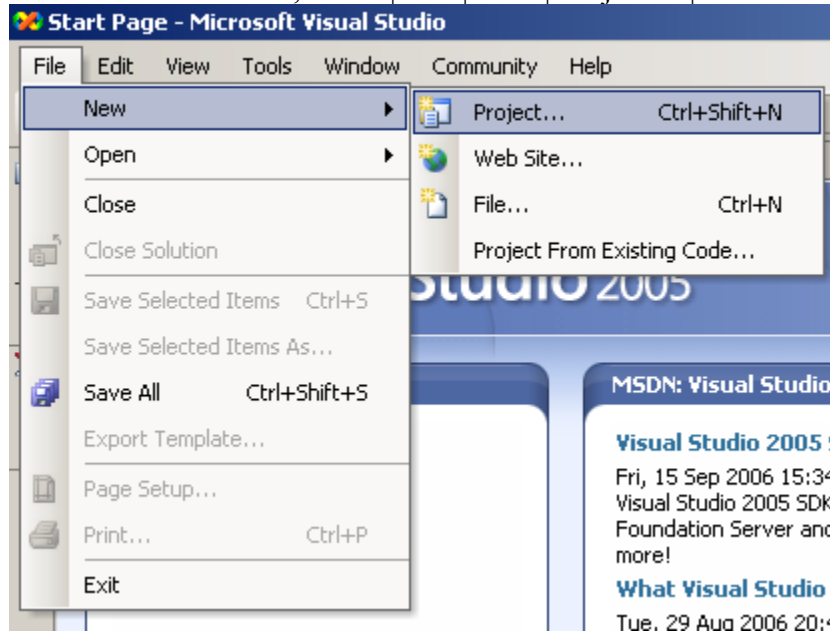
Under the “Demos” directory are located the demo files. The help files and the documentation are located under the “Help” directory. The component .NET 2.0/3.5/4.0 assemblies and the redistributable DLL files are located under the “Bin” directory. It is a great idea to start by opening and compiling the demo files. The demo projects were developed with Visual C# 2005.

Creating a new InstrumentLab project in Visual C#

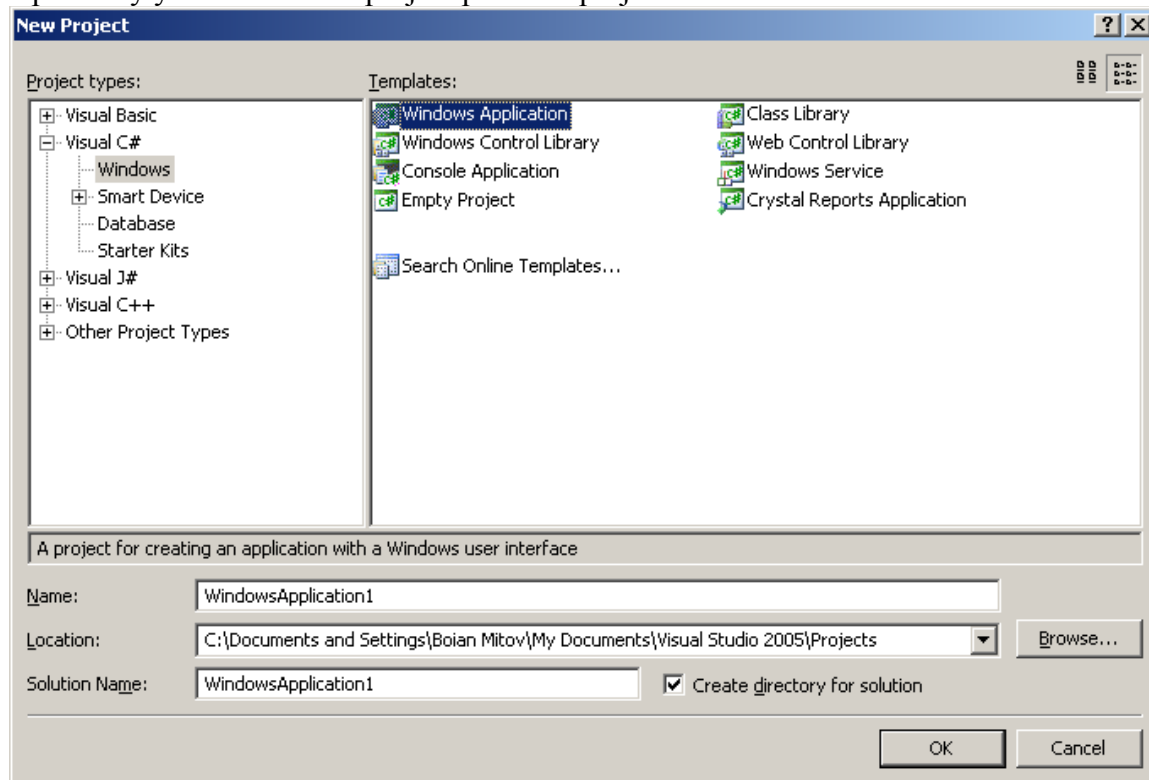
All of the examples in this manual start with creating a C# Windows .NET based project. The following chapters will assume that you have created the project and will teach you how to add specific InstrumentLab functionality.

Start by creating a new project.

From the VC++ menu, select | File | New | Project... |



In the "New Project" dialog select | Visual C# | Windows Application |
Optionally you can select a project path and project name:



Click OK.

Installing the components on the Toolbox

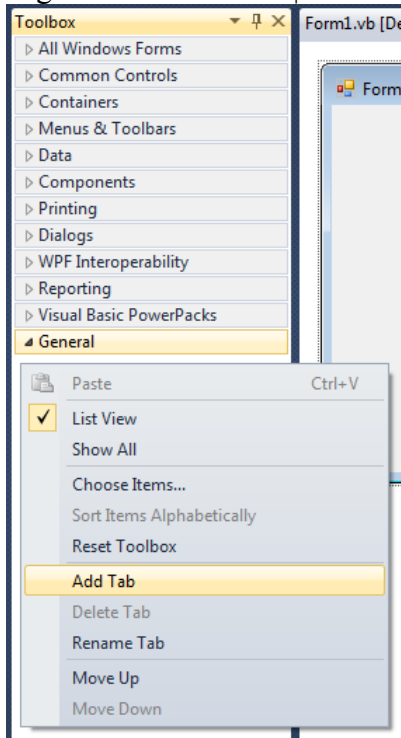
Before using the components in your project, you will have to install them on the component Toolbox.

The install in version 3.1 and up will automatically install the components on the toolbar, however if it fails, or if you have selected not to do so during the installation, here is a way to install the components manually:

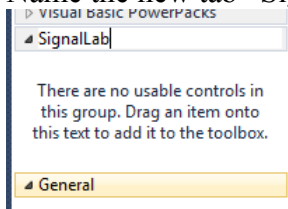
We assume that you have already created a project, and the toolbox with the .NET components has appeared.

Open the component toolbox and expand the General section.

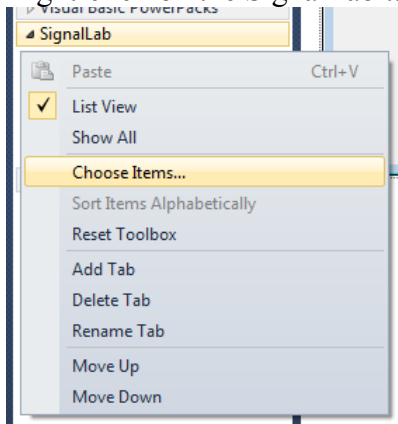
Right-click and select **Add Tab** from the menu:



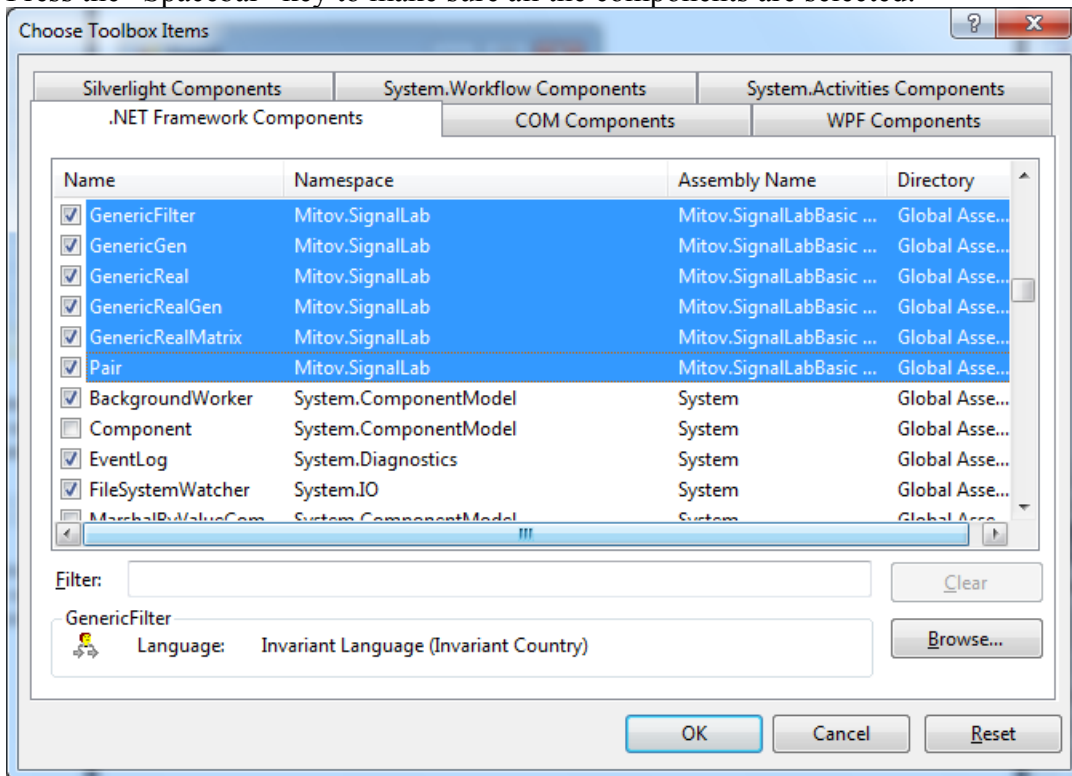
Name the new tab “SignalLab”:



Right-click on the SignalLab tab and select |Choose Items...| from the menu:

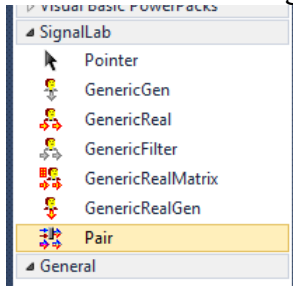


In the “Choose Toolbox Items” dialog select the components that belong to the Mitov.SignalLabBasic.DLL (You can order the components by “Assembly Name”). Press the “Spacebar” key to make sure all the components are selected:



Click OK.

You should see the SignalLabBasic components on your toolbox:



Close and restart the Visual Studio IDE, then reopen the project.

Continue repeating the same steps and install the following assemblies:

On the “SignalLab” tab install Mitov.SignalLabAdditional.dll.

On the “UserLab” tab and install Mitov.UserLabBasic.dll.

On the “InstrumentLab” tab and install Mitov.InstrumentLabDigital.dll and Mitov.InstrumentLab.dll.

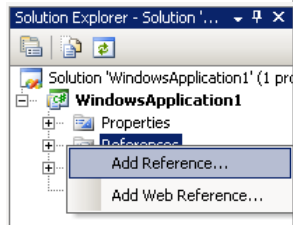
Now you can start using the components in your .NET development.

Adding the necessary assembly references to your application

Visual studio will automatically add the assemblies being referenced when adding components to the project. If this mechanism fails, you can manually add the necessary assemblies as shown here:

In the “Solution Explorer” select the “References” node and right-click on it.

From the menu select |Add Reference...|



Navigate to the Select the AudioLabBasic.dll from the LabPacks\Bin\2.0 subdirectory and add the necessary assemblies.

Here is the list of necessary assemblies:

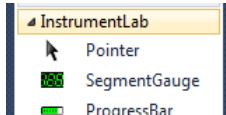
- For Mitov.BasicLab.DLL – None.
- For Mitov.SignalLabBasic.DLL
 - a. Mitov.BasicLab.DLL
- For Mitov.SignalLabAdditional.DLL:
 - a. Mitov.BasicLab.DLL
- For Mitov.UserLabBasic.DLL:
 - a. Mitov.BasicLab.DLL
- For Mitov.InstrumentLabDigital.DLL:
 - a. Mitov.BasicLab.DLL

- b. Mitov.UserLabBasic.DLL
- For Mitov.InstrumentLab.DLL:
 - a. Mitov.InstrumentLabDigital.DLL
 - b. Mitov.BasicLab.DLL
 - c. Mitov.UserLabBasic.DLL

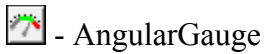
Creating application with angular gauge

Create and setup a new project as described in the “Creating a new InstrumentLab project in Visual C#” chapter.

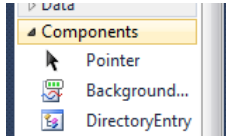
From the “Toolbox” select the “Instrument Lab” tab:



select and drop on the form the following component:



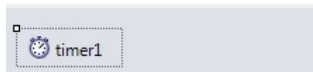
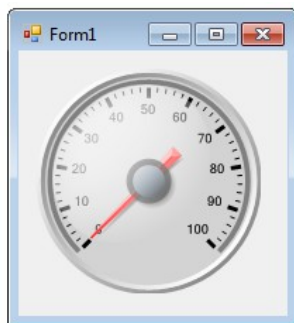
From the “Toolbox” select the “Components” tab:



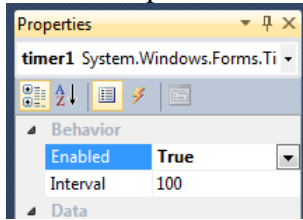
select and drop on the form the following component:



On the form select the timer1:



In the “Properties” set the Enabled property to True:



Double-click the timer1.

In the Form1 add the highlighted lines:

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

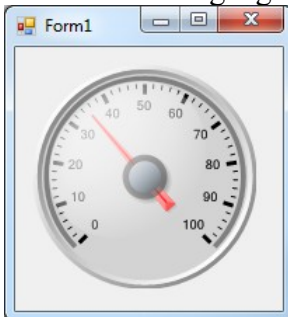
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        private Random m_RandomGen = new Random(123);

        public Form1()
        {
            InitializeComponent();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            angularGauge1.Value = m_RandomGen.NextDouble() * 100;
        }
    }
}
```

Compile and run the application.

You should see gauge showing the random values:

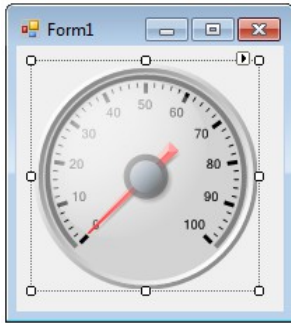


You have just learned how to use InstrumentLab gauges.

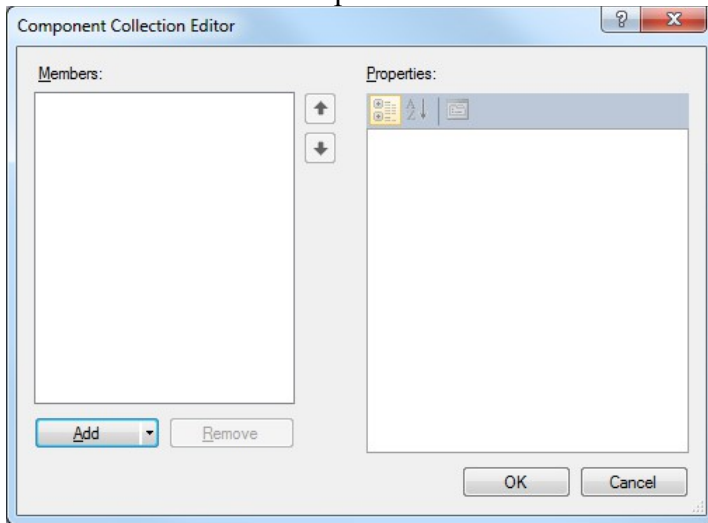
Adding component elements to a gauge

Open the application already created in the “Creating application with angular gauge” chapter.

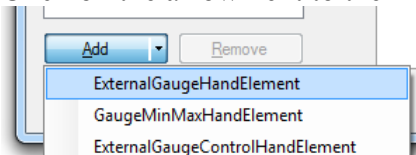
Double-click the gauge1:



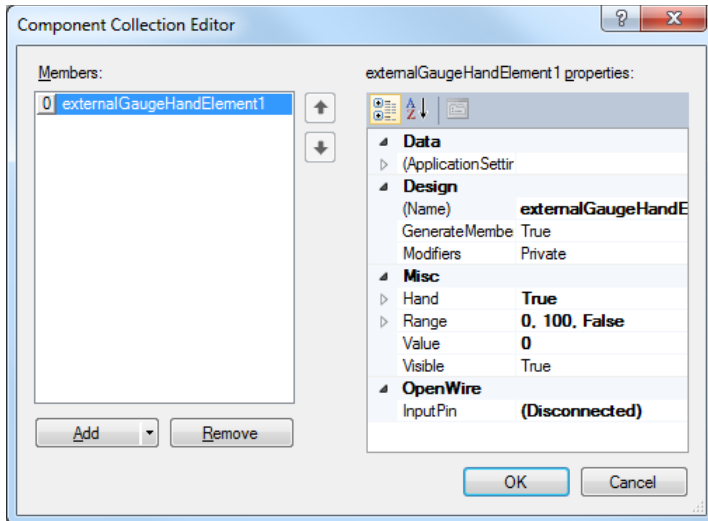
You should see the “Component Collection Editor”:



Click on the arrow next to the “Add” button and select “ExternalGaugeHandElement”:



Click OK:

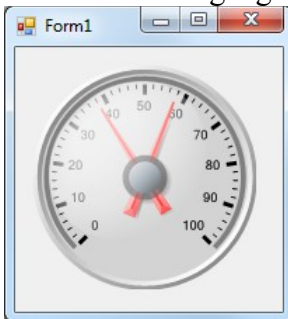


Switch to the code and add the highlighted line to the timer1_Tick event handler:

```
private void timer1_Tick(object sender, EventArgs e)
{
    angularGauge1.Value = m_RandomGen.NextDouble() * 100;
    externalGaugeHandElement1.Value =
m_RandomGen.NextDouble() * 100;
}
```

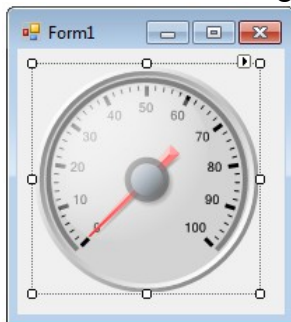
Compile and run the application.

You should see gauge showing the random values with both the hands:

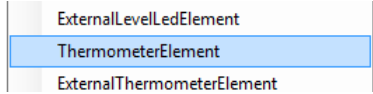


Stop the application.

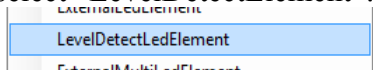
Double-click the Gauge1:



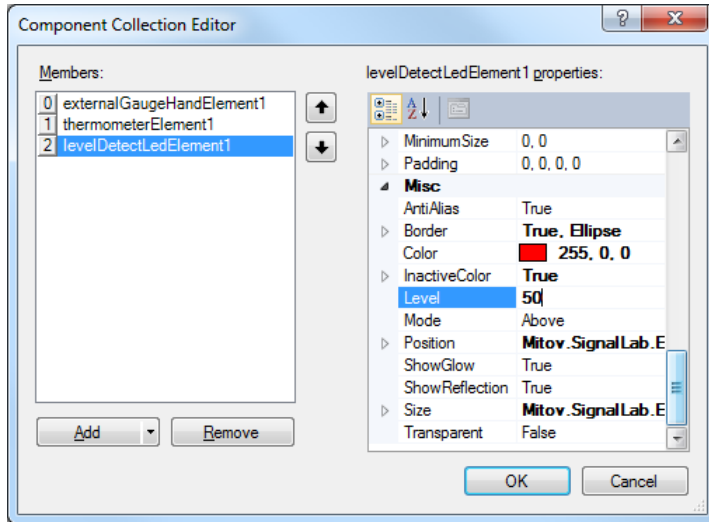
In the “Component Collection Editor” click on the arrow next to the “Add” button and select “ThermometerElement”:



In the “Component Collection Editor” click on the arrow next to the “Add” button and select “LevelDetectElement”:

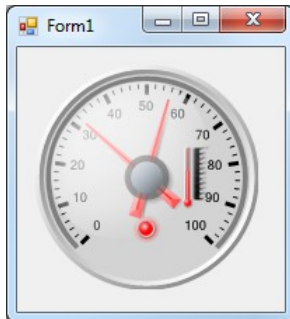


Click OK:



Compile and run the application.

You should see gauge showing the random values with both the hands, the LED will be On when the first hand is above 50 and the Thermometer will show the value of the first hand:



You have just learned how to create composite components in InstrumentLab.

Distributing your application

Once you have finished the development of your application you most likely will need to distribute it to other systems. Version 5.0.2 and higher of the library will move all the necessary DLL files in the Release directory of your project. You will only need to distribute the files in the directory. To use this feature, make sure that the “Copy Local” property is set for all the assemblies from the library. Please check with the Visual Studio

help for your version of Video Studio on how to configure assemblies as private assemblies.

Deploying your 32 bit application with the IPP DLLs

The compiled applications can be deployed to the target system by simply copying the executable. The application will work, however the performance can be improved by also copying the Intel IPP DLLs provided with the library.

The DLLs are under the [install path]\LabPacks\IppDLL\Win32 directory([install path] is the location where the library was installed).

In 32 bit Windows to deploy IPP, copy the files to the [Windows]\System32 directory on the target system.

In 64 bit Windows to deploy IPP, copy the files to the [Windows]\SysWOW64 directory on the target system.

[Windows] is the Windows directory - usually C:\WINNT or C:\WINDOWS

This will improve the performance of your application on the target system.

Deploying your 64 bit application

The current version of the library requires when deploying 64 bit applications, the Intel IPP DLLs to be deployed as well.

The DLLs are under the [install path]\LabPacks\IppDLL\Win64 directory([install path] is the location where the library was installed).

To deploy IPP, copy the files to the [Windows]\System32 directory on the target system.

[Windows] is the Windows directory - usually C:\WINNT or C:\WINDOWS

This will improve the performance of your application on the target system.