https://www.halvorsen.blog

This Tutorial uses the new **ScottPlot 5**

Plotting Data in Windows Forms

Hans-Petter Halvorsen

Contents

Introduction

Plotting Data in Windows Forms

Table of Contents

Introduction

- Plotting Data in Windows Forms is something you often want to do
- The built-in Chart Control for Windows Forms is no longer supported by Microsoft in the latest .NET versions
- So, we need to find and use alternative solutions

Charting/Plotting in WinForms

- No built-in Charts in Visual Studio
  - System.Windows.Forms.DataVisualization no longer exist for .NET 6 or newer
- Many third-party Chart Tools exist
- Most of these are commercial and costs money while others are free
- This Tutorial will use **ScottPlot**, which is a free and open-source plotting library for .NET

ScottPlot.NET
ScottPlot

Getting Started with ScottPlot

ScottPlot is a free and open-source plotting library for .NET

Hans-Petter Halvorsen

Table of Contents

ScottPlot

- ScottPlot is a free and open-source plotting library for .NET
- You start using it by installing a NuGet Package directly from Visual Studio
- ScottPlot 5 is released! – and not directly backward compatible with ScottPlot 4.x!!
- https://scottplot.net/

Getting Started with ScottPlot

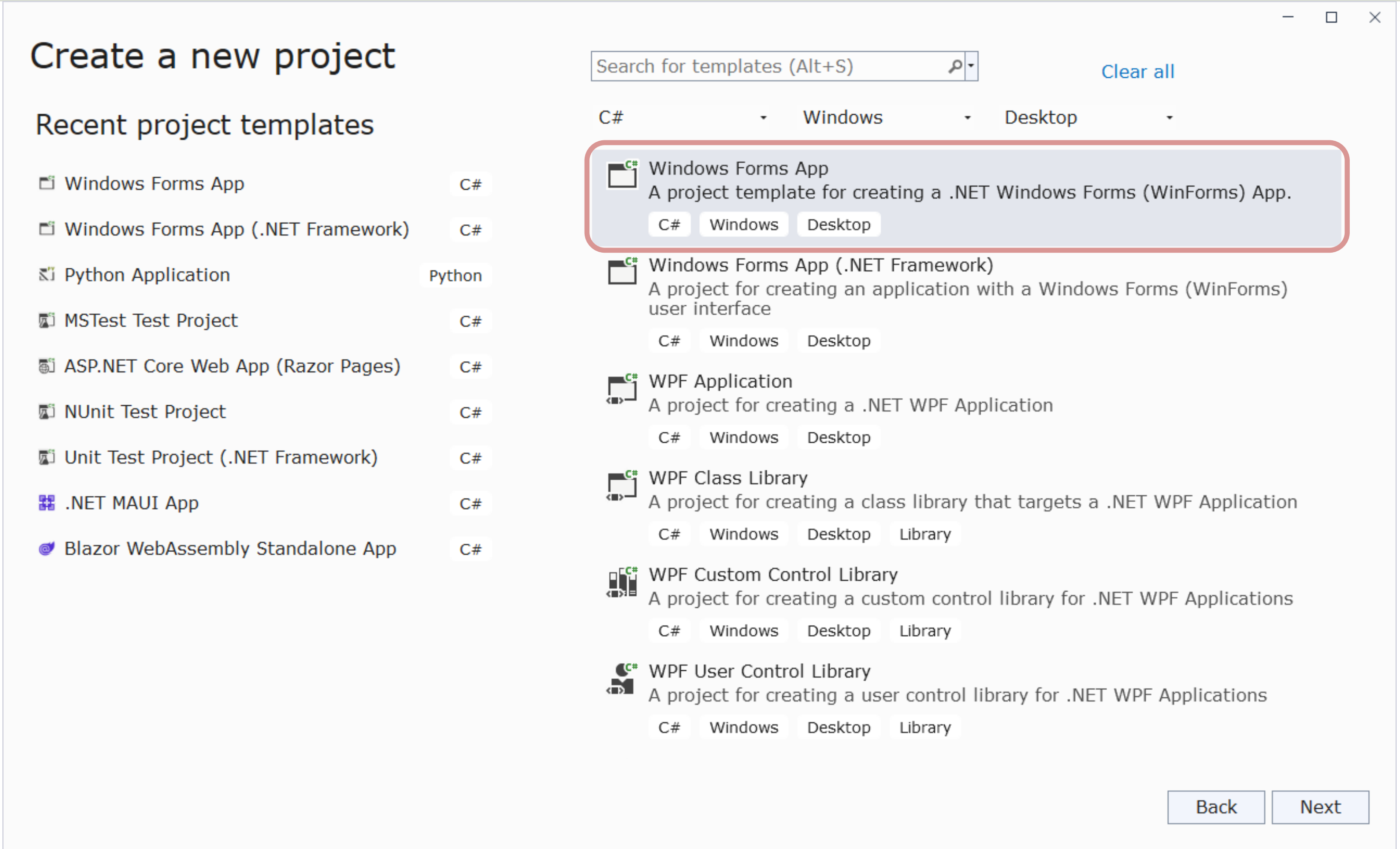
Getting Started:

- Step 1: Install the **ScottPlot.WinForms** NuGet package
- Step 2: Drag a **FormsPlot** from the Toolbox onto your Form

Basic Code Example:

```
double[] dataX = new double[] {1, 2, 3, 4, 5};
double[] dataY = new double[] {1, 4, 9, 16, 25};
formsPlot1.Plot.Add.Scatter(dataX, dataY);
formsPlot1.Refresh();
```

Windows Forms App



Create a new project

Recent project templates

Windows Forms App	C#
Windows Forms App (.NET Framework)	C#
Python Application	Python
MSTest Test Project	C#
ASP.NET Core Web App (Razor Pages)	C#
NUnit Test Project	C#
Unit Test Project (.NET Framework)	C#
.NET MAUI App	C#
Blazor WebAssembly Standalone App	C#

Search for templates (Alt+S)

Clear all

C# · Windows · Desktop

Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
C# Windows Desktop

Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Windows Desktop

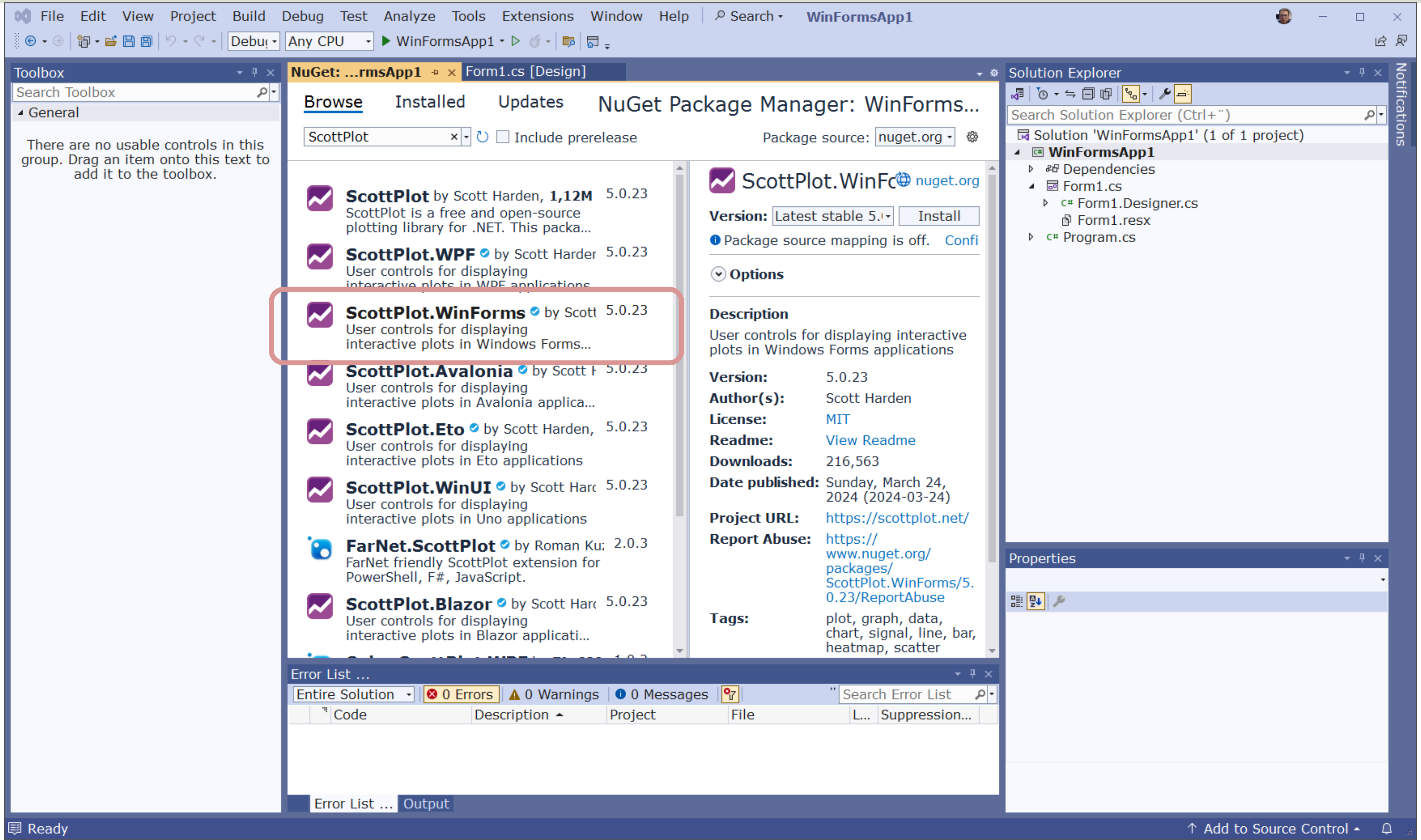
WPF Application
A project for creating a .NET WPF Application
C# Windows Desktop

WPF Class Library
A project for creating a class library that targets a .NET WPF Application
C# Windows Desktop Library

WPF Custom Control Library
A project for creating a custom control library for .NET WPF Applications
C# Windows Desktop Library

WPF User Control Library
A project for creating a user control library for .NET WPF Applications
C# Windows Desktop Library

ScottPlot NuGet Package


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search
WinFormsApp1
Debug
Any CPU
WinFormsApp1
Toolbox
Search Toolbox
General

There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

NuGet Package Manager: WinForms...

ScottPlot

Include prerelease

Package source: nuget.org

ScottPlot by Scott Harden, **1,12M** ScottPlot is a free and open-source plotting library for .NET. This packa...	5.0.23
ScottPlot.WPF by Scott Harder User controls for displaying interactive plots in WPF applications	5.0.23
ScottPlot.WinForms by Scott User controls for displaying interactive plots in Windows Forms...	5.0.23
ScottPlot.Avalonia by Scott H User controls for displaying interactive plots in Avalonia applica...	5.0.23
ScottPlot.Eto by Scott Harden, User controls for displaying interactive plots in Eto applications	5.0.23
ScottPlot.WinUI by Scott Harc User controls for displaying interactive plots in Uno applications	5.0.23
FarNet.ScottPlot by Roman Ku: FarNet friendly ScottPlot extension for PowerShell, F#, JavaScript.	2.0.3
ScottPlot.Blazor by Scott Harc User controls for displaying interactive plots in Blazor applicati...	5.0.23

ScottPlot.WinFc nuget.org

Version: Latest stable 5.0 Install

Package source mapping is off. Confi

Options

Description

User controls for displaying interactive plots in Windows Forms applications

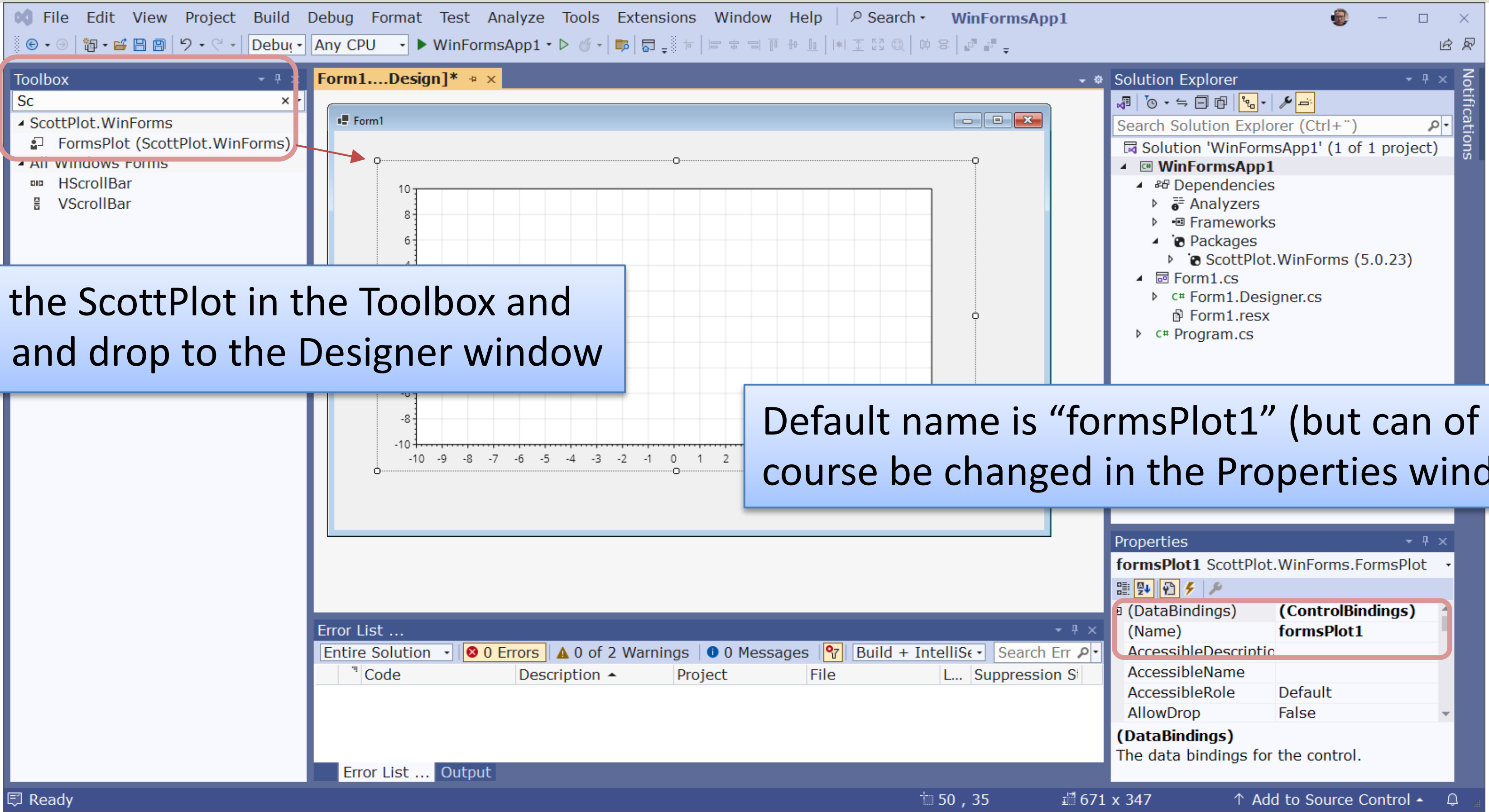
Version:	5.0.23
Author(s):	Scott Harden
License:	MIT
Readme:	View Readme
Downloads:	216,563
Date published:	Sunday, March 24, 2024 (2024-03-24)
Project URL:	https://scottplot.net/
Report Abuse:	https://www.nuget.org/packages/ScottPlot.WinForms/5.0.23/ReportAbuse
Tags:	plot, graph, data, chart, signal, line, bar, heatmap, scatter

Solution Explorer

Search Solution Explorer (Ctrl+")

```
Solution 'WinFormsApp1' (1 of 1 project)
  WinFormsApp1
    Dependencies
    Form1.cs
      Form1.Designer.cs
      Form1.resx
    Program.cs
```

Visual Studio Designer



Properties

formsPlot1 ScottPlot.WinForms.FormsPlot

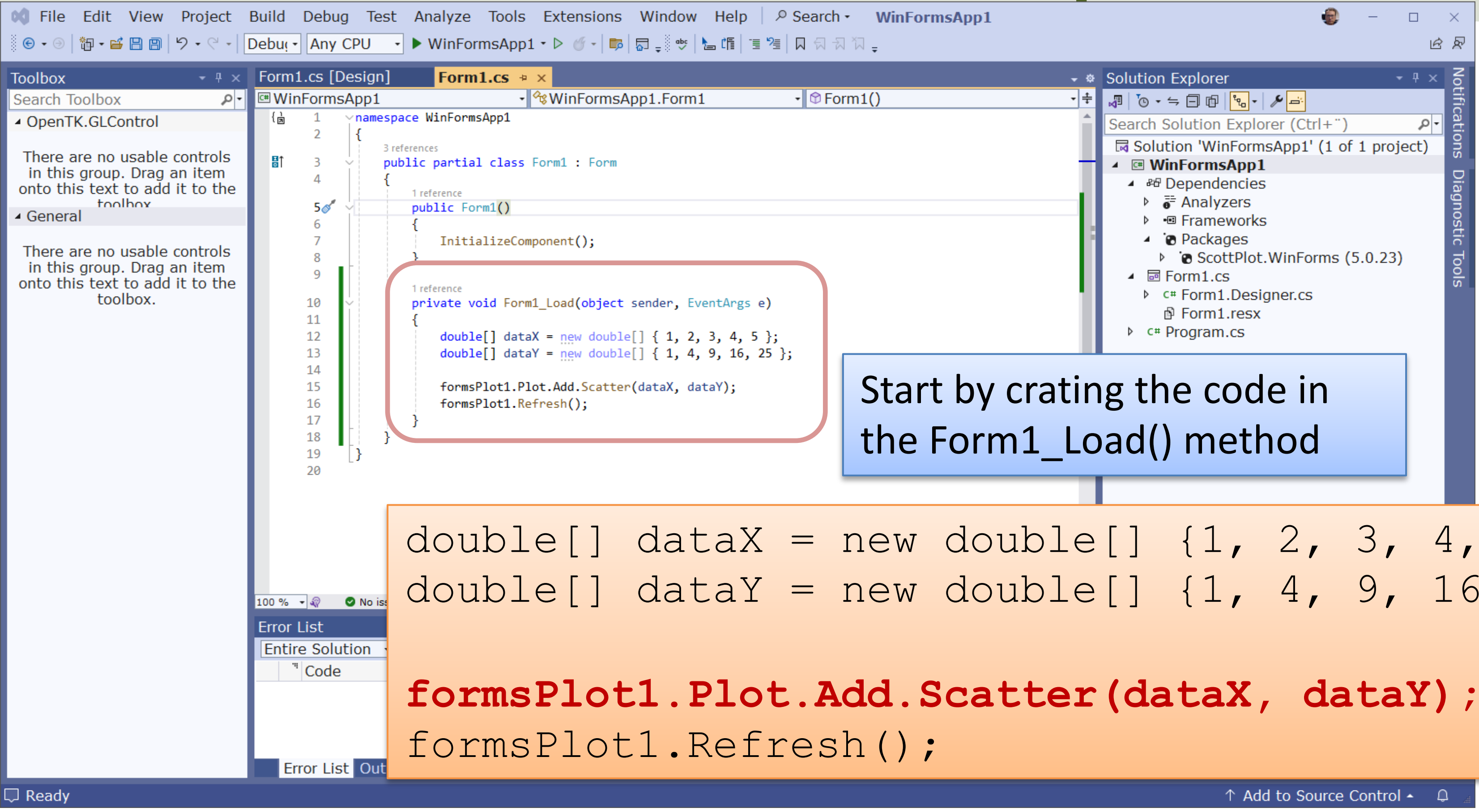
(DataBindings)	(ControlBindings)
(Name)	formsPlot1
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False

(DataBindings)
The data bindings for the control.

Find the ScottPlot in the Toolbox and drag and drop to the Designer window

Default name is "formsPlot1" (but can of course be changed in the Properties window)

Basic Example

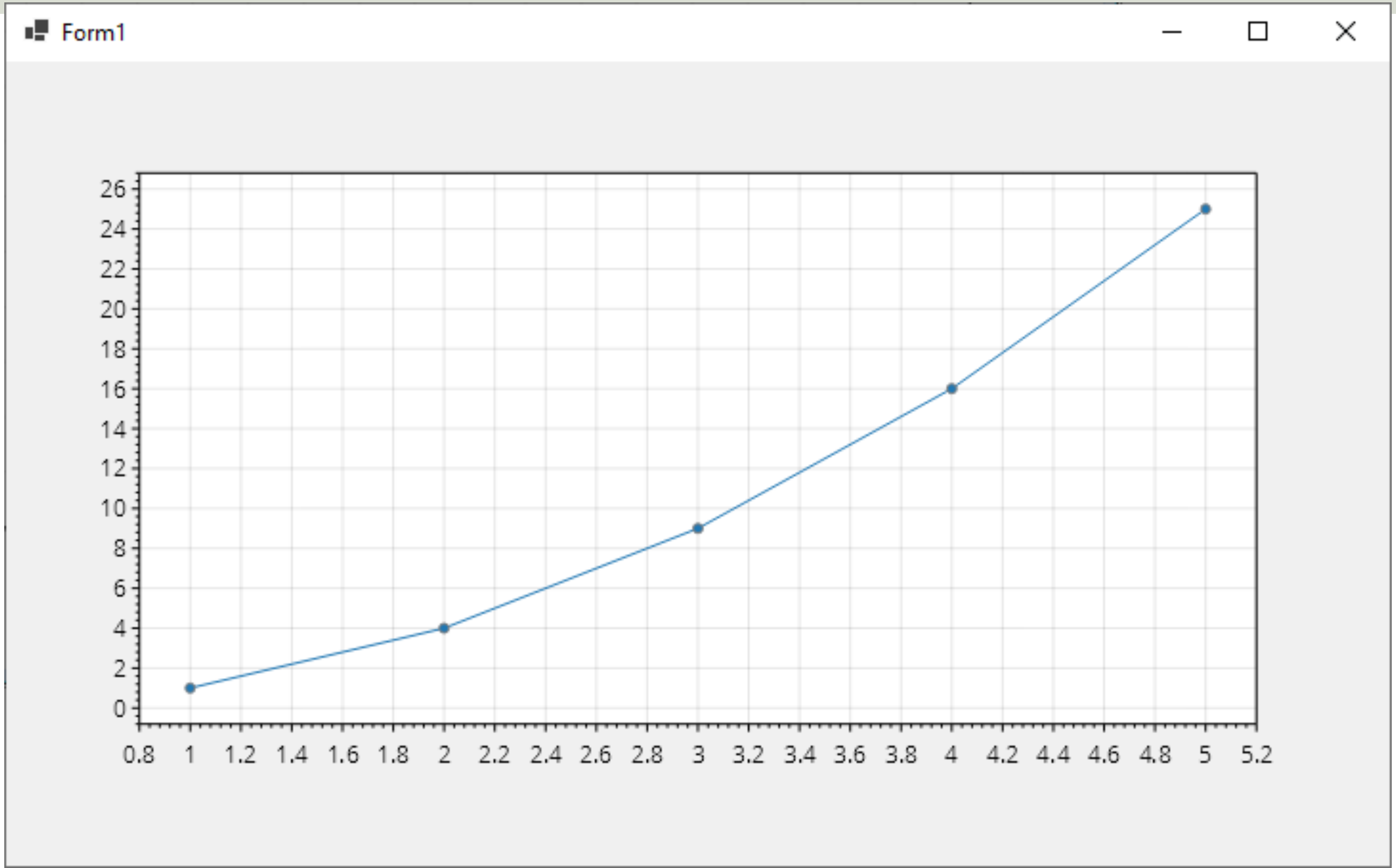

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search
WinFormsApp1
Debug
Any CPU
WinFormsApp1
Toolbox
Search Toolbox
OpenTK.GLControl
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Form1.cs [Design]
Form1.cs
WinFormsApp1
WinFormsApp1.Form1
Form1()
100 %
No iss
Error List
Entire Solution
Code
Error List
Out
Solution Explorer
Search Solution Explorer (Ctrl+¨)
Solution 'WinFormsApp1' (1 of 1 project)
WinFormsApp1
Dependencies
Analyzers
Frameworks
Packages
ScottPlot.WinForms (5.0.23)
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Notifications
Diagnostic Tools
Ready
Add to Source Control

```
namespace WinFormsApp1
{
    3 references
    public partial class Form1 : Form
    {
        1 reference
        public Form1()
        {
            InitializeComponent();
        }

        1 reference
        private void Form1_Load(object sender, EventArgs e)
        {
            double[] dataX = new double[] { 1, 2, 3, 4, 5 };
            double[] dataY = new double[] { 1, 4, 9, 16, 25 };

            formsPlot1.Plot.Add.Scatter(dataX, dataY);
            formsPlot1.Refresh();
        }
    }
}
```

Start by crating the code in the Form1_Load() method

```
double[] dataX = new double[] {1, 2, 3, 4, 5};
double[] dataY = new double[] {1, 4, 9, 16, 25};

formsPlot1.Plot.Add.Scatter(dataX, dataY);
formsPlot1.Refresh();
```

Basic Example



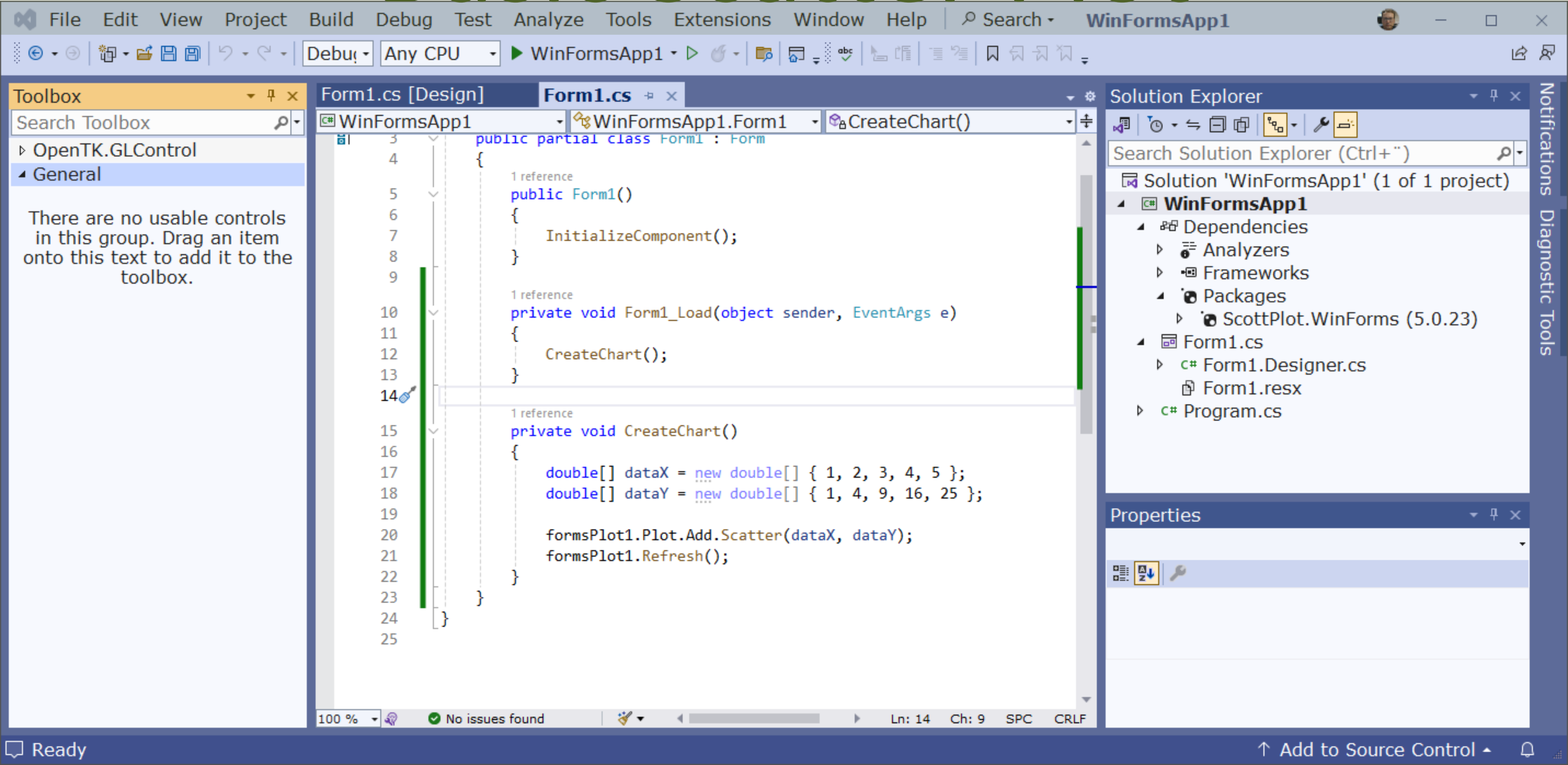
```
using ScottPlot;

namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            double[] dataX = new double[] { 1, 2, 3, 4, 5 };
            double[] dataY = new double[] { 1, 4, 9, 16, 25 };

            formsPlot1.Plot.Add.Scatter(dataX, dataY);
            formsPlot1.Refresh();
        }
    }
}
```

Basic Scatter Plot

Hans-Petter Halvorsen

Table of Contents

Basic Scatter Plot

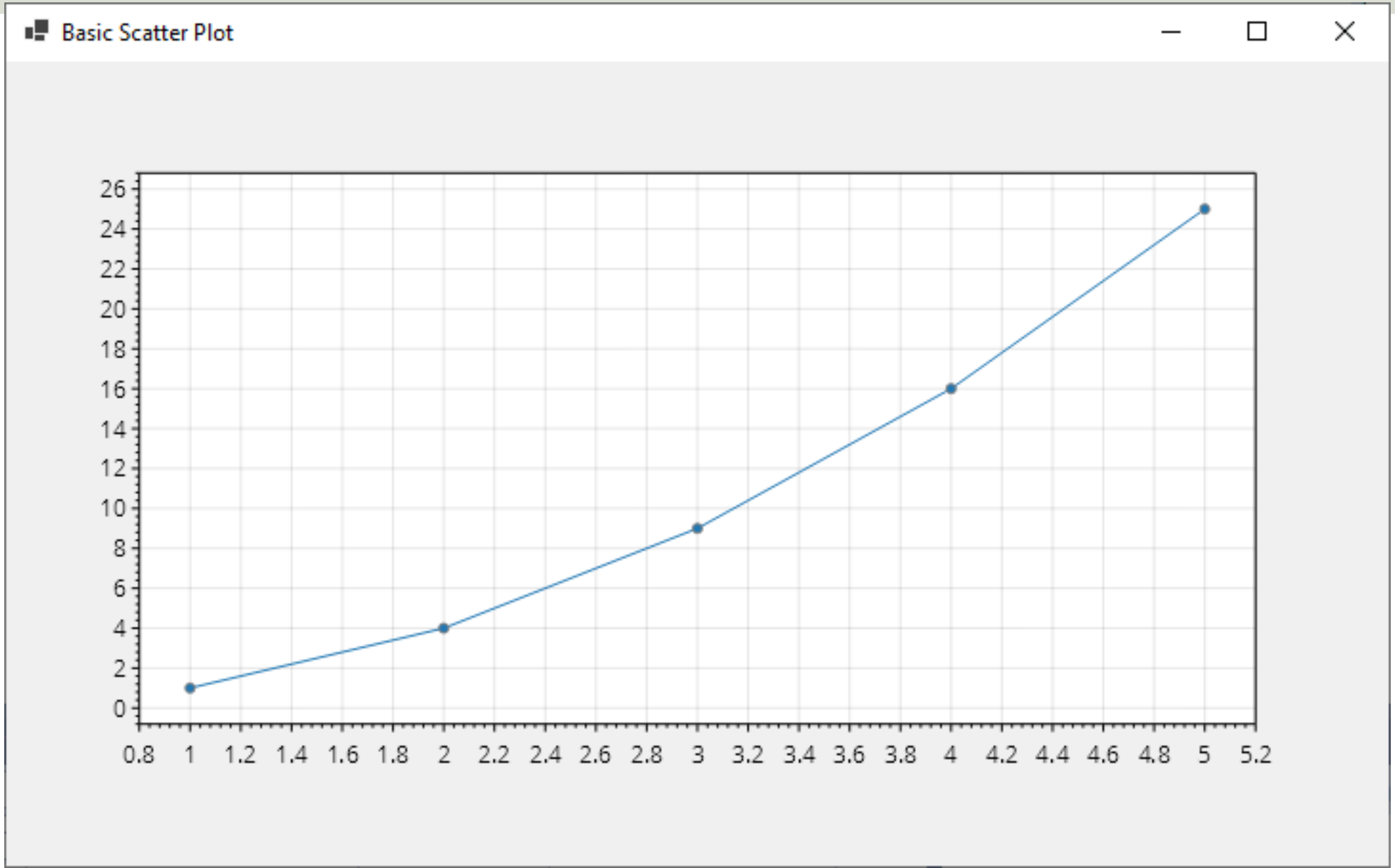


There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

```
    public partial class Form1 : Form
    {
        1 reference
        public Form1()
        {
            InitializeComponent();
        }

        1 reference
        private void Form1_Load(object sender, EventArgs e)
        {
            CreateChart();
        }

        1 reference
        private void CreateChart()
        {
            double[] dataX = new double[] { 1, 2, 3, 4, 5 };
            double[] dataY = new double[] { 1, 4, 9, 16, 25 };

            formsPlot1.Plot.Add.Scatter(dataX, dataY);
            formsPlot1.Refresh();
        }
    }
}
```

```
Solution 'WinFormsApp1' (1 of 1 project)
  WinFormsApp1
    Dependencies
      Analyzers
      Frameworks
      Packages
        ScottPlot.WinForms (5.0.23)
    Form1.cs
      Form1.Designer.cs
      Form1.resx
    Program.cs
```

Basic Scatter Plot



```
using ScottPlot;
namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            CreateChart();
        }

        private void CreateChart()
        {
            double[] dataX = new double[] { 1, 2, 3, 4, 5 };
            double[] dataY = new double[] { 1, 4, 9, 16, 25 };
            formsPlot1.Plot.Add.Scatter(dataX, dataY);
            formsPlot1.Refresh();
        }
    }
}
```

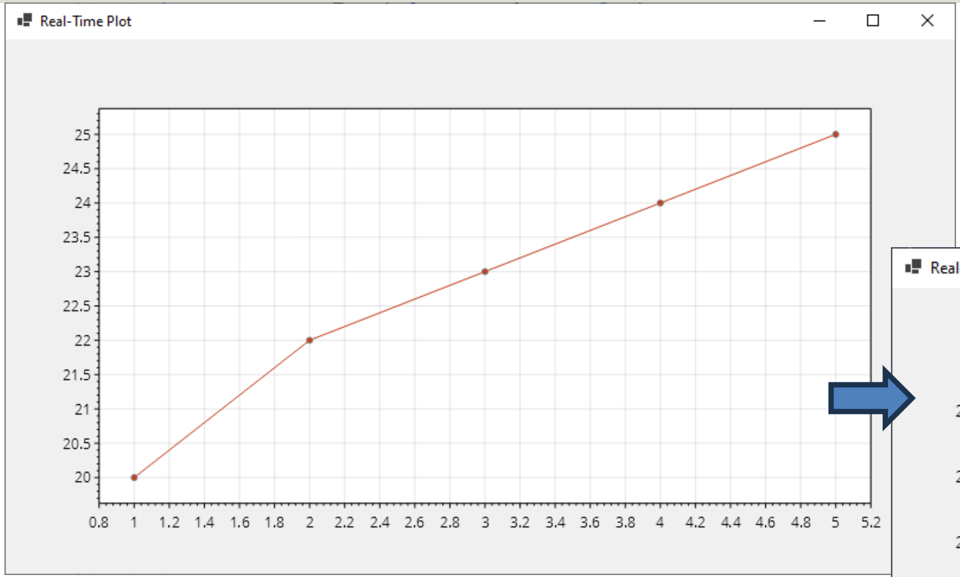
ScottPlot.NET
ScottPlot

Real-Time Scatter Plot

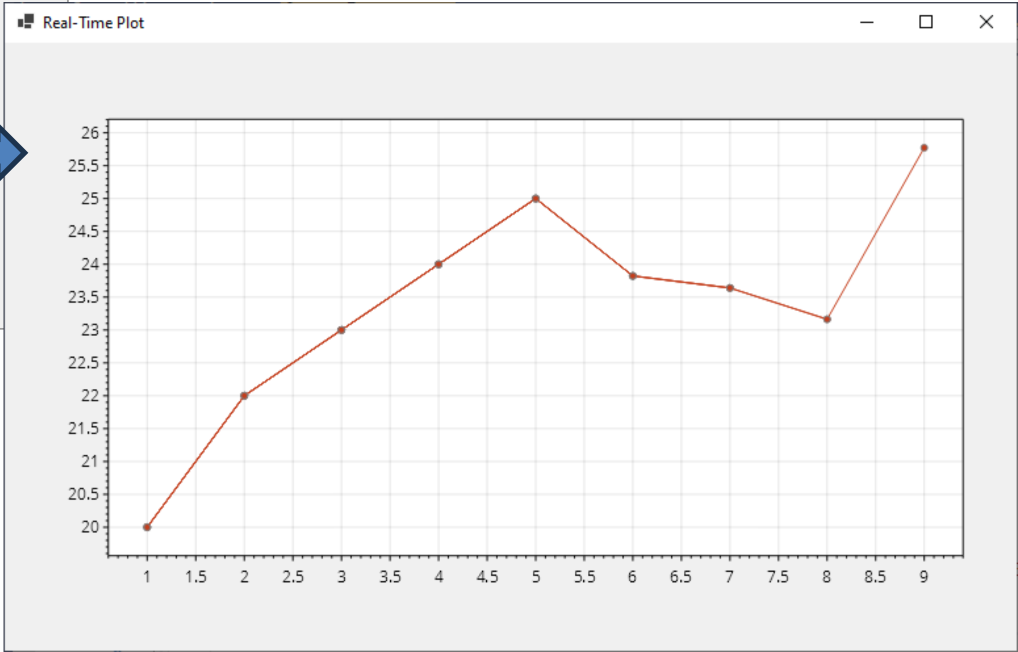
Hans-Petter Halvorsen

Table of Contents

Real-Time Scatter Plot



New values are added at the end using a Timer that updates the plot periodically



```
namespace RealTimePlot
{
    public partial class Form1 : Form
    {
        double[] dataX = new double[] { 1, 2, 3, 4, 5 };
        double[] dataY = new double[] { 20, 22, 23, 24, 25 };

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            CreateChart();

            timer1.Interval = 10000; //10 seconds
            timer1.Start();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            UpdateChart();
        }

        private void CreateChart()
        {
            formsPlot1.Plot.Add.Scatter(dataX, dataY, ScottPlot.Color.FromHex("C43E1C"));
            formsPlot1.Refresh();
        }

        private void UpdateChart()
        {
            Random rand = new Random();
            double newValue = rand.NextDouble() * 10 + 20; //Random Value between 20 and 30
            int k = dataX.Length + 1;
            dataX = dataX.Append(k).ToArray();
            dataY = dataY.Append(newValue).ToArray();
            formsPlot1.Plot.Add.Scatter(dataX, dataY, ScottPlot.Color.FromHex("C43E1C"));
            formsPlot1.Plot.Axes.AutoScale();
            formsPlot1.Refresh();
        }
    }
}
```

Alternative Solution with no Initial Data Set

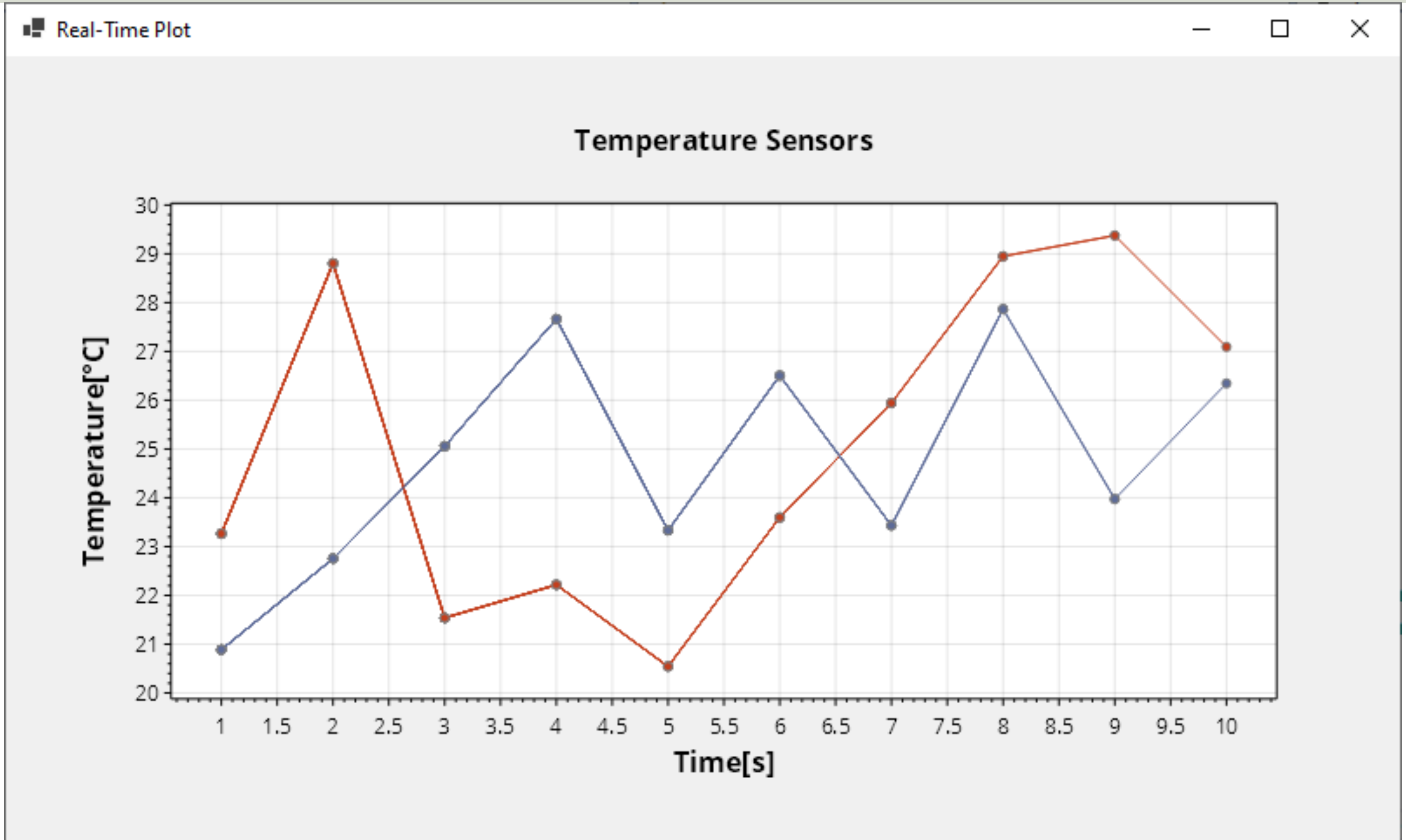
```
using ScottPlot.WinForms;

namespace RealTimePlot
{
   public partial class Form1 : Form
   {
      double[] dataX = new double[0];
      double[] dataY = new double[0];
      int k = 0;

      public Form1()
      {
         InitializeComponent();
         InitializeChart();
      }

      private void Form1_Load(object sender, EventArgs e)
      {
         timer1.Interval = 10000; //10 seconds
         timer1.Start();
      }

      private void timer1_Tick(object sender, EventArgs e)
      {
         UpdateChart();
      }

      private void InitializeChart()
      {
         formsPlot1.Plot.Axes.SetLimits(0, 1, 20, 30);
      }

      private void UpdateChart()
      {
         k++;
         Random rand = new Random();
         double newValue = rand.NextDouble() * 10 + 20; //Random Value between 20 and 30
         dataX = dataX.Append(k).ToArray();
         dataY = dataY.Append(newValue).ToArray();
         formsPlot1.Plot.Add.Scatter(dataX, dataY, ScottPlot.Color.FromHex("C43E1C"));
         formsPlot1.Plot.Axes.AutoScale();
         formsPlot1.Refresh();
      }
   }
}
```

Customize

You can add Xlabel, Ylabel, Title, etc

```
...

private void InitializeChart()
{
   formsPlot1.Plot.Axes.SetLimits(0, 1, 20, 30);
   formsPlot1.Plot.XLabel("Time[s]");
   formsPlot1.Plot.YLabel("Temperature[°C]");
   formsPlot1.Plot.Title("TC-01 Temperature Sensor");
   formsPlot1.Refresh();
}
...
```

Multi-Line Plot

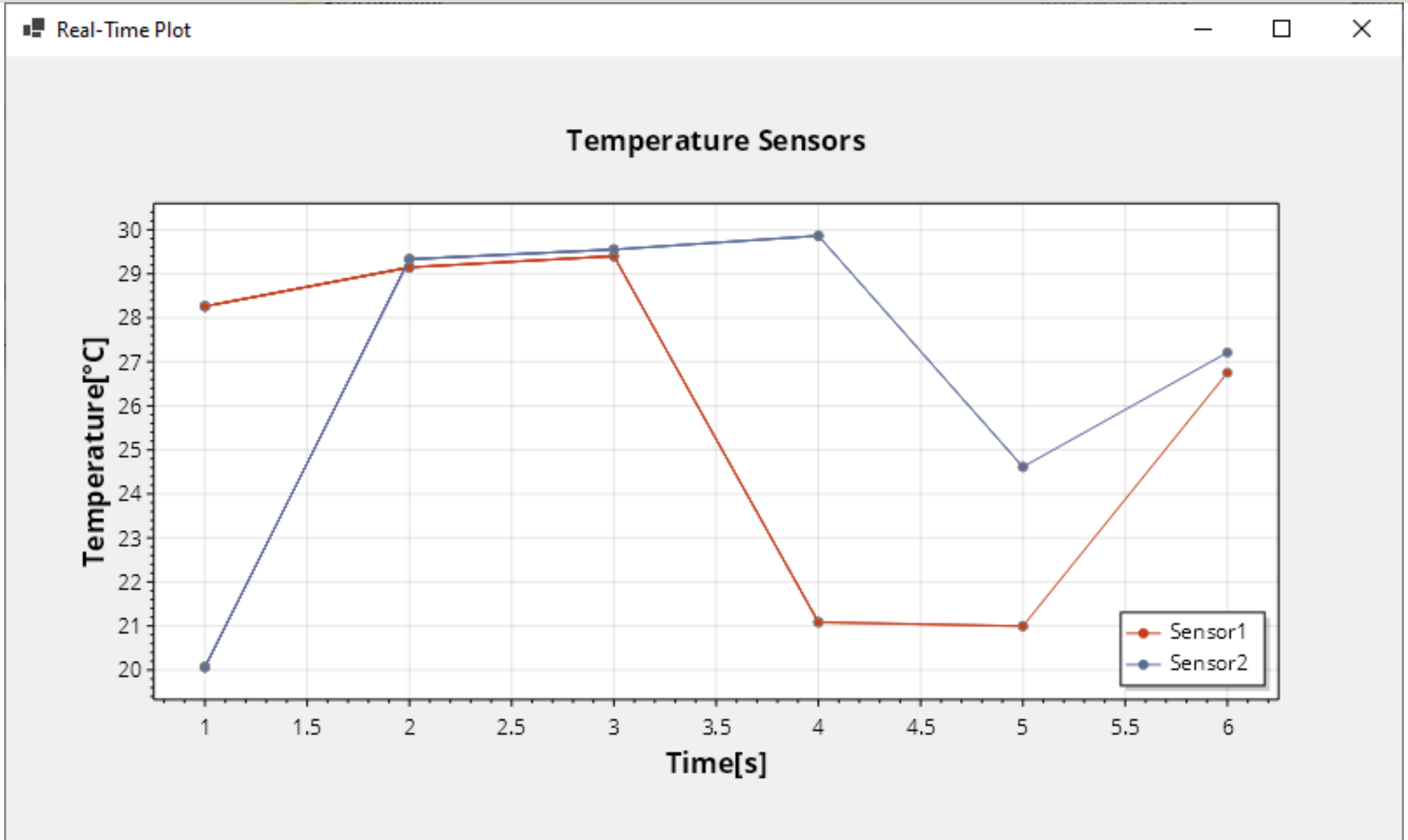


```
using ScottPlot.WinForms;

namespace RealTimePlot
{
    public partial class Form1 : Form
    {
        double[] dataX = new double[0];
        double[] dataY1 = new double[0];
        double[] dataY2 = new double[0];
        int k = 0;

        public Form1()
        {
            InitializeComponent();
            InitializeChart();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            timer1.Interval = 10000; //10 seconds
            timer1.Start();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            UpdateChart();
        }

        private void InitializeChart()
        {
            formsPlot1.Plot.Axes.SetLimits(0, 1, 20, 30);

            formsPlot1.Plot.XLabel("Time[s]");
            formsPlot1.Plot.YLabel("Temperature[°C]");
            formsPlot1.Plot.Title("Temperature Sensors");

            formsPlot1.Refresh();
        }

        private void UpdateChart()
        {
            k++;
            Random rand = new Random();
            double newValue1 = rand.NextDouble() * 10 + 20; //Random Value between 20 and 30
            double newValue2 = rand.NextDouble() * 10 + 20; //Random Value between 20 and 30

            dataX = dataX.Append(k).ToArray();
            dataY1 = dataY1.Append(newValue1).ToArray();
            dataY2 = dataY2.Append(newValue2).ToArray();

            formsPlot1.Plot.Add.Scatter(dataX, dataY1, ScottPlot.Color.FromHex("C43E1C"));
            formsPlot1.Plot.Add.Scatter(dataX, dataY2, ScottPlot.Color.FromHex("5D6B99"));

            formsPlot1.Plot.Axes.AutoScale();
            formsPlot1.Refresh();
        }
    }
}
```

Add Legend

```
...

private void InitializeChart()
{
        formsPlot1.Plot.Axes.SetLimits(0, 1, 20, 30);

        formsPlot1.Plot.XLabel("Time[s]");
        formsPlot1.Plot.YLabel("Temperature[°C]");
        formsPlot1.Plot.Title("Temperature Sensors");

        LegendItem item1 = new()
        {
            LineColor = ScottPlot.Color.FromHex("C43E1C"),
            MarkerColor = ScottPlot.Color.FromHex("C43E1C"),
            Label = "Sensor1"
        };

        LegendItem item2 = new()
        {
            LineColor = ScottPlot.Color.FromHex("5D6B99"),
            MarkerColor = ScottPlot.Color.FromHex("5D6B99"),
            Label = "Sensor2"
        };

        LegendItem[] items = {item1, item2};
        formsPlot1.Plot.ShowLegend(items);

        formsPlot1.Refresh();
}
```

Add Legend


Real-Time Plot
Temperature Sensors
Temperature[°C]
30
29
28
27
26
25
24
23
22
21
20
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6
Time[s]
Sensor1
Sensor2

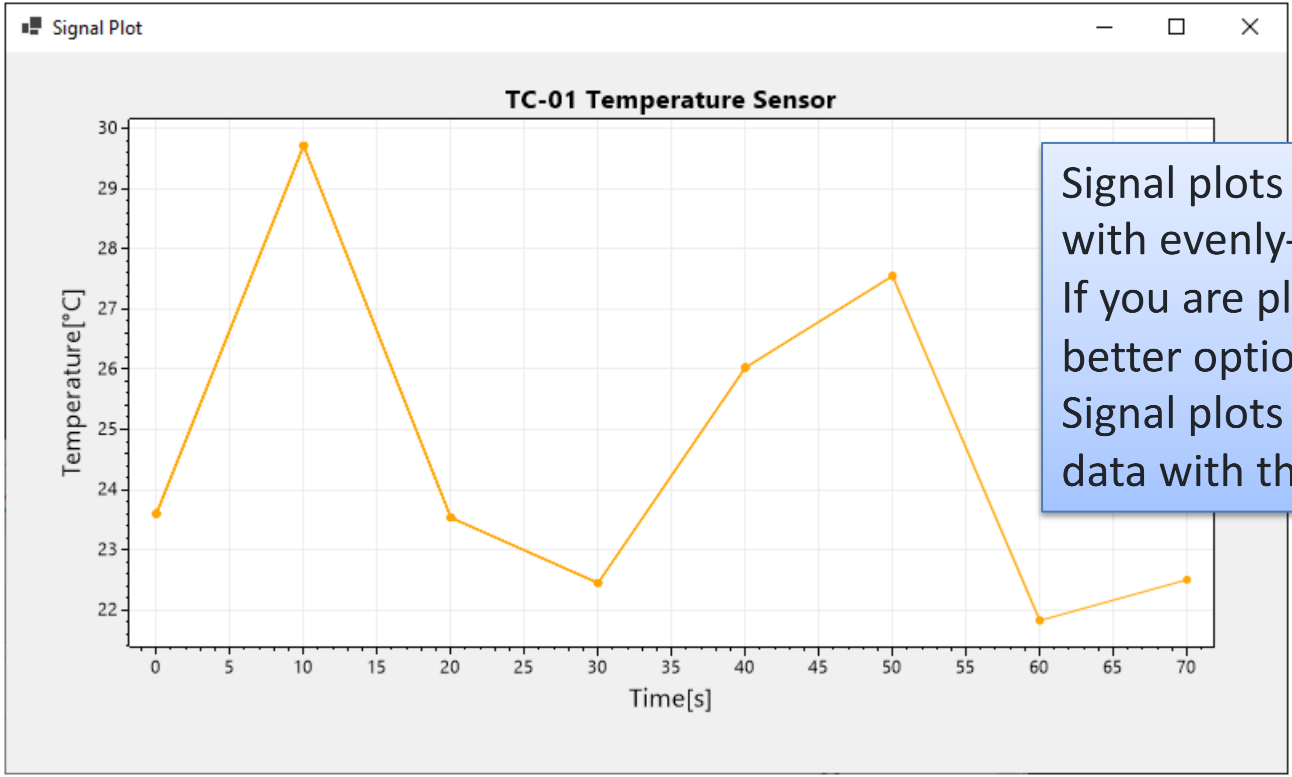
ScottPlot.NET
ScottPlot

Signal Plot

Hans-Petter Halvorsen

Table of Contents

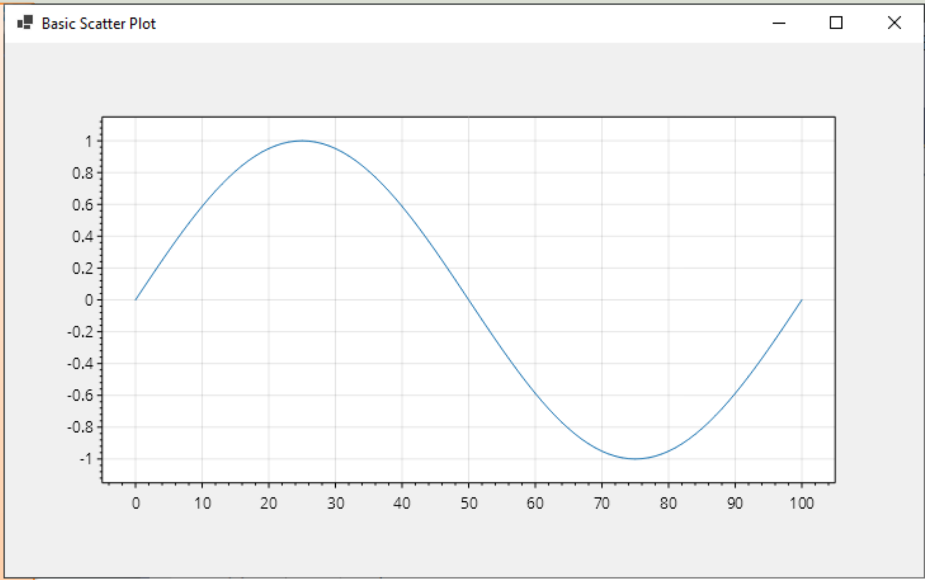
Signal Plot



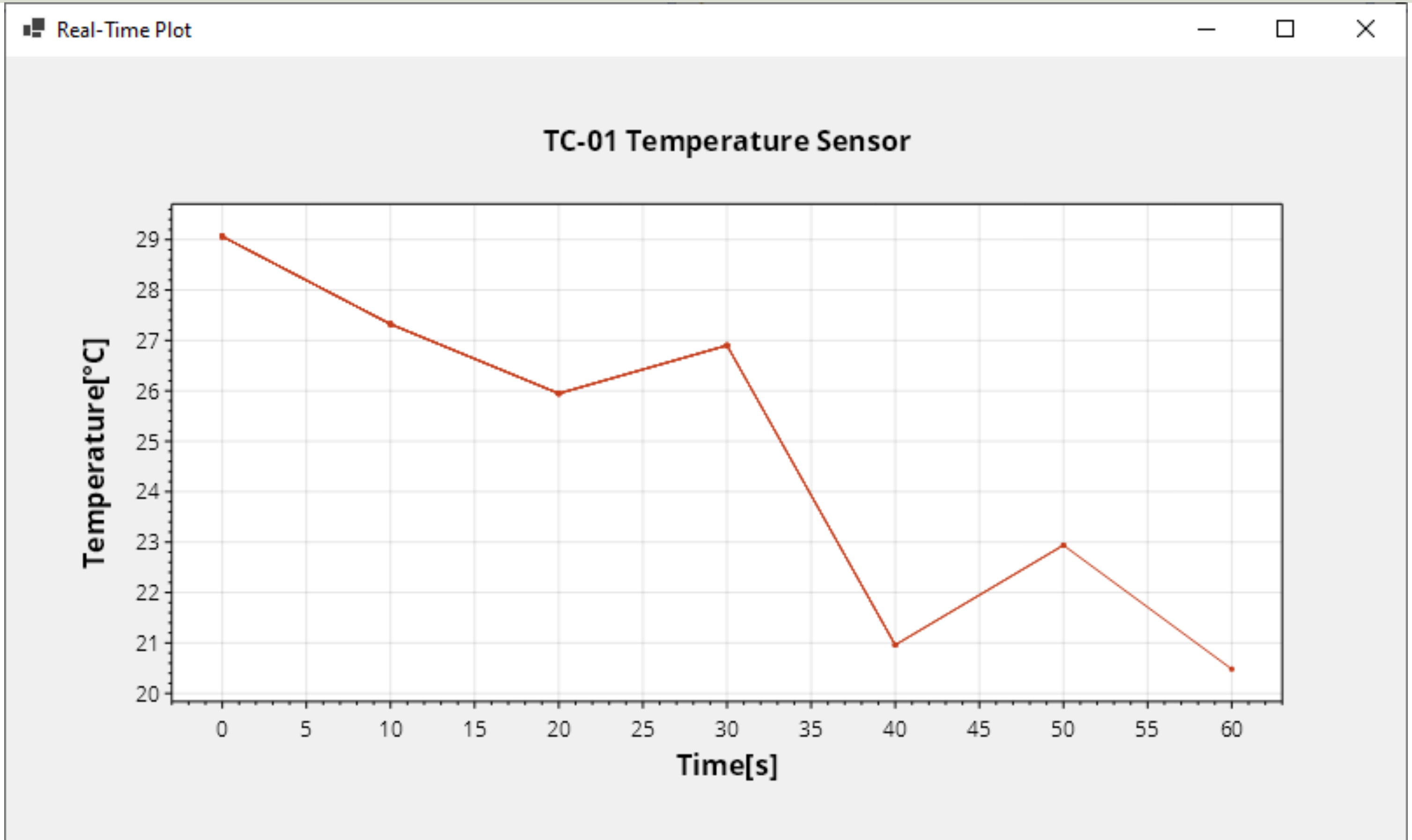
Signal plots are optimized to display data with evenly-spaced X values.
If you are plotting lots of Data, this is a better option than the Scatter Plot.
Signal plots are ideal for evenly-spaced data with thousands or millions of points.

Signal Plot

```
using ScottPlot;

namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            CreateChart();
        }

        private void CreateChart()
        {
            int count = 100;
            double[] data = Generate.Sin(count+1);

            formsPlot1.Plot.Add.Signal(data);
            formsPlot1.Refresh();
        }
    }
}
```



Signal Plot – Real Time Example



```
using ScottPlot.WinForms;

namespace RealTimePlot
{
    public partial class Form1 : Form
    {
        double[] data = new double[0];
        int sampleTime = 10; //[seconds]

        public Form1()
        {
            InitializeComponent();
            InitializeChart();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            timer1.Interval = sampleTime * 1000;
            timer1.Start();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            UpdateChart();
        }

        private void InitializeChart()
        {
            formsPlot1.Plot.Axes.SetLimits(0, 1, 20, 30);

            formsPlot1.Plot.XLabel("Time[s]");
            formsPlot1.Plot.YLabel("Temperature[°C]");
            formsPlot1.Plot.Title("TC-01 Temperature Sensor");
            formsPlot1.Refresh();
        }

        private void UpdateChart()
        {
            Random rand = new Random();
            double newValue = rand.NextDouble() * 10 + 20; //Random Value between 20 and 30
            data = data.Append(newValue).ToArray();

            formsPlot1.Plot.Add.Signal(data, sampleTime, ScottPlot.Color.FromHex("C43E1C"));

            formsPlot1.Plot.Axes.AutoScale();
            formsPlot1.Refresh();
        }
    }
}
```

https://www.halvorsen.blog

ScottPlot.NET ScottPlot SQL Server + ScottPlot

Plotting Data from Database using ScottPlot

ScottPlot is a free and open-source plotting library for .NET

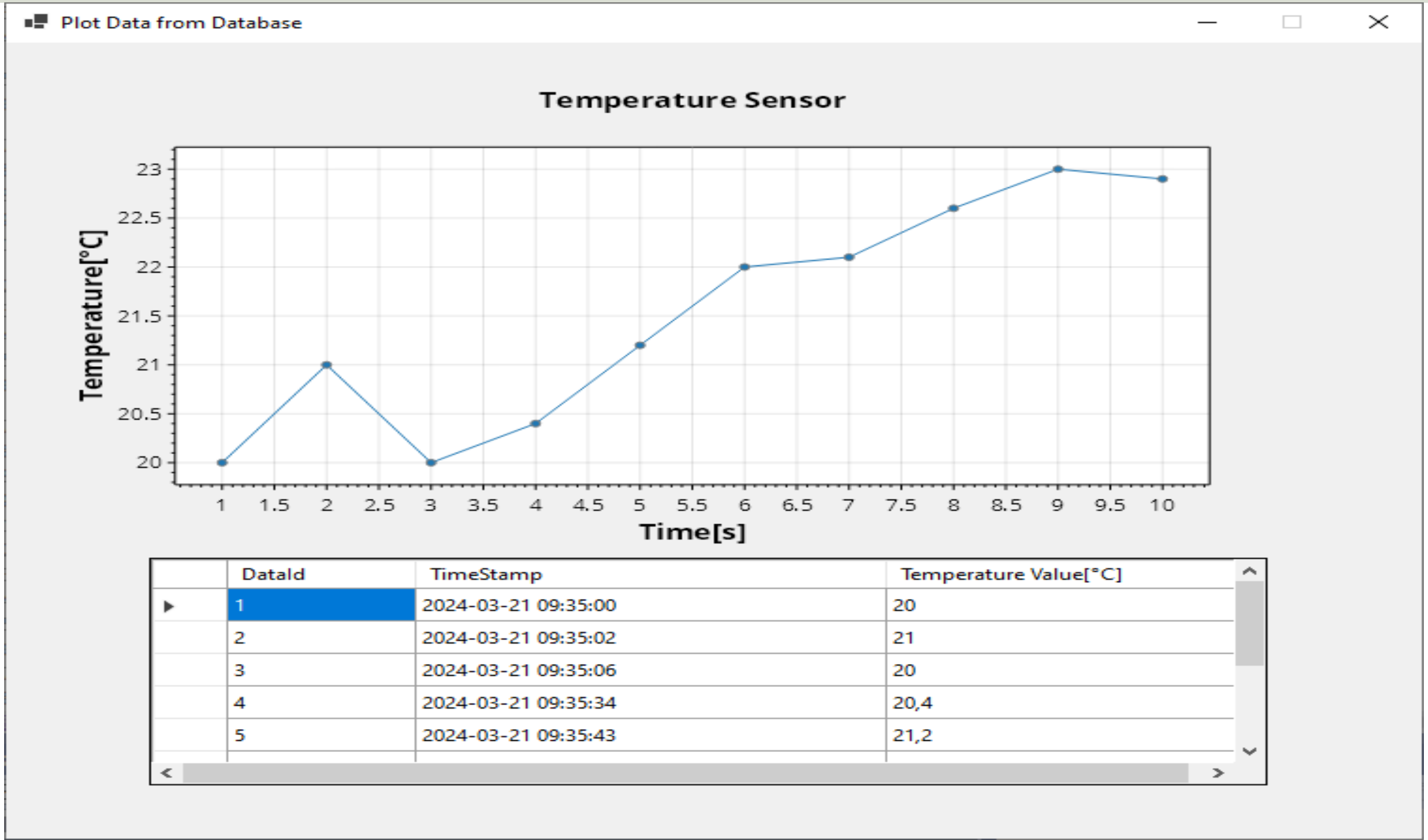
Hans-Petter Halvorsen

Table of Contents

Plotting Data from Database

- Here we will demonstrate how to retrieve data from a Database
- We will use SQL Server
- We will Put the Data into a GridView
- We will plot the Data using ScottPlot

Plotting Data from Database

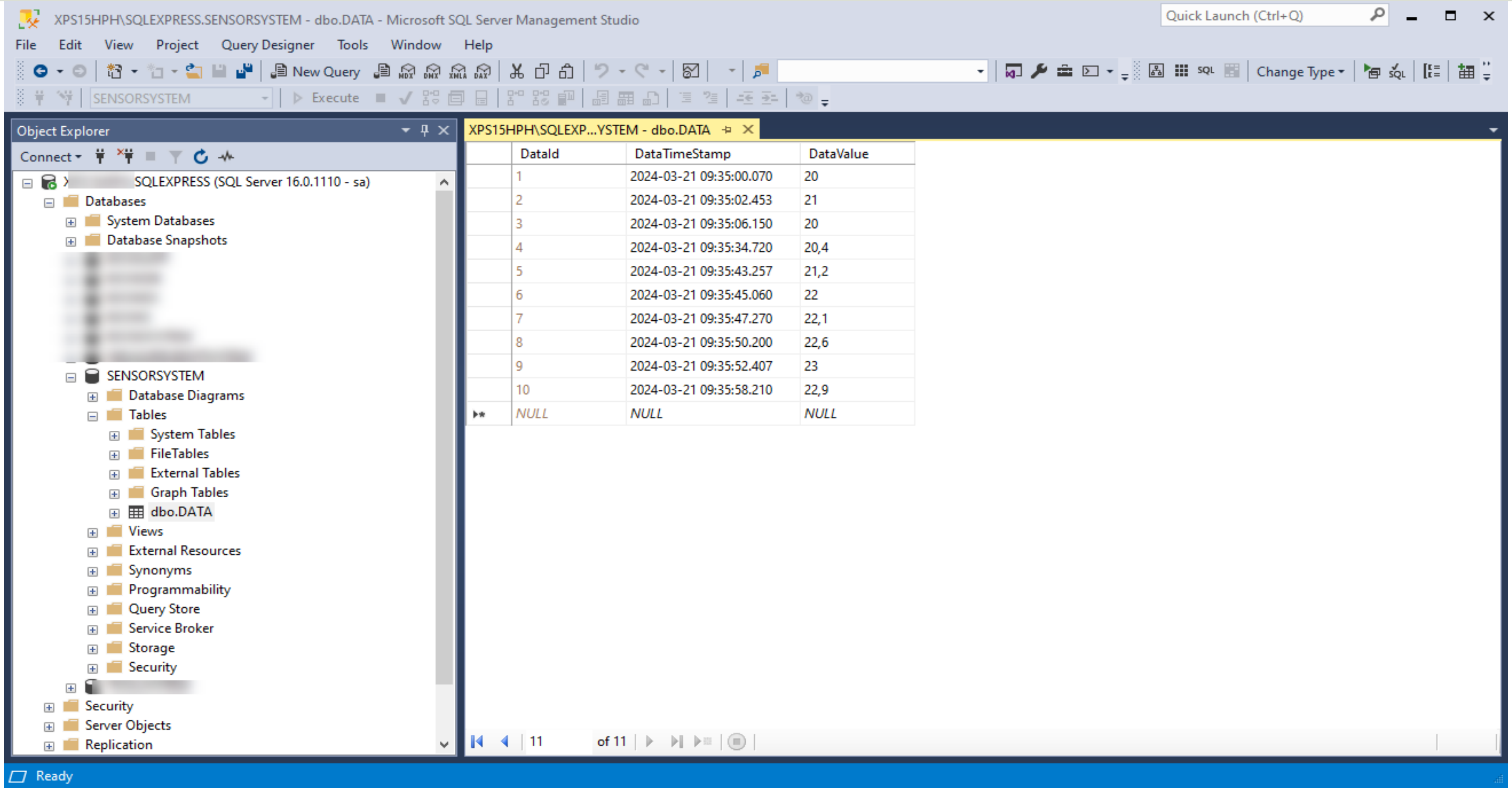


DataId	TimeStamp	Temperature Value[°C]
1	2024-03-21 09:35:00	20
2	2024-03-21 09:35:02	21
3	2024-03-21 09:35:06	20
4	2024-03-21 09:35:34	20,4
5	2024-03-21 09:35:43	21,2

Database Table

```
CREATE TABLE [DATA]
(
    [DataId]  int  NOT NULL  IDENTITY ( 1,1 ) Primary Key,
    [DataTimeStamp] datetime  NOT NULL DEFAULT GETDATE(),
    [DataValue]   float  NOT NULL
)
go
```

SQL Server

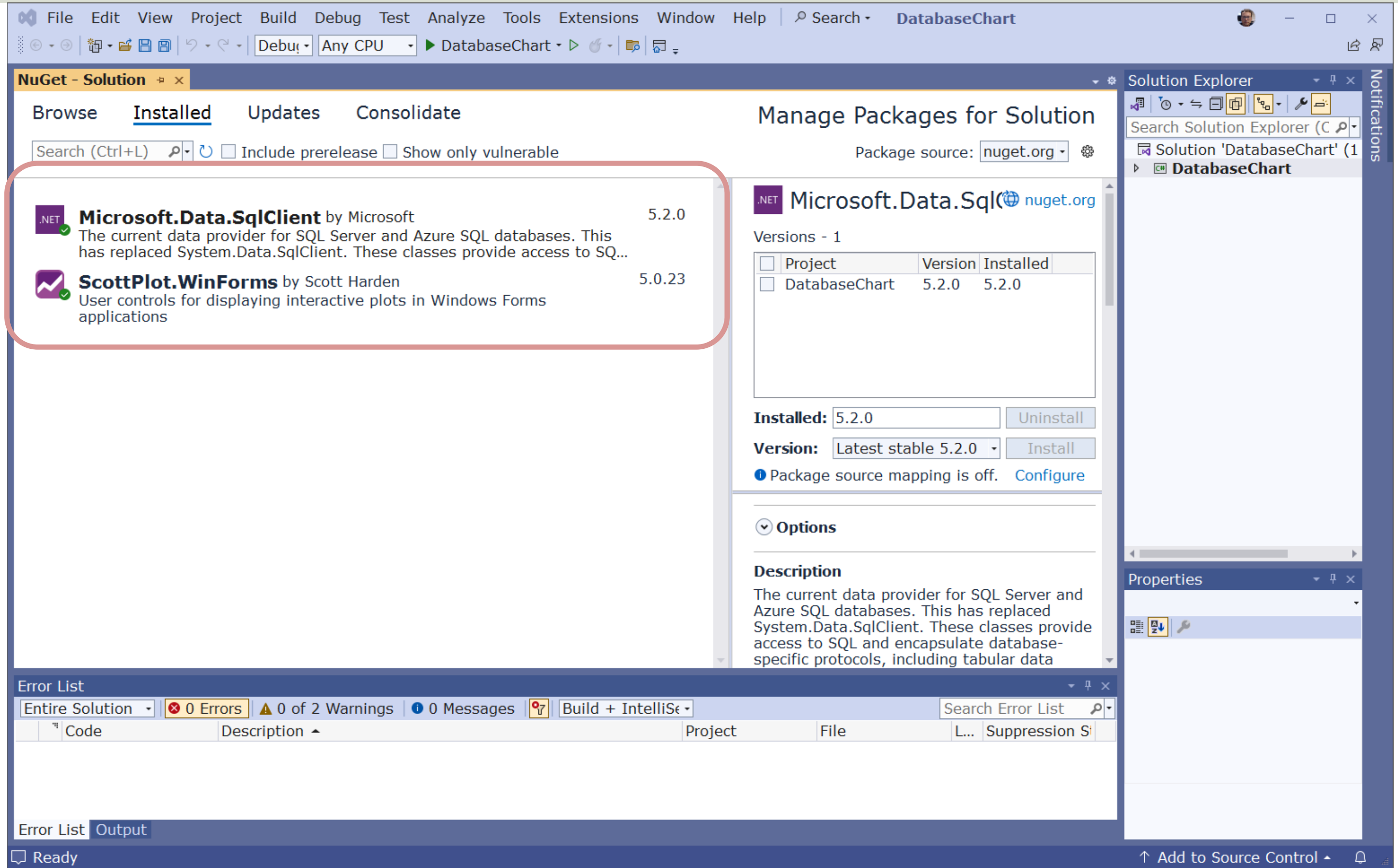


```
[redacted]SQLEXPRESS (SQL Server 16.0.1110 - sa)
  Databases
    System Databases
    Database Snapshots
    [redacted]
    SENSORSYSTEM
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
        dbo.DATA
      Views
      External Resources
      Synonyms
      Programmability
      Query Store
      Service Broker
      Storage
      Security
    [redacted]
  Security
  Server Objects
  Replication
```

	DataId	DataTimeStamp	DataValue
	1	2024-03-21 09:35:00.070	20
	2	2024-03-21 09:35:02.453	21
	3	2024-03-21 09:35:06.150	20
	4	2024-03-21 09:35:34.720	20,4
	5	2024-03-21 09:35:43.257	21,2
	6	2024-03-21 09:35:45.060	22
	7	2024-03-21 09:35:47.270	22,1
	8	2024-03-21 09:35:50.200	22,6
	9	2024-03-21 09:35:52.407	23
	10	2024-03-21 09:35:58.210	22,9
▸*	NULL	NULL	NULL

11 of 11

NuGet Packages



File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search DatabaseChart
Debug Any CPU DatabaseChart
NuGet - Solution
Browse Installed Updates Consolidate
Search (Ctrl+L) Include prerelease Show only vulnerable
Solution Explorer
Search Solution Explorer
Solution 'DatabaseChart' (1
DatabaseChart
Properties
Notifications
Error List Output
Ready
Add to Source Control

Manage Packages for Solution

Package source: nuget.org

Microsoft.Data.SqlClient by Microsoft 5.2.0
The current data provider for SQL Server and Azure SQL databases. This has replaced System.Data.SqlClient. These classes provide access to SQ...

ScottPlot.WinForms by Scott Harden 5.0.23
User controls for displaying interactive plots in Windows Forms applications

Microsoft.Data.Sql(nuget.org

Versions - 1

Project	Version	Installed
DatabaseChart	5.2.0	5.2.0

Installed: 5.2.0 Uninstall

Version: Latest stable 5.2.0 Install

Package source mapping is off. Configure

Options

Description

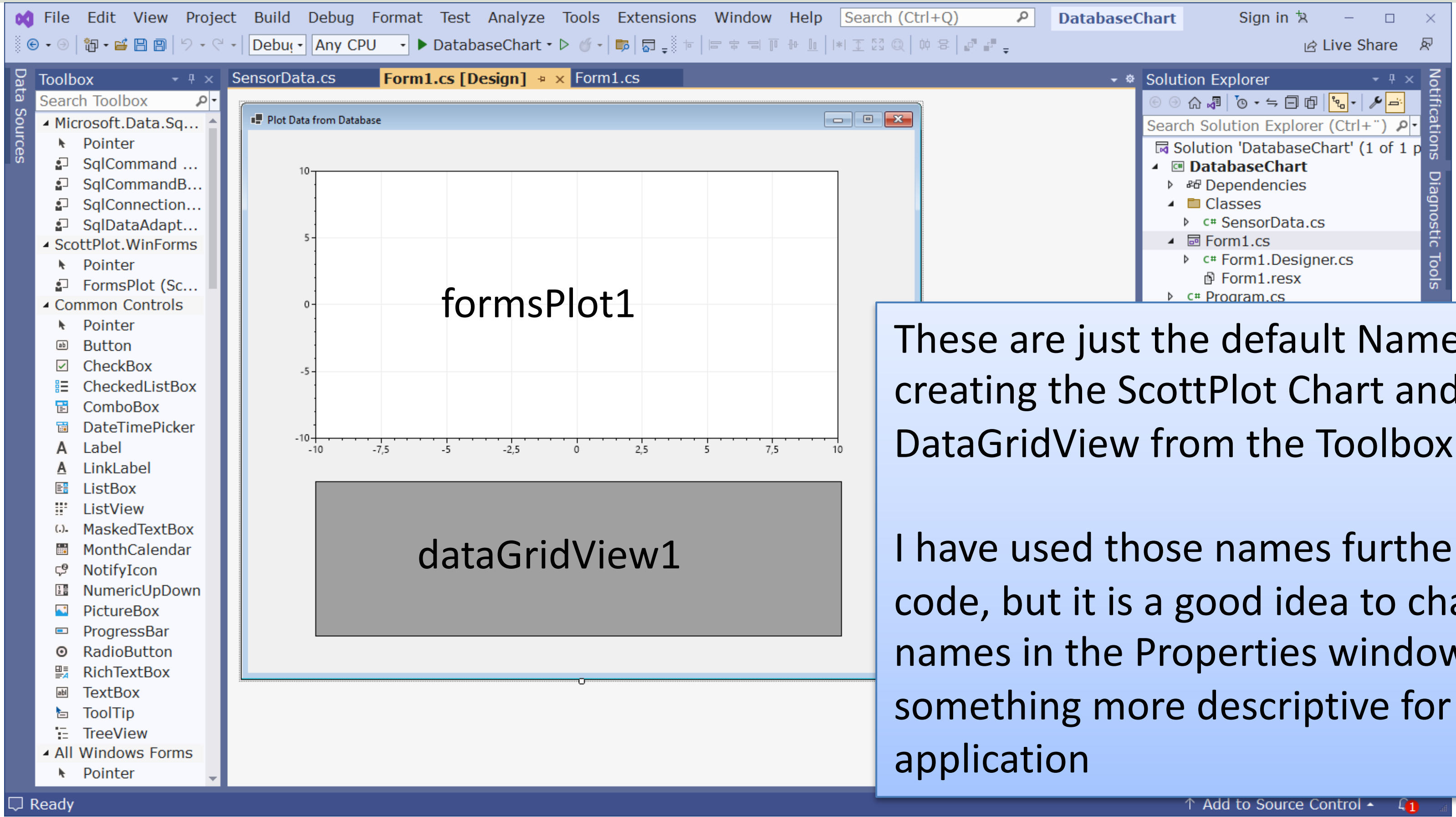
The current data provider for SQL Server and Azure SQL databases. This has replaced System.Data.SqlClient. These classes provide access to SQL and encapsulate database-specific protocols, including tabular data

Error List

Entire Solution | 0 Errors | 0 of 2 Warnings | 0 Messages | Build + IntelliSe | Search Error List

Code	Description	Project	File	L...	Suppression S

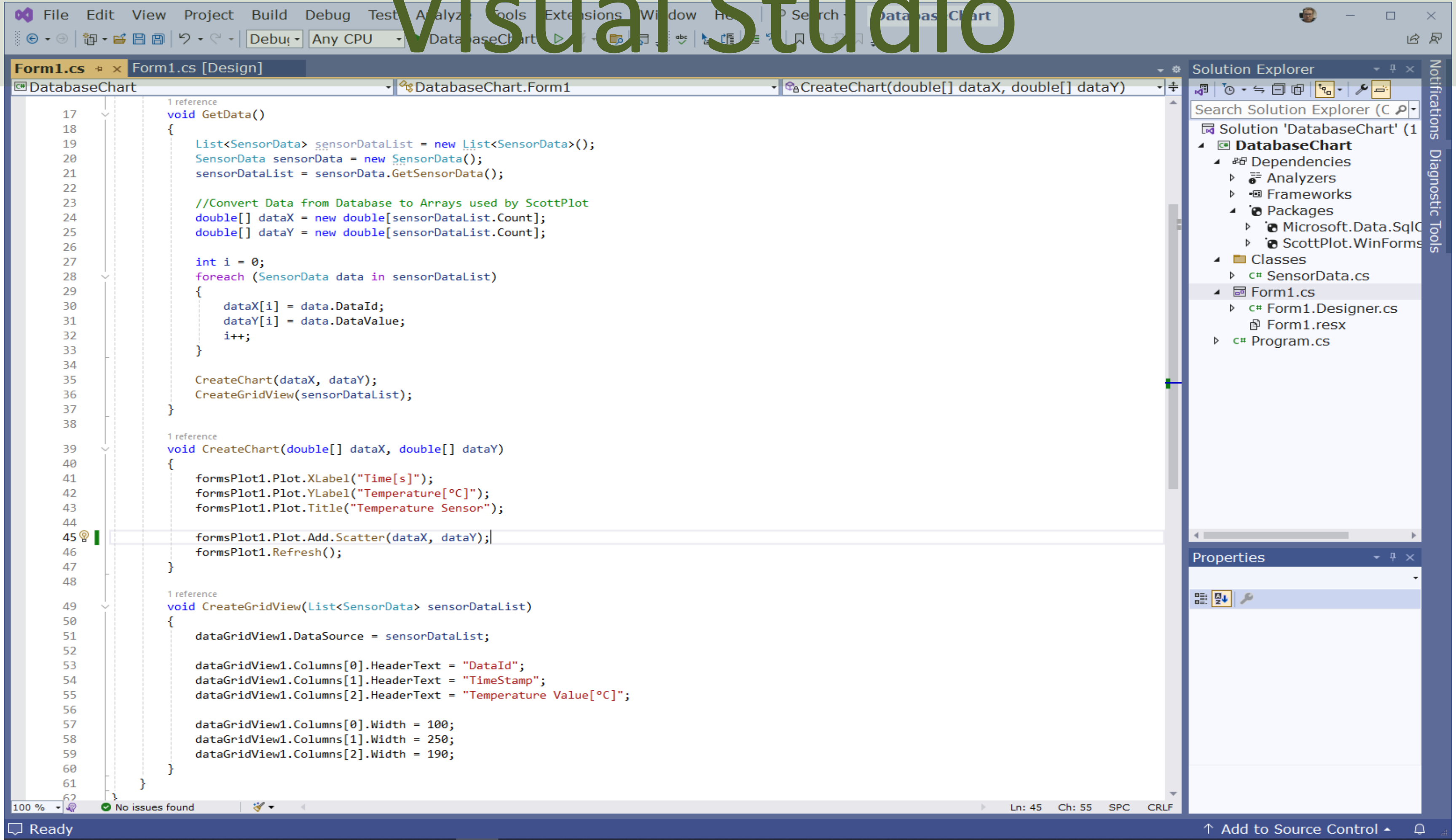
Visual Studio



These are just the default Names when creating the ScottPlot Chart and the DataGridView from the Toolbox

I have used those names further in the code, but it is a good idea to change their names in the Properties window to something more descriptive for your application

Visual Studio



```
1 reference
void GetData()
{
    List<SensorData> sensorDataList = new List<SensorData>();
    SensorData sensorData = new SensorData();
    sensorDataList = sensorData.GetSensorData();

    //Convert Data from Database to Arrays used by ScottPlot
    double[] dataX = new double[sensorDataList.Count];
    double[] dataY = new double[sensorDataList.Count];

    int i = 0;
    foreach (SensorData data in sensorDataList)
    {
        dataX[i] = data.DataId;
        dataY[i] = data.DataValue;
        i++;
    }

    CreateChart(dataX, dataY);
    CreateGridView(sensorDataList);
}

1 reference
void CreateChart(double[] dataX, double[] dataY)
{
    formsPlot1.Plot.XLabel("Time[s]");
    formsPlot1.Plot.YLabel("Temperature[°C]");
    formsPlot1.Plot.Title("Temperature Sensor");

    formsPlot1.Plot.Add.Scatter(dataX, dataY);
    formsPlot1.Refresh();
}

1 reference
void CreateGridView(List<SensorData> sensorDataList)
{
    dataGridView1.DataSource = sensorDataList;

    dataGridView1.Columns[0].HeaderText = "DataId";
    dataGridView1.Columns[1].HeaderText = "TimeStamp";
    dataGridView1.Columns[2].HeaderText = "Temperature Value[°C]";

    dataGridView1.Columns[0].Width = 100;
    dataGridView1.Columns[1].Width = 250;
    dataGridView1.Columns[2].Width = 190;
}
}
}
```

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search DatabaseChart
Debug Any CPU DatabaseChart
Form1.cs Form1.cs [Design]
DatabaseChart DatabaseChart.Form1 CreateChart(double[] dataX, double[] dataY)
Solution Explorer
Search Solution Explorer (C
Solution 'DatabaseChart' (1
DatabaseChart
Dependencies
Analyzers
Frameworks
Packages
Microsoft.Data.SqlC
ScottPlot.WinForms
Classes
SensorData.cs
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Properties
Notifications Diagnostic Tools
100 % No issues found Ln: 45 Ch: 55 SPC CRLF
Ready
Add to Source Control

SensorData Class (Filename: SensorData.cs)

```csharp
using Microsoft.Data.SqlClient;

namespace DatabaseChart.Classes
{
  internal class SensorData
  {
    public int DataId { get; set; }
    public string? DataTimeStamp { get; set; }
    public double DataValue { get; set; }

    public List<SensorData> GetSensorData()
    {
      string connectionString = "Data Source=xxx\\SQLEXPRESS;Initial Catalog=SENSORSYSTEM;Integrated Security=True; TrustServerCertificate=True";

      List<SensorData> sensorDataList = new List<SensorData>();
      SqlConnection con = new SqlConnection(connectionString);

      string selectSQL = "SELECT DataId, FORMAT(DataTimeStamp, 'MM.dd HH:mm') AS DataTimeStamp, DataValue FROM DATA";
      con.Open();
      SqlCommand cmd = new SqlCommand(selectSQL, con);
      SqlDataReader dr = cmd.ExecuteReader();
      if (dr != null)
      {
        while (dr.Read())
        {
          SensorData sensorData = new SensorData();

          sensorData.DataId = Convert.ToInt32(dr["DataId"]);
          sensorData.DataTimeStamp = dr["DataTimeStamp"].ToString();
          sensorData.DataValue = Convert.ToDouble(dr["DataValue"]);
          sensorDataList.Add(sensorData);
        }
      }
      return sensorDataList;
    }
  }
}
```

Form1.cs

```
using DatabaseChart.Classes;

namespace DatabaseChart
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            GetData();
        }
    }
}
```

GetData() Method

```
void GetData()
{
    List<SensorData> sensorDataList = new List<SensorData>();
    SensorData sensorData = new SensorData();
    sensorDataList = sensorData.GetSensorData();

    //Convert Data from Database to Arrays used by ScottPlot
    double[] dataX = new double[sensorDataList.Count];
    double[] dataY = new double[sensorDataList.Count];

    int i = 0;
    foreach (SensorData data in sensorDataList)
    {
        dataX[i] = data.DataId;
        dataY[i] = data.DataValue;
        i++;
    }

    CreateChart(dataX, dataY);
    CreateGridView(sensorDataList);
}
```

CreateChart() Method

```
void CreateChart(double[] dataX, double[] dataY)
{
    formsPlot1.Plot.XLabel("Time[s]");
    formsPlot1.Plot.YLabel("Temperature[°C]");
    formsPlot1.Plot.Title("Temperature Sensor");

    formsPlot1.Plot.Add.Scatter(dataX, dataY);
    formsPlot1.Refresh();
}
```

CreateGrideView() Method

```
void CreateGridView(List<SensorData> sensorDataList)
{
    dataGridView1.DataSource = sensorDataList;

    dataGridView1.Columns[0].HeaderText = "DataId";
    dataGridView1.Columns[1].HeaderText = "TimeStamp [MM.dd HH:mm]";
    dataGridView1.Columns[2].HeaderText = "Temperature Value[°C]";
    dataGridView1.Columns[0].Width = 100;
    dataGridView1.Columns[1].Width = 250;
    dataGridView1.Columns[2].Width = 190;
}
```

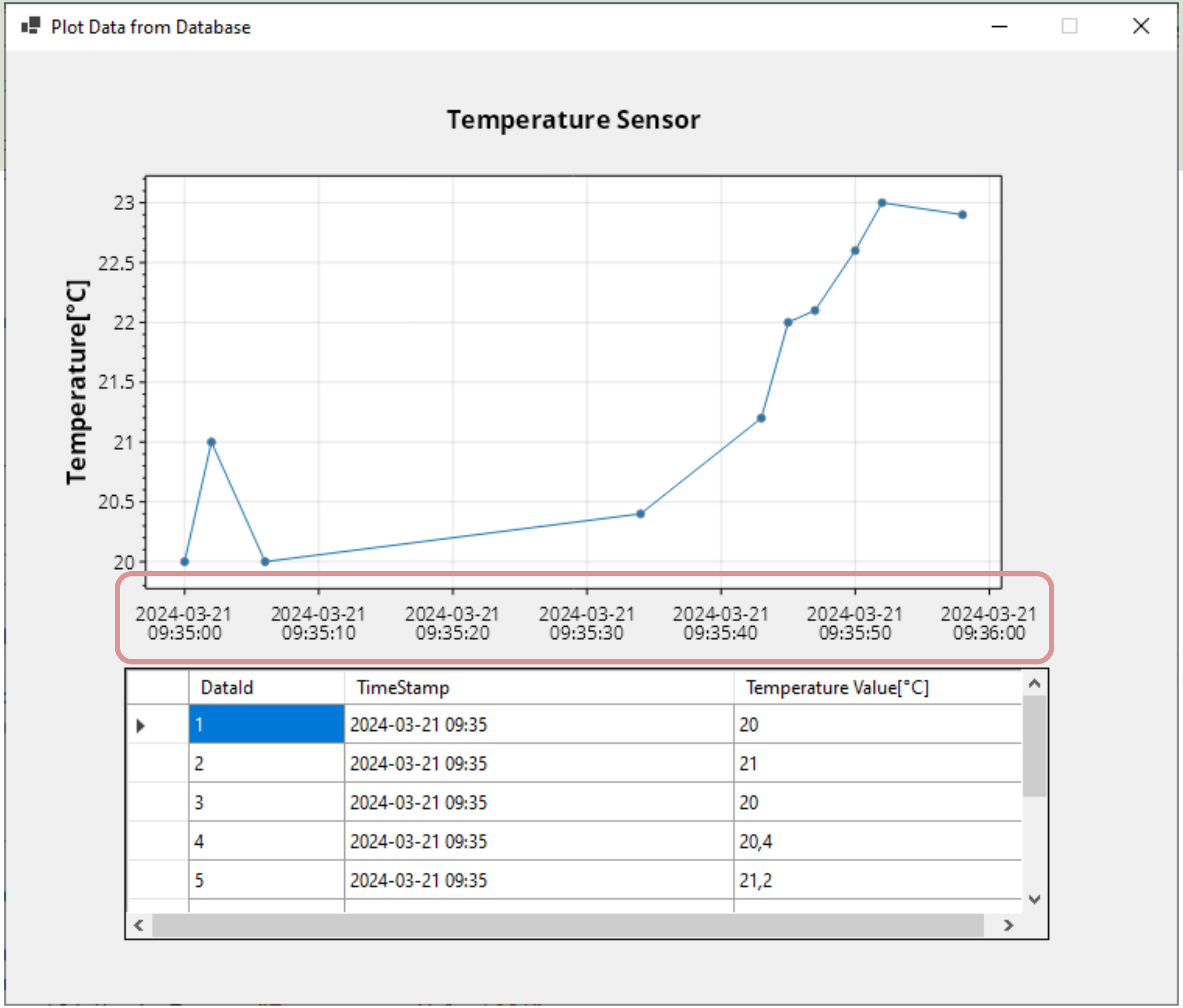
Plotting DateTime Data

SQL Server + ScottPlot

Hans-Petter Halvorsen

Table of Contents

DateTime



DataId	TimeStamp	Temperature Value[°C]
1	2024-03-21 09:35	20
2	2024-03-21 09:35	21
3	2024-03-21 09:35	20
4	2024-03-21 09:35	20,4
5	2024-03-21 09:35	21,2

SensorData Class (Filename: SensorData.cs)

```
using Microsoft.Data.SqlClient;

namespace DatabaseChart.Classes
{
  internal class SensorData
  {
    public int DataId { get; set; }
    public DateTime DataTimeStamp { get; set; }
    public double DataValue { get; set; }

    public List<SensorData> GetSensorData()
    {
      string connectionString = "Data Source=xxx\\SQLEXPRESS;Initial Catalog=SENSORSYSTEM;Integrated Security=True; TrustServerCertificate=True";

      List<SensorData> sensorDataList = new List<SensorData>();
      SqlConnection con = new SqlConnection(connectionString);

      string selectSQL = "SELECT DataId, FORMAT(DataTimeStamp, 'yyyy-MM-dd HH:mm:ss') AS DataTimeStamp, DataValue FROM DATA";
      con.Open();
      SqlCommand cmd = new SqlCommand(selectSQL, con);
      SqlDataReader dr = cmd.ExecuteReader();
      if (dr != null)
      {
        while (dr.Read())
        {
          SensorData sensorData = new SensorData();

          sensorData.DataId = Convert.ToInt32(dr["DataId"]);
          sensorData.DataTimeStamp = Convert.ToDateTime(dr["DataTimeStamp"]);
          sensorData.DataValue = Convert.ToDouble(dr["DataValue"]);
          sensorDataList.Add(sensorData);
        }
      }
      return sensorDataList;
    }
  }
}
```

```
void GetData()
{
    List<SensorData> sensorDataList = new List<SensorData>();
    SensorData sensorData = new SensorData();
    sensorDataList = sensorData.GetSensorData();

    //Convert Data from Database to Arrays used by ScottPlot
    double[] dataX = new double[sensorDataList.Count];
    double[] dataY = new double[sensorDataList.Count];

    int i = 0;
    foreach (SensorData data in sensorDataList)
    {
        dataX[i] = data.DataTimeStamp.ToOADate();
        dataY[i] = data.DataValue;
        i++;
    }

    CreateChart(dataX, dataY);
    CreateGridView(sensorDataList);
}
```

```
void CreateChart(double[] dataX, double[] dataY)
{
    formsPlot1.Plot.XLabel("Time[s]");
    formsPlot1.Plot.YLabel("Temperature[°C]");
    formsPlot1.Plot.Title("Temperature Sensor");

    formsPlot1.Plot.AddScatter(dataX, dataY);
    formsPlot1.Plot.Axes.DateTimeTicksBottom();
    formsPlot1.Refresh();
}
```

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog