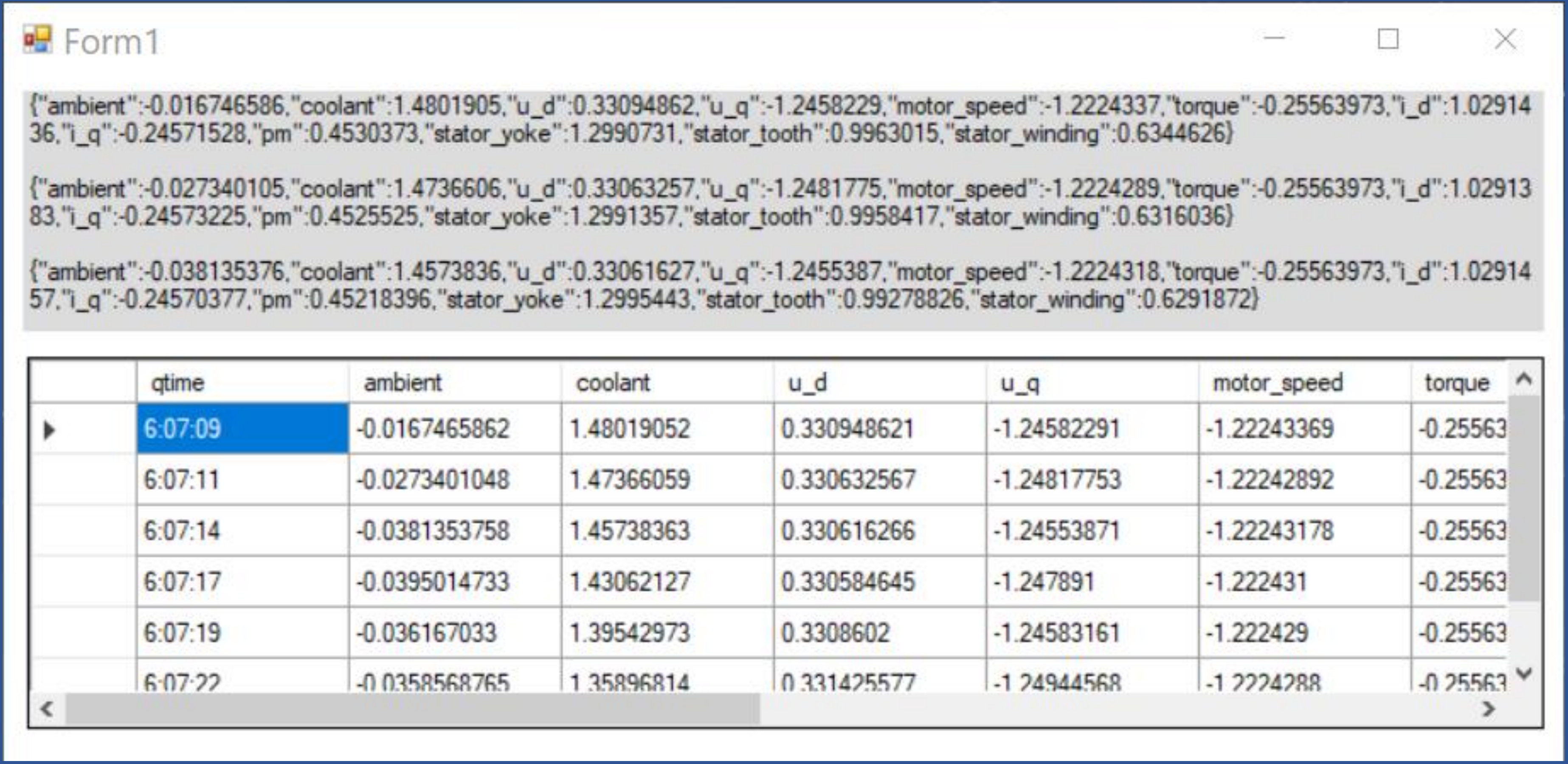
WinForm backend DataGridView

Create WinForm show DataGridView



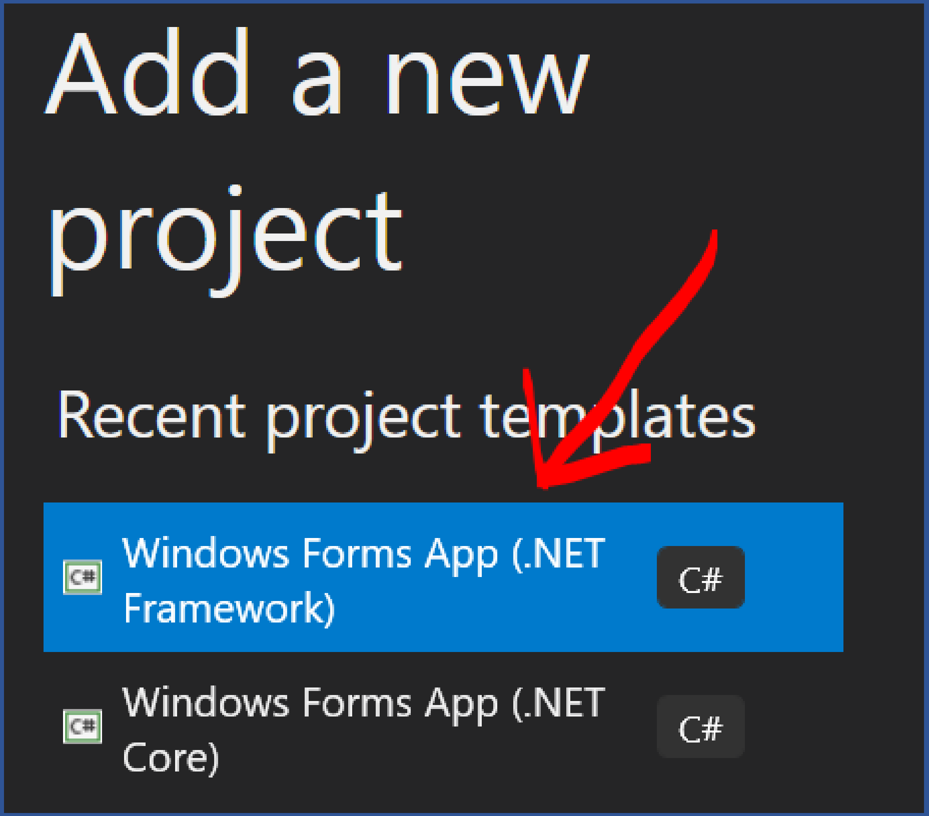
```
{"ambient":-0.016746586,"coolant":1.4801905,"u_d":0.33094862,"u_q":-1.2458229,"motor_speed":-1.2224337,"torque":-0.25563973,"i_d":1.02914
36,"i_q":-0.24571528,"pm":0.4530373,"stator_yoke":1.2990731,"stator_tooth":0.9963015,"stator_winding":0.6344626}

{"ambient":-0.027340105,"coolant":1.4736606,"u_d":0.33063257,"u_q":-1.2481775,"motor_speed":-1.2224289,"torque":-0.25563973,"i_d":1.02913
83,"i_q":-0.24573225,"pm":0.4525525,"stator_yoke":1.2991357,"stator_tooth":0.9958417,"stator_winding":0.6316036}

{"ambient":-0.038135376,"coolant":1.4573836,"u_d":0.33061627,"u_q":-1.2455387,"motor_speed":-1.2224318,"torque":-0.25563973,"i_d":1.02914
57,"i_q":-0.24570377,"pm":0.45218396,"stator_yoke":1.2995443,"stator_tooth":0.99278826,"stator_winding":0.6291872}
```

	qtime	ambient	coolant	u_d	u_q	motor_speed	torque
▶	6:07:09	-0.0167465862	1.48019052	0.330948621	-1.24582291	-1.22243369	-0.25563
	6:07:11	-0.0273401048	1.47366059	0.330632567	-1.24817753	-1.22242892	-0.25563
	6:07:14	-0.0381353758	1.45738363	0.330616266	-1.24553871	-1.22243178	-0.25563
	6:07:17	-0.0395014733	1.43062127	0.330584645	-1.247891	-1.222431	-0.25563
	6:07:19	-0.036167033	1.39542973	0.3308602	-1.24583161	-1.222429	-0.25563
	6:07:22	-0.0358568765	1.35896814	0.331425577	-1.24944568	-1.2224288	-0.25563

Open Visual Studio / Create C# .NET Framework WinForm



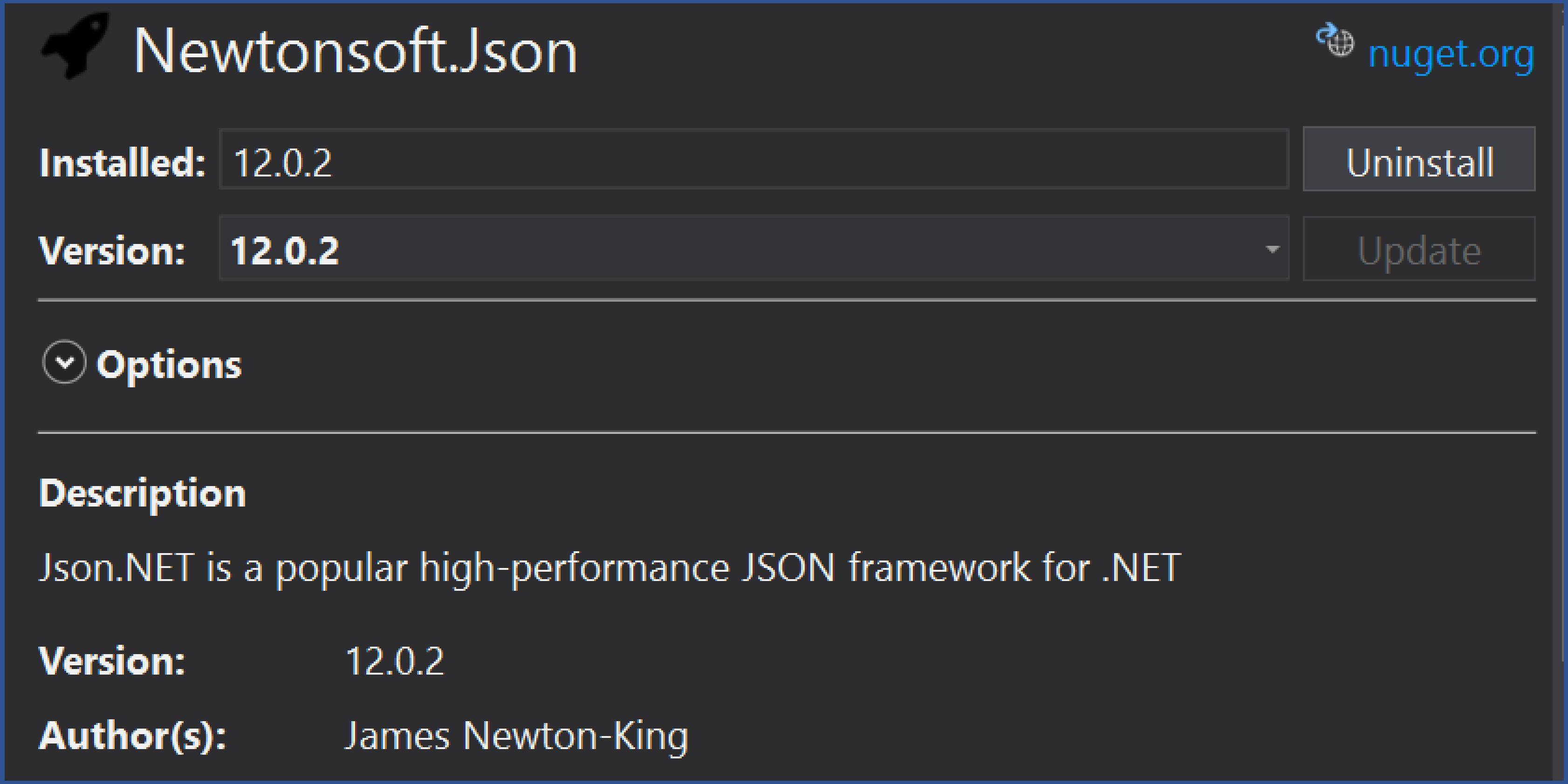
Add a new project

Recent project templates

Windows Forms App (.NET Framework) C#

Windows Forms App (.NET Core) C#

Add Newtonsoft.Json


Newtonsoft.Json

nuget.org

Installed: 12.0.2

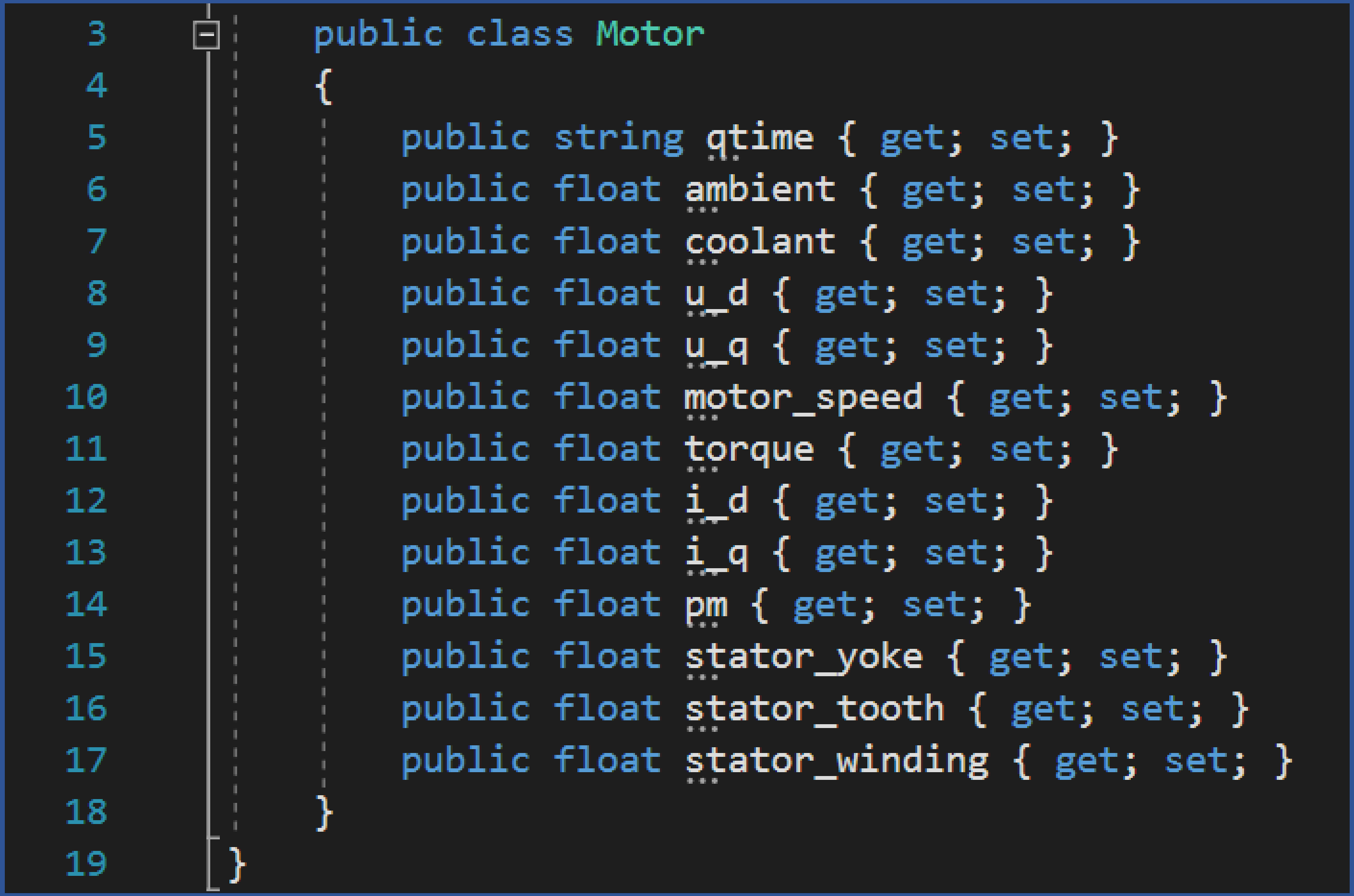
Version: **12.0.2**

Description

Json.NET is a popular high-performance JSON framework for .NET

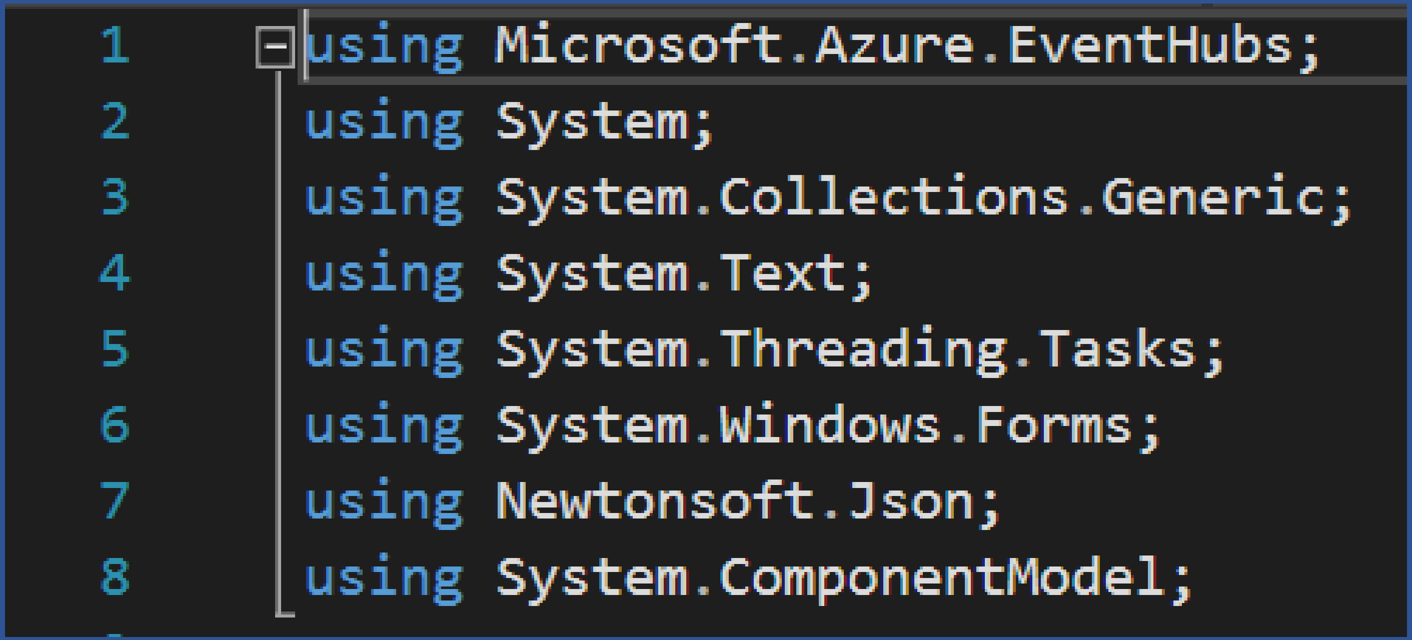
Version:	12.0.2
Author(s):	James Newton-King

Add new class file



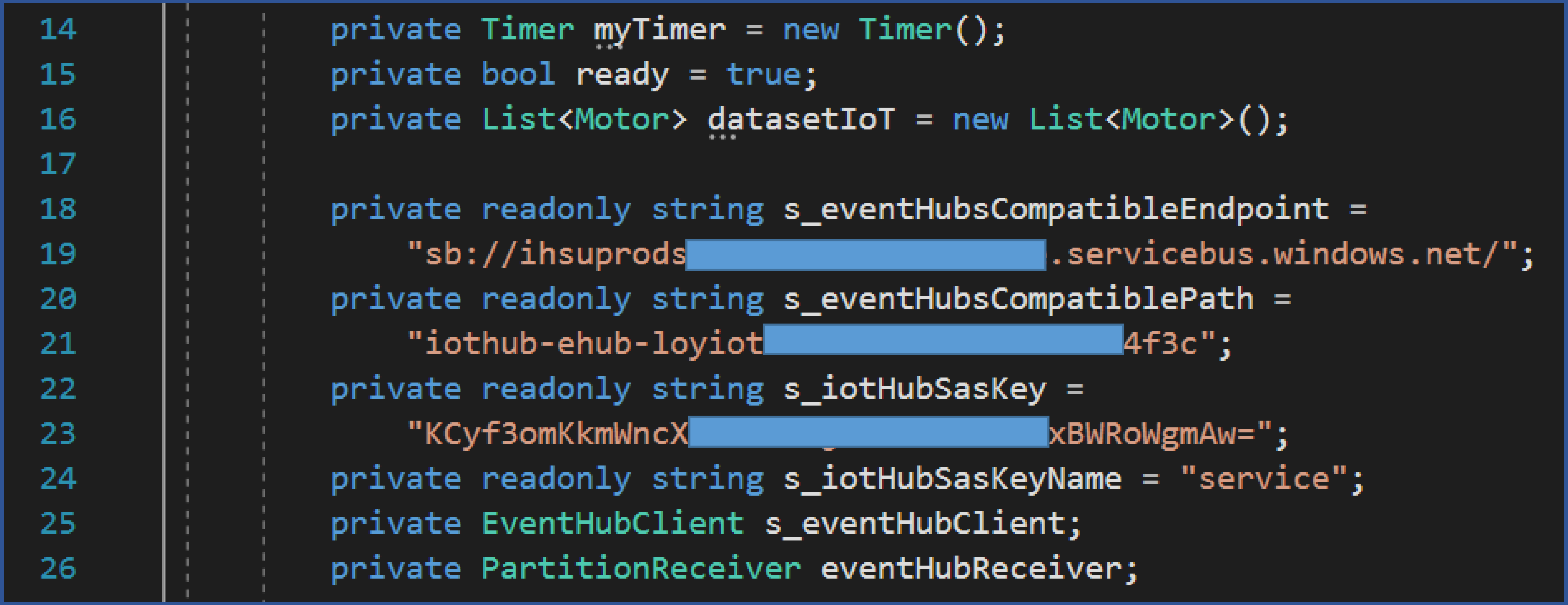
```
    public class Motor
    {
        public string qtime { get; set; }
        public float ambient { get; set; }
        public float coolant { get; set; }
        public float u_d { get; set; }
        public float u_q { get; set; }
        public float motor_speed { get; set; }
        public float torque { get; set; }
        public float i_d { get; set; }
        public float i_q { get; set; }
        public float pm { get; set; }
        public float stator_yoke { get; set; }
        public float stator_tooth { get; set; }
        public float stator_winding { get; set; }
    }
}
```

Add Namespace to Form1



```
using Microsoft.Azure.EventHubs;
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Newtonsoft.Json;
using System.ComponentModel;
```

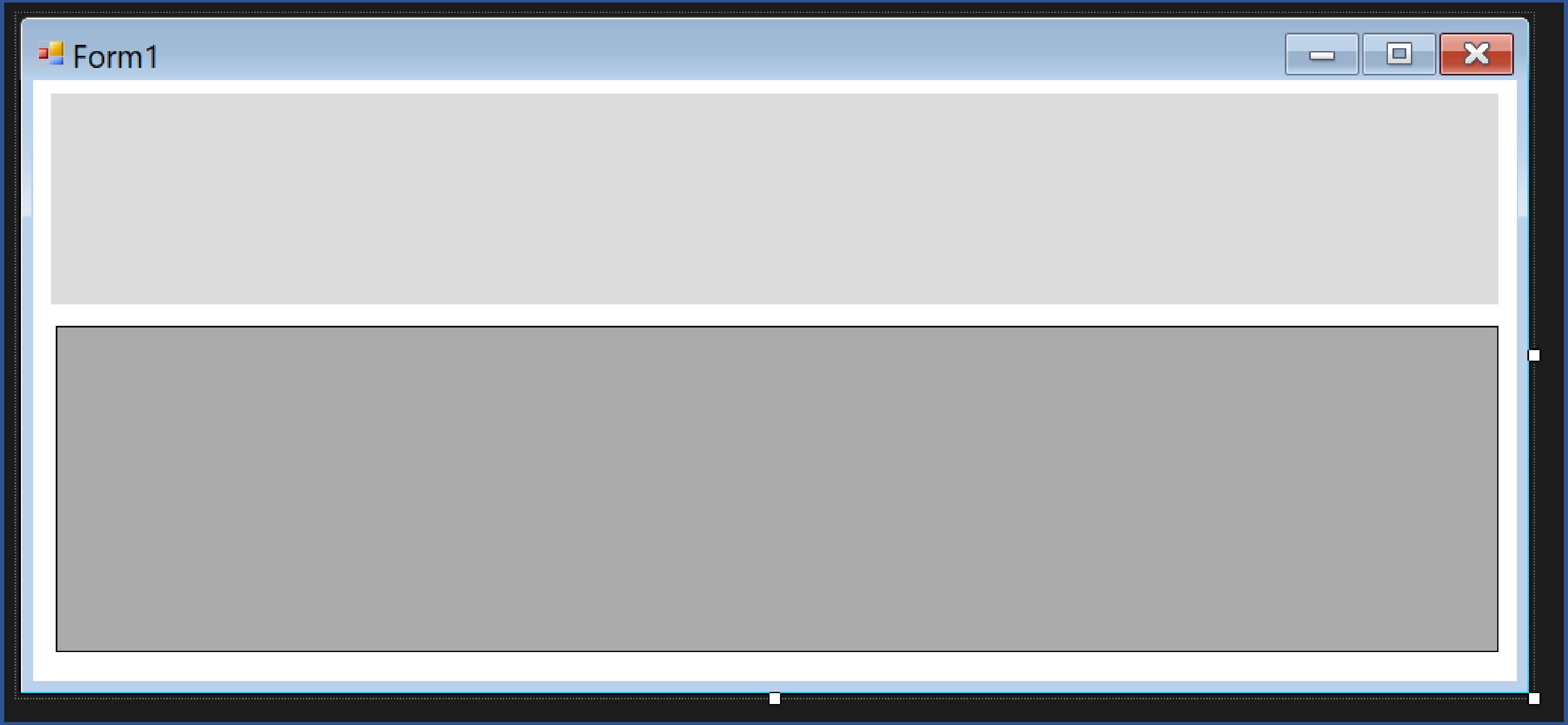
Add class members



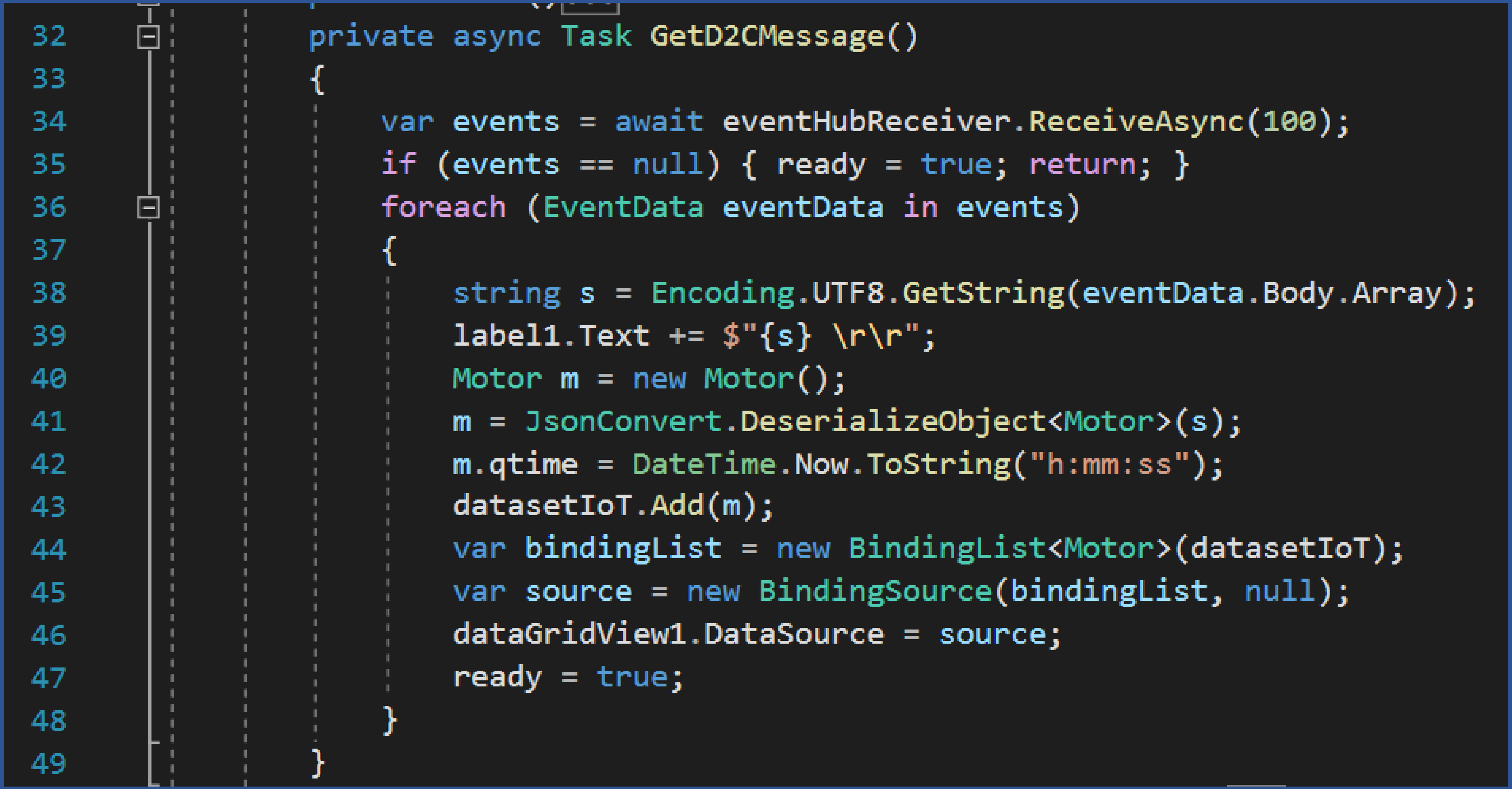
```
private Timer myTimer = new Timer();
private bool ready = true;
private List<Motor> datasetIoT = new List<Motor>();

private readonly string s_eventHubsCompatibleEndpoint =
    "sb://ihsuprods[redacted].servicebus.windows.net/";
private readonly string s_eventHubsCompatiblePath =
    "iothub-ehub-loyiot[redacted]4f3c";
private readonly string s_iotHubSasKey =
    "KCyf3omKkmWncX[redacted]xBWRoWgmAw=";
private readonly string s_iotHubSasKeyName = "service";
private EventHubClient s_eventHubClient;
private PartitionReceiver eventHubReceiver;
```

Add a Label and a DataGridView to Form1

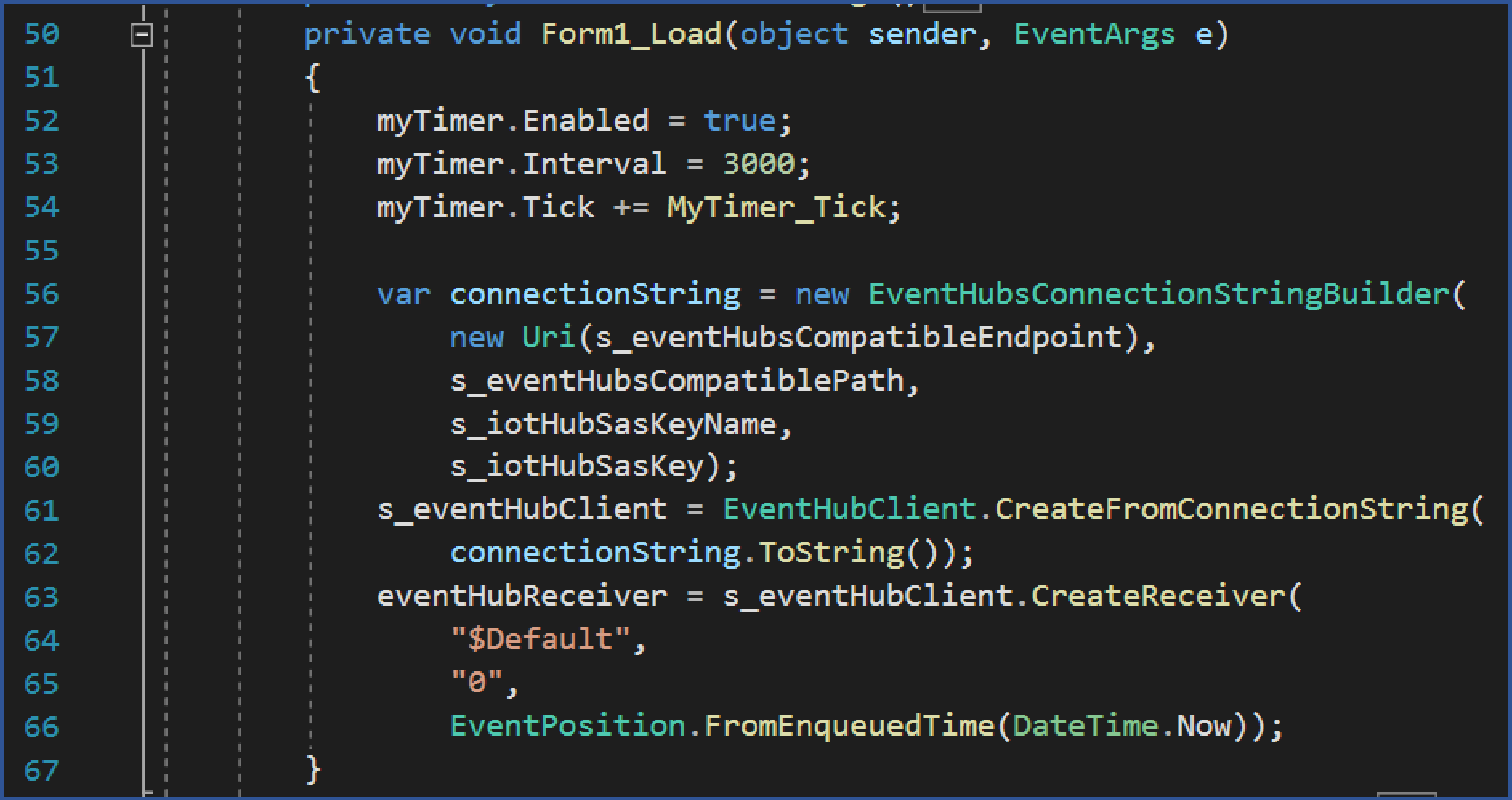


Add method GetD2CMessage()



```
private async Task GetD2CMessage()
{
    var events = await eventHubReceiver.ReceiveAsync(100);
    if (events == null) { ready = true; return; }
    foreach (EventData eventData in events)
    {
        string s = Encoding.UTF8.GetString(eventData.Body.Array);
        label1.Text += $"{s} \r\r";
        Motor m = new Motor();
        m = JsonConvert.DeserializeObject<Motor>(s);
        m.qtime = DateTime.Now.ToString("h:mm:ss");
        datasetIoT.Add(m);
        var bindingList = new BindingList<Motor>(datasetIoT);
        var source = new BindingSource(bindingList, null);
        dataGridView1.DataSource = source;
        ready = true;
    }
}
```

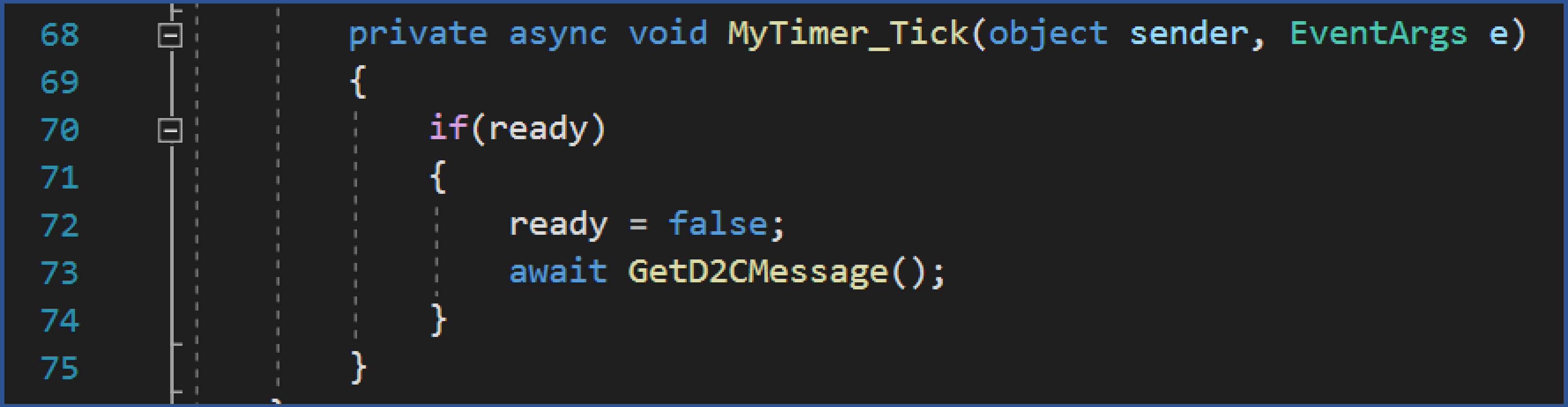
Add code to Form1_Load



```
private void Form1_Load(object sender, EventArgs e)
{
    myTimer.Enabled = true;
    myTimer.Interval = 3000;
    myTimer.Tick += MyTimer_Tick;

    var connectionString = new EventHubsConnectionStringBuilder(
        new Uri(s_eventHubsCompatibleEndpoint),
        s_eventHubsCompatiblePath,
        s_iotHubSasKeyName,
        s_iotHubSasKey);
    s_eventHubClient = EventHubClient.CreateFromConnectionString(
        connectionString.ToString());
    eventHubReceiver = s_eventHubClient.CreateReceiver(
        "$Default",
        "0",
        EventPosition.FromEnqueuedTime(DateTime.Now));
}
```

Add code to myTimer_Tick



```
        private async void MyTimer_Tick(object sender, EventArgs e)
        {
            if(ready)
            {
                ready = false;
                await GetD2CMessage();
            }
        }
```

What's next?