University of South-Eastern Norway

IIA1319: Software Engineering

Assignment #1: C#

Isak Skeie, 245362

Faculty of Technology, Natural sciences and Maritime Sciences

Campus Porsgrunn

Contents

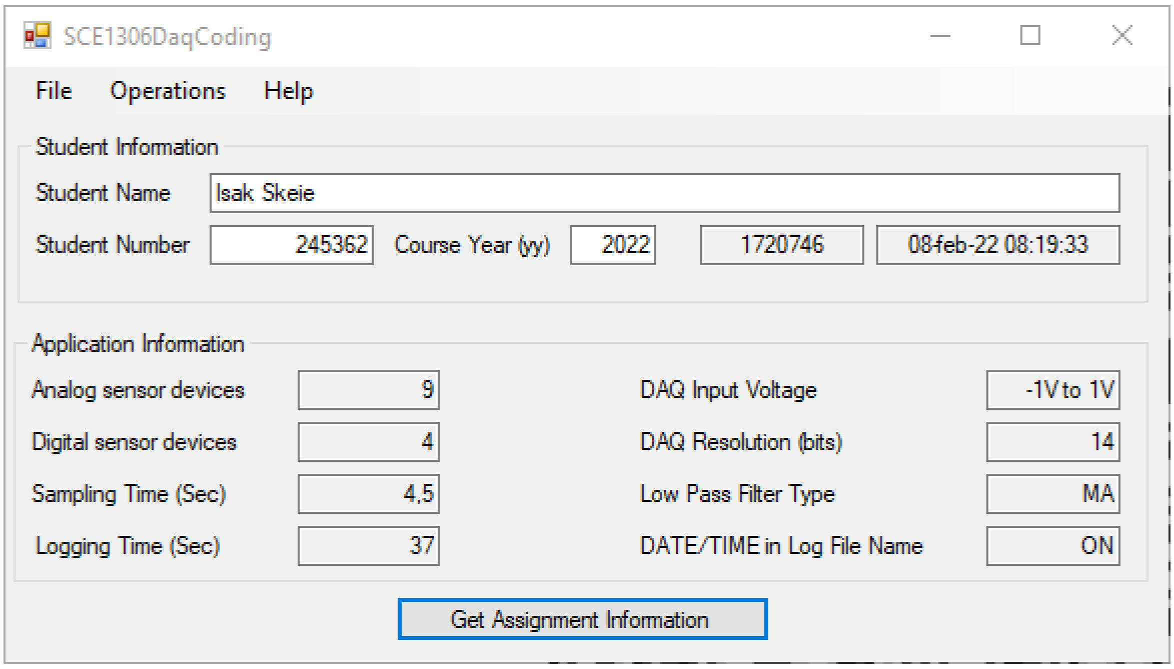
1 Introduction

The purpose of this assignment is to learn C# while using Git for source control. By doing this you obtain a way of programming that builds version control and good traceability in the code. This is done by creating a DAQ-Simulator. A tool useful for engineers testing Process control systems.

1.1 Assignment setup

The parameters for the DAQ-simulation are given by a executable name SCE1306DaqCoding, as seen in Figure 1.

Figure 2 shows the configuration of the git repository. This includes the path of the repository, username, and email.



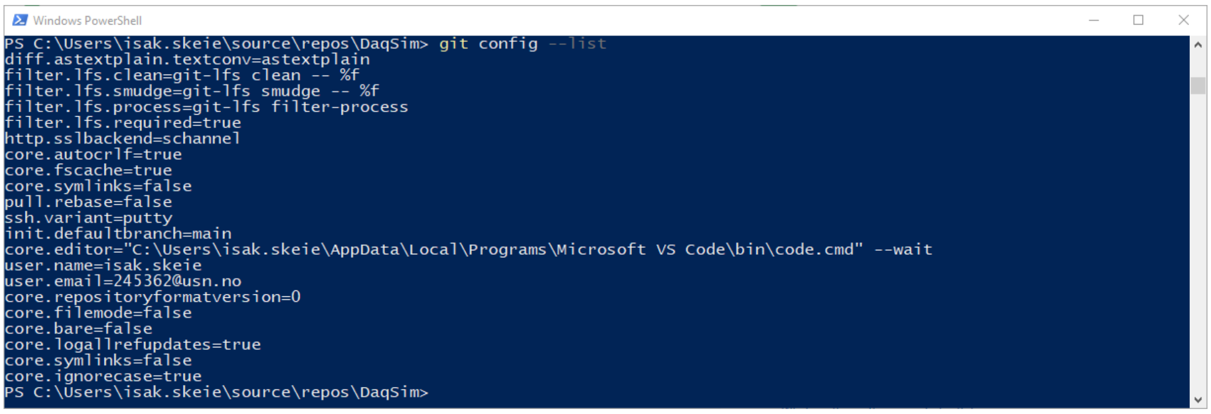
Student Information

Student Name	Isak Skeie				
Student Number	245362	Course Year (yy)	2022	1720746	08-feb-22 08:19:33

Application Information

Analog sensor devices	9	DAQ Input Voltage	-1V to 1V
Digital sensor devices	4	DAQ Resolution (bits)	14
Sampling Time (Sec)	4,5	Low Pass Filter Type	MA
Logging Time (Sec)	37	DATE/TIME in Log File Name	ON

Figure 1: The DAQ parameters given by SCE1306DaqCoding



Windows PowerShell

```
PS C:\Users\isak.skeie\source\repos\DaqSim> git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=schannel
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
ssh.variant=putty
init.defaultbranch=main
core.editor="C:\Users\isak.skeie\AppData\Local\Programs\Microsoft VS Code\bin\code.cmd" --wait
user.name=isak.skeie
user.email=245362@usn.no
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
PS C:\Users\isak.skeie\source\repos\DaqSim>
```

Figure 2: Configuration of local repo

2 DAQ

The first step of the application is to create a gui, then create the functionality behind the sampling button, and finalize the application with functionality for the logging button.

2.1 GUI

The layout of the gui can be seen in Figure 3. Based upon the Gui from the assignment specification. The commit of GUI to the repo can be seen in Figure 4.

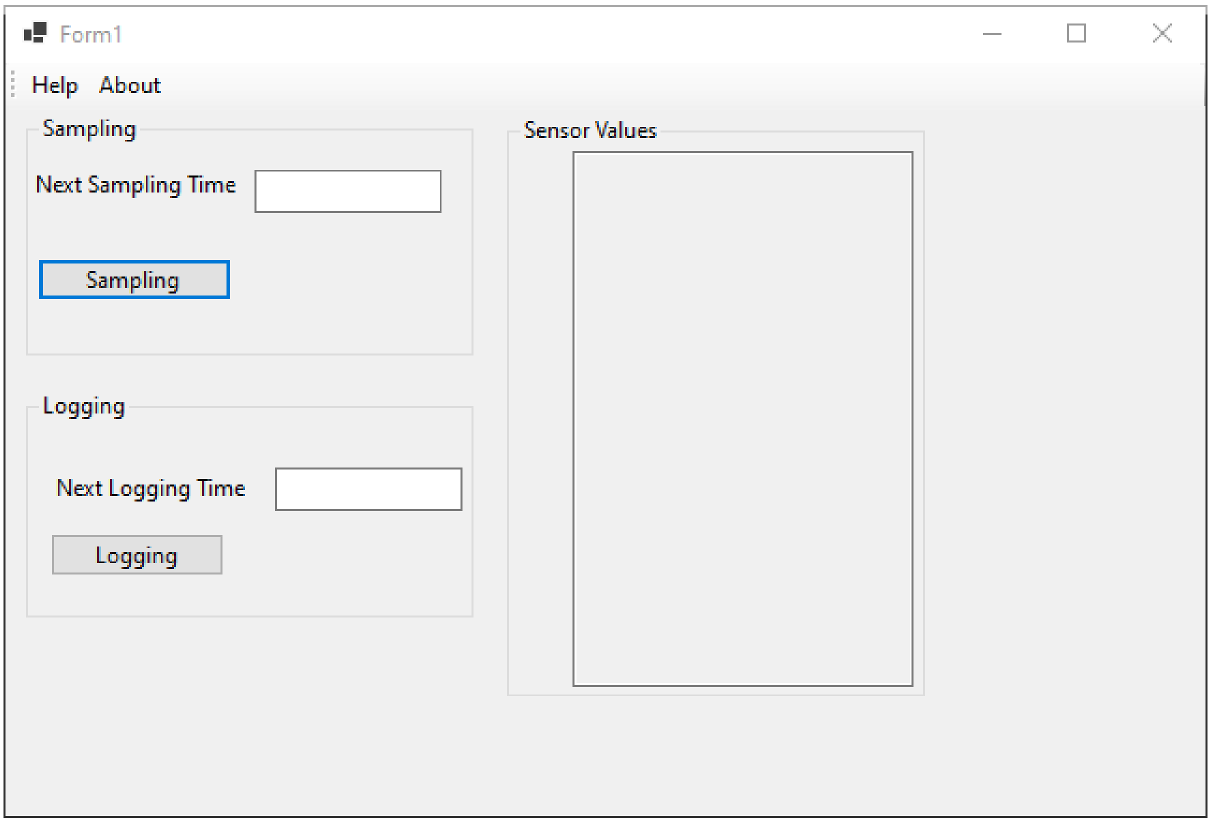
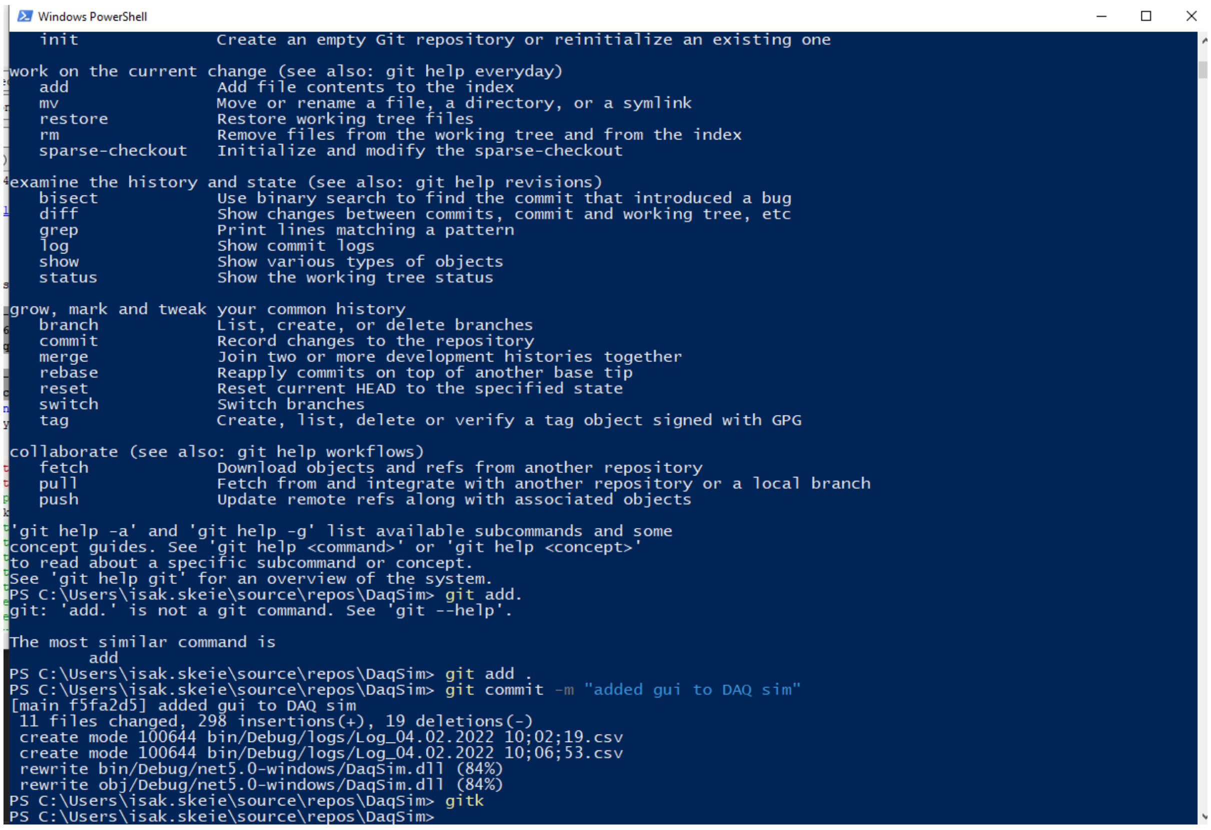


Figure 3: Screenhot of Gui



```
   init               Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add                Add file contents to the index
   mv                 Move or rename a file, a directory, or a symlink
   restore            Restore working tree files
   rm                 Remove files from the working tree and from the index
   sparse-checkout    Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect             Use binary search to find the commit that introduced a bug
   diff               Show changes between commits, commit and working tree, etc
   grep               Print lines matching a pattern
   log                Show commit logs
   show               Show various types of objects
   status             Show the working tree status

grow, mark and tweak your common history
   branch             List, create, or delete branches
   commit             Record changes to the repository
   merge              Join two or more development histories together
   rebase             Reapply commits on top of another base tip
   reset              Reset current HEAD to the specified state
   switch             Switch branches
   tag                Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch              Download objects and refs from another repository
   pull               Fetch from and integrate with another repository or a local branch
   push               Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
PS C:\Users\isak.skeie\source\repos\DaqSim> git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
        add
PS C:\Users\isak.skeie\source\repos\DaqSim> git add .
PS C:\Users\isak.skeie\source\repos\DaqSim> git commit -m "added gui to DAQ sim"
[main f5fa2d5] added gui to DAQ sim
 11 files changed, 298 insertions(+), 19 deletions(-)
 create mode 100644 bin/Debug/logs/Log_04.02.2022 10;02;19.csv
 create mode 100644 bin/Debug/logs/Log_04.02.2022 10;06;53.csv
 rewrite bin/Debug/net5.0-windows/DaqSim.dll (84%)
 rewrite obj/Debug/net5.0-windows/DaqSim.dll (84%)
PS C:\Users\isak.skeie\source\repos\DaqSim> gitk
PS C:\Users\isak.skeie\source\repos\DaqSim>
```

Figure 4: Commit of GUI

2.2 Sampling Sensor Device values

2.2.1 Sample button

When the sample button is pressed it starts two asynchronous tasks. It collects sensor readings and formats it, concurrently with another task displaying the samples in a textbox in the GUI. By having the tasks run async, its possible to run the sampling continuous. The sample button then works like a toggle, which starts/stops the sampling when pressed. This makes the requirement of checking for a valid sample interval obsolete.

2.2.2 Digital sampling

C# does not offer a way to sample random Boolean values [1]. A solution to this would be to create a random number the same way as for Analog values, with NextDouble. This creates a random double number between 0 and 1. By casting this to an integer, it rounds the value up to 1 if the random number is over 0.5 or 0 if its under 0.5. This assumes a normal distribution of Boolean values are preferable.

2.3 Help About Information

A toolbar is added to the GUI. This includes a Help button and a About button. The functionality is added to the GUI without having any content linked to it, this is to be setup at a later stage, when/if the application gets used by another user that’s not developing the application.

2.4 Logging Sensor device values

2.4.1 Logging button

When the button is pressed, continuous logging starts. This makes the requirement for having a method check for a valid logging interval obsolete. The sample button works like a toggle button. The button start/stops the process of logging filtered samples. When the button is pressed to start the logging, an asynchronous method starts. This checks if there’s enough samples created, to create a filtered value. If that’s the case, a portion of the end of the list of samples gets filtered and written to a CSV-file. The format of this CSV-file is show in Table 1.

Table 1: Format of logfile generate by DAQ

A0	A1	A2	A3	A4	A5	A6	A7	A8	D0	D1	D2	D3
0.13	0.001	0.018	-0.029	-0.005	0.031	0.207	0.061	-0.041	0	0	0	1
-0.039	-0.095	0.18	0.063	-0.098	0.155	-0.053	-0.005	-0.024	0	0	0	0
-0.011	-0.162	-0.174	0.099	0.088	-0.092	-0.062	0.008	-0.111	0	0	1	1

When a filtered sample is available, it gets written to a CSV-file. This is done with the help of a library called CsvHelper, which uses the structure of the DataStruct class to write headers, as well as append filtered samples on lines. The date and time get used in the filename of the CSV-file, this is to automatically create a new file for each run. And be able to easily see previous log sessions.

2.4.2 Filter as interconnection

When writing the code, two methods for logging the correct values are created. The first is for filtering the sampled sensor values, and the second is for writing the filtered data to a CSV-file. The first method takes a portion of the samples, filters the readings, and then returns a filtered data collection. The second method then takes this filtered collection of data as input and writes it to a CSV-file. By placing the filter-method as the input for the logging method. The low pass filter interconnects the samples and logging.

3 Documentation

The figure below shows a simplified flowchart of how the different classes and method work together to make the DAQ-application work.

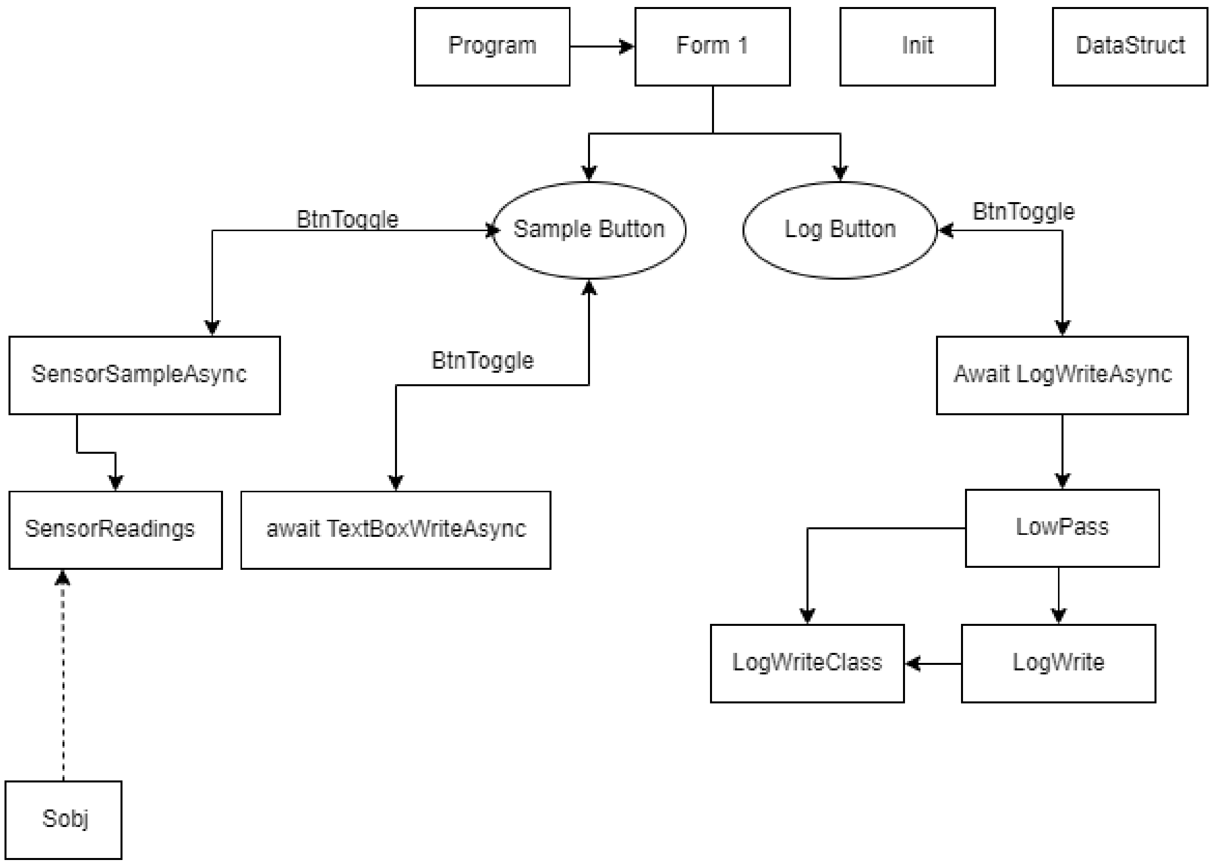


Figure 5: Flowchart of code

3.1.1 Classes and Methods in application

- Program: Initializes and starts Form 1.
- Form 1: Manages the components in UI
  - TextBoxWriteAsync: Method for writing sample data to UI.
  - LogWriteAsync: Method for filtering and writing data to a CSV-file.
  - SampleBtn: Method which starts sampling and timers once sample button is pressed.
  - LogBtn: Method which starts logging method LogWriteAsync with push of button.

- Init: Static class which contains parameters for DAQ.
- DataStruct: Class which creates the structure of the data being handled.
- SensorReadings: Class For sensor data
  - SensorSampleAsync: Method which gets sensor data from SObj and formats it to datastructure and writes it to a list
- LogWriteClass: Class for processing data
  - LogWrite: Method for Writing data to a CSV-file.
  - LowPass: Method that filters data from input. Returns filtered data
- sObj: Class for creating sensor data.
  - aSensor: Creates random ranged double integers for analog sensors.
  - dSensor: Creates random Boolean value for digital sensor.

3.1.2 Git Log

The log is collected using *git log >log.txt*, which creates a text file with the log information. This is then copied over.

```
•  commit 8a2bf0a848de854a6d630e58cbadebf2baa2f3a5
•  Author: isak.skeie <245362@usn.no>
•  Date:   Tue Feb 8 10:16:00 2022 +0100
•
•      Rounds the logged values
•
•  commit c64a3944b28fa8a9d48155b055268d67b0c2f89e
•  Author: isak.skeie <245362@usn.no>
•  Date:   Tue Feb 8 09:08:19 2022 +0100
•
•      Cleaned up code, and added method headers
•
•  commit cbb30bf05460b1b1caa8c9061f48ba3fbdd969f2
•  Author: isak.skeie <245362@usn.no>
•  Date:   Mon Feb 7 20:09:14 2022 +0100
•
•      Finnished Code :)
•
•  commit 4b10e07020bdf291dae3d6d3529e8d9ffb706c42
•  Author: isak.skeie <245362@usn.no>
•  Date:   Mon Feb 7 19:59:07 2022 +0100
•
•      LogWrite method now appends log, after initial write
•
•  commit 0604d5c0b94e60d0f41869c2a19ad3ffb26f35bf
•  Author: isak.skeie <245362@usn.no>
•  Date:   Mon Feb 7 17:15:41 2022 +0100
•
```

```
•      BugFixed Logwrite method
•
•   commit 5dc7746ffe6faee9e64b0e2996a5f31d76d31621
•   Author: isak.skeie <245362@usn.no>
•   Date:   Mon Feb 7 16:46:01 2022 +0100
•
•      Added LowPass filter
•
•   commit 930e52af52844df274eaabb13b36f0a22e173529
•   Author: isak.skeie <245362@usn.no>
•   Date:   Mon Feb 7 14:45:58 2022 +0100
•
•      Added Log writer class
•
•   commit 8e57f65911f1321a58ea52e12846bb0da89fb650
•   Author: isak.skeie <245362@usn.no>
•   Date:   Mon Feb 7 14:26:30 2022 +0100
•
•      Fixed continous sampling with list. Stops when Sampling button is pressed again
•
•   commit 64592c04205732c2f0a170754b1ffcf651577ed7
•   Author: isak.skeie <245362@usn.no>
•   Date:   Sun Feb 6 18:15:25 2022 +0100
•
•      Added Help and Abot on ToolStrip
•
•   commit 58b064c2b15f622ed456e74e005be959b386ad1a
•   Author: isak.skeie <245362@usn.no>
•   Date:   Sun Feb 6 17:56:55 2022 +0100
•
•      Added async sampling button functionality
•
•   commit f5fa2d5d39dec39e15df3fb43d384e9d4f68f988
•   Author: isak.skeie <245362@usn.no>
•   Date:   Fri Feb 4 10:11:18 2022 +0100
•
•      added gui to DAQ sim
•
•   commit f977c3e501a8114cbf344a8a29ce8c3fcef62b4b
•   Author: isak.skeie <245362@usn.no>
•   Date:   Wed Feb 2 22:39:19 2022 +0100
•
•      Added DateTimeStamp For each Log file
•
•   commit d366c32cdd52e8a13e76407335bed1f681bb3e94
•   Author: isak.skeie <245362@usn.no>
•   Date:   Wed Feb 2 20:19:27 2022 +0100
```

-
-     Added SensorData class. Published data to CSV
-
- commit 17871e7f4149bf2bd2c7eebe049bfea6c193b765
- Author: isak.skeie <245362@usn.no>
- Date: Fri Jan 28 11:39:29 2022 +0100
-
-     First commit

4 Summary

The purpose of this assignment have been to learn C# and the use of Git. To be able to create the DAQ-application specified in the assignment, its necessary to know the basics C# programming. This includes declaring objects, classes, methods, filewrit, WinForm , Object Oriented Programming and Asynchronous programming. By making the task as open as it is, you obtain a programming methodology needed to design applications in C#. With the addition of being instructed to use Git. Which gives you a valuable tool for version control in programming. You obtain a method for programming with good traceability and a clarity that makes it easier for others to pick up your work.

The application created for this assignment have been made with asynchronous Methods for sampling and logging data. This enables the sampling and logging to be continuous, working in parallel, and the ability to stop the sampling/logging whenever. If the application were to be handed down to a user. There should be made some improvements and fixes to the application. The filename created for the log file is awkward, it contains both the date and the time in a fomat that's unusuall, and should be fixed some time. Functionality for creating a new logfile without restarting the application should be added, as of now, if the logging is stopped then start again, the logging continuous on the same file. Lastly, the parameters for the DAQ should be moved to a xml file. This makes it possible to change the parameters of the application after it has been built, and sent off to as user. Furthermore, the application should bee tested by a user to find eventual bugs in the program before a build of the DAQ-application.

Figures

5 References

[1] L. V. D. Ouweland, «Random boolean in cshrap,» 08 2 2022. [Internett]. Available: https://www.loekvandenouweland.com/content/random-boolean-in-csharp.html.

Appendices

Appendix A: Code

Program

```
using CsvHelper;
using CsvHelper.Configuration;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DaqSim
{


    static public class Program
    {
        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]



        static public void Main()
        {

            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }


    public static class Init
    {
        /*
         * Paramaters given by SCE1306DaqCoding
         * V1
         * Created by Isak SKeie
         */
        public static int seconds = 0;
        public static double SampleInterval = 4.5;
        public static int LogInterval = 37;
        public static DateTime SampleStartTime = new DateTime();
        public static DateTime LogStartTime = new DateTime();
        public static DateTime SampleTimer = new DateTime();
        public static bool sampleStart;
        public static bool logStart;


    }

    public class DataStruct
    {
        /*
         * DataStructure for data being generated and handled
         * V1
         * Created by Isak SKeie
         */

        public double A0 { get; set; }
        public double A1 { get; set; }
        public double A2 { get; set; }
        public double A3 { get; set; }
        public double A4 { get; set; }
        public double A5 { get; set; }
        public double A6 { get; set; }
        public double A7 { get; set; }
        public double A8 { get; set; }

        public double D0 { get; set; }
        public double D1 { get; set; }
        public double D2 { get; set; }
        public double D3 { get; set; }

    }



    public class SensorReadings
    {
        public sObj reading = new sObj();
        public static List<DataStruct> DataList = new List<DataStruct>();

        public async Task<Task> SensorSampleAsync()
        {
            /*
             * Assigns SensorReadings in a given sampling interval
             * V3
             * Created by Isak Skeie
             */


            int StartTime = DateTime.Now.Second;
            int i = 1;
            TimeSpan span;



            while (Init.sampleStart)
            {

                Init.SampleTimer = DateTime.Now;
                if (Init.SampleTimer.Subtract(Init.SampleStartTime).TotalSeconds
> Init.SampleInterval * i)
                {
                    var Data = new DataStruct
                    {
                        A0 = reading.aSensor(),
                        A1 = reading.aSensor(),
                        A2 = reading.aSensor(),
                        A3 = reading.aSensor(),
                        A4 = reading.aSensor(),
                        A5 = reading.aSensor(),
                        A6 = reading.aSensor(),
                        A7 = reading.aSensor(),
                        A8 = reading.aSensor(),

                        D0 = reading.dSensor(),
                        D1 = reading.dSensor(),
                        D2 = reading.dSensor(),
                        D3 = reading.dSensor()
                    };

                    DataList.Add(Data);
                    i++;
                    Console.WriteLine(DataList.Sum(x => x.A0));
                }

                await Task.Delay(100);
            }

            return Task.CompletedTask;
        }
    }


    public class LogWriteClass
    {
        public DateTime LogDate;
        public string FileName;

        public LogWriteClass()
        {
            LogDate = DateTime.Now;
        }



        public void LogWrite(List<DataStruct> filteredData)
        {
            /*
             * Writes filtered samples to a CSV-File
             * V2
             * Created by Isak Skeie with append functionality from:
https://joshclose.github.io/CsvHelper/examples/writing/appending-to-an-existing-
file/ (07.02.2022)
             */

            if (FileName == null)
            {
                FileName = @"..\logs\Log_" + LogDate.ToString("dd/MM/yyyy
HH;mm;ss") + ".csv";

                using (var writer = new StreamWriter(FileName))
                using (var csv = new CsvWriter(writer,
CultureInfo.InvariantCulture))
                {
                    csv.WriteRecords(filteredData);
                }
            }

            else
            {
                var config = new CsvConfiguration(CultureInfo.InvariantCulture)
                {
                    HasHeaderRecord = false,
                };
                using (var stream = File.Open(FileName, FileMode.Append))
                using (var writer = new StreamWriter(stream))
                using (var csv = new CsvWriter(writer, config))
                {
                    csv.WriteRecords(filteredData);
                }
            }
        }



        public List<DataStruct> LowPass(List<DataStruct> Samples)
        {
            /*Filters input data
             * V1
             * Created by Isak SKeie
             */
            List<DataStruct> filtered = new List<DataStruct>();
            int nSamples = Samples.Count();
            int rounding = 3;

            var Data = new DataStruct
            {
                A0 = Math.Round(Samples.Sum(x => x.A0) / nSamples, rounding),
                A1 = Math.Round(Samples.Sum(x => x.A1) / nSamples, rounding),
                A2 = Math.Round(Samples.Sum(x => x.A2) / nSamples, rounding),
                A3 = Math.Round(Samples.Sum(x => x.A3) / nSamples, rounding),
                A4 = Math.Round(Samples.Sum(x => x.A4) / nSamples, rounding),
                A5 = Math.Round(Samples.Sum(x => x.A5) / nSamples, rounding),
                A6 = Math.Round(Samples.Sum(x => x.A6) / nSamples, rounding),
                A7 = Math.Round(Samples.Sum(x => x.A7) / nSamples, rounding),
                A8 = Math.Round(Samples.Sum(x => x.A8) / nSamples, rounding),

                D0 = Math.Round(Samples.Sum(x => x.D0)/nSamples),
                D1 = Math.Round(Samples.Sum(x => x.D1)/nSamples),
                D2 = Math.Round(Samples.Sum(x => x.D2)/nSamples),
                D3 = Math.Round(Samples.Sum(x => x.D3)/nSamples),
            };


            filtered.Add(Data);
            return filtered;
        }
    }
}
```

sObj

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace DaqSim
{
    public class sObj
    {

        public Random rSensVal = new Random();
        private static double bits = 14;
        public double resolution = Math.Pow(2, bits);



        public double aSensor()
        {
            /*Creates random double numbers with the right range
            *V2,
            *Created by Isak SKeie
            */

            double aVal = rSensVal.NextDouble();
            aVal = ((aVal * resolution)-resolution/2);
            aVal = aVal / resolution;
            return aVal;
        }

        public int dSensor()
        {
           /*Creates random bool value
            * V1
            * Created by Isak Skeie
            */
           int dVal =  (int)Math.Round(rSensVal.NextDouble());
           return dVal;

        }
    }
}
```

Form1 (Unused methods are removed for this appendix)

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

namespace DaqSim
{
    public partial class Form1 : Form
    {
        //Initializes SensorReading class and LogList
        public SensorReadings InitializeSensors = new SensorReadings();
        public List<DataStruct> LogList = new List<DataStruct>();

        public Form1()
        {

            InitializeComponent();

        }


        public async void logBtn_Click(object sender, EventArgs e)
        {
            /*Toggles button bool, starts writing to CSV
             * V2
             * Created by Isak SKeie
             */
            Init.logStart = !Init.logStart;
            Init.LogStartTime = DateTime.Now;
            await LogWriteAsync();
        }


        private async void sampleBtn(object sender, EventArgs e)
        {
            /*Toggles buttoon bool, starts samplings sensor values, and
displays/formats the data concurrently
             * V3
             * Created by Isak SKeie
             */

            Init.sampleStart = !Init.sampleStart;
            Init.SampleStartTime = DateTime.Now;

            InitializeSensors.SensorSampleAsync();

            await TexBoxWriteAsync();

        }

        public async Task<Task> TexBoxWriteAsync()
        {
            /*
             * Writes sampled data to Textbox, Displays sampling time and which
sample in UI
             * V4
             * Created by Isak Skeie
             */

            int i = 0;
            string textBox = "";
            DateTime SampleTimer;
            DateTime start = DateTime.Now;
            double secondsLeft;


            while (Init.sampleStart)
            {
                SampleTimer = DateTime.Now;
                textBox3.Text =Math.Round(Init.SampleInterval*(i+1)-
SampleTimer.Subtract(Init.SampleStartTime).TotalSeconds,1).ToString();

             if (SensorReadings.DataList.Count == i + 1)
             {
               groupBox3.Text = $"Sample: { i.ToString() }";
                textBox = "";
                textBox+="A0: "+SensorReadings.DataList[i].A0.ToString()+ "\r\n";
                textBox+="A1: "+SensorReadings.DataList[i].A1.ToString()+"\r\n";
                textBox+="A2: "+SensorReadings.DataList[i].A2.ToString()+"\r\n";
                textBox+="A3: "+SensorReadings.DataList[i].A3.ToString()+"\r\n";
                textBox+="A4: " +SensorReadings.DataList[i].A4.ToString()+"\r\n";
                textBox+="A5: "+SensorReadings.DataList[i].A5.ToString()+"\r\n";
                textBox += "A6:"+SensorReadings.DataList[i].A6.ToString()+"\r\n";
                textBox += "A7:"+SensorReadings.DataList[i].A7.ToString()+"\r\n";
               textBox += "A8: "+SensorReadings.DataList[i].A8.ToString()+"\r\n";
               textBox += "D0: "+SensorReadings.DataList[i].D0.ToString()+"\r\n";
               textBox += "D1: "+SensorReadings.DataList[i].D1.ToString()+"\r\n";
               textBox += "D2: "+SensorReadings.DataList[i].D2.ToString()+"\r\n";
               textBox += "D3: "+SensorReadings.DataList[i].D3.ToString()+"\r\n";


                    ValueDisplay.Text = textBox;
                    i++;
                }
                await Task.Delay(20);
            }
            return Task.CompletedTask;
        }


        public async Task<Task> LogWriteAsync()
        {
            /*Filters sampled data and writes to CSV
             * V2
             * Created By Isak SKeie
             */

            int n = 1;
            DateTime LogTimer;
            LogWriteClass Writer = new LogWriteClass();
            int samples;
            List<DataStruct> SampleList = new List<DataStruct>();
            int samplePeriod = (int) Init.LogInterval / (int)Init.SampleInterval
- 1;


            while(Init.logStart)
            {
                LogTimer = DateTime.Now;
                samples = SensorReadings.DataList.Count;

                textBox2.Text = Math.Round(Init.LogInterval * n -
LogTimer.Subtract(Init.LogStartTime).TotalSeconds, 1).ToString();

                if ( Init.LogInterval*n < samples*Init.SampleInterval )
                {
                    SampleList =
SensorReadings.DataList.TakeLast(samplePeriod).ToList();
                    var filtered = Writer.LowPass(SampleList);
                    Writer.LogWrite(filtered);
                    n++;
                }
                await Task.Delay(10);
            }
            return Task.CompletedTask;
        }
    }
}
```