Assignment 1 in IIA1319

Torstein Solheim Ølberg | 263054

30. januar 2024

Innhold

1 Introduction **3**
- 1.1 Background 3
- 1.2 Report Structure 3
- 1.3 Method 4

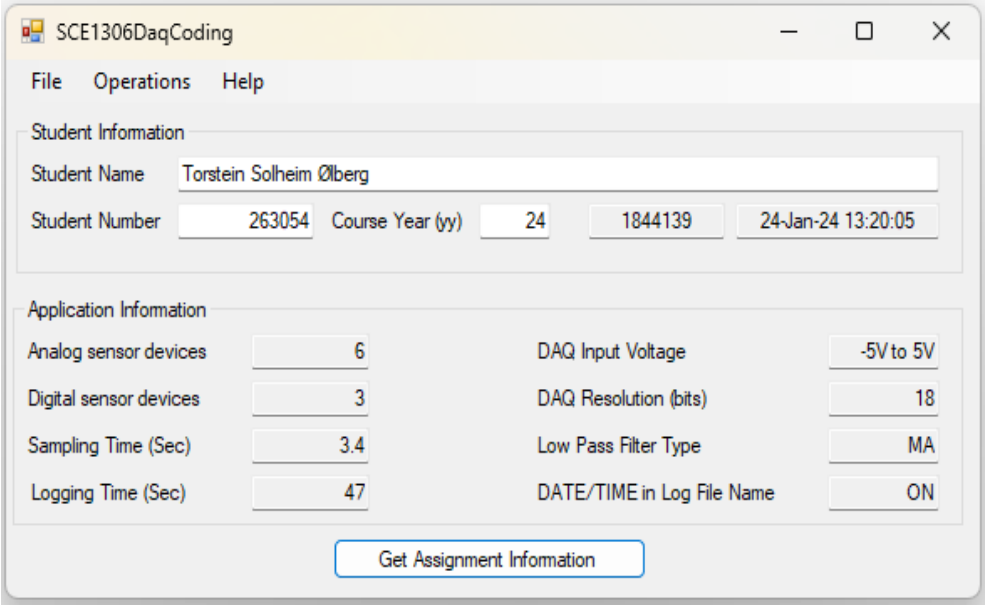
2 Results **4**

3 Discussion **5**

4 Conclusion **6**

5 Appendix A **6**

6 Appendix B **8**



Figur 1: Screenshot of the SCE1309DaqCoding application with parameters used for the project.

1 Introduction

1.1 Background

The testing of data manipulation software using datasets from sensors is an important and useful task, but can often become a costly affair if multiple expensive sensors are needed. As a result, the creation of a program capable of simulating multiple sensors and collecting data from them. This Data AcQuisition simulation program would sample sensors at an interval and save the data for later use. The exact parameters for program is decided by the SCE1306DaqCoding application, supplied by Nils-Olav Skeie at `https://usn.instructure.com/courses/31275/files/3251058/download?wrap=1`, as shown in figure 1.

1.2 Report Structure

This report contains a description of the resulting program developed, a discussion of the result and a summary of the report. Finally, there are two appendixes. The first appendix is a collection of the documentation for

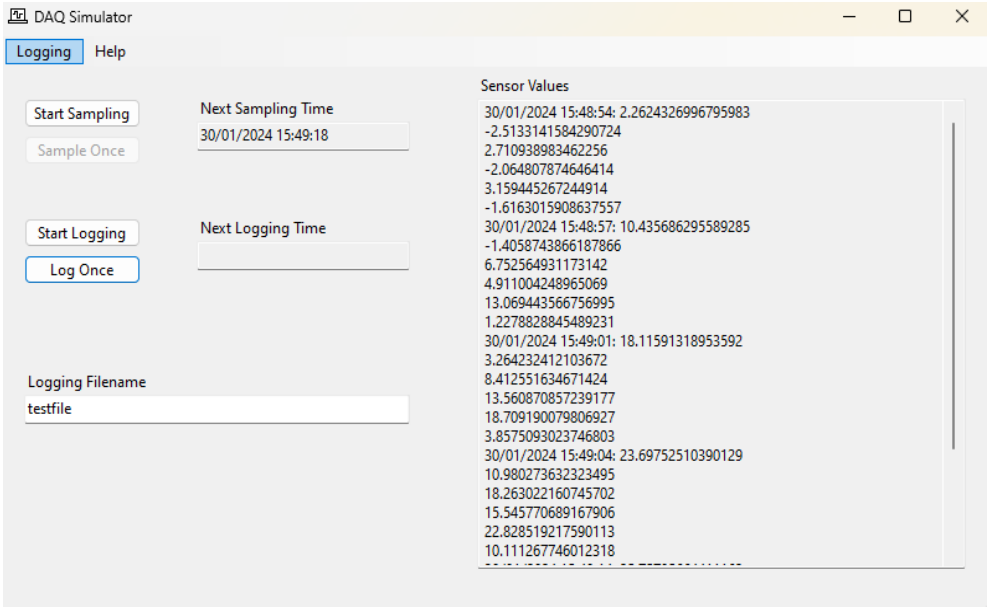
program. The second is a list of the programs developed by an author, and not auto generated by the IDE.

1.3 Method

To develop the DAQ simulation program the C# programming language is used to write the logic and handle user input. As an integrated development environment, Microsoft Visual Studio is used with the inbuilt tools for a Windows Forms Application. Git is used as the version control system, and the repository is uploaded to GitHub.
First the development of the GUI front is made. The main view with four buttons, two for logging and two for sampling, and four textboxes for displaying and inputting info. Alongside these are labels, explaining the textboxes purpose, and a menu-strip with a help page option. Then, a sensor class capable of storing a cumulative number and drawing new random number within a range is created. This class is used to set up six sensor instances and produce new samples when needed. Furthermore, a data class is created, capable of storing the samples of six sensors and the date of when they where taken. Also, the handlers for clicking the different buttons and the menu-strip help option are created. They either start a sampling or logging session, sample or log once, or display a message box with info on the app. Then a method each for performing the sampling, extracting the sensor values and saving them to a list of sensor value data classes, and logging, opening a csv file and writing the data from the data class list into the file. Finally, a pair of timers are added, which keep track of how much time has passed since last logging or sampling was performed and re-enables them as needed.

2 Results

The program created, as seen in figure 2 consists of a control to start a sampling session, or sample sensors once. The program checks if there has passed a sufficient amount of time since last sampling and either automatically samples again or allows for sampling manually. The same controls are also available for logging, saving all sampled results since last logging was preformed, at a minimum interval of 47 seconds. There is also the option to input a name for the file where the program should log the data. Then there are three windows, two where the time when the next sampling and logging will be available and one where all values whom have been sampled and are reddy to be logged can be viewed. Finally, there is a help page where info



Figur 2: Screenshot of the application during use.

on how to use the application can be viewed. The data logged is saved in a CSV file on the format displayed by this extract:

```
Date,Sensor 1,Sensor 2,Sensor 3,Sensor 4,Sensor 5,Sensor 6
30/01/2024
    15:43:41,2.2624326996795983,-2.5133141584290724,2.710938983462256,-2.064807874646414,3.159
```

3 Discussion

The program developed proves that it is possible to create a DAQ program simulating intake from a sensor and saving it in a simple format. However, much more customizability is required for the application to be useful for more than a very specific case. It would be useful to be able to control both the size of the intervals for logging and sampling, as well as the range the sensors pick their random numbers from. It would also be useful for the numbers of sensors used to be changed by the user. Automatic tests for the program to check if it works as expected, and an in-depth analysis of the probability distribution of the random number selector used in this program would also be a good idea, as this will make sure the program functions as

expected both now and when further developed.
Finally, a low pass filter could be implemented to better simulate analogue data from real sensors, as these can often times have noise which could result in data way out of the normal scope. This filter would be a check of the numbers acquired by the sensor classes and applied to the data before they are stored in the data classes. This would also allow for future real sensors to be used with the application, so as to produce even better testing data.

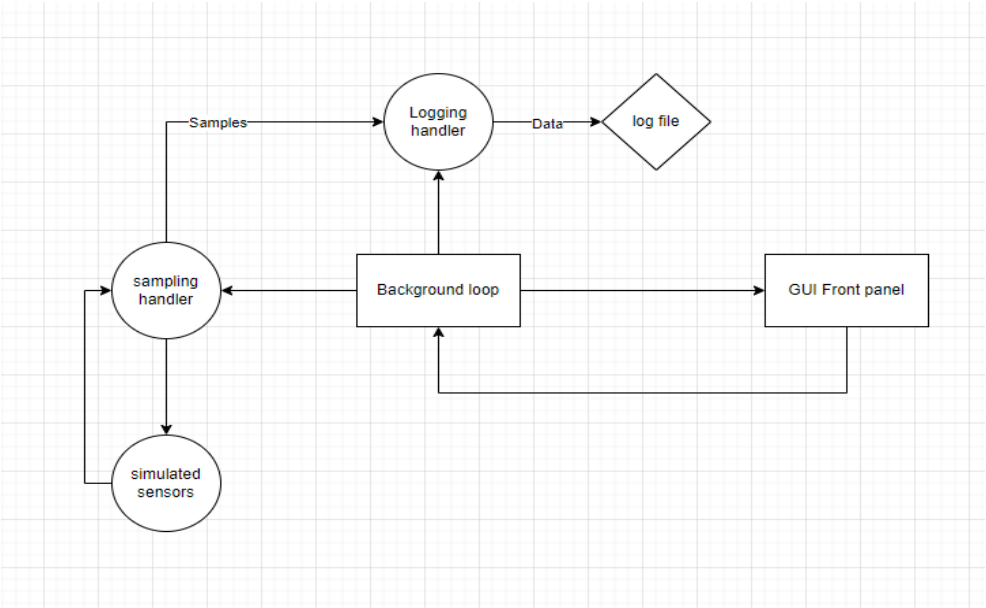
4 Conclusion

A DAQ simulation program is possible and would be useful to have as an alternative to expensive sensors. This project has produced a simple DAQ application and it can sample and log data from six sensors at intervals. However, this program is very simple and could need a lot more work fro it to be very useful.

5 Appendix A

Here you can find the documentation for the program.
The program consists of three classes, and an increasing number of objects as a new object is created each time a sampling is performed. However, the base application before any sampling is done consists of 7 objects of these classes. Below follows an example of a log file produced by the program. The lines have been broken to fit the page and are normally not.

```
Date,Sensor 1,Sensor 2,Sensor 3,Sensor 4,Sensor 5,Sensor 6
30/01/2024 15:05:58,2.2624326996795983,-2.5133141584290724,
2.710938983462256,-2.064807874646414,3.159445267244914,-1.6163015908637557
30/01/2024 15:06:01,10.435686295589285,-1.4058743866187866,
6.752564931173142,4.911004248965069,13.069443566756995,1.2278828845489231
30/01/2024 15:06:04,18.11591318953592,3.264232412103672,
8.412551634671424,13.560870857239177,18.709190079806927,3.8575093023746803
30/01/2024 15:06:08,23.69752510390129,10.980273632323495,
18.263022160745702,15.545770689167906,22.828519217590113,10.111267746012318
30/01/2024 15:06:11,25.75785664411162,17.55546257018832,
19.353068496265017,21.150674422341712,22.948280348418415,14.74588627449511
```



Figur 3: A flowchart depicting the general structure of the application.

```
commit 716aeaa3d57afff31f51542b0bfd3d29b4c2097b (HEAD -> main)
Author: Torstein S. Ølberg <mr.torstein@gmail.com>
Date:   Tue Jan 30 13:38:09 2024 +0100

    Added method headers to all selfmade method

commit b225e617d6a6ded5c5b12831e6c0d2942cff9e1b
Author: Torstein S. Ølberg <mr.torstein@gmail.com>
Date:   Tue Jan 30 13:01:42 2024 +0100

    Added Help page and scrollbar to values textbox

commit 78230c768e52c39028e6013f0adab21c2891155d
Author: Torstein S. Ølberg <mr.torstein@gmail.com>
Date:   Tue Jan 30 12:13:13 2024 +0100

    Added saving time of sampling to csv

commit d92f9fdae62c3d0ce928169d9f830999515f02f1e
Author: Torstein S. Ølberg <mr.torstein@gmail.com>
Date:   Tue Jan 30 11:57:48 2024 +0100

    working sampling and logging application

commit 9d5a27bd0efe77db4117bd8e7bf6d1376a34d350 (origin/main, origin/HEAD)
Author: Torstein S. Ølberg <mr.torstein@gmail.com>
Date:   Wed Jan 24 16:47:31 2024 +0100

    Created visual display of DAQ application
:
```

Figur 4: A list of git commits.

6 Appendix B

Here you can find the code files not made by Visual Studio itself.

```
using System.ComponentModel;
using System.Diagnostics;
using System.DirectoryServices;
using System.Globalization;
using static System.Windows.Forms.LinkLabel;
using SensorApp;

namespace DAQ_Simulation_Application
{
    // Purpose: Controlling the programs behaviour
    // Version: 30/01-24
    // Author: Torstein Solheim berg
    public partial class main_window : Form
    {
        readonly int sampling_time = 3400;
        readonly int logging_time = 47000;
        readonly TimeSpan min_sample_time = new(0, 0, 0, 3, 400);
        readonly TimeSpan min_log_time = new(0, 0, 0, 47, 0);

        readonly List<Sensor> sensors = [];
        readonly List<Sensordata> samples = [];

        Boolean sampling_status = false;
        Boolean logging_status = false;


        // Purpose: Initialize class and GUI components
        // Version: 30/01-24
        // Author: Torstein Solheim berg
        public main_window()
        {
            InitializeComponent();

            for (int i = 0; i < 6; i++)
            {
                sensors.Add(new Sensor(i));
            }

            sampling_timer.Interval = sampling_time;
            logging_timer.Interval = logging_time;
        }
```

```
// Purpose: Take one sample for each six sensors, save them
    and display them
// Version: 30/01-24
// Author: Torstein Solheim berg
private void TakeSample()
{
    DateTime time = DateTime.Now;
    sensor_value_textbox.Text += time.ToString() + ": ";
    samples.Add(new Sensordata());
    samples[^1].Date = time.ToString();
    for (int i = 0; i < sensors.Count; i++)
    {
        samples[^1][i] = sensors[i].GetNewValue();
        sensor_value_textbox.Text +=
            samples[^1][i].ToString() + Environment.NewLine;
    }
    next_sampling_time_textbox.Text =
        time.Add(min_sample_time).ToString();
}

// Purpose: Save samples gotten since last log time to a csv
    file
// Version: 30/01-24
// Author: Torstein Solheim berg
private void LogSamples()
{
    string path = "../../../data/";
    string filename;


    if (logging_filename_textbox.Text == "")
    {
        filename = DateTime.Now.ToString().Replace("/",
            "_").Replace(" ", "_").Replace(":", "_") + ".csv";
    }
    else if
        (logging_filename_textbox.Text.ToLower().Contains(".csv",
        StringComparison.CurrentCulture))
    {
        filename = logging_filename_textbox.Text;
    }
    else
    {
        filename = logging_filename_textbox.Text + ".csv";
```

```
    }

    FileInfo outfile_info = new(path + filename);
    Boolean appending = false;
    if (outfile_info.Exists)
    {
        appending = true;
    }

    using (StreamWriter outfile = new(Path.Combine(path,
        filename), appending))
    {
        if (!appending)
        {
            outfile.WriteLine("Date,Sensor 1,Sensor 2,Sensor
                3,Sensor 4,Sensor 5,Sensor 6");
        }
        foreach (Sensordata sample in samples)
            outfile.WriteLine("{0},{1},{2},{3},{4},{5},{6}",
                sample.Date, sample[0], sample[1], sample[2],
                sample[3], sample[4], sample[5]);
    }

    samples.Clear();
    sensor_value_textbox.Text = "";
    next_logging_time_textbox.Text =
        DateTime.Now.Add(min_log_time).ToString();
}

// Purpose: Activate or deactivate a sampling session when
    sampling button is clicked
// Version: 30/01-24
// Author: Torstein Solheim berg
private void SampleButtonClick(object sender, EventArgs e)
{
    if (!sampling_status)
    {
        sampling_status = true;
        sample_button.Text = "Stop Sampling";
        sample_once_button.Enabled = false;

        if (!sampling_timer.Enabled)
        {
            if (samples.Count == 0)
            {
```

```
                sensor_value_textbox.Text = "";
            }
            TakeSample();
            sampling_timer.Start();
        }
    }
    else
    {
        sampling_status = false;
        sample_button.Text = "Start Sampling";
    }
}

// Purpose: Activate or deactivate a logging session when
    logging button is clicked
// Version: 30/01-24
// Author: Torstein Solheim berg
private void LoggingButtonClick(object sender, EventArgs e)
{
    if (!logging_status)
    {
        logging_status = true;
        logging_button.Text = "Stop Logging";
        log_once_button.Enabled = false;

        if (!logging_timer.Enabled)
        {
            LogSamples();
            logging_timer.Start();
        }
    }
    else
    {
        logging_status = false;
        logging_button.Text = "Start Logging";
    }
}

// Purpose: Sample sensors once and deactivate sampling
    until interval is passed
// Version: 30/01-24
// Author: Torstein Solheim berg
private void SampleOnceButtonClick(object sender, EventArgs
    e)
{
```

```
    TakeSample();
    sampling_timer.Start();
    sample_once_button.Enabled = false;
}

// Purpose: Log samples collected once and deactivate
    logging until interval is passed
// Version: 30/01-24
// Author: Torstein Solheim berg
private void LogOnceButtonClick(object sender, EventArgs e)
{
    LogSamples();
    logging_timer.Start();
    log_once_button.Enabled = false;
}

// Purpose: Sample or reactivate sampling once when sampling
    interval has passed
// Version: 30/01-24
// Author: Torstein Solheim berg
private void SamplingTimerTick(object sender, EventArgs e)
{
    if (sampling_status)
    {
        TakeSample();
    }
    else
    {
        sampling_timer.Stop();
        sample_once_button.Enabled = true;
    }
}

// Purpose: Log or reactivate logging once when sampling
    interval has passed
// Version: 30/01-24
// Author: Torstein Solheim berg
private void LoggingTimerTick(object sender, EventArgs e)
{
    if (logging_status)
    {
        LogSamples();
    }
    else
    {
```

```
                logging_timer.Stop();
                log_once_button.Enabled = true;
            }
        }

        // Purpose: Display Help message when Help is clicked
        // Version: 30/01-24
        // Author: Torstein Solheim berg
        private void HelpToolStripMenuItemClick(object sender,
            EventArgs e)
        {
            string first_line = "This is a program generating a
                dataset and saving it to a csv file.\n";
            string second_line = "You can either start a sampling
                session where a sample is collected\n";
            string third_line = "every 3.4 seconds, or sample
                yourself. But sampling is not allowed more\n";
            string fourth_line = "often than every 3.4 seconds.\nThe
                same goes for logging aswell, but";
            string fifth_line = "with an interval of 47
                seconds.\nYou can specify a name for the csv file";
            string sixth_line = "to log to, or the program will
                \nuse the time of the logging as the name file.";
            MessageBox.Show(first_line + second_line + third_line +
                fourth_line + fifth_line + sixth_line,
                          "Help Page", MessageBoxButtons.OK);
        }
    }
}
```

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

namespace SensorApp
{
    // Purpose: Simulate a sensor which can give acumulating sensor
        values between -5 and 5
    // Version: 29/01-24
    // Author: Torstein Solheim lberg
    public class Sensor(int sensor_number)
   {
```

```
  double value = 0.0F;
  readonly Random random_sensor_value = new(sensor_number);

    // Purpose: Return a new accumulated value. Adds a random
        value between -5 and 5 to the last.
    // Version: 30/01-24
    // Author: Torstein Solheim lberg
    public double GetNewValue()
  {
     double min_value = -5F;
     double max_value = 5F;

     value += random_sensor_value.NextDouble();
     return ((max_value - min_value) * value) + min_value;
  }
}

// Purpose: Dataclass for storing sensor data of 6 sensors
// Version: 30/01-24
// Author: Torstein Solheim lberg
public class Sensordata : IEnumerable<double>
{
    public string? Date { get; set; }
  public double _sensor_0 { get; set; }
    public double _sensor_1 { get; set; }
    public double _sensor_2 { get; set; }
    public double _sensor_3 { get; set; }
    public double _sensor_4 { get; set; }
    public double _sensor_5 { get; set; }

    // Purpose: returning sensors values in order
    // Version: 30/01-24
    // Author: Torstein Solheim lberg
    public IEnumerator<double> GetEnumerator()
    {
        yield return _sensor_0;
        yield return _sensor_1;
        yield return _sensor_2;
        yield return _sensor_3;
        yield return _sensor_4;
        yield return _sensor_5;
    }

    // Purpose: Making the dataclass iterable
    // Version: 30/01-24
```

```
  // Author: Torstein Solheim lberg
  IEnumerator IEnumerable.GetEnumerator()
  {
      return GetEnumerator();
  }

  // Purpose: Making the data class indexable
  // Version: 30/01-24
  // Author: Torstein Solheim lberg
  public double this[int index]
{
   get
   {
         return index switch
         {
             0 => _sensor_0,
             1 => _sensor_1,
             2 => _sensor_2,
             3 => _sensor_3,
             4 => _sensor_4,
             5 => _sensor_5,
             _ => 0,
         };
     }
     set
   {
         switch (index)
         {
             case 0:
                 _sensor_0 = value;
                 break;
             case 1:
                 _sensor_1 = value;
                 break;
             case 2:
                 _sensor_2 = value;
                 break;
             case 3:
                 _sensor_3 = value;
                 break;
             case 4:
                 _sensor_4 = value;
                 break;
             case 5:
                 _sensor_5 = value;
```

```
                    break;
                default:
                    break;
            }
        }
    }
}
```