University of
South-Eastern Norway

Faculty of Technology, Natural Sciences and Maritime Sciences
(TNM) - Campus Porsgrunn

Industrial Information Technology and Automation (IIA)
Master Programme

Course IIA1319
Software Engineering
(Object-Oriented Analysis, Design, and Programming)

C# coding Assignment

Version 1.0

January 20, 2021

Chapter 1

C# coding assignment

1.1 Introduction

This assignment is an introduction to C# programming with the goal to make you a little more familiar with 1) classes, objects, and object-oriented programming, and 2) version control system. The basis for the coding will be an application simulating a DAQ (Data Acquisition) device with sensor devices connected. You must have a C# IDE (Integrated Development Environment) supporting a GUI (Graphical User Interface) available, any version of the Microsoft Visual Studio and the Git version control system. If you are new to the Git system please read at least the “Getting Starting” section (Chacon 2009) explaining the configuration and the stage/commit file states. You should make you own local repository, not using a ready-make GitHub repository.

This assignment must be delivered as a report within the deadline (due time) given in *the learning management system (LMS)* and must be approved in order to get a grade better than F in this course. Minimum 60% of the tasks must be completed to get the assignment approved. The report should be a technical report containing at least the sections 1) Introduction, 2) results, 3) conclusion and 4) appendix, but please keep it as short as possible. Minimize the usage of other sources, but use references whenever you do that. The report should be only one file with the figures and appendix included, preferable in the PDF file format. *Let your name or student number be part of the filename.*

Normally the task description should be included as Appendix A in such a report, but no need to include the task description in any assignment report. A technical report should however always contain listings of any developed source code, in an appendix. Focus on your own code, not the code generated from the IDE.

This assignment is designed for students not so experienced in programming. If you look at yourself as an experienced programmer using the C# programming language, knowing how to make a GUI and using objects, see Appendix A.5 if you want to make your own application. The versions of this document are:

Version	Description	Date
0.1	First version of the document (HiT)	NOS / 2006
0.2	Add more information about the DAQ simulator	NOS / 2008
0.3	Better description of sampling and logging	NOS / 2012
0.4	Updated the text for approved / not approved	NOS / 2013
0.5	Include the usage of a version control system	NOS / 2014
0.6	Include the low pass filter functionality	NOS / 2015
0.7	Include option for experienced programmer application	NOS / 2016
0.8	Minor updates of text and institution (HSN)	NOS / 2017
0.9	Minor updates of text and institution (USN)	NOS / 2020
1.0	The user can choose between Form or WPF type of GUI	NOS / 2021

1.2 Assignment setup (2.5%)

Download and run the SCE1306DaqCoding.exe application from *LMS*, input your name, student number and the course year, and click the *Get Assignment Information* button. This application will give you a

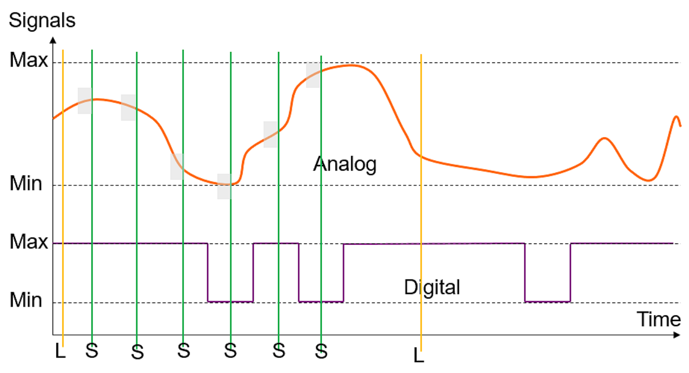


Figure 1.1: The sampling (S) and logging (L) of analog and digital signals. The analog signals should be filtered, indicating by the grey box just before the sampling.

set of parameters that you will use as basis for developing your own DAQ simulation applications. See Appendix A.1 for more information about the SCE1306DaqCoding application.

- **Include a screen shot of the SCE1306DaqCoding application in your report.**
- Select an empty folder and init the Git repository at this folder. Configure the repository by adding your name and email address (as described in the First-Time Git Setup documentation).
- Start the IDE and make a C# application in this folder. The application should be either a Windows Form or Windows Presentation Foundation (WPF) project for Windows Desktop. Windows Form is more easy to develop while WPF is more general. Source code snippet for message box in the appendix is for windows form type of application.

1.3 DAQ Simulation Application (10%)

The DAQ Simulation Application is an application used for manually sampling and logging of sensor values. You may need such an application for testing of a Process Control System (PCS) without any sensor devices connected to the PCS, and need to develop this application as part of the PCS testing. The application will collect sensor values at each sampling time, filter any analog sensor values using a low pass filter. At the logging time the filtered sensor values should be written to a CSV (Comma Separated Values) text file that will be read by the PCS. The CSV file should append the sensor values, with one line for each of the data sets.

The application will simulate a set of sensor values, the number of sensor values is defined by the SCE1306DaqCoding application. At specific time intervals the sensor values should be sampled, the requirement given in the SCE1306DaqCoding application is the minimum time. A button is used for start sampling the sensor values, however the next sampling is not allowed until a new valid sampling time has passed.

At specific time intervals the sensor values should be logged on a CSV file, the requirements given in the SCE1306DaqCoding application is the minimum time. The logging time must be equal or higher than the sampling time. A button is used for start of logging the sensor values, however the next logging is not allowed until a new valid logging time has passed. The current date may be included in the file name, given by the SCE1306DaqCoding application. This option can be useful for making several test files for the PCS.

Figure 1.1 shows the sampling (S) and Logging (L) of an analog and digital signal, where the analog signal should be filtered before sampling (indicated by a grey box). A filtered value of the analog signal should be available at the sampling time, based on the information as close as possible to the sampling time. The logging operation (L) should log either the available logging values since last log operation or only the last logging value, your decision.

A proposal for a minimum requirement for the user interface is given in Figure 1.2. However, you are advised to make your own UI design. The sampling box contains a field for the next sampling time and a button for make a sampling of all the sensors devices. The Logging box contains a field for the next logging time and a button for logging the filtered sensor values to a CSV file.

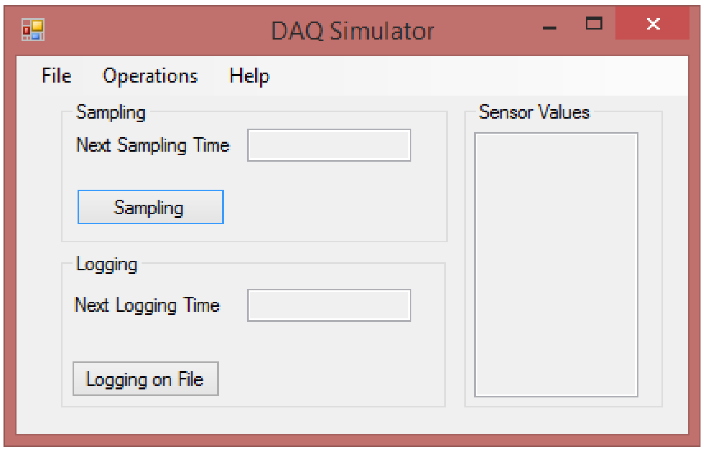


Figure 1.2: A proposal for the GUI of the DAQ Simulator Application. Two buttons, one for sampling and one for logging. Three text fields, the next sampling time, the next logging time and the current sensor values.

- Make a graphical user interface for the application supporting the requirements in the SCE1306DaqCoding application, compile and run the application. The sensor value text box is a standard text box with the Multiline property set to TRUE. Make a screen shot for your report.
- Add the files to the Git repository. Use a couple of sentences to describe this operation and indicate the Git command(s) you have used for this operation.

1.3.1 Sampling Sensor device values (30%)

- Make a class for the sensor devices according to the requirements given in the SCE1306DaqCoding application. Include the code for correct voltage range of the sensor device. Use method headers for documentation and include the source file an appendix in your report. The appendix gives an example of a sensor class, however without method headers. Method headers MUST be included in your source code.
- Make the code for supporting the Sampling button. The sampling button should 1) check for a valid sampling time interval by using the NextSampling Time field, 2) read all the sensor values, 3) display the sensor values in the sensor Values group window, 4) update the Next Sampling Time field. You can choose if you want to have only a single sampling or continuos sampling until the button is pressed again. The appendix gives an example of collecting data from a group of sensor devices.
- Update your Git repository with the new files and changes.
- Make a short comment, maximum 5 lines, about sampling digital sensor values.

1.3.2 Help About Information (5%)

- Make a Help About Information box for your application. See the appendix for an example of a MessageBox.

1.3.3 Logging Sensor device values (30%)

- Make the code for supporting the Logging button. The Logging button should 1) check for a valid logging time interval by using the NextLogging Time field, 2) write the last read sensor values to a text file, include the time and append to the file contents if the file already exists, 3) include a field showing the file name and number of writings, 4) update the Next Logging Time field. You can choose if you want to have only a single logging or continuos logging until the button is pressed again. Include the text file (CSV) in an appendix in the report, but list a couple of the lines in the report to document the format of the file.

- Explain briefly (maximum half a page) how you could use a low pass filter as the interconnection between the sampling and logging methods.
- Update your Git repository with the new files and changes.

1.3.4 Documentation (12.5%)

- Make a flow diagram documenting your application.
- Indicate the number of classes and objects used in your application.
- List the history of the Git repository. Tip: Some of the available commands are Git log or gitk. Include a screen shot to document your results.

1.4 Summary (5%)

- Make a short conclusion of this assignment.

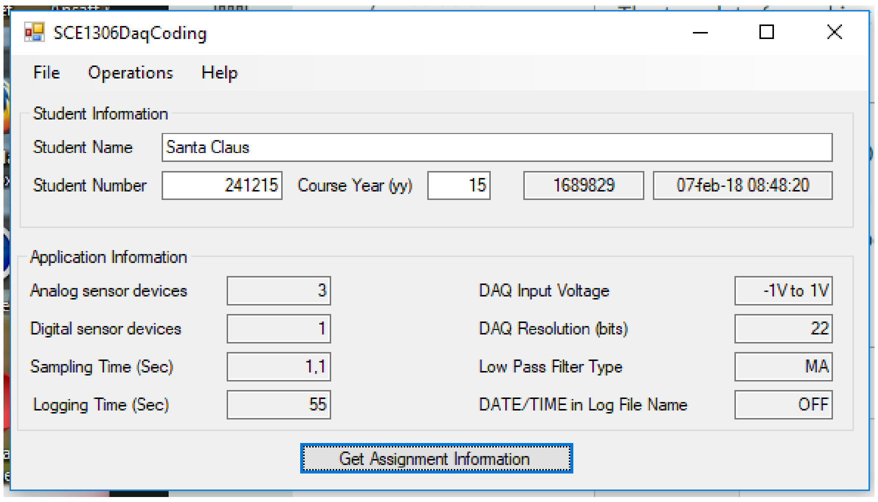
Appendix A

Appendix

This appendix shows an example of running the SCE1306DaqCoding application and some source code in C#. The lecture notes ((Skeie n.d.)) contains sections about the usage of Visual Studio, short description of the C# programming language, and more C# source code listing.

A.1 SCE1306DaqCoding application

A screen shot of the SCE1306DaqCoding application with an example of the information for your exercise is shown in Figure A.1. Remember that the assignment information will be different for each student, you must download the application and run the application on your computer. The application can be downloaded from the assignment section of the learning management system. **Important**: remember to make a screen shot for your report!



Student Information

Student Name: Santa Claus
Student Number: 241215
Course Year (yy): 15
1689829
07-feb-18 08:48:20

Application Information

Analog sensor devices	3	DAQ Input Voltage	-1V to 1V
Digital sensor devices	1	DAQ Resolution (bits)	22
Sampling Time (Sec)	1,1	Low Pass Filter Type	MA
Logging Time (Sec)	55	DATE/TIME in Log File Name	OFF

Figure A.1: The *sce1306daqcoding* application with a set of requirements. The white fields are the input fields and the requirements will depend on the input field information.

A.2 List of sensor objects

This section shows an example of making and referring 16 sensor objects using an array in C#.

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

```
namespace SensorApplication
{
    ////////////////////////////////////////////////////////
    class Sensor
    {
        double dVal;
        int sId ;
        Random rSenVal;

        public Sensor(int id)
        {
            sId = id ;
            rSensVal = new Random(id) ;
            dVal = 0.0F ;
        }
        public double GetValue()
        {
            dVal += rSensVal.NextDouble() ;
            return dval;
        }
        public int GetSensId()
        {
            return sId;
        }
    }
    ////////////////////////////////////////////////////////
    class Program
    {
        static void Main(string[] args)
        {
            int counter, maxSid = 16 ;
            string sTxt ;
            // Create an array of sensor objects
            Sensor[] sObj = new Sensor[maxSid];
            for (counter = 0; counter < maxSid; counter++)
            {
                sObj[counter] = new Sensor(counter);
            }
            // Get the object values as a string
            for (counter = 0; counter < maxSid; counter++)
            {
                sTxt = sObj[counter].GetValue().ToString("F3") ;
            }
        }
    }
}
```

A.3 MessageBox in Visual Studio

```
// Make a message box if any of the text fields are empty
private void helpToolStripMenuItem1_Click(object sender, EventArgs e)
{
  // Make the messagebox
  MessageBox.Show("Please input text in all input fields (use your own text here!!!).",
                  "Input Information", System.Windows.Forms.MessageBoxButtons.OK);
}
```

A.4 Design Comments

Often these simulators are designed using a set of threads to do the automatic sampling and logging. In this case a manual approach has been chosen to simplify the coding of the application. Anybody wants to use threading and automatic sampling and logging of sensor data is free to do that. However a manual simulator can also in some situations be better to use for testing than an automatic simulator.

A.5 Experienced programmer

Any experienced programmers in the C# programming language can use this assignment to make their own application since the focus is programming. The application must contain a GUI, at least six classes, at least two objects of one class, and configuration data from a text or XML file. Extend the introduction of the report with a specification of the application, use Git for version control, and include some sort of flow diagram for documenting the application.

Bibliography

Chacon, S. (2009), *Pro Git; Description of the Git version control system.*

Skeie, N.-O. (n.d.), 'IIA1317: Object-oriented analysis, design, and programming', Lecture notes for the master course IIA1317 at the University College of Southeast Norway (USN) (and SCE1306 at the Telemark University College (TUC)). The Unified Process (UP) as the software process, the Unified Modeling Language(UML) for Object Oriented Analysis and Design, and the C-Sharp programming language for implementation.