University of South-Eastern Norway

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

Industrial IT and Automation (IIA)
Master Programme

Course IIA2017
Industrial Information Technology

Data Communication and Digitalization Assignment

Version : 0.52

© Nils-Olav Skeie

February 20, 2023

Chapter 1

Data Communication assignment

1.1 Introduction

This assignment contains several tasks to be carried out on your computer, preferably a lap-top computer, since wireless communication is part of the assignment. A stationary computer can be used if a lap-top computer is not available. You should use the same computer for the whole assignment and the report should start with some general documentation of this computer like brand, type of computer (lap-top or stationary), and the operating system in use. You should also state the reason(s) for using a stationary computer, if used.

The report will consists of several screen shots from your computer together with your own explanations of the contents in these figures. If the screen shots contain a language different from English or Norwegian, the screen shots should be post processed in some sort of presentation program by adding arrows and English text for the important fields.

The motivation for this assignment is to get some experience in getting information of the I/O devices of your computer (data communication), getting information about the network devices and the setup of these devices, being able to test the network part(s) on a computer, some information about cloud systems and working with digitalization. Some of the parts will include programming, using the C# programming language. The code should be included as appendices.

The version history of this document:

Version	Description	
0.1	First version, including a student application.	NOS−08
0.2	Extend document with more report specifications.	NOS−11
0.3	Extend document and application configuration and analysis part.	NOS−13
0.4	Extend document and application with theory section.	NOS−14
0.5	Extend document with digitalization and cloud systems.	NOS−23
0.51	Minor clarifications of the required tasks and delivery parts.	NOS−23
0.52	Clarifications of csv, xml and json file formats.	NOS−23

1.2 Student information

Each student will have some specific parameters for parts of the tasks and these parameters will be given by the *DataCommStudentInfo* application. A screen shot of the application *DataCommStudentInfo* after the student name, student number and year of semester has been entered **must** be included in the report if the operating system is Windows. This application can be downloaded together with the task description and run on any computer. An example of a screen shot from this application is shown in Figure 1.1.

Note that this application will run only on Windows based computers. If your computer is using another operating system such as Max OS, Linux, Solaris or others, just document the type of operating system and use the parameters listed in the Figure.

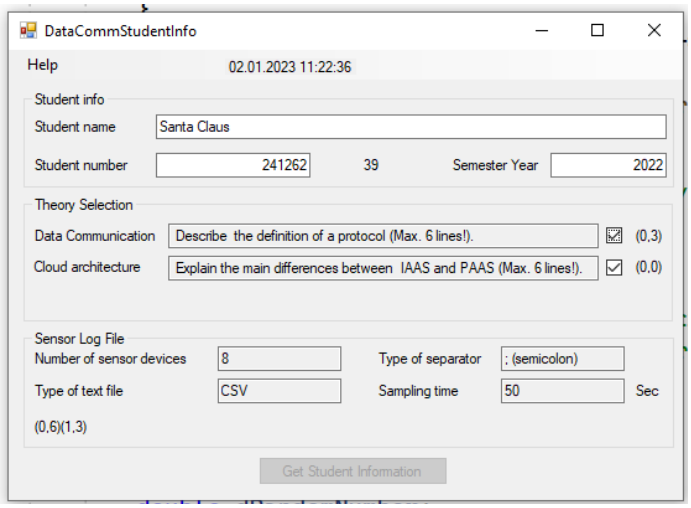

Figure 1.1: A screen shot of the DataCommStudentInfo application after entering the student information.

1.3 Assignment introduction

The report should be a technical report containing minimum an introduction, the results, and a conclusion. In a technical report the focus will be on the results, not the process of getting the results. The conclusion should focus on a summary of the results, your main findings from the tasks. All information should be documented by screen shots, each screen shot must contain a figure text, and each figure MUST be explained (referred) in the report. This will greatly affect the evaluation of the report. You should show the reader that you understand why a figure is included in the report, and the information in the figure. Use references whenever you copy information from any source, or need to document that others agree with your statement(s). A technical report must include references. It is considered cheating if you copy information without a reference, including copy source code from this document. The report should be delivered as a single pdf document within the time given in the Learning Management System (LMS).

The requirements for getting this assignment approved is that 1) **minimum** 65% of the tasks must be completed and approved, and 2) all tasks must be started with some type of analysis.

1.4 System information (2.5%)

All computers must have some sort of storage devices and serial ports. Include a set of screen shot(s) from your computer showing this type of information together with your own words explaining this information. Use the device manager to document:

- the interface protocol(s) between the motherboard of your computer and the storage devices, and the size of the storage devices.
- the number of serial ports like USB ports,
- any Bluetooth communication devices.

1.5 Network information (20%)

Computers normally contain both wired and wireless network interfaces. Some tasks about network description, devices and protocols:

- Answer the *Data communication* part in the *Theory Description* section from the *StudentCommStudentInfo* application.
- Use the device manager of your computer to document the number of network devices.

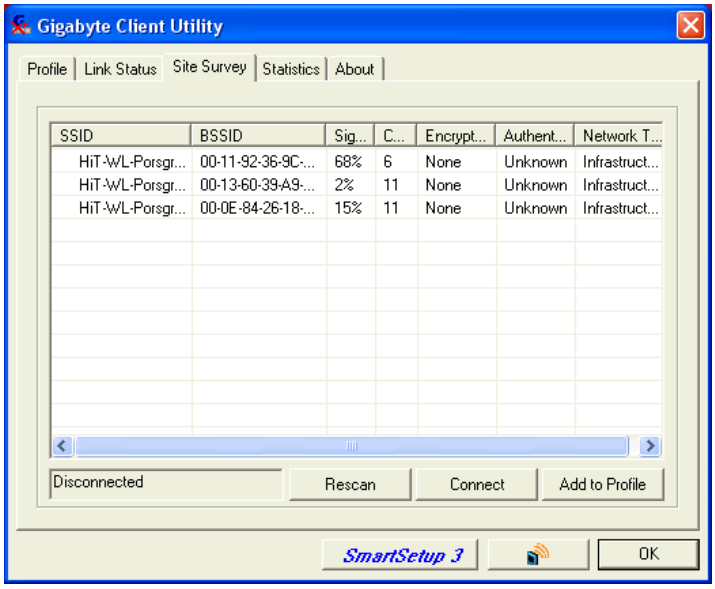

Figure 1.2: The figure shows the wireless networks (using the old HIT name), at a specific location at campus Porsgrunn.

Computers have different protocols for different layers, layers are here referring to layers in the OSI model. The data link layer is responsible for the physical addressing of the computers.

- What is the physical address of the active network for your computer?

The network layer is responsible for the logical addressing of the computers.

- What is the logical address of the active network for your computer?
- The TCP/IP protocol is often used, explain why this may be a TCP/IP address with class type, subnet masking and static or dynamic addressing.
- The TCP/IP protocol is necessary for accessing the internet. Have your computer any other network protocols installed?
- The TCP/IP protocol is based on two standards, TCP/IP v4 and TCP/IP v6. Have your computer support for the TCP/IP v6 standard?

1.5.1 Wireless

Figure 1.2 shows some of the wireless networks available at campus Porsgrunn.

- Which of the networks in the list in Figure 1.2 will you use for connecting to the campus LAN? Why?
- Do you need a password for connecting to this network?
- All the networks in the list in Figure 1.2 belongs only to the campus LAN. Why do we not see any other networks?

- Figure 1.3 shows a list of available wireless networks at another location in the area. Any assumptions about the owners of these networks, the location, and the security of these networks?
- Use your own computer to list the available wireless networks at your location.

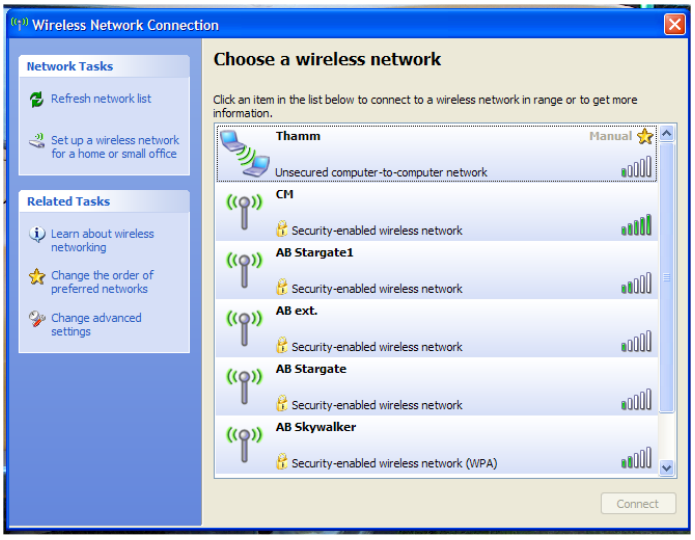

Figure 1.3: A list of available wireless networks.

1.6 Network testing (35%)

Network communication can be a challenge especially when designing, building and/or checking new systems. A set of tools that can be very useful for checking the network configurations and connections are the internal commands *ipconfig* and *ping*, and the application "WireShark"[1]. The ping command is testing the lower layers of a network connection, a very useful tool for checking physical network connections, and is simple to implement using the Visual Studio and C#. Implementing your own ping application can be useful if you want or need some special options, or want to integrate a network test utility into your process control system (PCS).

- Use the internal ping command of your operating system to check the communication to a remote node, like your favorite web site. Comment on the response time.
- Use this ping command to check the communication to the same remote node, but running as "continuos" pinging. Use WireShark to trace this specific message. Indicate the TCP/IP address of the remote node.

Figure 1.4 shows the information from the WireShark application. Select a specific network device and a specific message to get details for this specific message.

The appendix *A* shows the C# code for a ping application, implemented as a console application. Define a C# console application in Visual Studio and copy the source code into this project in Visual Studio to make your own application. Use your own information as for the *data* and the *host* strings. Figure 1.5 shows the output from this application running on a computer.

Figure 1.6 shows the information from WireShark when tracing both the request and reply of the ping message to the remote node. Also notice the that the message include in the ping message is available as plain text in WireShark as it will be in any other application tracing the TCP/IP communication. This also shows how easy it is to "hack" into a TCP/IP communication and have a copy of the information sent between several nodes on internet.

Required tasks for the ping tool:

- Test the application with your own data string on your favorite web site. Comment on the response time. Include the source code in the appendix of your report.
- Use WireShark to show that another application can get a copy of the data string sent in your ping requests.
- Use pseudocode to document the working of a ping application for accessing the same web site a number of times and estimate the average response time.

[1] The WireShark application can be downloaded from www.wireshark.org (dec-16).

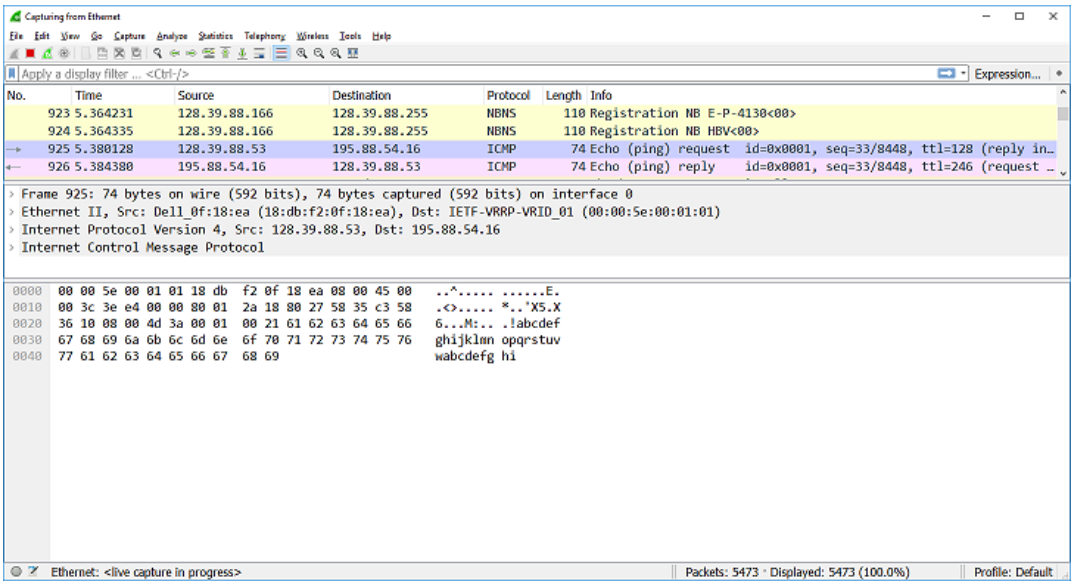

Figure 1.4: Using WireShark to check a ping request. Notice the there will be three windows, one with all the messages on a specific device, one window with the network details for the specific message, and the lowest window with details for the specific message.

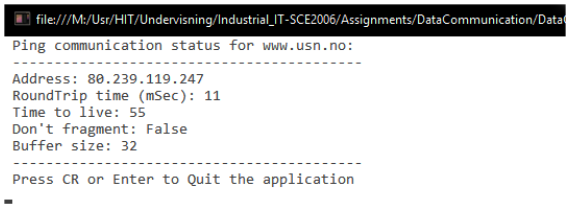

Figure 1.5: The ping application running on a computer testing the communication with the www.usn.no domian server.

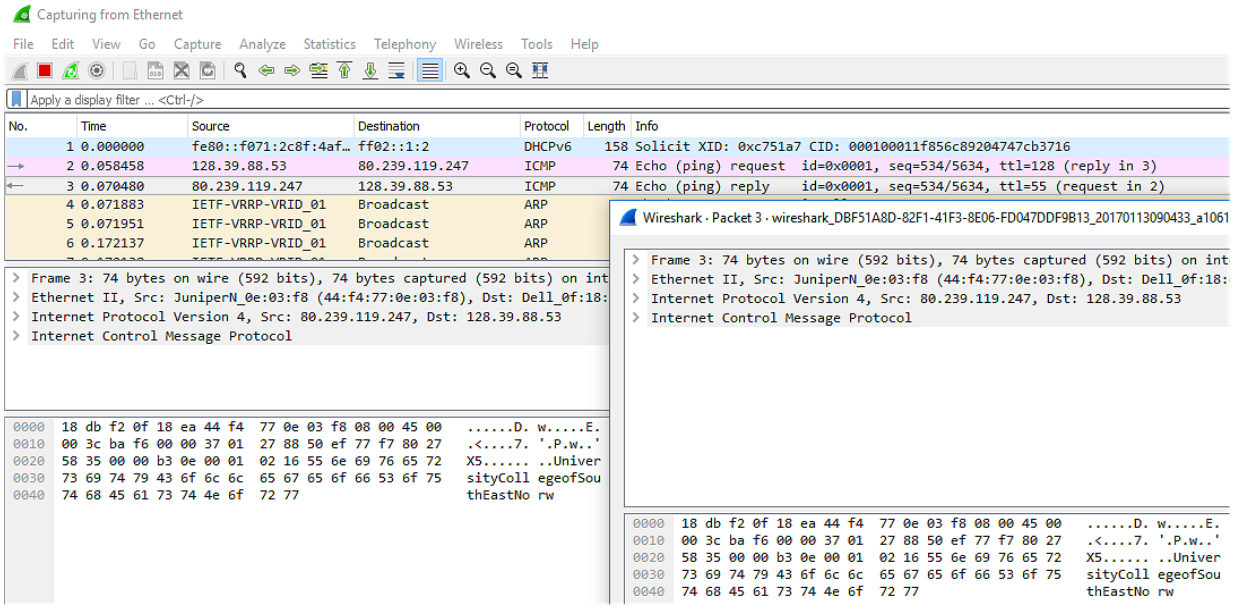

Figure 1.6: Using the WireShark to trace the ping message from the developed ping command. Tracing both the request and trace, and the text message is avaialble as plain text.

- Use pseudocode for document the working of a ping application for checking all the nodes in your network segment.
- Implement the source code for checking the number of nodes in your network segment, and include the source code in the appendix of your report.

Students looking for some programming challenges can extend the application with a GUI, buttons for sending a single ping message, continuous ping messages or sending to several nodes at the same time (with threads), and showing the results from the communication. Note however that the GUI extension is not a requirement for this assignment.

1.7 Digitalization (30%)

Digitalization is about sharing and reusing information between applications, systems and companies. A database is a good way of storing information but may not be the best solution for digitalization, in stead sharing data on text formats should be an additional feature. The main part in this section is to develop a C# application generating and logging a set of sensor values on a text file, one line for each sample. Each sample should contain the time stamp and the set of sensor values. An extract of the text file must be included in the report. The number of sensors, the sampling time, the type of separator between the fields in the text file and type of text file is given in the DataCommStudentInfo application. Assume only analog sensor devices and use the Random() method to get different sensor values for each sample.

- Develop the C# application; some example code is listed in the appendix of how to use the random() method and make an array of sensor objects; Hints: make the sensor objects first, then use a main loop collecting the sensor values, make a string, write the string to a file, and wait the sampling time before repeating.
- Document your software with some type of flow chart,
- Discuss any usage and implementation of a low-pass filter for the sensor values in your c# application,
- Explain the difference of the cvs, xml and json text formats and verify the format from your c# application,
- Any comments about data security for your application,

1.8 Cloud systems (10%)

Computers today are normally connected to both a local area network (LAN) and internet so cloud storage, cloud systems and cloud services are becoming more and more common. The cloud can be used for collection information, hosting software applications and running different type of services to mention some. The “cloud” based tasks are:

- Answer the *Cloud architecture* part in the *Theory Description* section from the *StudentCommStudentInfo* application.
- Elaborate how your C# application from the digitalization part can be extended with:
  - collecting sensor values from a cloud service,
  - storing the results in a cloud based storage system.

1.9 Conclusion (2.5%)

End your report with a short discussion/conclusion about the communication capabilities of this computer and the assignment. Note that at least 65% of the tasks MUST be completed and approved to get this assignment approved. Make a self evaluation of your effort for each sections and include the following table with your evaluation in the report. The weighting of the tasks are:

Sections	Weighting (%)	Your evaluation (%)	Review (%)	Comments
System Information	2.5			
Network Information	20			
Network testing	35			
Digitalization	30			
Cloud systems	10			
Conclusion	2.5			
Sum	100			

Include the percentages of the tasks that you have included in your report.

Appendix A

Ping source code based on Microsoft MSDN[1] information.

```
//////////////////////////////////////////////////////////////////////
///
/// PingAppConsile: application testing the ping() connection to a node
///
/// Based on code form Microsoft MSDN about the ping() command
///
/// Version: 1.0: 8-JAN-17: NOS
///
//////////////////////////////////////////////////////////////////////
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;
using System.Net;
using System.Net.NetworkInformation;
//
namespace PingAppConsole
{
    class Program
    {
        /// <summary>
        /// ////////////////////////////////////////////////////////
        static void Main(string[] args)
        /// Purpose: the main function in the application handling the
        ///          ping()communication
        /// Version: 1.0: 8-JAN-17: NOS
        /// </summary>
        {
            string host, data;
            byte[] buffer;
            int timeout;

            Ping pingSender = new Ping();
            PingOptions options = new PingOptions();

            // Use the default Ttl value which is 128,
```

[1] MSDN: Microsoft Developer Network

```
            // but change the fragmentation behavior.
            options.DontFragment = true;

            // Create a buffer of 32 bytes of data to be transmitted.
            data = "UniversityCollegeofSouthEastNorw";
            buffer = Encoding.ASCII.GetBytes(data);
            timeout = 120;
            // Name or address of node to access
            host = "www.usn.no";
            PingReply reply = pingSender.Send(host, timeout, buffer, options);
            if (reply.Status == IPStatus.Success)
            {
                Console.WriteLine(" Ping communication status for {0}:", host);
                Console.WriteLine(" ----------------------------------------");
                Console.WriteLine(" Address: {0}", reply.Address.ToString());
                Console.WriteLine(" RoundTrip time (mSec): {0}",
                                  reply.RoundtripTime);
                Console.WriteLine(" Time to live: {0}", reply.Options.Ttl);
                Console.WriteLine(" Don't fragment: {0}",
                                  reply.Options.DontFragment);
                Console.WriteLine(" Buffer size: {0}", reply.Buffer.Length);
                Console.WriteLine(" ----------------------------------------");
            }
            else
            {
                Console.WriteLine(" Error connecting to network address/name {0}", host);
            }
            Console.WriteLine(" Press CR or Enter to Quit the application");
            Console.ReadLine();
        }
    }
}
```

Appendix B

Sensor value source code

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

namespace DigitalizationLog
{
    class Sensor
    {
        int sId;
        Random ranSensVal;
        public Sensor(int id)
        {
            sId = id;
            ranSensVal = new Random(id);
        }
        public double GetValue
        {
            get
            { return ranSensVal.NextDouble(); }
        }
        public int GetId
        {
            get { return sId; }
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            int counter, iMaxSid = 128;
            string sTxt;
            // Create an array of sensor objects
            Sensor[] sObj = new Sensor[iMaxSid];
            for (counter = 0; counter < iMaxSid; counter++)
            {
                sObj[counter] = new Sensor(counter);
            }
            // Get the sensor object values as a string
            for (counter = 0; counter < iMaxSid; counter++)
```

```
            {
                sTxt = sObj[counter].GetValue.ToString("F3");
            }
        }
    }
}
```