

<https://www.halvorsen.blog>



Industrial Datalogging and Monitoring

C# Project

Hans-Petter Halvorsen

Industrial Datalogging and Monitoring

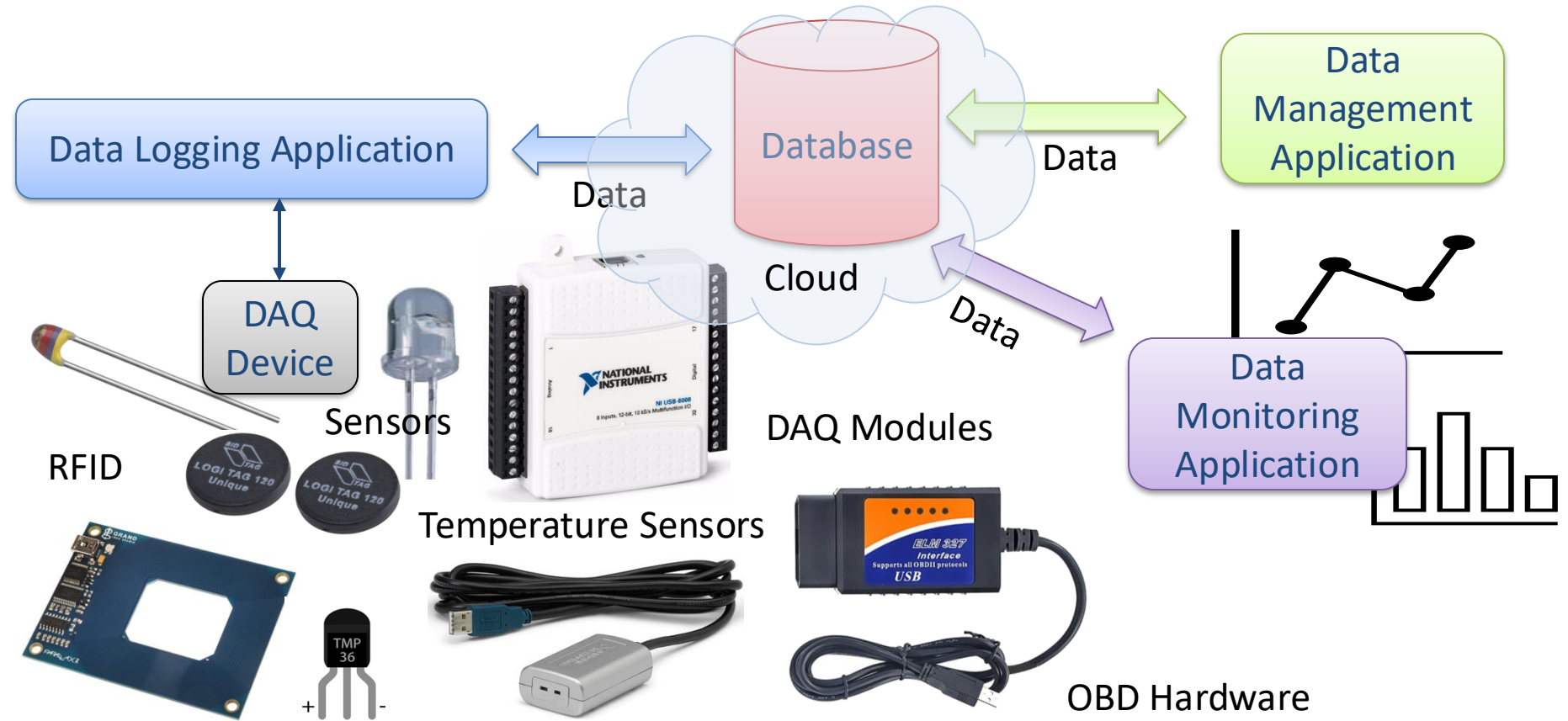


Table of Contents

1. [System Overview](#)
2. [System and Hardware Overview](#)
 - [System #1 - TC-01 Temperature Sensor](#)
 - [System #2 - USB-6008 DAQ Device](#)
 - [System #3 - RFID System](#)
 - [System #4 - OBD Car System](#)
3. [Database](#)
4. [Applications](#)
 - [Data Management Application](#)
 - [Data Logging Application](#)
 - [Data Monitoring Application](#)
5. [UML](#)
6. [C# Code](#)
7. [Final Delivery](#)

Industrial Datalogging and Monitoring

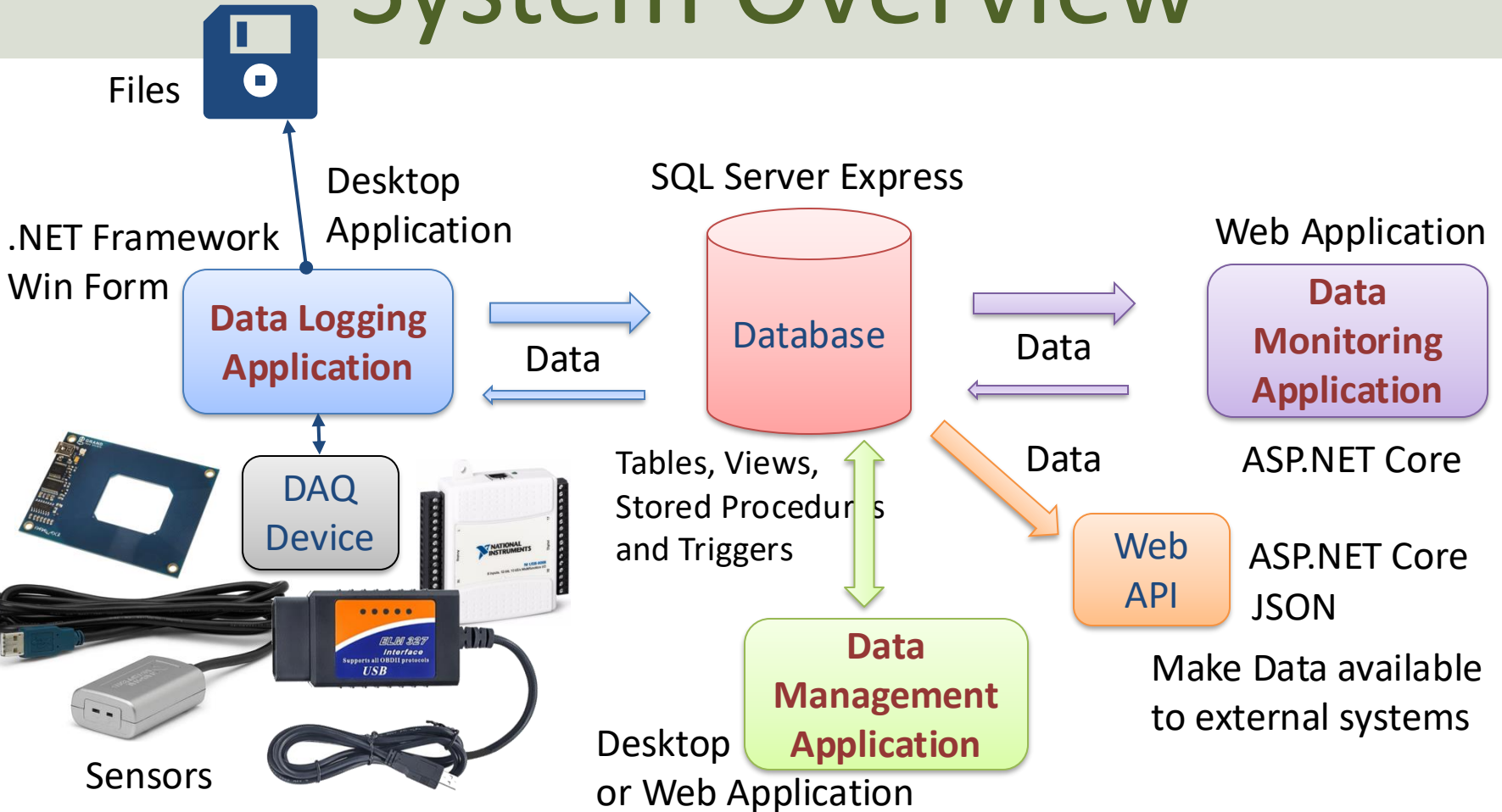
System Overview

<https://www.halvorsen.blog/documents/teaching/courses/csharp>

Project Description and Goal

- The system should log and monitor data from a data collection module (either using a TC-01, an USB-6008, an RFID Reader or an OBD module).
- The system must log data to file and database.
- The system should be module-based so that you have different modules for data configuration/management, data logging and monitoring.
- This document contains the overall specifications for the Industrial Datalogging and Monitoring System given by the customer, the detailed specification should be made by you because the customer has no skills in programming.
- The complete data system with documentation should be delivered to the customer by the end of the project.

System Overview



Select on of these Systems

You can choose between the following DAQ systems:

- **System #1:** Logging Data from **TC-01** Thermocouple Temperature Sensor.
- **System #2:** Logging Data from different Sensors (Temperature, etc.) using a **USB-6008** DAQ device.
- **System #3:** Read and Log **RFID** Data.
- **System #4:** Logging Data from a Car using an **OBD** module.

Hardware

System #1 (TC-01)

Temperature Sensor

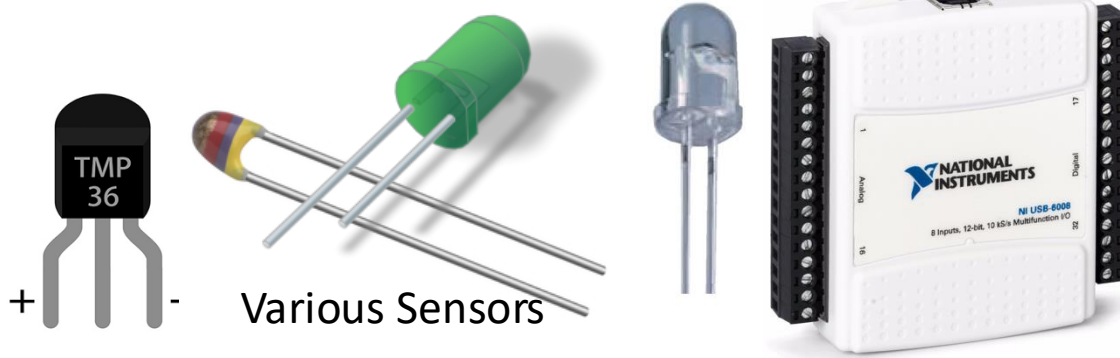


System #4 (OBD)



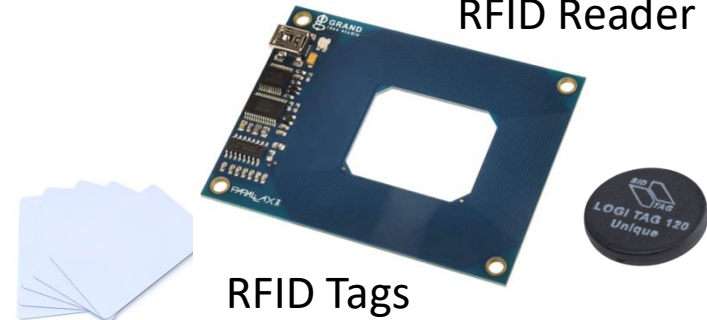
System #2 (USB-6008)

Various Sensors



System #3 (RFID)

RFID Reader



System Description

- The system shall log and monitor values from different Devices and Sensors.
- The sensors can be inside a building, inside a room, inside a house or a factory, in a vehicle, etc. It can be inside or outside.
- We can log temperature data or data from other sensors, like air quality, humidity, vehicle data, etc.
- Start simple and then try to make the system more flexible and general by adding more Tables, Classes, C# code and improved GUI.

Data Security

- Basic Data Security aspects of the system should be taken care of.
- The system should typically include basic Data Security, such as, e.g., Login using UserName and Password.
- You should make necessary considerations according to the General Data Protection Regulation (GDPR).
- In the report you should discuss the Data Security aspects, i.e., what you have implemented, but also other aspects that are relevant for such a system (e.g., What weaknesses does the system have? What can make the system more secure? What about GDPR?).

Project Assignment Guidelines

- Think about the Project Assignment as a small real-life industrial Project, and not a set of tasks, exercises or a “School Assignment”.
- What does the company that hire you expect from you when you deliver this project? What kind of Quality is expected?
- Try to see your work in a larger context than just an Assignment or a set of exercises.
- Try to see the big picture. The tasks within the assignment are just small building blocks that ends up with a fully working system.
- It is recommended that you make simple a Work Plan and a System Sketch that gives you an overview of WHAT you should do and WHEN you should do it. Start working on the Project Today! It is like a puzzle. You need to do it piece by piece.
- Practical Programming Skills: The only way to learn Programming is to do a lot of coding by yourself, and not only small examples and code snippets with a few lines of code. You need to make “large” Applications. It takes time and may be demanding, but that's the only way! The reward is knowledge that goes deep, and you will gain skills that are highly desired by the industry and work life.

Add Value!

- The Tasks described in the Project Assignment are somewhat loosely defined and more like guidelines, so feel free to interpret the Tasks in your own way with some **Personalized touch**.
- Feel free to **Explore!** Make sure to **Add Value** and **Creativity** to your Applications!
- Try to **Add Extra Value** and be creative compared to the simplified examples given by the supervisors, in that way you learn so much more.
- **Think Outside the Box!**
- Just don't follow a recipe - **Think like a Chef** that makes his own recipes. Add spices and ingredients and make it your own
- **The reward is knowledge** that goes deep, and you will gain skills that are highly desired by the industry and work life.

Deliveries

Final Delivery (at the end of the semester) in Canvas:

Deliver 3 separate parts:

- **Final Report** (PDF)
- **Video** (MP4). Focus: Live Demonstration of the System/Applications
 - **Note!** The video is only a supplement to the report. It should be possible to read the report and get a total overview of the system without seeing the video. This means you also need to have screenshots of the applications inside the report as well.
- **Source Code** (ZIP)

Microsoft Teams

- Do you need **Help**? Want to **Collaborate**? Want to **Discuss Technical Issues** with Others? **Share Knowledge**?
- Do you have Questions regarding the Project? We will use Microsoft Teams.
- In Microsoft Teams you can get help from one of the supervisors or from other students. You can chat, have video meetings, ask questions, respond to questions, etc. Basically, you can use Teams to communicate with the persons involved in this course.
- Very often someone else is wondering about the same as you - or perhaps someone else has experienced the same thing and found a solution for the problem? Then post information about this in the Teams room.
- Need help outside normal office hours? Perhaps a fellow student can help you if you ask your questions here? For example, if you have installation problems, etc., a fellow student can usually respond better than the supervisor can (outside scheduled hours, evenings, weekends, etc.). You also learn a lot from helping each other.
- You can also use Microsoft Teams for collaboration with other students.

System and Hardware Overview

Systems

- **System #1:** Logging Data from **TC-01** Thermocouple Temperature Sensor.
- **System #2:** Logging Data from different Sensors (Temperature, etc.) using a **USB-6008** DAQ device.
- **System #3:** Read and Log **RFID** Data.
- **System #4:** Logging Data from a Car using an **OBD** module.

TC-01 Temperature Thermocouple System



System #1 (TC-01)

Files

Desktop Application

SQL Server Express

Web Application

.NET Framework
Win Form

Data Logging Application

Data

Database

Data

Data Monitoring Application

TC-01

DAQ Device

Tables, Views,
Stored Procedures
and Triggers

Data

ASP.NET Core



Thermocouple

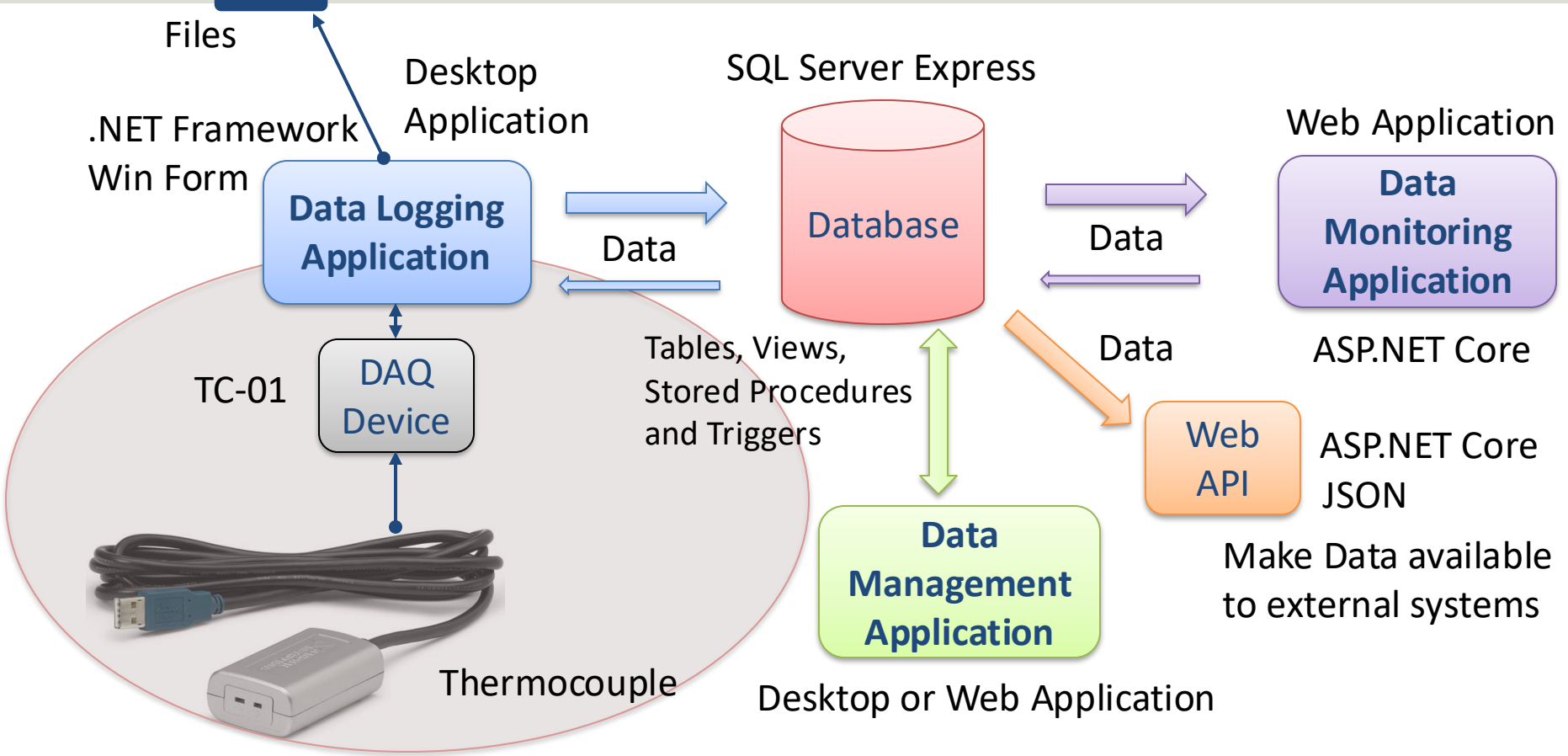
Data Management Application

Web API

ASP.NET Core
JSON

Make Data available
to external systems

Desktop or Web Application



System #1 – Hardware Overview



Box

USB A Cable



Device for connecting the Sensor to the PC

J Type Thermocouple Probe



TC-01 Temperature Thermocouple System

- **TC-01** from National Instruments
- TC-01 can be used to read the Temperature from the soundings
- You need to install the **NI-DAQmx** software in order to be able to communicate with the device from Visual Studio/C#



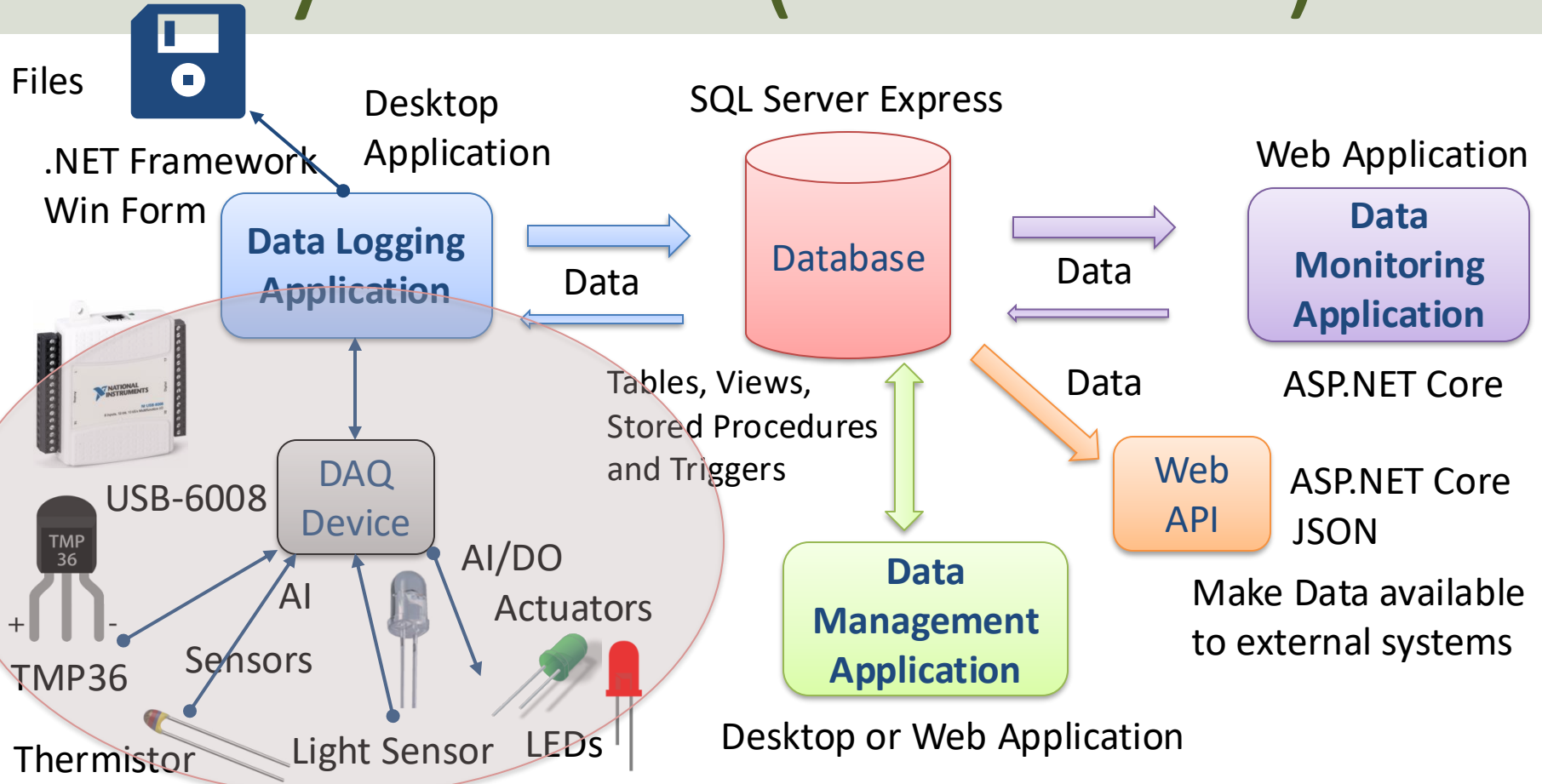
TC-01 Thermocouple



J-Type Grounded Probe Thermocouple

USB-6008 DAQ System

System #2 (USB-6008)



System #2 – Hardware Overview

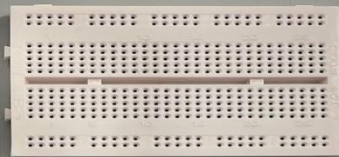
USB A-B Cable



NI USB-6008 DAQ Device



Breadboard



Bag with Sensors and electronic components



Screwdriver



Plastic Box



Pack with 10 Wires in different colors

USB-6008 DAQ System

- **USB-6008** from National Instruments can be used to acquire data from different types of Sensors
- You can, e.g., make a Home Automation System where you have placed one or more such DAQ devices in different rooms in your home and you have different sensors attached to these DAQ devices
- USB-6008 has Analog and Digital Input and Output Channels
- You need to install the **NI-DAQmx** software in order to be able to communicate with the device from Visual Studio/C#

USB-6008

Analog Channels

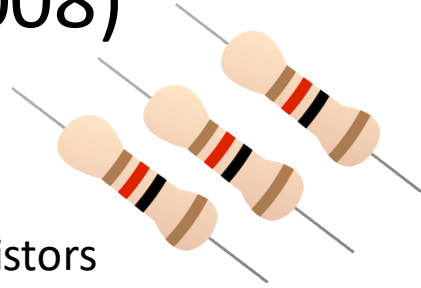


Digital Channels

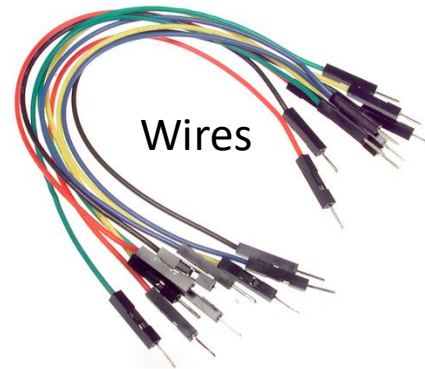
Hardware

- DAQ Device (USB-6008)
- Breadboard
- Wires
- Resistors
- Sensors and Actuators

Resistors

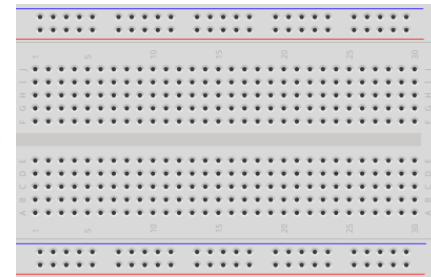


Wires



USB-6008

Breadboard



Sensors and Actuators

Light Sensor

Sensors*:

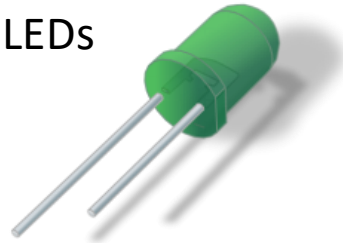
- TMP36
- Thermistor
- Light Sensor

Actuators

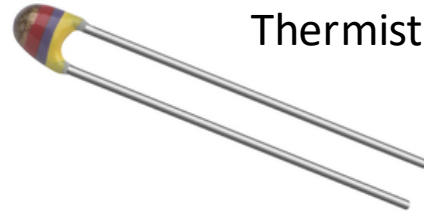
- LED



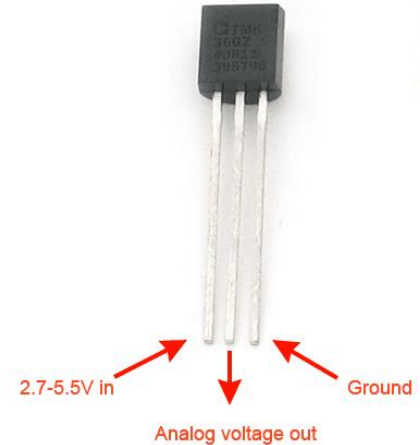
LEDs



Thermistor



TMP36 Sensor



*More Sensors can be used in addition if you want to make a larger system than the minimum required system

RFID System

System #3 (RFID)



Files

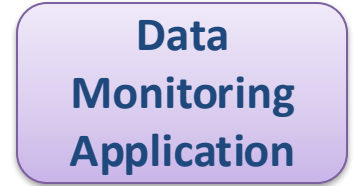
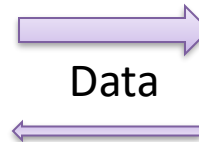
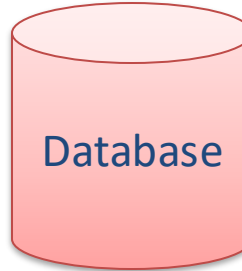
Desktop Application

SQL Server Express

Web Application

.NET Framework
Win Form

Data Logging Application



ASP.NET Core

ASP.NET Core
JSON

Make Data available
to external systems

USB RFID Reader

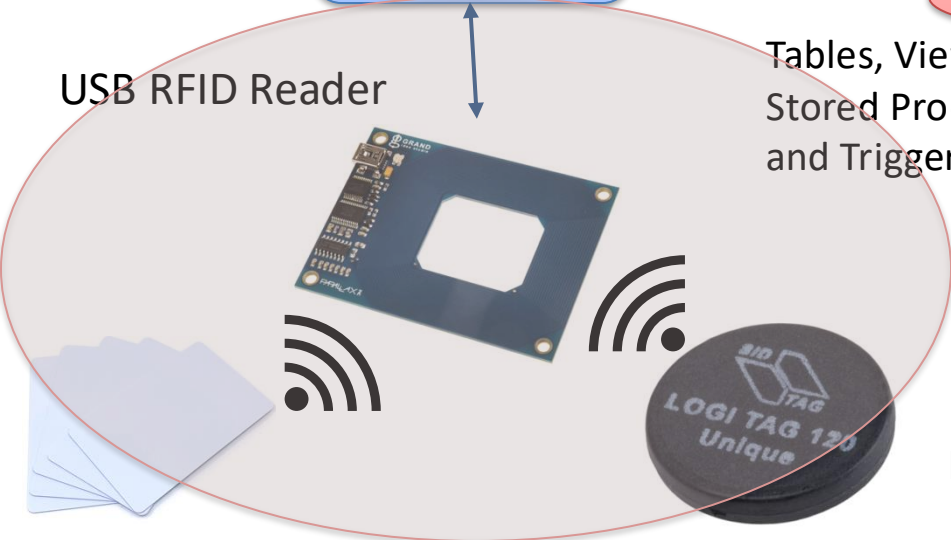
Tables, Views,
Stored Procedures
and Triggers

Data

Web
API

Data Management Application

Desktop or Web Application



System #3a – Hardware Overview



13.56MHz ISO 14443-A
(Mifare Classic 1k) Tags




RFID Reader with built-in Antenna

System #3b – Hardware Overview



USB-A to Mini-B Cable

RFID Reader with built-in Antenna



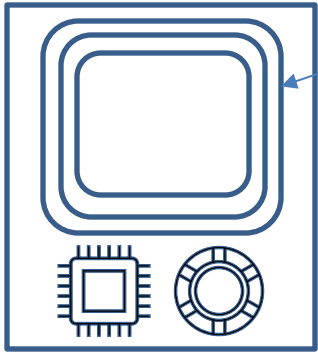
125KHz Tags in different shapes

RFID System

- You are going to read and log **RFID** Data
- It could be an inventory/library system for Tools, CDs, DVDs, Books or other Equipment (you choose)
- Add, Edit, Delete Equipment in the inventory (Data Management Application)
- Read RFID Tags and Log who borrows equipment, etc. (Data Logging Application) using RFID
- Show Lending History and Statistics, etc. (Data Monitoring Application)

RFID System

RFID Reader



RFID Antenna



RFID Tags



RFID Tags exist in many flavors and shapes

The RFID Reader is typically a Microcontroller

The Antenna is typically integrated within the RFID Reader, but you can also get external Antennas for better range



RFID

- RFID System is an abbreviation of Radio Frequency Identification System.
- It is a system for identification of items
- It uses using wireless communication that transfer data between Tags and the RFID Reader/Antenna
- We get RFID systems with different Frequencies
 - LF (125KHz), HF (13.56MHz), UHF
- We have Active and Passive Tags.
 - Passive tags are powered by energy from the RFID reader
 - Active tags have a battery

RFID Reader Example

```
using System.IO.Ports;
```

```
SerialPort port = new System.IO.Ports.SerialPort("COM4", 2400, System.IO.Ports.Parity.None,  
8, System.IO.Ports.StopBits.One);
```

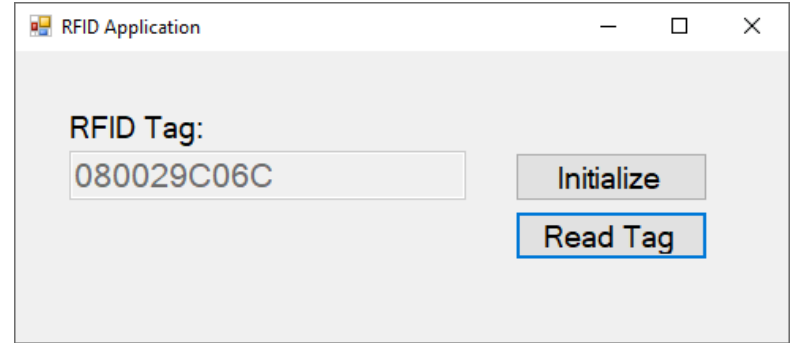
```
port.Open();  
port.DtrEnable = true;
```

```
int numberBytesToRead = 12;  
byte[] data = new byte[numberBytesToRead];  
port.ReadTimeout = 1000;  
port.Read(data, 0, numberBytesToRead);
```

```
string rfidTag;  
rfidTag = System.Text.Encoding.UTF8.GetString(data, 0, data.Length);
```

```
rfidTag = rfidTag.Replace("\n", "");  
rfidTag = rfidTag.Replace("\r", "");
```

```
port.Close();
```



OBD Car System

System #4 (OBD)



Files

Desktop Application

SQL Server Express

Web Application

.NET Framework
Win Form

Data Logging Application

Data

Database

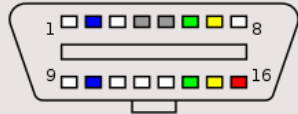
Data

Data Monitoring Application

Tables, Views,
Stored Procedures
and Triggers

Data

ASP.NET Core



OBD Module for self-diagnostics in vehicles

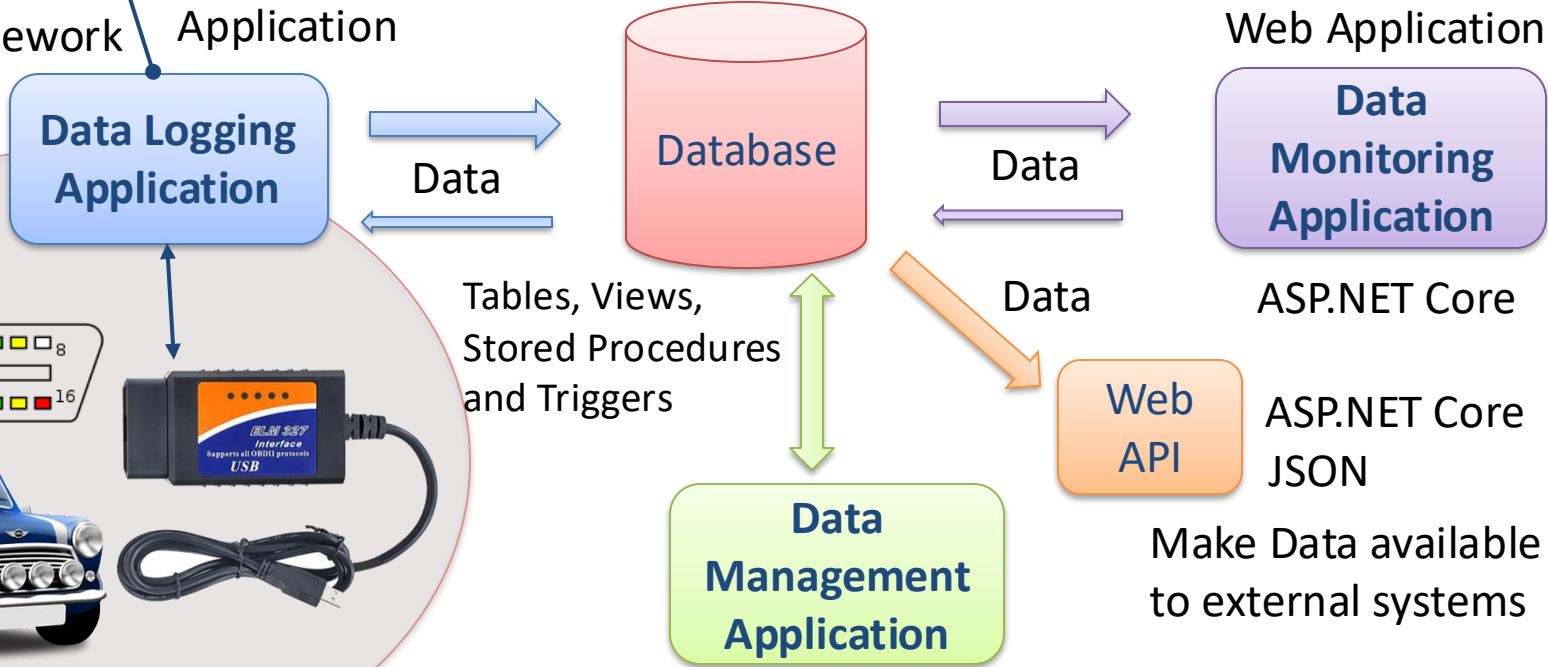
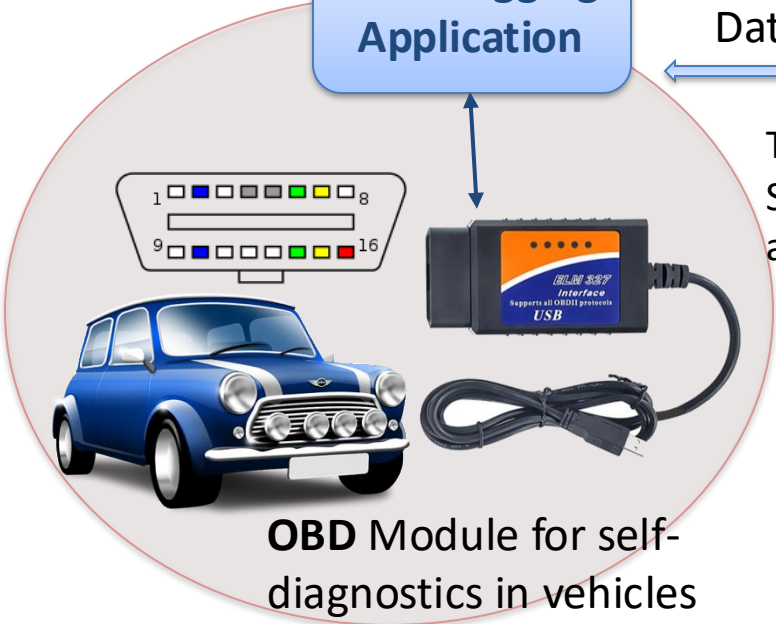
Data Management Application

Desktop or Web Application

Web API

ASP.NET Core
JSON

Make Data available
to external systems



System #4 – Hardware Overview



Please don't operate the system while driving the car. Create a system that don't need user interaction while driving the car or bring a friend that can operate the system.

OBD Vehicle Diagnostics Module



An OBDII Interface ELM327 with USB is recommended for connection to your PC.

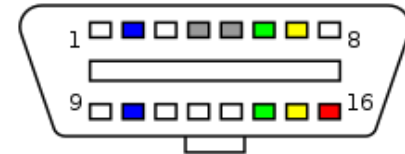
Bluetooth and Wi-Fi OBDII devices also exist but those are more cumbersome to connect to your PC and make an Application in C#

Price: appr. 150-300 NOK

<https://estore.no/feilkodeleser/497-elm327-elm-327-obd2-usb-bildiagnostikk-feilkodeleser.html>

OBD Module

- **On-board diagnostics (OBD)** is an automotive term referring to a vehicle's **self-diagnostic and reporting capability**.
- OBD systems give the vehicle owner or repair technician **access to the status of the various vehicle sub-systems**
- OBD implementations use a standardized **digital communications port to provide real-time data**
- A standardized series of **diagnostic trouble codes, or DTCs**, are also provided
- **OBD-II** is the latest standard used today

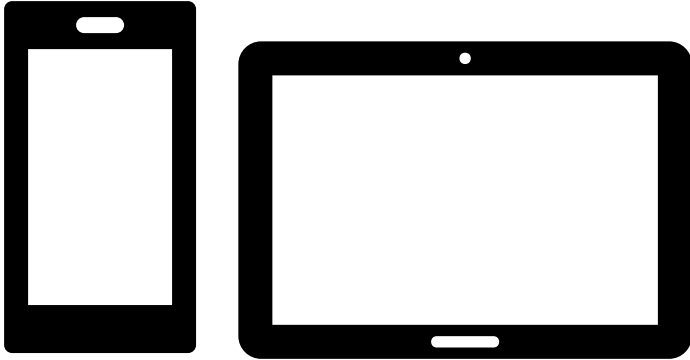


Getting Started with OBD

- On-Board Diagnostics, or “OBD”, is a computer-based system built into all vehicles from 1996 or newer
- OBD systems are designed to monitor the performance of some of an engine's major components including those responsible for controlling emissions.
- In other words, OBD is the language of the **Engine Control Unit (ECU)**, and it was designed to help fight emissions and engine failures.
- <https://learn.sparkfun.com/tutorials/getting-started-with-obd-ii/all>

OBD Apps and Diagnostic Tools

You can use one of the many OBD Apps for your Smartphone to get an overview of what kind of data you can expect to read and get into your C# Application



You may also get different Diagnostic Tools



These apps typically require a Wi-Fi or Bluetooth 4.0 (Bluetooth LE) OBD2 ELM327 compatible adapter (device) to work.

<https://estore.no/feilkodeleser/1943-can-obd-ii-feilkodeleser-ms300.html>

<https://apps.apple.com/us/app/car-scanner-elm-obd2/id1259933623>

Database

What should be stored in the Database?

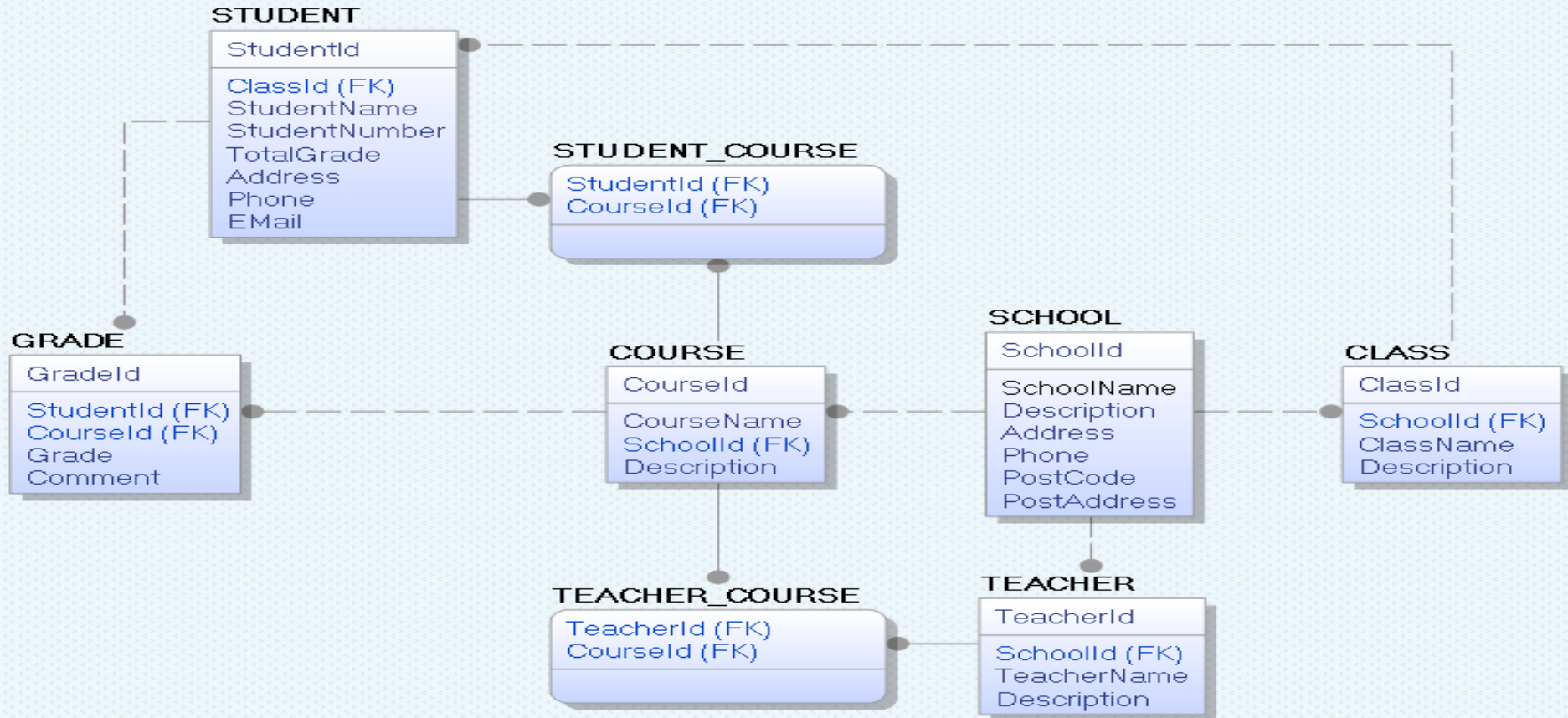
Here are some information that typically should be stored in the Database for such as system (these are only suggestions; you specify your own database structure):

- **Login** Information (typically UserName and Password)
- Information about the **DAQ Device(s)** (e.g., Name, Type, Vendor, etc.)
- **Logging** Information. How often shall you log data, etc.?
- Information about the **Sensors** (e.g., Name, Type, Vendor, Unit, Accuracy, Resolution, Max/Min Limits, etc.)
- **Measurement Data**: Data from the Sensors (at least value and date/time)
- **Statistics** (e.g., max, min, mean, standard deviation)
- **Alarms/Limits**: High and Low Temperature Alarm Limits

Database

- Install **SQL Server Express** and **SQL Server Management Studio (SSMS)**
- Use SSMS to create the following (which should be used in the C# applications)
 - Create at least 5 **Tables**
 - Create at least one **View**
 - Create at least one **Stored Procedure**
 - Create at least one **Trigger** (optional)

ER Diagram Example



Database Tips & Tricks

- Assume you have multiple logging devices in different buildings and rooms (or different Cars for System#3)
- Assume you have different logging devices that have different types and different numbers of sensors connected to it
- Assume you have different types of Sensors, like Temperature, Pressure, etc. In addition, you can have different types or categories of those, e.g., you have different types of Temperature sensors* like thermistors, thermocouples, RTDs, Semiconductor based ICs.
- Do we need Login (UserName, Password, etc.) and User Access (what kind of data should the different users have access to, who should be allowed to edit or delete information, etc.)?
- E-Mail information (when alarms occur)

* <https://www.digikey.com/en/blog/types-of-temperature-sensors>

Database - “Best Practice”

- **Tables:** Use upper case and singular form in table names – not plural, e.g., “STUDENT” (not students)
- **Columns:** Use Pascal notation, e.g., “StudentId”
- **Primary Key:**
 - If the table name is “COURSE”, name the Primary Key column “CourseId”, etc.
 - “Always” use Integer and Identity(1,1) for Primary Keys. Use UNIQUE constraint for other columns that needs to be unique, e.g. RoomNumber
- Specify **Required** Columns (NOT NULL) – i.e., which columns that need to have data or not
- Standardize on few/these **Data Types:** *int, float, varchar(x), datetime, bit*
- Use English for table and column names
- Avoid abbreviations! (Use RoomNumber – not RoomNo, RoomNr, ...)

Database Recommendations

- It is strongly recommended that you keep all your tables in a Table Script (Tables.sql). This Table script can be generated from erwin Data Modeler.
- When creating Views and Stored Procedures you should also create them in separate .sql files.
- Store all these files on your hard drive/OneDrive, etc.
- In that way you can easily create a new database with all the contents based on these .sql files.

Applications

C# Applications

You shall create the following Applications:

1. Data Management Application
2. Data Logging Application
3. Data Monitoring Application

C# Applications

1. Data Management Application
2. Data Logging Application
3. Data Monitoring Application

(They may be developed in random order or in parallel)

- The Database (SQL Server) and all C# applications should be running on your personal computer. Please don't use Microsoft Azure, because that is something we will focus on in the course "Software Engineering"
- Code structure: You can choose to have 3 different Projects in the same Visual Studio solution or 3 different Visual Studio Solutions (or both). Pros and Cons with these alternatives?

Why 3 Applications?

- Easier to develop 3 small applications than one large application (Module based development)
- Easier to share the development work if more than one developer
- Easier to find bugs and maintain the applications
- The applications are typically used by different people
- The applications uses different technology and frameworks (desktop, web)
- **Data Logging App**
 - Needs to be robust and it needs to be a **Desktop App** (not a Web App) because it should be running 24x7 without interruption. The GUI is not important because it should just be running without any interruption.
 - You only need to install it on one computer, and it is only used by a System Administrator
 - It needs to log data at specific intervals without any interruption
 - When it is running (without bugs), you don't need to touch it
- **Data Management App**
 - Used by only the System Administrator or Super Users
 - You use it to configure DAQ devices and sensors, when this is done, you typically only need it when you want to add more sensors or change some of the information
 - No "real-time" behavior is necessary (so a **Web Application** is a good choice here)
- **Data Monitoring App**
 - Used by many users (End users). Easier to use web because you then don't need to install it on many computers
 - It needs to be easy to use/user-friendly/intuitive because it will be used by many people typically with little skills in computers
 - No "real-time" behavior is necessary (so a **Web Application** is a good choice here)

Example: Microsoft Office is the name of the system. It consists of 3 Applications: Word, Excel and PowerPoint. Several applications have been added later, like Project, Skype, Teams, etc. What would it look like if everything was baked into one large application? Would it be user-friendly? Easy to use? Easy to develop? Easy to maintain? No!

Data Management Application

Data Management Example

BookApp Home Books

Read

Books

Below you see all the Books in the Book Store:

BookId	Title	ISBN	Publisher	Author	Category	Action
1	Introduction to Linear Algebra	0-07-066781-0	Prentice Hall	Gilbert Strang	Science	Delete Book
2	Modern Control System	1-08-890781-0	Wiley	Dorf Bishop	Programming	Delete Book
3	The Lord of the Rings	2-09-066556-2	McGraw-Hill	J.R.R Tolkien	Novel	Delete Book
4	Python					Delete Book

A typical Data Management Application implements the so-called CRUD, i.e. this means it **Create, Read, Update and Delete** data from a Database

Delete

New Book

BookApp Home Books

New Book

Create

Title:

ISBN:

Publisher:

Author:

Category:

Save

© 2019 - BookApp - Privacy

BookApp Home Books

Edit Book

Update

Title:

ISBN:

Publisher:

Author:

Category:

Save

© 2019 - BookApp - Privacy

© 2019 - BookApp

Data Management Example

Blog Device Management





Device Management

Data Management and Monitoring System

Devices

Devices are typically Sensor Nodes that include one or more DAQ devices or other IO modules.

List of Devices registered in the database:

Device Name
 Office
 Separation Rig
 TestDevice1
 Weather Station

[New Device](#)

Blog Tag Management

Tag Management

Data Management and Monitoring System









Tags

Devices are typically Sensor Nodes that include one or more Measurements, so-called DAQ devices or other IO modules.

Select Device:

Weather Station

List of Tags registered in the database:

TagName	Tag Alias
 mtAdjBaromPress	Barometric Pressure
 mtAdjWindDir	Wind Direction
 mtRainToday	Rain
 mtRelHumidity	Humidity
 mtSolarRadiation	Solar Radiation
 mtTemp1	Temperature
 mtWindChill	Wind Chill
 mtWindSpeed	Wind Speed

[New Tag](#)

Create New Tag

Select Device:

Not Specified

Tag Name:










Tag Alias:

Tag Type:

Air Quality

Unit:

[Save](#)




°C	Weather Station	  
°C	Weather Station	  
m/s	Weather Station	  

Data Management Example

Tool Management Tools Persons User

Tools

Below you see all the available Tools in the Inventory:

ToolId	Name	Tag	Status	Action
2	Hilti Hammer Drill	080029C06C	 	Details Delete Tool
5	Bosch Vacuum Cleaner	0800297F02	 	Details Delete Tool
6	Ryobi R18RS7-1 Bayonet Saw	080029CB9B	 	Details Delete Tool
7	Laptop	55555555	 	Details Delete Tool

New Tool

© 2021 - Hans-Petter Halvorsen

Tool Management Tools Persons User

New Tool

Tool Name*:

Tool Tag*:

Tool Description:

Save

Tool Management Tools Persons User

Edit Tool

Tool Name*:

Tool Tag*:

Tool Description:

Save

© 2021 - Hans-Petter Halvorsen

Data Management Application

- Create a Windows Forms Application, or preferably an **ASP.NET Core Web** Application
- It should be possible to show, insert and update information about the **DAQ Device(s)** (e.g., Name, Type, Vendor, etc.)
- It should be possible to show, insert and update information about the **Sensors** (e.g., Name, Type, Vendor, Unit, etc.)
- Set, e.g., Logging Interval, e.g., every minute or every 10. minute
- Set up what kind of sensors that are connected to a specific DAQ device.
- Support for multiple DAQ devices in different rooms and buildings?
- etc.

These are just suggested features

Data Management Application - Resources

“ASP.NET Core CRUD Application” Example

Video (YouTube): <https://youtu.be/k5TCZDwTYcE>

BookApp Home Books

Books

Below you see all the Books in the Book Store:

BookId	Title	ISBN	Publisher	Author	Category	Action
1	Introduction to Linear Algebra	0-07-066781-0	Prentice Hall	Gilbert Strang	Science	Delete Book
2	Modern Control System	1-08-890781-0	Wiley	Dorf Bishop	Programming	Delete Book
3	The Lord of the Rings	2-09-066556-2	McGraw-Hill	J.R.R Tolkien	Novel	Delete Book
4	Python	55555	Halvorsen	Hans-Petter	Programming	Delete Book

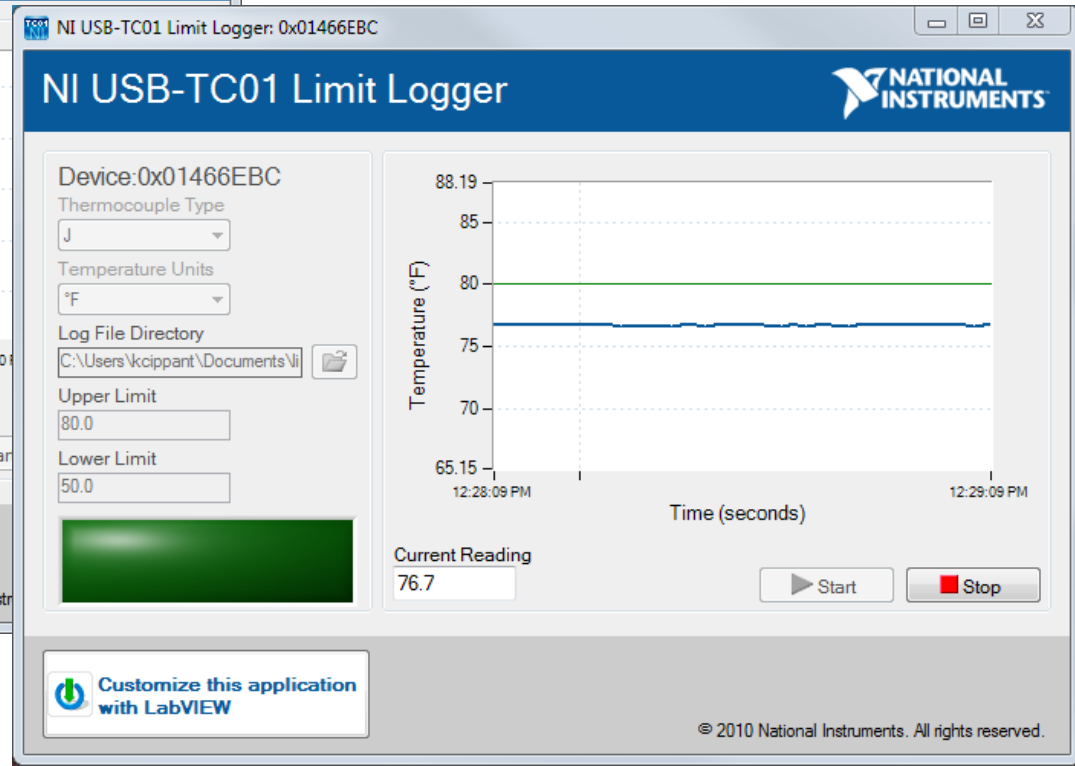
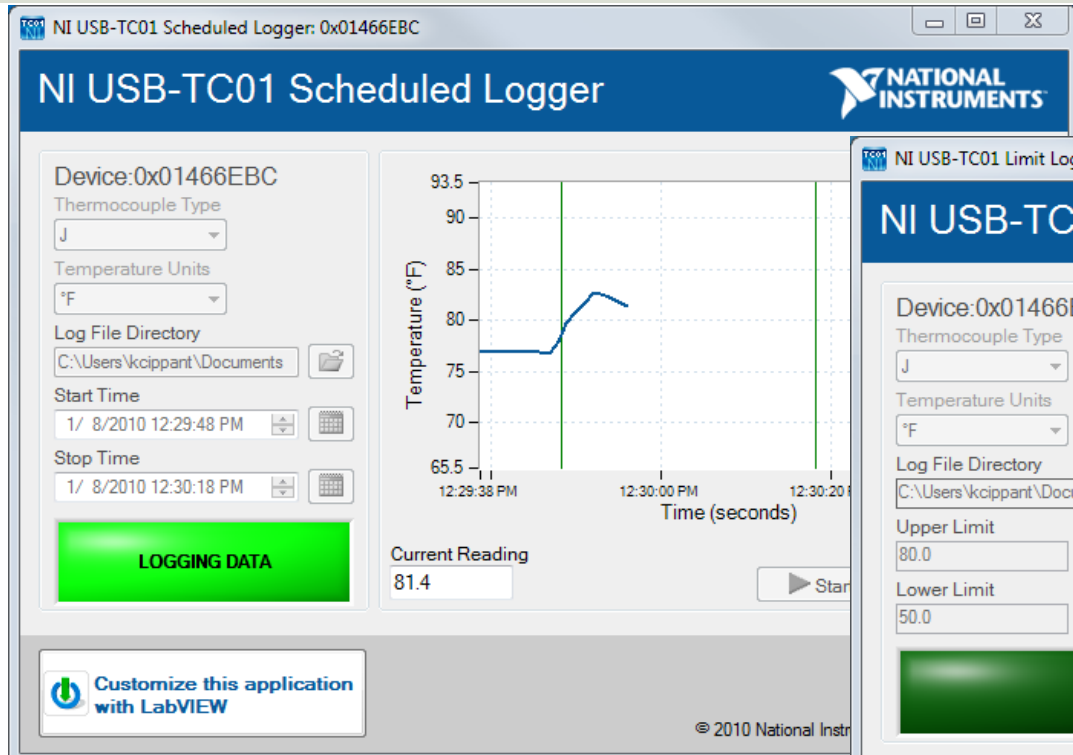
New Book

You can see the video or/and download the code for the entire application and use it as a starting point for your application

<https://halvorsen.blog/documents/teaching/courses/csharp/aspnet.php>

Data Logging Application

Data Logging Example



Data Logging Example

Datalogging

Sensor Name:
Temperature1

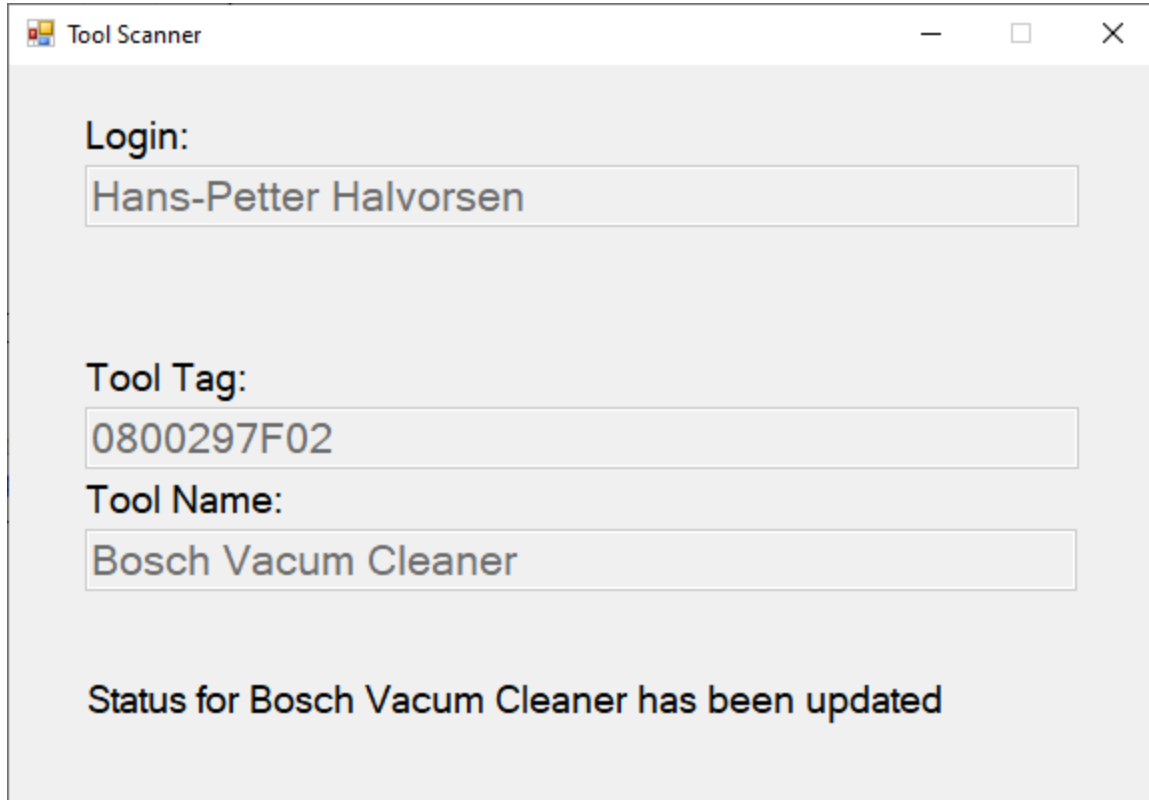
Logging Interval:
1000

Start Stop

Sensor Value: 28.46 C DateTime: 2021-06-21 15:49:08

#	Value	DateTime
1	22.85	2021-06-21 15:...
2	22.2	2021-06-21 15:...
3	27.46	2021-06-21 15:...
4	25.84	2021-06-21 15:...
5	29.63	2021-06-21 15:...
6	20.92	2021-06-21 15:...
7	20.42	2021-06-21 15:...
8	24.99	2021-06-21 15:...
9	26.64	2021-06-21 15:...

Data Logging Example



The screenshot shows a window titled "Tool Scanner" with a light gray background. It contains three text input fields stacked vertically. The first field is labeled "Login:" and contains the text "Hans-Petter Halvorsen". The second field is labeled "Tool Tag:" and contains the text "0800297F02". The third field is labeled "Tool Name:" and contains the text "Bosch Vacum Cleaner". Below these fields, there is a status message: "Status for Bosch Vacum Cleaner has been updated". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Login:
Hans-Petter Halvorsen

Tool Tag:
0800297F02

Tool Name:
Bosch Vacum Cleaner

Status for Bosch Vacum Cleaner has been updated

This Application reads RFID Tags and save/store relevant information to the Database.

The User “tap” their Access Card on the RFID Reader in order to Login, then they “tap” the item(s) they want to borrow/buy

Data Logging Application

These are just suggested features

- You should create a .NET **WinForm** Application (not .NET Core WinForm, because NI-DAQmx does not support .NET Core?)
- Information about Logging, DAQ devices and Sensors should be retrieved from the Database (read-only)
- Save data from sensors in a text file that should be opened, viewed and plotted in **Excel**
- Save data (values) from sensors to **SQL Server**
- **Alarms:** If the Temperature is above a High limit, turn on the red LED. If the Temperature is below a Low limit, turn on the green LED (in System #1: in GUI only).
- What if logging is not working? How can the application handle it? E.g., Send E-mail to System Administrator?

Data Logging Application (cont.)

- Plot values from the different Sensor in one or more plots/charts
- Logging data at specific intervals.
- Have different Light levels (Day, Night, etc.)
- See if you can present (and save) the Light value from the sensor in Lux
- Create a nice User Interface
- Create proper icon(s) for your Forms, etc.
- Make an Executable Application

Not everything is relevant for all projects

Data Logging Application (cont.)

- Alarms: Send E-mail when alarms to a specific person or person (that information should be stored in the database)?
- Store Alarm information in the Database than can be read by the Monitoring Application?
- Etc.

Data Logging Application - Resources

Videos explaining how to use the DAQ devices TC-01 and USB-6008

Videos explaining how to use the sensors, etc. like Temperature sensors, Light sensors, LEDs, etc.

C# DAQ Videos (YouTube Playlist):

https://www.youtube.com/playlist?list=PLdb-TcK6Aqj1z8onXRakywLpn7Tn7_Bti

Lux (System #2)

Illuminance (lux)	Surfaces illuminated by
0.0001	Moonless, overcast night sky (starlight) ^[4]
0.002	Moonless clear night sky with airglow ^[4]
0.05–0.3	Full moon on a clear night ^[5]
3.4	Dark limit of civil twilight under a clear sky ^[6]
20–50	Public areas with dark surroundings ^[7]
50	Family living room lights (Australia, 1998) ^[8]
80	Office building hallway/ toilet lighting ^{[9][10]}
100	Very dark overcast day ^[4]
150	Train station platforms ^[11]
320–500	Office lighting ^{[8][12][13][14]}
400	Sunrise or sunset on a clear day.
1000	Overcast day; ^[4] typical TV studio lighting
10,000–25,000	Full daylight (not direct sun) ^[4]
32,000–100,000	Direct sunlight

Design a Luxmeter Using a Light Dependent Resistor:

<https://www.allaboutcircuits.com/projects/design-a-luxmeter-using-a-light-dependent-resistor/>

Tip! Use a **Lux App** on your Smart Phone

<https://en.wikipedia.org/wiki/Lux>

Data Monitoring Application

Data Monitoring Example

Weather Today

2019-12-18 10:22

Below you find an overview of the weather at USN Porsgrunn:

 -1.04°C
 Calm (0.0 m/s)

For more details, select View

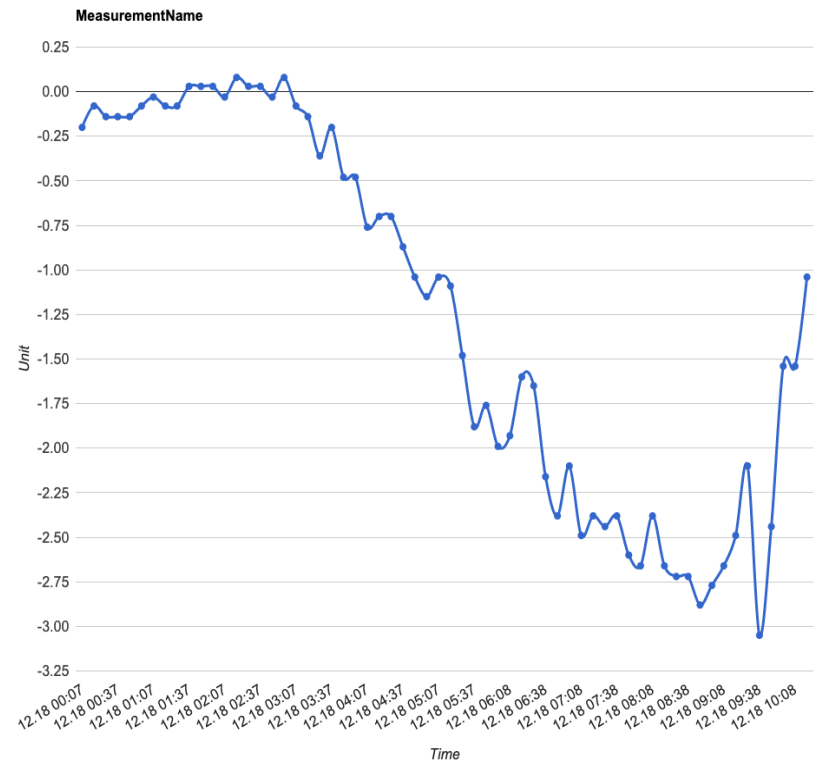
Developed by [Hans-Petter Halvorsen](https://www.halvorsen.blog)

Weather

Below you find an overview of the weather at USN Porsgrunn:

Measurement Alias	Measurement Name	Current Value	Unit	TimeStamp
Temperature	mtTemp1	-1.04	°C	12/18/2019
Wind Speed	mtWindSpeed	0	m/s	12/18/2019
Wind Direction	mtAdjWindDir	191	degrees	12/18/2019
Relative Humidity	mtRelHumidity	87	%	12/18/2019
Barometric Pressure	mtAdjBaromPress	1001.58	hPa	12/18/2019
Rain	mtRainToday	0	mm	12/18/2019
Solar Radiation	mtSolarRadiation	104	W/m2	12/18/2019
Wind Chill	mtWindChill	-1.04	°C	12/18/2019

mtTemp1











Data Monitoring Example

Example of a Data Monitoring Application for lending Tools

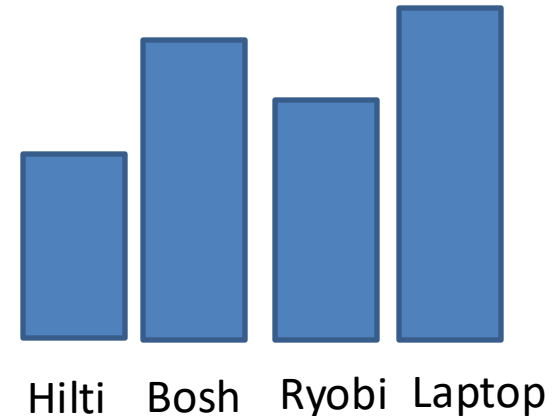
Tools

Below you see all the available Tools in the Inventory:

TooldId	Name	Tag	Status	Action
2	Hilti Hammer Drill	080029C06C		
5	Bosch Vacuum Cleaner	0800297F02		
6	Ryobi R18RS7-1 Bayonet Saw	080029CB9B		
7	Laptop	555555555		

The Application gives an overview id a Tool is borrowed or not.

In addition, it shows Lending history, etc.



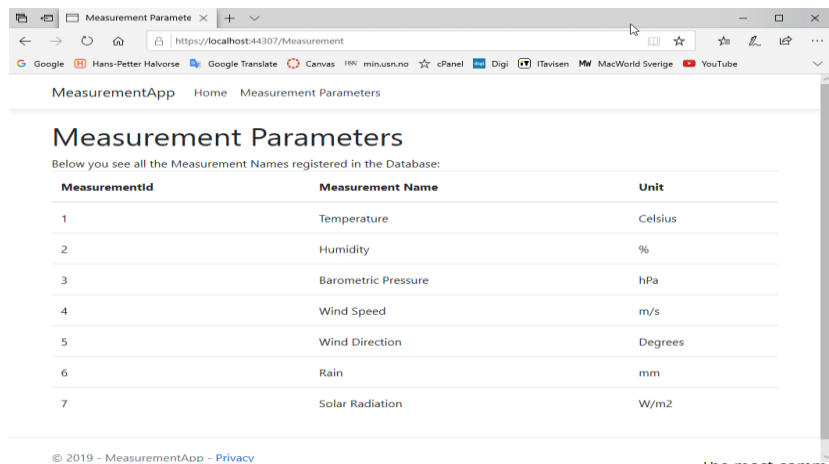
Data Monitoring Application

These are just suggested features

- Create an **ASP.NET Core** Application
- Show latest data from the sensors
- Create one or more **Charts** that show the data from the Sensors
- Show **Statistics** Data
- Possible to choose to see the temperature values in Celsius or Fahrenheit
- Possible to view important information about the sensors

Data Monitoring Application - Resources

“ASP.NET Core Database Communication” Example



The screenshot shows a web browser window with the URL <https://localhost:44307/Measurement>. The page title is "Measurement Parameters" and the content displays a table of measurement parameters. The table has three columns: "MeasurementId", "Measurement Name", and "Unit".

MeasurementId	Measurement Name	Unit
1	Temperature	Celsius
2	Humidity	%
3	Barometric Pressure	hPa
4	Wind Speed	m/s
5	Wind Direction	Degrees
6	Rain	mm
7	Solar Radiation	W/m2

© 2019 - MeasurementApp - Privacy

Video (YouTube):

<https://youtu.be/0Ta3dQ3rxzs>

“ASP.NET Core Chart” Example

Video (YouTube):

<https://youtu.be/mksUls9fx-Q>

Chart Examples

This ASP.NET Core Web Application demonstrates how to implement Charts in your Web Application.

Use Google Charts. [Here you see a Basic Google Chart Example](#)

Fetches the Data from a SQL Server Database:

You can see the videos or/and download the code for the entire applications and use it as a starting point for your application

The most common way to use Google Charts is with simple JavaScript that you embed in your web page. You load some Google Chart libraries, list the data to be charted, select options to customize your chart, and finally create a chart object with an id that you choose. Then, later in the web page, you create a `<div>` tag with that id to display the Google Chart.

[Google Charts](#)

Developed by [Hans-Petter Halvorsen](https://www.halvorsen.blog) (<https://www.halvorsen.blog>)

<https://www.halvorsen.blog/documents/programming/web/aspnet>

Web API (Optional)

- ASP.NET Core
- Make Data available to external systems
- JSON Format

You may also consider making a Web API that can be used by your Data Logging Application. In that way you don't need a direct connection to your database, you instead use the API as the middleware. This is a great benefit when your Desktop Application and your Database is not on the same computer (but located in a LAN or over Internet)

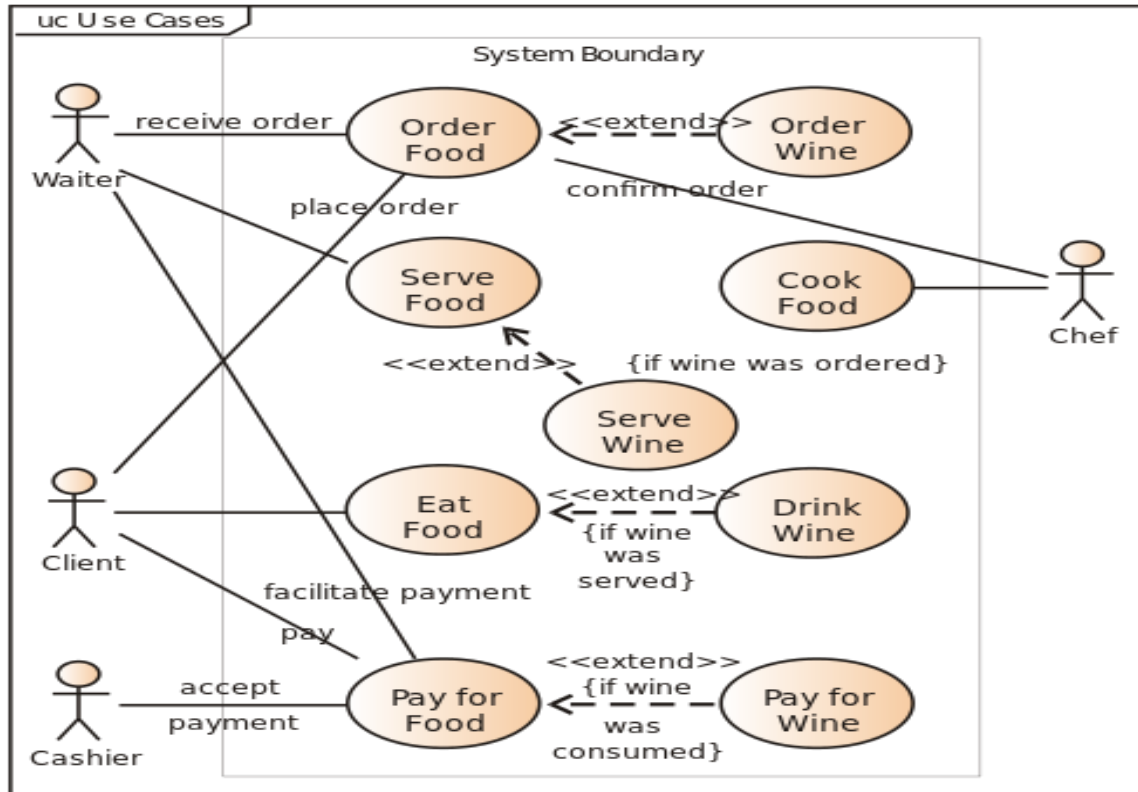
UML

UML

- A **Use Case Diagram** for the system should be created
- A **Class Diagram** for the system should be created
- Se UML examples on the next pages

Use Case Diagram (Example)

A Use Case diagram is a representation of a user's interaction with the system



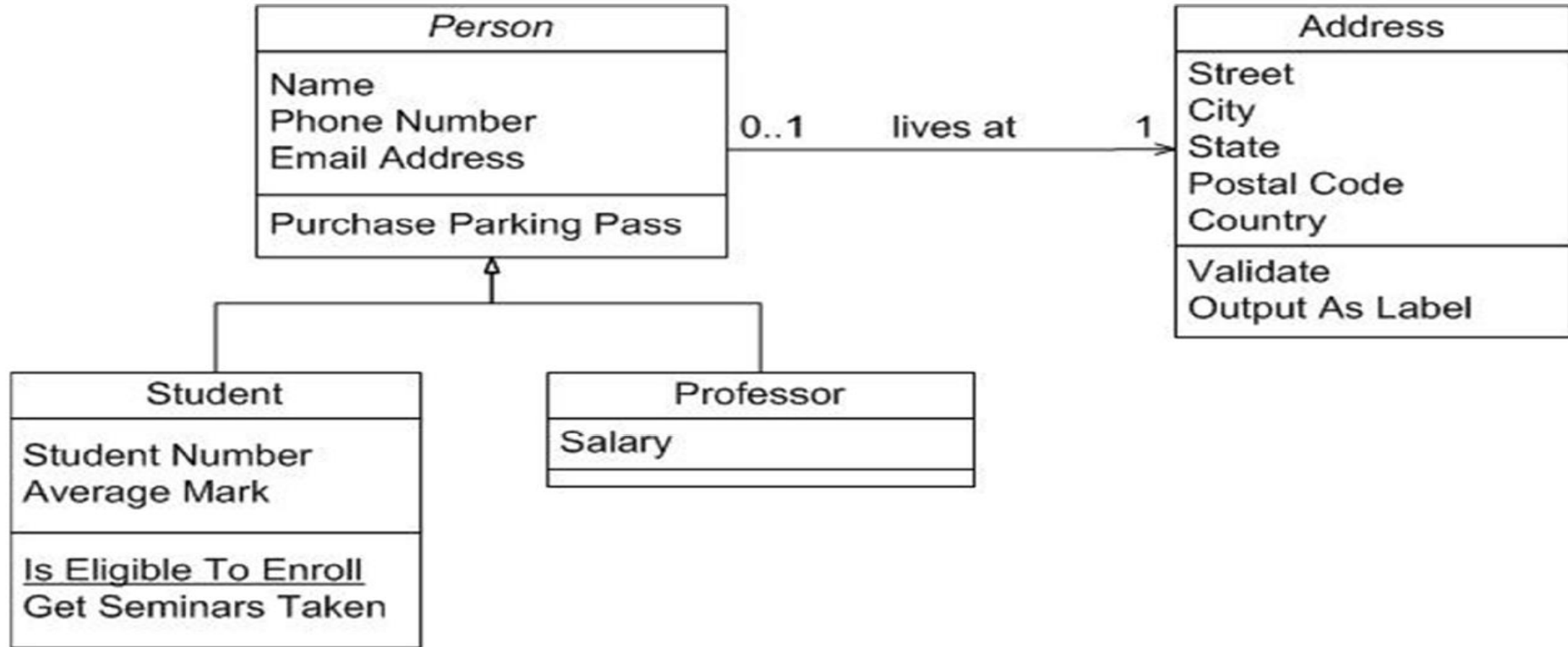
Order
Food

The “circles” are called **Use Cases**. They say what is the main functionality of the application. Here you typically use **verbs** (e.g., OrderFood).



The small “persons” are called **Actors**. They can be real persons or a system. The actors interact with the system in some way.

Class Diagram Example



C# Code

C# Programming

- Create 3 C# Applications:
 1. Data Management App
 2. Data Logging App
 3. Data Monitoring App
- Create and use the main C# concepts like (in addition to If-else, For Loops and While Loops, etc.)
 - Classes with Constructors, Methods, Fields and Properties, Encapsulation, etc.
 - Inheritance
 - Interfaces
- Create and use at least 5 **Classes** with necessary Methods and Properties
- Classes may be shared and used by the different applications, typically you may want to create a Class Library (but that is optional)

User Identity and Login in ASP.NET Core

- Typically, the Web Applications should have login for this system
- The Namespace “Microsoft.AspNetCore.Identity” contains functionality for Identity handling
- We can use the **PasswordHasher<TUser>** Class that has the following Methods:
 - **HashPassword(TUser, String)**
 - Returns a hashed representation of the supplied password for the specified user.
 - **VerifyHashedPassword(TUser, String, String)**
 - Returns a PasswordVerificationResult indicating the result of a password hash comparison.

Create User and Login Example

Create User

Name:

E-Mail:

Password:

Information given by User



```
passwordHashed = HashPassword(userName, password);
```

Save

Store Hashed Password in the Database

Login

Enter UserName and Password in order to get access to the system.

E-Mail:

Password:

Compare Hashed Password stored in the Database with Password given by User in Login Page

```
valid = VerifyHashedPassword(userName, passwordDB, password);
```

Login

Microsoft.AspNetCore.Identity

Example:

```
using Microsoft.AspNetCore.Identity;
```

```
...
```

```
string username; //UserName given by user when creating a User  
string passwordHashed;
```

```
PasswordHasher<string> pw = new PasswordHasher<string>();
```

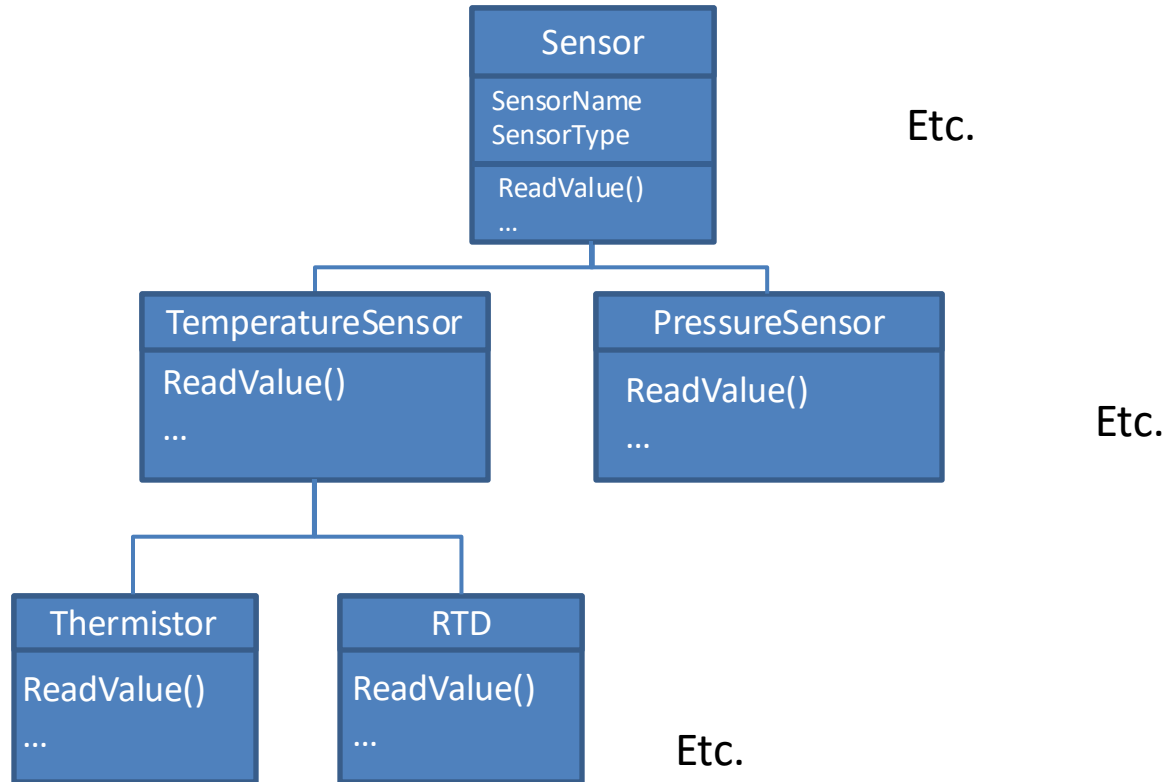
```
passwordHashed = pw.HashPassword(userName, password);
```

Inheritance (Examples)

Inheritance is a feature of object-oriented programming languages that allows you to define a base class that provides specific functionality (data and behavior) and to define derived classes that either inherit or override that functionality.

- Assume you have different types of sensors sharing some common features, then you can, e.g., have a Base Class called `Sensor()` and then other derived Classes like `TemperatureSensor()`, etc. that either inherit or override the functionality of the base class.
- If you have different types of Temperature Sensor, you can make derived classes like `Thermistor()` that inherit/override functionality of `TemperatureSensor()`

Inheritance (Examples)



Inheritance (Examples)

```
Class Sensor
```

```
{  
  ...  
  ...  
}
```

```
Class TemperatureSensor : Sensor
```

```
{  
  ...  
  ...  
}
```

```
Class Thermistor : TemperatureSensor
```

```
{  
  ...  
  ...  
}
```

```
...
```

```
Thermistor thermistor1 = new Thermistor();  
temperaturevalue = thermistor1.ReadValue();
```


Polymorphism (Examples)

Class **TemperatureSensor**

```
{  
  public virtual double y Scaling(double x)  
  {  
    y = 5*x;  
  }  
  ...  
}
```

Class **Thermistor** : **TemperatureSensor**

```
{  
  public override double y Scaling(double x)  
  {  
    y = 2*x + 2;  
  }  
  ...  
}
```

Class **Tmp36** : **TemperatureSensor**

```
{  
  public override double y Scaling(double x)  
  {  
    y = x*(9/5) + 45;  
  }  
  ...  
}
```

The **override** modifier allows a method to override the virtual method of its base class at run-time.

```
...  
N = //Get number of sensors stored in the Database  
TemperatureSensor[] tempsensor = new TemperatureSensor[N]  
  
//Here you should use a for loop instead  
tempsensor[0] = new TemperatureSensor()  
tempsensor[1] = new Thermistor()  
tempsensor[2] = new Tmp36()  
  
foreach (Temperature sensor in tempsensor)  
{  
  x = //Get Value from DAQ device  
  y = sensor.Scaling(x); //Scale the value  
  SaveData(x); //Save Data to the Database  
}
```

Here, depending of the Sensor, different Scaling formula will be used

Interfaces (Examples)

Interfaces are used along with classes to define what is known as a **contract**. A contract is an agreement on what the class will provide to an application.

An interface declares the properties and methods. It is up to the class to define exactly what the method will do.

- Assume you make this system as an open platform meaning other developers can use it to add logging functionality from other sensors.
- The system will not work if they don't implement a Name for the Sensor and a ReadValue() method
- To make sure that they follow this, you should implement Interfaces

Interfaces (Examples)

Interface Example:

```
public interface ISensorType
{
    ...
    ...
    public double ReadValue (int id);
}
```

A Class that Implements the Interface

```
public class Rain : ISensorType
{
    ...
    public double ReadValue (int id)
    {
        double value;
        value = ReadDaq(id);
        return value*(9/5) +32;
    }
}
```

Final Delivery

Delivery (in Canvas)

1. Technical Report (PDF)

- Introduction, Methods, Results, Discussions and Conclusions (IMRaD). See “**Report Checklist**” for details

2. Video (MP4). About 5minutes

1. Short Overview of System
2. Real-life Demonstration of the Prototype
3. Short overview of the main Code Structure

3. Code (C# + DB Scripts) as ZIP File

Individual Delivery!

Note! The Report and the Video should be independent of each other, meaning it should be possible to get an overview of the system either you read the report or watching the video

Technical Report

- The IMRaD structure is a way of structuring a scientific report or article
- IMRaD is short for **Introduction – Methods – Results** – and – **Discussion** (and Conclusions)
- IMRaD is a way of structure the contents. It may not necessary be the names of the chapters within the report, but more the different sections that need to be in such a report and the order and the presentation of the contents

<https://sokogskriv.no/en/writing/the-imrad-format.html#introduction>

In this project you shall **NOT** use/make

- Microsoft Azure (you shall use your personal computer only)
- Source Code Control Systems (like GitHub)
- Make detailed project plans and other documentation

In this Project, Programming is in focus

The topics above are important in the course “Software Engineering”

Project Checklist (Recommendations)

- erwin Data Modeler is installed and used on your PC
- SQL Server Express is installed and used on your PC
- Visual Studio is installed and used on your PC
- Simple **Use Case Diagram** (s)
- Data Logging** Application
- (Data Management Application)
- Data Monitoring** Application
- Database** > 5 Tables
- Simple **Database Diagram (ER)**
- Stored Procedure(s)**
- Database View(s)**
- (Database Trigger(s)) Optional
- Class Diagram**
- Made >5 **Classes**
- Use of **Inheritance** (some places)
- Use of **Polymorphism** (some places)
- Use of **Interfaces** (some places)
- Try-Catch** (some places) is used to make the code more robust
- (Class Library) Optional
- (Web API) Optional
- Report** is delivered
- Video** (about 5 min) is delivered
- Code** as ZIP File is delivered

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: <https://www.halvorsen.blog>

