Control and Simulation in Visual Studio/C#

Hans-Petter Halvorsen, M.Sc.

Contents

- PID Control
- Dynamic Systems
- Discrete Systems
  - Discretization and Simulation of Continuous Systems
  - Discrete PI Controller
  - Discrete Lowpass Filter
- ...

Software

- DAQmx Driver
- Visual Studio/C#

Note! It is an advantage if you have acknowledge of basic Programming, Math, Control Theory, such as Differential Equations, PID control, basic C# programming, etc.

Industrial Control Systems (ICS)

Theory

Industrial Control Systems are computer controlled systems that monitor and control industrial processes that exist in the physical world

cRIO

LabVIEW

1

I/O Module

5

Programmable Automation Controller (**PAC**)

4

Industrial **PID** Controller

PC based Control System/**SCADA** System (Supervisory Control And Data Acquisition)

2

Distributed Control Systems (**DCS**)

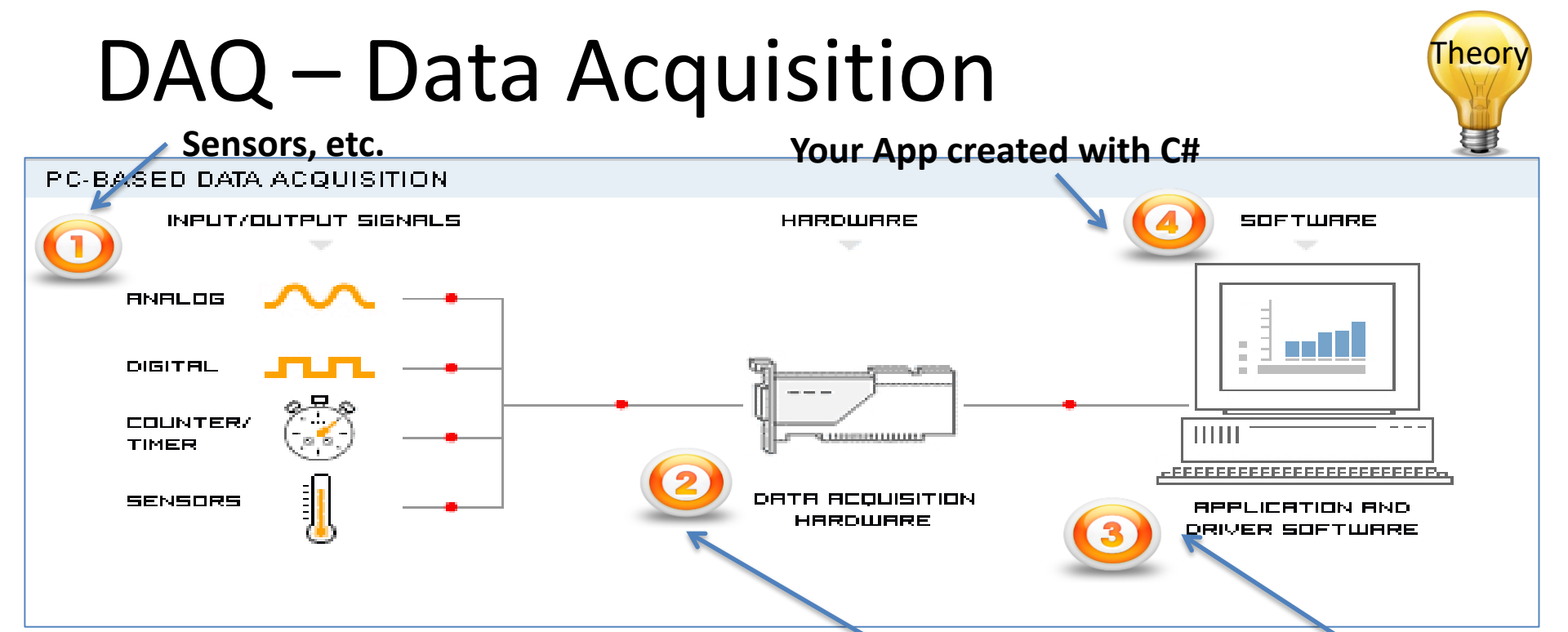
DeltaV

PLC (Programmable Logic Controller)

3

Siemens PLC

DAQ – Data Acquisition



A DAQ System consists of 4 parts:

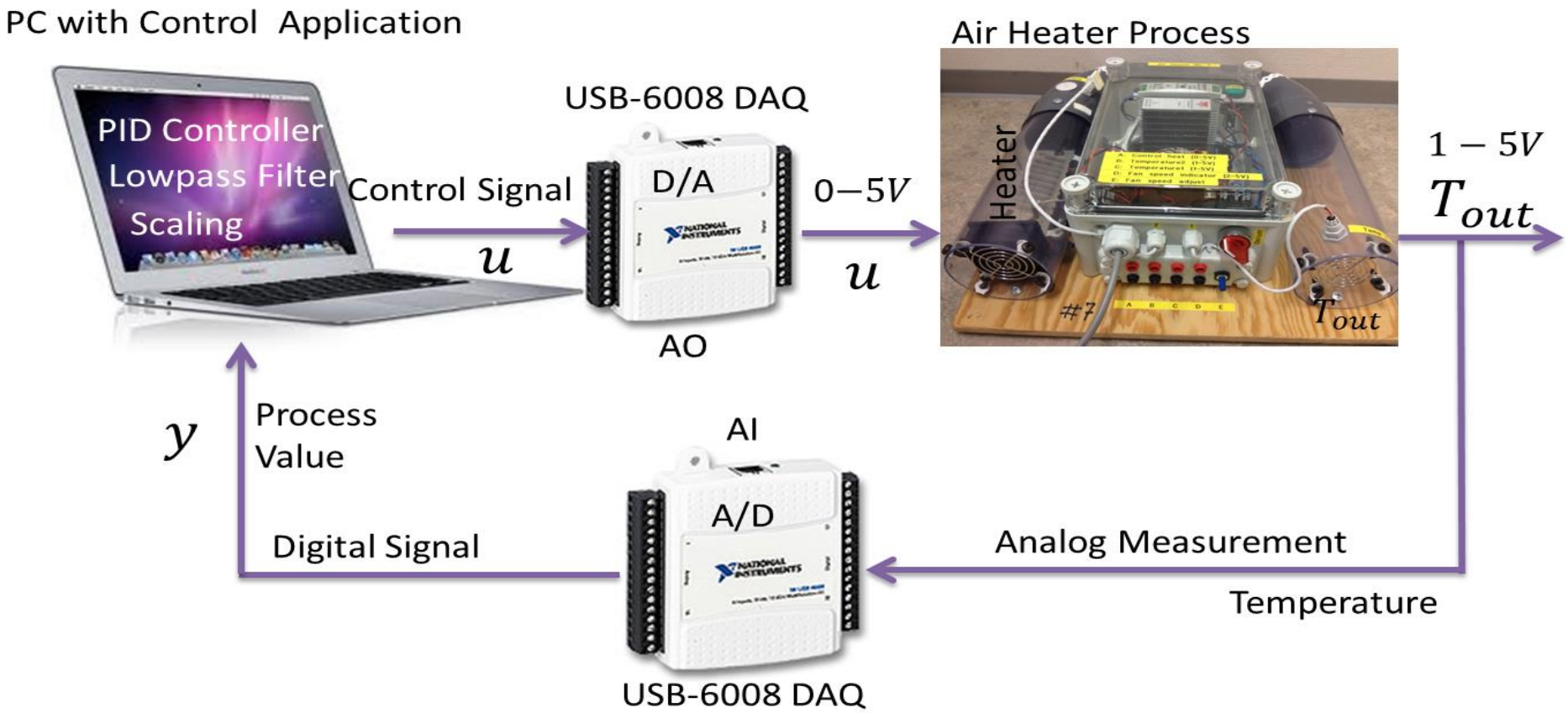
1. Physical input/output signals
2. DAQ device/hardware
3. Driver software
4. Your software application (Application software)

NI USB 6008 DAQ Device

NI DAQmx Driver or similiar

PC-based Control System

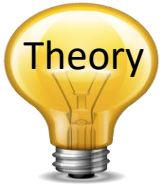
Theory



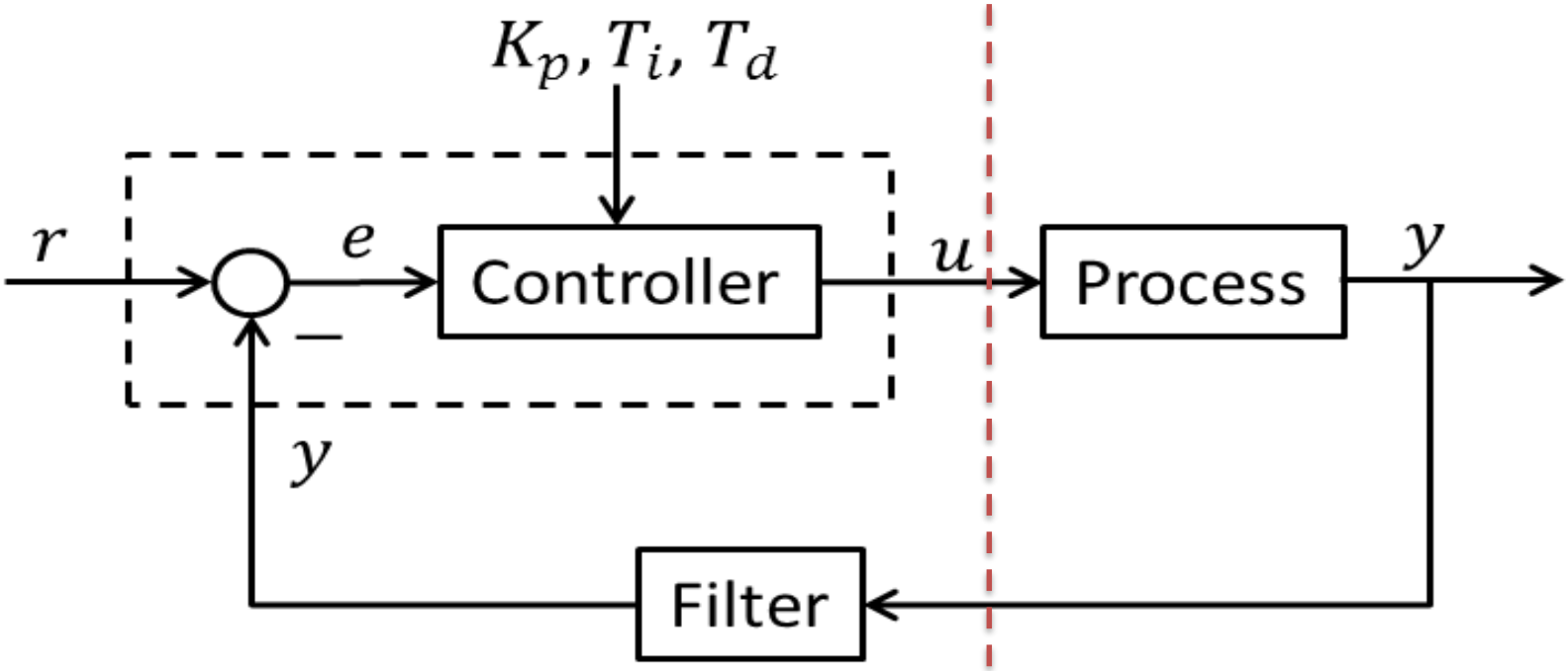
PID Control

Hans-Petter Halvorsen, M.Sc.

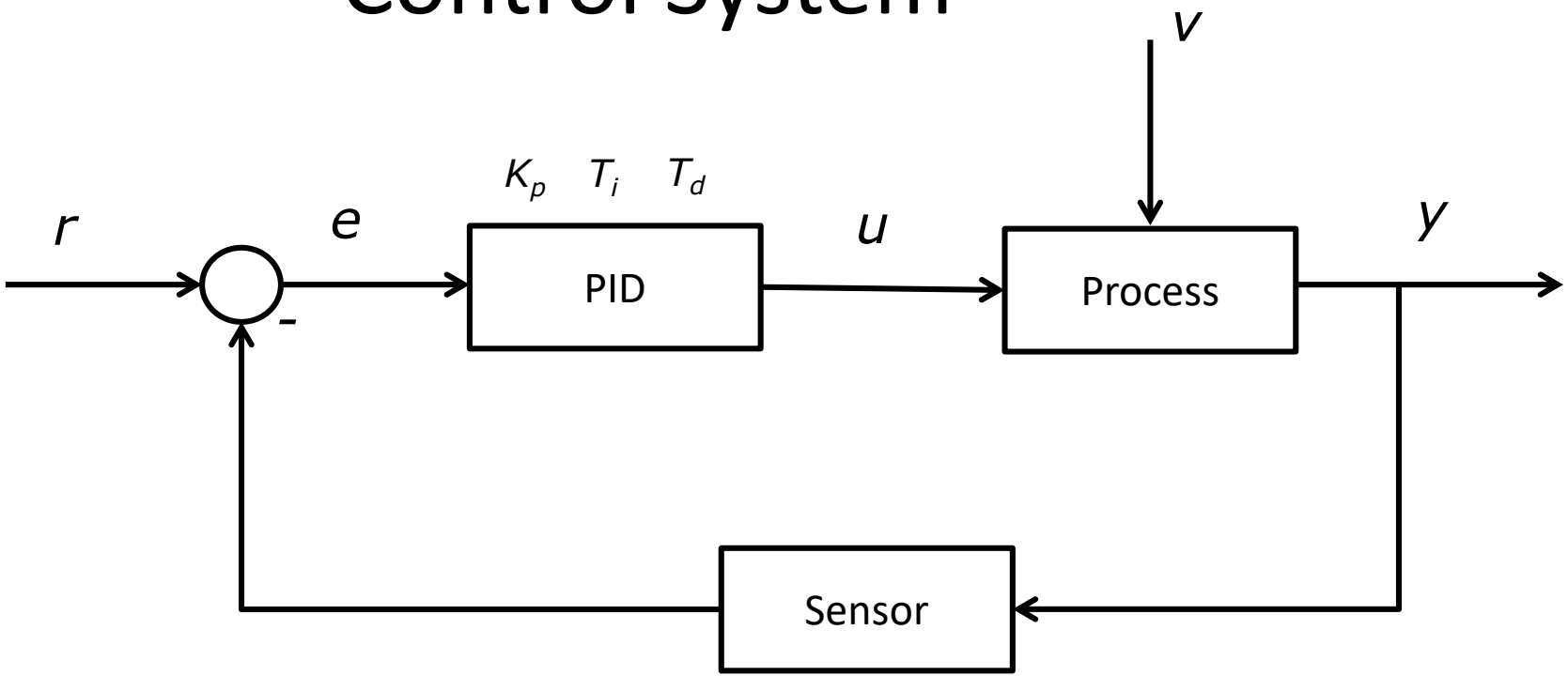
Implementing a Control System



While the real process is continuous, normally the Controller and the Filter is implemented in a computer.



Control System



r – Reference Value, SP (Set-point), SV (Set Value)
y – Measurement Value (MV), Process Value (PV)
e – Error between the reference value and the measurement value (e = r – y)
v – Disturbance, makes it more complicated to control the process
Kp, Ti, Td – PID parameters

The PID Algorithm

$$u(t) = K_p e + \frac{K_p}{T_i}\int_0^t e d\tau + K_p T_d \dot{e}$$

Where u is the controller output and e is the control error:

$$e(t) = r(t) - y(t)$$

r is the Reference Signal or Set-point

y is the Process value, i.e., the Measured value

Tuning Parameters:

K_p Proportional Gain

T_i Integral Time [sec.]

T_d Derivative Time [sec.]

Dynamic Systems

Hans-Petter Halvorsen, M.Sc.

Dynamic Systems Examples

Water Tank:

h – Level in the tank

Mathematical Models (differential equations):

Alt 1 (Integrator):

$$\dot{h} = \frac{1}{A}\left[K_p u - F_{out}\right]$$

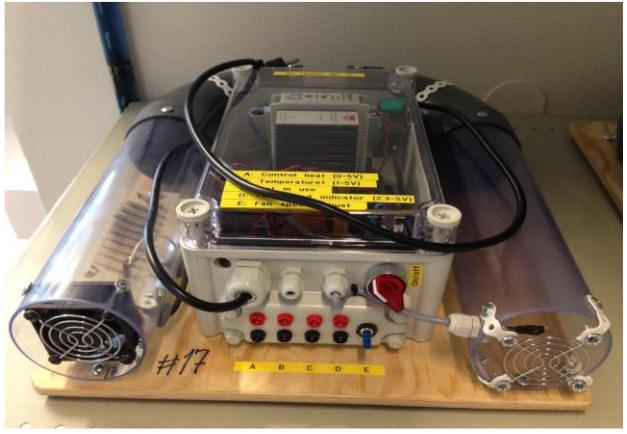
Alt 2 (Time constant/1.order):

$$\dot{h} = \frac{1}{A}\left[K_p u - K_v h\right]$$

Alt 3 (Nonlinear):

$$\dot{h} = \frac{1}{A}\left[K_p u - K_v \sqrt{\frac{\rho g h}{G}}\right]$$

Air Heater:



$$\dot{T}_{out} = \frac{1}{\theta_t}\{-T_{out} + [K_h u(t-\theta_d) + T_{env}]\}$$

T – Temperature in the tube

Dynamic Systems (1.order)

Dynamic system represented as a differential equation (1.order system):

$$\dot{x} = -ax + bu$$

$$\frac{dx}{dt}$$

Note!

Where:

x - Process variable, e.g., Level, Pressure, Temperature, etc.

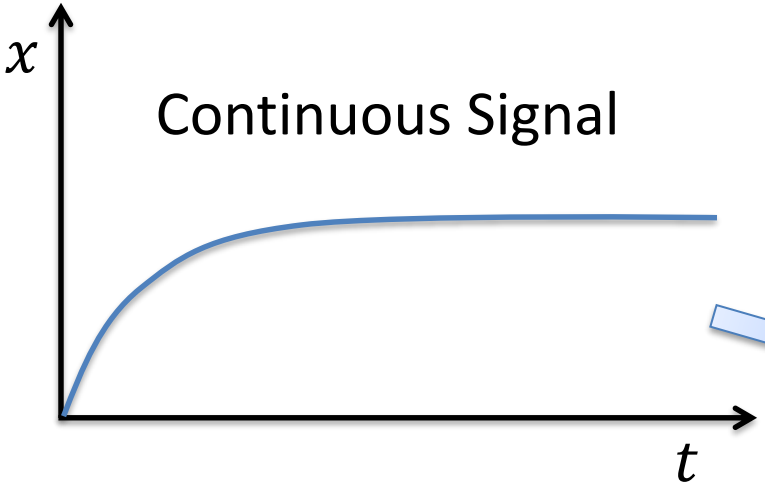
u - Input variable, e.g., Control Signal from the Controller

a, b - Constants

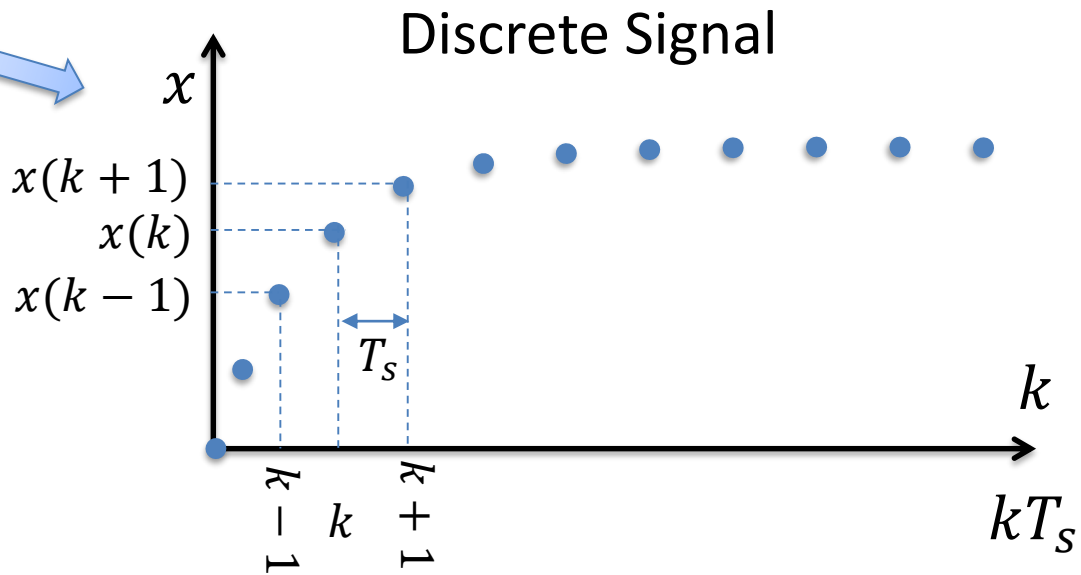
Discrete Systems

Hans-Petter Halvorsen, M.Sc.

Continuous vs. Discrete Systems



A computer can only deal with discrete signals



T_s - Sampling Interval

$x(k-1)$ - Previous Value

$x(k)$ - Current Value

$x(k-1)$ - Next Value

Different Discrete Symbols and meanings

Previous Value: $x(k-1) = x_{k-1} = x(t_{k-1})$

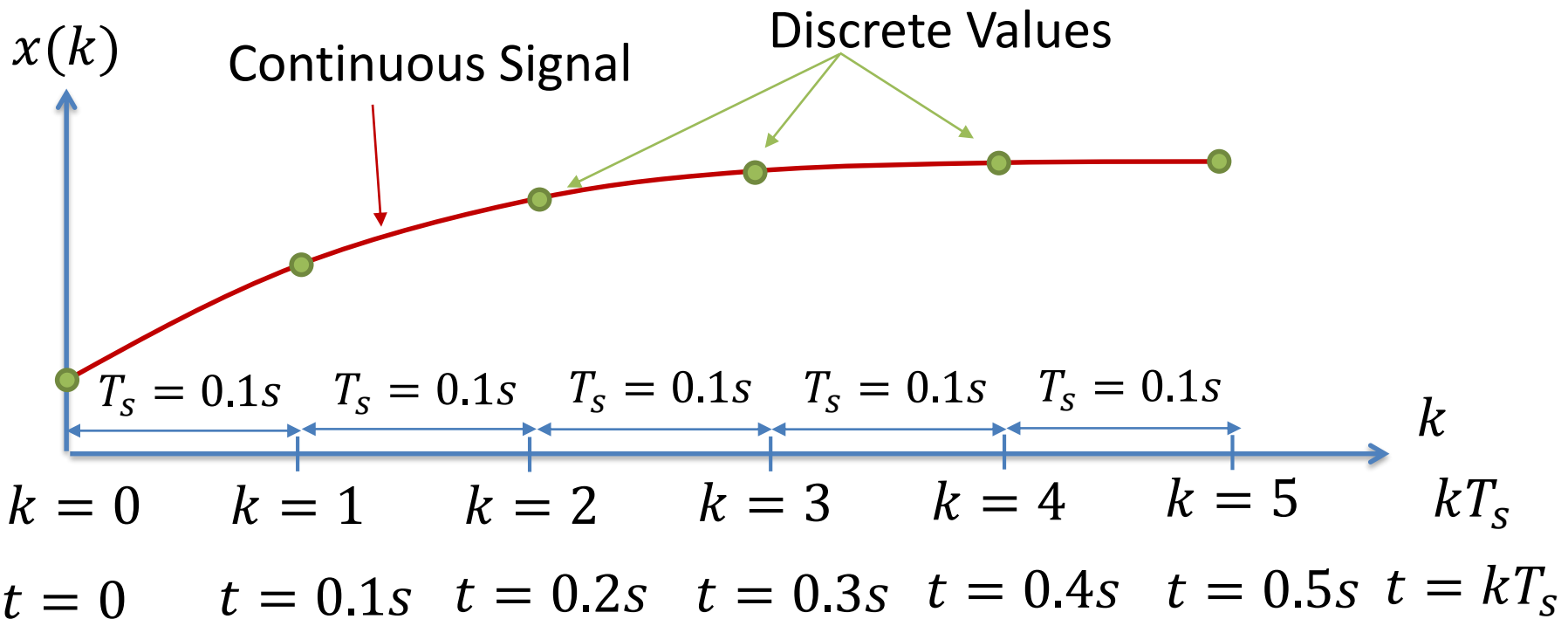
Present Value: $x(k) = x_k = x(t_k)$

Next (Future) Value: $x(k+1) = x_{k+1} = x(t_{k+1})$

Note! Different Notation is used in different literature!

Continuous vs. Discrete Systems - Example

In this Example we have used Sampling Interval $T_s = 0.1s$



Continuous vs. Discrete Systems - Example

We have:

$$x(t) \rightarrow x(k)$$

$$x(t) \rightarrow x(kT_s)$$

Example: $T_s = 0.1s$

$t = 0.5s$

$$x(t = 0.5s) \rightarrow x\left(\frac{0.5s}{0.1s}\right) = x(k = 5)$$

or:

$$x(t_{0.5}) \rightarrow x\left(\frac{0.5s}{0.1s}\right) = x(k_5)$$

$t = 5s$

$$x(t = 5) \rightarrow x\left(\frac{5s}{0.1s}\right) = x(k = 50)$$

or:

$$x(t_5) \rightarrow x\left(\frac{5s}{0.1s}\right) = x(k_{50})$$

Euler Approximation

A general continuous differential equation: $\dot{x} = f(x, t)$

1.order continuous differential equation: $\dot{x} = ax + bu$

We can use the Euler Approximation (Euler Forward discretization method) in order to make a discrete version of a continuous system:

$$\dot{x} \approx \frac{x(k+1) - x(k)}{T_s} \quad \text{or:} \quad \dot{x} \approx \frac{x_{k+1} - x_k}{T_s}$$

T_s - Sampling Time

Continuous vs. Discrete Systems

Continuous model:

$$\dot{x} = f(x, u)$$

Discrete model:

$$x_{k+1} = f(x_k, u_k)$$

Example (linear, 1.order system):

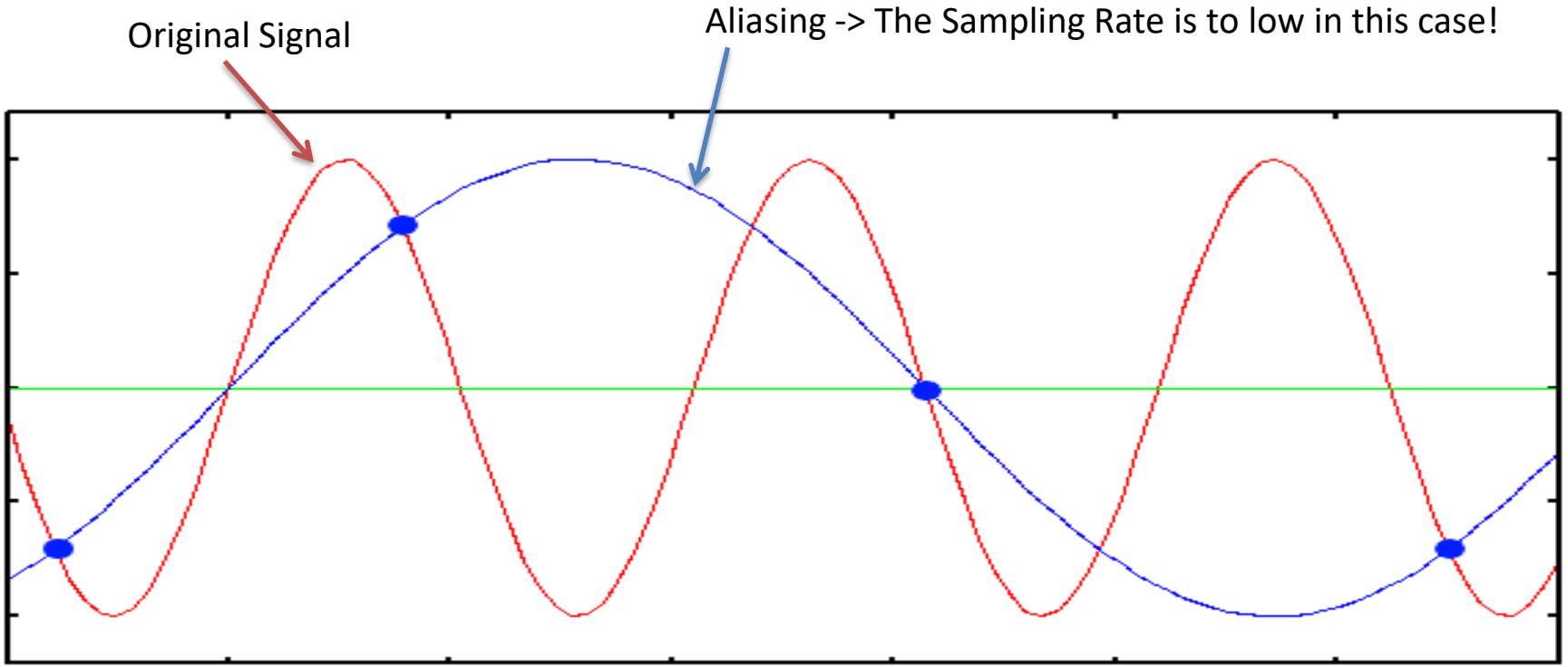
Continuous model:

$$\dot{x} = -ax + bu$$

Discrete model:

$$x_{k+1} = (1 - T_s a)x_k + T_s b u_k$$

Sampling and Aliasing



Discretization

Given the following differential equation:

$$\dot{x} = -ax + bu$$

In order to simulate this system in C#, we need to find the discrete differential equation.

We can use e.g., the **Euler Approximation**:

$$\dot{x} \approx \frac{x_{k+1} - x_k}{T_s}$$

T_s - Sampling Interval

Then we get:

$$\frac{x_{k+1} - x_k}{T_s} = -ax_k + bu_k$$

This gives the following discrete differential equation:

$$x_{k+1} = (1 - T_s a)x_k + T_s b u_k$$

Simulation Example

$$\dot{x} = -ax + bu$$

set $a = 0.25$ and $b = 2$

$$x(k+1) = (1 - T_s a)x(k) + T_s b u(k)$$

Lage C# Example av dette!

Plotte x(t)


```
% Simulation of discrete model
clear, clc

% Model Parameters
a = 0.25;b = 2;

% Simulation Parameters
Ts = 0.1; %s
Tstop = 20; %s
uk = 1; % Step Response
x(1) = 0;

% Simulation
for k=1:(Tstop/Ts)
   x(k+1) = (1-a*Ts).*x(k) + Ts*b*uk;
end

% Plot the Simulation Results
k=0:Ts:Tstop;
plot(k,x)
grid on
```

Discretization of Time Delay - Example

We have: Example: $T_s = 0.1s$

$x(t) \rightarrow x(k)$

$x(t) \rightarrow x(kT_s)$

With Time Delay (τ): $x(t-\tau) \rightarrow x(k-\tau_k)$

Assume Time Delay $\tau = 2s$:

$\tau = 2s \rightarrow \tau_k = \frac{\tau}{T_s} = \frac{2s}{0.1s} = 20$ discrete intervals

This gives: $x(t-\tau) \rightarrow x_{k-\frac{\tau}{T_s}} = x_{k-20}$

In general we have: $x(t-\tau) \rightarrow x(k-\frac{\tau}{T_s})$ or: $x_{t-\tau} \rightarrow x_{k-\frac{\tau}{T_s}}$

Discretization with Time Delay

Given the following differential equation:

$$\dot{x} = -ax + bu(t - \tau)$$

We can use e.g., the **Euler Approximation**:

$$\dot{x} \approx \frac{x_{k+1} - x_k}{T_s}$$

T_s - Sampling Interval

Then we get:

$$\frac{x_{k+1} - x_k}{T_s} = -ax_k + bu_{k-\frac{\tau}{T_s}}$$

This gives the following discrete differential equation:

$$x_{k+1} = (1 - T_s a)x_k + T_s b u_{k-\frac{\tau}{T_s}}$$

If we set $T_s = 0.1s$ and $\tau = 2s$

$$x_{k+1} = (1 - 0.1a)x_k + 0.1bu_{k-20}$$


DEMO

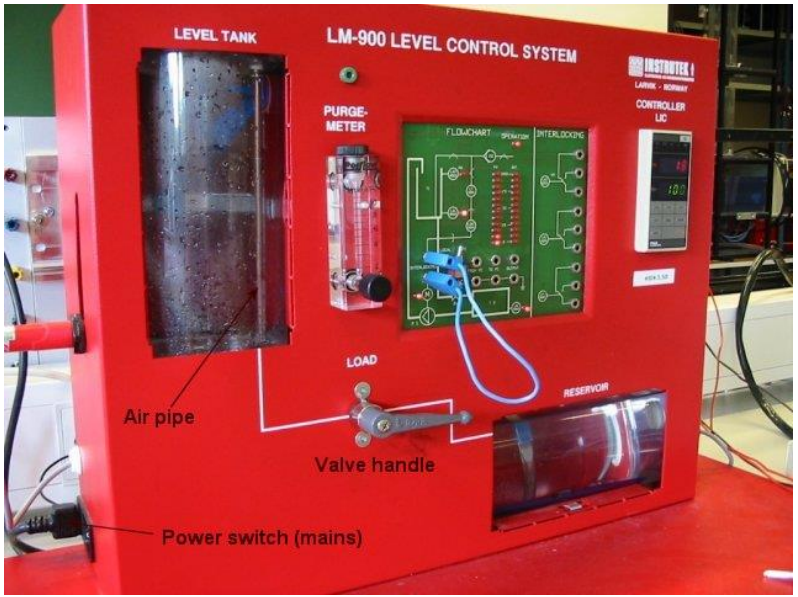
Simulation Example

Controlling a Mathematical Model of a given Level Tank

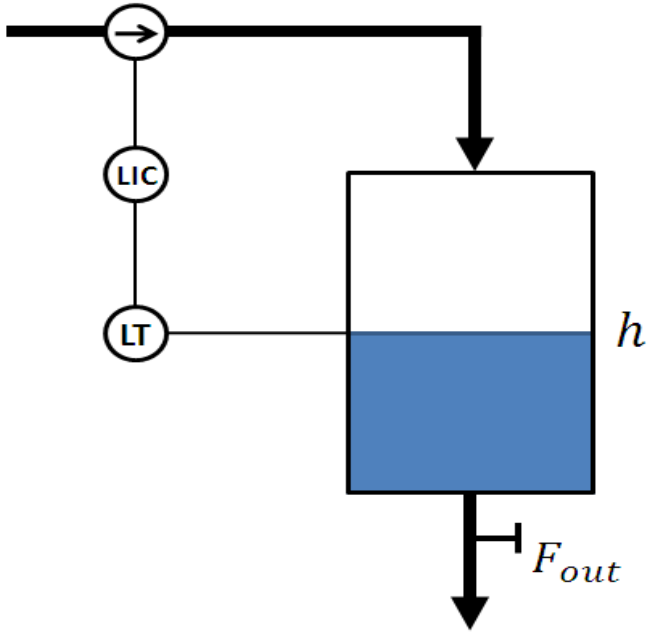
Hans-Petter Halvorsen, M.Sc.

Level Tank Example

Real System:

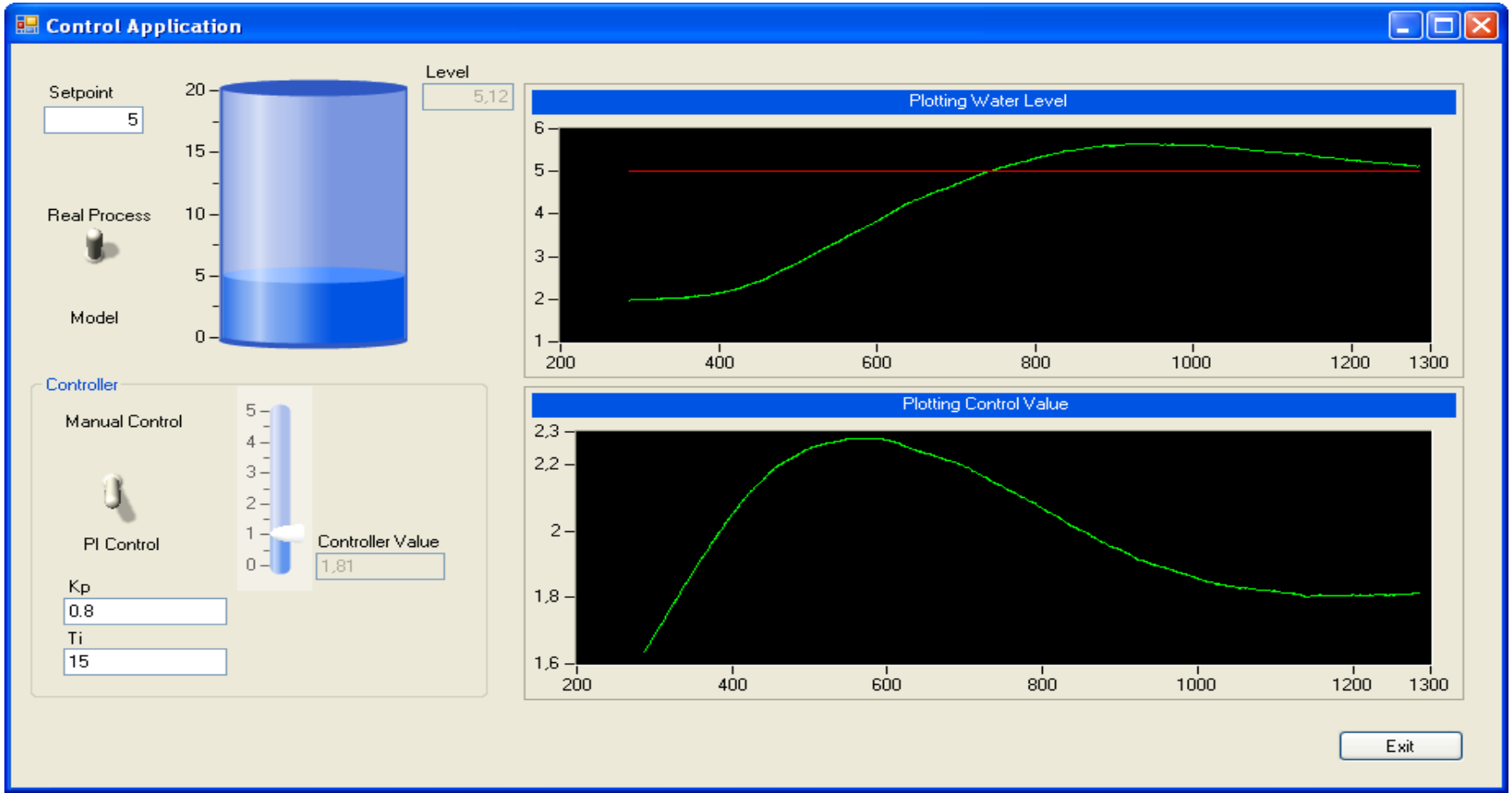


Control Strategy:

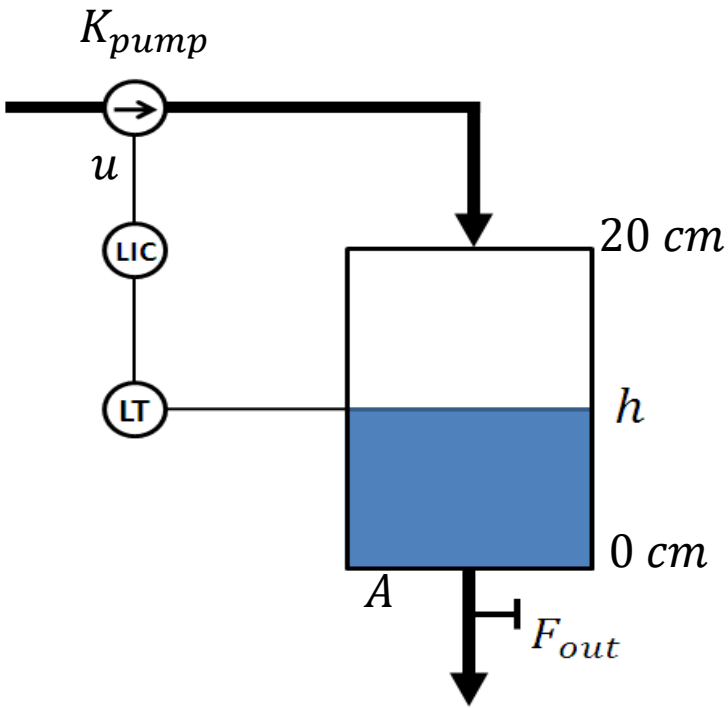


We will show an example where we create a Control and Simulation Application in C#. We will create a Control System that controls the level in water tank. We will use a mathematical model of the water tank (Simulation).

HMI


Control Application
Setpoint
5
Real Process
Model
Level
5,12
20
15
10
5
0
Plotting Water Level
6
5
4
3
2
1
200
400
600
800
1000
1200
1300
Controller
Manual Control
PI Control
5
4
3
2
1
0
Controller Value
1,81
Kp
0.8
Ti
15
Plotting Control Value
2,3
2,2
2
1,8
1,6
200
400
600
800
1000
1200
1300
Exit

Mathematical Model of Level Tank



Mathematical Model: $A\dot{h} = K_{pump} \cdot u - F_{out}$

or:

$$\dot{h} = \frac{1}{A}\left[K_{pump} \cdot u - F_{out}\right]$$

Where:

h [cm] is the level in the water tank
u [V] is the pump control signal to the pump
A_t [cm2] is the cross-sectional area in the tank
K_{pump} [(cm3/s)/V] is the pump gain
F_{out} [cm3/s] is the outflow through a manual valve in the bottom of the tank. F_{out} is constant.

Discretization of Model

We will use the Euler Forward discretization method: $\dot{x} \approx \frac{x_{k+1} - x_k}{T_s}$

We apply Euler on the continuous model: $\dot{h} = \frac{1}{A}\left[K_{pump} \cdot u - F_{out}\right]$

This gives:

$$\frac{h_{k+1} - h_k}{T_s} = \frac{1}{A}\left[K_{pump} \cdot u_k - F_{out}\right]$$

Finally we get the following discrete model of the level tank:

$$h_{k+1} = h_k + \frac{T_s}{A}\left[K_{pump} \cdot u_k - F_{out}\right]$$

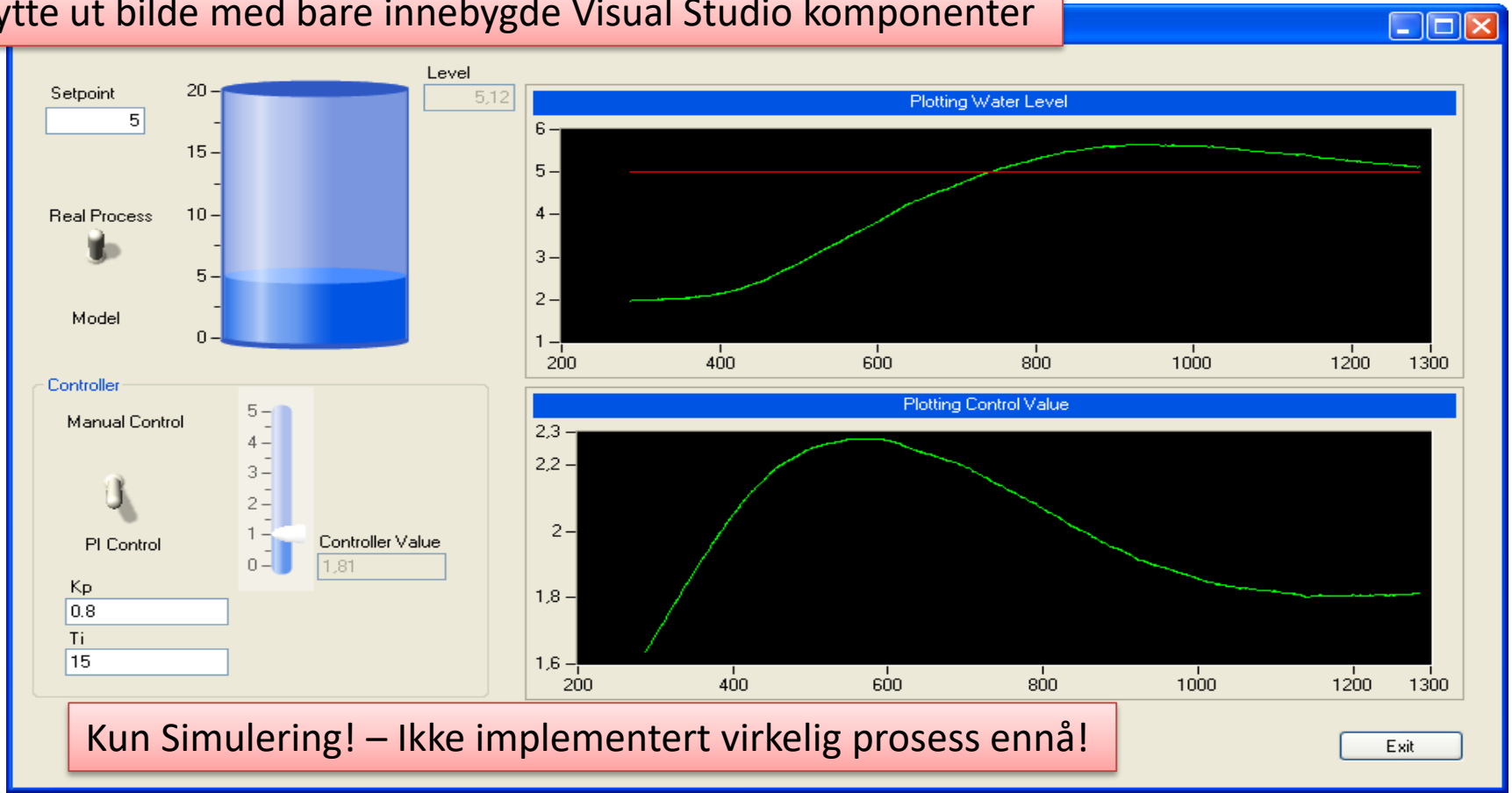

DEMO

Discrete PI(D) Controller

Hans-Petter Halvorsen, M.Sc.

HMI

Bytte ut bilde med bare innebygde Visual Studio komponenter



Kun Simulering! – Ikke implementert virkelig prosess ennå!

Discrete PID Control

Continuous PID Controller:

$$u(t) = K_p e + \frac{K_p}{T_i}\int_0^t e d\tau + K_p T_d \dot{e}$$

- We need to make a discrete version of the PID controller in order to be able to implement it in C#
- In out case we may only need a PI controller, so for simplicity we just create a discrete PI controller.

Discrete PI Controller Example

Continuous PI Controller:

$$u(t) = u_0 + K_p e(t) + \frac{K_p}{T_i}\int_0^t e d\tau$$

$$\dot{u} = \dot{u}_0 + K_p \dot{e} + \frac{K_p}{T_i} e$$

We use the Euler Backward method:

$$\dot{x} = \frac{x_k - x_{k-1}}{T_s}$$

$$\frac{u_k - u_{k-1}}{T_s} = \frac{u_{0,k} - u_{0,k-1}}{T_s} + K_p \frac{e_k - e_{k-1}}{T_s} + \frac{K_p}{T_i} e_k$$

$$u_k = u_{k-1} + u_{0,k} - u_{0,k-1} + K_p(e_k - e_{k-1}) + \frac{K_p}{T_i} T_s e_k$$

We may set:

$$\Delta u_k = u_k - u_{k-1}$$

This gives the following discrete PI algorithm:

$$e_k = r_k - y_k$$

$$\Delta u_k = u_{0,k} - u_{0,k-1} + K_p(e_k - e_{k-1}) + \frac{K_p}{T_i} T_s e_k$$

$$u_k = u_{k-1} + \Delta u_k$$

This algorithm can be easly implemented in a Programming language

Students:
Create a PI(D) Controller in C#

Visual Studio

Simple Discrete **PI Controller** – C#

```
class PidController
{
        public double r;
        public double Kp;
        public double Ti;
        public double Ts;
        private double z;

        public double PiController(double y)
        {
            double e;
            double u;

            e = r - y;
            u = Kp * e + (Kp / Ti) * z;
            z = z + Ts * e;
            return u;
        }
}
```

Note! This is just an Example

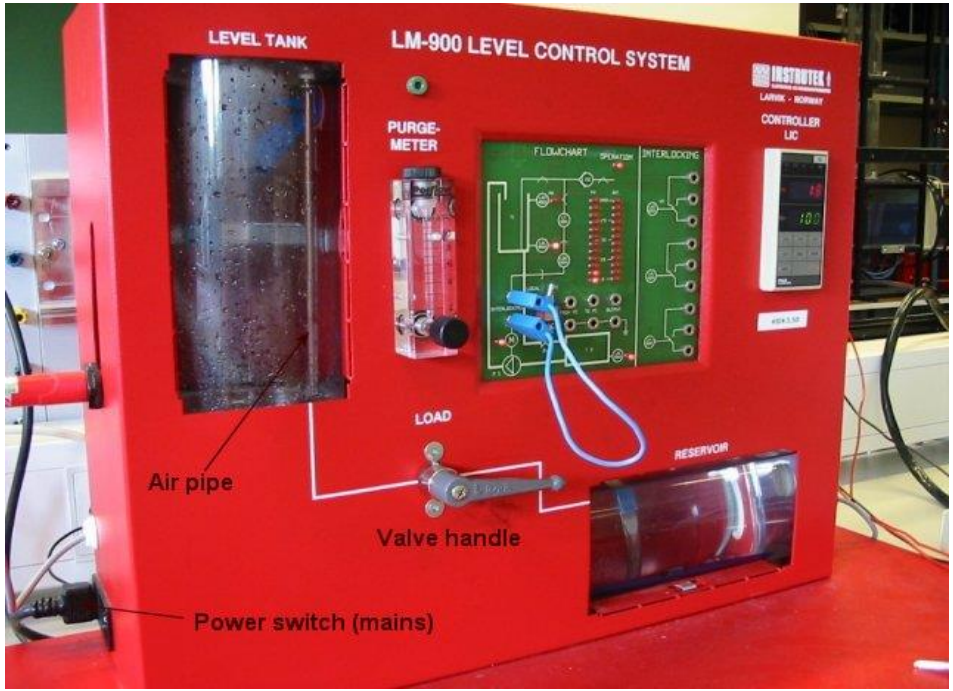

DEMO

Real Control System

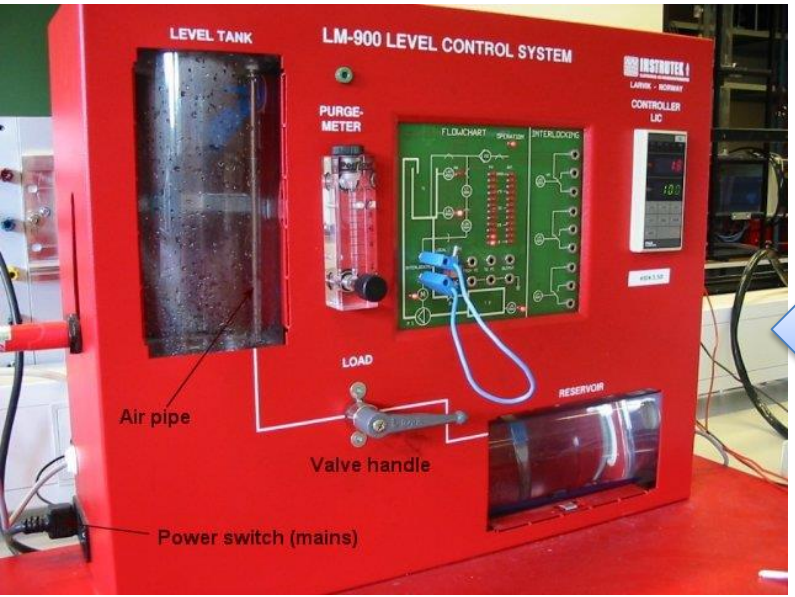
Controlling the Real Level Tank System

Hans-Petter Halvorsen, M.Sc.

Level Tank System



Level Tank Control System



In this example we will use a small-scale laboratory process called LM-900 Level System

NI USB-6008 I/O Module

Note! **DAQmx** Driver is needed!!

Specifications:

-10-10V

- **8 analog inputs**, AI (12-bit, 10 kS/s, -10-10V)
- **2 analog outputs**, AO (12-bit, 150 S/s, 0-5V)
- **12 digital I/O** (DI/DO)
- 32-bit counter

0-5V

4 different types of Signals:

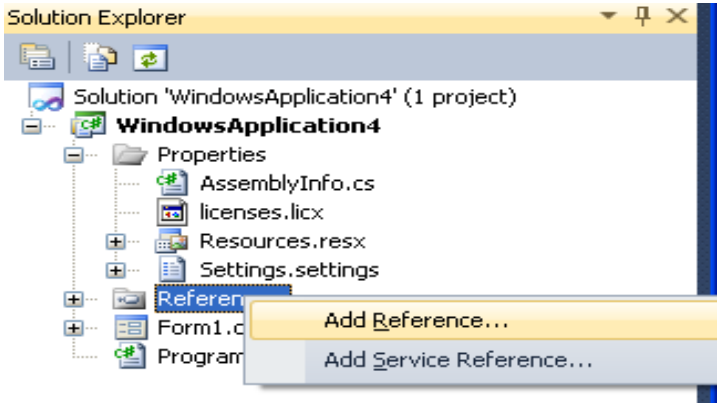
AO – Analog Output

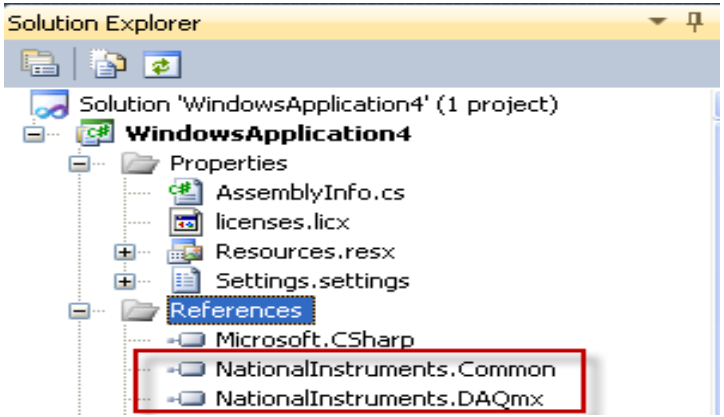
AI – Analog Input

DO – Digital Output

DI – Digital Input

Add References to the **DAQmx** Driver in Visual Studio



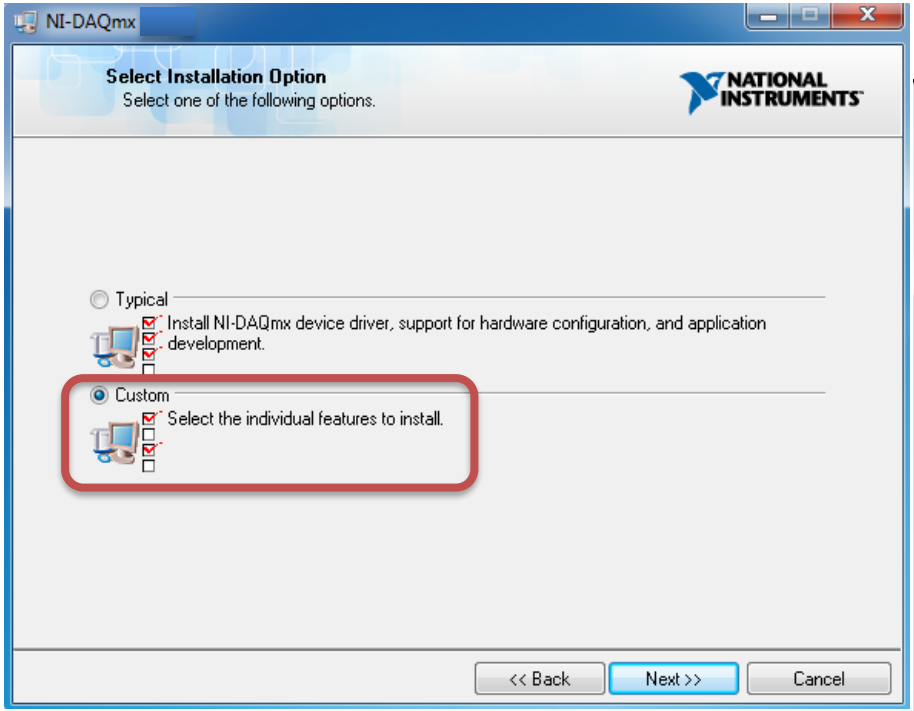



```
NationalInstruments.Common.dll
NationalInstruments.DAQmx.dll
```

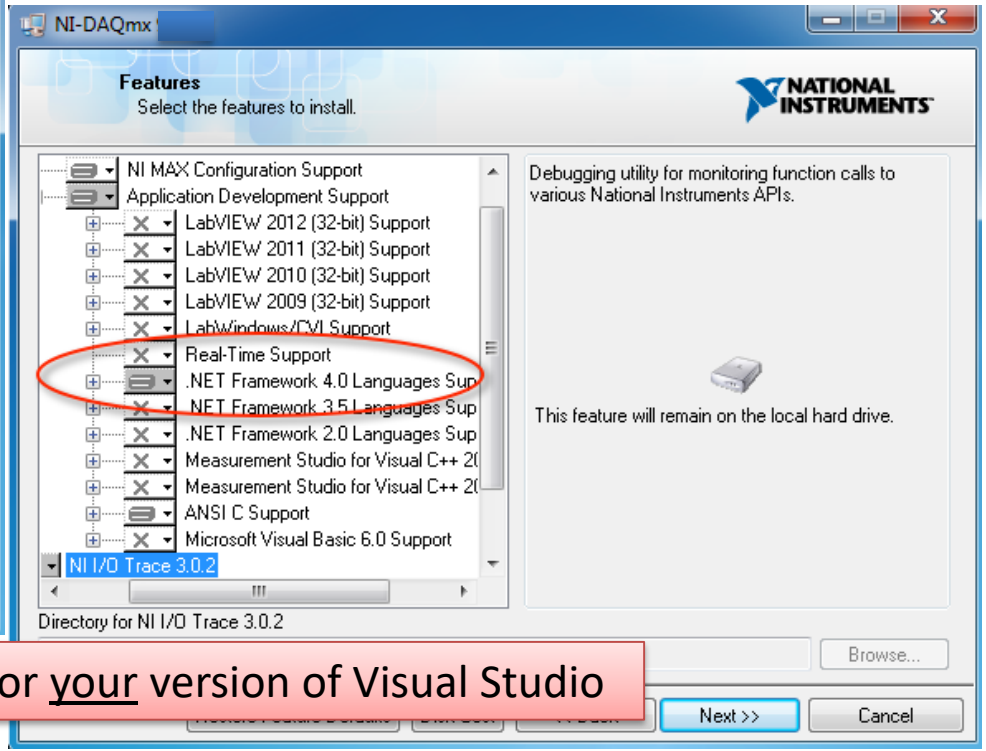
We also need to add the following Namespaces:

```
using NationalInstruments;
using NationalInstruments.DAQmx;
```

Support for .NET Framework with DAQmx Driver



Note! Also included with Measurement Studio



Make sure to add .NET Framework support for <u>your</u> version of Visual Studio

Simple **DAQ** in C# with DAQmx


```
private void btnGetAnalogIn_Click(object sender, EventArgs e)
        {
            Task analogInTask = new Task();

            AIChannel myAIChannel;

            myAIChannel = analogInTask.AIChannels.CreateVoltageChannel(
                "dev1/ai0",
                "myAIChannel",
                AITerminalConfiguration.Differential,
                0,
                5,
                AIVoltageUnits.Volts
                );

            AnalogSingleChannelReader reader = new
             AnalogSingleChannelReader(analogInTask.Stream);

            double analogDataIn = reader.ReadSingleSample();

            txtAnalogIn.Text = analogDataIn.ToString();
        }
```

Analog In Example

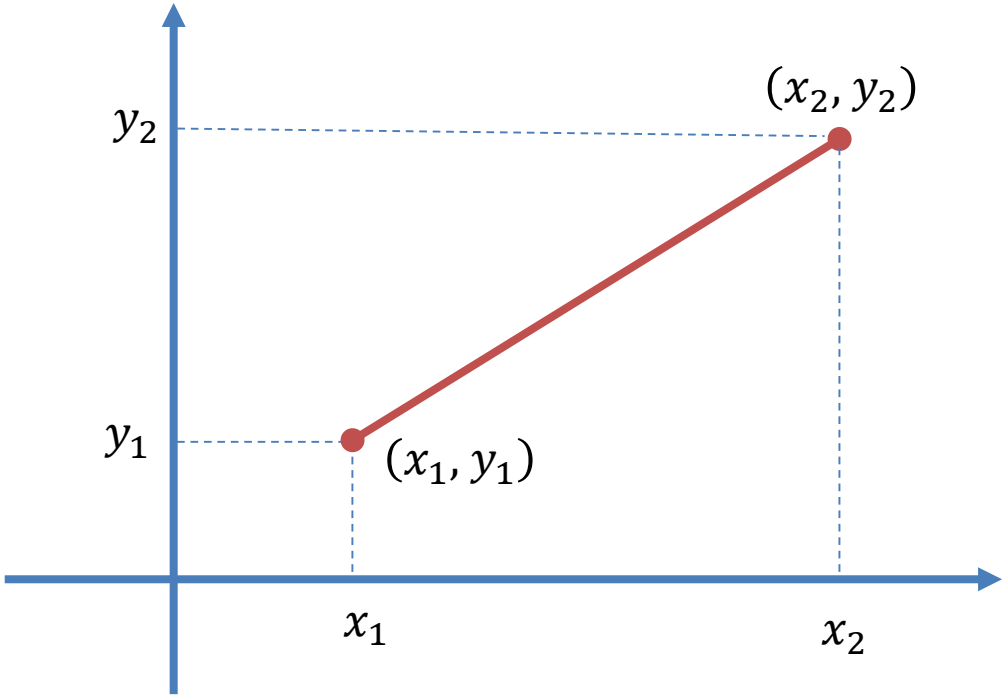
Simple **DAQ** in C# with DAQmx


```
private void btnWriteAnalogOut_Click(object sender, EventArgs e)
        {
            Task analogOutTask = new Task();

            AOChannel myAOChannel;

            myAOChannel = analogOutTask.AOChannels.CreateVoltageChannel(
                "dev1/ao0",
                "myAOChannel",
                0,
                5,
                AOVoltageUnits.Volts
                );

            AnalogSingleChannelWriter writer = new
             AnalogSingleChannelWriter(analogOutTask.Stream);

            double analogDataOut;
            analogDataOut = Convert.ToDouble(txtAnalogOut.Text);

            writer.WriteSingleSample(true, analogDataOut);
        }
```

Analog Out Example

Scaling

Converting from Voltage to Engineering Units



We assume a Linear relationship:

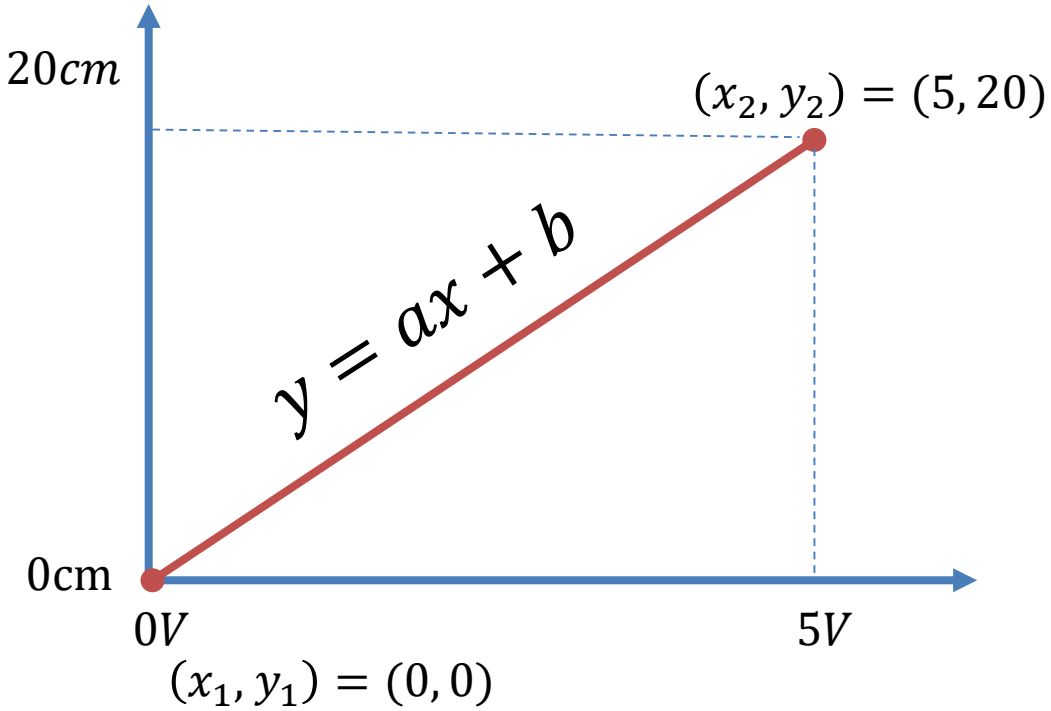
$$y = ax + b$$

We find a and b using this Formula:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

Scaling – Level Tank Example

Converting from Voltage to Engineering Units



For this Level Tank we have:

$0V \rightarrow 0cm$

$5V \rightarrow 20cm$

We use:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

This gives:

$$y[cm] = 4x[V]$$


DEMO

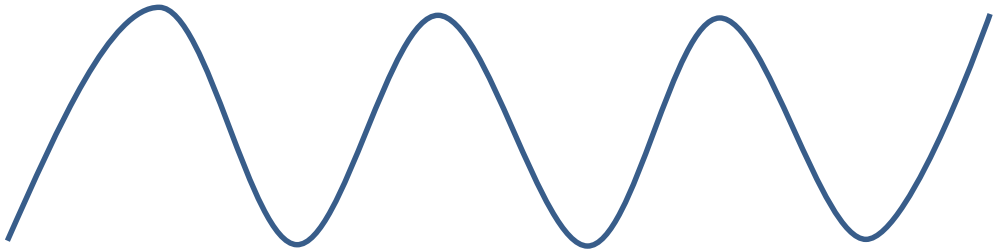
Discrete Lowpass Filter

Hans-Petter Halvorsen, M.Sc.

Lowpass Filter

What is a Lowpass Filter?
Why do we need a Lowpass filter?

Show a plot with Noise



Lowpass Filter

Lowpass Filter Transfer function:

$$H(s) = \frac{y(s)}{u(s)} = \frac{1}{T_f s + 1}$$

T_f - Filter Time constant

Typically: $T_s \leq \frac{T_f}{5}$

Discrete Lowpass Filter

- We need to make a discrete version of the Lowpass Filter in order to be able to implement it in C#

Discrete Lowpass Filter Example

Lowpass Filter Transfer function:

$$H(s) = \frac{y(s)}{u(s)} = \frac{1}{T_f s + 1}$$

Inverse Laplace the differential Equation:

$$T_f \dot{y} + y = u$$

We use the Euler Backward method:

$$\dot{x} = \frac{x_k - x_{k-1}}{T_s}$$

This gives:

$$T_f \frac{y_k - y_{k-1}}{T_s} + y_k = u_k$$

$$y_k = \frac{T_f}{T_f + T_s} y_{k-1} + \frac{T_s}{T_f + T_s} u_k$$

We define:

$$\frac{T_s}{T_f + T_s} \equiv a$$

This gives:

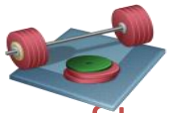
$$y_k = (1 - a) y_{k-1} + a u_k$$

Filter output

Noisy input signal

$$T_s \leq \frac{T_f}{5}$$

This algorithm can be easly implemented in a Programming language



Students: Create a Lowpass Filter in C#

Discrete **Lowpass** Filter – C#

```
class Filter
{
        public double yk;
        public double Ts;
        public double Tf;

        public double LowPassFilter(double yFromDaq)
        {
            double a;
            double yFiltered;

            a = Ts / (Ts + Tf);
            yFiltered = (1 - a) * yk + a * yFromDaq;
            yk = yFiltered;
            return yFiltered;
        }
}
```

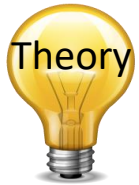
Note! This is just an Example


DEMO

Hardware in the Loop Simulation and Testing

Hans-Petter Halvorsen, M.Sc.

HIL Simulation



- Hardware-in-the-loop (HIL) simulation is a technique that is used in the development and **test of complex process systems**
- The HIL simulation includes a **mathematical model** of the process and a hardware device/ECU you want to test, e.g. an industrial PID controller we will use in our example. The hardware device is normally an **embedded system**
- The main purpose with the HIL Simulation is to **test the hardware device on a simulator** before we implement it on the real process
- It is also very useful for **training** purposes, i.e., the process operator may learn how the system works and operate by using the hardware-in-the-loop simulation
- Another benefit of Hardware-In-the-Loop is that testing can be done **without damaging equipment** or endangering lives**.**

HIL Example

- Typically, a simulator communicates with an “ECU” (“Electronic Control Unit”) via ordinary I/O. Such a system - where the real controller is controlling a simulated process is denoted Hardware-in-the-loop (HIL) simulation.
- The main purpose of this Example is to test the hardware device on a simulator before we implement it on the real process.
- If the mathematical model used in the simulator is an accurate representation of the real process, you may even tune the controller parameters (e.g. the PID parameters) using the simulator.
- We will test the Fuji PGX5 PID controller on a model, and if everything is OK we will implement the controller on the real system.

HIL Simulation & Testing Setup

PID Control

Analog Out
(Process Value)
1-5V

AO0

GND

Note!!

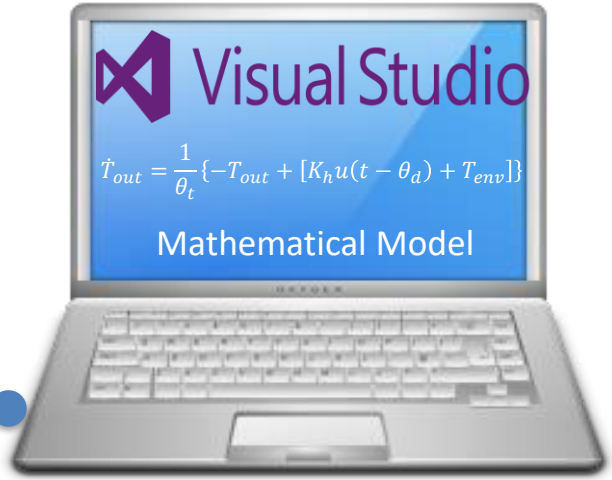
AI0+

AI0-

Analog In
(Control Signal)
0-5V

I/O Module
(USB-6008)

USB



Model of Process (Air Heater)
Computer (with Visual Studio/C#)

With this setup you can Test the Fuji PID using HIL Simulation and Testing

Fuju PXG5 + Real Air Heater + PC for Monitoring

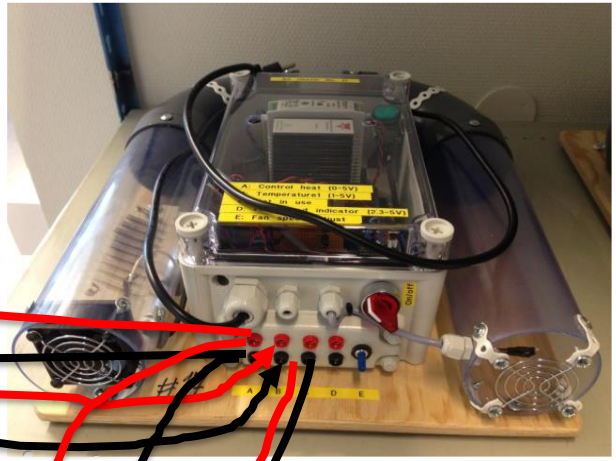
Industrial PID Controller

Process Value

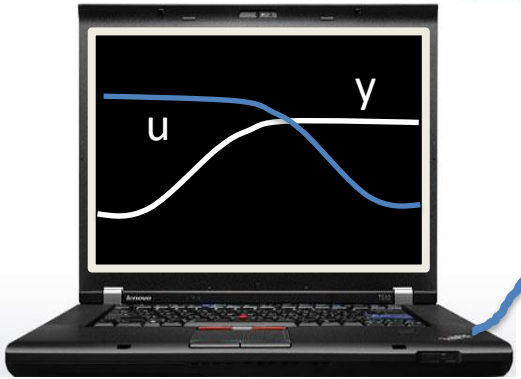
1-5V

0-5V

Control Signal



Trending/Monitoring the Process Value and Control Signal on the PC



PC + C# App

USB

Process Value

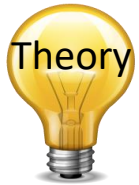
1-5V

Control Signal

0-5V

With this setup you can Monitor the Process Value and Control Signal on your PC

HIL with Fuji PXG5/PXR5 PID

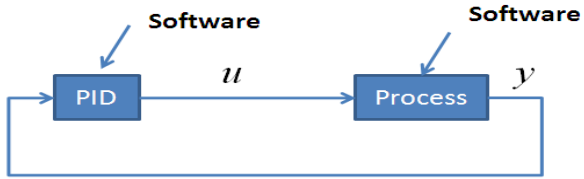


- It may be very useful to test a controller function with a simulated process before the controller is applied to the real (physical) process.
- If the mathematical model used in the simulator is an accurate representation of the real process, you may even tune the controller parameters (e.g. the PID parameters) using the simulator.

HIL Simulation Lab - Step-by-step

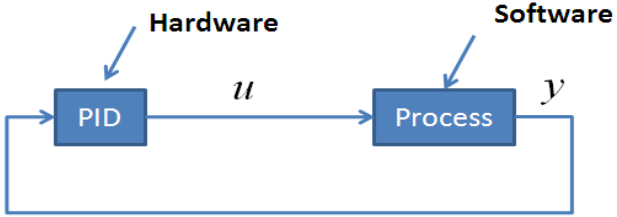
1

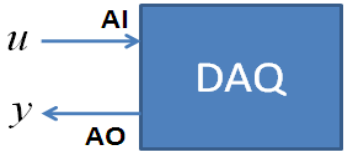
Step 1: Ordinary Software Simulation:



2

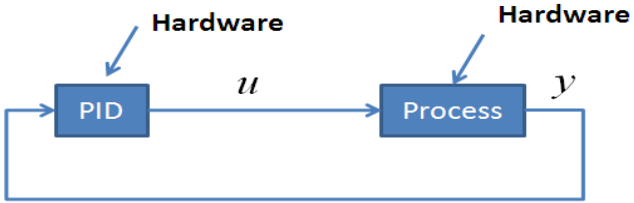
Step 2: HIL Simulation:





3

Step 3: Running the Real System:




DEMO

Measurement Studio

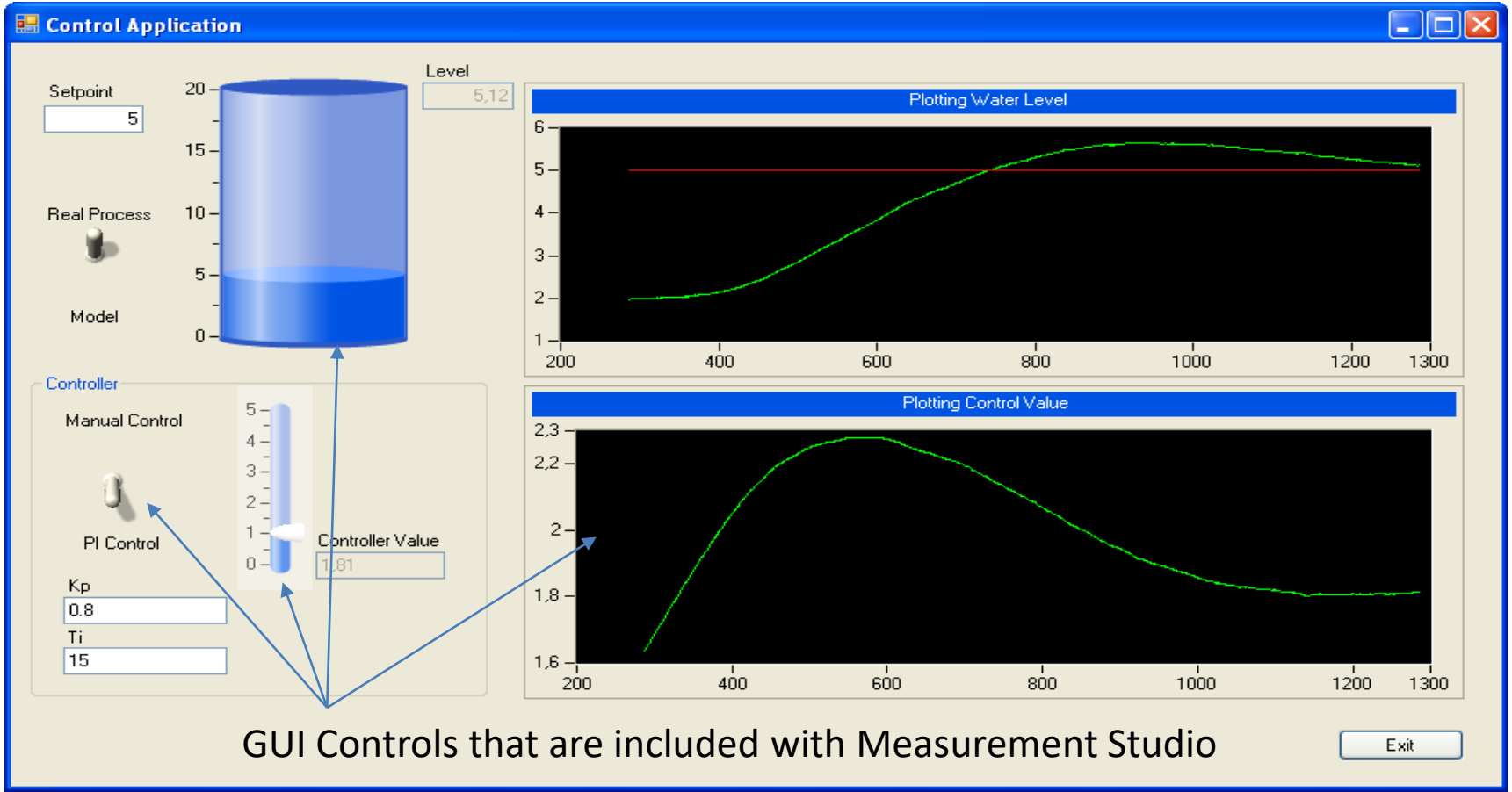
Hans-Petter Halvorsen, M.Sc.

Measurement Studio

- C# is a powerful programming language, but has few built-in features for measurement and control applications.
- Measurement Studio is used for development of measurement, control and monitoring applications using .NET and Visual Studio.
- Measurement Studio is an add-on to Visual Studio which makes it easier to create such applications. With Measurement Studio we can implement Data Acquisition and a graphical HMI.
- Download Software here: http://www.ni.com/academic/download

Measurement Studio

Visual Studio + Measurement Studio



Trending Data

You may use the “**WaveformGraph**” Control included with Measurement Studio..

You only need one line of code, e.g. inside a Timer Event:

```
…
{
        …
        waveformGraph.PlotYAppend(value);
}
```

Name of your WaveformGraph control

Name of the Method to use

Name of the variable with Temperature data

Datalogging

Hans-Petter Halvorsen, M.Sc.

Datalogging

Typically you will want to log your Data.

Typical Examples:

- File
- OPC
- Database

With Measurement Studio you can easily Log Data to Files or an OPC Server

Datalogging using OPC and SQL Server are explained in other Videos I have made.

Hans-Petter Halvorsen, M.Sc.

University College of Southeast Norway

www.usn.no

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

Blog: http://home.hit.no/~hansha/