https://www.halvorsen.blog

Industry 4.0 and Automation

Hans-Petter Halvorsen

Table of Contents

*If you prefer, you can use Python in some parts of the assignment

Introduction

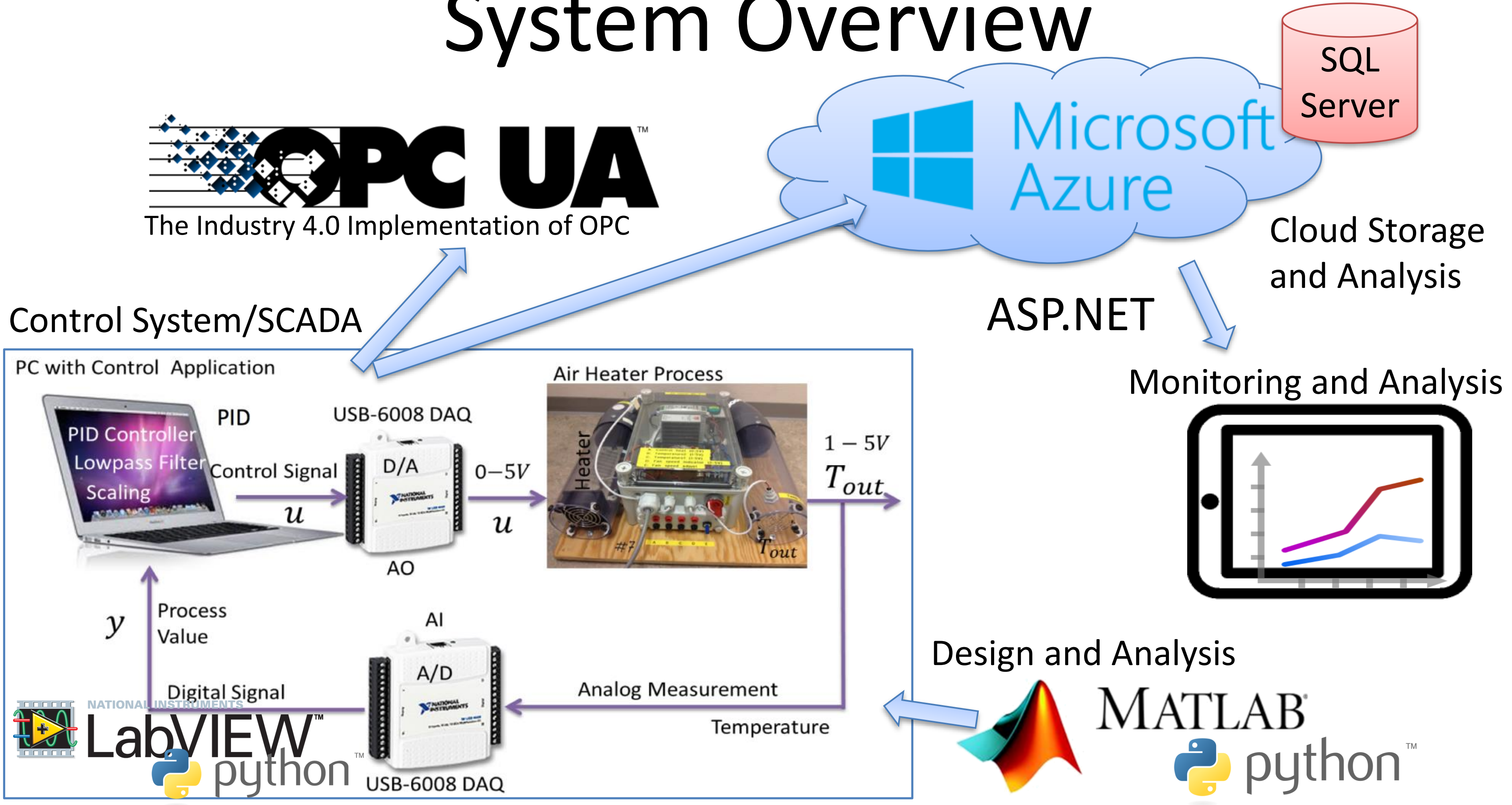
- In this Assignment we will create the Next Generation **Industry 4.0** Control System using **OPC UA** and **Cloud** Services like Microsoft Azure, Cloud Computing and Analysis using modern **Web Technology**.
- **Industry 4.0** is the new term for the combination of industry, automation and the current **Internet of Things (IoT)** technology.

Lab Assignment Overview

1. Modelling and Simulation. **Control Design and Analysis with *MATLAB***
   – *Frequency Response*, Stability Analysis, Simulations, etc.
2. Implement **Control System** in LabVIEW
3. Use ***OPC UA*** – The Industry 4.0 Implementation of OPC
4. Cloud-based **Datalogging**. SQL Server stored in ***Microsoft Azure***
5. **Monitoring and Analysis** in the Cloud. Web-based *ASP.NET*/C# system hosted at Microsoft Azure
6. Give an overview and analyze issues regarding **Cyber Security** and **GDPR** for the system you create

See next slides for details...

System Overview



Keywords

- Practical **Industry 4.0** Applications
- Control Theory including **Frequency Response** and Stability Analysis
- Control Design and Simulations
- Practical Implementation of Control Systems and **PID**
- **OPC UA** - The Industry 4.0 Implementation of OPC
- **MATLAB** (Design and Simulations) and **LabVIEW** (Implementation) Programming
- **Cloud** Hosting and Computing (**Microsoft Azure**)
- Monitoring and Analysis – **Web** Application, **ASP.NET**

Learning Goals

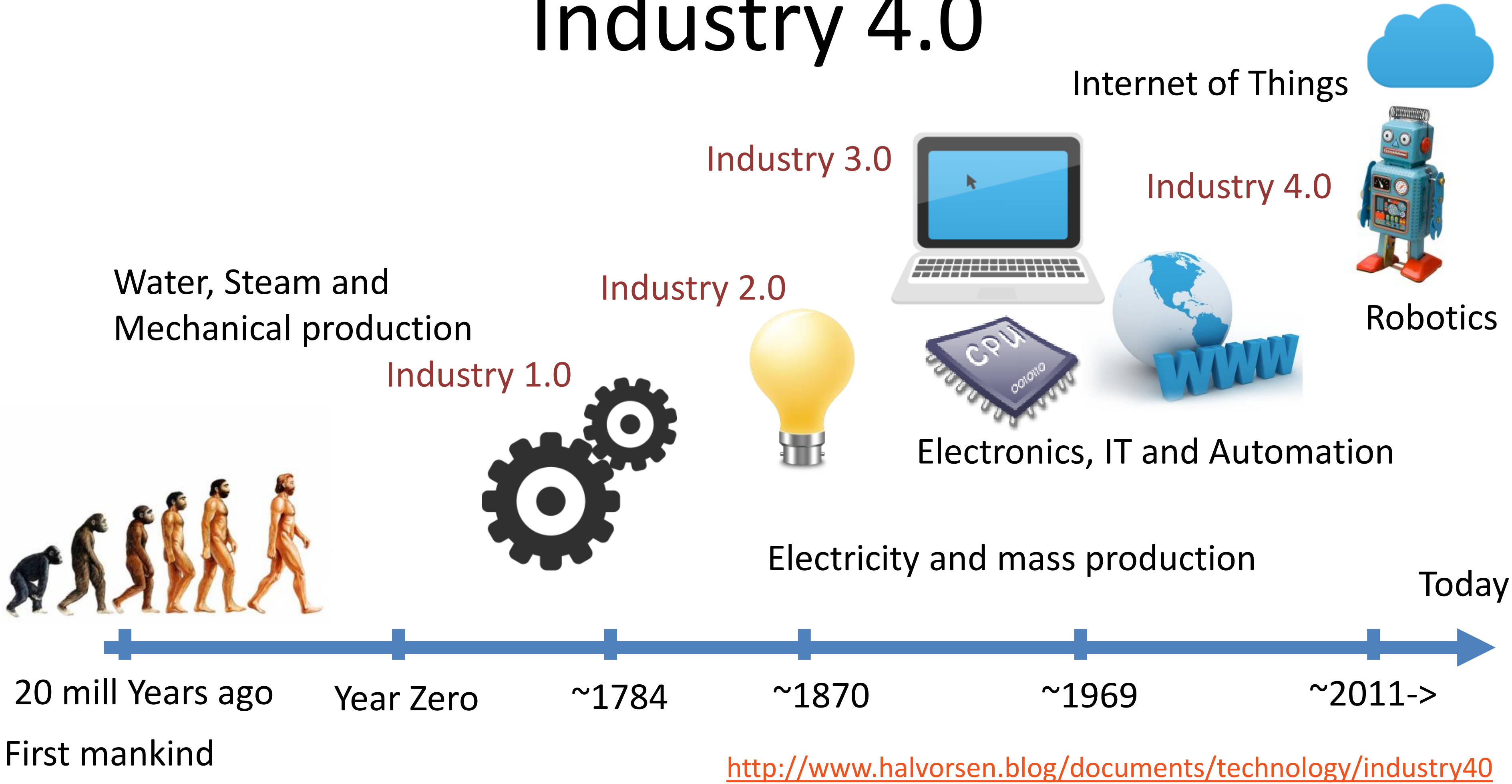
- Introduction to the term Industry 4.0 and how it affects the next generation Control and Automation Systems
- Learn practical skills in Modelling, Control and Simulation, Digital Twins
- Learn practical implementation of PID Control Systems
- Learn practical use of Frequency Response Design and Analysis for Feedback Systems
- Learn more Programming; LabVIEW, MATLAB, Python, C#, Web Programming
- Learn about Hardware-Software Interactions
- Learn Practical Skills and Implementations
- Learn Software Installation, which can be cumbersome with many pitfalls
- Learn to use and create Software
- Learn about Cloud Hosting and Cloud Computing
- Learn about Web Technology

Industry 4.0: Industrial IT + Automation

- Industrial IT is the integration of Automation and Information Systems across the business.
- You could say Industrial IT is use of IT in industrial applications, everything from Process Control Systems, Sensor Technology, Data Acquiring, Data Logging and Monitoring and Software and Systems Engineering.
- You need to have knowledge of Data Acquisition, Database Systems, Data Communication and Networks, Automation and Control, etc.
- Terms such as Internet of Things (IoT), Smart Technology, Cloud Computing are key factors within Industry 4.0

http://www.halvorsen.blog/documents/technology/industry40

Industry 4.0



http://www.halvorsen.blog/documents/technology/industry40

Industry 4.0

- Industry 4.0 is the new buzzword for the combination of industry, automation and the current Internet of Things (IoT) technology.
- IIoT – Industrial use of IoT Technology. Industrial Internet of Things (IIoT) is another word for Industry 4.0.
- You could say that IoT is consumer oriented with applications like Smart Home, Home Automation, etc., while IIoT has more industrial focus and applications.
- The term "Industrie 4.0" was first used in 2011 in Germany.
- Industry 4.0 is also called the fourth industrial revolution.

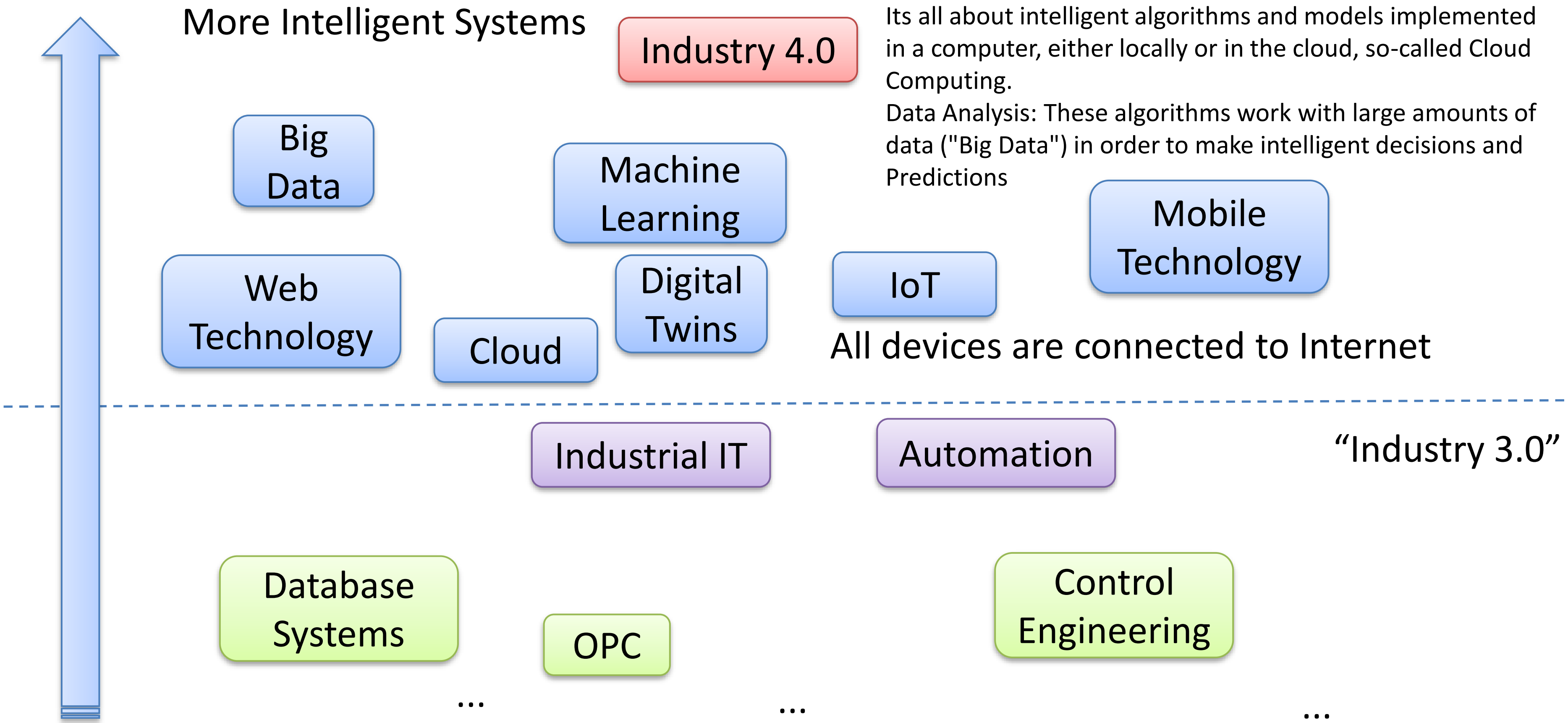
Industry 4.0

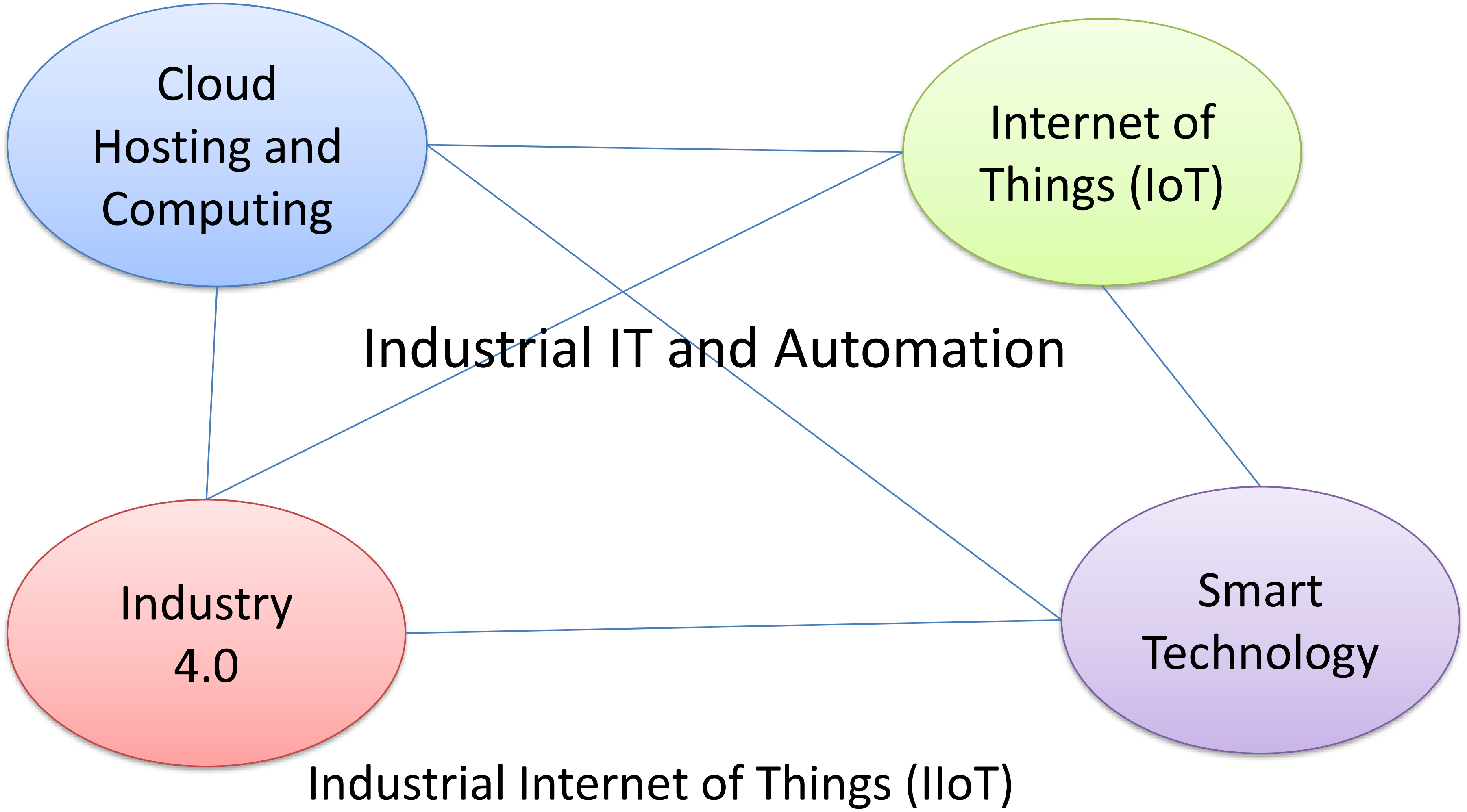
Industry 4.0 is also called the fourth industrial revolution.

- **Industry 1.0**: Mechanization of production using Water and Steam Power.
- **Industry 2.0**: Mass production with the help of Electric Power.
- **Industry 3.0**: The Digital Revolution. From Analog to Digital Devices and Signals. Use of Electronics and IT to further Automate Production
- **Industry 4.0**: The combination of industry, automation, digitalization and the current Internet of Things (IoT) technology.

Industry 4.0

Its all about intelligent algorithms and models implemented in a computer, either locally or in the cloud, so-called Cloud Computing.
Data Analysis: These algorithms work with large amounts of data ("Big Data") in order to make intelligent decisions and Predictions




Cloud Hosting and Computing
Internet of Things (IoT)
Industrial IT and Automation
Industry 4.0
Smart Technology
Industrial Internet of Things (IIoT)

SCADA Systems

SCADA History:

- 1. Generation: Early SCADA system computing was done by large minicomputers.
  - Common network services did not exist at the time SCADA was developed.
  - Thus SCADA systems were independent systems with no connectivity to other systems
- 2. Generation: Distributed Systems

  – The system was distributed across multiple stations which were connected through a LAN.
- 3. Generation: Networked Systems
- **Next Generation - 4. Generation: Internet of Things (IoT) and Industry 4.0** (Which is the focus in this Assignment)

Cloud Computing

- **SaaS** – Software as a Service
  - Software as a Service provides you with a completed product that is run and managed by the service provider.
  - You don't have to worry about the installation, setup and running of the application. Service provider will do that for you. You just have to pay and use it through some client.
  - Examples: Google Apps, Microsoft Office 365, web-based email systems
- **PaaS** – Platform as a Service
  - Providing a platform on which software can be developed and deployed.
  - Platforms as a service remove the need for organizations to manage the underlying infrastructure (usually hardware and operating systems) and allow you to focus on the deployment and management of your applications.
  - Examples: AWS, Microsoft Azure,... (e.g., use a preinstalled Web Server without worrying about anything else)
- **IaaS** – Infrastructure as a Service
  - Providing a full infrastructure in the cloud, such as Virtual Machines, Servers, OS, ...
  - Highest level of flexibility and management control over your IT resources and is most similar to existing IT resources that many IT departments and developers are familiar with today.
  - Examples: AWS, Microsoft Azure,...

Software

Software

NATIONAL INSTRUMENTS LabVIEW™

Microsoft® SQL Server®

python™

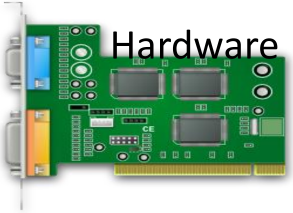
MATLAB®

Visual Studio

C#

Microsoft Azure

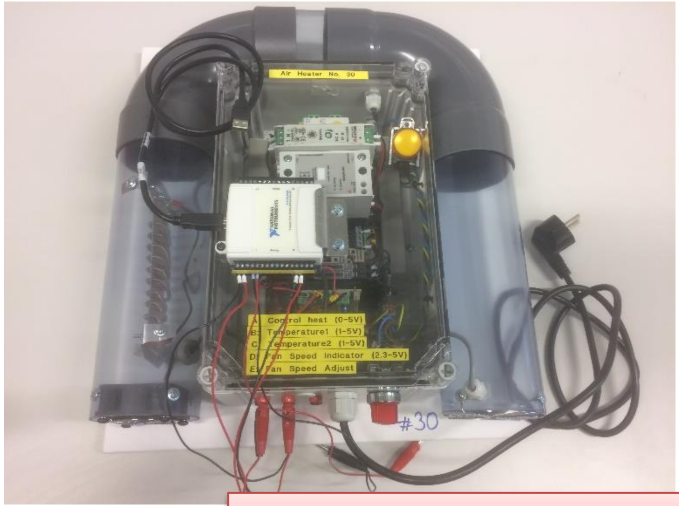
Hardware



Your Personal Computer

DAQ Device,
e.g. USB-6008

Air Heater



Only available in the Laboratory!

Online Students: You can do 95% of the assignment without this hardware using simulators and a provided “Black Box Model”

The teacher have not done all the Tasks in detail, so he may not have all the answers! That's how it is in real life also!

Very often it works on one computer but not on another. You may have other versions of the software, you may have installed it in the wrong order, etc...
In these cases Google is your best friend!

The Teacher dont have all the answers (very few actually ☹)!! Sometimes you just need to “Google” in order to solve your problems, Collaborate with other Students, etc. Thats how you Learn!

Troubleshooting & Debugging

Visual Studio

Use the **Debugging Tools** in your Programming IDE.
Visual Studio, LabVIEW, etc. have great Debugging Tools! Use them!!

Use available Resources such as User Guides, Datasheets, Text Books, Tutorials, Examples, Tips & Tricks, etc.

My System is not Working??

Multimeter, etc.

NATIONAL INSTRUMENTS LabVIEW

"Google It"!

You probably will find the answer on the Internet

Google

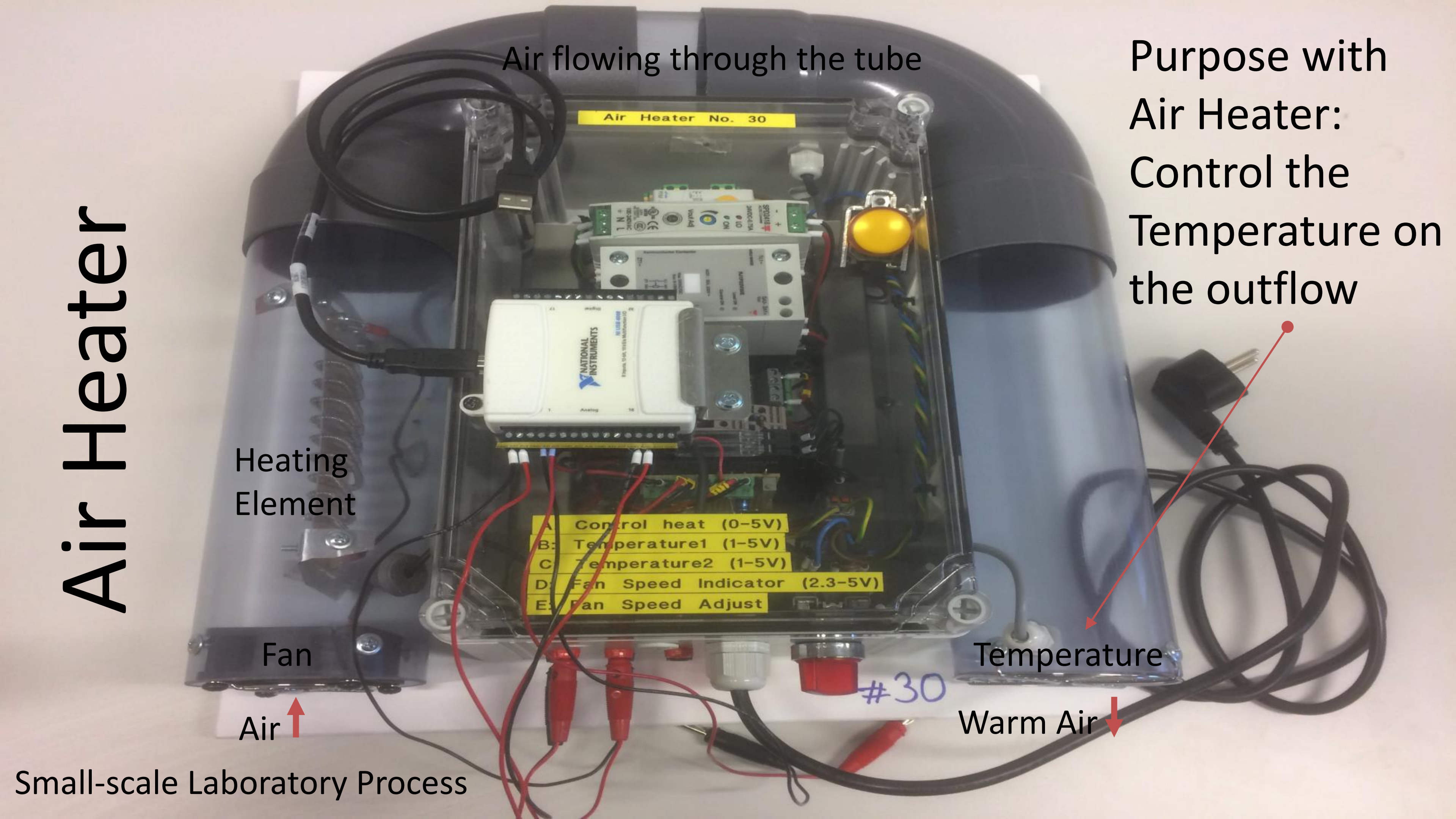
Another person in the world probably had a similar problem

Check your electric circuit, electrical cables, DAQ device, etc. Check if the wires from/to the DAQ device is correct. Are you using the same I/O Channel in your Software as the wiring suggest? etc.

Modelling and Simulation

This part is known from previous courses, feel free to reuse previous code and results

Hans-Petter Halvorsen



Air Heater
Air flowing through the tube
Purpose with Air Heater: Control the Temperature on the outflow
Air Heater No. 30
A Control heat (0-5V)
B: Temperature1 (1-5V)
C: Temperature2 (1-5V)
D: Fan Speed Indicator (2.3-5V)
E: Fan Speed Adjust
Heating Element
Fan
Air
Temperature
#30
Warm Air
Small-scale Laboratory Process

Air Heater Mathematical Model

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

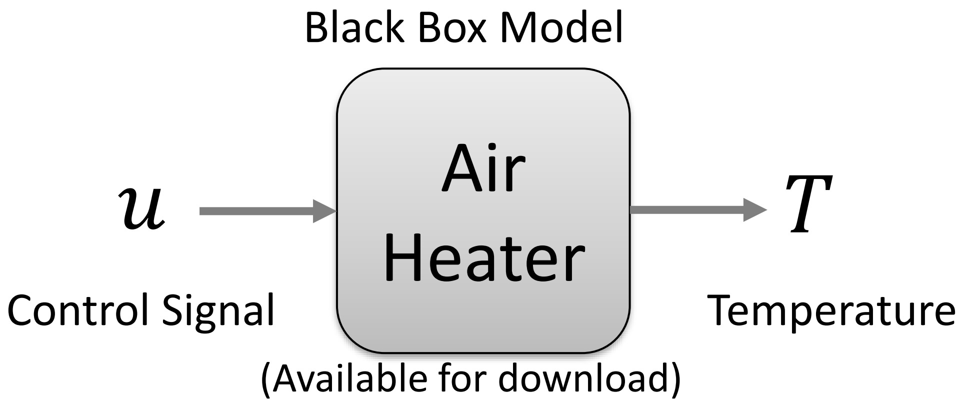
Where:

- T_{out} is the air temperature at the tube outlet
- $u\ [V]$ is the control signal to the heater
- $\theta_t\ [s]$ is the time-constant
- $K_h\ [deg\ C\ /\ V]$ is the heater gain
- $\theta_d\ [s]$ is the time-delay representing air transportation and sluggishness in the heater
- T_{env} is the environmental (room) temperature. It is the temperature in the outlet air of the air tube when the control signal to the heater has been set to zero for relatively long time (some minutes)

“Real Process” → “Black Box Model”

- The Real Air Heater is only available in the Laboratory
- A “Real” Air Heater will we provided as a “black box”. Actually, it is a LabVIEW SubVI where the Block Diagram and the Process Parameters are hidden.
- Useful for Online Students and when you are working with the Assignment outside the Laboratory

“Real Process” → “Black Box Simulator”



You can assume that the following model is a good representation of the “Black Box Model”:

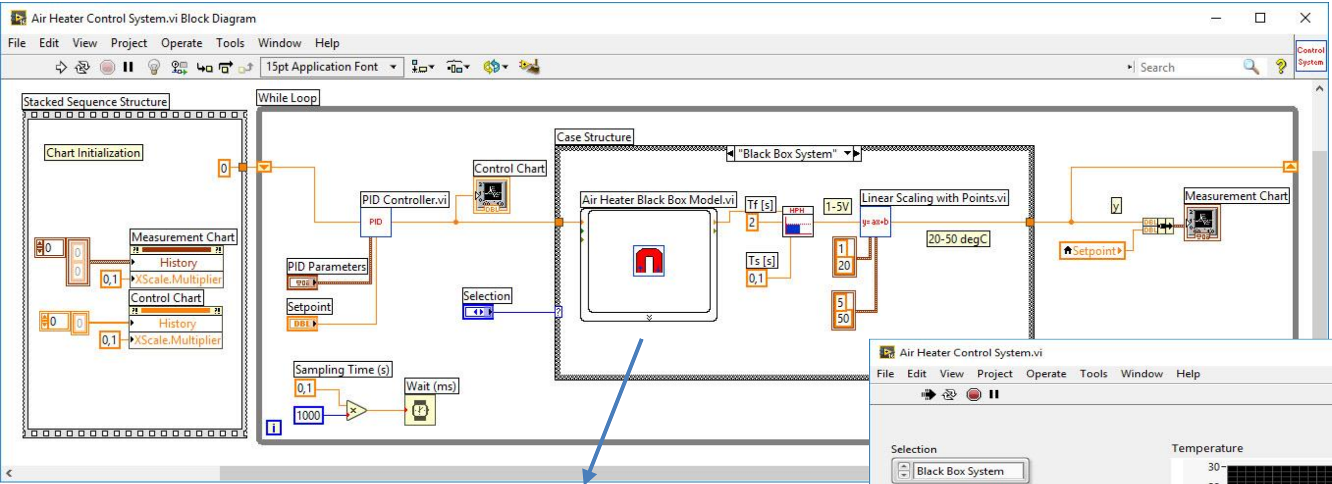
$$\dot{T}_{out} = \frac{1}{\theta_t}\{-T_{out} + [K_h u(t - \theta_d) + T_{env}]\}$$

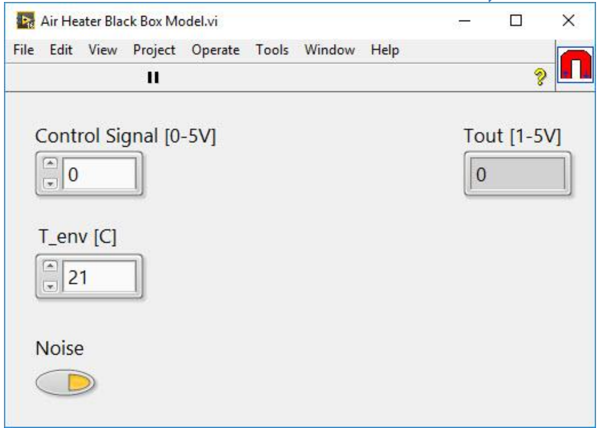
This means you need to need to find $\theta_t, K_h, \theta_d, T_{env}$

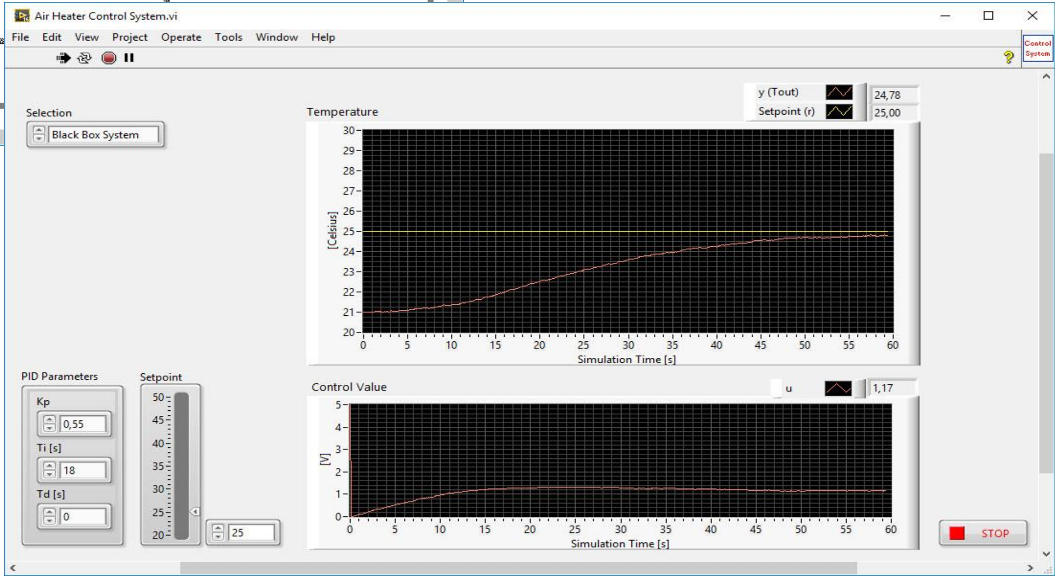
T_{env} is the temperature in the room

"Real Process" → "Black Box Model"

Here we see an example where we control the "Black Box" Model, which we "pretend" is the Real System







Model Parameters

Find Proper Model Parameters using LabVIEW

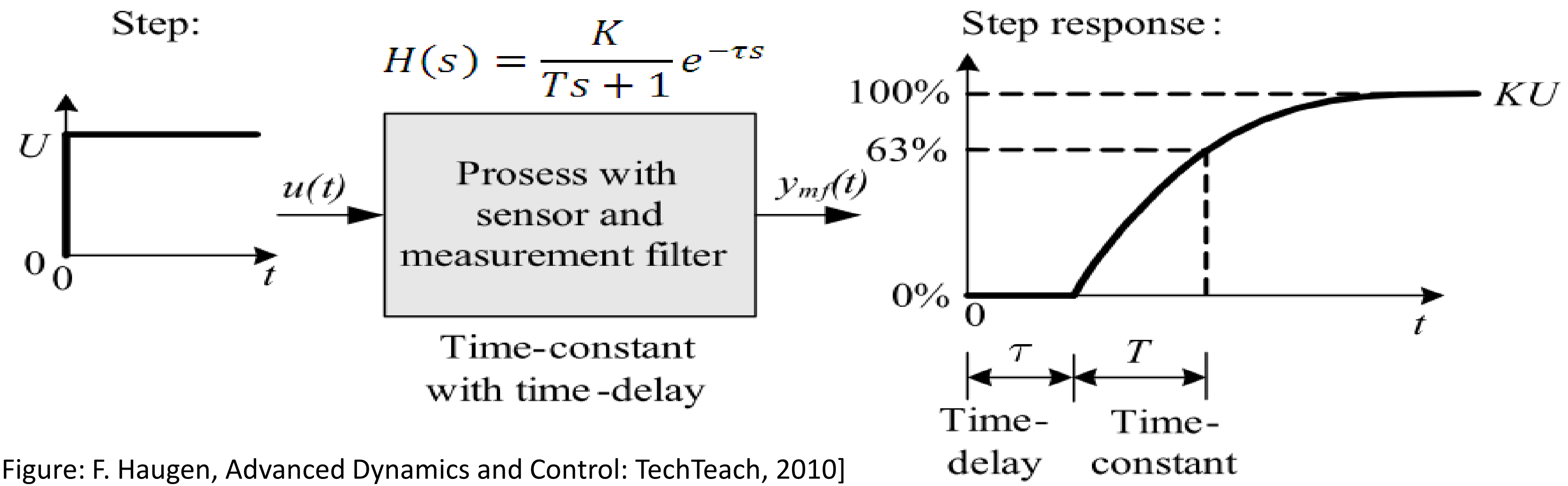
Suggested Steps:

1. Use the “Step Response” method to find initial model parameters
2. Then use “Trial and Error” method to verify and “fine-tune” if necessary

Use the “Black Box Model” when you are not in the laboratory

Theory

Step Response Method



[Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

Assuming e.g. a 1.order model you can easily find the model parameters (Process Gain, Time constant and a Time delay if any) from the step response of the real system/or “Black-box” Simulator (plotting logged data)

Air Heater Transfer function

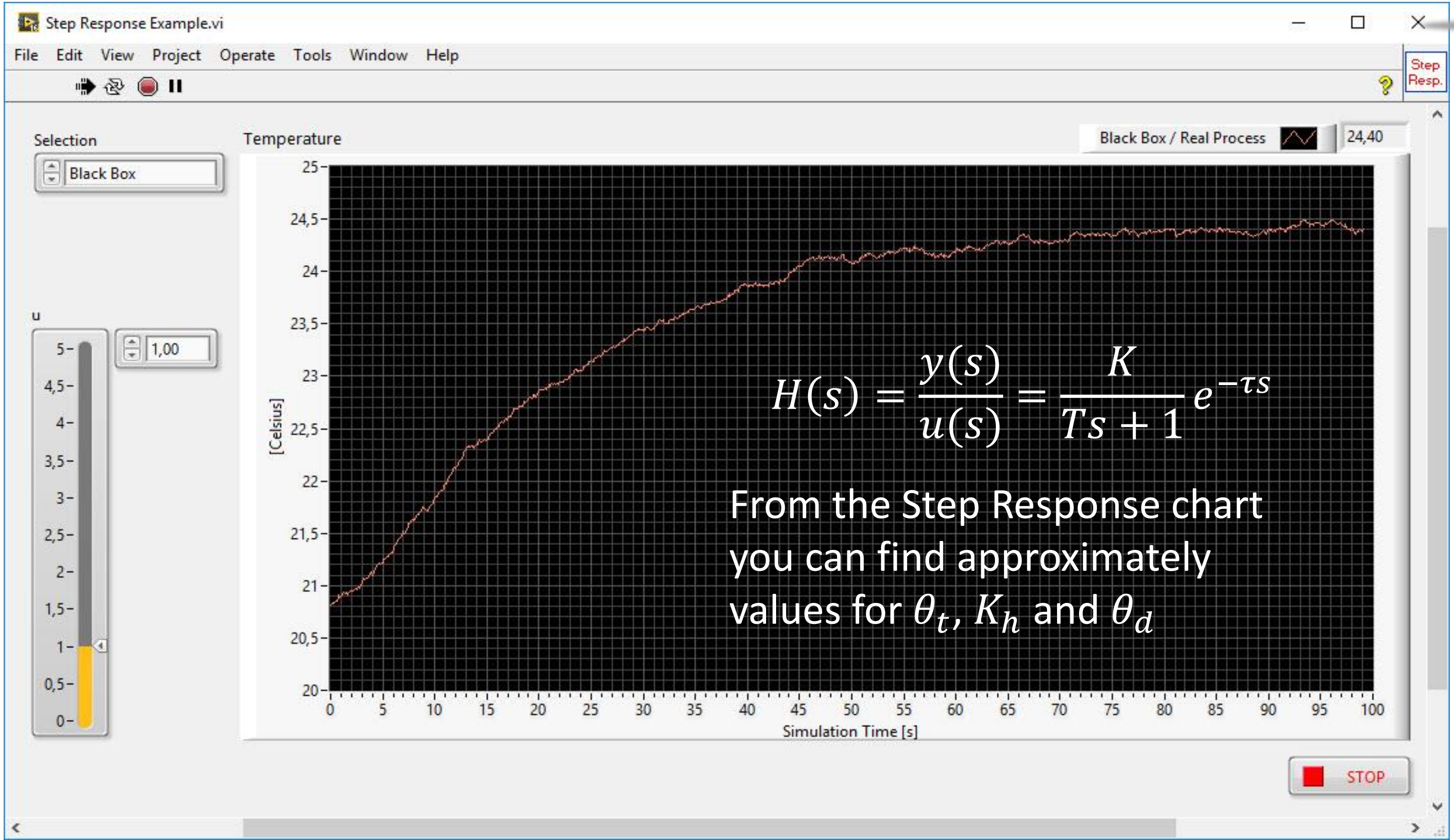
The Air Heater process is a 1.order process with time-delay, so a transfer function on the following general form should be expected:

$$H(s) = \frac{y(s)}{u(s)} = \frac{K}{Ts + 1} e^{-\tau s}$$

Tip! Use **Laplace** transformation on the differential equation for the Air Heater and find the transfer function from $u(s)$ to $T_{out}(s)$.

$$H_{heater}(s) = \frac{T_{out}(s)}{u(s)} = ?$$

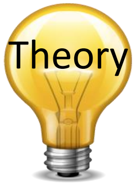
Step Response Method

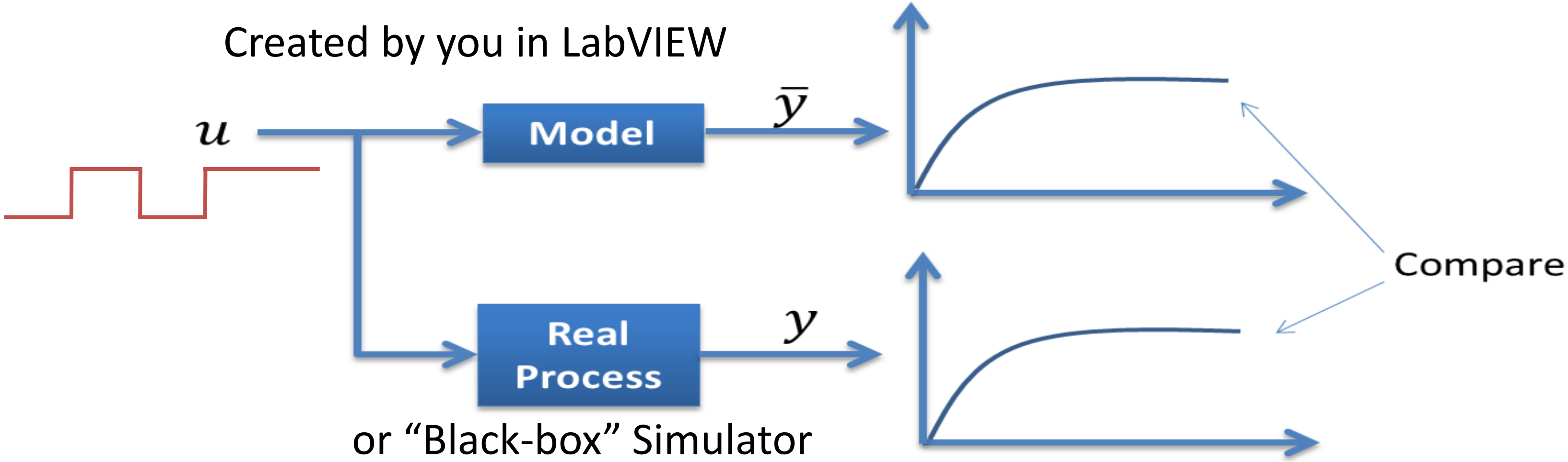


$$H(s) = \frac{y(s)}{u(s)} = \frac{K}{Ts+1}e^{-\tau s}$$

From the Step Response chart you can find approximately values for θ_t, K_h and θ_d

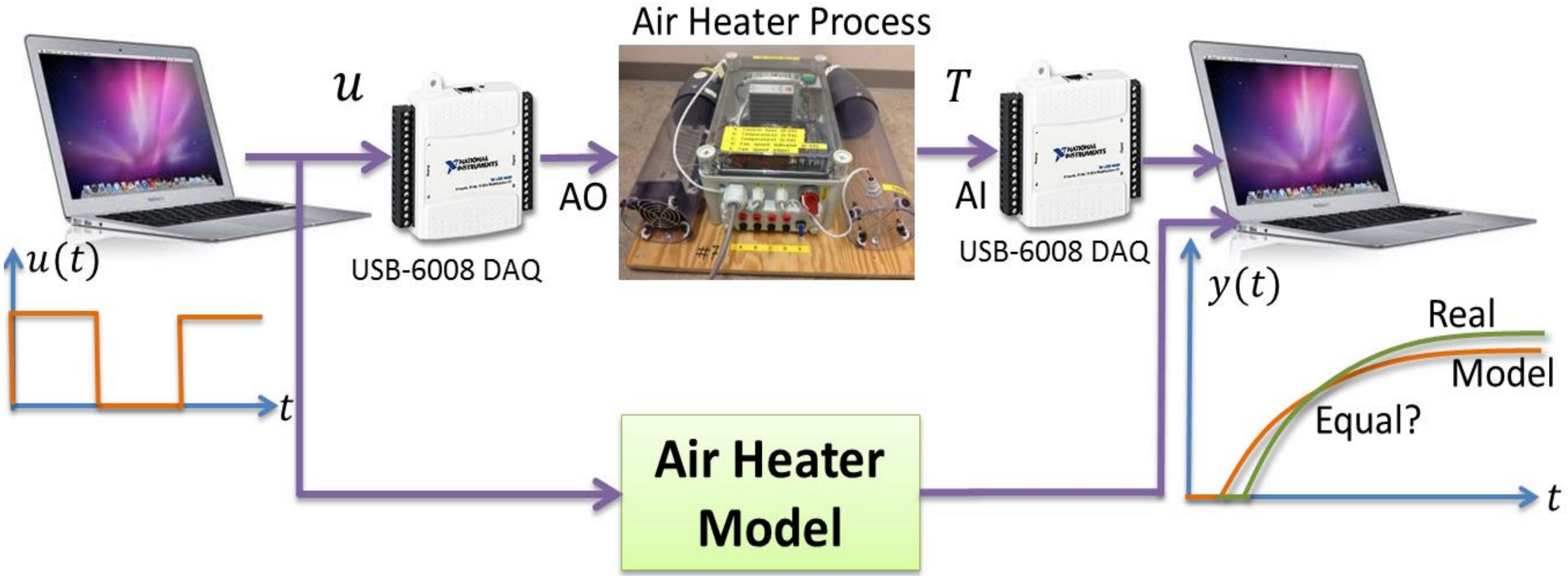
Trial & Error Method





Adjust model parameters and then compare the response from the real system with the simulated model. If they are "equal", you have probably found a good model (at least in that working area)

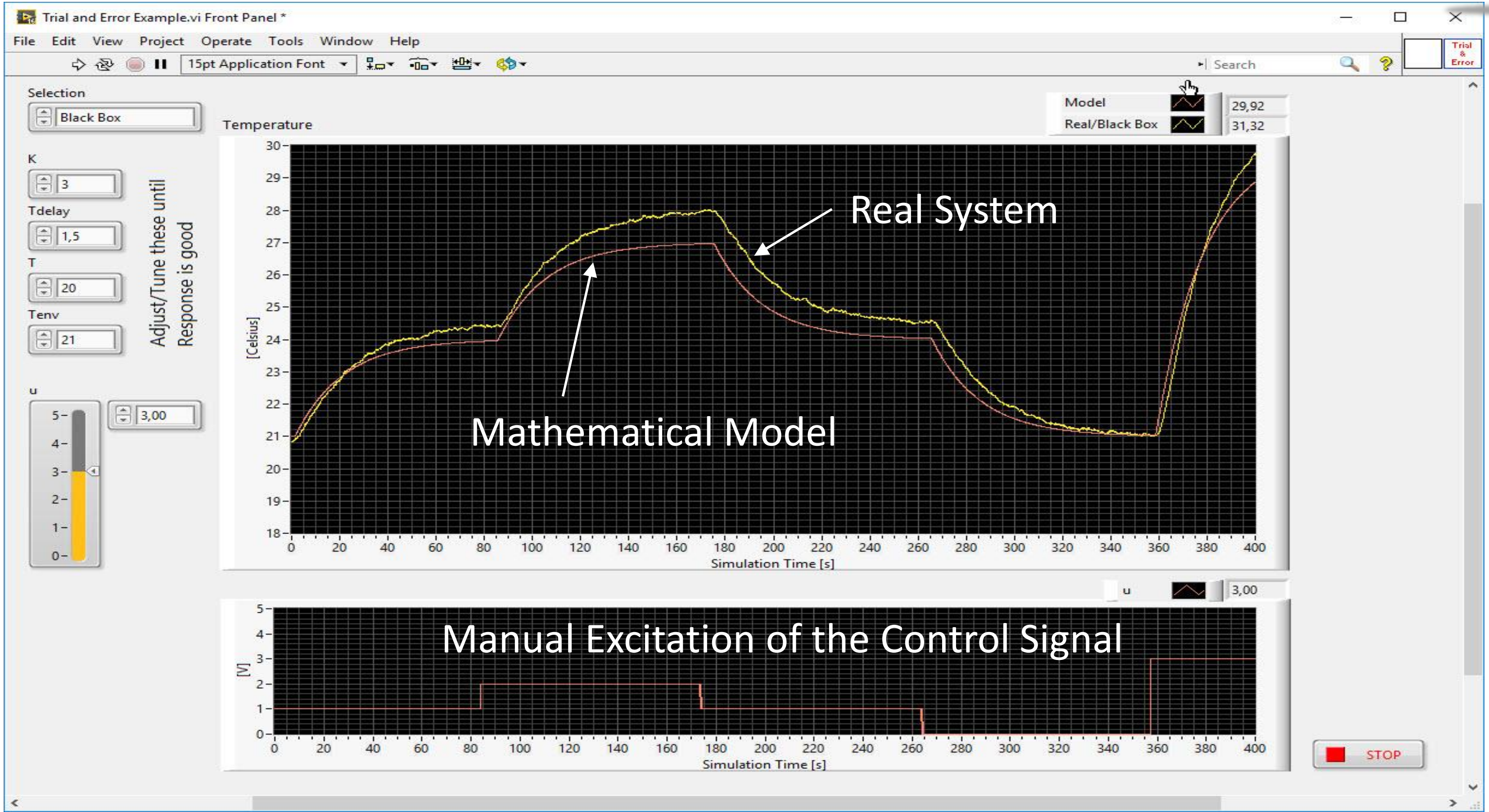
Model Validation



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

You always validate the model by running the model in parallel with the real system, or test it against logged data from the real system.

Trial and Error and Model Validation



Congratulations! - You are finished with the Task

Frequency Response using MATLAB

If you prefer, you can use Python

This part is known from previous courses, feel free to reuse previous code or use the examples given here as a starting point for your work.

Hans-Petter Halvorsen

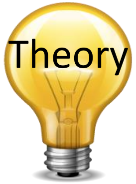
Transfer Function

- Since much of the control design theory is based on transfer functions, we need to find the transfer function $H(s)$ for the Air Heater process based on the given differential equation.
- **Tip!** Use **Laplace** transformation on the differential equation for the Air Heater and find the transfer function from $u(s)$ to $T_{out}(s)$.
- The Air Heater process is a 1.order process with time-delay, so a transfer function on the following general form should be expected:

$$H(s) = \frac{y(s)}{u(s)} = \frac{K}{Ts + 1} e^{-\tau s}$$

- Implement the transfer function of the Air Heater in MATLAB, perform step response, find poles and zeros, etc. using MATLAB.

$$H_{heater}(s) = \frac{T_{out}\ (s)}{u(s)} = ?$$



1. order system with time-delay

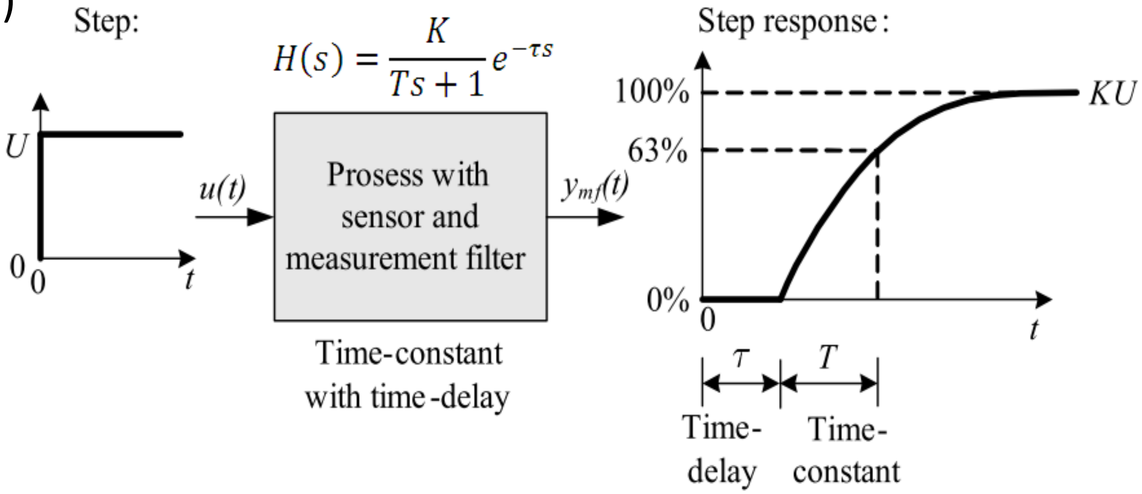
A 1.order transfer function with time-delay may be written as:

$$H(s) = \frac{K}{Ts+1}e^{-\tau s}$$

Where K is the Gain, T is the Time constant and τ is the time-delay

(The Air Heater is such a system)

$$H_{heater}(s) = \frac{T_{out}(s)}{u(s)} = ?$$



[Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

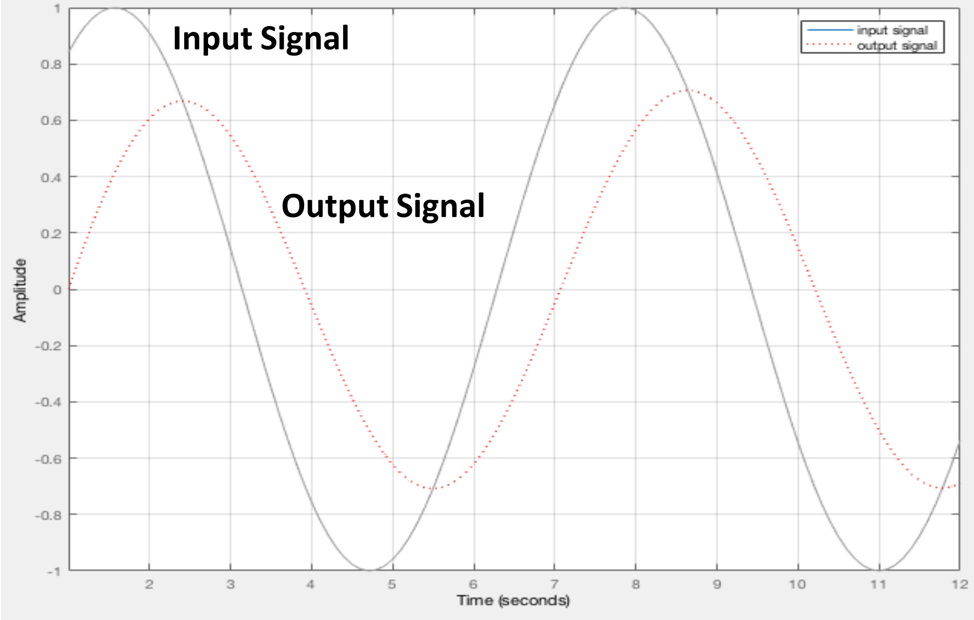
Theory

Frequency Response

- The frequency response of a system is a frequency dependent function which expresses how a sinusoidal signal of a given frequency on the system input is transferred through the system. Each frequency component is a sinusoidal signal having certain amplitude and a certain frequency.
- The frequency response is an important tool for analysis and design of signal filters and for analysis and design of control systems.
- The frequency response can be found experimentally or from a transfer function model.
- The frequency response of a system is defined as the steady-state response of the system to a sinusoidal input signal. When the system is in steady-state, it differs from the input signal only in amplitude/gain (A) and phase lag (ϕ).

Theory

Frequency Response



Input Signal

$$u(t) = U \cdot \ sin\omega t$$

Amplitude Frequency

Dynamic System

$$H(s)$$

Output Signal

$$y(t) = \underbrace{UA}_{Y} \ sin(\omega t + \phi)$$

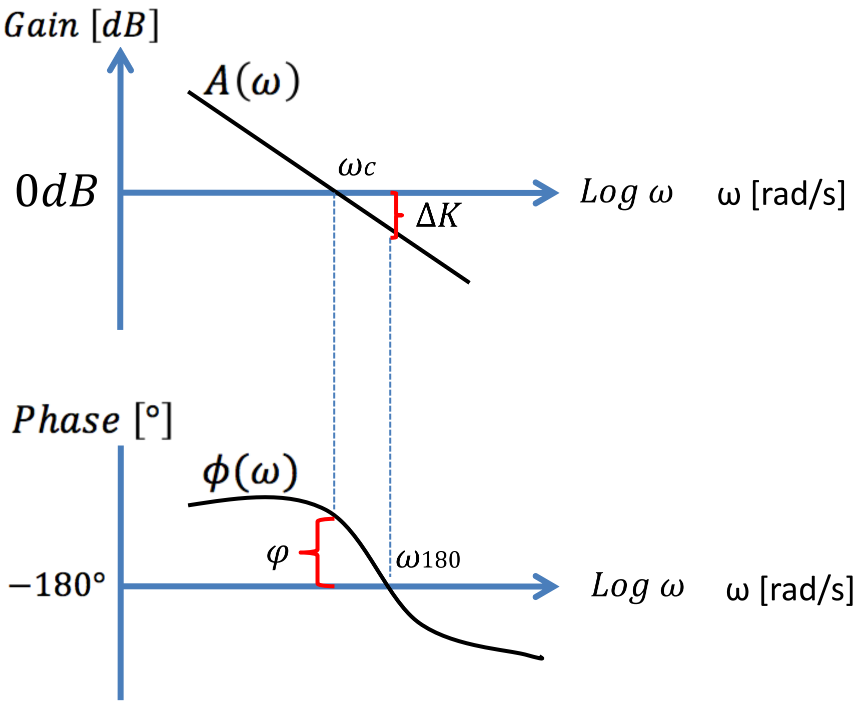
Gain Phase Lag

The frequency response of a system expresses how a sinusoidal signal of a given frequency on the system input is transferred through the system. The only difference is the gain and the phase lag.

Bode Diagram

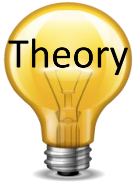
Theory

You can find the Bode diagram from experiments on the physical process or from the transfer function (the model of the system). A simple sketch of the Bode diagram for a given system:



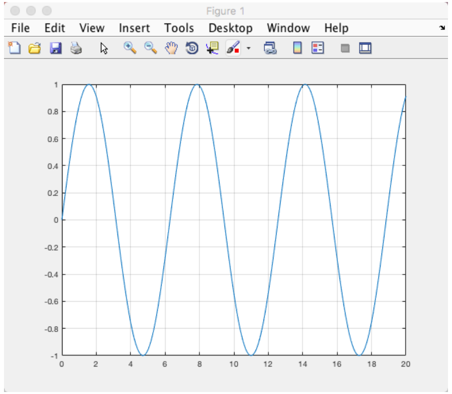
The Bode diagram gives a simple Graphical overview of the Frequency Response for a given system. A Tool for Analyzing the Stability properties of the Control System.

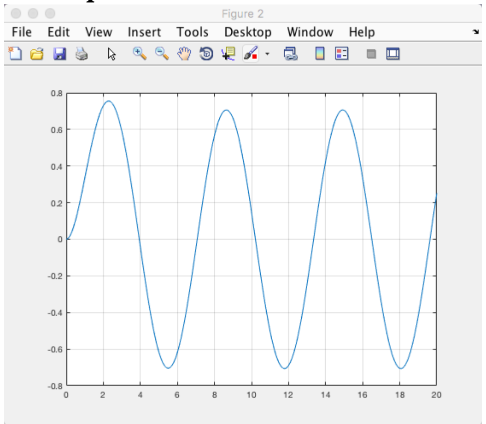
Frequency Response - Definition



$u(t) = U \cdot \sin\omega t$

$$y(t) = \underbrace{UA}_{Y} \sin(\omega t + \phi)$$





$\omega = 1\, rad/s$

$\omega = 1\, rad/s$

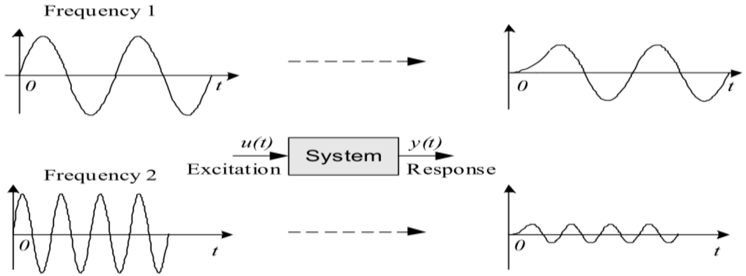
and the same for Frequency 2, 3, 4, 5, 6, etc.

- The frequency response of a system is defined as the **<u>steady-state</u> response** of the system to a **sinusoidal** input signal.
- When the system is in steady-state, it differs from the input signal only in **amplitude/gain** (A) (Norwegian: “forsterkning”) and **phase lag** (φ) (Norwegian: “faseforskyvning”).

Bode Diagram

Theory

1. Find Data

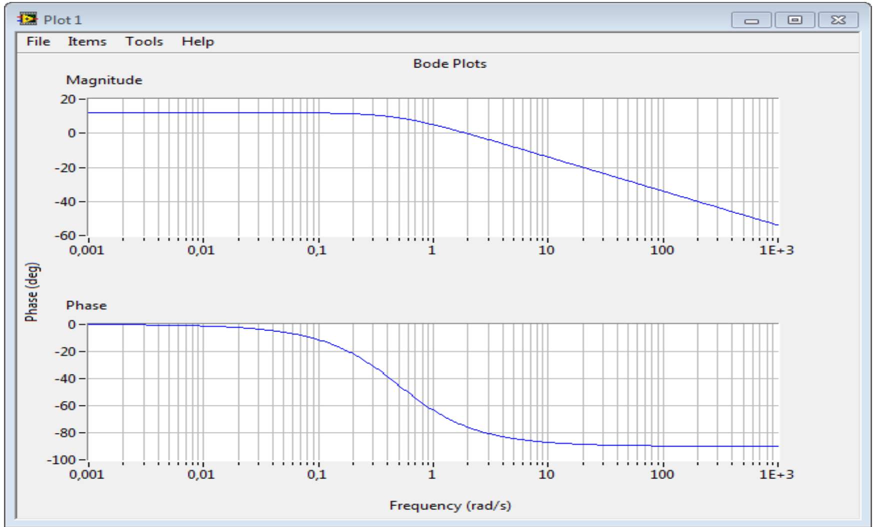


The same for frequency 3, 4, …, n

2. We find A and ϕ for each of the frequencies,

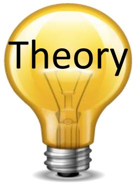
ω	$A(\omega)$	$\phi(\omega)$
0.1	11.9	-11.3
0.16	11.6	-17.7
0.25	11.1	-26.5
0.4	9.9	-38.7
0.625	7.8	-51.3
2.5	-2.1	-78.6

3. Based on that we can plot the Frequency Response in a so-called Bode Diagram:

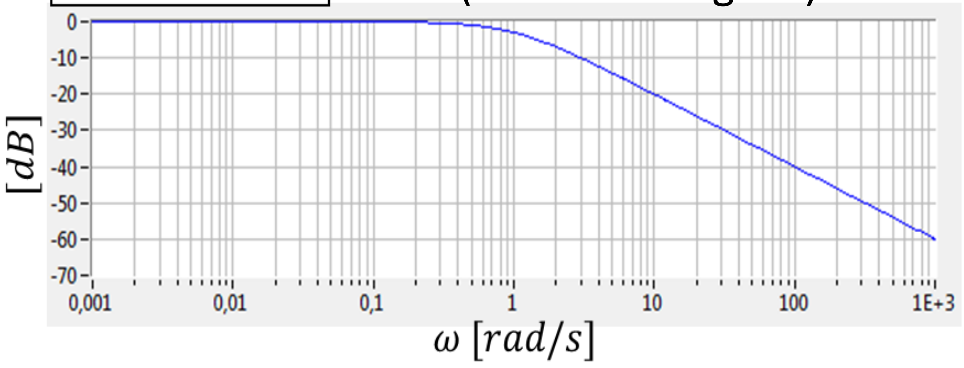


Bode Diagram

The x-scale is **logarithmic**



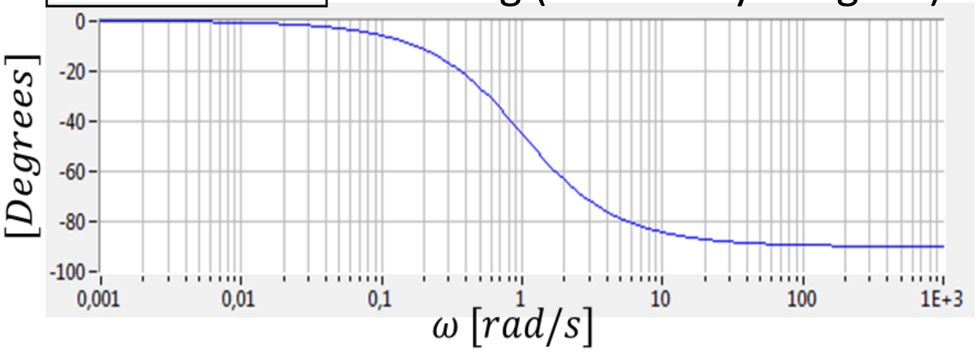
$A(\omega) = |H(j\omega)|$ Gain ("Forsterkningen")



The y-scale is in [dB]

$$x[dB] = 20log_{10}x$$

$\phi(\omega) = \angle H(j\omega)$ Phase lag ("Faseforkyvningen")



The y-scale is in [degrees]

$$2\pi\ rad = 360^o$$

$$d\ [degrees] = r[radians] \cdot \left(\frac{180}{\pi}\right)$$

$$r[radians] = d[degrees] \cdot \left(\frac{\pi}{180}\right)$$

Normally, the unit for frequency is Hertz [Hz], but in frequency response and Bode diagrams we use radians ω [rad/s]. The relationship between these are as follows:

$$\omega = 2\pi f$$

Frequency Response – MATLAB

Example

Transfer Function:

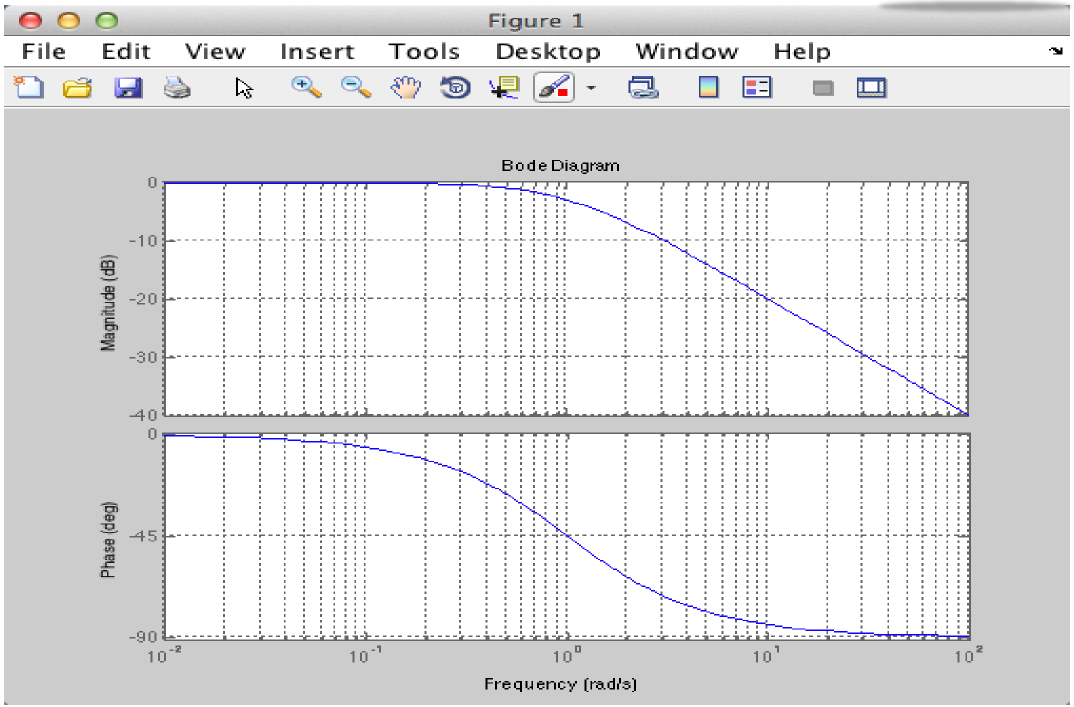
$$H(s) = \frac{y(s)}{u(s)} = \frac{1}{s+1}$$

MATLAB Code:

```
clear
clc
close all

% Define Transfer function
num=[1];
den=[1, 1];
H = tf(num, den)

% Frequency Response
bode(H);
grid on
```



The frequency response is an important tool for analysis and design of signal filters and for analysis and design of control systems.

Bode Diagram – MATLAB Example

MATLAB Code:

```
clear, clc

% Transfer function
num=[1];
den1=[1,0];
den2=[1,1]
den3=[1,1]
den = conv(den1,conv(den2,den3));
H = tf(num, den)

% Bode Diagram
bode(H)
subplot(2,1,1)
grid on
subplot(2,1,2)
grid on
```

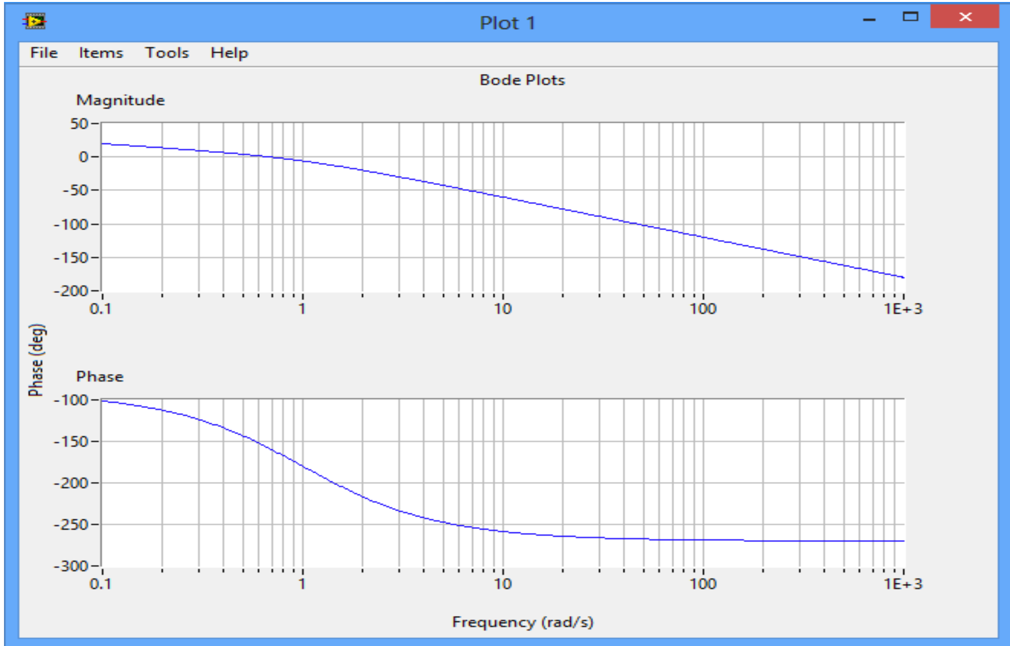
or:

```
clear, clc

% Transfer function
num=[1];
den=[1,2,1,0];
H = tf(num, den)

% Bode Diagram
bode(H)
subplot(2,1,1)
grid on
subplot(2,1,2)
grid on
```

$$H(s) = \frac{1}{s(s+1)^2} = \frac{1}{s^3 + 2s^2 + s}$$



Frequency Response Air Heater

- Typically, we need to Plot the Frequency Response for the Air Heater in a Bode plot. Use, e.g., the *bode()* function in MATLAB.
- Find, e.g., $A(\omega)$ and $\phi(\omega)$ for the frequencies given below using MATLAB. MATLAB has many built-in functions for dealing with Frequency Analysis

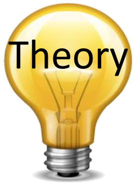
$\omega\ [rad/s]$	$A(\omega)$	$\phi(\omega)[degrees]$
0.001		
0.01		
0.1		
1		
3		
5		
10		

Frequency Response Air Heater

- Typically, we also need to find the mathematical expressions for $A(\omega)\ [dB]$ and $\phi(\omega)$. You typically use pen and paper for this.
- Find, e.g., $A(\omega)$ and $\phi(\omega)$ for the same frequencies above using the mathematical expressions for $A(\omega)$ and $\phi(\omega)$. Tip: Use a For Loop or/and define a vector, e.g., $w = [0.01, 0.1, \dots]$.
- It is recommended to use the *semilogx()* function in order to plot the Bode diagram based on these values.
- Typically, You need to compare and discuss the results.

$\omega\ [rad/s]$	$A(\omega)$	$\phi(\omega) [degrees]$
0.001		
0.01		
0.1		
1		
3		
5		
10		

Manually find the Frequency Response from the Transfer Function



For a transfer function:

$$H(S) = \frac{y(s)}{u(s)}$$

We have that:

$$H(j\omega) = |H(j\omega)|e^{j\angle H(j\omega)}$$

Where $H(j\omega)$ is the frequency response of the system, i.e., we may find the frequency response by setting $s = j\omega$ in the transfer function. Bode diagrams are useful in frequency response analysis.

The Bode diagram consists of 2 diagrams, the Bode magnitude diagram, $A(\omega)$ and the Bode phase diagram, $\phi(\omega)$.

The **Gain** function:

$$A(\omega) = |H(j\omega)|$$

The **Phase** function:

$$\phi(\omega) = \angle H(j\omega)$$

The $A(\omega)$-axis is in decibel (dB), where the decibel value of x is calculated as: $x[dB] = 20log_{10}x$

The $\phi(\omega)$-axis is in degrees (not radians!)

Manually find the Frequency Response from the Transfer Function

Given the following transfer function:

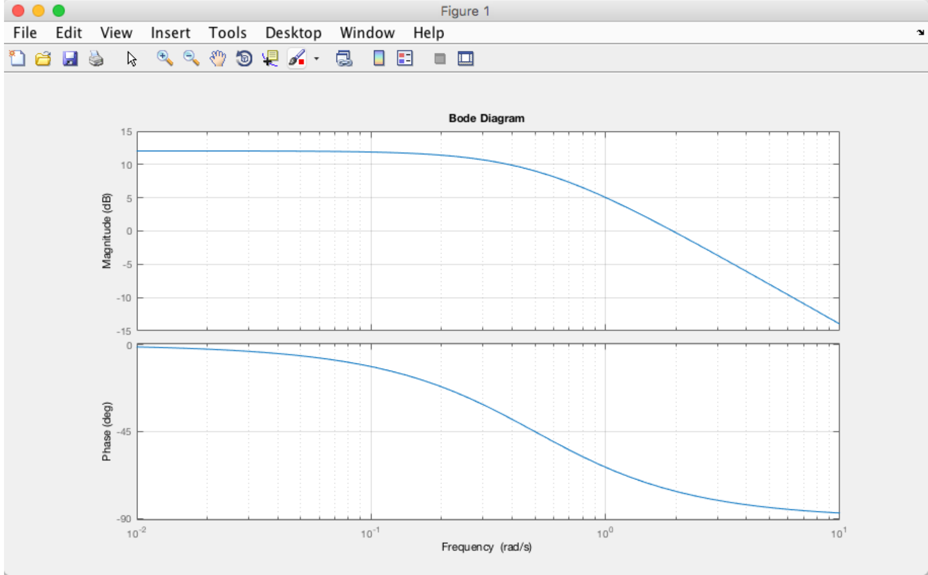
$$H(S) = \frac{4}{2s+1}$$

The mathematical expressions for $A(\omega)$ and $\phi(\omega)$ become:

$$|H(j\omega)|_{dB} = \underline{20log4 - 20log\sqrt{(2\omega)^2+1}}$$

$$\angle H(j\omega) = \underline{-\arctan(2\omega)}$$

Bode Plot:



MATLAB Code

Example

```
clear
clc

% Transfer function
num=[4];
den=[2, 1];
H = tf(num, den)

% Bode Plot
figure(1)
bode(H)
grid on

% Margins and Phases for given Frequencies

% Alt 1: Use bode function directly
disp('----- Alternative 1 -----')
w = [0.1, 0.16, 0.25, 0.4, 0.625, 2.5, 10];

[magw, phasew] = bode(H, w);

for i=1:length(w)
  mag(i) = magw(1,1,i);
  phase(i) = phasew(1,1,i);
end

magdB = 20*log10(mag); %convert to dB
mag_data = [w; magdB]
phase_data = [w; phase]
```

```
clear
clc

w = [0.1, 0.16, 0.25, 0.4, 0.625, 2.5, 10];

% Alt 2: Use Mathematical expressions for H and <H
disp('----- Alternative 2 -----')
gain = 20*log10(4) - 20*log10(sqrt((2*w).^2+1));
phase = -atan(2*w);
phasedeg = phase * 180/pi; %convert to degrees

mag_data2 = [w; gain]
phase_data2 = [w; phasedeg]

figure(2)
subplot(2,1,1)
semilogx(w,gain)
grid on

subplot(2,1,2)
semilogx(w,phasedeg)
grid on
```

Congratulations! - You are finished with the Task

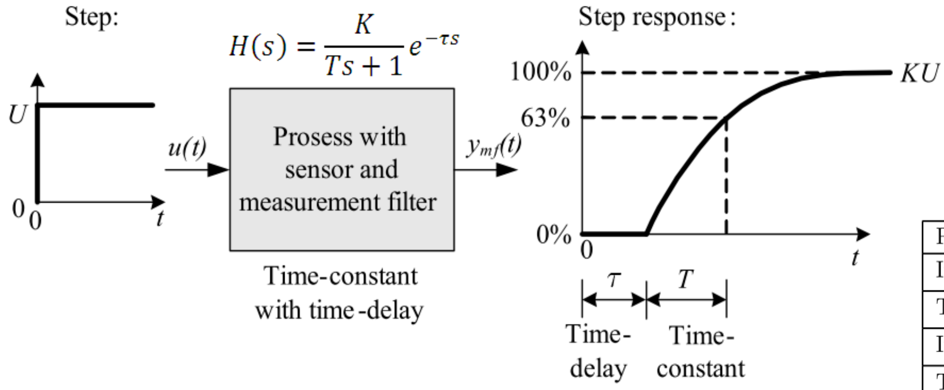
Stability Analysis using MATLAB

If you prefer, you can use Python

Hans-Petter Halvorsen

Skogestad's method

- Find Proper PI Parameters. Use, e.g., the Skogestad's method, which should be a starting point for further design and analysis in MATLAB
- The Skogestad's method assumes you apply a step on the input (u) and then observe the response and the output (y), as shown below.
- If we have a model of the system (which we have in our case), we can use the following Skogestad's formulas for finding the PI(D) parameters directly.



[Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

Tip! We can, e.g., set $T_C = 10\ s$ and $c = 1.5$ (or try with other values if you get poor PI parameters).

Process type	$H_{psf}(s)$ (process)	K_p	T_i	T_d
Integrator + delay	$\frac{K}{s}e^{-\tau s}$	$\frac{1}{K(T_C+\tau)}$	$c\,(T_C+\tau)$	0
Time-constant + delay	$\frac{K}{Ts+1}e^{-\tau s}$	$\frac{T}{K(T_C+\tau)}$	$\min\,[T,\ c\,(T_C+\tau)]$	0
Integr + time-const + del.	$\frac{K}{(Ts+1)s}e^{-\tau s}$	$\frac{1}{K(T_C+\tau)}$	$c\,(T_C+\tau)$	T
Two time-const + delay	$\frac{K}{(T_1s+1)(T_2s+1)}e^{-\tau s}$	$\frac{T_1}{K(T_C+\tau)}$	$\min\,[T_1,\ c\,(T_C+\tau)]$	T_2
Double integrator + delay	$\frac{K}{s^2}e^{-\tau s}$	$\frac{1}{4K(T_C+\tau)^2}$	$4\,(T_C+\tau)$	$4\,(T_C+\tau)$

Stability Analysis

How do we figure out that the Feedback System is stable before we test it on the real System? We have 3 different methods:

1. Poles
2. Frequency Response/Bode
3. Simulations (Step Response)

We will do all these things using MATLAB

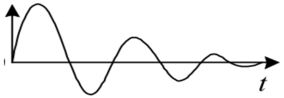
Stability Analysis

③ Time domain

$T(s)$

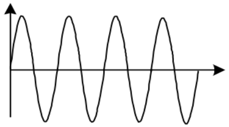
Tracking transfer function

Asymptotically stable system



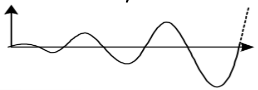
$$\lim_{t\to\infty} y(t) = k$$

Marginally stable system



$$0 < \lim_{t\to\infty} y(t) < \infty$$

Unstable system

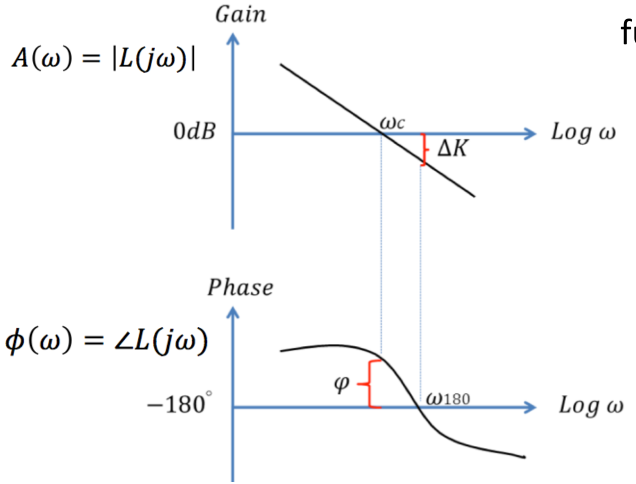


$$\lim_{t\to\infty} y(t) = \infty$$

② Frequency domain

$L(s)$

Loop transfer function



$\omega_c < \omega_{180}$ Asymptotically stable system

$\omega_c = \omega_{180}$ Marginally stable system

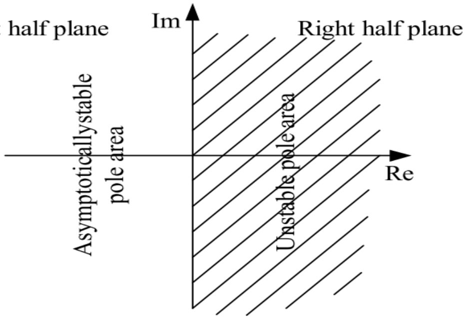
$\omega_c > \omega_{180}$ Unstable system

① The Complex domain

Poles

$T(s)$

Tracking transfer function

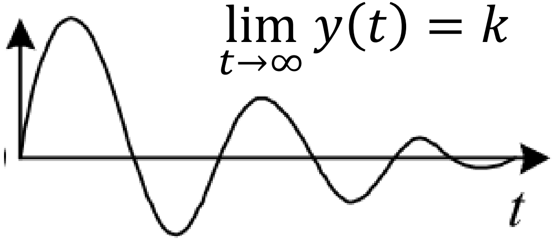


Theory

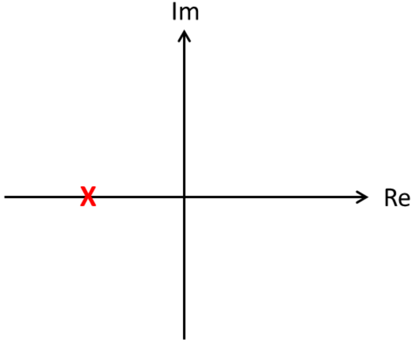
Stability Analysis

1 Asymptotically stable system:

$$\lim_{t\to\infty} y(t) = k$$

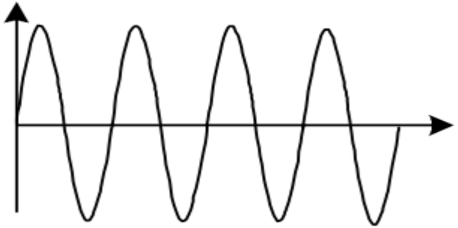


[Figures above: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

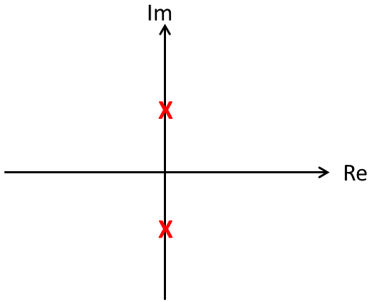


Each of the poles of the transfer function lies strictly in the left half plane (has strictly negative real part).

2 Marginally stable system:



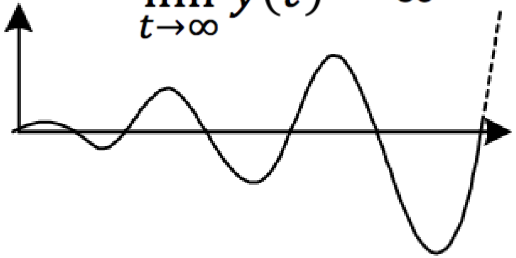
$$0 < \lim_{t\to\infty} y(t) < \infty$$

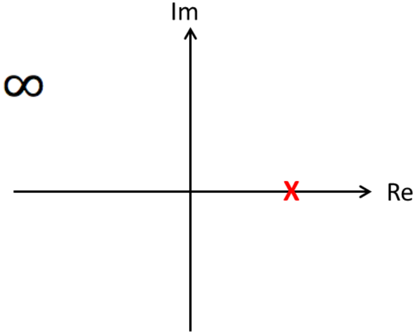


One or more poles lies on the imaginary axis (have real part equal to zero), and all these poles are distinct. Besides, no poles lie in the right half plane.

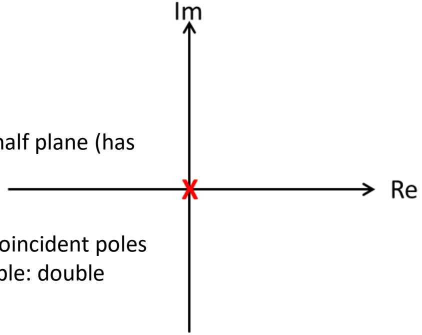
3 Unstable system:

$$\lim_{t\to\infty} y(t) = \infty$$





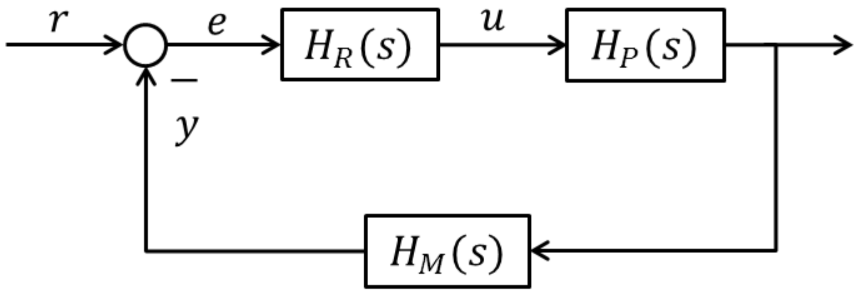
At least one pole lies in the right half plane (has real part greater than zero).



Or: There are multiple and coincident poles on the imaginary axis. Example: double integrator $H(s) = \frac{1}{s^2}$

Stability Analysis of Feedback Systems

Theory



1. **Loop Transfer Function** ("Sløyfetransferfunksjonen"):

$$L(s) = H_R H_P H_M$$

```
Hr = ...
Hp = ...
Hm = ...
L = series(series(Hr, Hp), Hm)
```

Used in Frequency Response Stability Analysis (Bode Diagram)

2. **The Tracking Function** ("Følgeforholdet"):

$$T(s) = \frac{y(s)}{r(s)} = \frac{H_R H_P H_M}{1 + H_R H_P H_M} = \frac{L(s)}{1 + L(s)}$$

```
L = ...
T = feedback(L, 1)
```

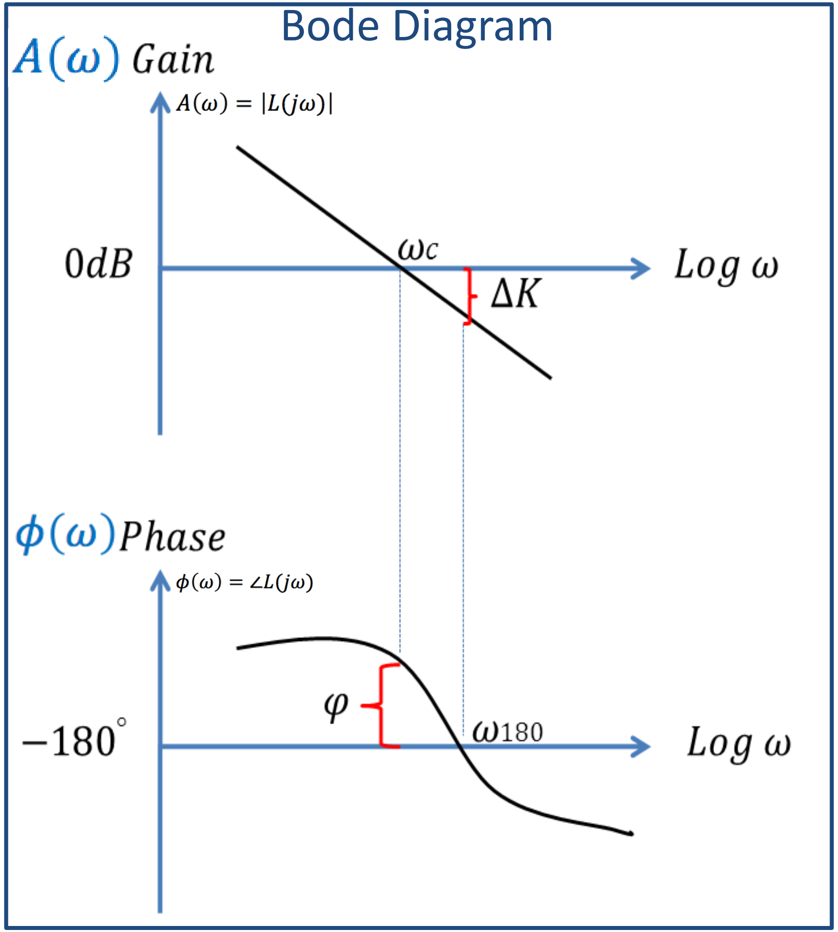
3. **The Sensitivity Function** ("Sensitivitetsfunksjonen"):

$$S(s) = \frac{e(s)}{r(s)} = \frac{1}{1 + L(s)} = 1 - T(s)$$

```
T = ...
S = 1-T
```

Frequency Response and Stability Analysis

Theory



ω_c and ω_{180} are called the crossover-frequencies (Norwegian: “kryssfrekvens”)

$A(\omega) = |L(j\omega)|$

ΔK is the gain margin (GM) of the system (Norwegian: “Forsterkningsmargin”).
How much the loop gain can increase before the system becomes unstable

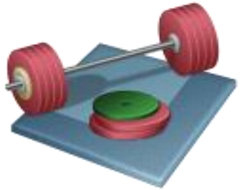
$\phi(\omega) = \angle L(j\omega)$

ϕ is the phase margin (PM) of the system (Norwegian: “Fasemargin”).
How much phase shift the system can tolerate before it becomes unstable.

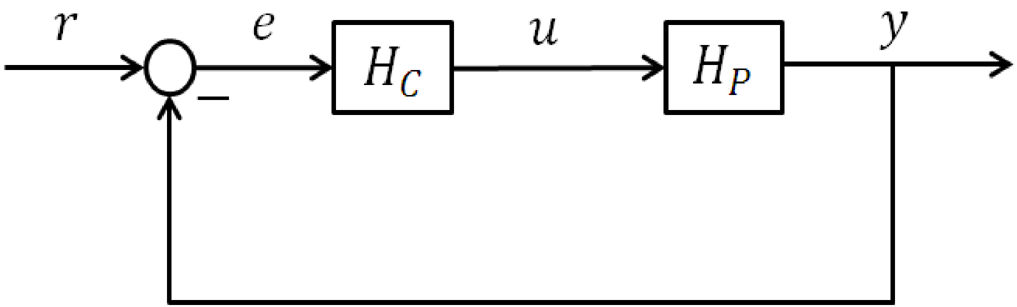
We have the following:

$\omega_c < \omega_{180}$	Asymptotically stable system
$\omega_c = \omega_{180}$	Marginally stable system
$\omega_c > \omega_{180}$	Unstable system

Analysis of the Air Heater Feedback System



Below we see the block diagram of the feedback system:



Process (Air Heater):

The transfer function for the process is as follows:

$$H_p(s) = \frac{T(s)}{u(s)} = \frac{K}{Ts+1} e^{-\tau s}$$

Use values for K_h, θ_d, θ_t from a previous task.

PI controller:

The PI controller is defined as:

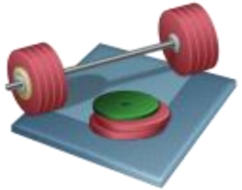
$$u(t) = K_p e + \frac{K_p}{T_i} \int_0^t e d\tau$$

We need to find the transfer function for the PI Controller:

$$H_c(s) = \frac{u(s)}{e(s)}$$

Tip! Use Laplace on the equation to the left.
You should also plot the Frequency Response for the PI controller ($H_c(s)$) in a Bode plot.
Use values for K_p and T_i found previously.

Analysis of the Air Heater Feedback System



Loop transfer function: $\boldsymbol{L(s)}$

We need to find the Loop transfer function $L(s)$ using MATLAB.

The Loop transfer function is defined as:

$$L(s) = H_c H_p$$

Tip! Use the built-in function *series()* in MATLAB.

Tracking transfer function: $\boldsymbol{T(s)}$

We need to find the Tracking transfer function $T(s)$ using MATLAB.

The Tracking transfer function is defined as:

$$T(s) = \frac{y(s)}{r(s)} = \frac{L(s)}{1+L(s)}$$

Tip! Use the built-in function *feedback()* in MATLAB.

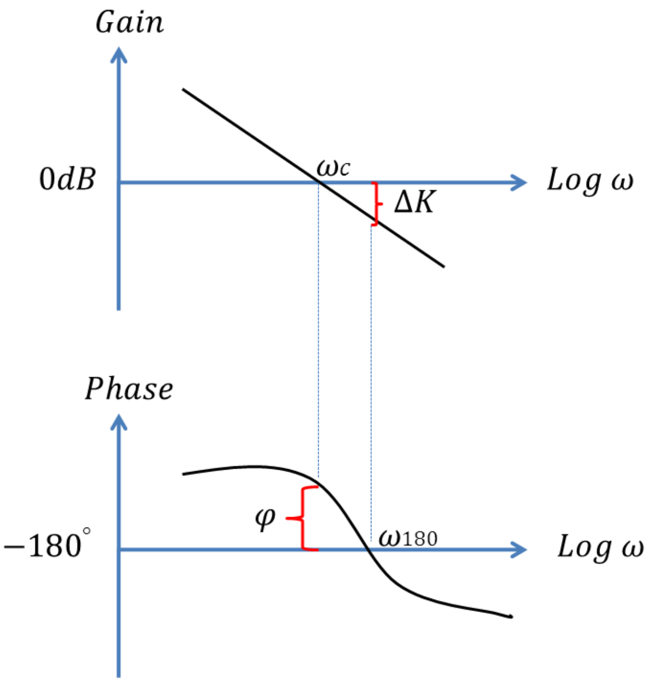
Sensitivity transfer function: $\boldsymbol{S(s)}$

We need to find the Sensitivity transfer function $S(s)$ using MATLAB.

The Sensitivity transfer function is defined as:

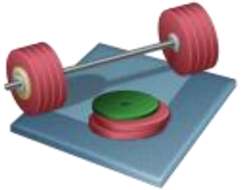
$$S(s) = \frac{e(s)}{r(s)} = \frac{1}{1+L(s)} = 1 - T(s)$$

Stability Analysis

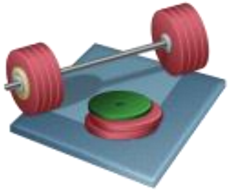


- Plot the Bode plot for the system using e.g., the bode() function in MATLAB
- Find the crossover-frequencies (ω_{180}, ω_c) and stability margins GM ($A(\omega)$), PM ($\phi(\omega)$) of the system ($L(s)$) from the Bode plot.
- Plot also Bode diagram where the crossover-frequencies, GM and PM are illustrated. Tip! Use the *margin()* function in MATLAB.
- Use also the *margin()* function in order to find values for $\omega_{180}, \omega_c, A(\omega), \phi(\omega)$ directly.
- You should compare and discuss the results.
- How much can you increase K_p before the system becomes unstable?

Stable vs. Unstable System



- You should find and use different values of K_p where you get a *marginally stable system*, an *asymptotically stable system* and an *unstable system*.
- Plot the time response for the *tracking function* using, e.g., use the step() function in MATLAB for all these 3 categories. How can we use the step response to determine the stability of the system?
- Find $\omega_{180}, \omega_c, A(\omega)$ and $\phi(\omega)$ in all 3 cases. How can we use ω_c and ω_{180} to determine the stability of the system?
- Find and plot the *poles* and *zeros* for the system for all the 3 categories mentioned above. How can we use the poles to determine the stability of the system?
- Plot the *Loop transfer function* $L(s)$, the *Tracking transfer function* $T(s)$ and the *Sensitivity transfer function* $S(s)$ in a Bode diagram for the system for all the 3 categories mentioned above.
- Discuss the results.

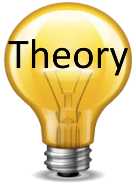


PI(D) Controller Design

1. Find Proper PI Parameters using the Ziegler–Nichols Frequency Response method
2. Compare and discuss the results compared to Skogestad's method used earlier

See next slides for details...

Ziegler–Nichols Frequency Response method



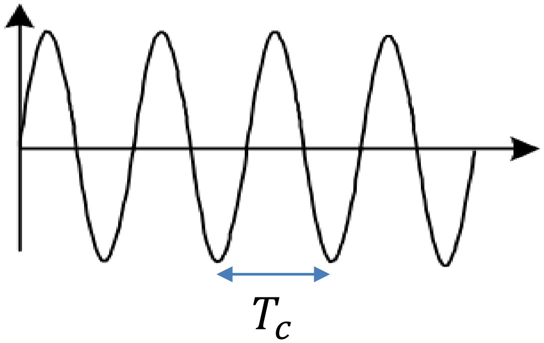
Assume you use a P controller only ($T_i = \infty, T_d = 0$). Then you need to find for which K_p the closed loop system is a marginally stable system ($\omega_c = \omega_{180}$). This K_p is called K_c (critical gain). The T_c (critical period) can be found from the damped oscillations of the closed loop system. Then calculate the PI(D) parameters using the formulas below.

Controller	K_p	T_i	T_d
P	$0.5K_c$	∞	0
PI	$0.45K_c$	$\frac{T_c}{1.2}$	0
PID	$0.6K_c$	$\frac{T_c}{2}$	$\frac{T_c}{8}$

K_c- Critical Gain

T_c- Critical Period

Marginally stable system:



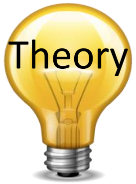
$$\omega_c = \omega_{180}$$

$$0 < \lim_{t\to\infty} y(t) < \infty$$

$$T_c = \frac{2\pi}{\omega_{180}}$$

https://en.wikipedia.org/wiki/Ziegler–Nichols_method

“Golden rules” of Stability Margins for a Control System

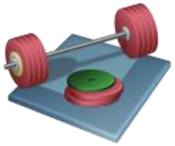


Gain Margin (GM): (Norwegian: “Forsterkningsmargin”)

$$2\ (6dB) < \Delta K < 4\ (12dB)$$

Phase Margin (PM): (Norwegian: “Fasemargin”)

$$30^\circ < \varphi < 60^\circ$$



What is the Stability Margins for the different PID tuning methods you are using?

Congratulations! - You are finished with the Task

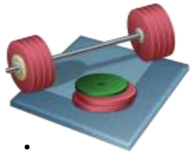
Control System in LabVIEW

If you prefer, you can use Python

This part is known from previous courses, feel free to reuse previous code and examples

Hans-Petter Halvorsen

Control System Implementation

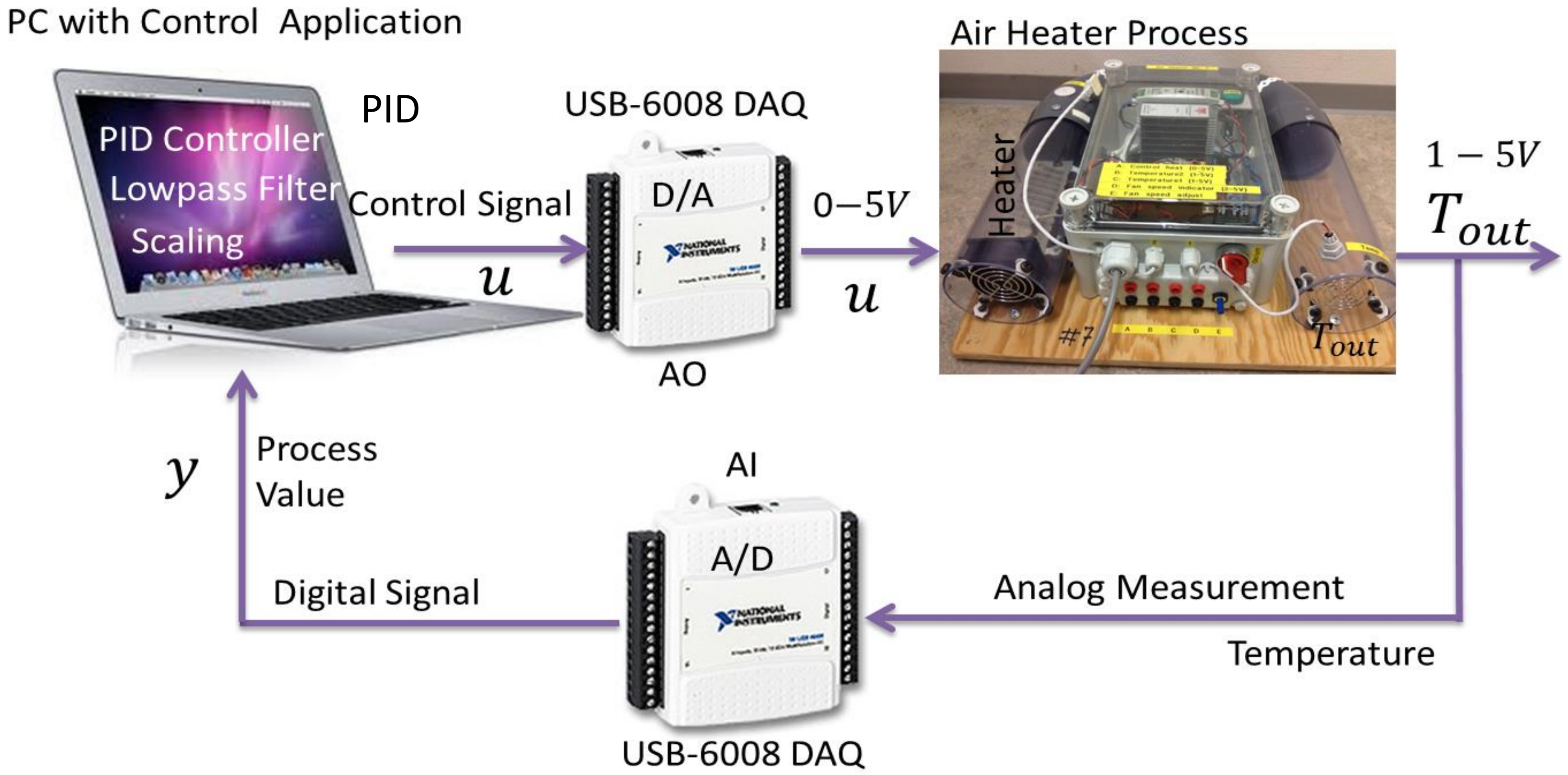


We need to implement a temperature control system of the Air Heater in LabVIEW using a PI controller and a Low-pass filter. Test the system with the PI parameters found in a previous tasks. Tune the parameters if necessary.

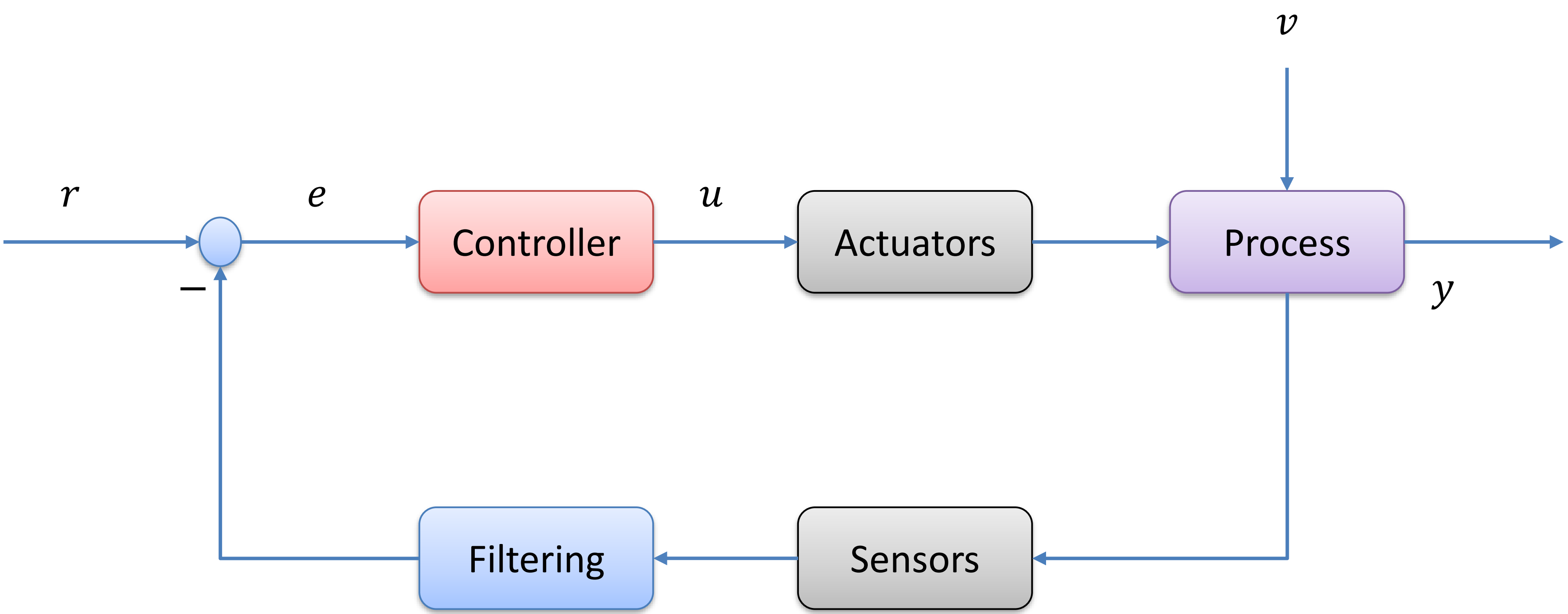
The implementation should be according to the following **specifications**:

- A **PI controller**, implemented from scratch with C-code in **Formula Node** in LabVIEW
- The **Control signal** (the controller output) shall be represented in unit of voltage ($0 - 5V$).
- The **Measurement signal**, being connected to the controller, shall be represented in unit of degree Celsius ($20 - 50℃$).
- The temperature **set-point** shall be in degree Celsius ($20 - 50℃$)..
- The **time-step** (sampling time, T_s) of the system can be set to, e.g., **0.1 sec**.
- **Plot** the control signal, measurement signal and the set-point.
- Use Your (1) Mathematical Model implemented in LabVIEW, (2) the “Black Box Simulator” provided and (3) the Real Process located in the laboratory.
- Test, document and discuss the performance of the control system (both for changes in the set point and for disturbances

Control System Overview



Control System Overview



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

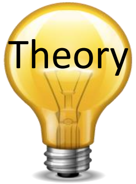
Tuning Parameters:

K_p Proportional Gain

T_i Integral Time $[sec.]$

T_d Derivative Time $[sec.]$

Industrial Control Systems (ICS)



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

cRIO

LabVIEW

Programmable Automation Controller (**PAC**) 4

I/O Module

1

PC based Control System/**SCADA** System (Supervisory Control And Data Acquisition) 5

Industrial **PID** Controller

2 Distributed Control Systems (**DCS**)

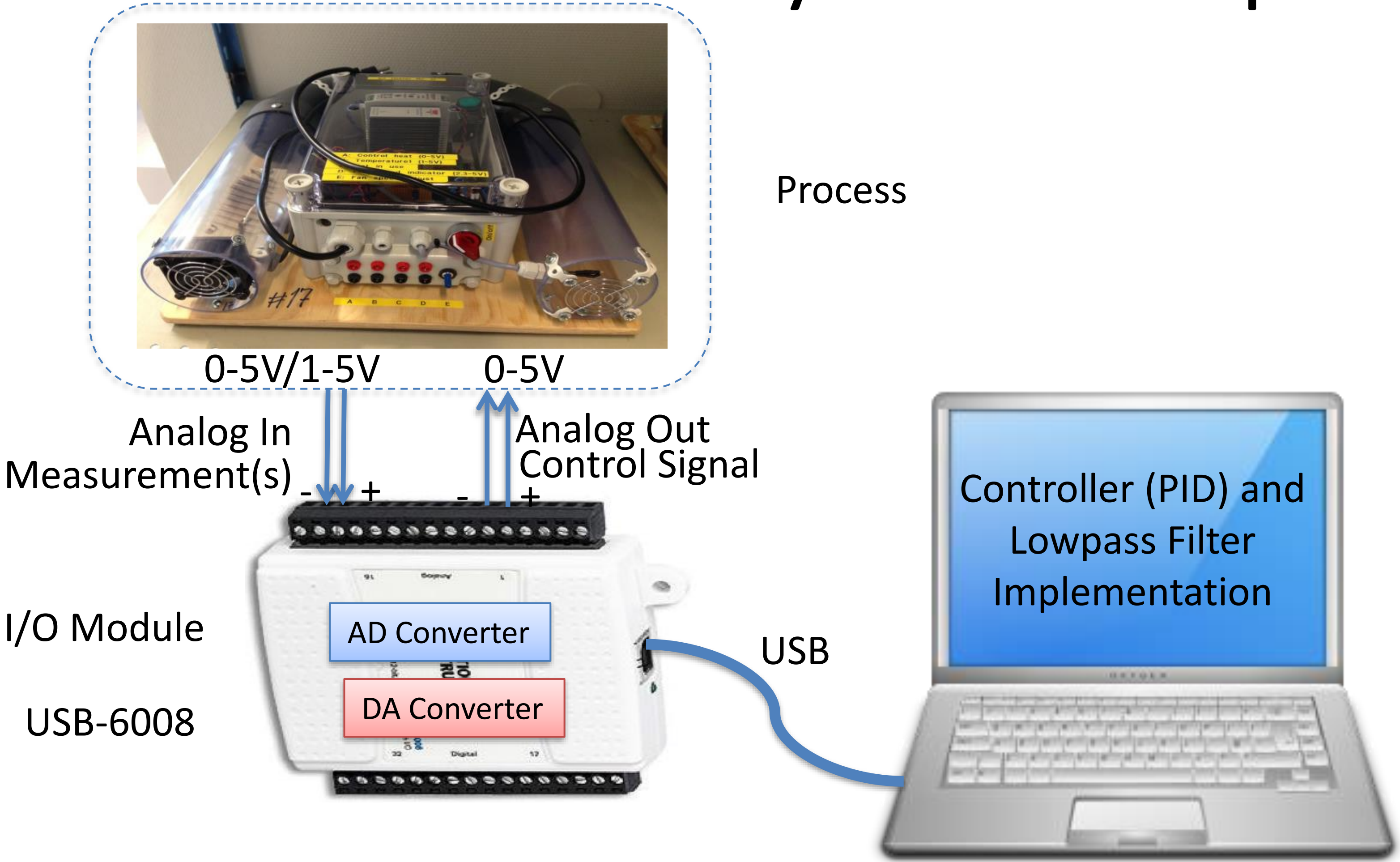
DeltaV

PLC (Programmable Logic Controller) 3

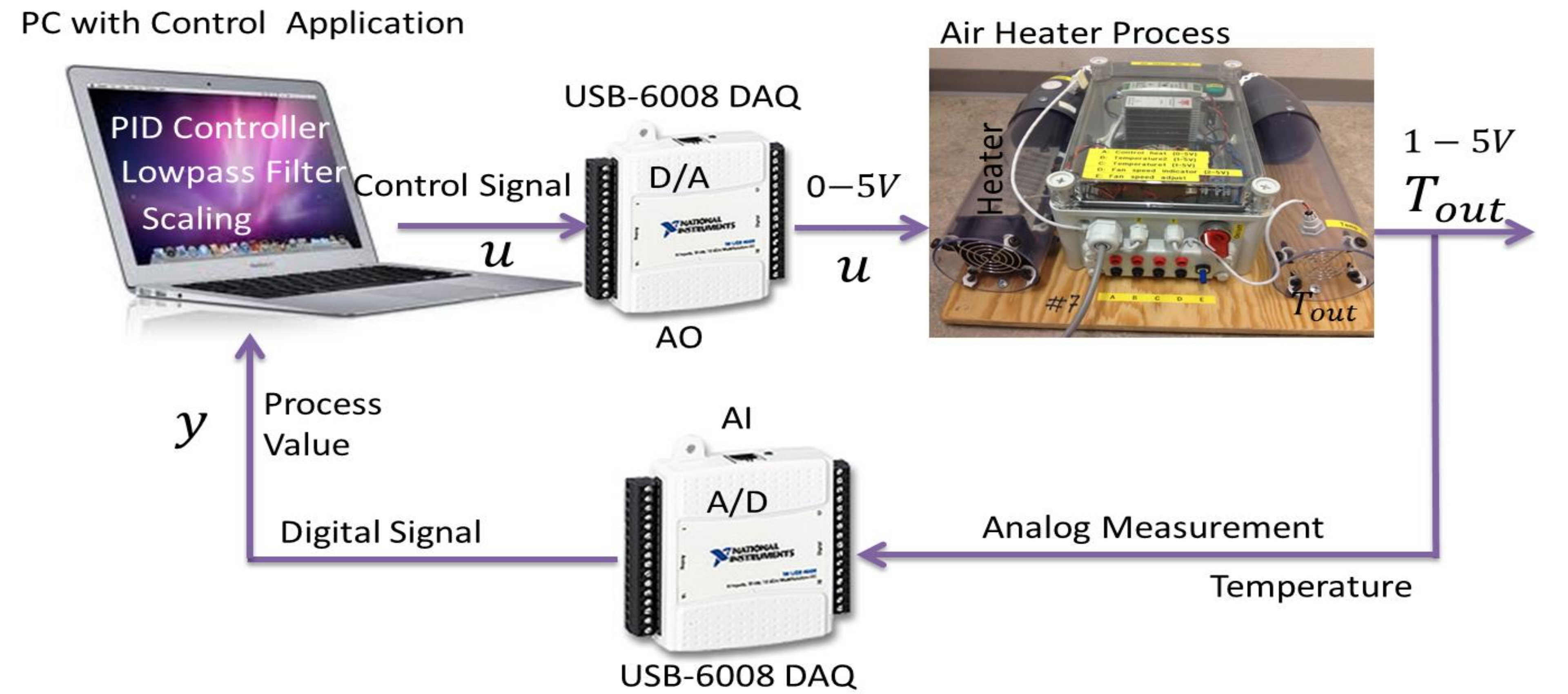
Siemens PLC

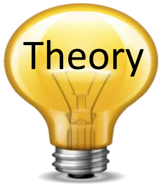
PC-based Control System Example

Theory



PC-based Control System



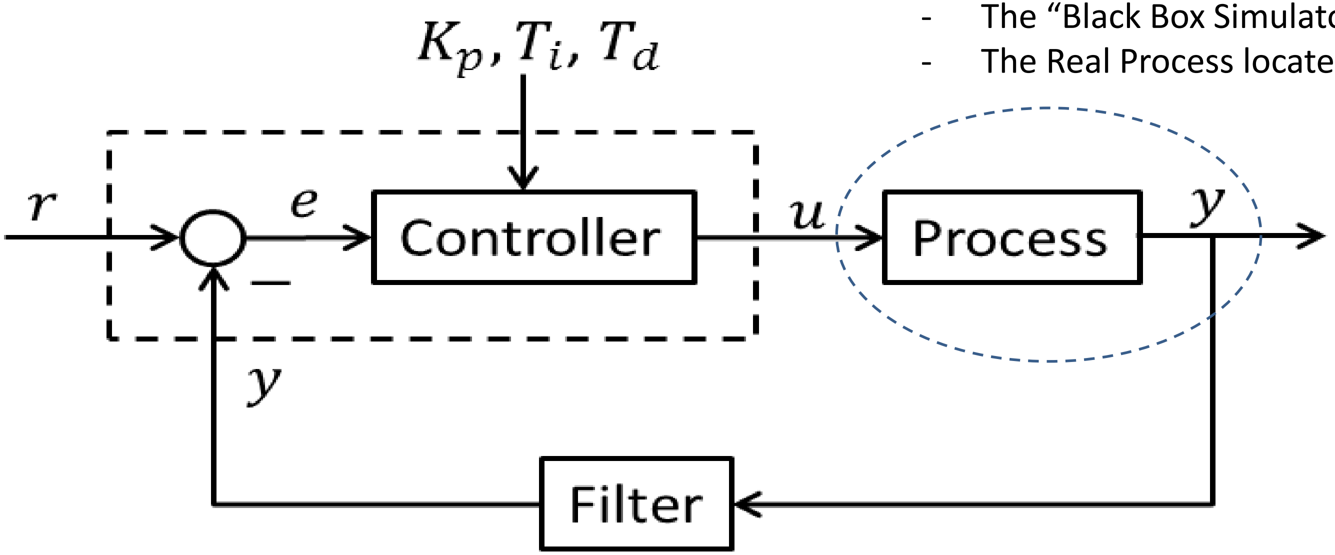


Control System Example

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

We have 3 different options:

- Your Mathematical Model implemented in LabVIEW
- The “Black Box Simulator” provided
- The Real Process located in the laboratory



Congratulations! - You are finished with the Task

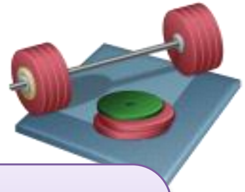
OPC UA

The Next Generation OPC used in Industry 4.0 Applications

This part is known from previous courses, feel free to reuse previous code and examples

Hans-Petter Halvorsen

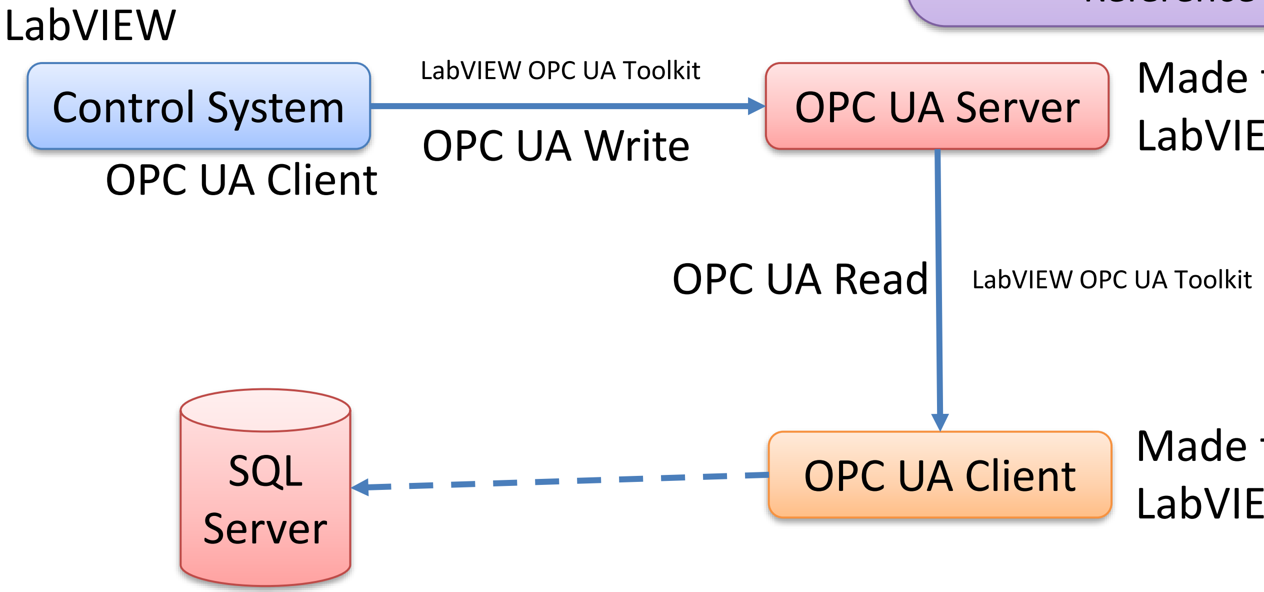
OPC Implementation Scenario



OPC UA - The Industry 4.0 Implementation of OPC

Example (feel free to solve it a different way):

It is enough sending the Temperature Data (T_{out}) using OPC UA
- No need to send Control Signal, Reference Signal, etc.

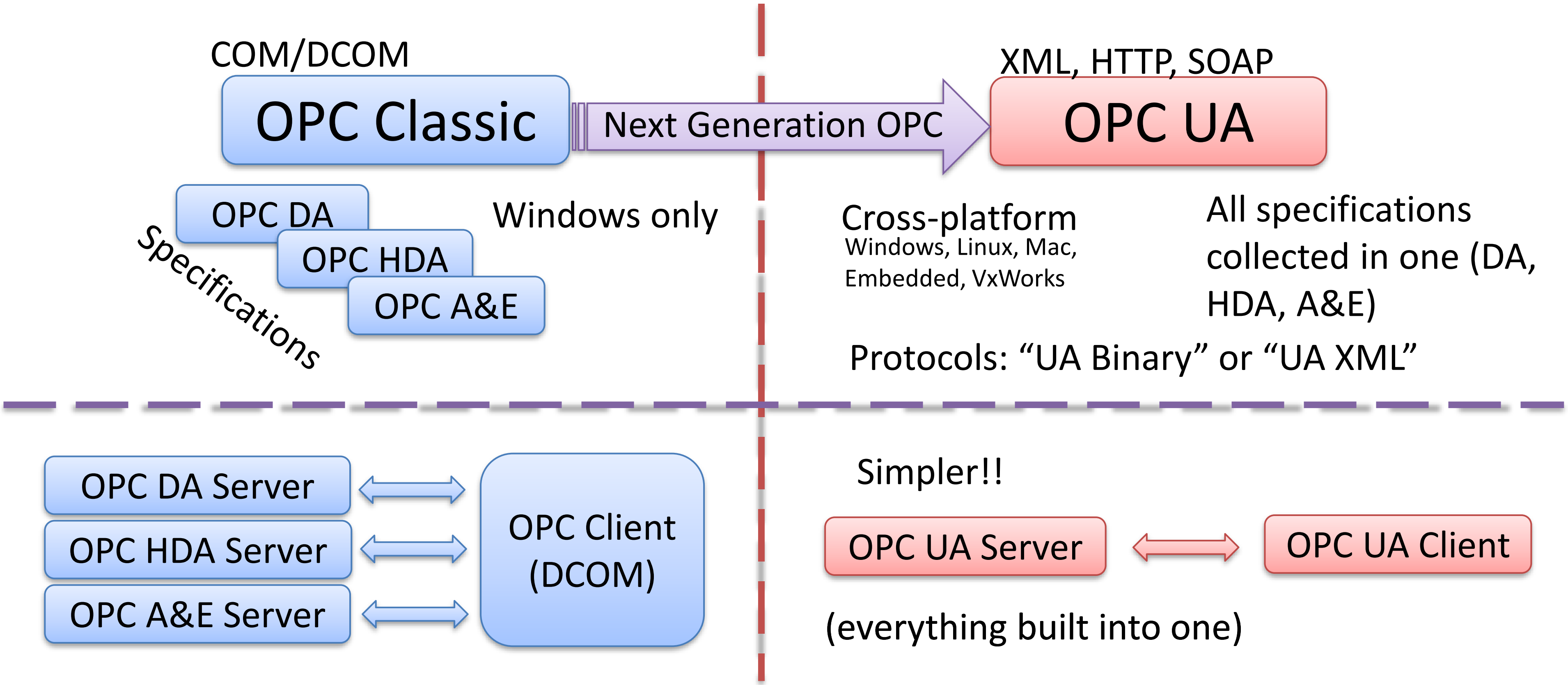


Made from scratch with LabVIEW OPC UA Toolkit

Made from scratch with LabVIEW OPC UA Toolkit

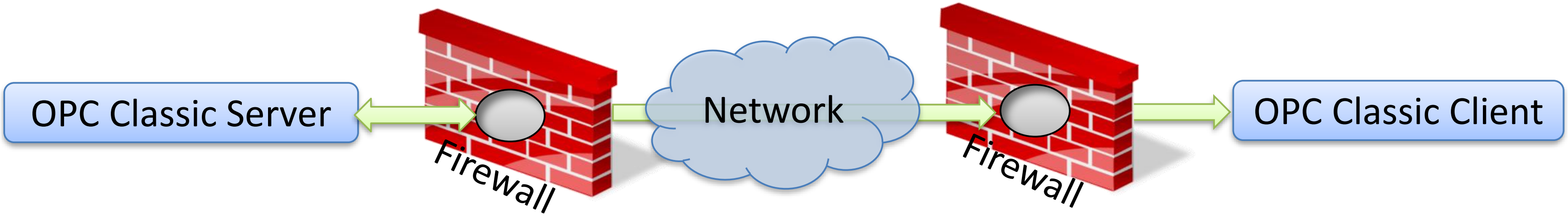
Make sure to Add Value to your Solution

Next Generation OPC

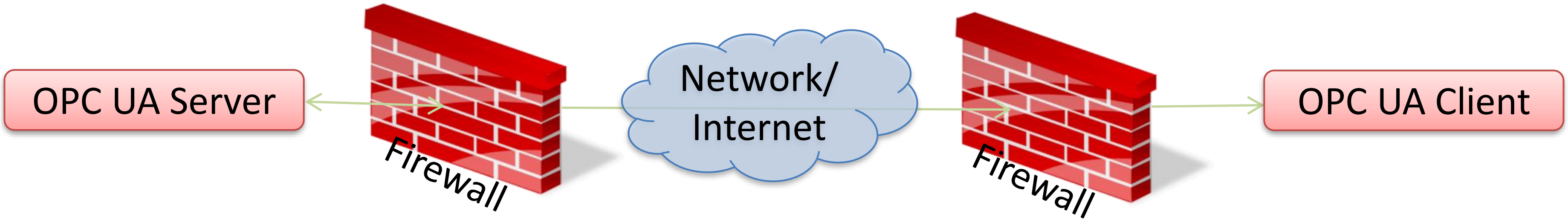


Theory

Next Generation OPC

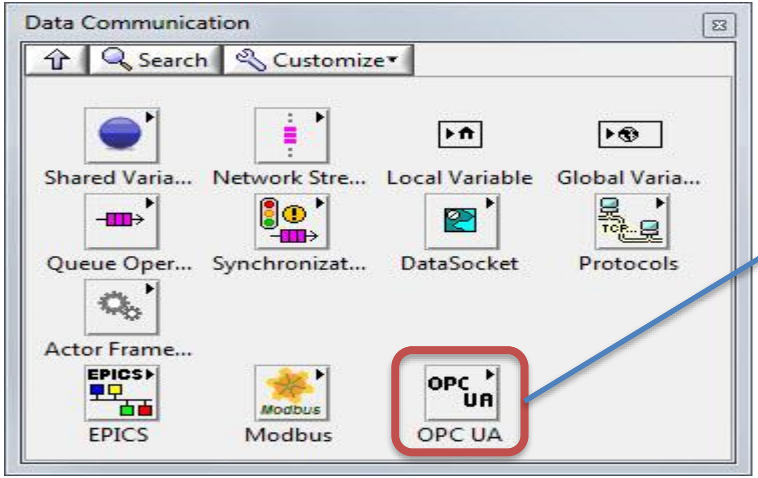


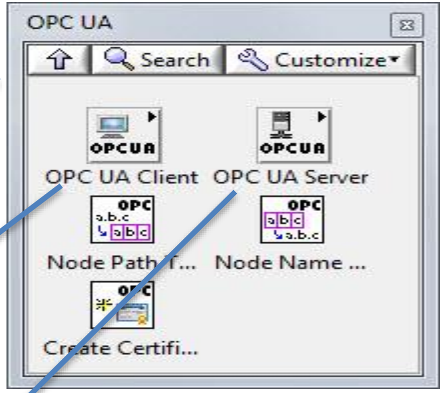
To open DCOM through firewalls demanded a large hole in the firewall!
Impossible to route over Internet!

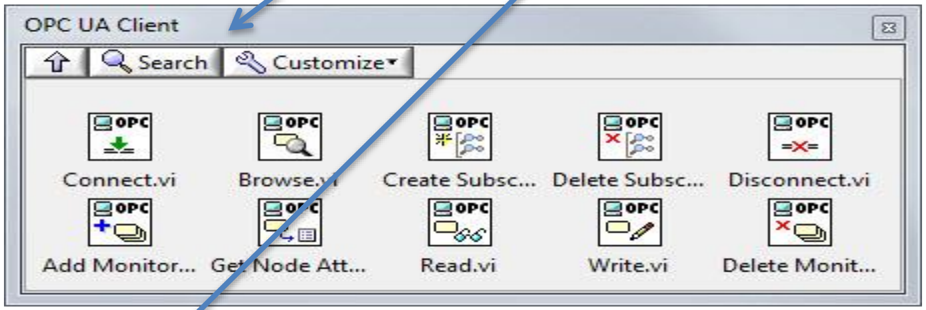


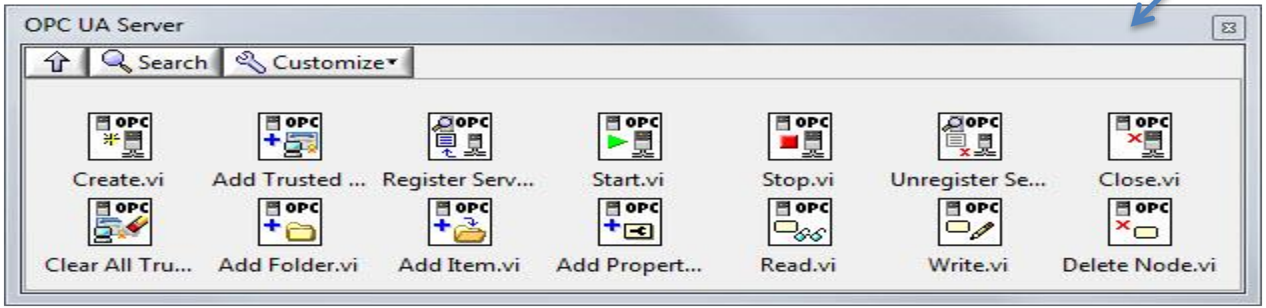
No hole in firewall (UA XML) or just a simple needlestick (UA Binary) is necessary
Easy to route over Internet!

OPC UA in LabVIEW







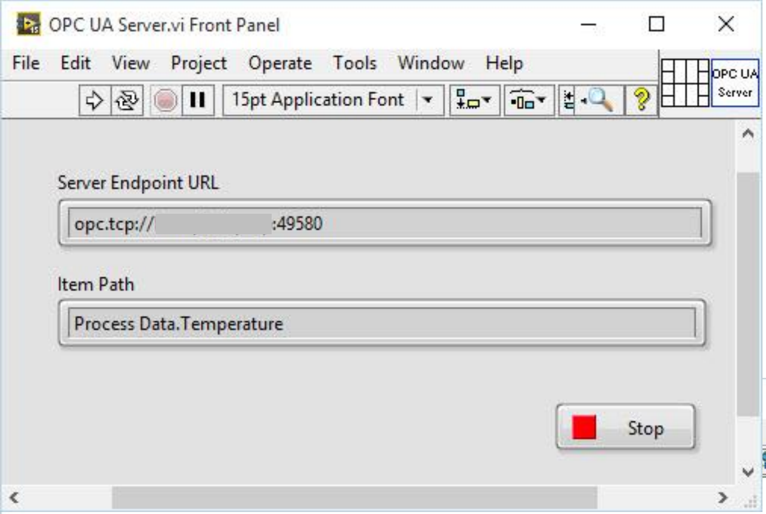


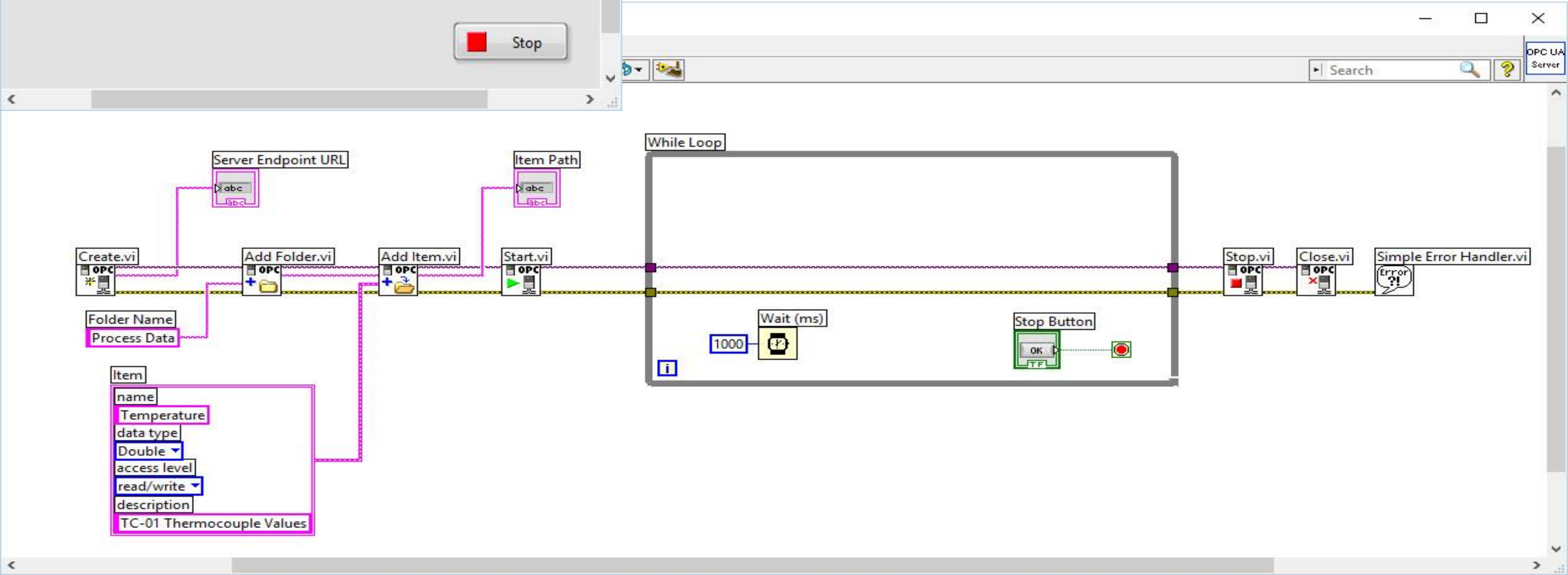
Note! You need to install the **LabVIEW OPC UA Toolkit**

Write/Read vs. Multiple Write/Read

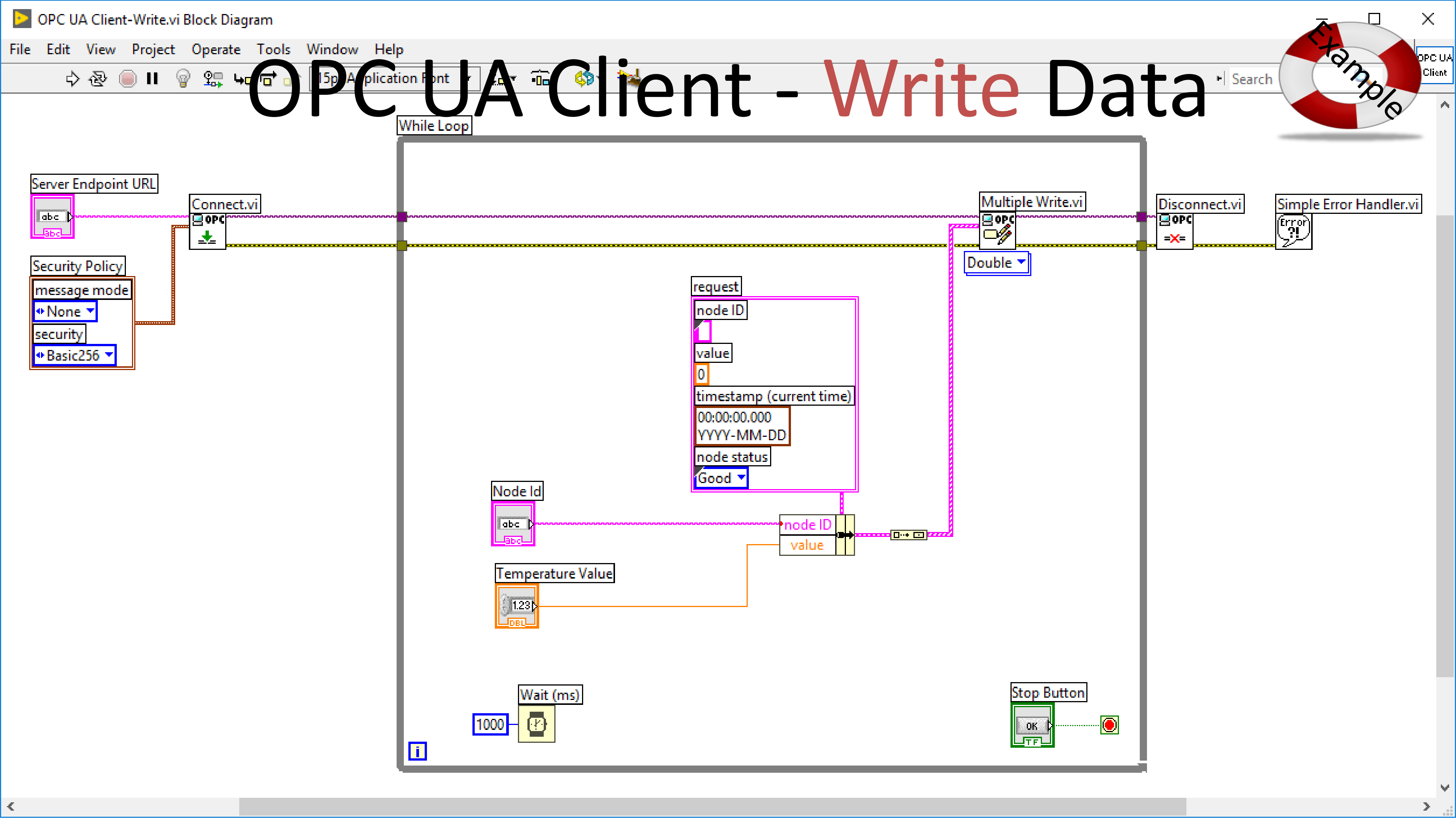
- You need to use the **OPC UA Toolkit** with LabVIEW 2017 or newer
- **Note!** When creating OPC Clients: The VIs **Write.vi** and **Read.vi** that was previously used in LabVIEW 2016 has been replaced with **Multiple Write.vi** and **Multiple Read.vi**.
- This means: It is recommended to use **Multiple Write.vi** and **Multiple Read.vi** instead of Write.vi and Read.vi for new applications.
- But if you open previous code (LabVIEW 2016 or earlier) in LabVIEW 2017 or newer, it will still work, because the old Write.vi and Read.vi are still included, but hidden in the palette in LabVIEW.
- You will find them here: C:\Program Files\National Instruments\LabVIEW 201x\vi.lib\OPCUA\client\internal\

OPC UA **Server** Example in LabVIEW

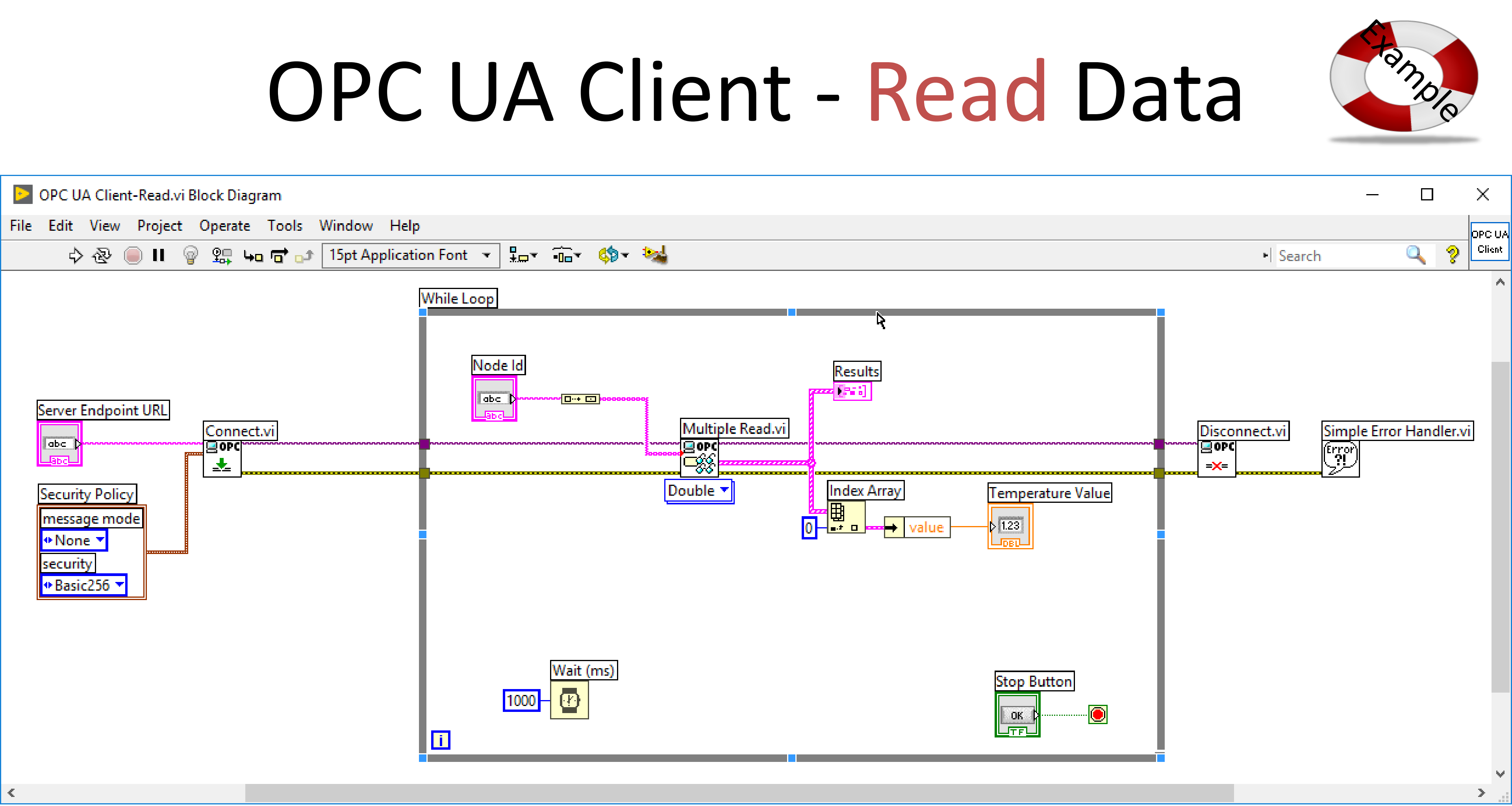




OPC UA Client - Write Data



OPC UA Client - Read Data



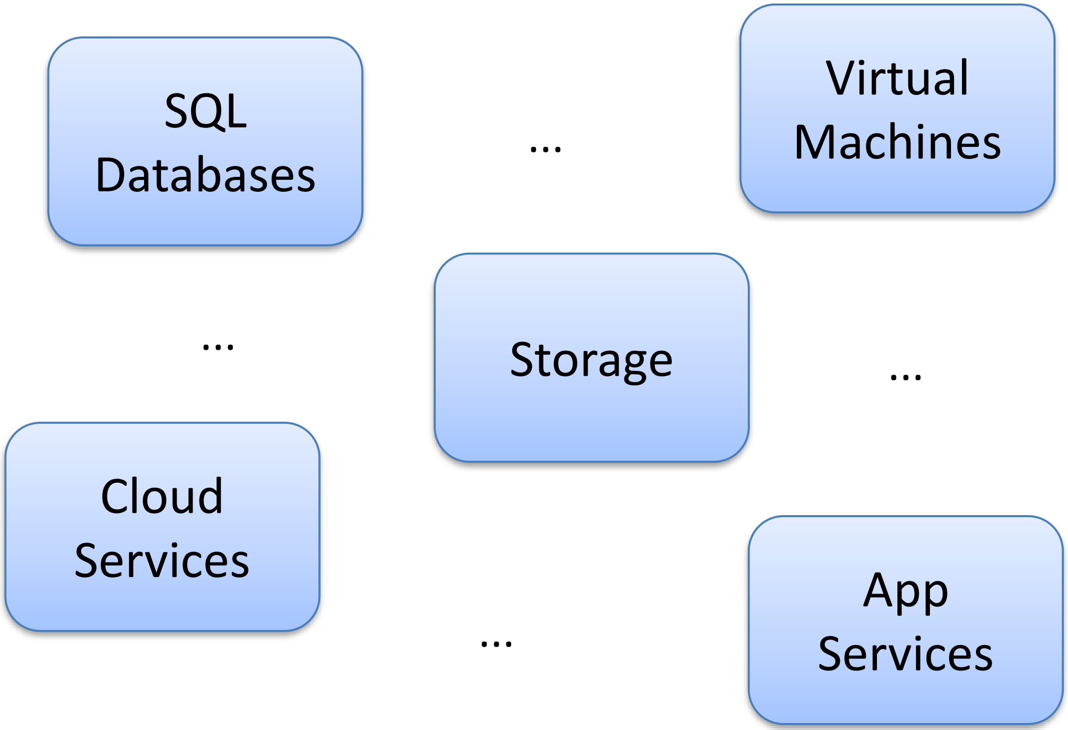
Congratulations! - You are finished with the Task

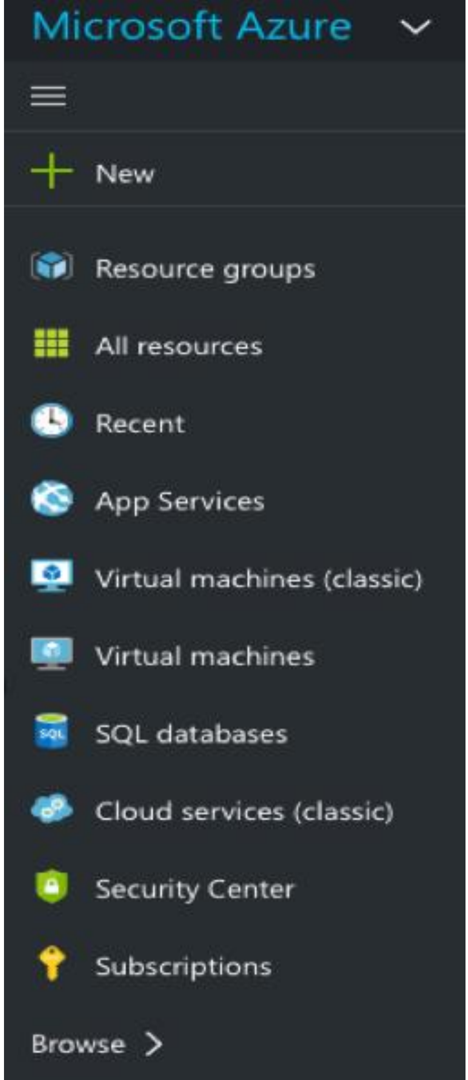
Microsoft Azure

Hans-Petter Halvorsen

Microsoft Azure

"Windows running in the Cloud"





Cloud Hosting

(Cloud Deployment of the Server-side parts of your system)

You can rent Cloud based services like Virtual Machines (Computers with OS running in the Cloud), Web Servers, Database Systems based on Monthly Fees

SQL Server

This part is known from previous courses, feel free to reuse previous code and examples. The only new here is that the SQL Server should be located in the Cloud (Azure) instead of having it on your local computer. It means this is a more realistic real-life scenario. Try also to improve your Database Model.

Hans-Petter Halvorsen

Theory

SQL Server

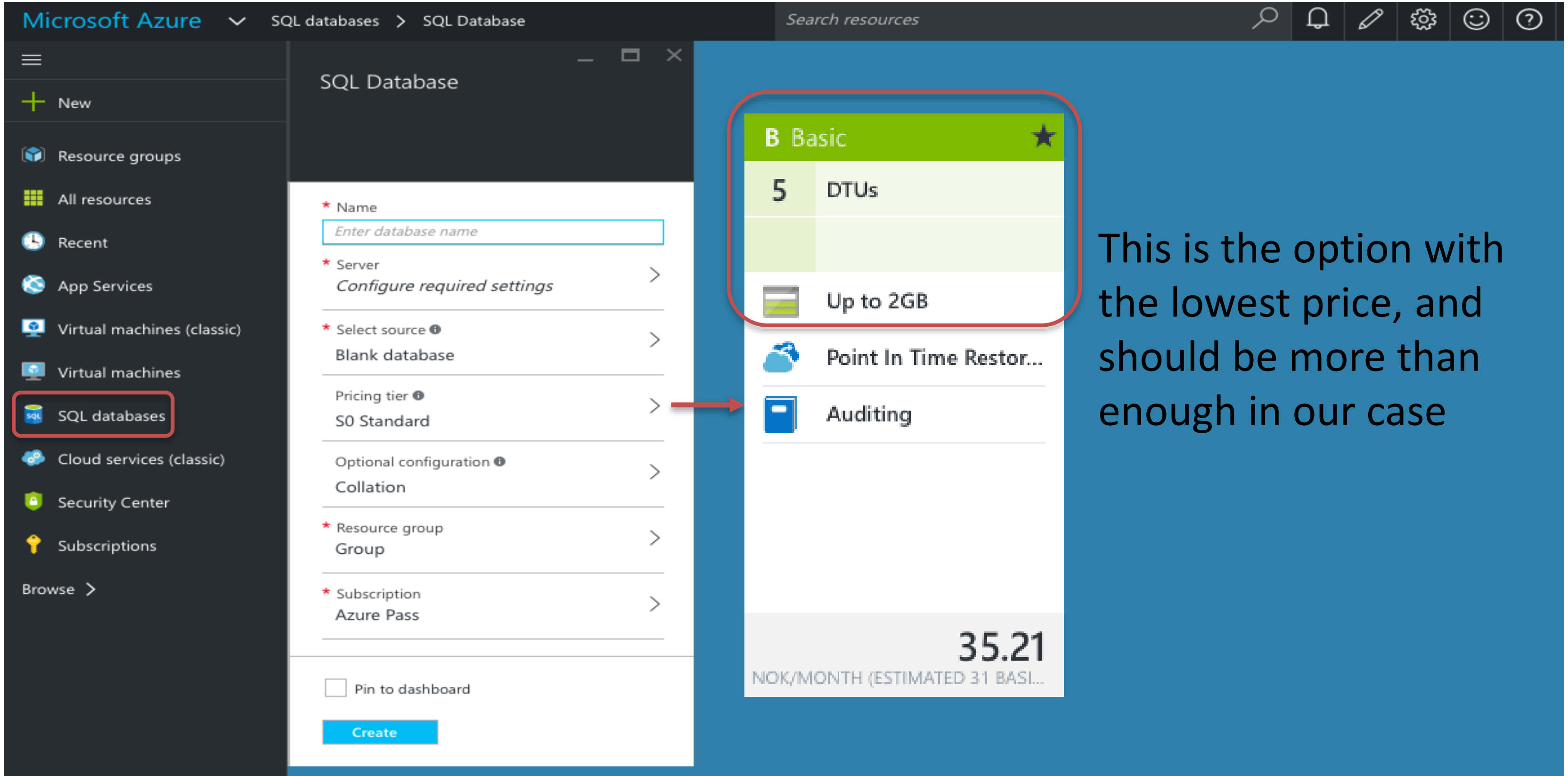
- SQL Server is a Database System from Microsoft
- You can use SQL Server locally, in a network or in a Cloud Service like Microsoft Azure
- In all cases you should have a local SQL Server Management Studio for Configuration (Create Tables, Views, Stored Procedures, etc.)

Cloud-based Datalogging

It is enough storing the Temperature Data (T_{out}) in the Database
- No need to store the Control Signal, Reference Signal, etc. (unless you want)

- Cloud-based Datalogging
- SQL Server stored in Microsoft Azure
- Design (You may use ERwin, but it is not required) and Create necessary Database/Tables.
- Deploy your SQL Server Database into the Cloud using Microsoft Azure
- Extend your existing Control System with Cloud Storage

Create SQL Server Database in Windows Azure



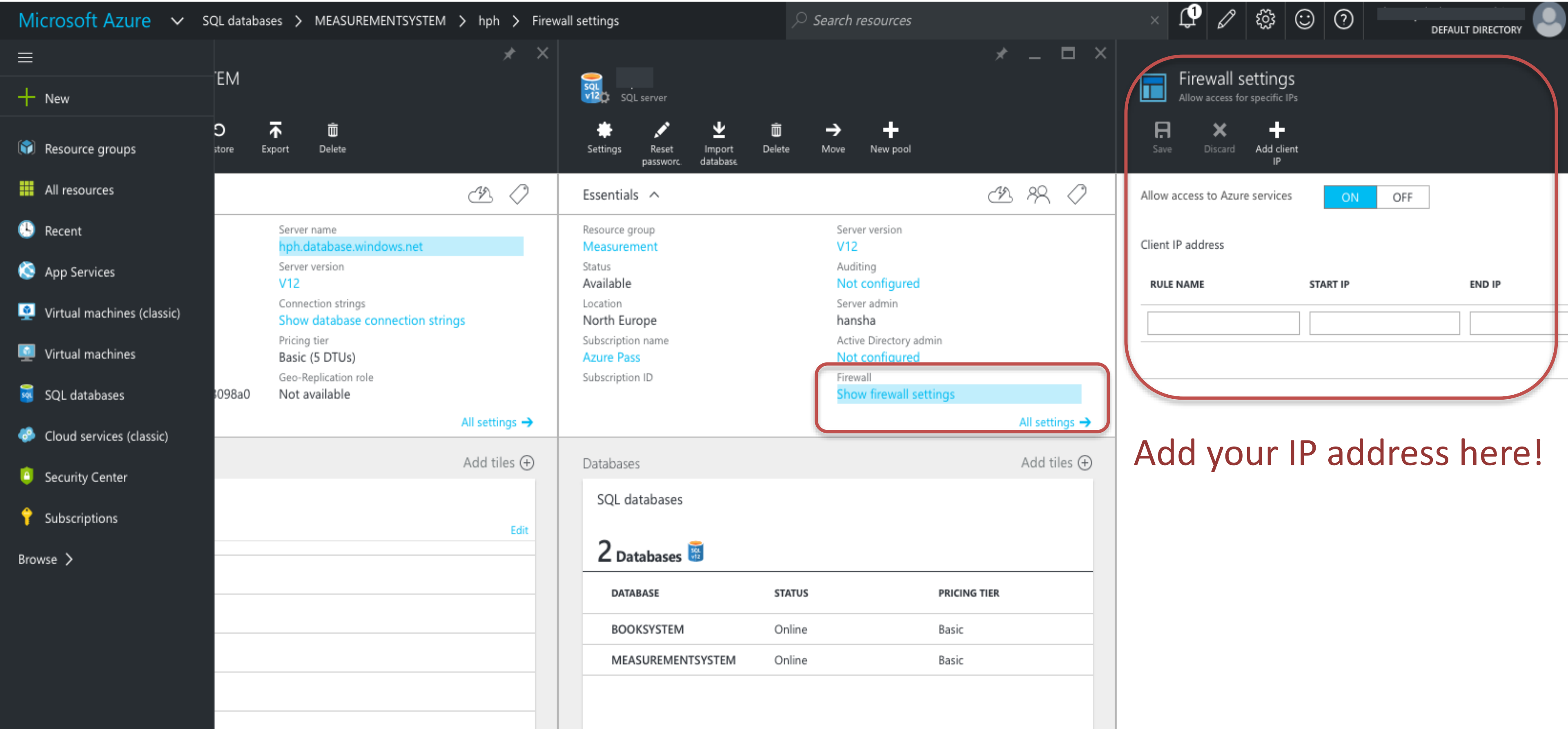
This is the option with the lowest price, and should be more than enough in our case

Connect to the Windows Azure SQL Server from your local SQL Management Studio

1. Configure **Firewall** Setting in Azure Web Portal
2. Your local Management Studio: You connect to the Windows Azure SQL Server Database in the same way as you connect to a local Database
3. Create Tables, Views, Stored Procedures, etc. -> using a SQL Script is recommended!

Note! An alternative is using “**Azure Data Studio**” (which is a lightweight multi-platform (Windows, macOS and Linux) version of SQL Server Management Studio)

Firewall Settings

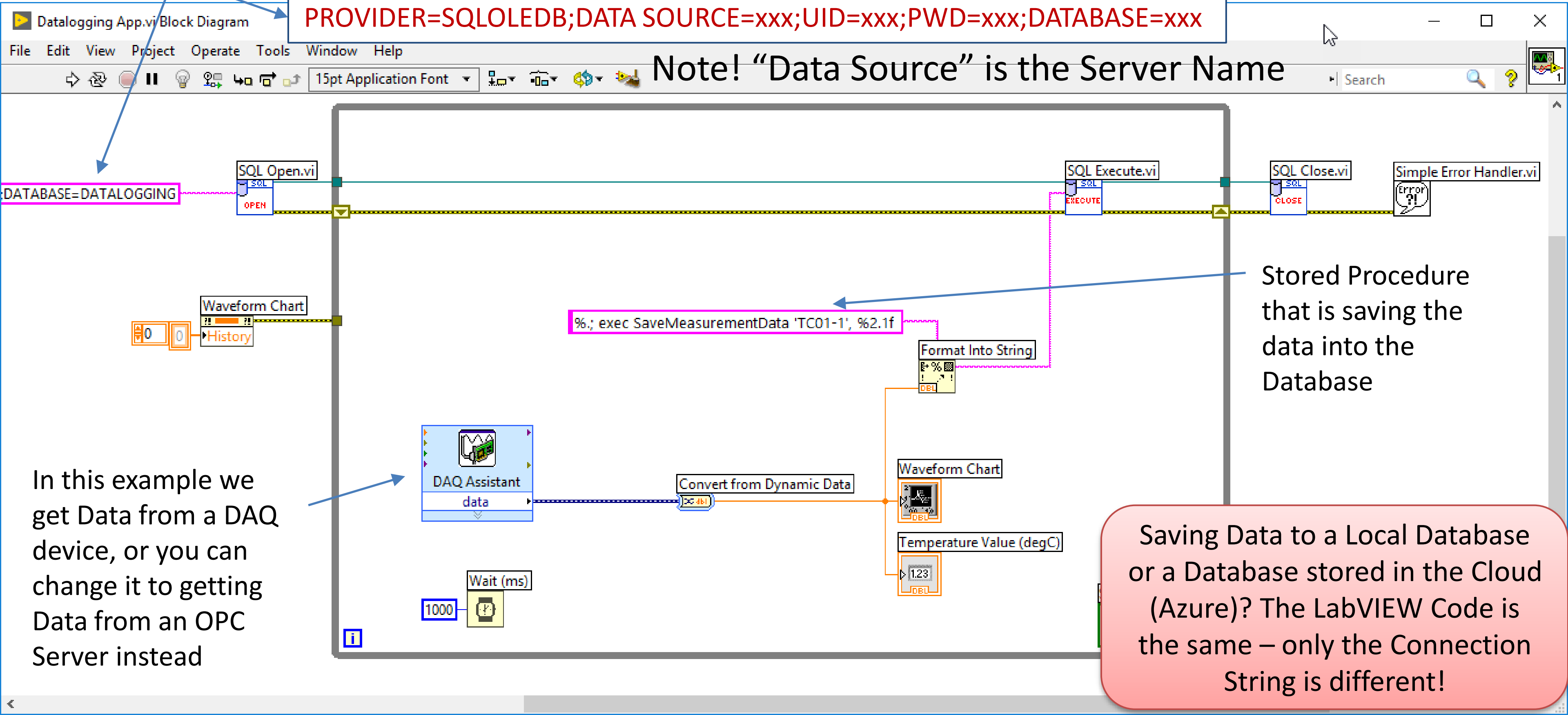


Data Logging App

Connection String to the Database

PROVIDER=SQLOLEDB;DATA SOURCE=xxx;UID=xxx;PWD=xxx;DATABASE=xxx

Note! “Data Source” is the Server Name



Stored Procedure that is saving the data into the Database

In this example we get Data from a DAQ device, or you can change it to getting Data from an OPC Server instead

Saving Data to a Local Database or a Database stored in the Cloud (Azure)? The LabVIEW Code is the same – only the Connection String is different!

Datalogging - Tips and Tricks

1. **Create a Local Database in SQL Server Management Studio**. Add necessary Tables, etc.
2. Start to **Create a Datalogging App** that stores Data in the Local Database. Use a Connection String like this:
   PROVIDER=SQLOLEDB;DATA SOURCE=xxx;UID=xxx;PWD=xxx;DATABASE=xxx
3. Make sure it works before you move on to the next step
4. Then, **Create a Database in the Azure** Portal Web Site. Make sure to write down the Server Name, your Server Login and Password.
5. **Connect to the Azure Database from your local SQL Server Management Studio** (Make sure to add your IP address in the Firewall Settings). Add necessary Tables, etc.
6. **Change the Connection String** in your LabVIEW Datalogging Application from using the Local Database to using the Azure Database. You only need to change Data SOURCE (Server), UID, PWD and DATABASE - the rest of the LabVIEW Code will remain the same. It should now work!

ASP.NET Core Web Programming

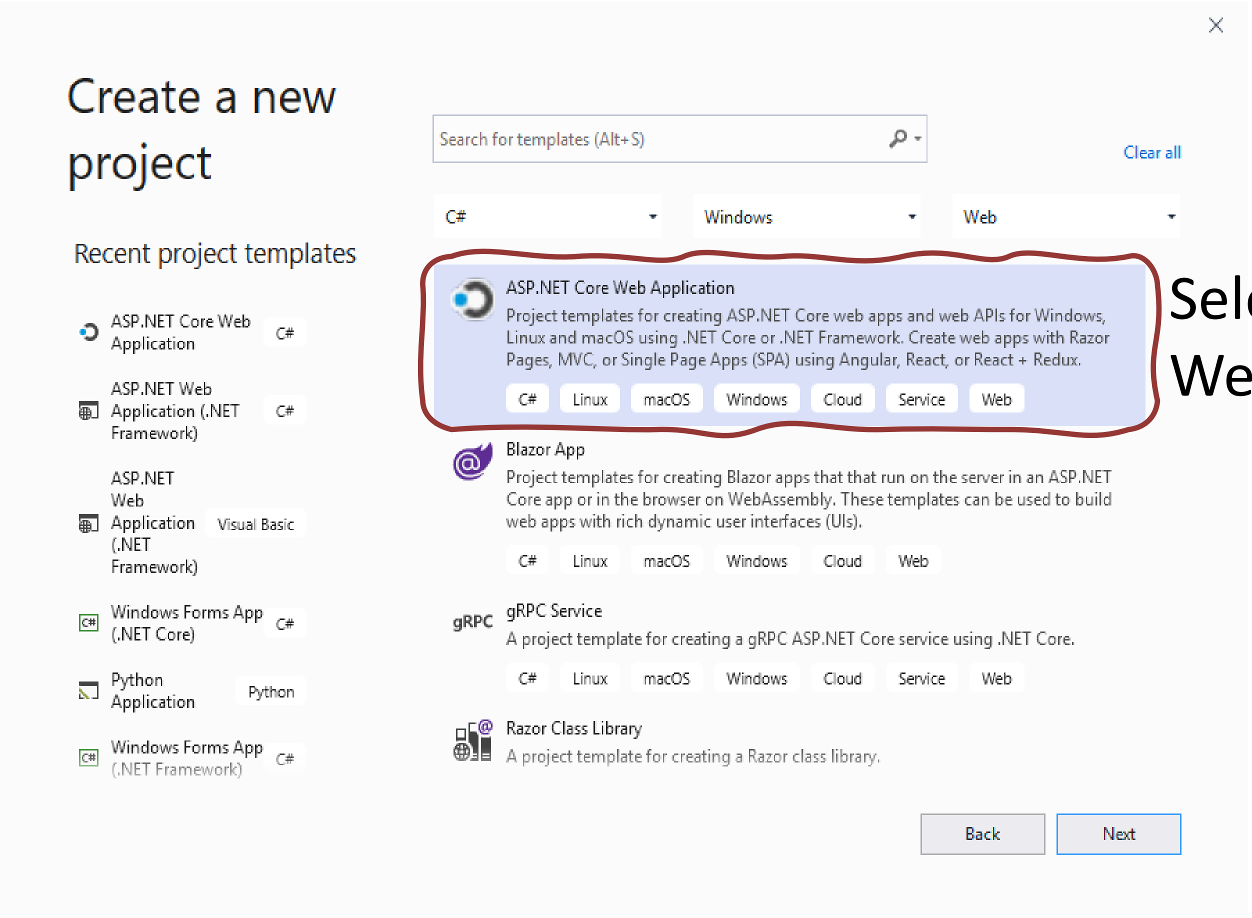
ASP.NET Core share a lot with ordinary Win Forms that you are already familiar with (you use C#).
Feel free to reuse previous code and examples from the SCADA assignment in Industrial IT, or other C# code you have created in previous courses.

Hans-Petter Halvorsen

ASP.NET Core Web Application

- ASP.NET is a Web Framework for creating Web Applications
- ASP.NET is integrated with Visual Studio and you will use the C# Programming Language
- .NET Core is cross-platform, meaning it will work on Windows, Linux and macOS.
- ASP.NET Core is Microsoft’s newest baby and it is the future of Web Programming

ASP.NET Core in Visual Studio



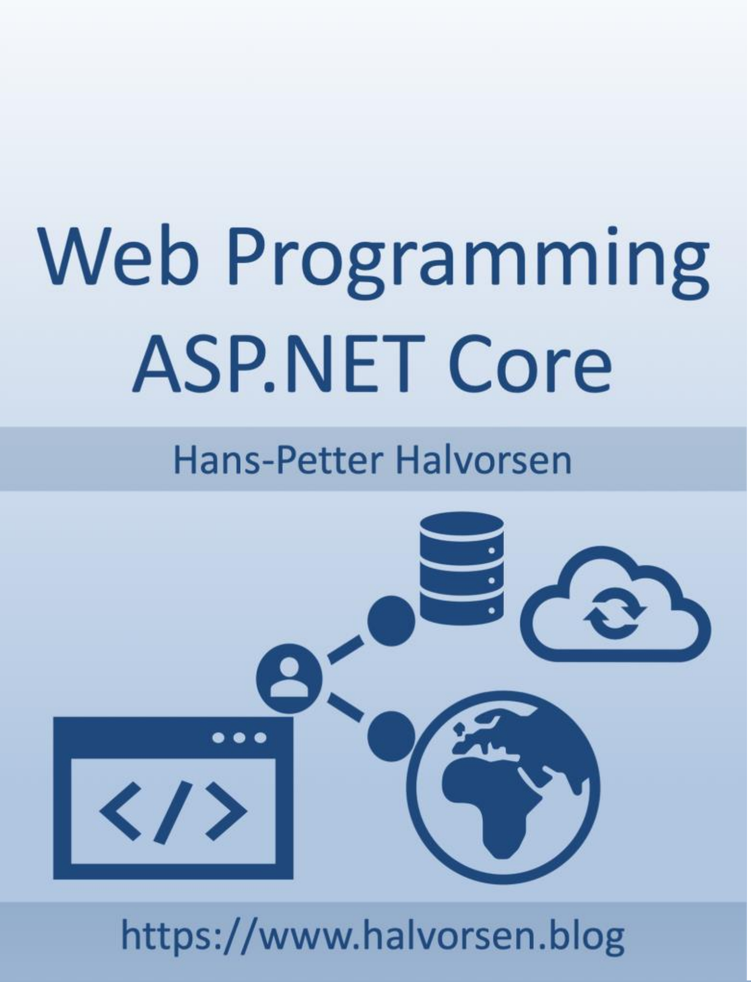
Select the ASP.NET Core Web Application Project

ASP.NET Core Examples

Recommended Videos:

- ASP.NET Core – Introduction: https://youtu.be/zkOtiBcwo8s
- ASP.NET Core - Database Communication: https://youtu.be/0Ta3dQ3rxzs
- ASP.NET Core - Database CRUD Application: https://youtu.be/k5TCZDwTYcE
- ASP.NET Core - Charts: https://youtu.be/mksUls9fx-Q

Download Examples here: https://www.halvorsen.blog/documents/programming/web/aspnet

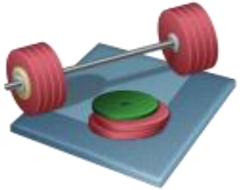


ASP.NET Core Resources

- Textbook
- Videos
- Tutorials
- Example Code
- Example Applications

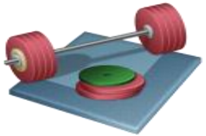
https://www.halvorsen.blog/documents/programming/web/aspnet

Monitoring and Analysis in the Cloud



- Monitoring and Analysis in the Cloud
- Web-based (ASP.NET Core/C#) system hosted at Microsoft Azure
- Create a ASP.NET Core Web Site for Monitoring your Data
- The Web Site shall be deployed to Microsoft Azure

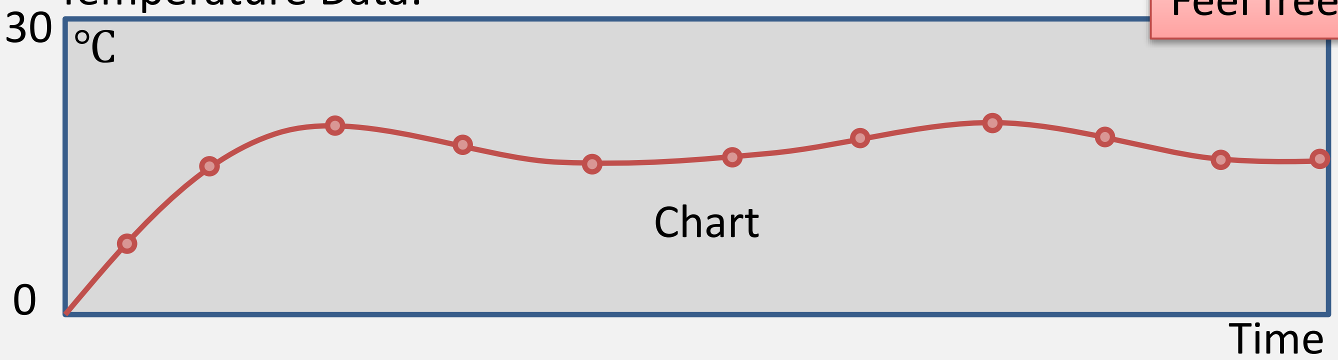
ASP.NET Core Example



Data Monitoring App

This is just a Design Sketch!
Feel free to make your own GUI!

Temperature Data:



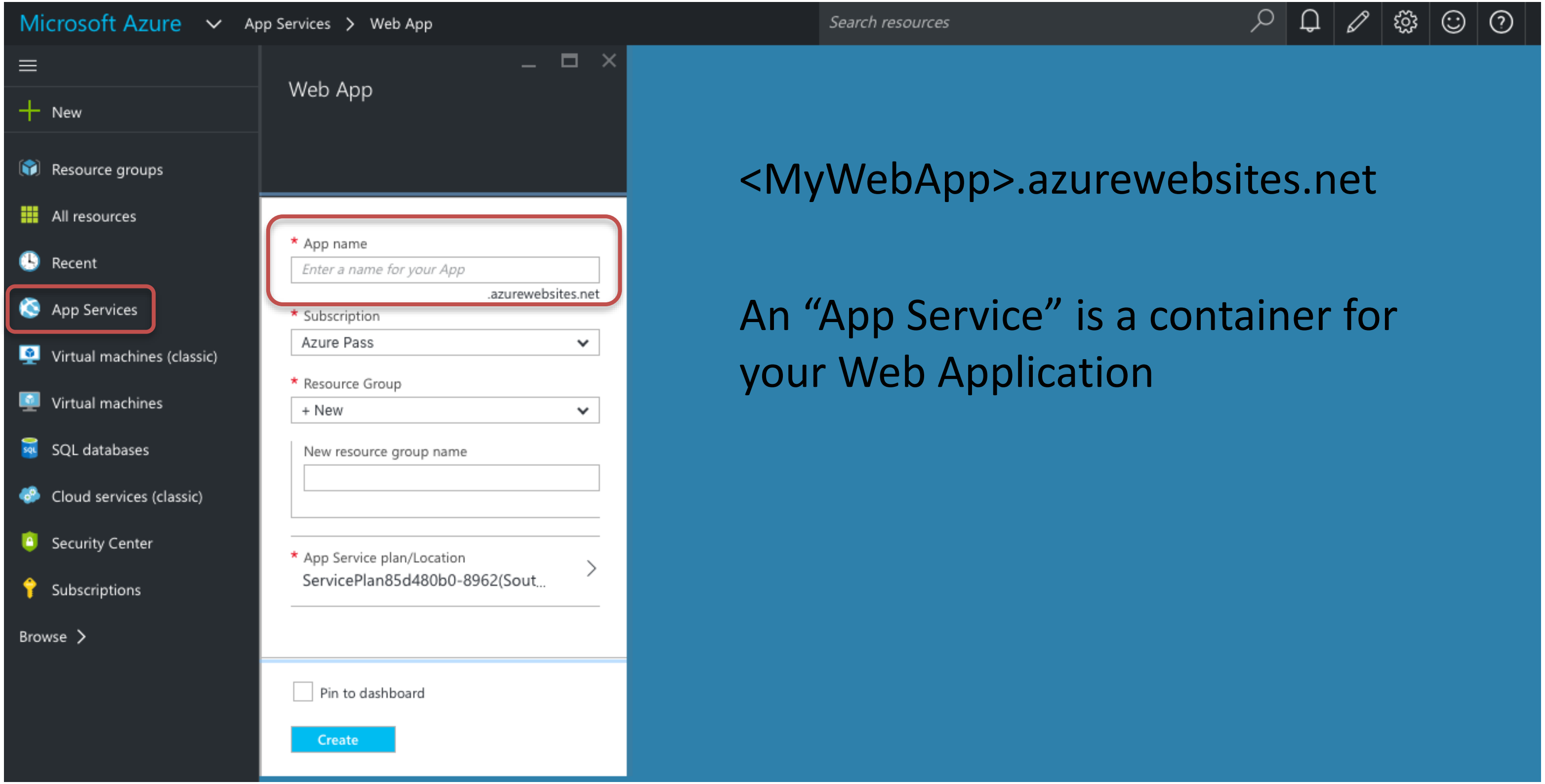
Date & Time	Process Value [C]	Control Value [%]
2016.03.22 14:45	22	
…	…	…
…	…	…
…	…	…

GridView

You should get the Data from the Database

Typically you get Data from the Database using Views and/or Stored Procedures

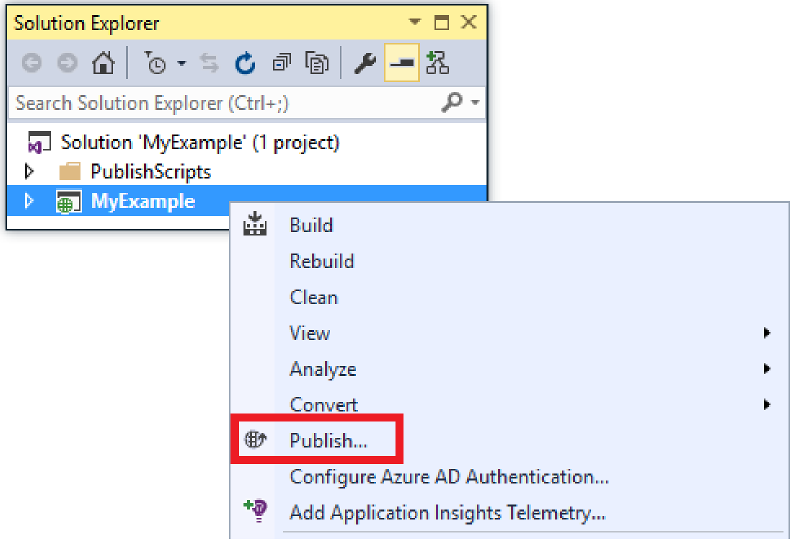
Create App Service from Azure Portal

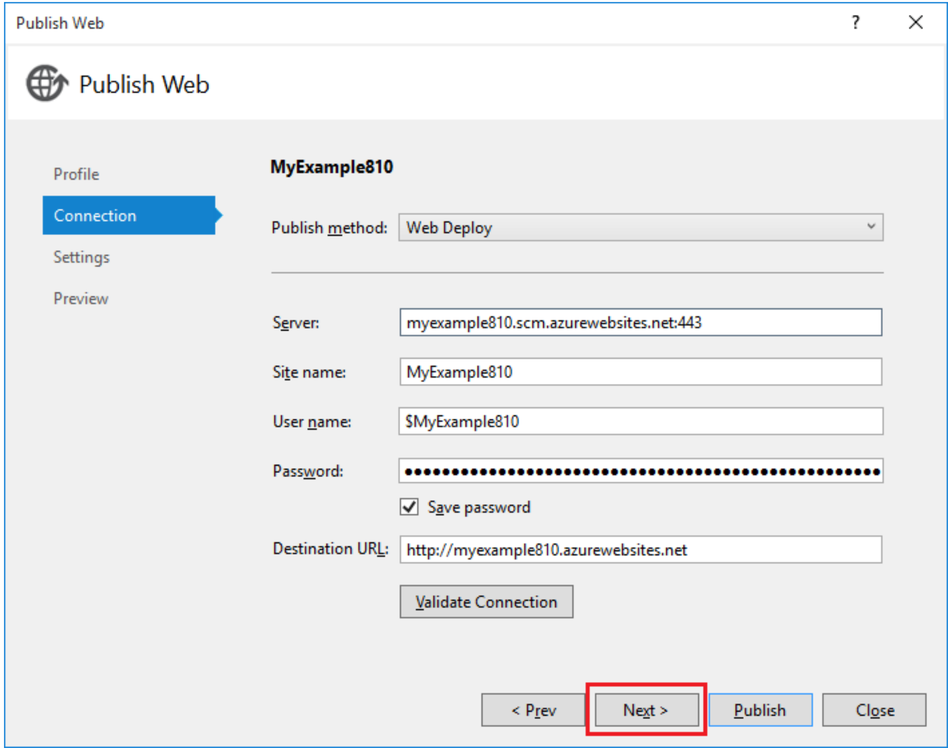


<MyWebApp>.azurewebsites.net

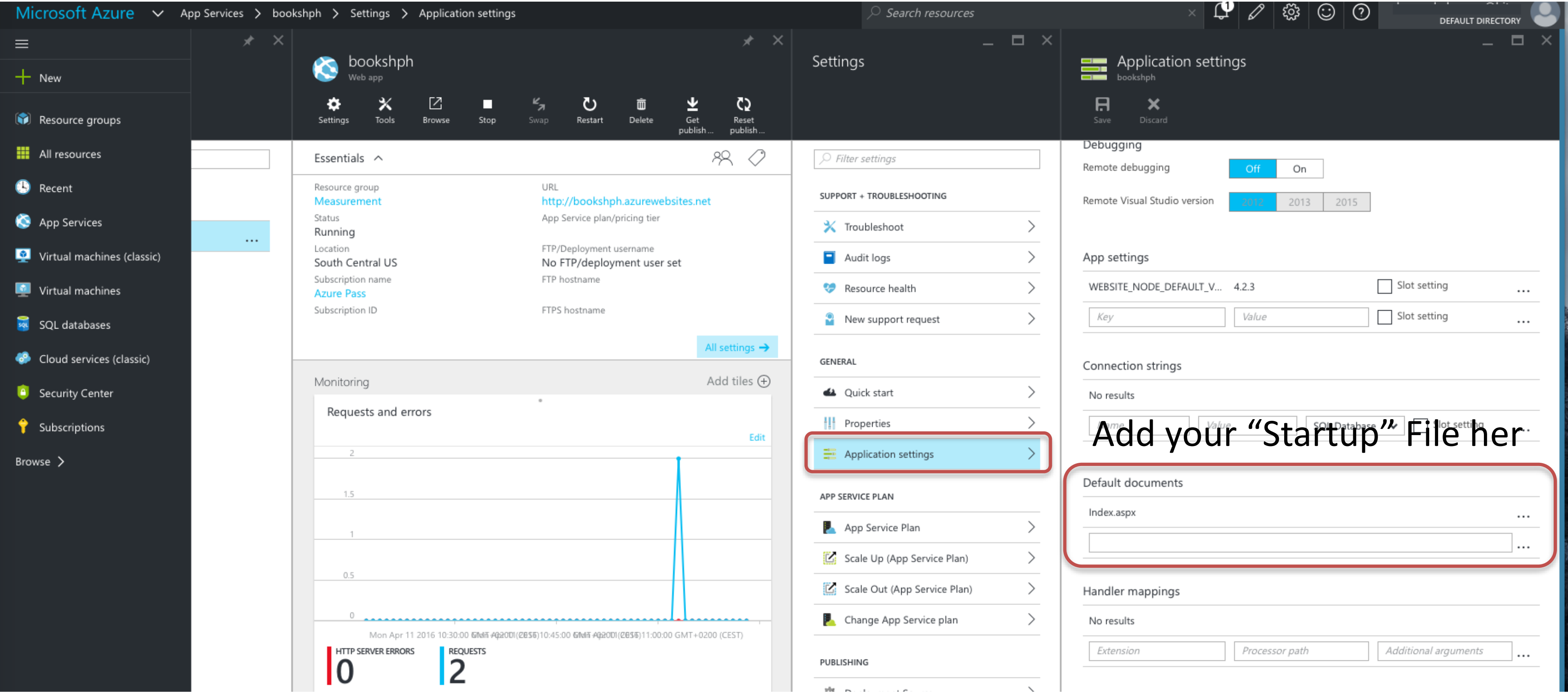
An “App Service” is a container for your Web Application

Deploy the Web Project to the Azure Web App from Visual Studio





Configure Default Document



Data Monitoring - Tips and Tricks

1. **Use your Local Database**.
2. Start to **Create the ASP.NET Core Monitoring App** that gets Data from your Local Database
3. Make sure it works (Test it from Visual Studio) before you move on to the next step
4. Then, Start using your Database in the Azure. In addition to your existing Azure Database you need to **Create an App Service in Azure Portal Web Site**
5. **Change the Connection String** in your ASP.NET Core Application from using the Local Database to using the Azure Database. You only need to change Data SOURCE (Server), UID, PWD and DATABASE - the C# Code will remain the same. Make sure it works (Test it from Visual Studio) before you move on to the next step
6. **Deploy** the ASP.NET Application to Azure. Then, type in the URL in your Web Browser and make sure it Works (https://**appname**.azurewebsites.net)

Web Pages and Real-time Monitoring?

- Web Pages are typically not used for Real-time Monitoring, and **not** necessary to to implement in this assignment.
- A simple solution though is to put like this in your web page:

```
<html>
<head>
  <title>Data Monitoring</title>
  <meta http-equiv="refresh" content="30"/>
</head>
<body>
  ..
</body>
</html>
```

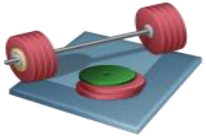
This line refreshes the web page every 30 seconds

Note! For more advanced Real-time updates of Web pages, you typically use something called AJAX and JavaScript – but that is really NOT part of this assignment!

Cyber Security and GDPR

Hans-Petter Halvorsen

Cyber Security and GDPR



- What is GDPR?
- Data Security in Automation Systems?
- IoT solutions and Data Security?
- Data Security in Cloud Storage and Cloud Services?
- What can be done to protect the system (and data) you have created?

Congratulations! - You are finished with all the Tasks in the Assignment!

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog