

<https://www.halvorsen.blog>



# Hardware in the Loop Simulation and Testing

Hans-Petter Halvorsen

# Background

- You work as a **System Engineer** for a System Engineering Company.
- An Industrial Production Company has announced a competition between several selected System Engineering companies to perform a preliminary Project.
- **Your Assignment is to develop a Control System Prototype/PoC** where **Hardware in the Loop Simulation** is an essential part of the Development and Testing.
- In the PoC, a **Fuji PXG5 PID** will be used to demonstrate the principles. The Fuji PXG5 PID shall be used to control an industrial small-scale process.
- The HIL System should be implemented in **LabVIEW**.
- To create proper and user-friendly **GUI/HMI** is an important part of the Prototype.
- **The delivery is a Technical Report** where you shall give an overview of the entire system made, including the Methods used and the Results archived.
- The PoC and the Report will be an important foundation for decision making within the company when it comes to the final implementation of the system sometime in the future. Note! Multiple System Engineering companies have been given this opportunity, so it is important that you “**Add Value**” and stand out compared to the others in order to be selected as the final Contractor.

# System Requirements

1. Perform Control and Simulations in LabVIEW using built-in PID Controller (Software only)
  - Create, Simulate and Control a Mathematical model of an Air Heater (Software only).
  - Mathematical Model of Air Heater is provided, but proper Model Parameters need to be found
  - Find Proper PI(D) Parameters. Data should be saved to a Text File.
2. Perform HIL Simulation using Fuji PXG5 PID (HW + SW)
  - Control Mathematical Model using Fuji PXG5 PID
  - Test Auto-tuning with Fuji PXG5 PID
3. Implement and Test Fuji PID PXG5 on Real Process (Hardware only)
4. Make sure to properly Test and Document the Performance of the Control System

These are the complete requirements for the assignment. The rest of this document contains resources like additional information, code examples, tips and tricks, step-by-step instructions, etc. that you can use at your own discretion.



# Resources



# Table of Contents

1. [Introduction](#)
2. [Introduction to HIL](#)
3. [Simulated Control System](#)
  - [Air Heater Model](#)
  - [LabVIEW PID Controller](#)
  - [Lowpass Filter](#)
4. [HIL + Fuji PXG5 PID Controller](#)
5. [Real Control System: Fuji PXG5 + Air Heater](#)
6. [Digital Twins](#)



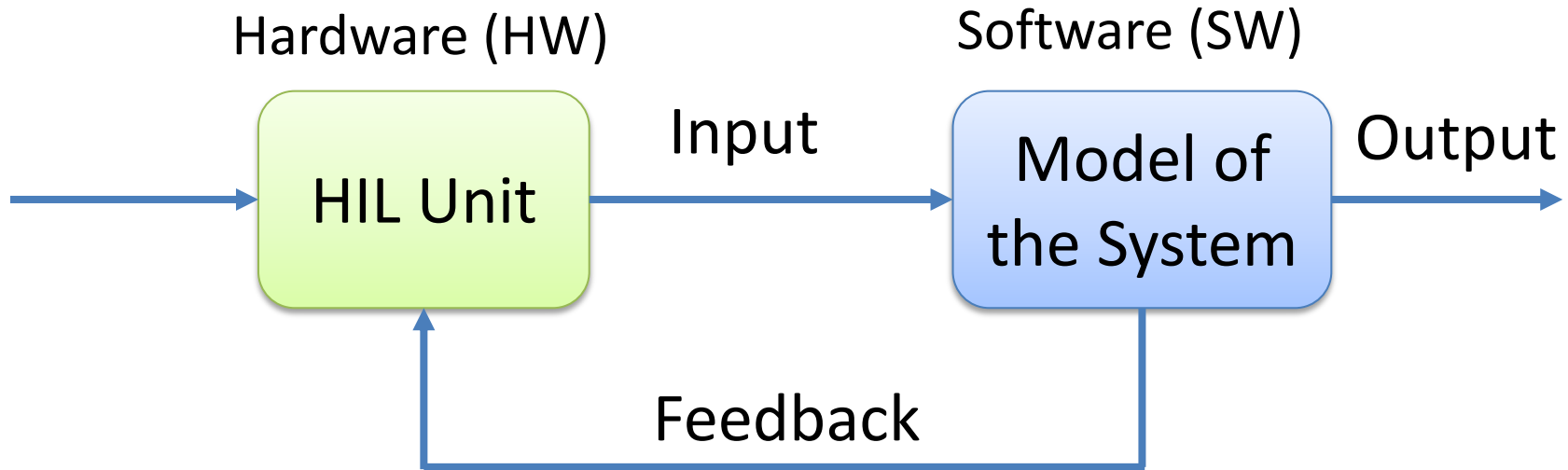
# Introduction

# Fuji PXG5 PID Controller

- We will create a Simulator (Mathematical model) that shall be controlled by the PID controller
- The main Hardware in this Lab is an industrial PID controller
- The PID controller device will be the “Hardware in the Loop”
- The aim is to test the the device before you put it into production. This is referred to as “Hardware in the Loop Simulation and Testing



# General HIL Concept Sketch



# Learning Goals

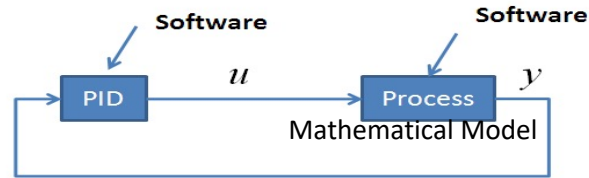
- Learn key concepts within Hardware in the loop (HIL) simulations and testing
- Learn practical skills and implementation of HIL applications
- Learn practical skills in Modelling, Control and Simulation
- Learn practical implementation of Industrial Control Systems
- Learn more LabVIEW Programming
- Learn about Hardware-Software Interactions
- Learn Practical Skills and Implementations in general
- Learn Software Installation in general, which can be cumbersome with many pitfalls
- Learn to use and create Software in general in a professional manner

# HIL Simulation - Step-by-step

1

## Step 1: Ordinary Software Simulation:

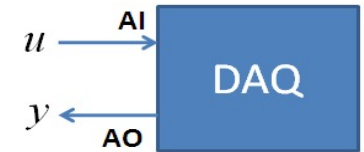
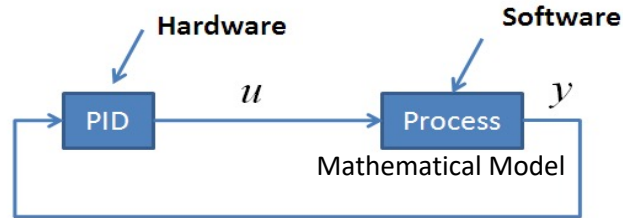
Purpose: Create, Simulate, Control and Test the Mathematical Model



2

## Step 2: HIL Simulation and Testing:

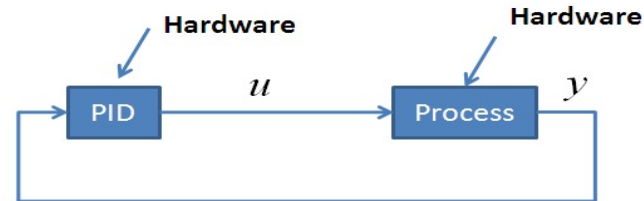
Purpose: Test your Hardware device before you apply it on your real system. Also useful for Training purposes.



3

## Step 3: Running the Real System:

Purpose: Apply your Hardware in the Production System



# Step 1: Simulation

Control and Simulation in LabVIEW using built-in PID Controller and Mathematical Model of the Process (Software only)



Computer (with LabVIEW)

# Step 2: Use PXG5 with Model

PXG5 Industrial  
PID Controller



PID Controller

Analog Out  
(Process Value)

1-5V  
AO0  
GND

Note!!

AIO+  
AIO-

Analog In  
(Control Signal)

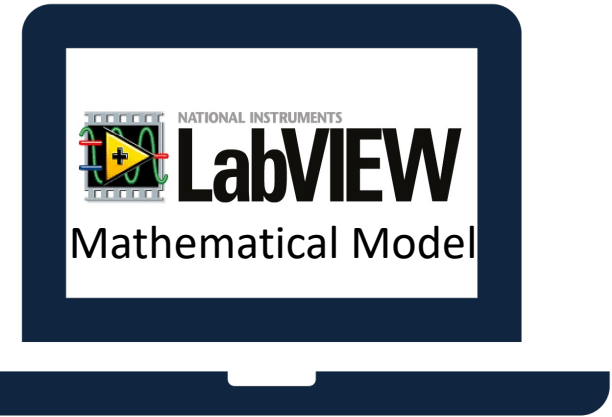
0-5V

USB



I/O Module  
(USB-6008)

Model of Process (Air Heater)



Computer (with LabVIEW)

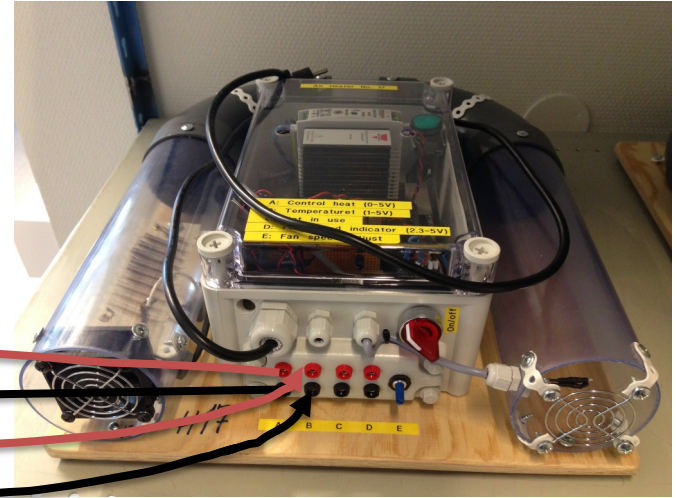


# Step 3: Use PXG5 with Real System

PXG5 PID Controller



Process (Air Heater)



Process Value  
1-5V

Control Signal  
0-5V

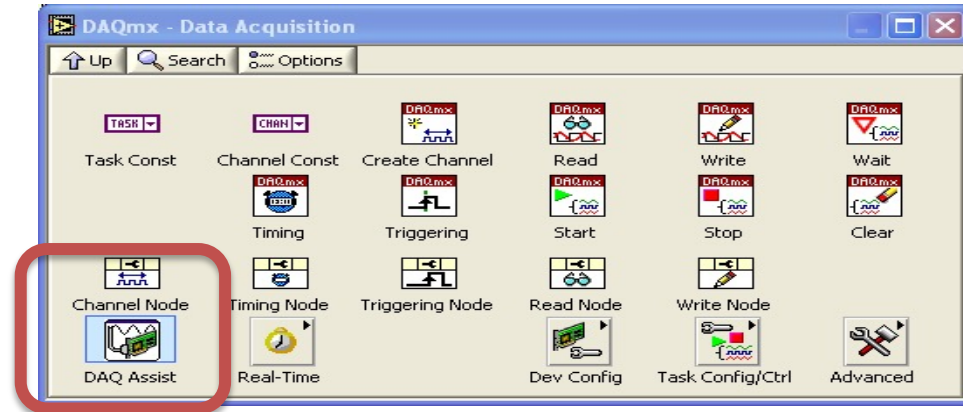
Industrial PID Controller

# Software

- LabVIEW
- LabVIEW Control & Simulation Module
- DAQmx Driver Software



Make sure to install the necessary Software before you go to the laboratory!

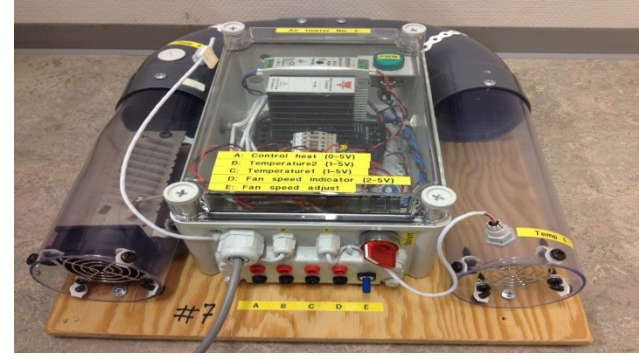


# Hardware



USB-6008

Air Heater



Fuji PXR5/PXG5 PID Controller



Multimeter



Your Personal Computer

We have limited numbers of PID Controllers, so, 2-3 students may need to share a PID Controller

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!

# HELP WANTED!

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

Use the **Debugging Tools** in your Programming IDE.

Visual Studio, LabVIEW, etc. have great Debugging Tools! Use them!!



“Google It”!

You probably will find the answer on the Internet



Another person in the world probably had a similar problem



My System is not Working??



Use Microsoft Teams



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

Multimeter, etc.



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.

# Lab Assignment Guidelines

- Make sure to read the whole assignment before you start to solve any of the problems.
- If you miss assumptions for solving some of the problems, you may define proper assumptions yourself.
- The Tasks described in the Assignment are somewhat loosely defined and more like guidelines, so feel free to interpret the Tasks in your own way with a personalized touch.
- Feel free to **Explore!** Make sure to **Add Value** and **Creativity** to your Applications!
- Try to add some extra value and be creative compared to the simplified examples given by me, in that way you learn so much more.

# Lab Assignment Guidelines

- Think about the Lab Assignment as a small real-life industrial Project, and not a set of tasks or exercises.
- 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 a Lab Assignment or a set of exercises.
- Try to see the big picture. The tasks within the assignment are just just small building blocks that ends up with a fully working system.
- It is recommended that you make a Work Plan and a System Sketch that gives you an overview of what YOU should do



# Lab Work Requirements

- Make sure to see the “**Big picture**” – you don’t need to document every single step you have made. Focus on what’s important (your final system).
- Your GUIs is important! - make sure to make them user friendly and intuitive. You create this on behalf of someone that are going to use your applications.
- Make sure to always add **Units** in your GUI, charts, documentation, etc.
- **Presenting values with 4+ decimals makes no sense!** E.g., a temperature sensor is not that accurate. You can easily change number of decimals that you present in your GUI in LabVIEW, C#, etc.
- The **Quality** of the LabVIEW code is important. Make sure to use "straight lines" in your LabVIEW code, etc. The code should also flow from left to right, not opposite direction. You create this on behalf of someone that are going to use your applications. Neat code makes it easier to develop, maintain, find code errors, etc.
- In general, make sure that you take some pride in your applications and the work that you do. It's not about getting finished as soon as possible. The mission is to learn as much as possible within a given timeframe. Try to change the mindset.
- To improve the LabVIEW code, please see this video: LabVIEW Applications using State Machine: <https://youtu.be/-b9St8wNhpQ>

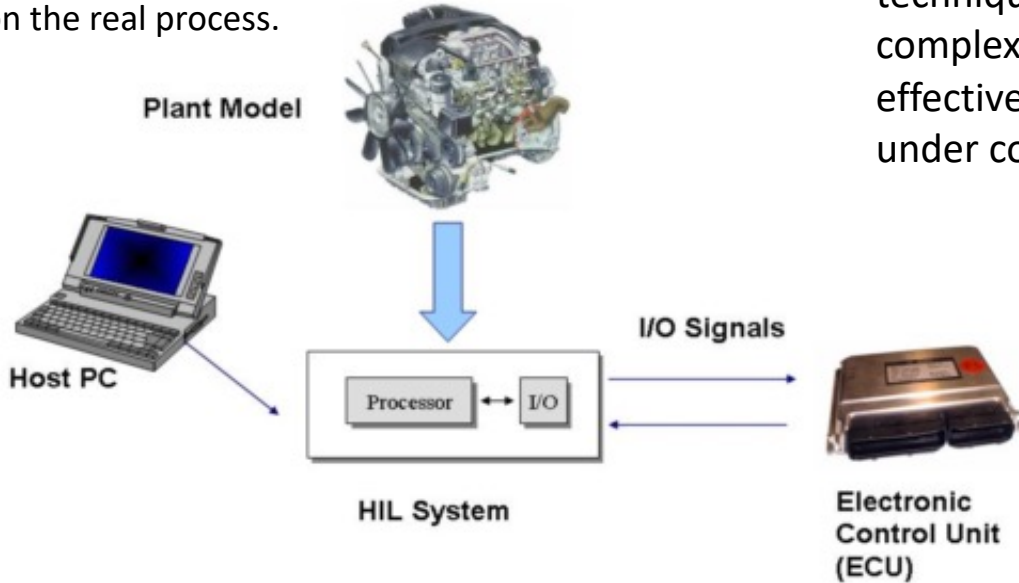




# Introduction to HIL

# What is HIL Simulation?

The main purpose with the HIL Simulation is to Test the Hardware device on a Simulator before we implement it on the real process.



Hardware-in-the-loop (HIL) simulation and testing is a technique that is used in the development and test of complex process systems. HIL simulation provides an effective platform by adding the complexity of the plant under control to the test platform.

The complexity of the plant under control is included in test and development by adding a mathematical representation of all related dynamic systems. These mathematical representations are referred to as the “plant simulation.”

Hardware-In-the-Loop is a form of real-time simulation. Hardware-In-the-Loop differs from real-time simulation by the addition of a real component in the loop. This component may be an “Electronic Control Unit” (ECU).

[Wikipedia]

# Example of PC-based Control System

(We shall not do like this in this Assignment!)

Process (Air Heater)

Computer (with LabVIEW)



PID Control and Monitoring

USB



I/O Module (USB-6008)

Analog In  
(Process Value)

A10+

1-5V

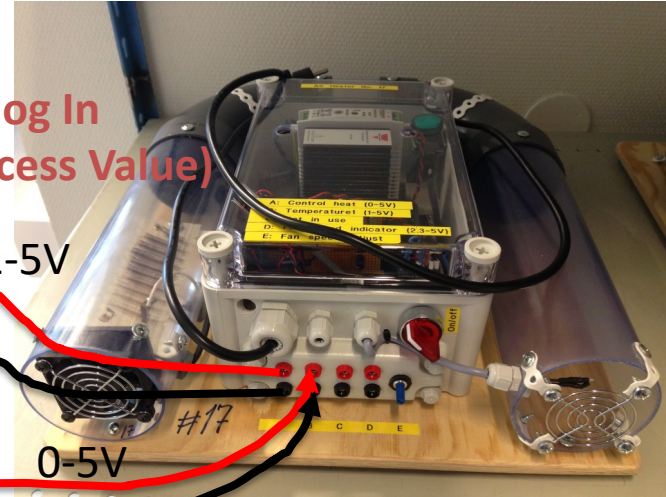
A10-

A00

0-5V

GND

Analog Out  
(Control Signal)

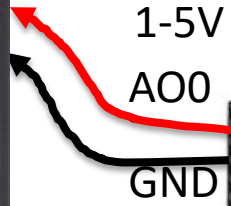


# Example of HIL Simulation

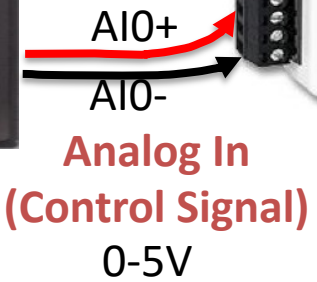


PID Control

Analog Out  
(Process Value)



Note!!



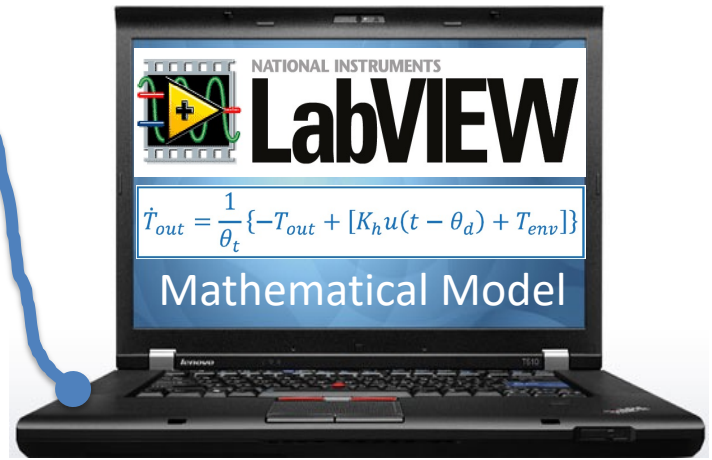
Analog In  
(Control Signal)

0-5V



I/O Module  
(USB-6008)

Model of Process (Air Heater)



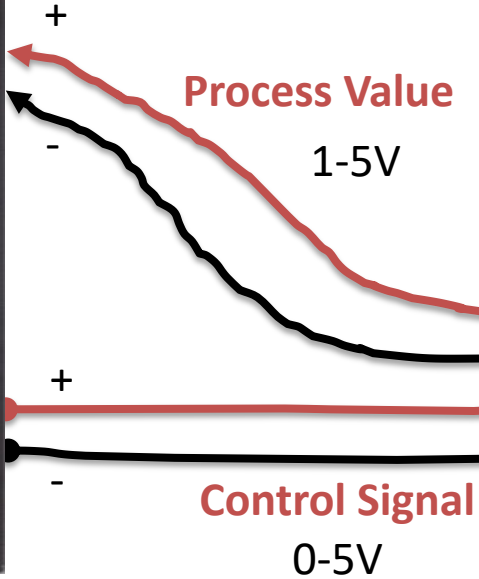
Computer (with LabVIEW)

# Example of Industrial Single Loop PID Control

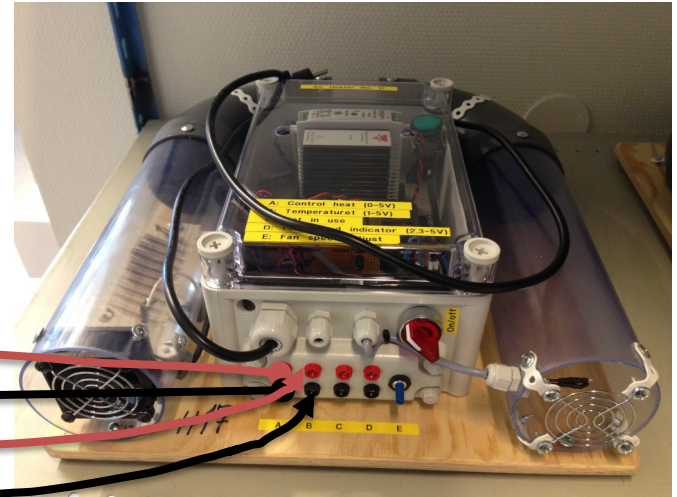
(This is our final goal in this Assignment)



Industrial PID Controller



Process (Air Heater)





# Examples of Industrial Control Systems (ICS)

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



National Instruments  
cRIO

Programmable Automation Controller (PAC)

4



LabVIEW

1

Industrial PID Controller



5

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



I/O Module

2 Distributed Control Systems (DCS)



Controller

I/O Modules

DeltaV from Emerson

3 PLC (Programmable Logic Controller)



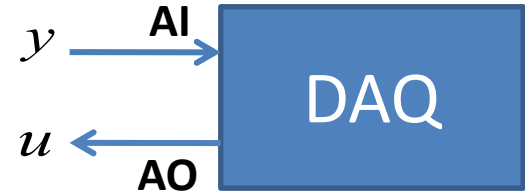
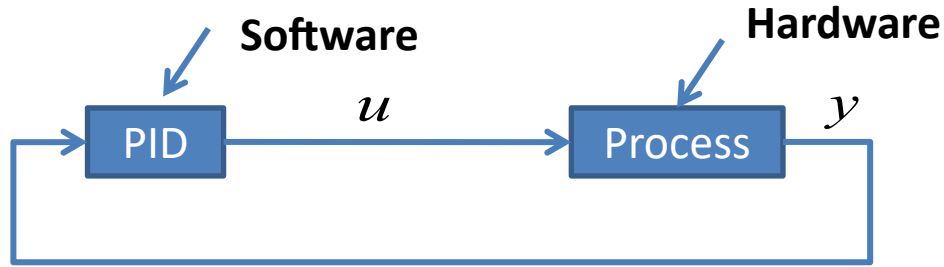
Siemens PLC

# HIL Background

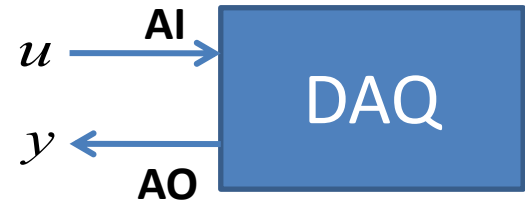
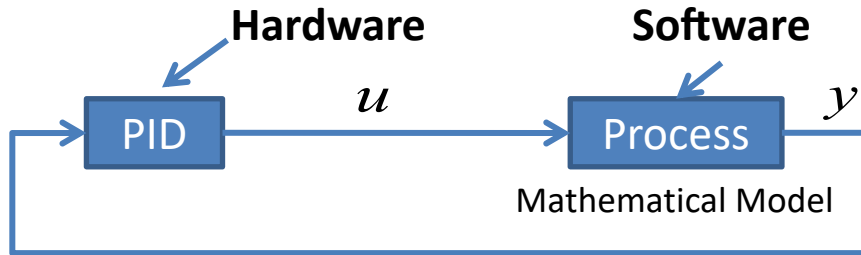
- Typically, a simulator communicates with an “ECU” (“Electronic Control Unit”) via ordinary I/O. Such a system - where the real controller is controlling a simulated process is denoted Hardware-in-the-loop (HIL) simulation.
- **The main purpose with HIL is to test the hardware device on a simulator 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

Traditional Process Control using Software for Implementing the Control System



## HIL Simulation

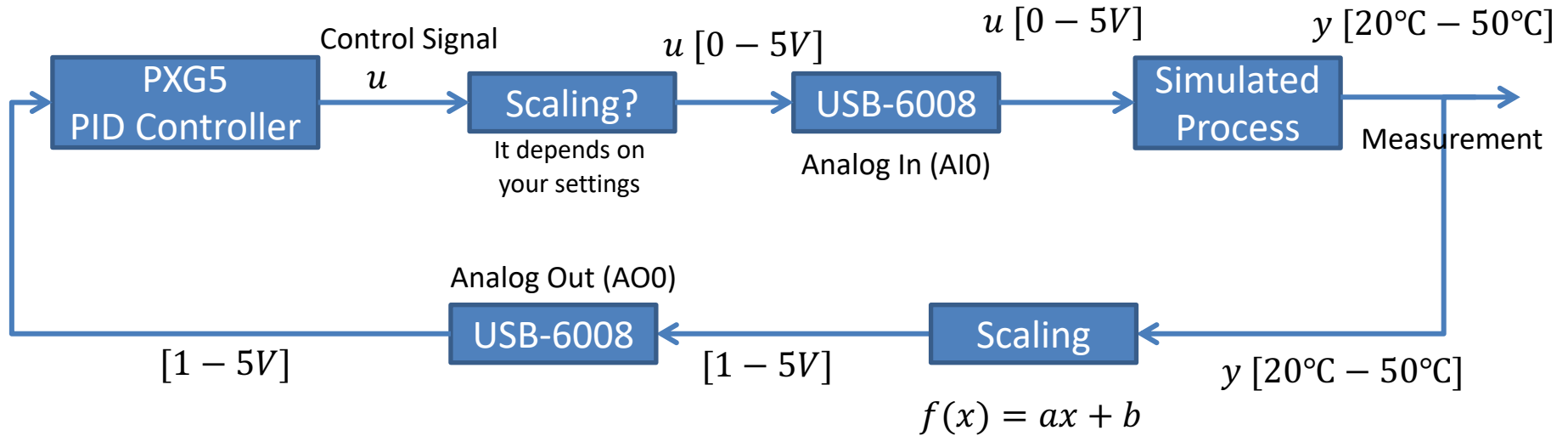




# HIL Simulation

- 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 Simulation using PXG5 PID





Step 1: Ordinary Software Simulation

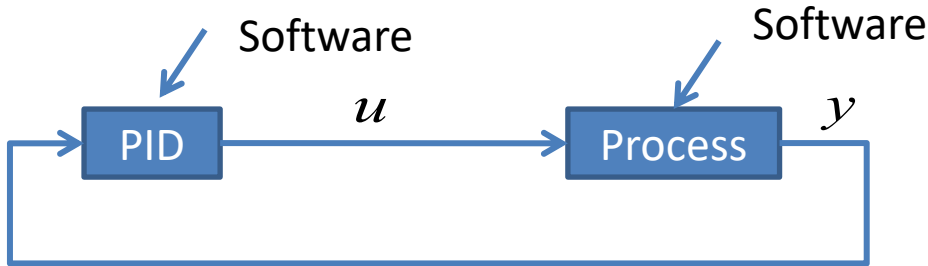
# Simulated Control System

Hans-Petter Halvorsen

[Table of Contents](#)

# Simulated Control System

Purpose: Create, Simulate, Control and Test the Mathematical Model.  
Find Proper PID Parameters.





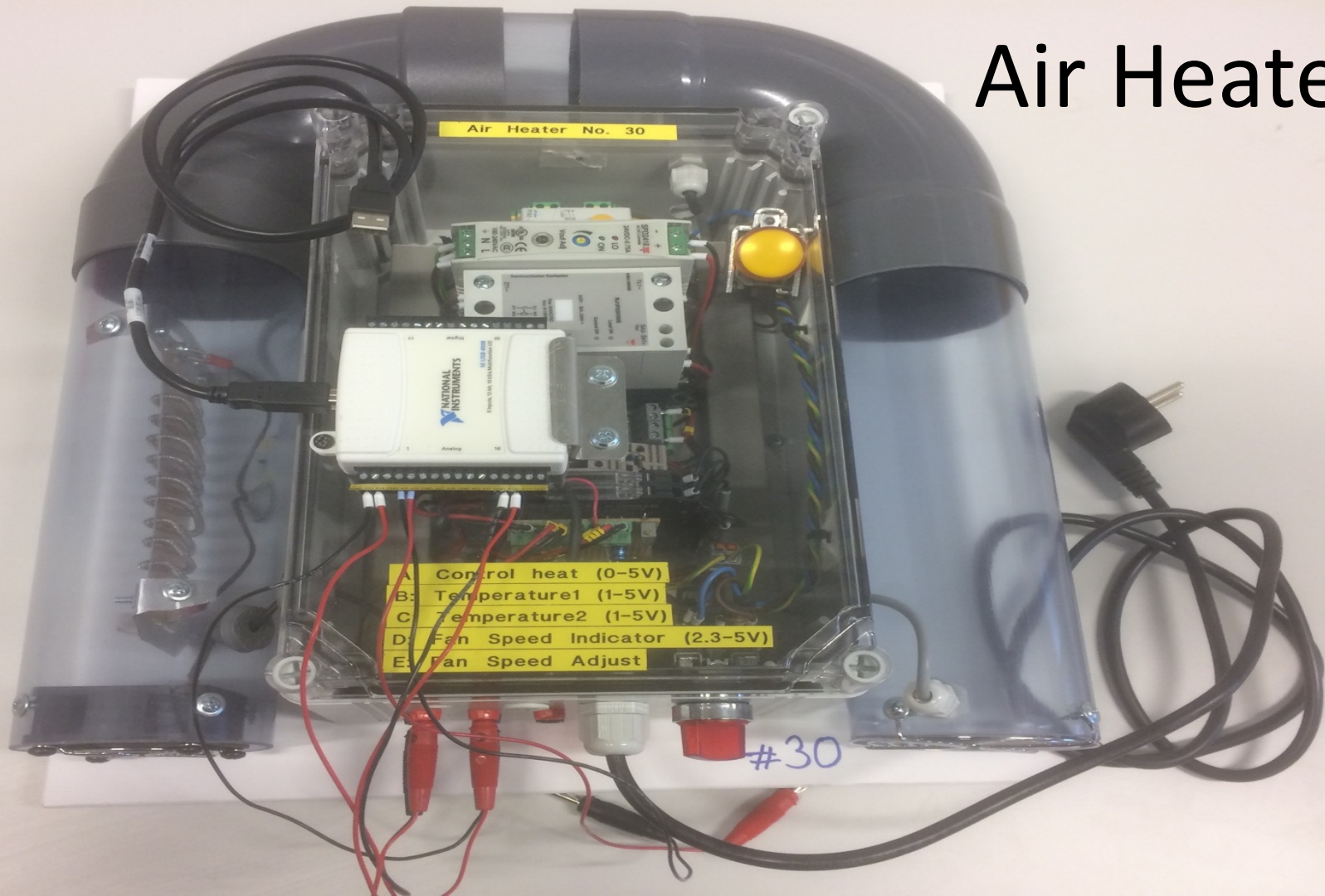
# Air Heater System

Small-scale Laboratory Process

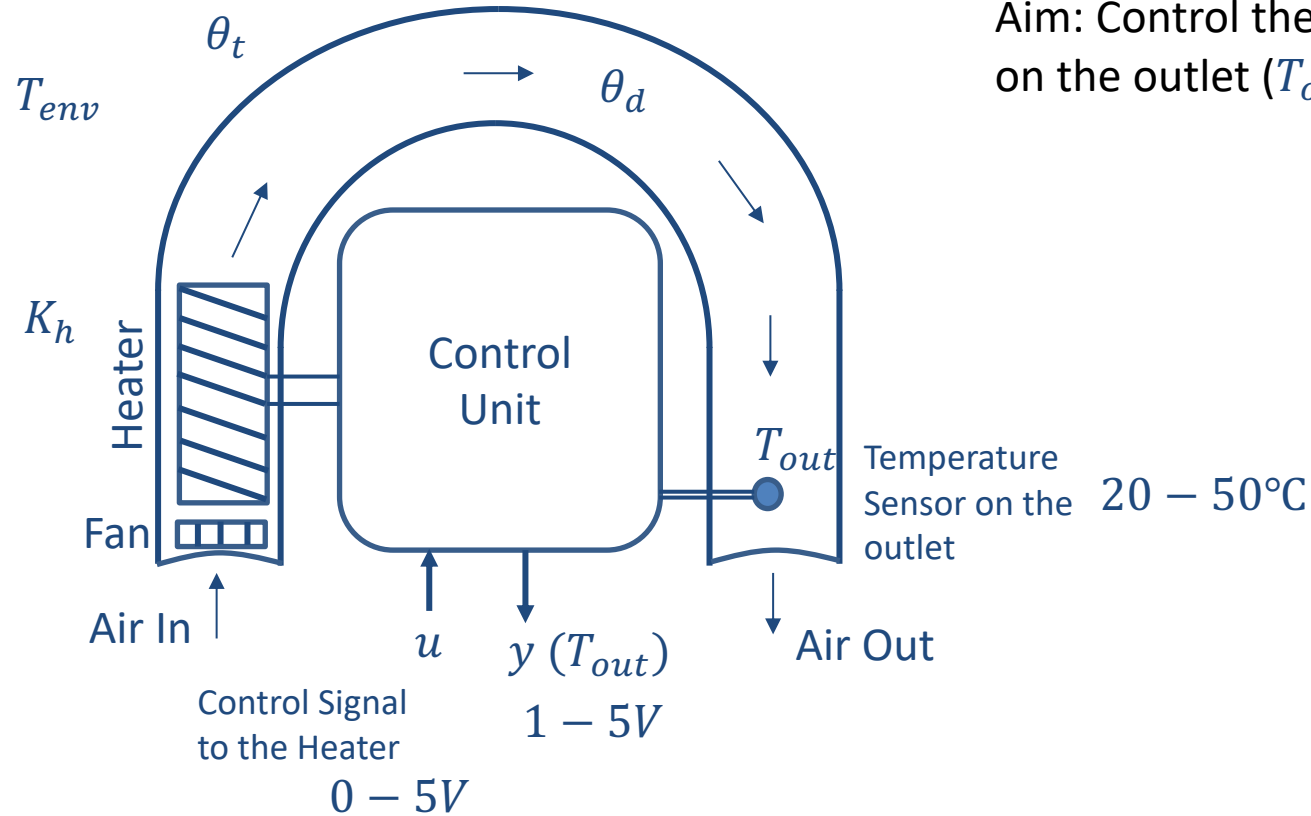
Hans-Petter Halvorsen

[Table of Contents](#)

# Air Heater



# Air Heater Overview



Aim: Control the Temperature on the outlet ( $T_{out}$ )

# Air Heater Overview

- **Heater:** The air is heated by an electrical heater. The supplied power is controlled by an external voltage signal in the range **0 - 5 V** (min power, max power).
- **Temperature sensors:** A Pt100 temperature sensor. The range is **1 - 5V** (Air Heater #1-17)/**0-5V** (Air Heater #18-32), and this voltage range corresponds with a linear relation to the temperature range **20 - 50°C** (Air Heater #1-17) or **0 - 50°C** (Air Heater #18-32).
- Example of Mathematical model for the Air Heater process:

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



# Air Heater Mathematical Model

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

$$\theta_t = 22 \text{ sec}$$

$$\theta_d = 2 \text{ sec}$$

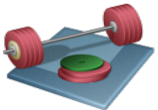
Where:

- $T_{out}$  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)

Use, e.g., these values in the Simulations:

$$K_h = 3.5 \frac{^{\circ}C}{V}$$

$$T_{env} = 21.5 \text{ }^{\circ}C$$



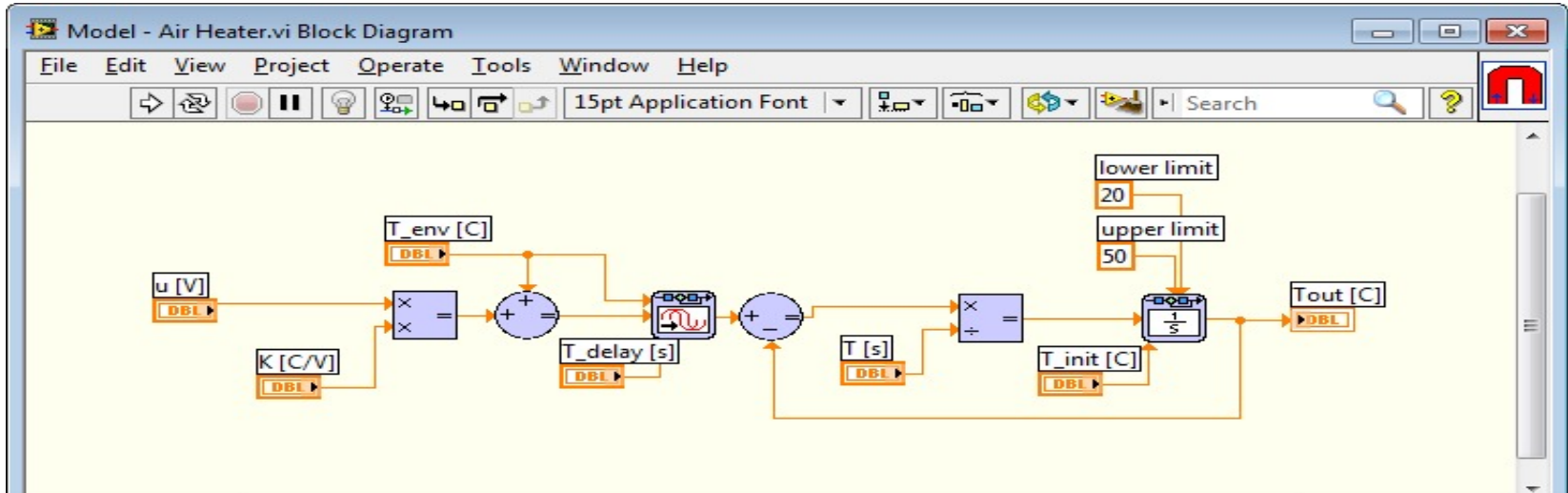
Students: Implement and Simulate a Mathematical Model of the Air Heater using LabVIEW.

# Air Heater in LabVIEW

**Heater:** The air is heated by an electrical heater. The supplied power is controlled by an external voltage signal in the range 0 - 5 V (min power, max power).

**Temperature sensors:** Two Pt100 temperature elements are available. The range is 1 - 5 V, and this voltage range corresponds to the temperature range 20 - 50°C (with a linear relation).

Example of Mathematical Model of Air Heater implemented in LabVIEW:

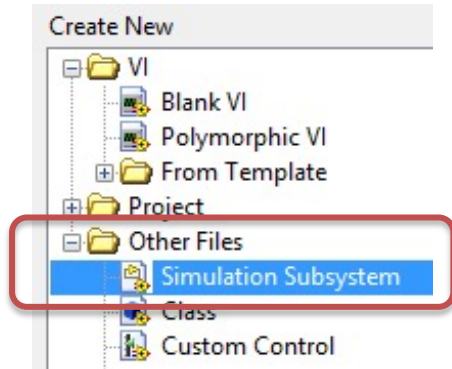


**Note!** This model is implemented in a so-called "Simulation Subsystem" (which is recommended!!!)

# Simulation Subsystem

A Way to structure your code, similar to SubVIs

This is the recommended way to do it! – You can easily reuse your Subsystems in different VIs and your code becomes more structured!



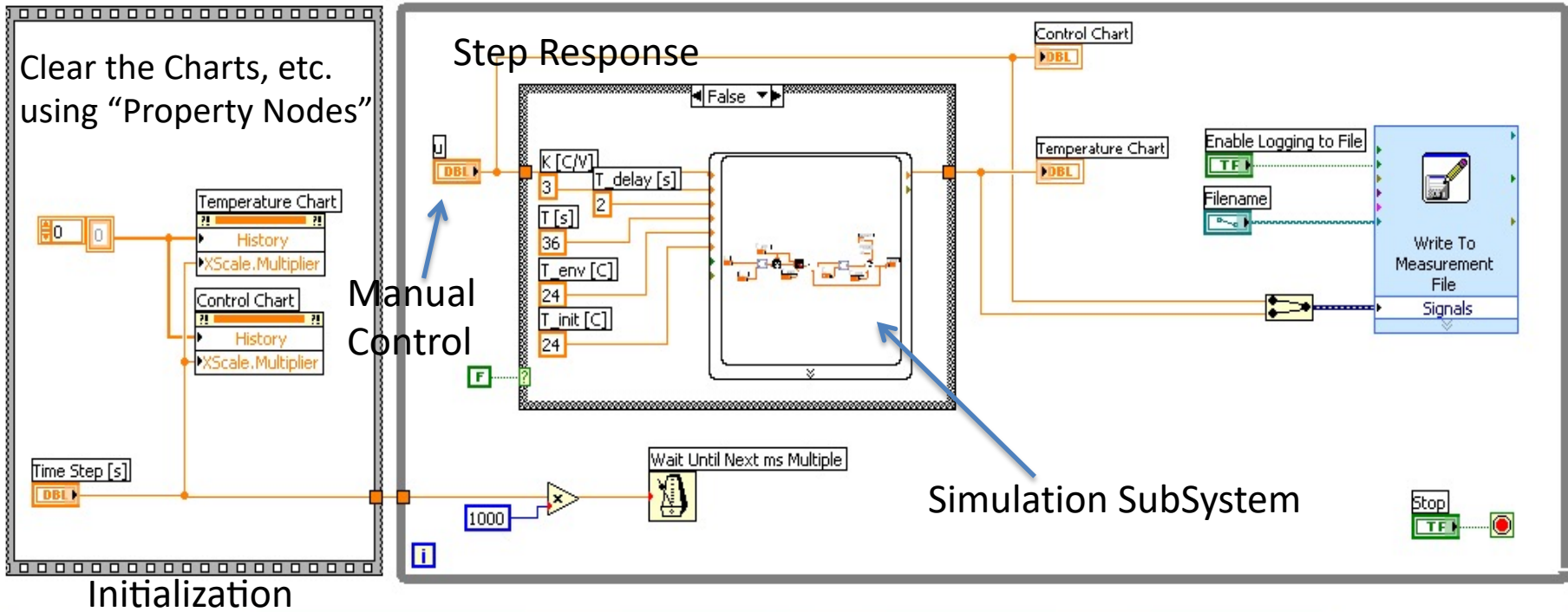
Select File -> New ..., Then choose “Simulation Subsystem”.

Create your Model within the Simulation Subsystem

# Basic Application Example



Note! This is just an Example! – You should create your own personal Application



# Finding Model Parameters using “Trial and Error”

You may use, e.g., the following Parameters as a starting point, but since every Air Heater is unique, you may want to adjust these parameters. The “Trial and Error Method” may be an easy way to find the Parameters for your Process.

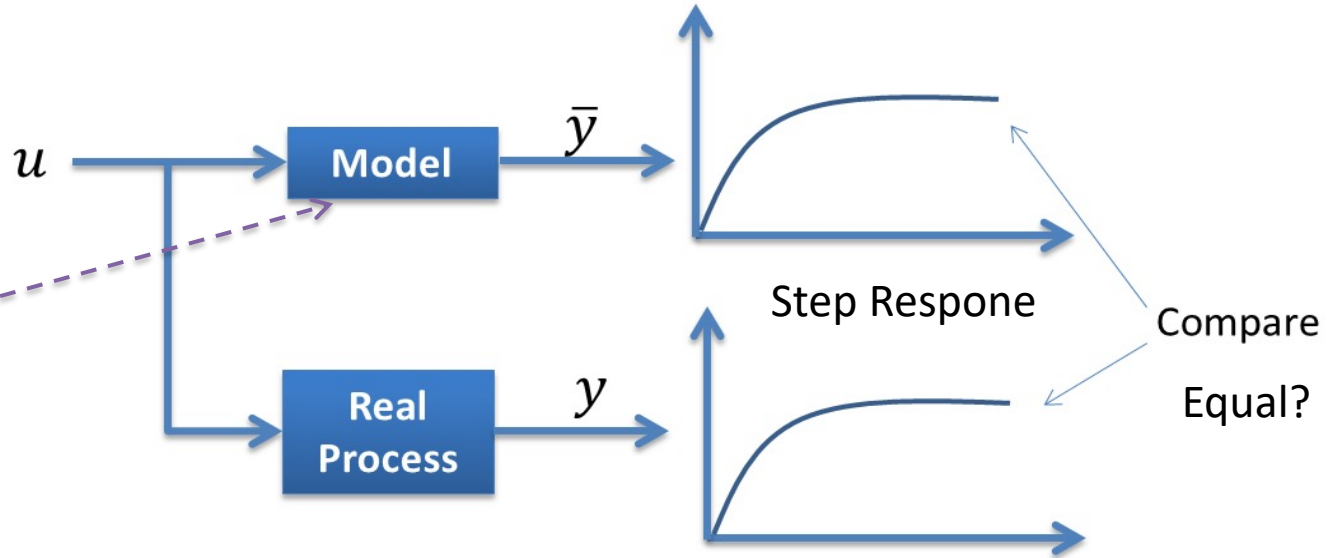
These values can be a good starting point:

$$\theta_t = 22 \text{ sec}$$

$$\theta_d = 2 \text{ sec}$$

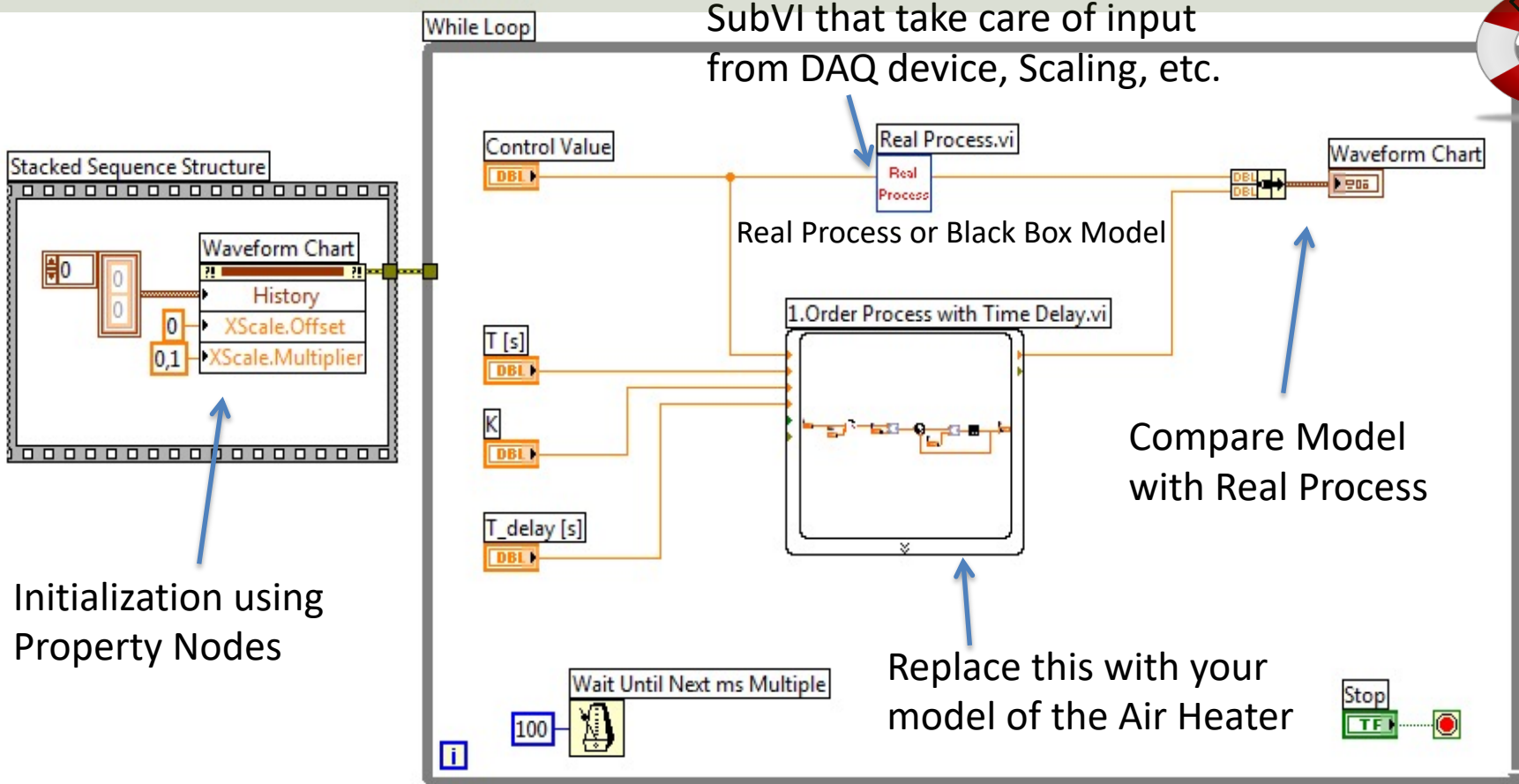
$$K_h = 3.5 \frac{^\circ\text{C}}{\text{V}}$$

$$T_{env} = 21.5 \text{ }^\circ\text{C}$$

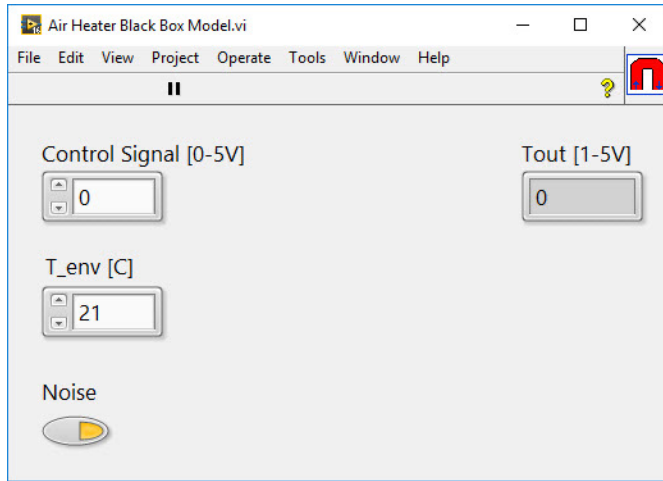


Procedure: You run the Model and the Real Process in Parallel. Adjust the Model Parameters until the output of the Model and the Real Process is “equal”.

# “Trial and Error” Example in LabVIEW



# “Air Heater Black Box Model”



For Online Students

- The Real Air Heater is only available in the Laboratory
- A “Real” Air Heater will we provided for download 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
- You can use it to find Model Parameters, etc. when you are not in the Laboratory



# LabVIEW PID Controller

Hans-Petter Halvorsen

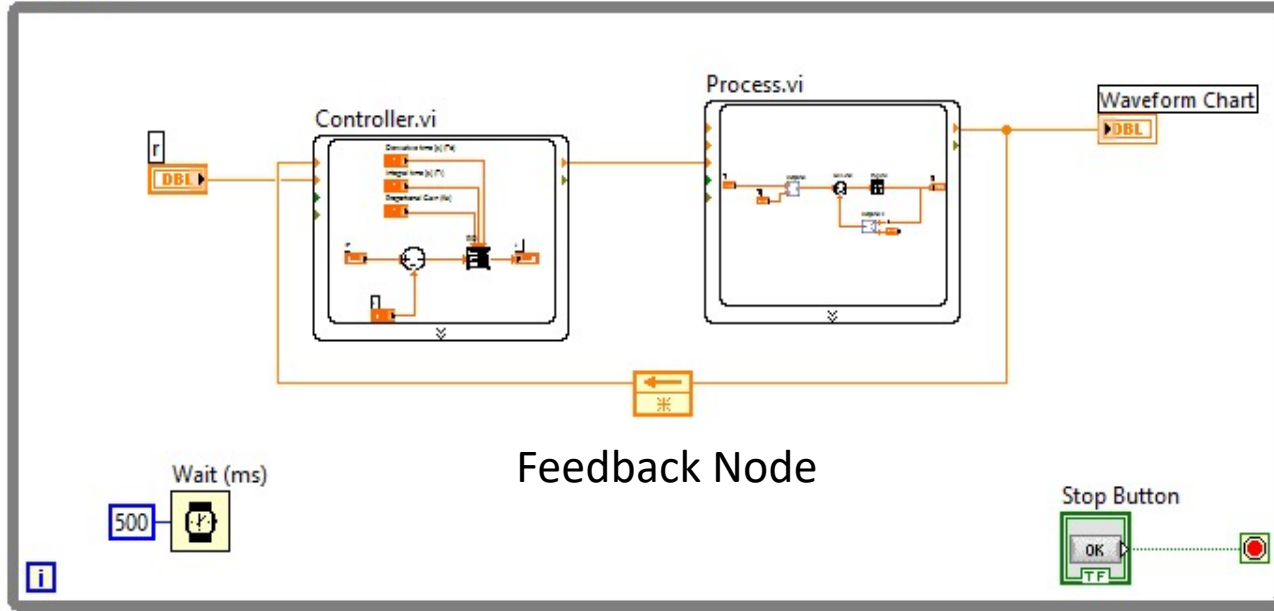
[Table of Contents](#)



# PID Control of Model in LabVIEW

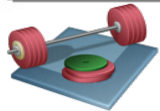
The Simulation Loop has some drawbacks/is more complicated to use than an ordinary While Loop. If we use Simulation Subsystems, we can use them inside a While Loop instead! - which becomes very handy!

While Loop

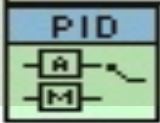


For real applications that involves more than just simulations (such as DAQ, File Logging, PID control of the real process, etc.), I recommend to use a While Loop instead of a Simulation Loop.

A State Machine approach is also recommended.

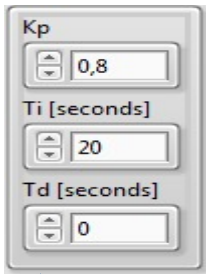


Students: Implement a Control System in LabVIEW where you use the built-in PID Controller and the Mathematical Model of the Air Heater



# LabVIEW PID Controller

Front Panel

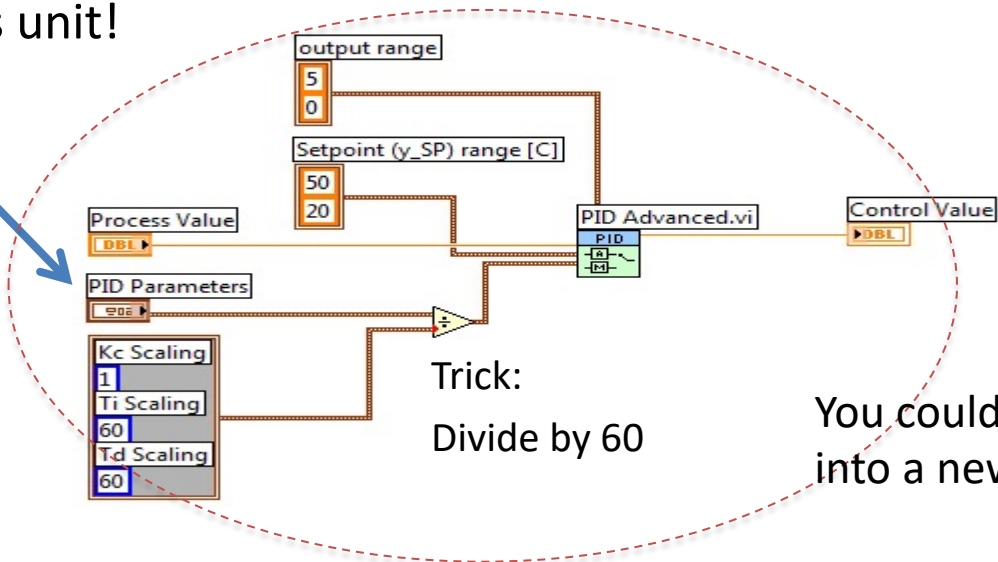


Cluster

Block Diagram:

Normally we use seconds as unit for Ti and Td (which is recommended!)

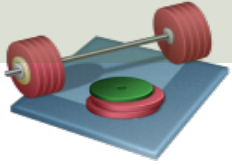
But the built-in PID algorithm in LabVIEW uses minutes as unit!



Trick:  
Divide by 60

You could also put this code into a new SubVI

# PID Parameters



- Find Proper PID Parameters for the system
- We will primarily focus on PI, but you may try PID as well
- E.g., use one or more of the following methods:
  - “Trial and Error” method
  - Skogestad’s method
  - Ziegler Nichols method
  - etc.

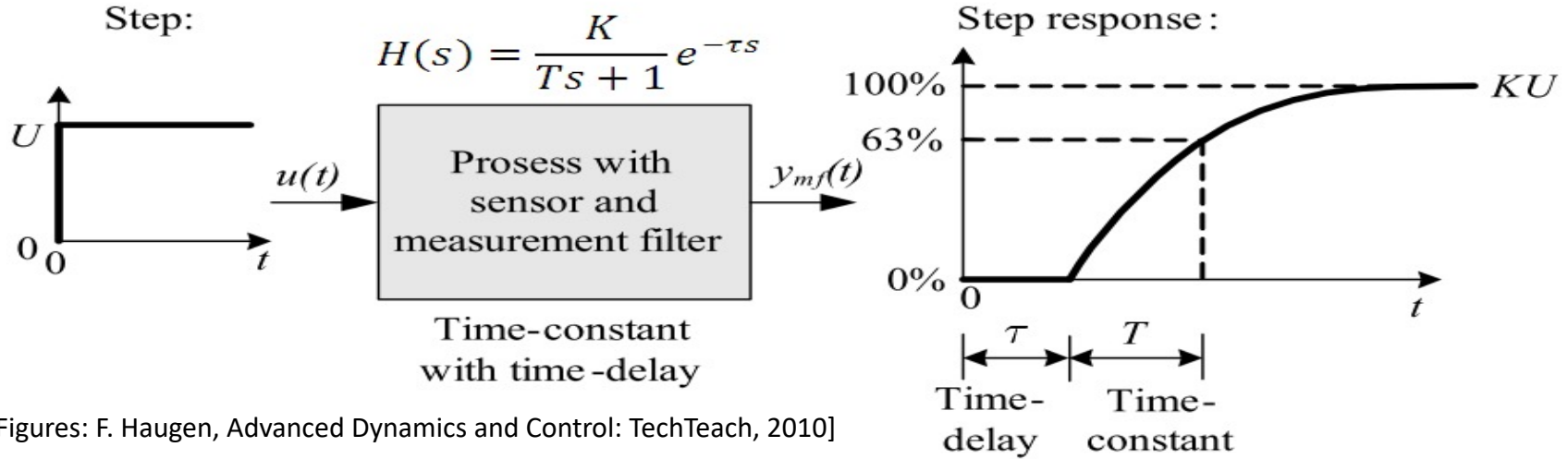
You may try these values  
as a starting point:

$$K_p = 0.8$$

$$T_i = 20s$$

$$T_d = 0s$$

# PID Tuning with Skogestad



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

We can set, e.g.,  $T_c=10$  sec. and  $c=1.5$ .

You may use other values if these values give a poor result.

| 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      | $\frac{K}{Ts+1} e^{-\tau s}$             | $\frac{1}{K(T_C + \tau)}$    | $\min [T, c(T_C + \tau)]$   | 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     | $\frac{K}{(T_1s+1)(T_2s+1)} e^{-\tau s}$ | $\frac{1}{K(T_C + \tau)}$    | $\min [T_1, c(T_C + \tau)]$ | $T_2$           |
| Double integrator + delay  | $\frac{K}{s^2} e^{-\tau s}$              | $\frac{1}{4K(T_C + \tau)^2}$ | $4(T_C + \tau)$             | $4(T_C + \tau)$ |

Table 1: Skogestad's formulas for PI(D) tuning.



# Lowpass Filter

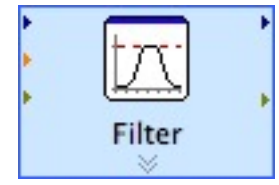
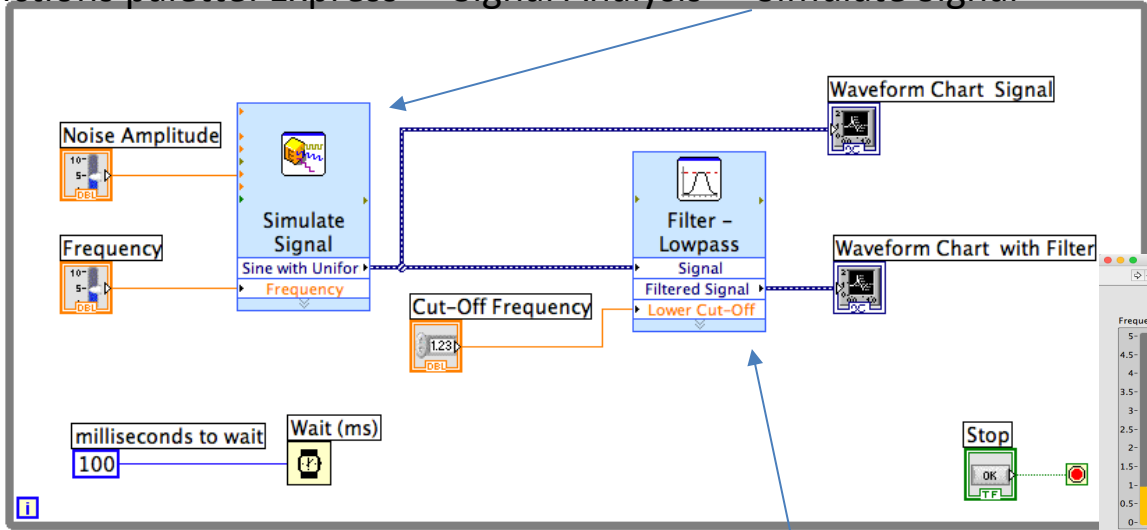
# Lowpass Filter

- You should consider using a Lowpass Filter in order to reduce Noise
- You can use a built-in Filter in LabVIEW
- Or you can create your own Lowpass Filter from scratch
- Or download a Lowpass Filter (LabVIEW SubVI) from the Web site of this Lab Assignment
- You should first Test it on the Mathematical model before you apply it on the real system (next Task)

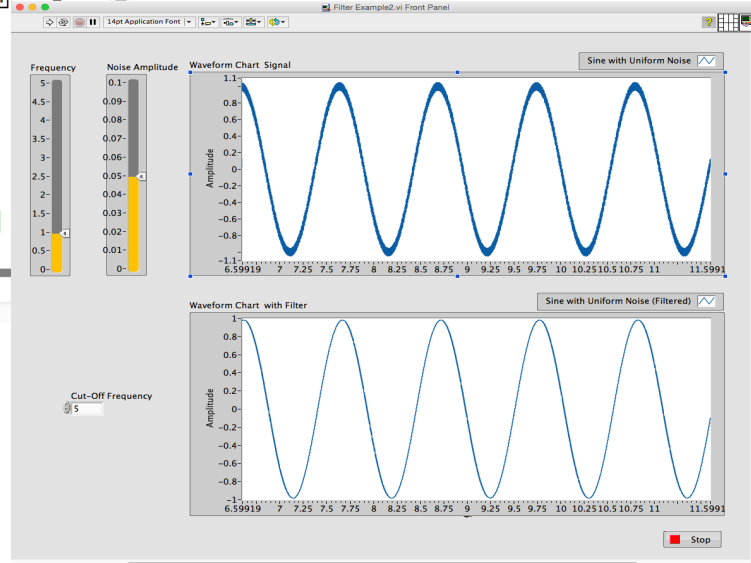
See next slides for examples

# Built-in Lowpass Filter to reduce Noise

Functions palette: Express -> Signal Analysis -> Simulate Signal



Functions palette: Express -> Signal Analysis -> Filter



# Properties

Configure Simulate Signal [simulate Signal]

**Signal**

Signal type  
Sine

Frequency (Hz) 10.3 Phase (deg) 0

Amplitude 1 Offset 0 Duty cycle (%) 50

Add noise

Noise type  
Uniform White Noise

Noise amplitude 0.6 Seed number -1 Trials 1

**Timing**

Samples per second (Hz) 20000  Simulate acquisition timing

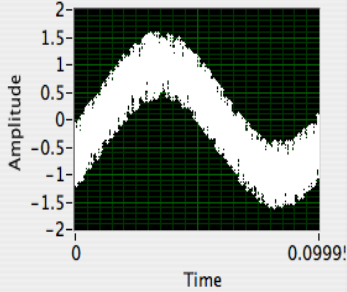
Number of samples 2000  Automatic  Run as fast as possible

Integer number of cycles

Actual number of samples 2000

Actual frequency 10.3

**Result Preview**



Amplitude

Time

**Time Stamps**

Relative to start of measurement

Absolute (date and time)

**Reset Signal**

Reset phase, seed, and time stamps

Use continuous generation

**Signal Name**

Use signal type name

Signal name  
Sine with Uniform Noise

OK Cancel Help

Configure Filter [Filter - Lowpass]

**Filtering Type**

Lowpass

**Filter Specifications**

Cutoff Frequency (Hz) 1500

High cutoff frequency (Hz) 400

Finite impulse response (FIR) filter

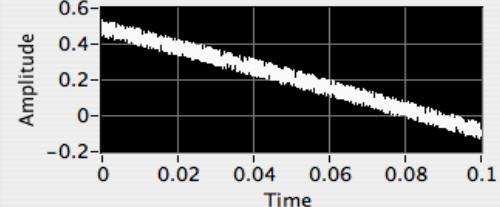
Taps 29

Infinite impulse response (IIR) filter

Topology  
Butterworth

Order 1

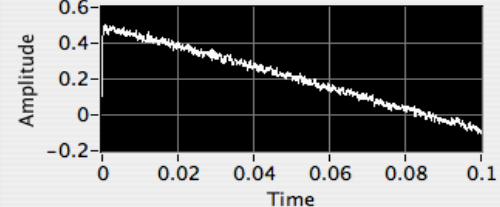
**Input Signal**



Amplitude

Time

**Result Preview**



Amplitude

Time

**View Mode**

Signals  Show as spectrum

Transfer function

**Scale Mode**

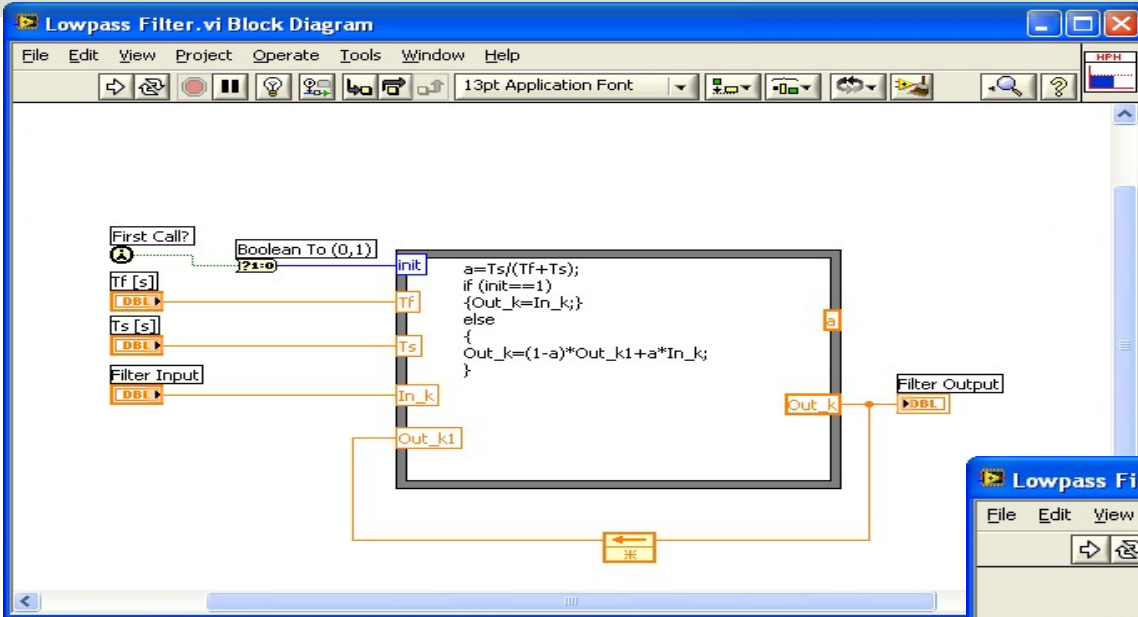
Magnitude in dB

Frequency in log

OK Cancel Help

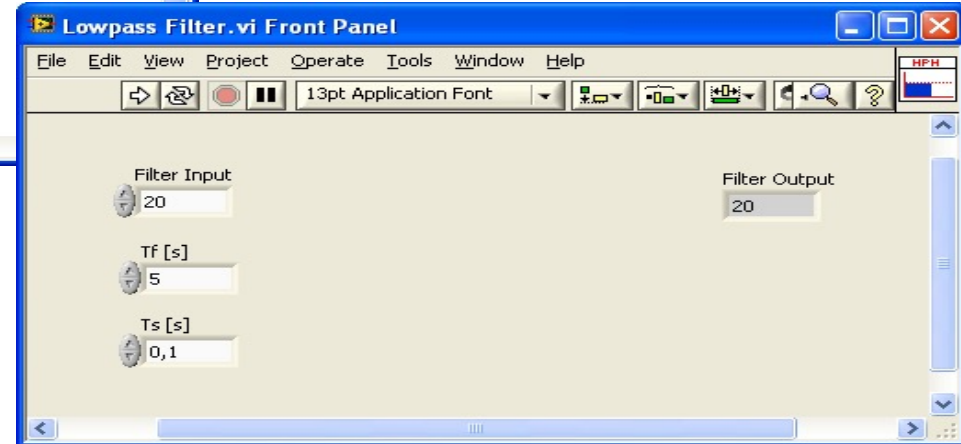


# Lowpass Filter created from Scratch



You may download this Lowpass Filter (LabVIEW SubVI) from the Web page of this Lab Assignment - or create your own from scratch

A golden rule is:  $T_s \leq \frac{T_f}{5}$



# 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:

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

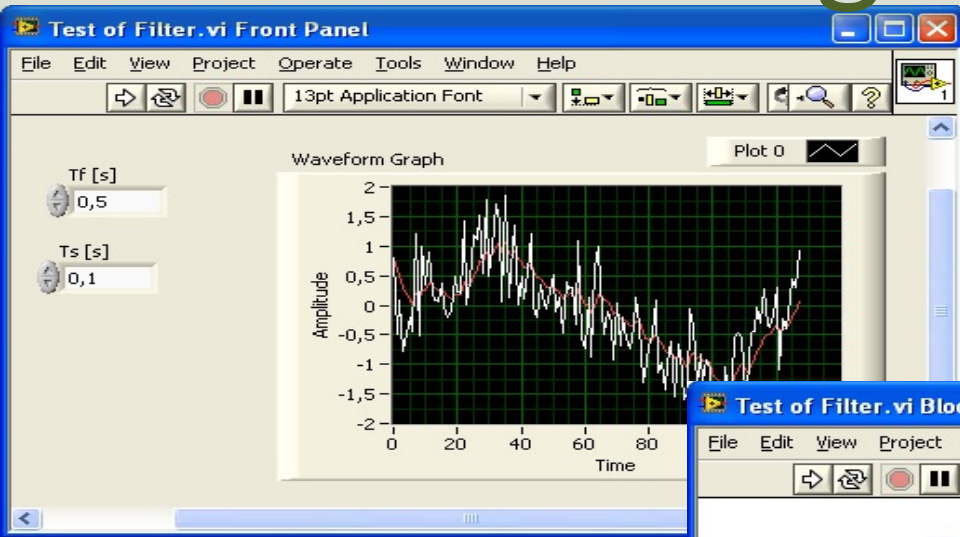
Filter output

Noisy input signal

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

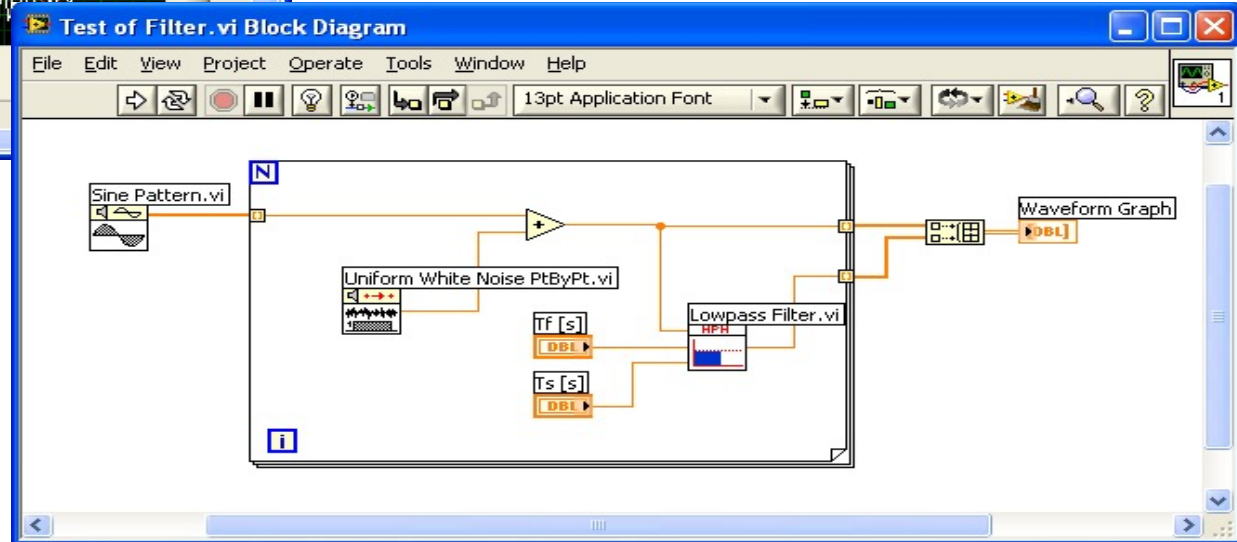
This algorithm can be easily implemented in a Programming language such as LabVIEW

# Testing the Filter

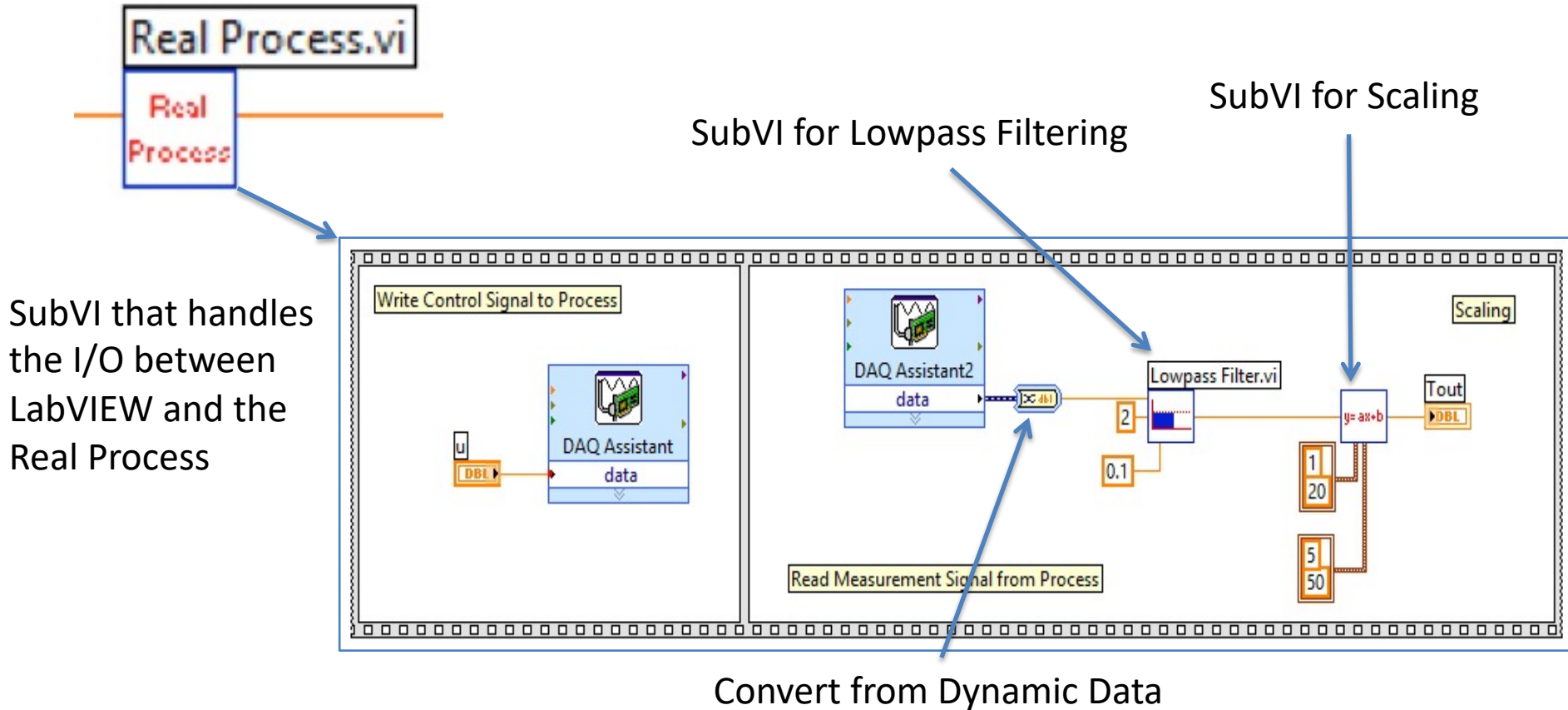


In this example we add noise to a Sine function. We then use the Measurement Filter to see if we can remove the noise afterwards.

As you can see this gives good results.  
The filter removes the noise from the signal.



# Creating and Using SubVIs



# SubVIs



Lowpass Filter.vi Front Panel

Filter Input: 20

Tf [s]: 2

Ts [s]: 0.1

Filter Output: 20

Scaling.vi Front Panel

$y = ax + b$

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

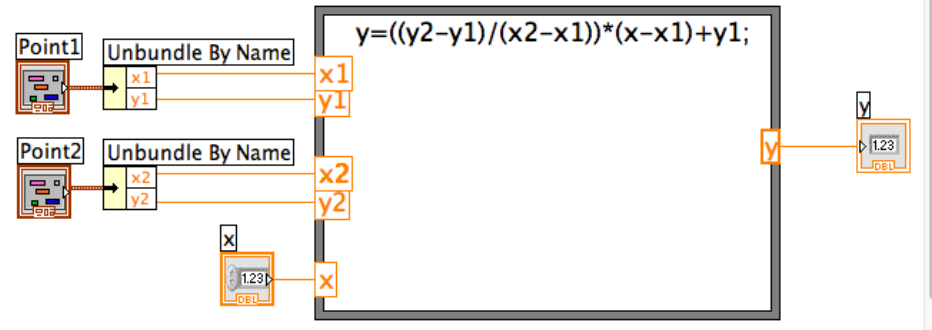
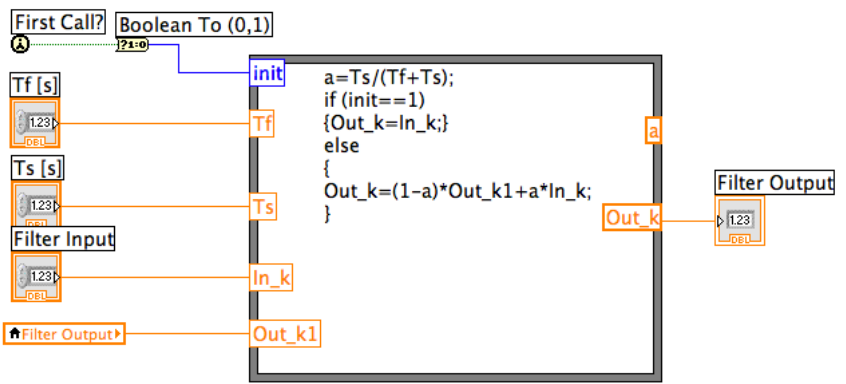
x: 0

y: 0

Point1: x1=0, y1=0

Point2: x2=0, y2=0

Hans-Petter  
hans.p.halvorsen  
Telemark University  
Department of



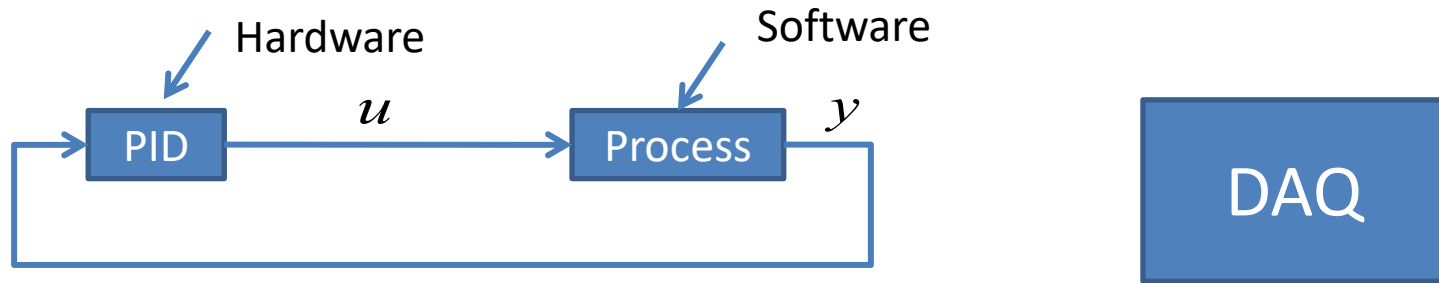


Step 2: HIL Simulation and Testing

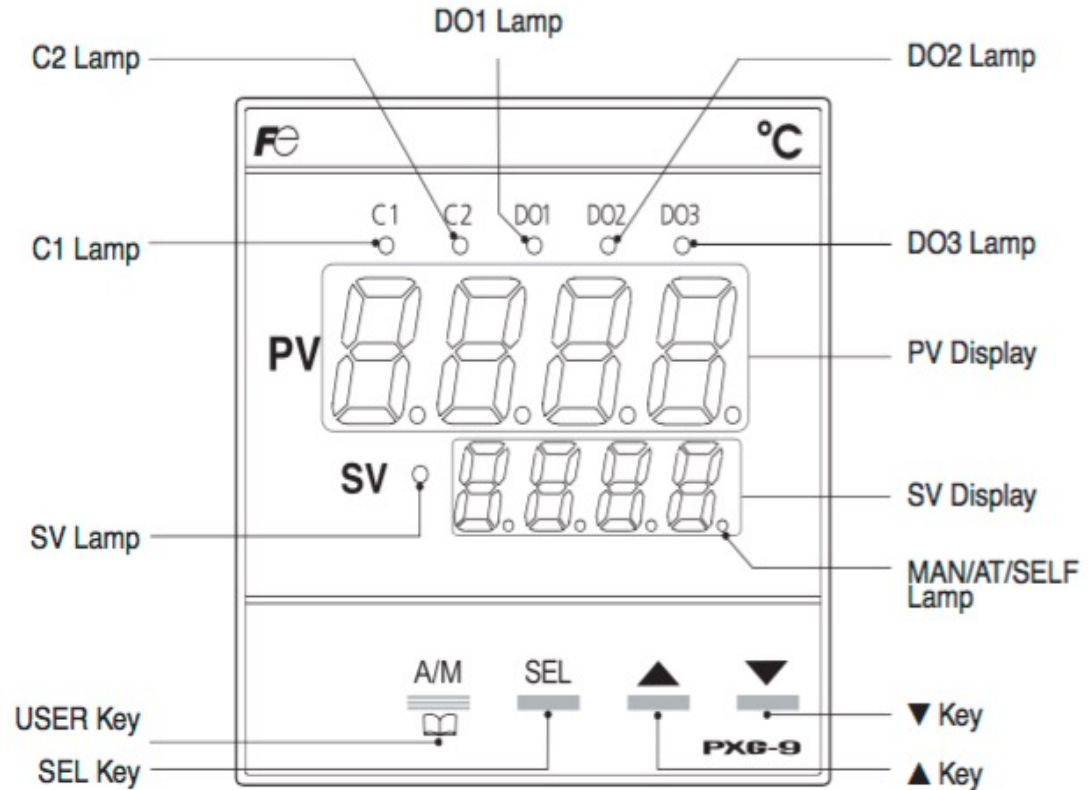
# HIL with Fuji PXG5 PID

# HIL with Fuji PXG5 PID

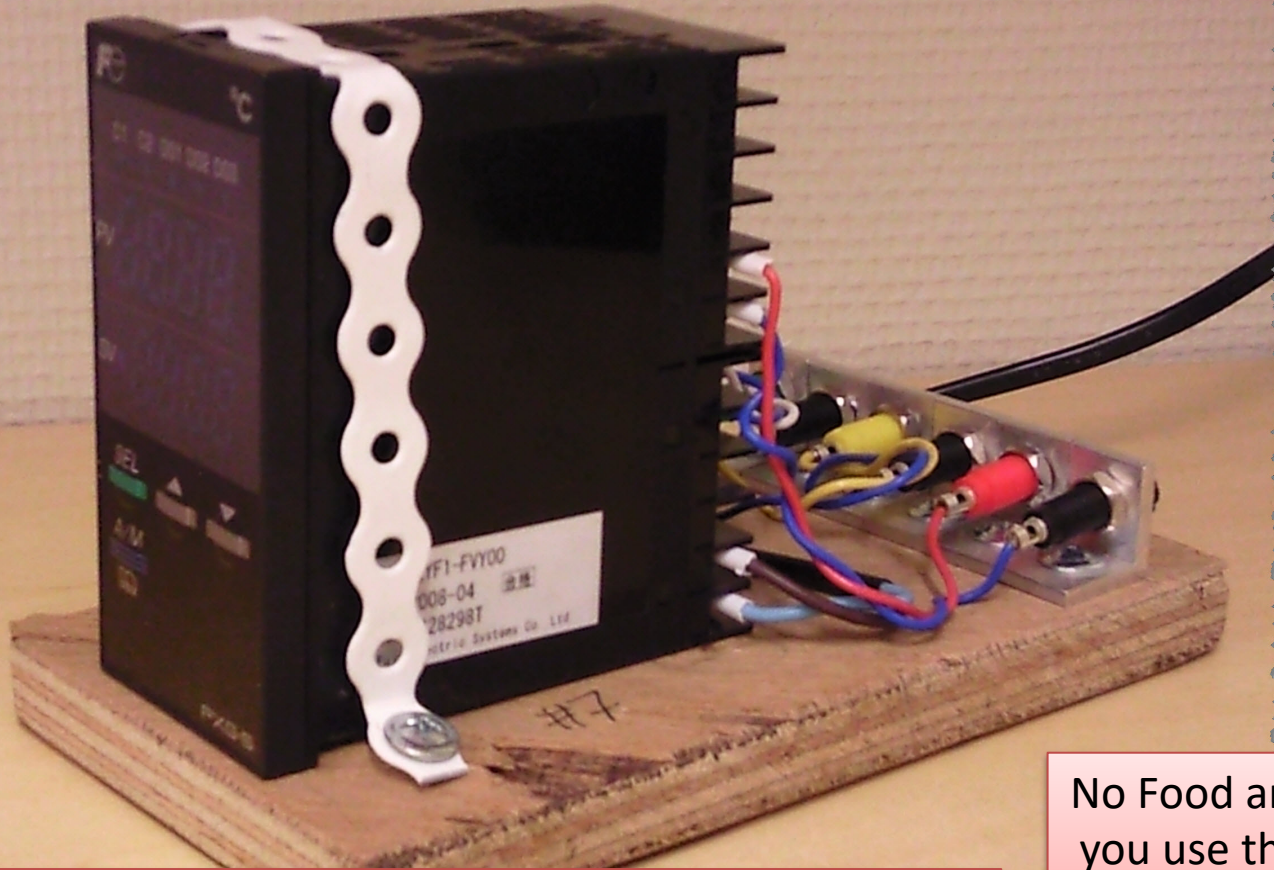
Purpose: Test your Hardware device before you apply it on your real system. Both to avoid damages, accidents, but also to tune the PID controller. It is also useful for Training purposes.



# Fuji PXG5 PID Controller







Make sure to disconnect the power cable when wiring

No Food and Drink allowed when you use the Fuji PXG5 hardware, because of risk of electric shock!

# HIL with Fuji PXG5/PXR5 PID



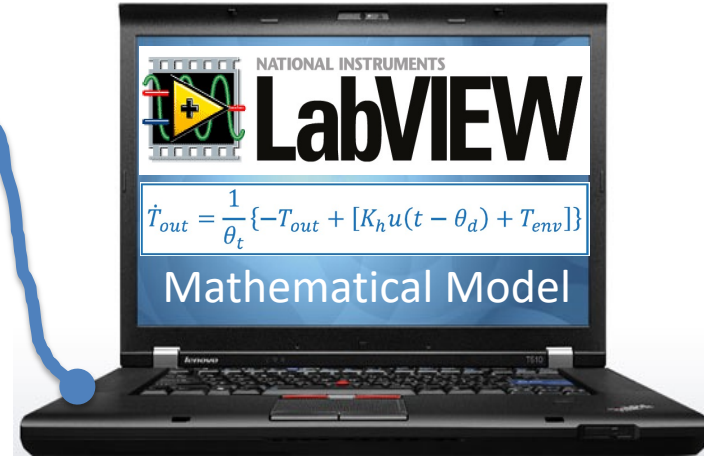
- It may be very useful to test a controller function with a simulated process before the controller is applied to the real (physical) process.
- If the mathematical model used in the simulator is an accurate representation of the real process, you may even tune the controller parameters (e.g. the PID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

# HIL Simulation Setup



PID Control

Model of Process (Air Heater)



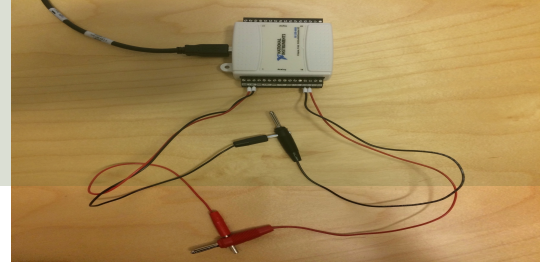
Computer (with LabVIEW)

# USB-6008 DAQ

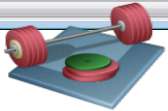
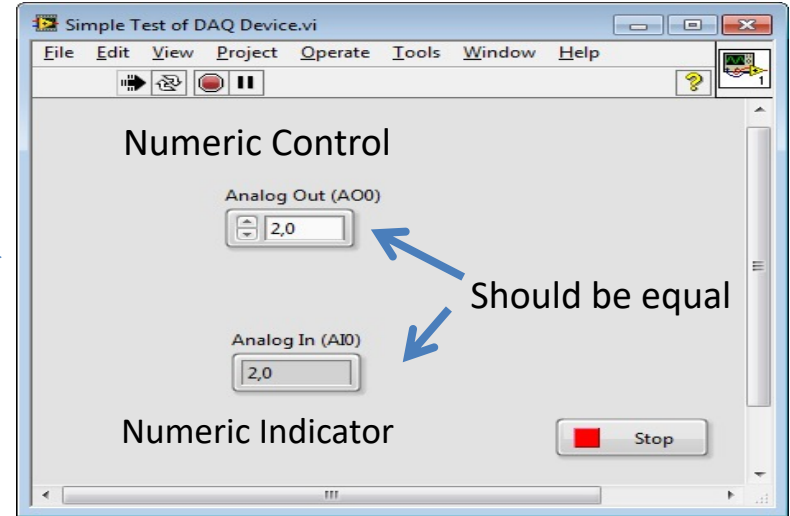
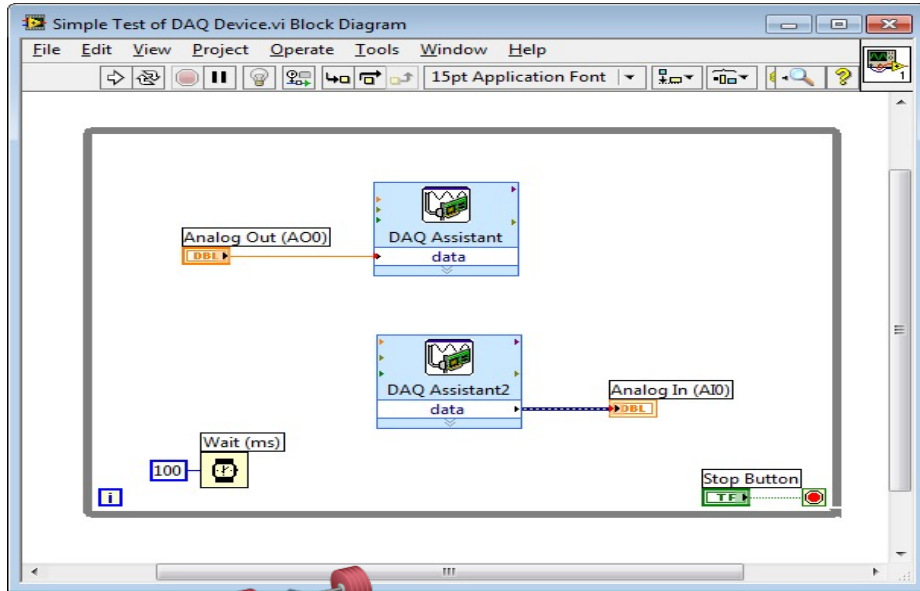
Always test your DAQ device before you use it in your application

## Loopback Test:

1. Create the simple test program as shown below.
2. Wire the AO0 and AI0 cables together.



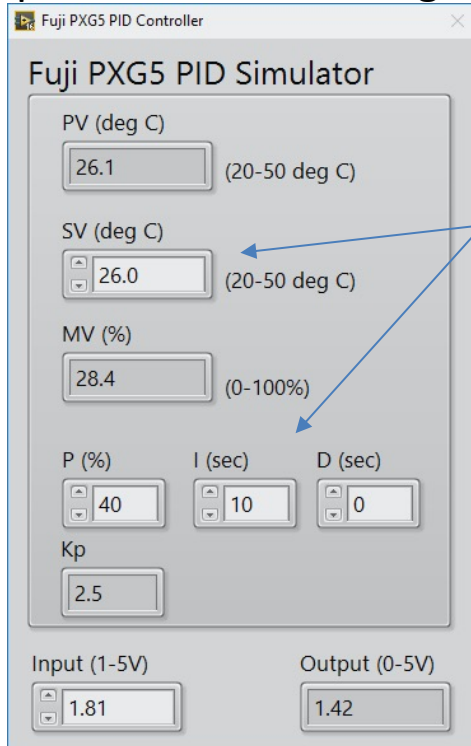
If you get the same value in the Analog Out and in the Analog In – you know your device is working properly



Students: Make sure your DAQ device works as expected

# Fuji PXG5 PID Simulator

! The Fuji PXG5 PID is only available in the Laboratory. If you work at home, you may want to use a “Fuji PXG5 PID Simulator”. You can create your own “Simulator” or download this Example from the Web Page of this Lab Work.



**Fuji PXG5 PID Simulator**

PV (deg C): 26.1 (20-50 deg C)

SV (deg C): 26.0 (20-50 deg C)

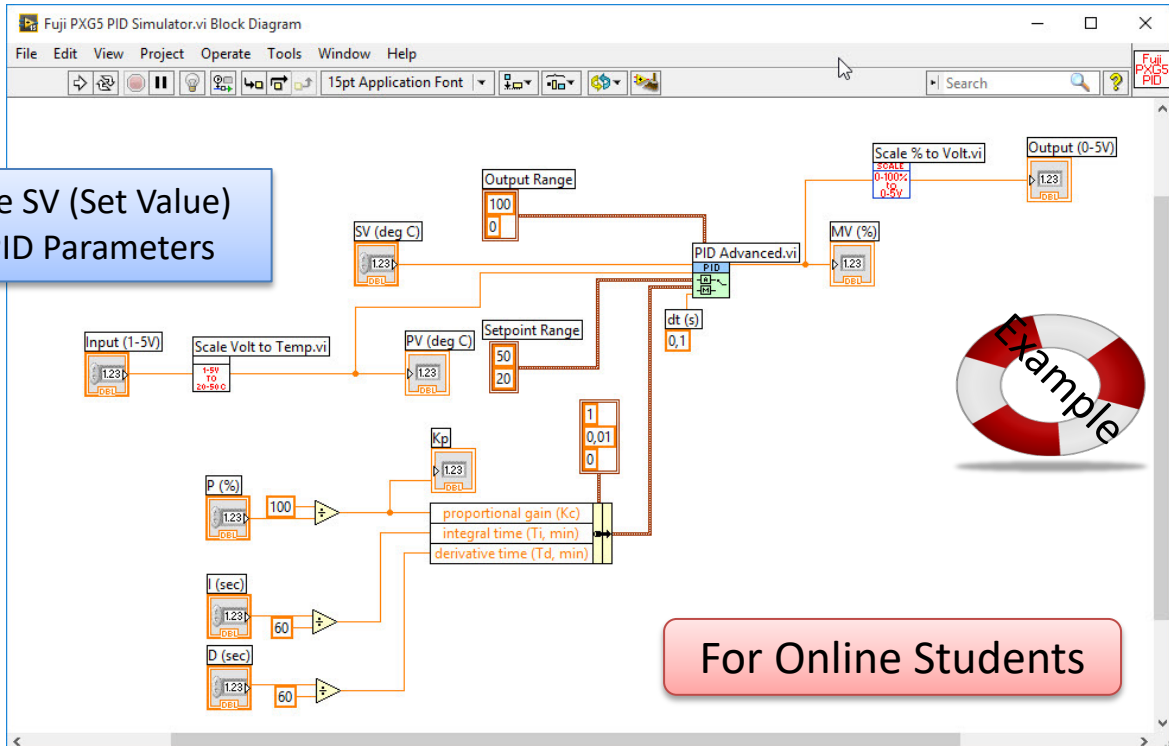
MV (%): 28.4 (0-100%)

P (%): 40    I (sec): 10    D (sec): 0

Kp: 2.5

Input (1-5V): 1.81    Output (0-5V): 1.42

Change SV (Set Value)  
and PID Parameters

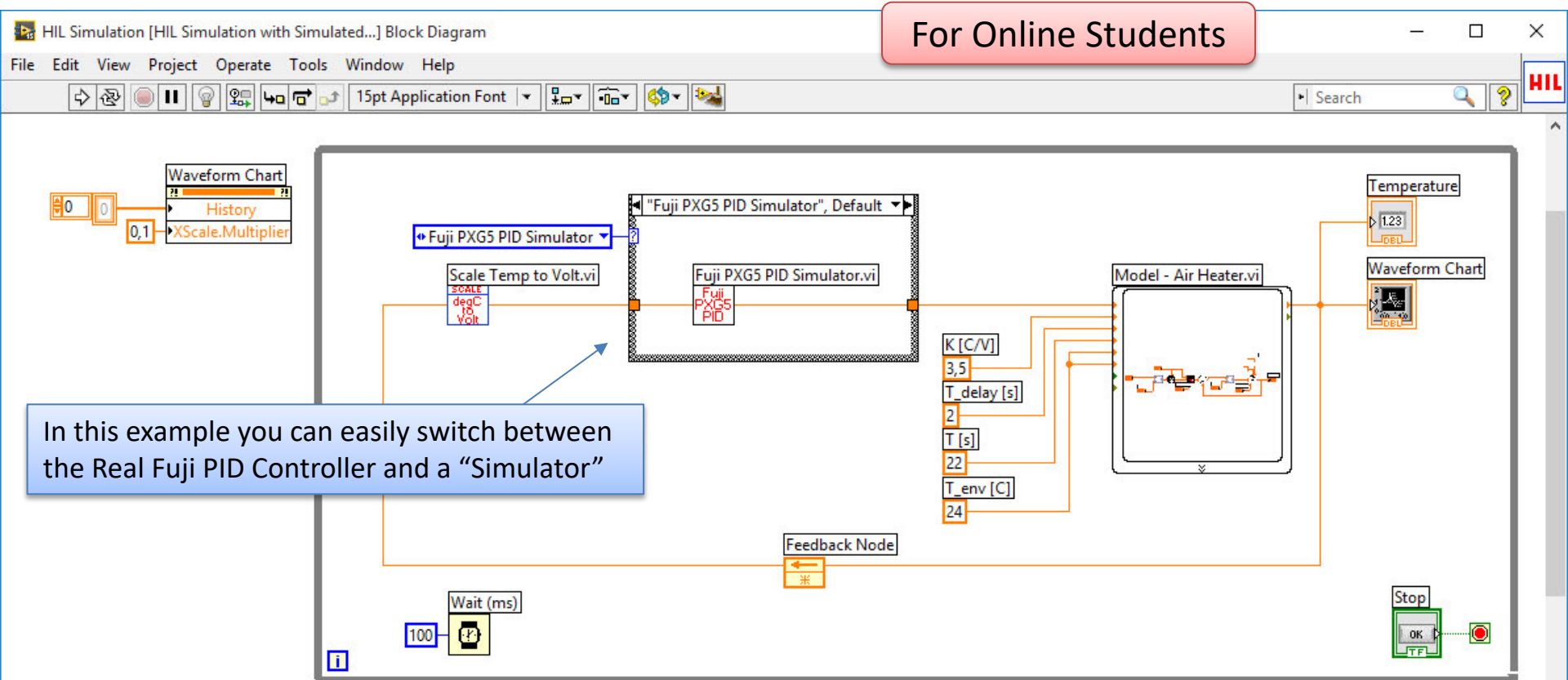


For Online Students

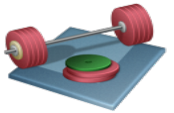


# HIL Example with Fuji PXG5 PID Simulator

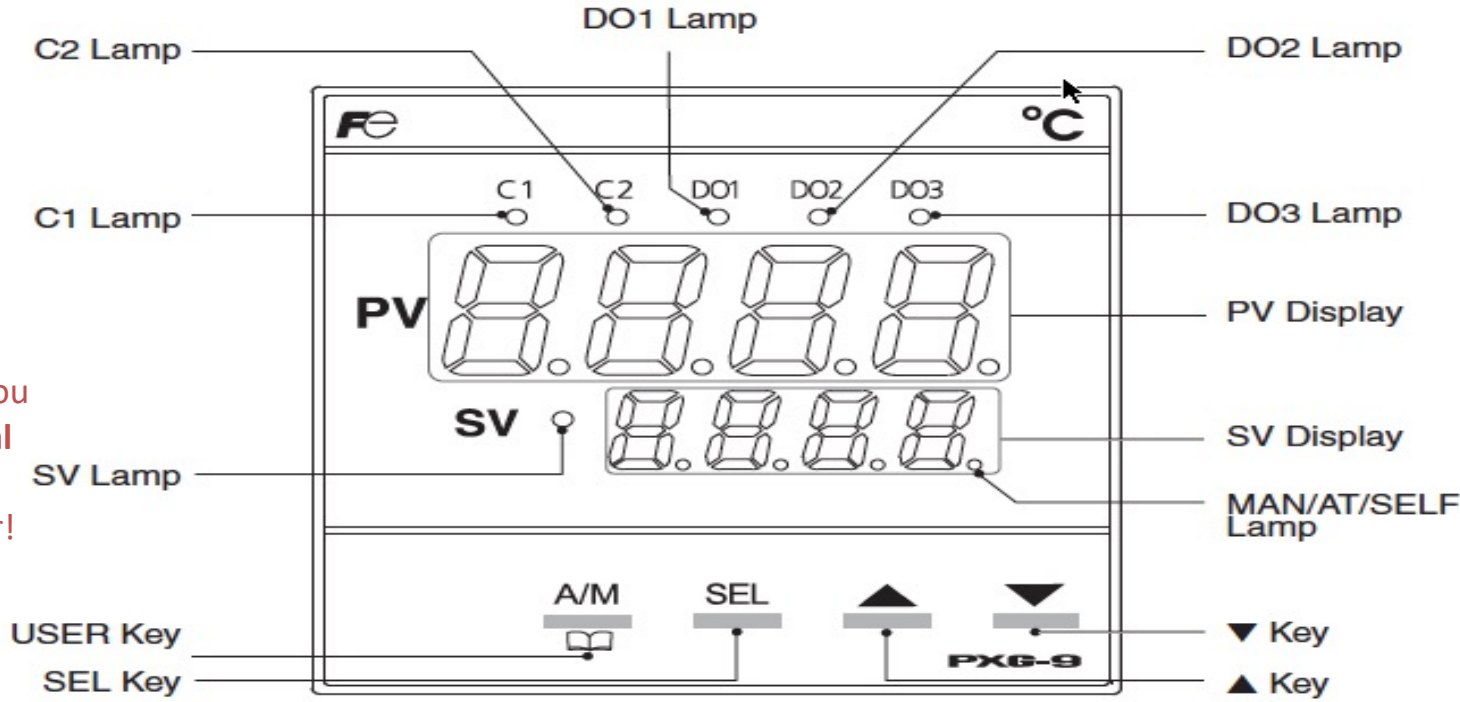
For Online Students



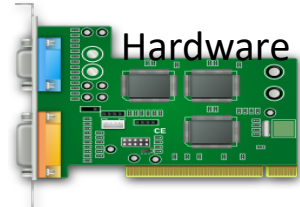
# PXG5 PID Controller



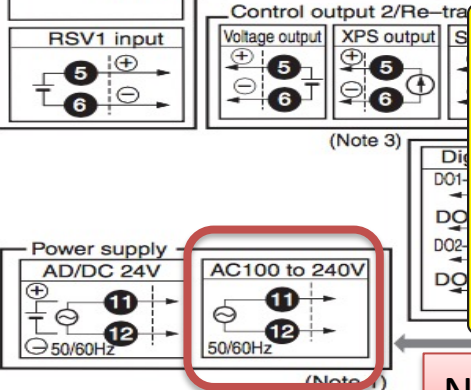
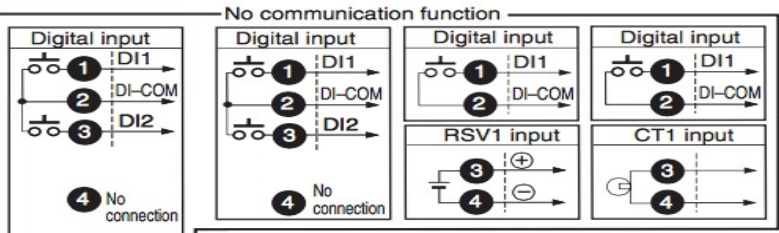
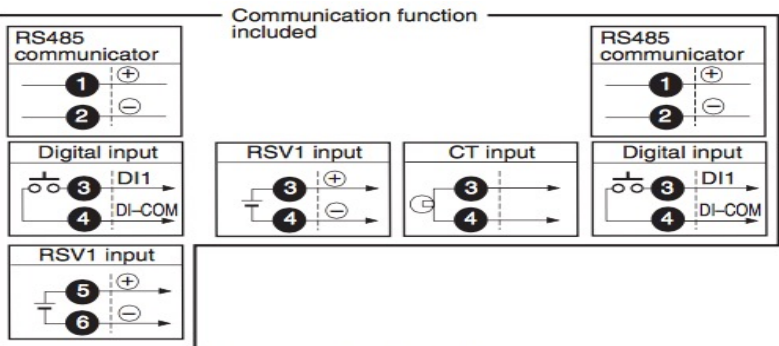
Students: It is crucial that you read the **Instruction Manual** carefully before you start using the Fuji PID Controller! Otherwise you may destroy it!



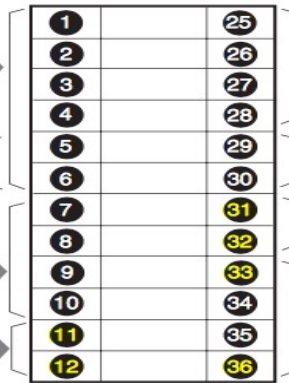
You can find the The PXG5 PID Instruction Manual googling it or on the lab assignment web page



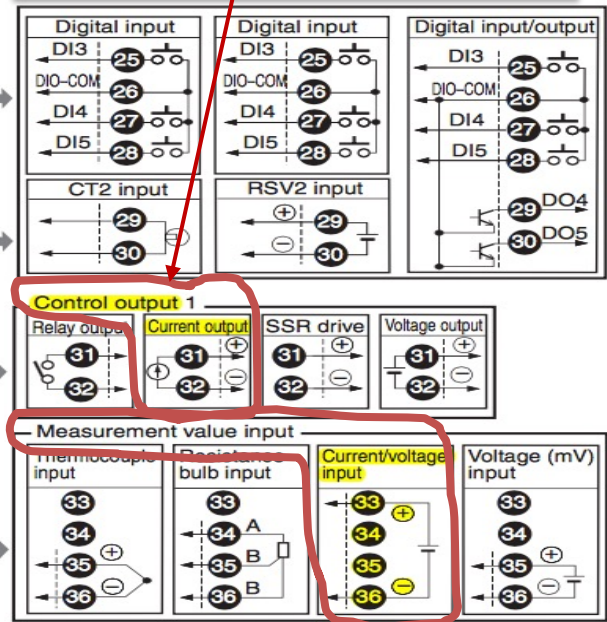
# PXC5 PID Controller Wiring Diagram



Note! 230V – Be careful!!



We use a 250Ω resistor to convert the signal from 0-20mA to 0-5V





# PXG5 PID Controller Wiring Diagram

How it works (Main Features):

## Display

### C1/C2 Lamp

Displays the condition of the control output. Lights ON at 100% output and goes out at 0% output. For values between 0% and 100%, the output is indicated by the length of time the lamp flickers. When acting as a valve control, the C1 lamp will light with OPEN output, and the C2 lamp will flicker with CLOSE output.

### DO1/2/3 Lamp

Lights ON when there is digital output is on state (DO1, DO2, DO3). The lamp flickers when delay behavior is on.

### PV Display

Displays the measurement value (PV). Displays the name of the parameter when setting parameters.

### SV Display

Displays the setting value (SV). Also can display the output value during manual mode. Displays the parameter setting value when setting parameters. Displays "rEn" during remote SV operation, and "SoFF" and set value alternately during soft start.

### SV Lamp

Lights when displaying the setting value (SV). Goes out when displaying the manual output value.

The lamp flickers while performing ramp soak or lamp SV operations.

### MAN/AT/SELF Lamp




Normally lights up during manual mode and blinks during auto-tuning or self-tuning.



For more details, please read the **PXG5 Instruction Manual!**

# PXG5 - Configuration













## How-To Change Setpoint:

- Changing SV (set values)
  - 1** Change the display to PV/SV display (shown when you turn on the power and the SV lamp is lit).
  - 2** Change the SV with the   keys.
  - 3** Press the  key to save the values.  
(The value will be automatically saved after 3 seconds even if a key is not pressed.)

## How-To Change Parameters:

### 5-5 / Setting Parameters

The following explains how to set the parameters.

- 1** Press and hold the  key in operation mode, or manual mode.  
This switches you to the monitor mode Mv1.
  - 2** Press and hold the  key in monitor mode  
This switches you to the channel menu of setup mode.
  - 3** Choose the channel with the   keys, then press and hold the  key.  
This switches you to the parameter menu.
  - 4** Choose the parameter with the   keys, then press the  key.  
The set value flickers.
  - 5** Choose the parameter with the   keys, then press the  key.  
The set value is fixed.
- No matter where you are in monitor or setup mode, pressing the  key returns you to operation mode. When setting the parameters in manual mode, pressing the key holds manual mode and returns you to operation mode.

Channel 1: Auto-tuning

Channel 2: PID Parameters

# PXG5 - Configuration

## Some recommended Channel Settings:

### Channel 1:

MA<sub>n</sub> = oFF

rEM = LoCl

AT = oFF

Closed-loop Control



### Channel 2:

SvL = 20 (Lower SV Limit)

Svh = 50 (Upper SV Limit)

You set SV locally in the Fuji PID



### Channel 6:

Pvb = 20 (Lower PV Limit)

PvF = 50 (Upper PV Limit)

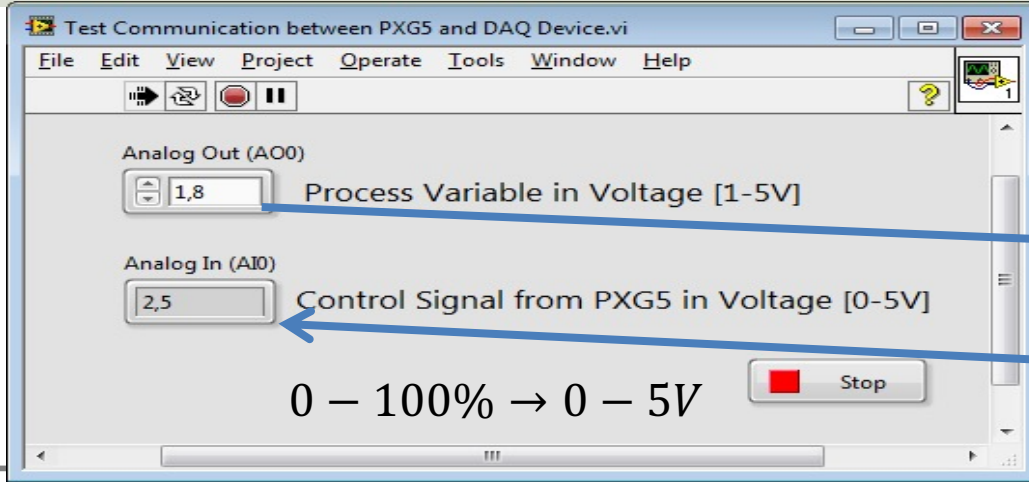
Pvd = 0 (or 1) #digits after decimal point

C1r = 0-20mA (Control output range, a 250ohm resistor is used to convert to 0-5V)

Note! The Temperature Range for most of the Air Heaters is 20 – 50°C (1 – 5V) - but some has 0 – 50°C (0 – 5V)

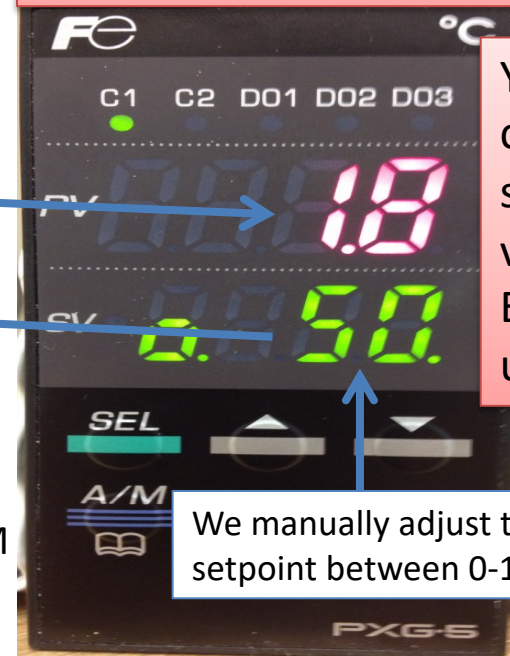
# Test Communication between PXG5 and DAQ Device

You should test the communication before you start working on the task

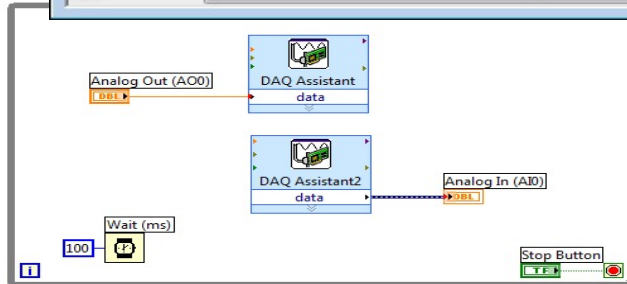


Note! Set Parameter C1r in Ch6: 0-20mA.  
A 250 Ω resistor converts the signal to 0-5V

You can choose to show voltage or Engineering units



We manually adjust the setpoint between 0-100%



Run the PXG5 in "Manual Mode" (hold down the A/M button for a few seconds)


● Changing MV (control output values)

1

Switch to manual mode.

2

Change the display to PV/MV display (MAN/AT/SELF lamp is lit).

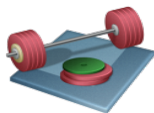
(Pressing the  key in manual mode toggles between PV/SV display and PV/MV display.)

3

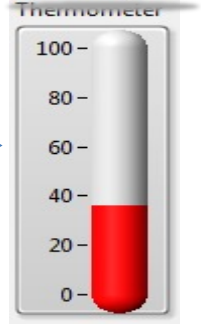
Change the MV with the   keys.

(Changes are reflected to the MV as it is changed.)

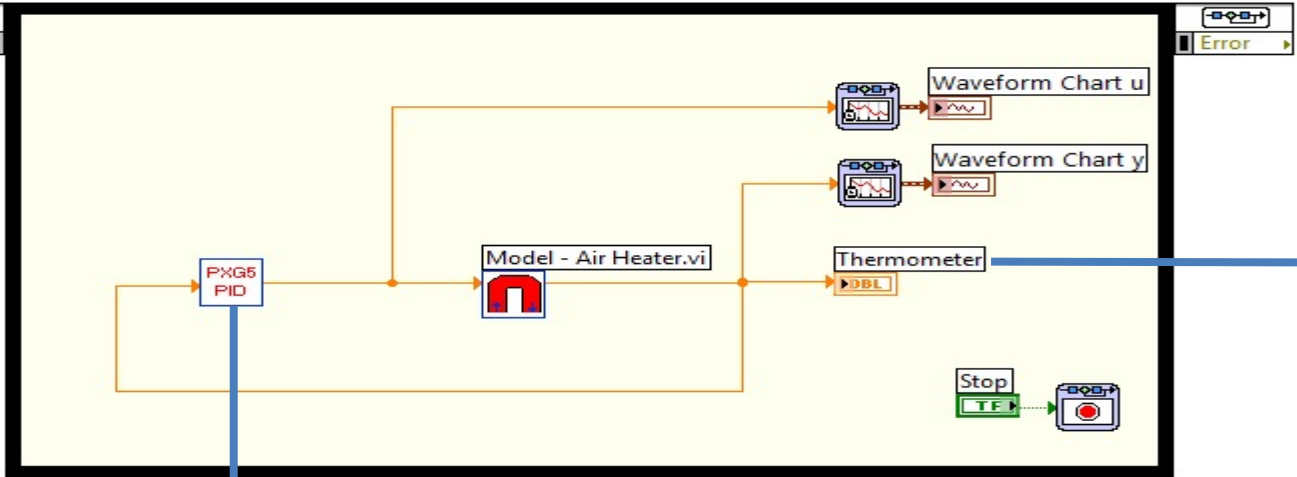
Students: It is recommended that you create such a VI in order to test the communication between LabVIEW and PXG5 PID



# HIL Simulation in LabVIEW - Example



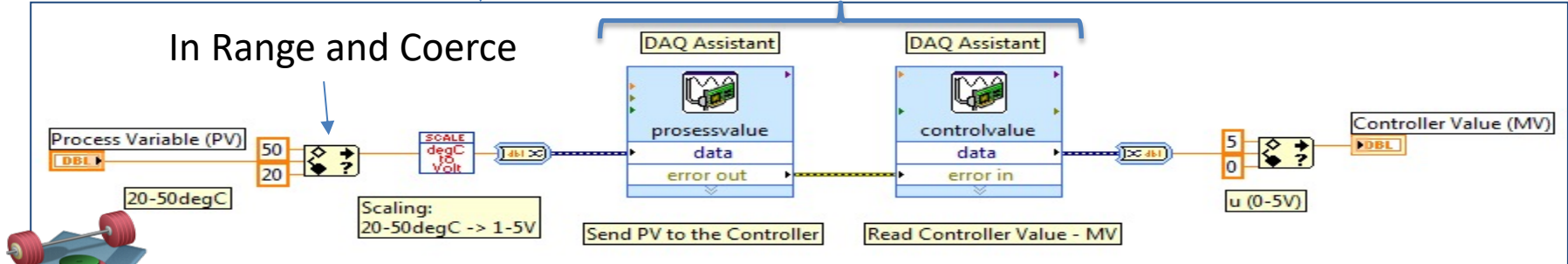
You can use a Simulation Loop or a While Loop. A While loop is recommended.



SubVI:

Note! This is opposite of what you normally do

In Range and Coerce



Students: Perform a HIL simulation using PXG5 and LabVIEW



# Setting PID Parameters on the PXG5

## 6-2 / PID (Ch2)

Note! PXG5 uses  
Proportional Band

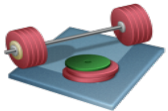
Sets parameters for controls such as PID.

| Parameter display symbol | Parameter name    | Function                                                                                          | Setting range    | Initial value | Remarks |
|--------------------------|-------------------|---------------------------------------------------------------------------------------------------|------------------|---------------|---------|
| "P" (P)                  | Proportional band | Sets the proportional band of the PID parameter. Setting "0.0" will turn it to an ON/OFF control. | 0.0 to 999.9%    | 5.0%          |         |
| "i" (i)                  | Integration time  | Sets the integration time of the PID parameter. Setting "0" will turn off integration.            | 0 to 3200 sec    | 240 sec       |         |
| "d" (d)                  | Differential time | Sets the differential time of the PID parameter. Setting "0.0" will turn off derivation.          | 0.0 to 999.9 sec | 60.0 sec      |         |

$$PB = \frac{100\%}{K_p} \Leftrightarrow K_p = \frac{100\%}{PB}$$

Example:

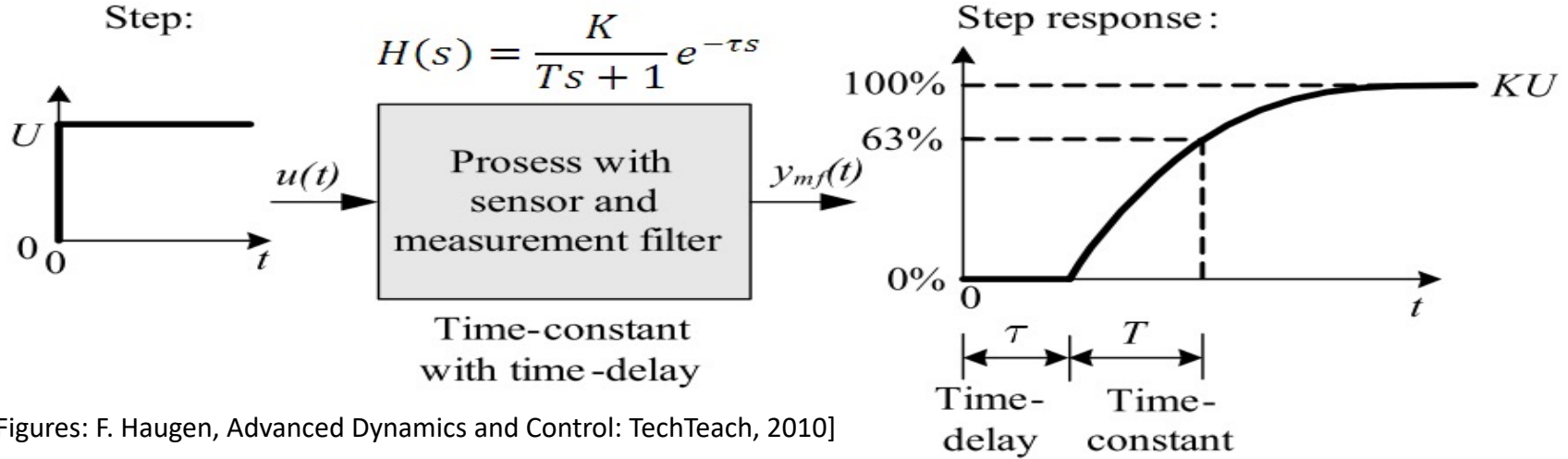
$$K_p = 0.8 \rightarrow PB = \frac{100\%}{K_p} = \frac{100\%}{0.8} = 125\%$$



Students: Find proper PID parameters using, e.g., Skogestad, Ziegler Nichols, etc.



# PID Tuning with Skogestad



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

We can set, e.g.,  $T_c=10$  sec. and  $c=1.5$ .

You may use other values if these values give a poor result.

| 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      | $\frac{K}{Ts+1} e^{-\tau s}$             | $\frac{1}{K(T_C + \tau)}$    | $\min [T, c(T_C + \tau)]$   | 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     | $\frac{K}{(T_1s+1)(T_2s+1)} e^{-\tau s}$ | $\frac{1}{K(T_C + \tau)}$    | $\min [T_1, c(T_C + \tau)]$ | $T_2$           |
| Double integrator + delay  | $\frac{K}{s^2} e^{-\tau s}$              | $\frac{1}{4K(T_C + \tau)^2}$ | $4(T_C + \tau)$             | $4(T_C + \tau)$ |

Table 1: Skogestad's formulas for PI(D) tuning.

# PXG5 – Auto-tuning

## 6-1 / Operation (Ch1)

A lamp is blinking when the auto-tuning is running

The following is a menu to operate the controller. Switchover between auto and manual control output, switchover between RUN and standby, and other such functions.

| Parameter display symbol | Parameter name                                   | Function                                              | Setting range                                             | Initial value | Remarks                                                  |
|--------------------------|--------------------------------------------------|-------------------------------------------------------|-----------------------------------------------------------|---------------|----------------------------------------------------------|
| "MAn" (MAn)              | Switchover between auto and manual mode          | Switchover between auto and manual modes              | oFF (auto) / on (manual)                                  | oFF           |                                                          |
| "STby" (STby)            | Switchover between RUN and standby               | Switchover the operation mode between RUN and standby | oFF (RUN) / on (standby)                                  | oFF           |                                                          |
| "rEM" (rEM)              | Switchover between local and remote SV operation | Switchover between local and remote SV operation      | LoCL (local) / rEM (remote)                               | LoCL          | (Note1)                                                  |
| "PrG" (PrG)              | Ramp soak control command                        | Changes ramp soak run states                          | oFF (stop)<br>rUn (run)<br>hLd (hold)                     | oFF           | Displays End (when ending) or GS (during guaranty soak). |
| "AT" (AT)                | Auto-tuning run command                          | Runs auto-tuning.                                     | oFF (stop/finish)<br>on (normal type)<br>Lo (low PV type) | oFF           |                                                          |


- Changing MV (control output values)

1

Switch to manual mode.

2

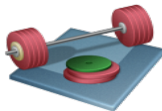
Change the display to PV/MV display (MAN/AT/SELF lamp is lit).

(Pressing the  key in manual mode toggles between PV/SV display and PV/MV display.)

3

Change the MV with the   keys.

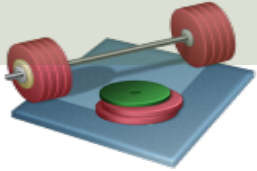
(Changes are reflected to the MV as it is changed.)



Students: Test the Auto-tuning functionality built into the PXG5 PID. Do you get the same results?



# PXG5 – Auto-tuning



## Students:

- **Execute auto-tuning:** What is the resulting P-, I- and D-values? What is the value of the controller gain ( $K_p$ ) that corresponds to the P-value from the auto-tuning?
- **Is the stability of the Control System OK?** (excite with a step in the Set-point/Reference Signal)
- **Apply a step change in the reference:** What is the steady-state control error?
- **Reverse vs. Direct Action:** What happens to the stability of the control system if the controller mode is changed from Reverse action to Direct action? (In the controller manual, Direct action is denoted Normal mode.)
- Fine-tune the PID parameters if necessary.



Step 3: Running Hardware with the Real System

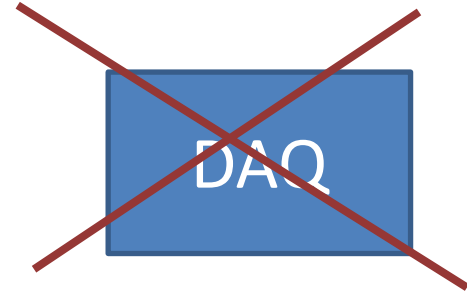
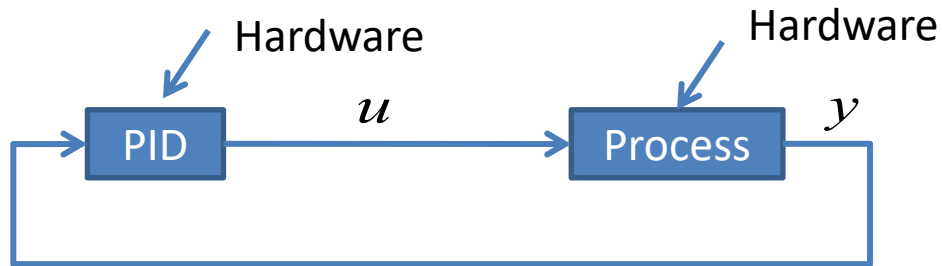
# Fuji PXG5 + Real Air Heater

Hans-Petter Halvorsen

[Table of Contents](#)

# Fuji PXG5 + Real Air Heater

Purpose: Apply your Hardware in the Production Environment



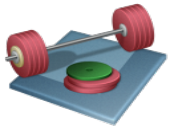
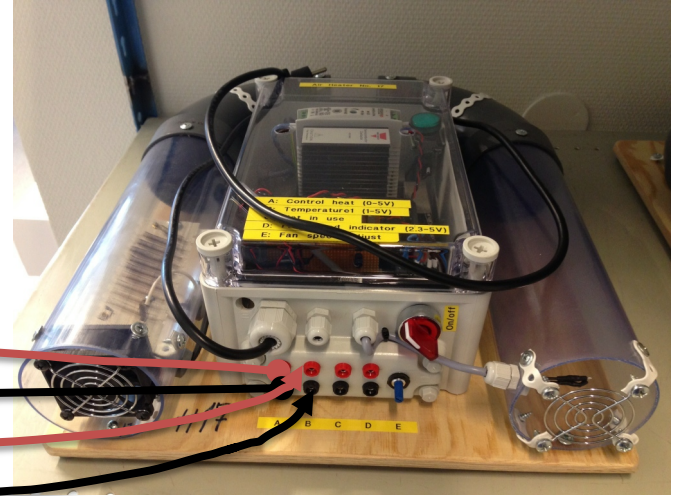
# Fuji PXG5 + Real Air Heater

Industrial PID Controller



Process Value  
1-5V

Control Signal  
0-5V



Students: Test the PXG5 PID Controller on the real Air Heater. Are you able to use the same PID settings you found using the Model? Test also the Auto-tuning functionality. Do you get the same parameters as using the model? Which PID parameters are best?

# When you are not in the Laboratory

For Online Students

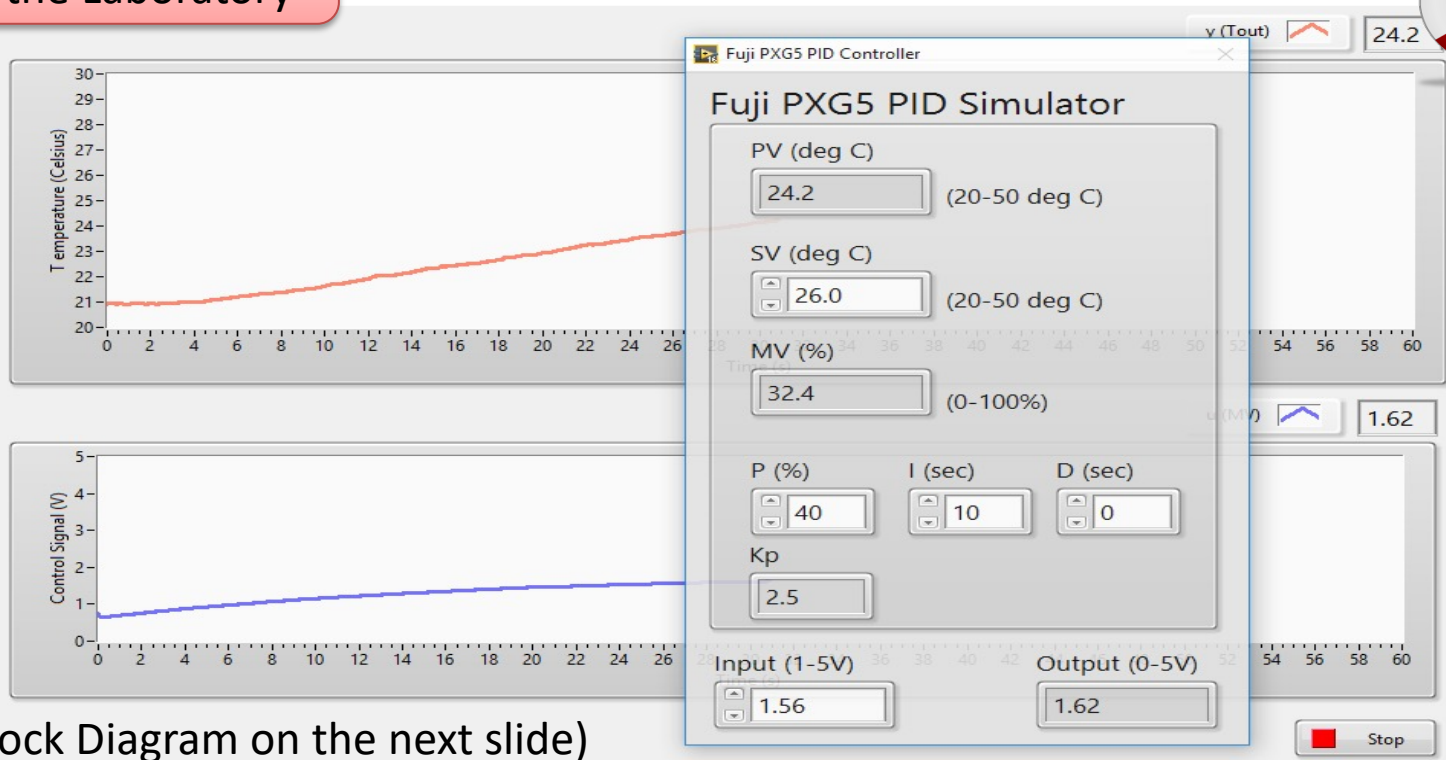
For Online Students and when you are not on the Laboratory, you can use:

- “Fuji PXG5 PID Simulator”
- “Air Heater Black Box Model”

These can be downloaded from the website for this assignment

# “Fuji PXG5 PID Simulator” + “Air Heater Black Box Model” Example

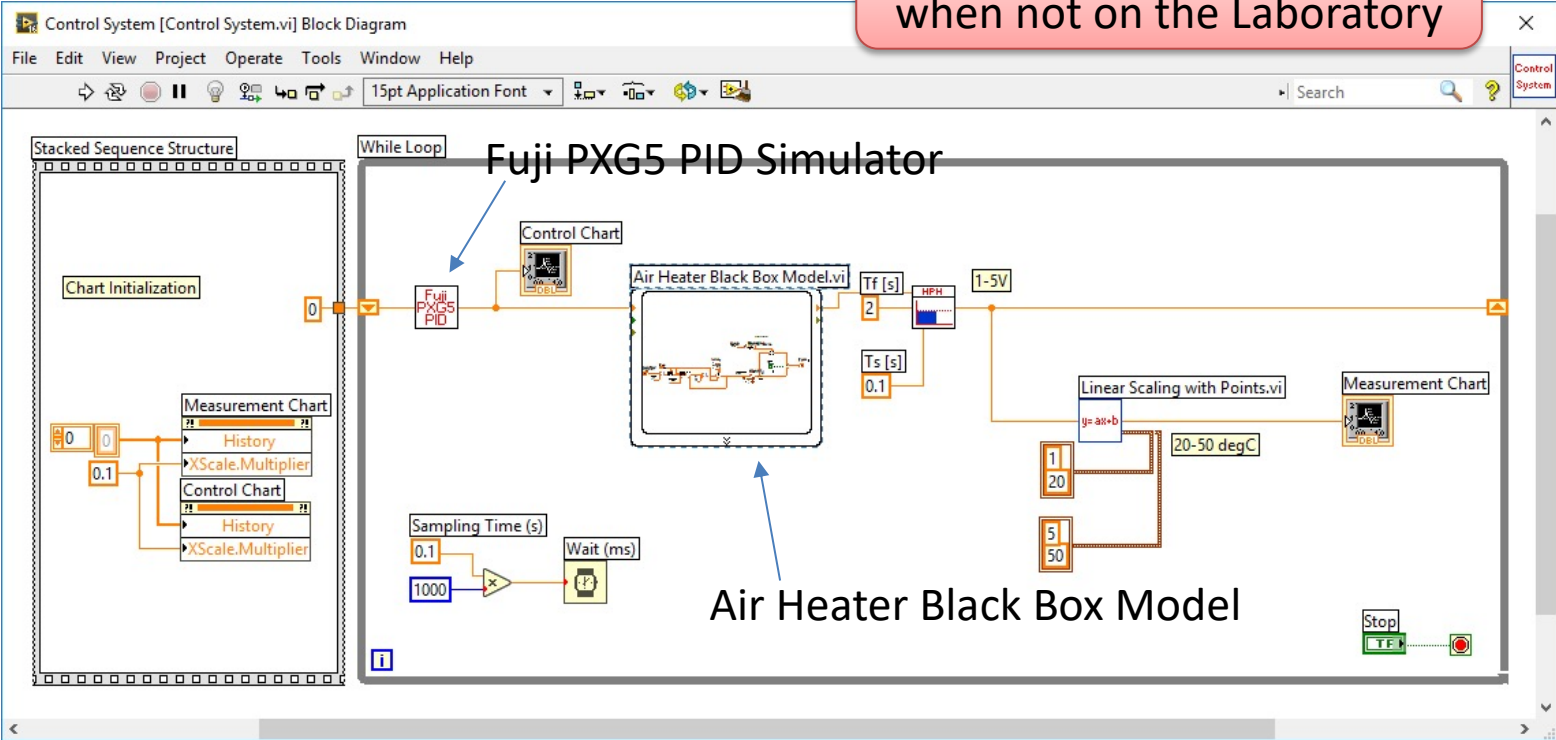
For Online Students and  
when not in the Laboratory



(You see the Block Diagram on the next slide)

# “Fuji PXG5 PID Simulator” + “Air Heater Black Box Model” Example

For Online Students and when not on the Laboratory



# Fuji PXG5 + Real Air Heater + PC for Monitoring

Industrial PID Controller

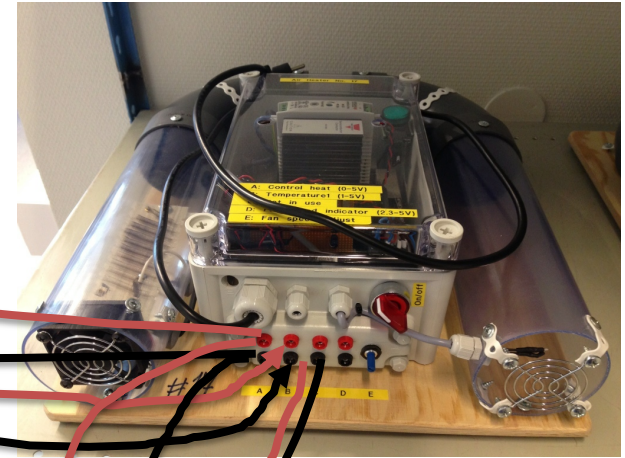


Process Value

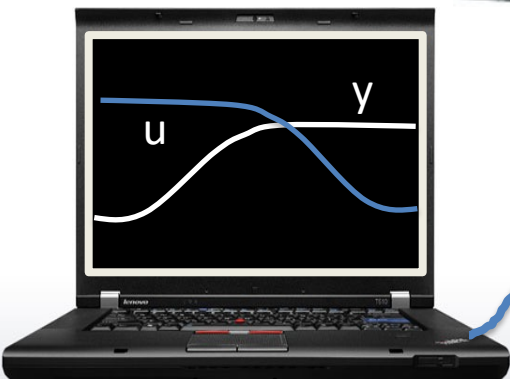
1-5V

0-5V

Control Signal



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



PC with LabVIEW

USB

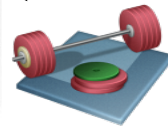


Process Value

1-5V

Control Signal

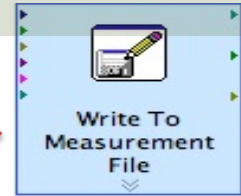
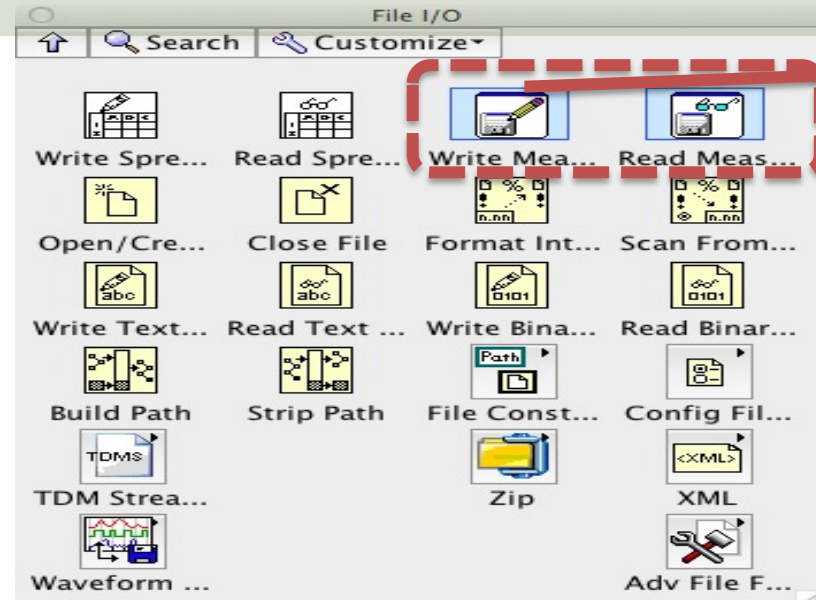
0-5V



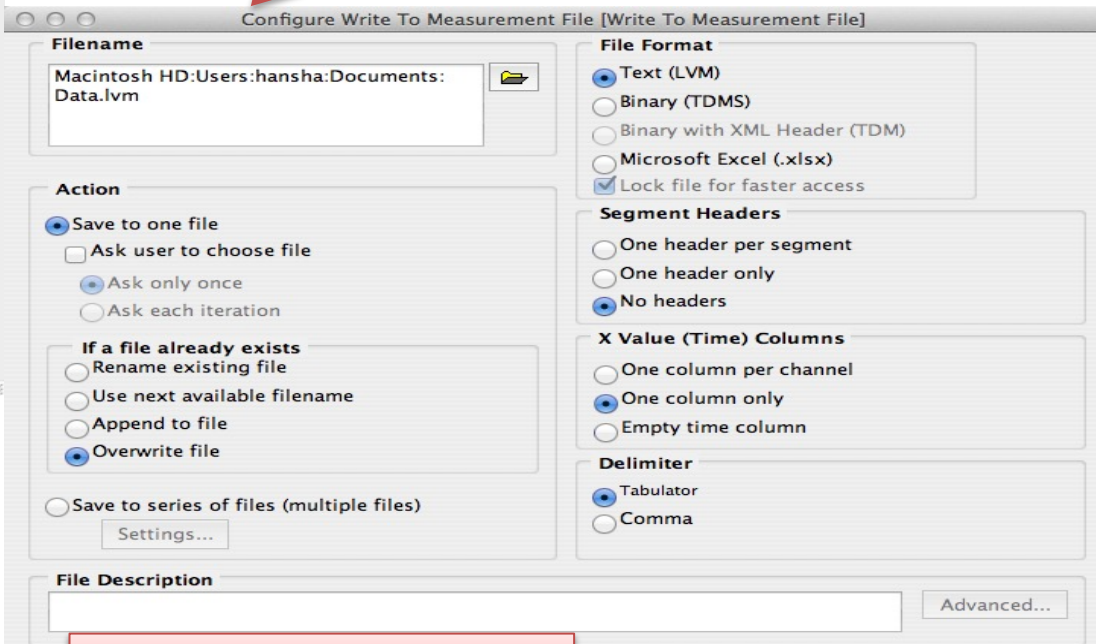
With this setup you can Monitor (Plot and Log Data to File) the Process Value and Control Signal on your PC



# Save Data to File (Data logging)



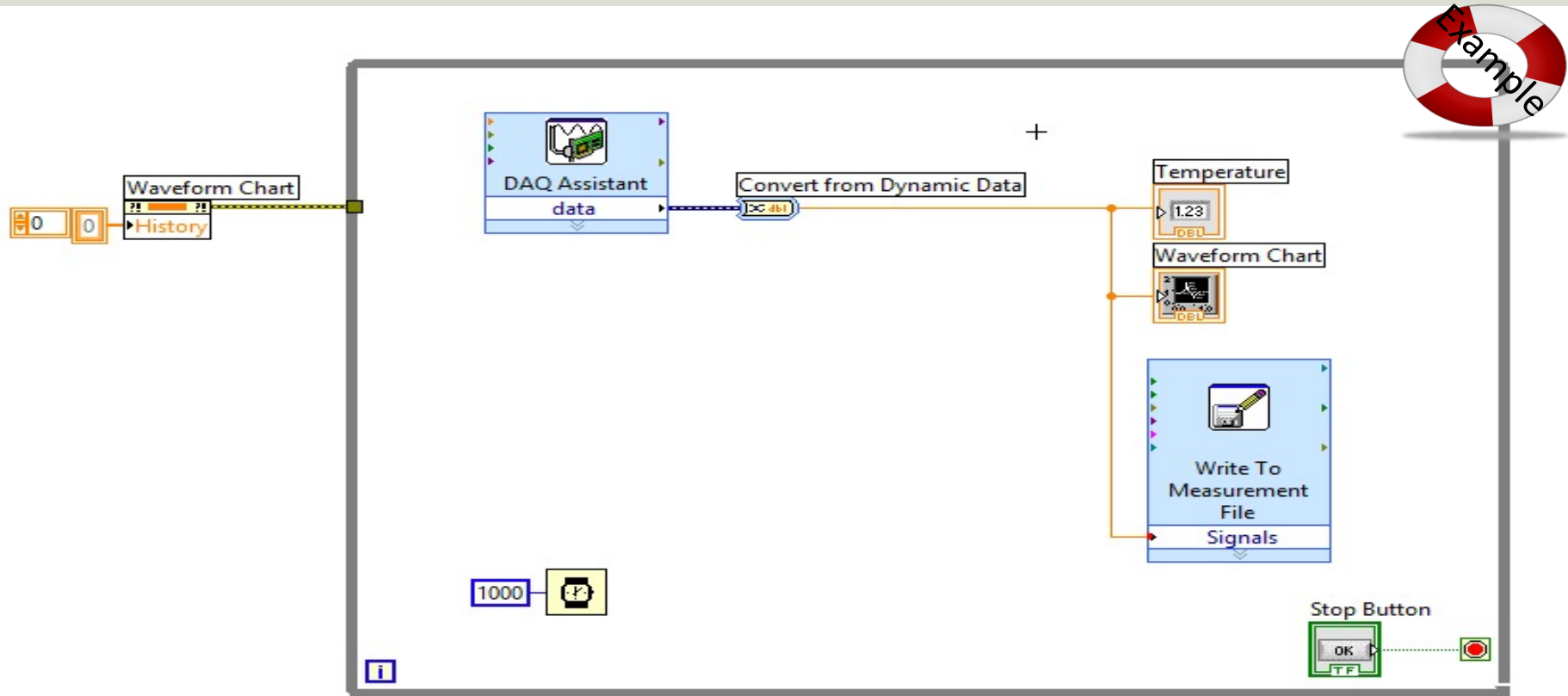
Right-click-Properties



Recommended Settings

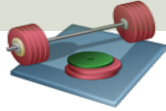


# Save Data to File (Datalogging)



# Measurement File – Data Visualization

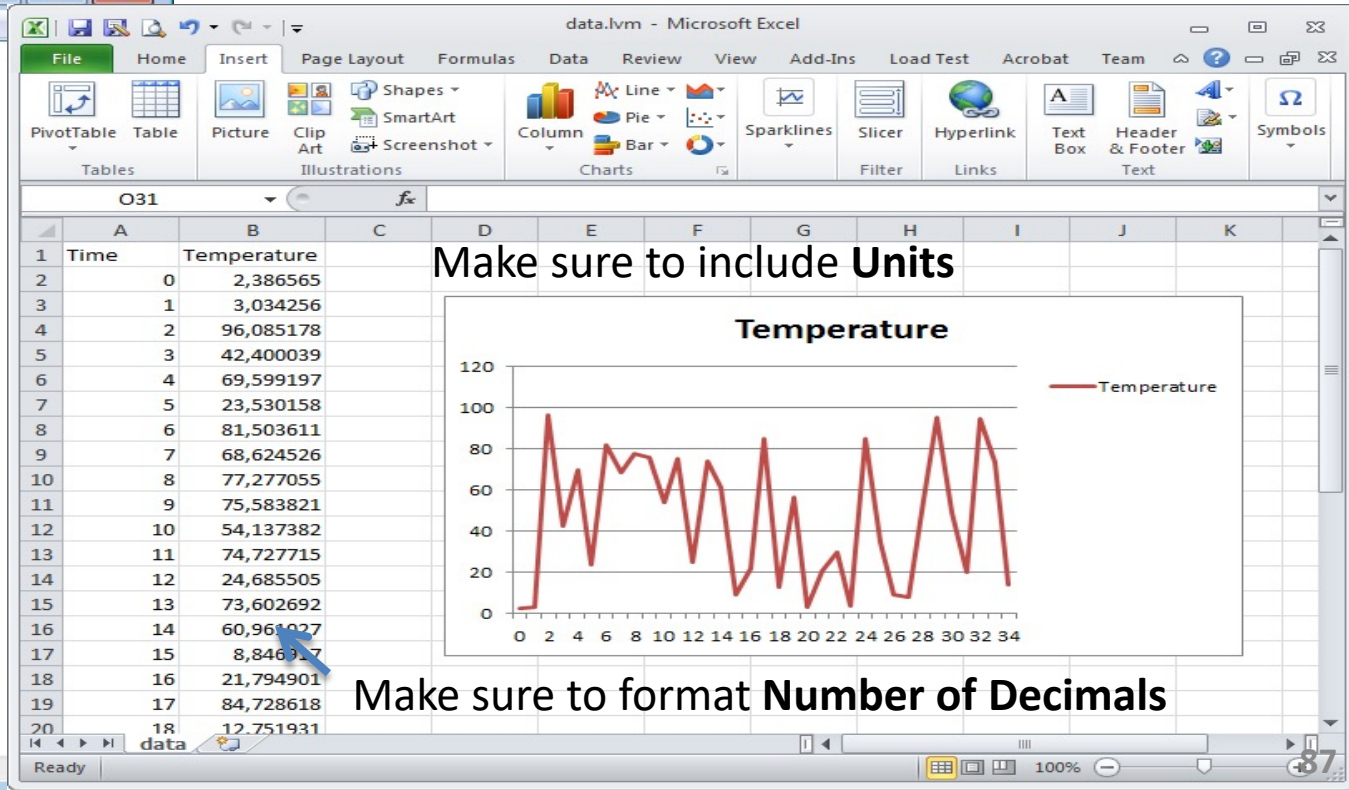
Open the File with Logged Data in e.g., Notepad:



Open the File with Logged Data in **MS Excel** and create a Chart (Measurement Value, Control Value)

data.lvm - Notisblokk

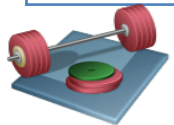
| Time      | Temperature |
|-----------|-------------|
| 0,000000  | 2,386565    |
| 1,000057  | 3,034256    |
| 2,000114  | 96,085178   |
| 3,000172  | 42,400039   |
| 4,000229  | 69,599197   |
| 5,000286  | 23,530158   |
| 6,000343  | 81,503611   |
| 7,000401  | 68,624526   |
| 8,000458  | 77,277055   |
| 9,000515  | 75,583821   |
| 10,000572 | 54,137382   |
| 11,000629 | 74,727715   |
| 12,000687 | 24,685505   |
| 13,000744 | 73,602692   |
| 14,000801 | 60,961027   |
| 14,999858 | 8,846917    |
| 15,999916 | 21,794901   |
| 16,999973 | 84,728618   |
| 18,000030 | 12,751931   |
| 19,000087 | 56,035920   |
| 20,000144 | 2,807199    |
| 21,000201 | 20,545774   |
| 22,000258 | 29,459929   |
| 23,000316 | 3,460532    |
| 24,000373 | 84,948125   |
| 25,000430 | 34,565660   |
| 26,000487 | 9,269143    |
| 27,000545 | 7,755839    |
| 28,000602 | 55,102455   |
| 29,000659 | 95,010244   |
| 30,000716 | 48,794147   |
| 31,000773 | 20,263779   |
| 32,000831 | 94,089261   |
| 32,999888 | 72,916191   |
| 33,999945 | 14,155009   |



# Comparison of PID Parameters

Example:

|                                 | Kp | PB [%] | Ti [sec] | Td [sec] | Tuning Method | Comment |
|---------------------------------|----|--------|----------|----------|---------------|---------|
| Built-in PID in LabVIEW         |    |        |          |          |               |         |
| Fuji on Model                   |    |        |          |          |               |         |
| Fuji Autotuning on Model        |    |        |          |          |               |         |
| Fuji on Real Process            |    |        |          |          |               |         |
| Fuji Autotuning on Real Process |    |        |          |          |               |         |
| ... others                      |    |        |          |          |               |         |



Fill out a similar Table with the values you find. Discuss the Results.

$$PB = \frac{100\%}{K_p} \Leftrightarrow K_p = \frac{100\%}{PB}$$

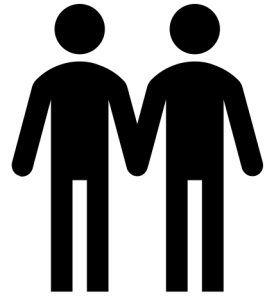


# Digital Twins

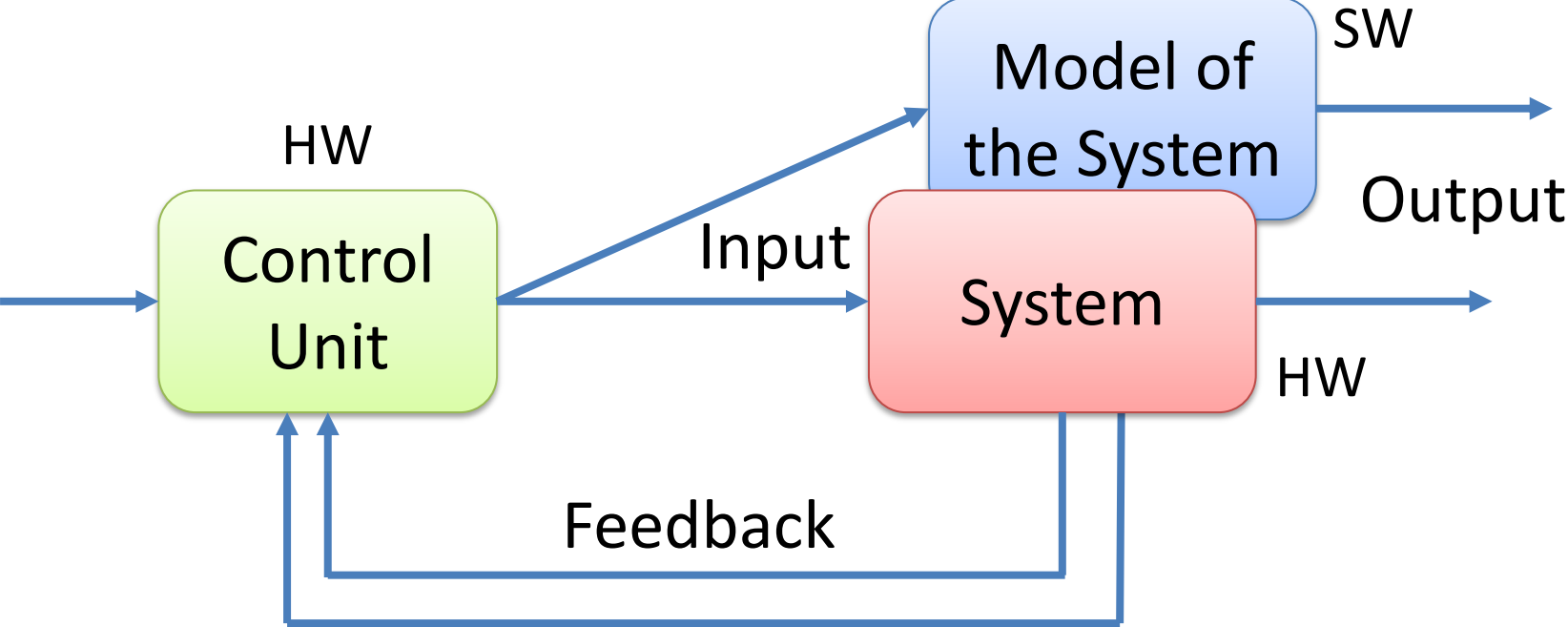
A brief Overview of a Digital Twins and how it is related to HIL Simulations

# Digital Twins

A digital twin is an “exact” digital replica of a product, process or service. This living model creates a thread between the physical and digital world



# General Digital Twin Concept Sketch



# Digital Twin

- Digital twins integrate internet of things, artificial intelligence, machine learning etc. to create living digital simulation models that update and change as their physical counterparts change.
- A digital twin continuously learns and updates itself from multiple sources (sensors, etc.) to represent its near real-time status



# Digital Twin and HIL

- The terms Digital Twin and Hardware in the Loop Simulation are closely related.
- National Instruments doesn't use the term digital twin, but their hardware-in-the-loop (HIL) testing technology platform provides similar functionality
- <http://www.ni.com/en-no/innovations/automotive/hardware-in-the-loop.html>
- <https://www.digitalengineering247.com/article/seeing-digital-twin-double/>

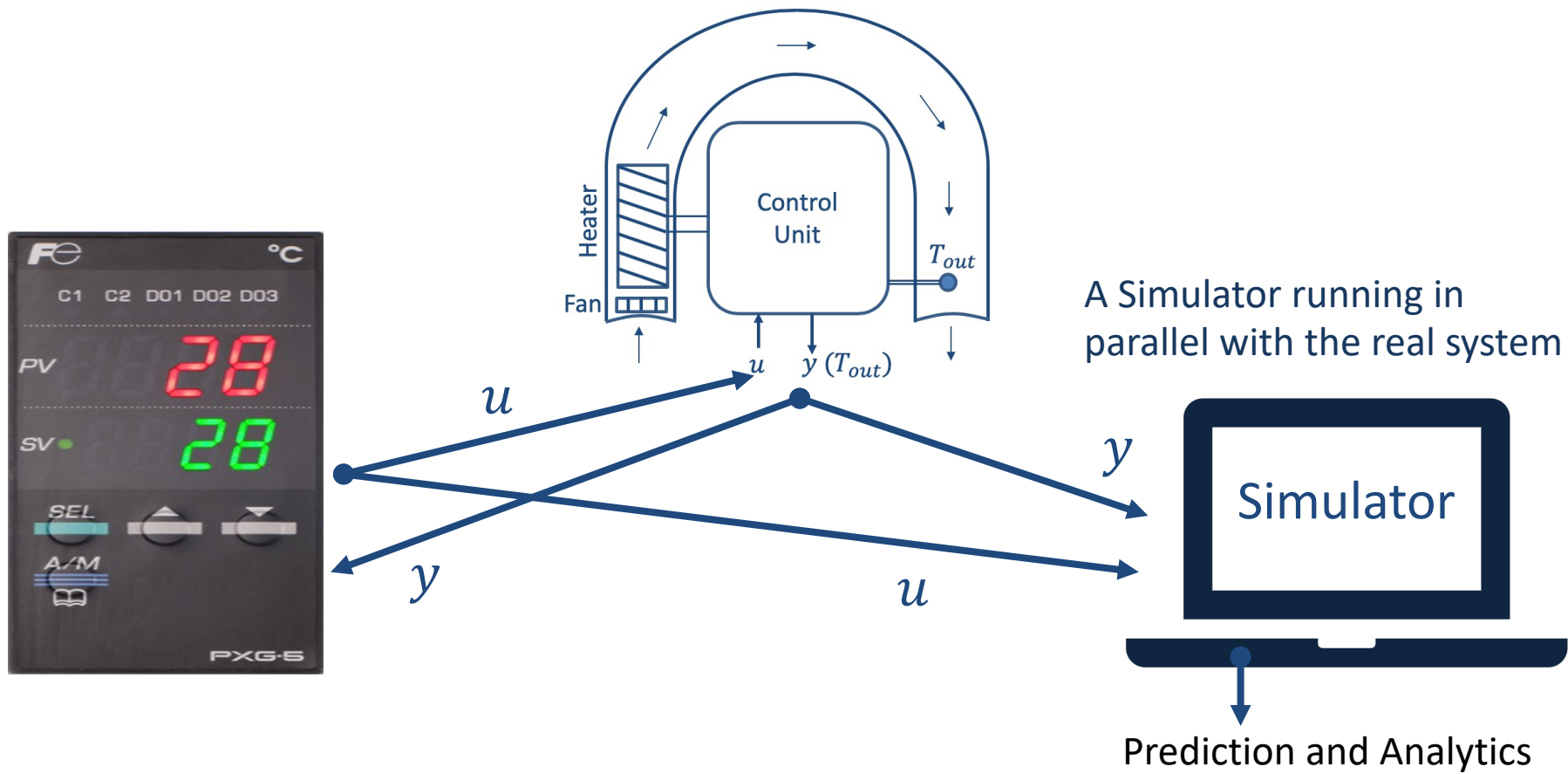
# Digital Twins in Automation Systems

- Preventive Maintenance
- Improved Control
- Model Based Control (MPC)
- State Estimation and Kalman Filter
- Can use the simulated environment to experiment with different controls strategies
- Model different scenarios to optimize the production

# Why Digital Twins?

- Preventive Maintenance
- Predictive Analytics
- Improved Control
- Can use the simulated environment to experiment with different controls strategies
- Model different scenarios to optimize the production

# Fuji PXG5 + Real Air Heater + Digital Twin



# Fuji PXG5 + Real Air Heater + Digital Twin

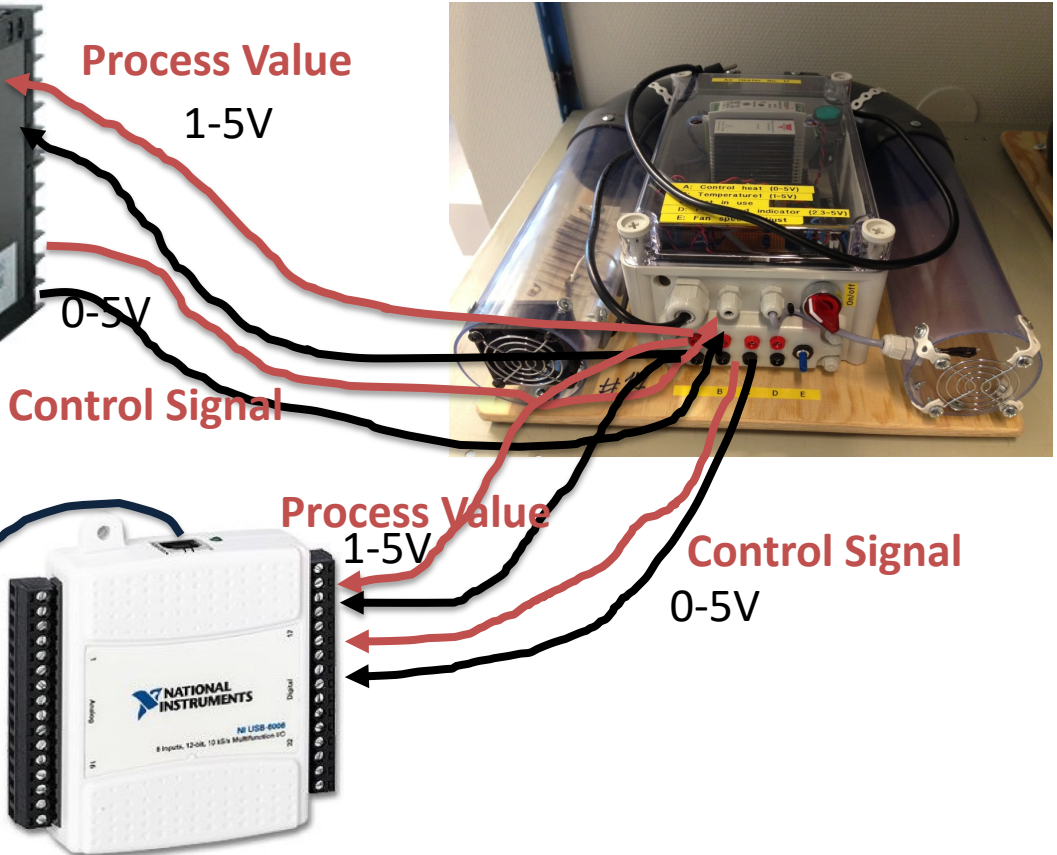
Industrial PID Controller



Model of Process  
(Air Heater)



Computer (with LabVIEW)



# Hans-Petter Halvorsen

University of South-Eastern Norway

[www.usn.no](http://www.usn.no)

E-mail: [hans.p.halvorsen@usn.no](mailto:hans.p.halvorsen@usn.no)

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

