2016.10.27

Embedded Arduino PID Controller

Hans-Petter Halvorsen, M.Sc.

System Overview

ARDUINO

Download your Application
and then remove USB cable

K_p, T_i, T_d r

Air Heater

Arduino

DAC

Analog In

y

Banana
Connector

Breadboard

Analog Out

u

Banana
Connector

Banana
Connector

AI

AO

Banana
Connector

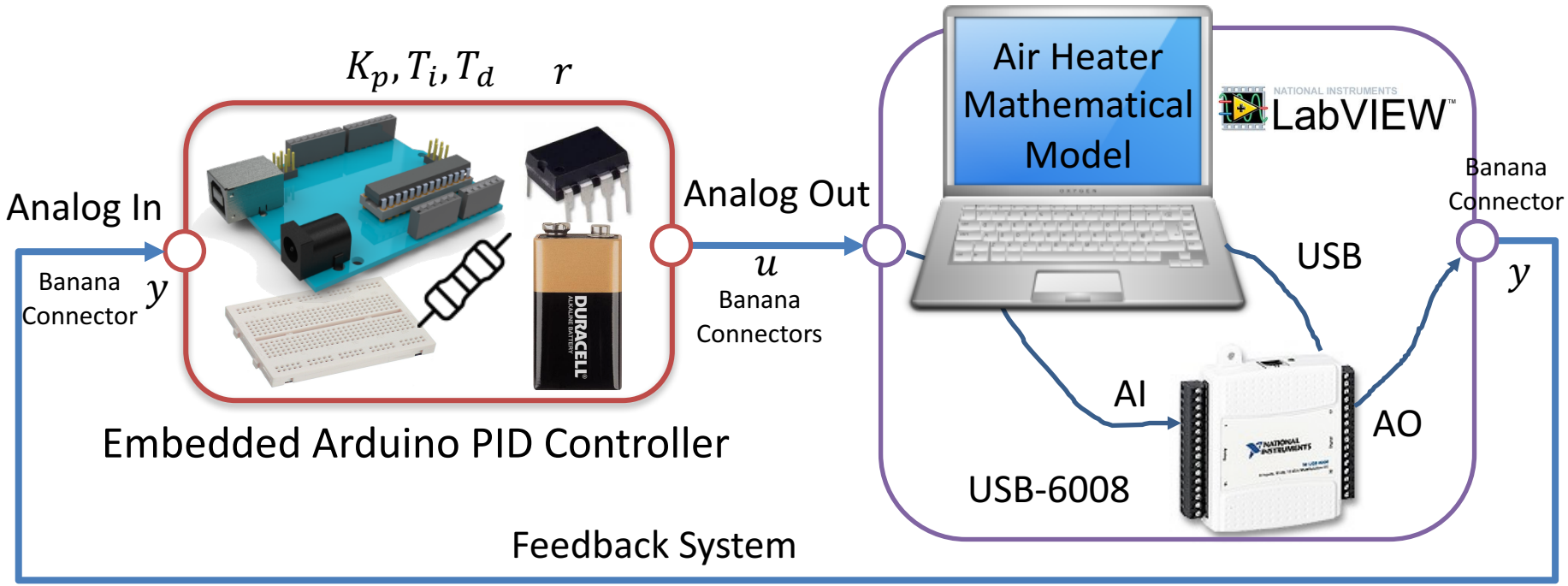
y

Embedded Arduino PID Controller

Test first using HIL
Simulation and Testing

Feedback System

HIL Simulation and Testing



Note! It's important that you test it out properly using HIL simulation, so you don't damage the real Air Heater Process.

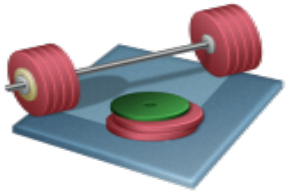
Keywords

- Arduino
- Control Design and Simulations
- Practical Implementation of Control Systems
- PID
- Hardware in the Loop Simulation and Testing

Learning Goals

- Learn PID Control, both Control theory and Practical implementations
- Learn Programming, Arduino Programming and LabVIEW
- Learn about Microcontrollers (Arduino)
- Learn about Hardware-Software Interactions
- Learn about Digital to Analog Conversion (DAC)
- Learn SPI Communication
- Learn HIL Simulation and Testing
- Learn about Electronics and Electrical Components

Lab Assignment Overview



1. Create Embedded PI(D) Controller + Lowpass Filter in Arduino code
   - Create “Arduino Library” with functions
   - Create “Analog Out”, either using DAC chip or use a simple RC circuit on a breadboard (Arduino has no built-in AO).
2. Test PI(D) controller using HIL Simulations and Testing
   - Create Model of Air Heater in LabVIEW and connect to Arduino embedded PI(D) controller using a USB-6008 DAQ device
3. Publish Data to Server

See next slides for details...

Software

Arduino IDE

PID Controller

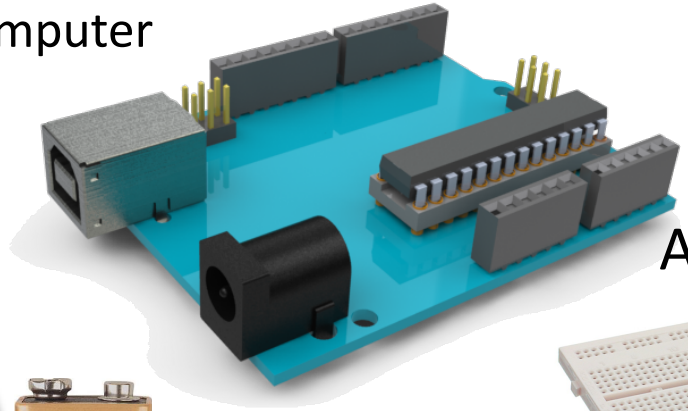
Lowpass Filter

Air Heater Simulator

HIL Simulation and Testing

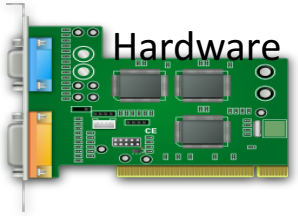
Hardware

Your Personal Computer

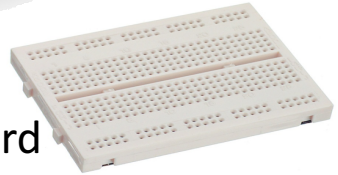


Arduino

DAQ Device, e.g. USB-6008, USB-6001, myDAQ

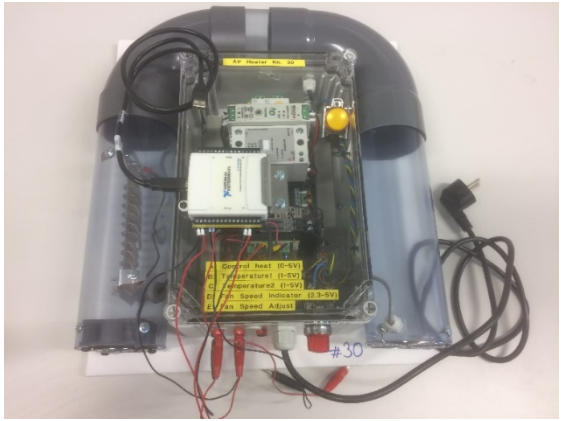


Hardware

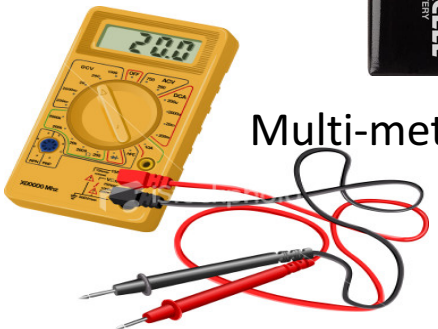


Breadboard

9V Battery

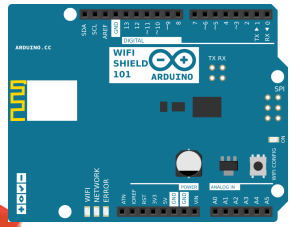


Air Heater



Multi-meter

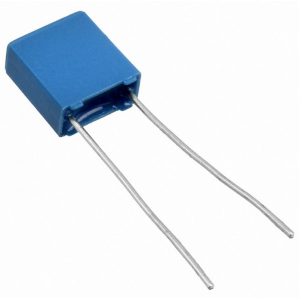
Banana Plugs/Cables



(Arduino Wi-Fi Shield)

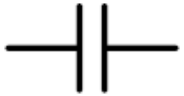
Electrical Components

Capacitor



$C = 0.1\mu F = 100nF$

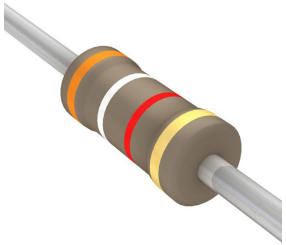
or $C = 10\mu F$? Try out!



A capacitor stores and releases electrical energy in a circuit. When the circuits voltage is higher than what is stored in the capacitor, it allows current to flow in, giving the capacitor a charge. When the circuits voltage is lower, the stored charge is released. Often used to smooth fluctuations in voltage

https://en.wikipedia.org/wiki/Capacitor

Resistor



$R = 3.9k\Omega$

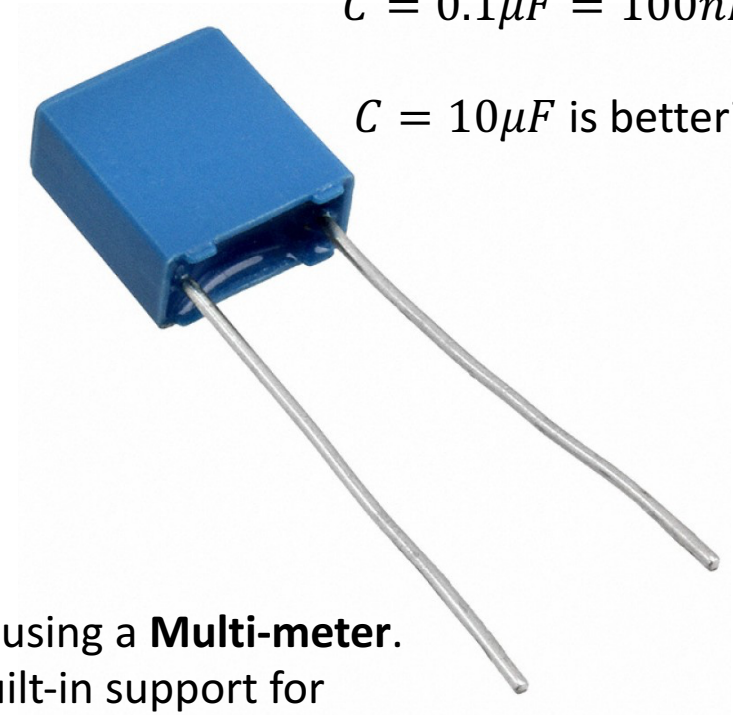
A resistor resists the flow of electrical energy in a circuit, changing the voltage and current as a result (according to Ohms law, $U = RI$). Resistor values are measured in ohms (Ω). The color stripes on the sides of the resistor indicate their values. You can also use a Multi-meter in order to find the value of a given resistor.

These electronics components are typically included in a “Starter Kit”, or they can be bought “everywhere” for a few bucks.

Capacitor

$C = 0.1\mu F = 100nF$

$C = 10\mu F$ is better?

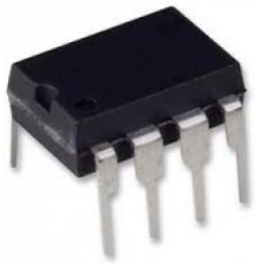


The Capacitor is included in the Arduino Starter Kit (or similar Kits).

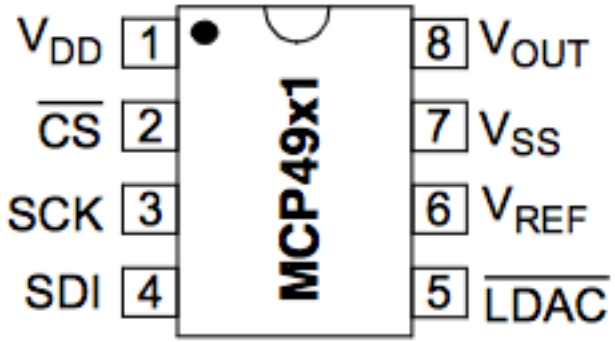
If you don't have such a Kit you may buy capacitors from Elfa, Kjell & Company, etc.

Note! You can also easily measure the capacitance using a **Multi-meter**.
A Multi-meter that cost from 400-500+ NOK has built-in support for measuring capacitors (same for resistors and resistance).

DAC



Arduino UNO has no Analog Output Pins, so we need a DAC such as, e.g., Microchip MCP4911



MCP4911: 10-bit single DAC

Microchip MCP4911 can be bought “everywhere” (10 NOK).

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.

NATIONAL INSTRUMENTS LabVIEW

My System is not Working??

Multimeter, etc.

"Google It"!

You probably will find the answer on the Internet

Google

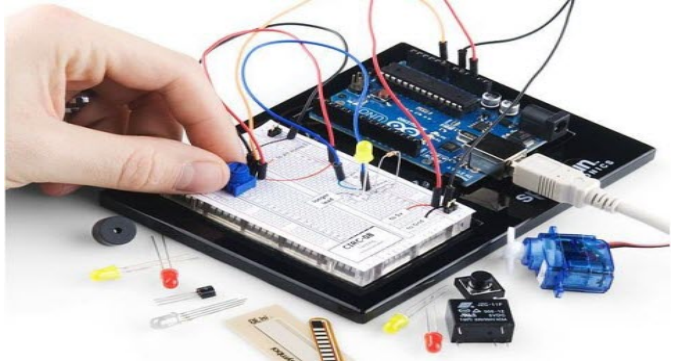
Another person in the world probably had a similar problem

Your hardware device most likely works, so you don't need a new device! Still not working after Troubleshooting & Debugging? Fill out an "Equipment Error Form"

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.

Arduino

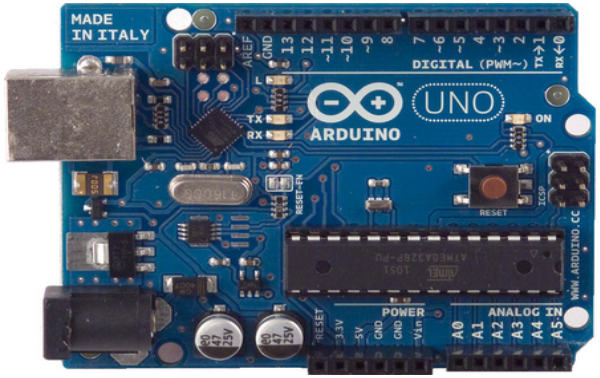
Hans-Petter Halvorsen, M.Sc.



- Arduino is an open-source physical computing platform designed to make experimenting with electronics and programming more fun and intuitive.
- Arduino has its own unique, simplified programming language and a lots of premade examles and tutorials exists.
- With Arduino you can easily explore lots of small-scale sensors and actuators like motors, temperature sensors, etc.
- The possibilities with Arduino are endeless.

http://www.arduino.cc

Arduino UNO



https://www.arduino.cc/en/Guide/HomePage

http://www.arduino.cc

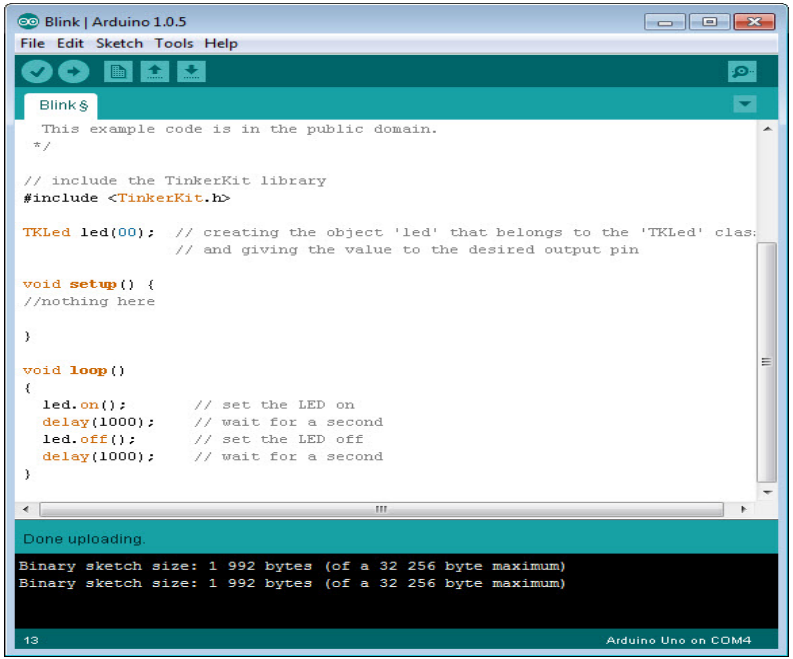
Pin Overview: http://pighixxx.com/unov3pdf.pdf

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Arduino Software

Programming with Arduino is simple and intuitive!

Arduino Sketch IDE



The syntax is similiar to C programming

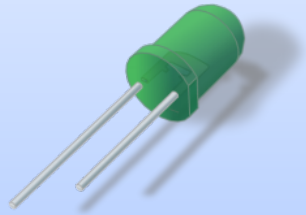
Example:

```
// include the TinkerKit library
#include <TinkerKit.h>

// creating the object 'led' that belongs to the 'TKLed' class
TKLed led(O0);


void setup()
 {
              //do something here
 }

void loop()
{
  led.on();         // set the LED on
  delay(1000);      // wait for a second
  led.off();        // set the LED off
  delay(1000);      // wait for a second
}
```

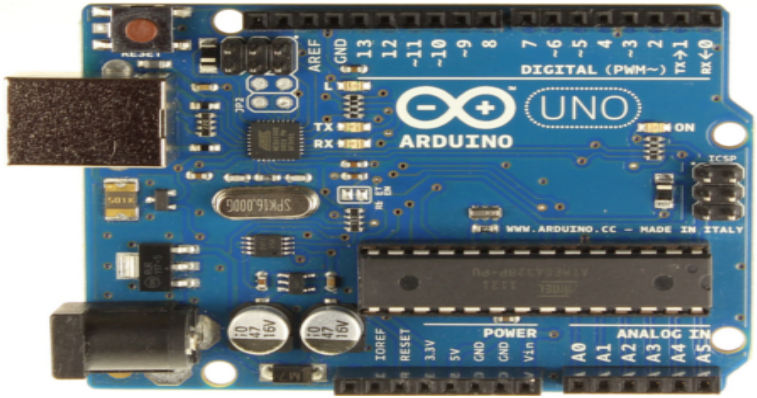


This program makes a LED blink

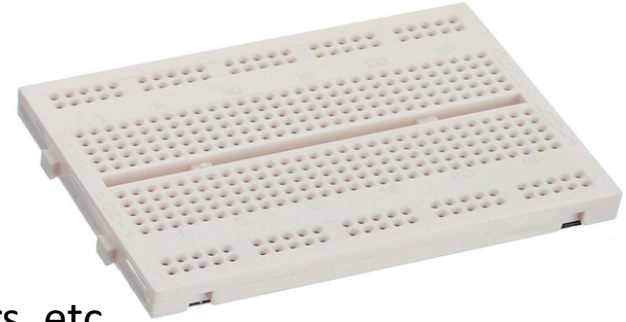
Software Installation: http://arduino.cc/en/Main/Software

Arduino Basics

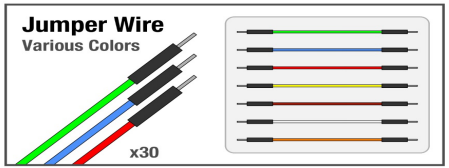
Arduino Uno Board

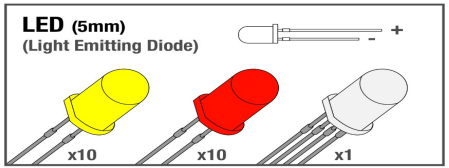


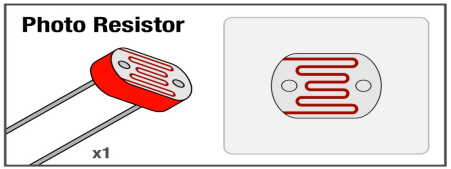
Breadboard

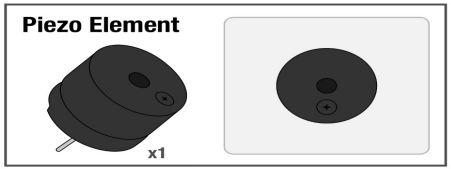


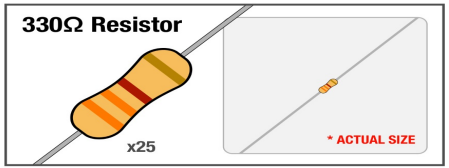
Sensors and Actuators, etc.

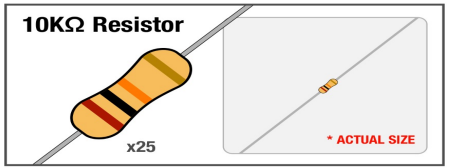


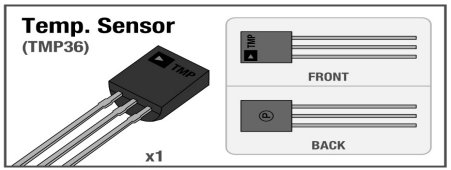


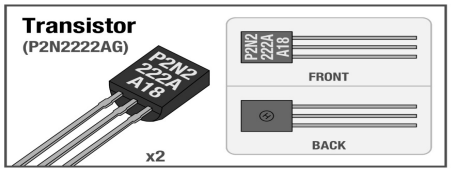


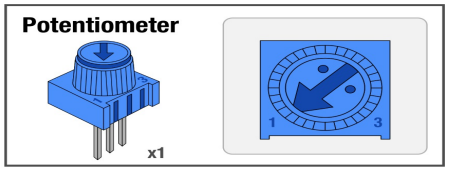


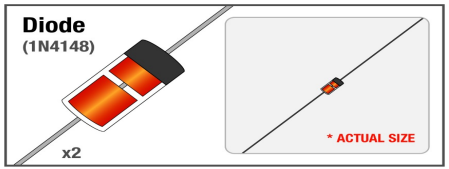


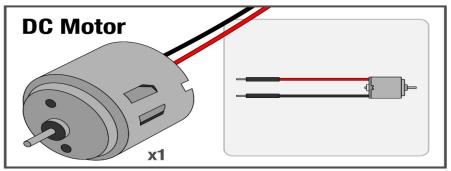


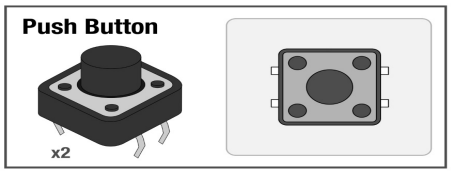




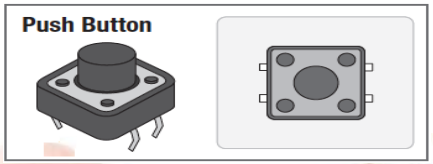


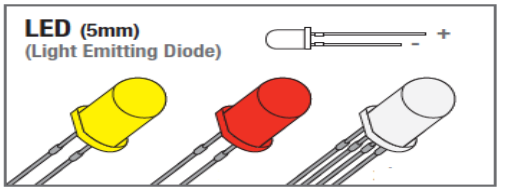


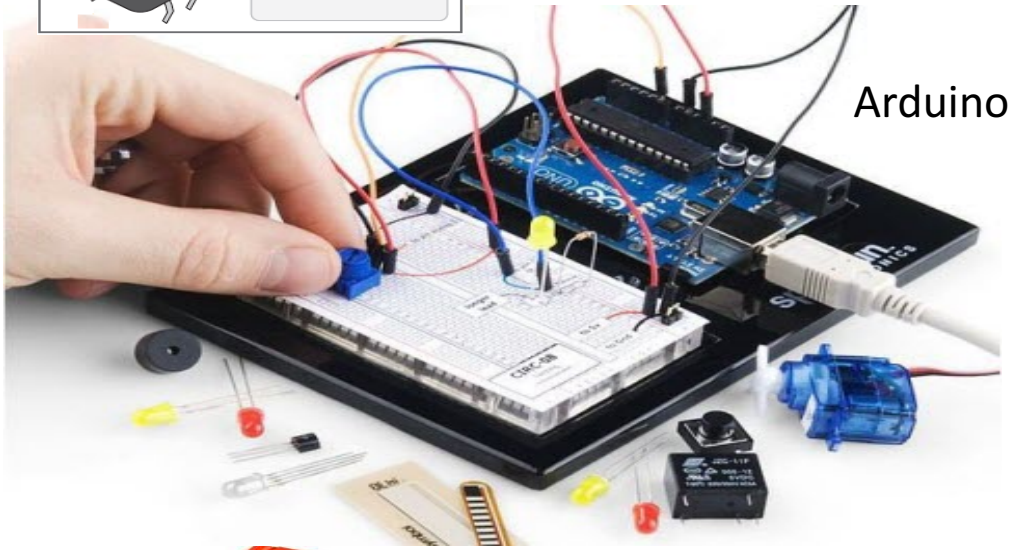




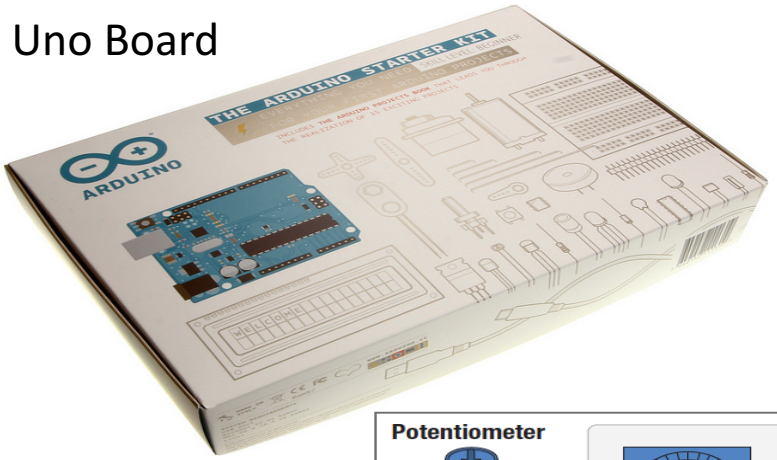
The Arduino Kit

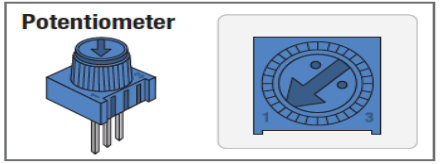




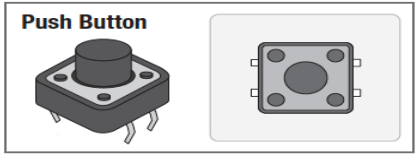


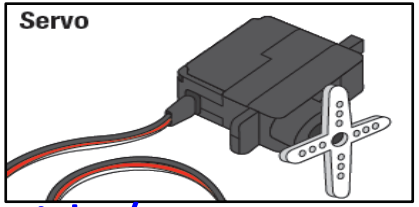
Arduino Uno Board





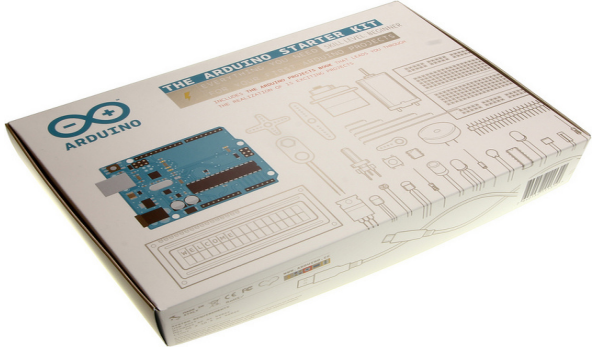
Small-size Sensors and Actuators





Getting Started with Arduino: http://arduino.cc/en/Guide/HomePage

The Arduino Kit



- Ardiono Home Page: http://arduino.cc
- The Arduino Starter Kit:

http://arduino.cc/en/Main/ArduinoStarterKit

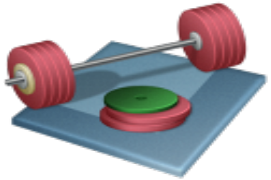
- Starter Kit Videos:
  https://www.youtube.com/playlist?feature=edit_ok&list=PLT6rF_I5kknPf2qlVFlvH47qHvqvzkknd

PID Control

Theory

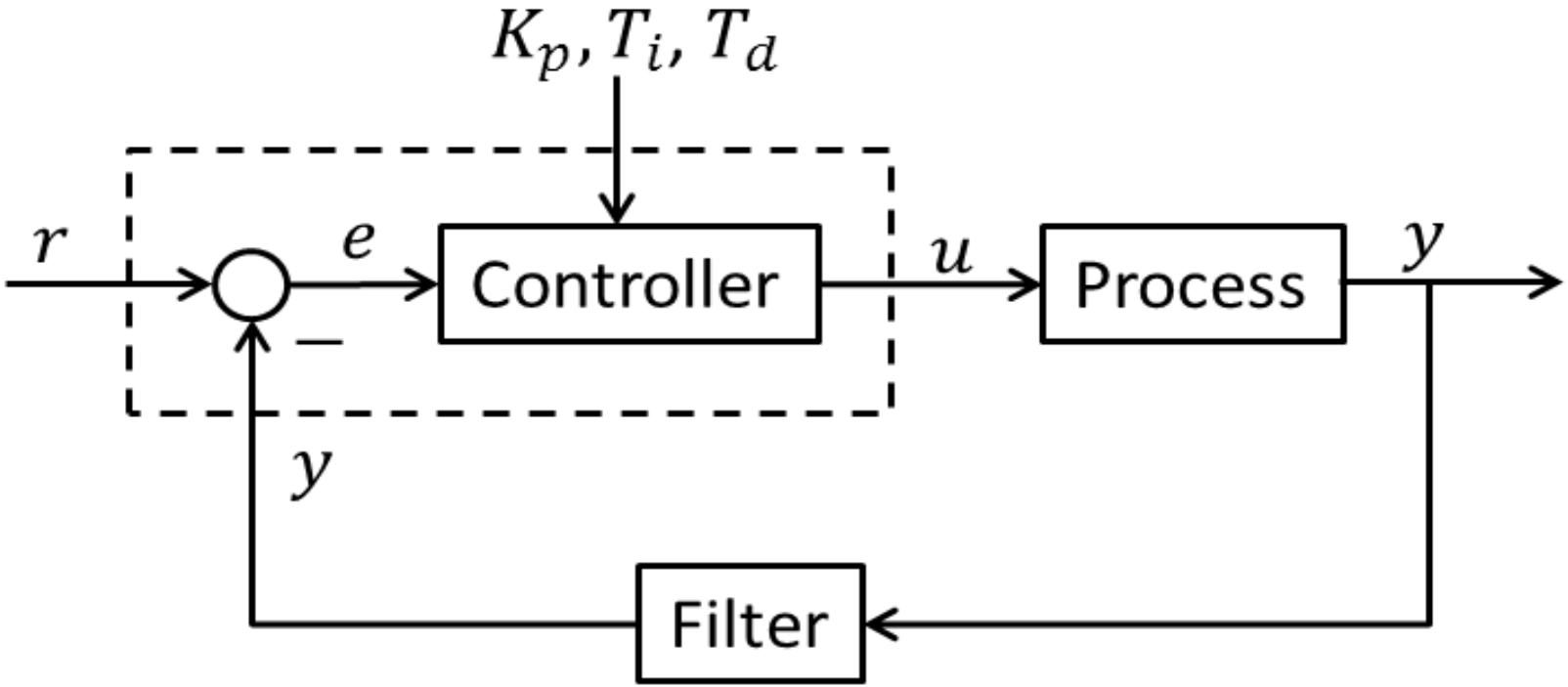
Hans-Petter Halvorsen, M.Sc.

Arduino PID Controller

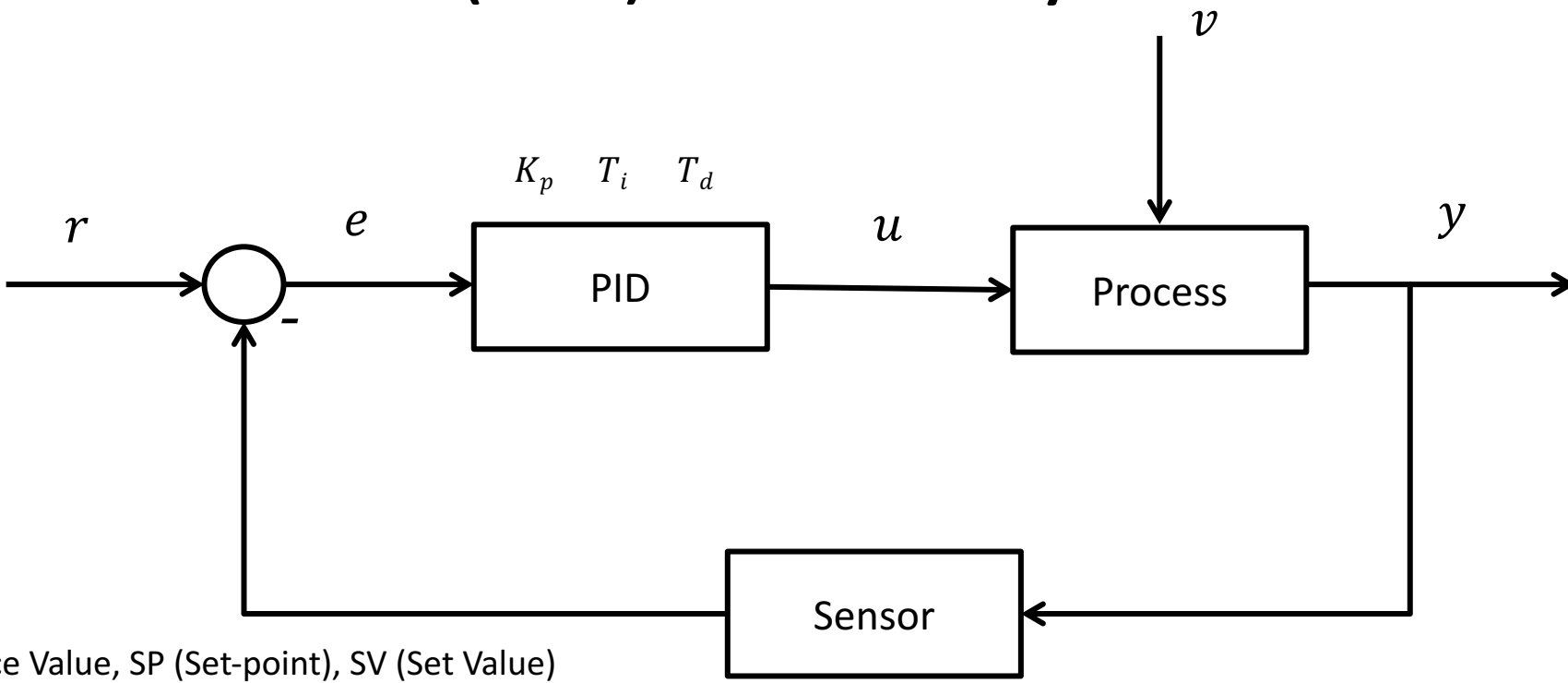


- Find a discrete PID algorithm (using pen and paper)
- Implement the PID algorithm using Arduino Programming

Feedback (PID) Control



Feedback (PID) Control System



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

The PID Algorithm

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

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

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

r is the Reference Signal or Set-point

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

Tuning Parameters:

K_p Proportional Gain

T_i Integral Time [sec.]

T_d Derivative Time [sec.]

Example of Discrete PID Controller

[F. Haugen, Discretization of simulator, filter, and PID controller: TechTeach, 2010

http://www.mic-journal.no/PDF/ref/Haugen2010.pdf

The starting point of deriving the discrete-time PID controller is the continuous-time PID (proportional + integral + derivate) controller:

$$u(t) = u_0 + K_p e(t) + \frac{K_p}{T_i}\int_0^t e\,d\tau + K_p T_d \dot{e}(t) \tag{19}$$

where u_0 is the control bias or manual control value (to be adjusted by the operator when the controller is in manual mode), u is the controller output (the control variable), e is the control error:

$$e(t) = r(t) - y(t) \tag{20}$$

where r is the reference or setpoint, and y is the process measurement.

Differentiating and applying the Backward differentiation method gives:

Solving for $u(t_k)$ finally gives the discrete-time PID controller:

$$\begin{aligned} u(t_k) &= u(t_{k-1}) + [u_0(t_k) - u_0(t_{k-1})] && (30)\\ &\quad + K_p\,[e(t_k) - e(t_{k-1})] && (31)\\ &\quad + \frac{K_p T_s}{T_i} e(t_k) && (32)\\ &\quad + \frac{K_p T_d}{T_s}\,[e(t_k) - 2e(t_{k-1}) + e(t_{k-2})] && (33) \end{aligned}$$

This is just one Example. You may implement another algorithm if you prefer.

The discrete-time PID controller algorithm (30) is denoted the *absolute* or *positional* algorithm. Automation devices typically implements the *incremental* or *velocity* algorithm. because it has some benefits. The incremental algorithm is based on splitting the calculation of the control value into two steps:

1. First the *incremental control value* $\Delta u(t_k)$ is calculated:

$$\begin{aligned} \Delta u(t_k) &= [u_0(t_k) - u_0(t_{k-1})] && (34)\\ &\quad + K_p\,[e(t_k) - e(t_{k-1})] && (35)\\ &\quad + \frac{K_p T_s}{T_i} e(t_k) && (36)\\ &\quad + \frac{K_p T_d}{T_s}\,[e(t_k) - 2e(t_{k-1}) + e(t_{k-2})] && (37) \end{aligned}$$

2. Then the *total or absolute control value* is calculated with

$$u(t_k) = u(t_{k-1}) + \Delta u(t_k) \tag{38}$$

The summation (38) implements the (numerical) integral action of the PID controller.

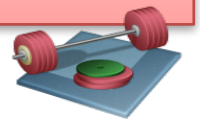
Congratulations! - You are finished with the Task

Lowpass Filter

Hans-Petter Halvorsen, M.Sc.

Discrete Lowpass Filter

Create and use a Lowpass Filter together with the PID Controller. Implement the Lowpass Filter as a separate Function



Lowpass Filter Transfer function:

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

Inverse Laplace gives the differential Equation:

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

We use the Euler Backward method:

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

This gives:

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

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

We define:

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

This gives:

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

y_k: Filter output

u_k: Noisy input signal

This algorithm can be easly implemented in a Programming language

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

Congratulations! - You are finished with the Task

Analog Out

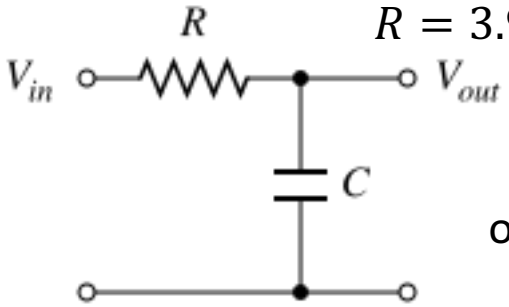
Hans-Petter Halvorsen, M.Sc.

Arduino Analog Out

- Arduino has no built-in Analog Output Channels
- We need Analog Out for the Control Signal ($0 - 5V$)
- We can use a DAC chip/IC or create a RC Lowpass Filter that converts PWM to Voltage
- Its recommended that you try both alternatives.

Alt 1: Convert PWM to Voltage

RC Lowpass Filter:



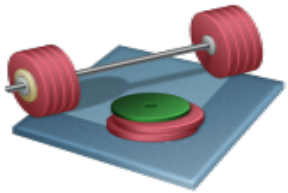
$C = 0.1\mu F = 100nF$

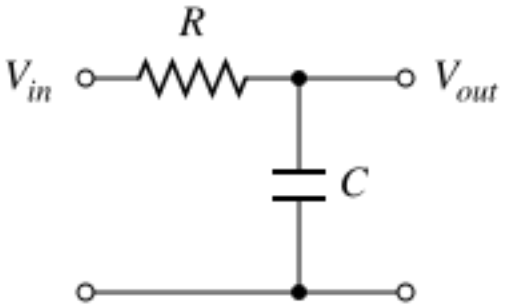
or $C = 10\mu F$?

Try out!

- http://www.instructables.com/id/Analog-Output-Convert-PWM-to-Voltage/
- http://provideyourown.com/2011/analogwrite-convert-pwm-to-voltage/

Alt 1: Design and Analysis





- Find the Transfer Function for the RC Lowpass Filter.
- Start finding the differential equation and then use Laplace in order to find the transfer function.

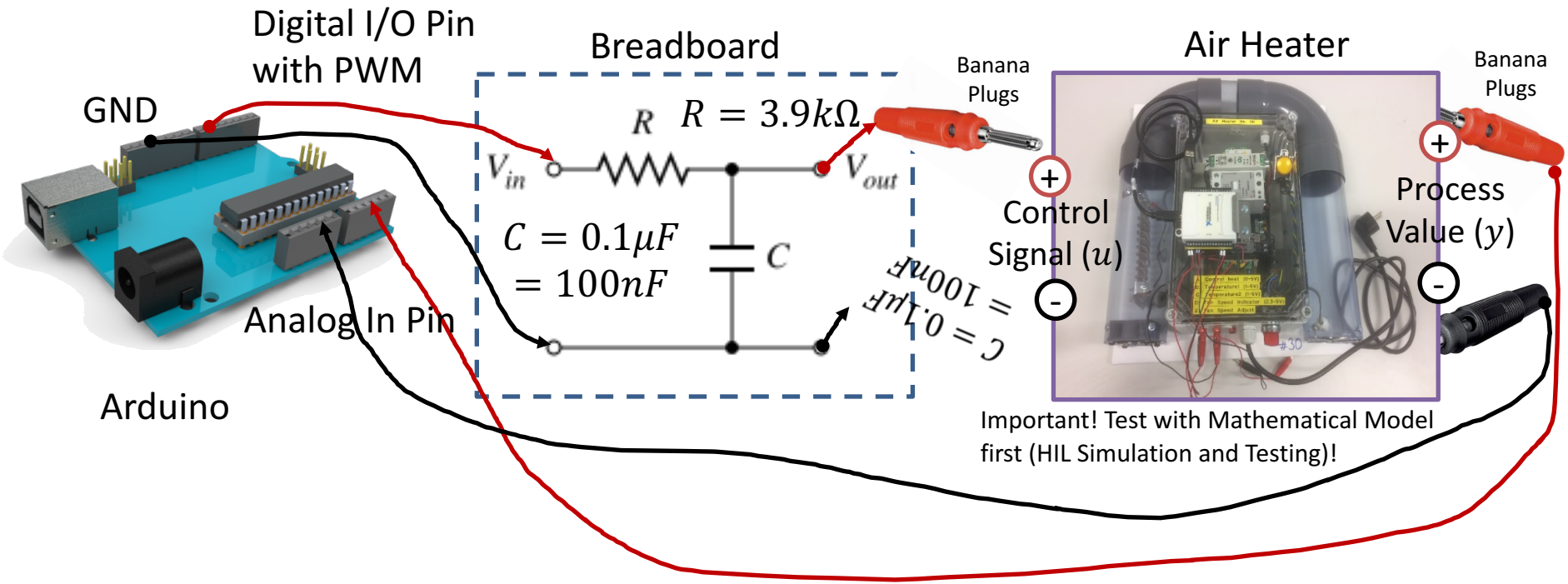
$$H(s) = \frac{V_{out}}{V_{in}} = ?$$

- Create and simulate a PWM signal in LabVIEW.
- Create the RC Lowpass Filter transfer function in LabVIEW.
- Design and Analyze the RC Lowpass Filter based on simulations (Time domain and Frequency domain) in LabVIEW. Find proper values for R and C. Find Bandwidth/Cut-off frequency, etc.

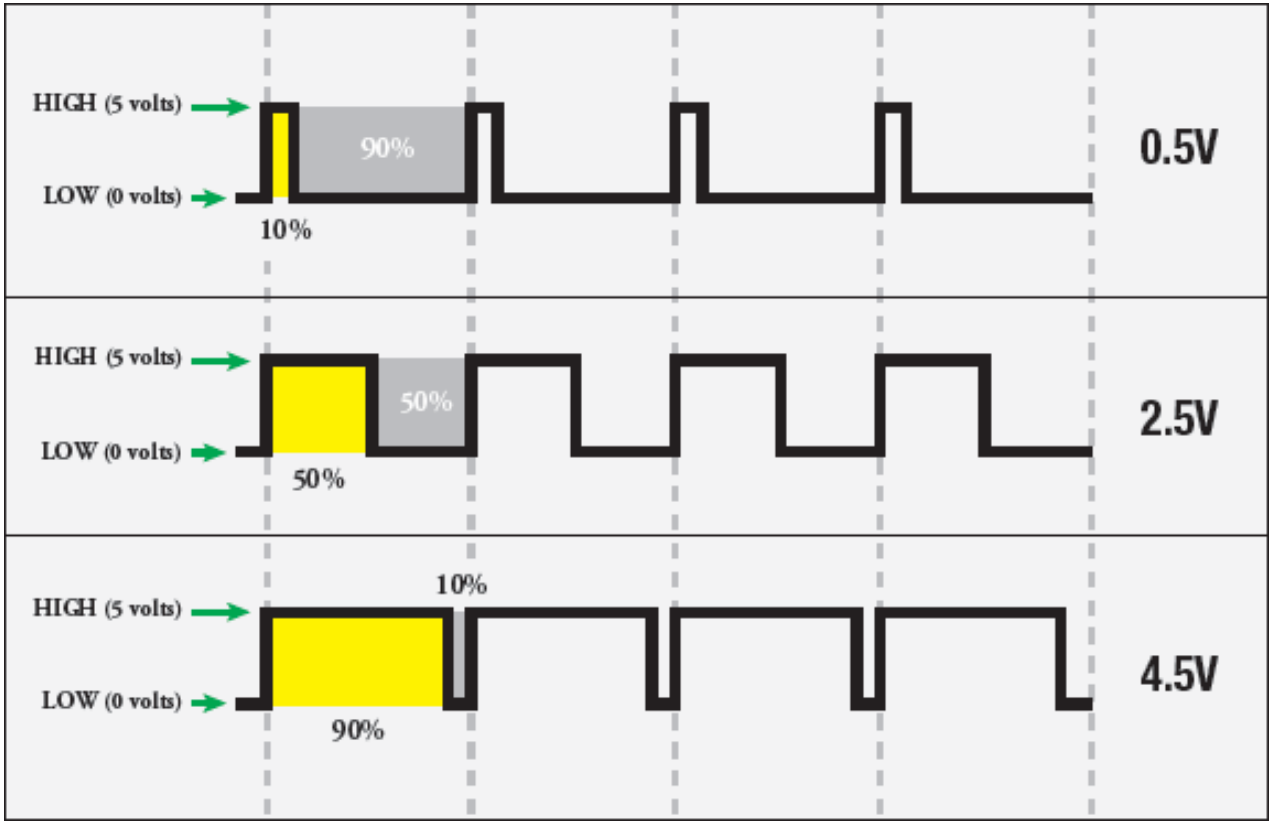
Examples:
http://techteach.no/simview/rc_circuit/
http://techteach.no/simview/pwm_control/

Alt 1: Convert PWM to Voltage

Below you see an example how you can wire:



Pulse-Width Modulation (PWM)



Pulse-Width Modulation (PWM)

The shocking truth behind analogWrite():

- We know that the Arduino can read analog voltages (voltages between 0 and 5 volts) using the analogRead() function.
- Is there a way for the Arduino to output analog voltages as well? The answer is no... and yes. Arduino does not have a true analog voltage output. But, because Arduino is so fast, it can fake it using something called PWM ("Pulse-Width Modulation"). The pins on the Arduino with “~” next to them are PWM/Analog out compatible.
- The Arduino is so fast that it can blink a pin on and of almost 1000 times per second. PWM goes one step further by varying the amount of time that the blinking pin spends HIGH vs. the time it spends LOW. If it spends most of its time HIGH, a LED connected to that pin will appear bright. If it spends most of its time LOW, the LED will look dim. Because the pin is blinking much faster than your eye can detect, the Arduino creates the illusion of a "true" analog output.
- To smooth the signal even more, we will create and use a RC circuit (Lowpass Filter)

Pulse-Width Modulation (PWM)

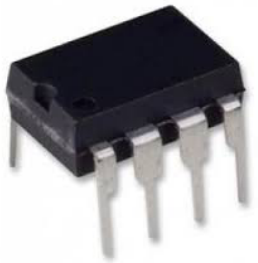
- The Arduino's programming language makes PWM easy to use; simply call analogWrite(pin, dutyCycle), where dutyCycle is a value from 0 to 255, and pin is one of the PWM pins (3, 5, 6, 9, 10, or 11).
- The analogWrite function provides a simple interface to the hardware PWM, but doesn't provide any control over frequency. (Note that despite the function name, the output is a digital signal, often referred to as a square wave.)

$$0 - 5V \rightarrow 0 - 255 \rightarrow y(x) = 51x$$
$$u = 0V \rightarrow analogWrite(0)$$
$$u = 5V \rightarrow analogWrite(255)$$
$$u = xV \rightarrow analogWrite(51 * x)$$

analogWrite():
https://www.arduino.cc/en/Reference/AnalogWrite

Secrets of Arduino PWM:
https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM

Alt 2: Use a DAC chip

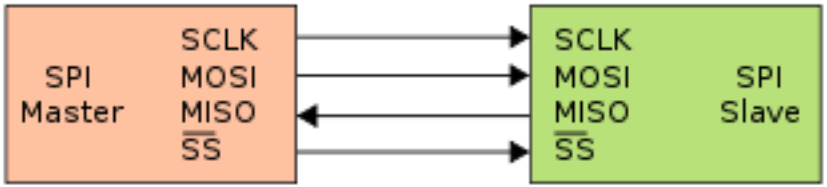


- DAC – Digital to Analog Converter
- Use, e.g., Microchip MCP4911
- **SPI** Arduino Library: https://www.arduino.cc/en/Reference/SPI
- MCP49XX Arduino Library: https://github.com/exscape/electronics/tree/master/Arduino/Libraries

SPI Bus

- Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances.
- With an SPI connection there is always one master device (usually a microcontroller) which controls the peripheral devices.
- SPI devices communicate in full duplex mode using a master-slave architecture with a single master.
- The interface was developed by Motorola and has become a de facto standard.
- Typical applications include sensors, Secure Digital cards, and liquid crystal displays (LCD).

SCLK : Serial Clock (output from master)
MOSI : Master Output, Slave Input (output from master)
MISO : Master Input, Slave Output (output from slave)
SS (or SC) : Slave Select (active low, output from master)



http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi

Arduino SPI

- https://www.arduino.cc/en/Reference/SPI
- http://tronixstuff.com/2011/05/13/tutorial-arduino-and-the-spi-bus/
- http://arduino.stackexchange.com/questions/16348/how-do-you-use-spi-on-an-arduino
- https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi

MCP4911: 10-bit single DAC

Arduino

SCK (13)

MISO (12)

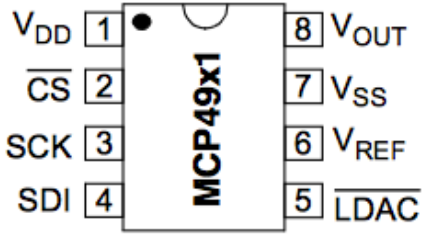
MOSI (11)

...

MCP4911

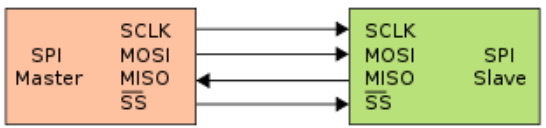
Vdd (1)

CS (2)



MCP4911: 10-bit single DAC

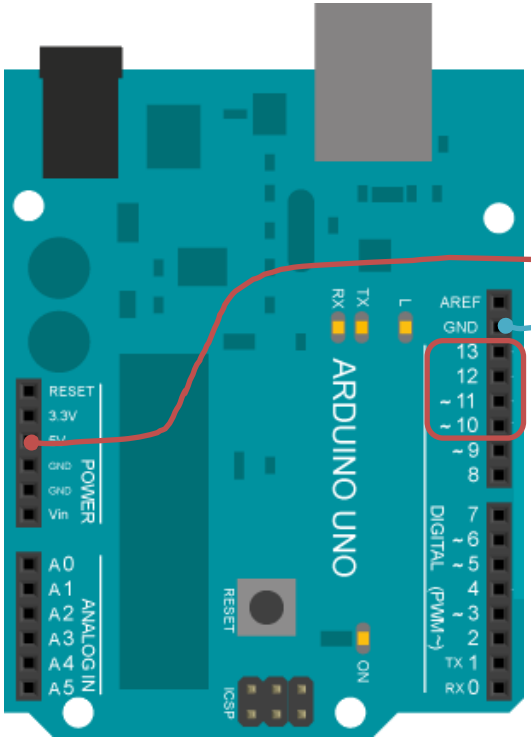
DAC



The LDAC input can be used to select the device, and you could use a GPIO pin to turn the device on and off through this pin. In this example, we just tie it to ground so it is always selected and powered.

$V_{SS} = 5V$

$V_{DD} = 0V$



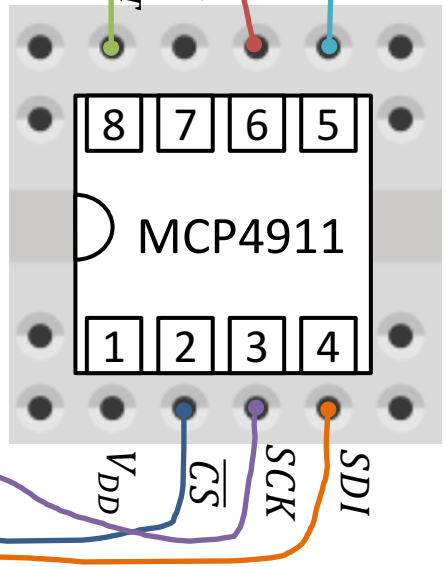
Analog Out (0-5V)

SCK (13)

MISO (12)

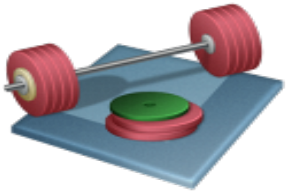
MOSI (11)

SS (10)



MISO Not Used, since we get nothing back from DAC IC

Analog Out



- Create Analog Out, which should be used for the Control Signal (u)
- Test both Alternative #1 and #2

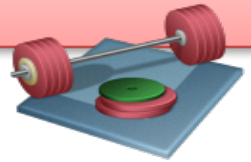
Congratulations! - You are finished with the Task

Arduino Library

Hans-Petter Halvorsen, M.Sc.

Arduino Libraries

Implement your PID, Scaling and Low-pass Filter functions as an Arduino Library



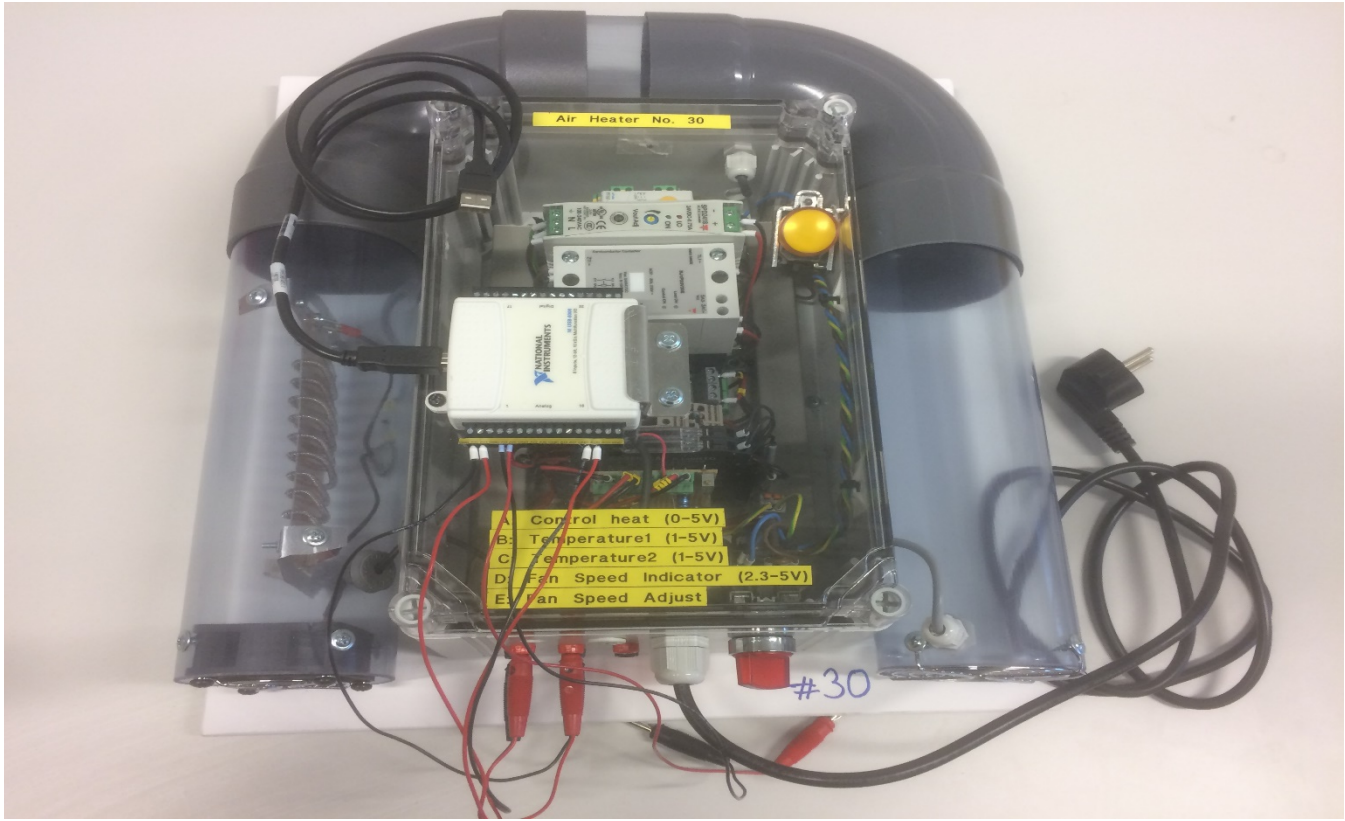
- Arduino Libraries: https://www.arduino.cc/en/Reference/Libraries
- Writing your own libraries: https://www.arduino.cc/en/Hacking/LibraryTutorial

Congratulations! - You are finished with the Task

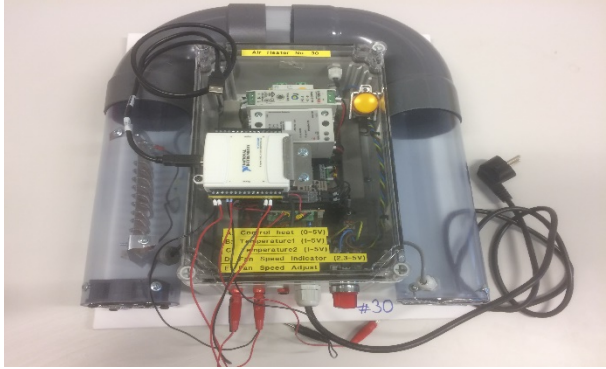
Modelling and Simulation

Hans-Petter Halvorsen, M.Sc.

Air Heater


Air Heater No. 30
Control heat (0-5V)
Temperature1 (1-5V)
Temperature2 (1-5V)
Fan Speed Indicator (2.3-5V)
Fan Speed Adjust
#30

Air Heater Mathematical Model

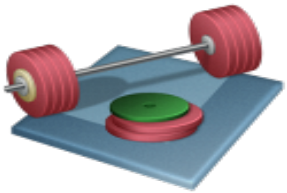


$$\dot{T}_{out} = \frac{1}{\theta_t}\{-T_{out} + [K_h u(t-\theta_d) + T_{env}]\}$$

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)

Model Parameters



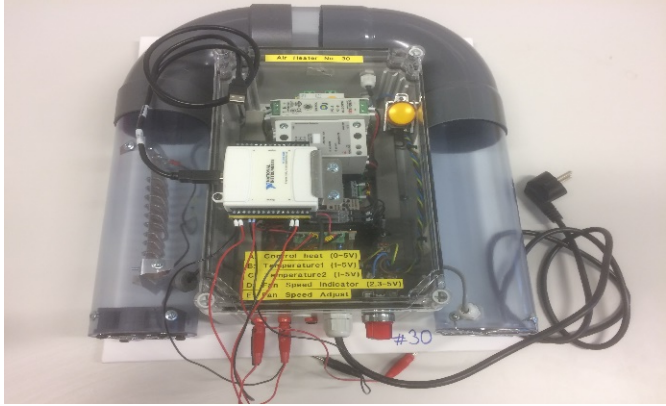
Find Proper Model Parameters using LabVIEW

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

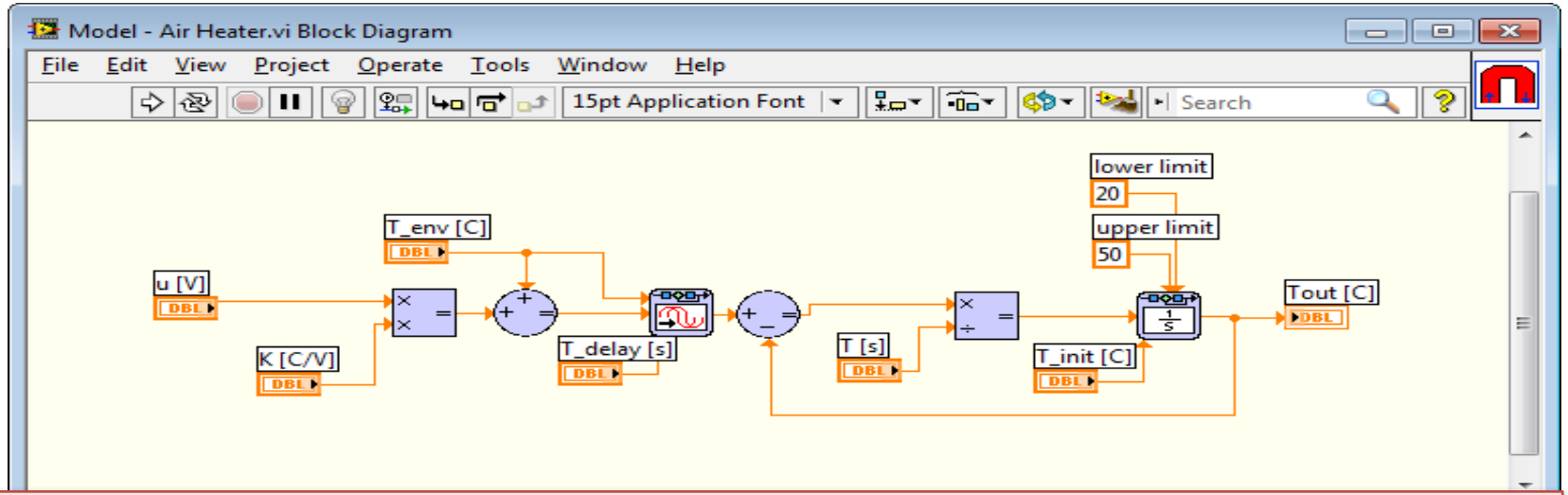
Air Heater in LabVIEW



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:

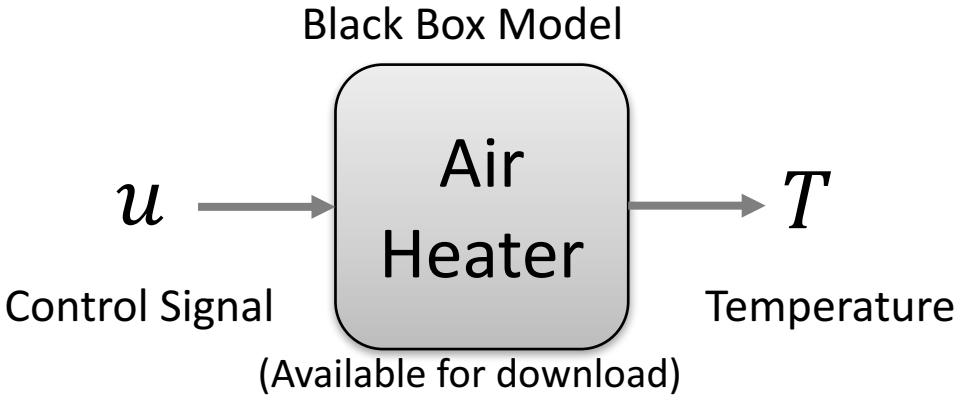


Note! This model is implemented in a so-called "**Simulation Subsystem**" (which is recommended!!!)

“Real Process” → “Black Box Simulator”

- 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”



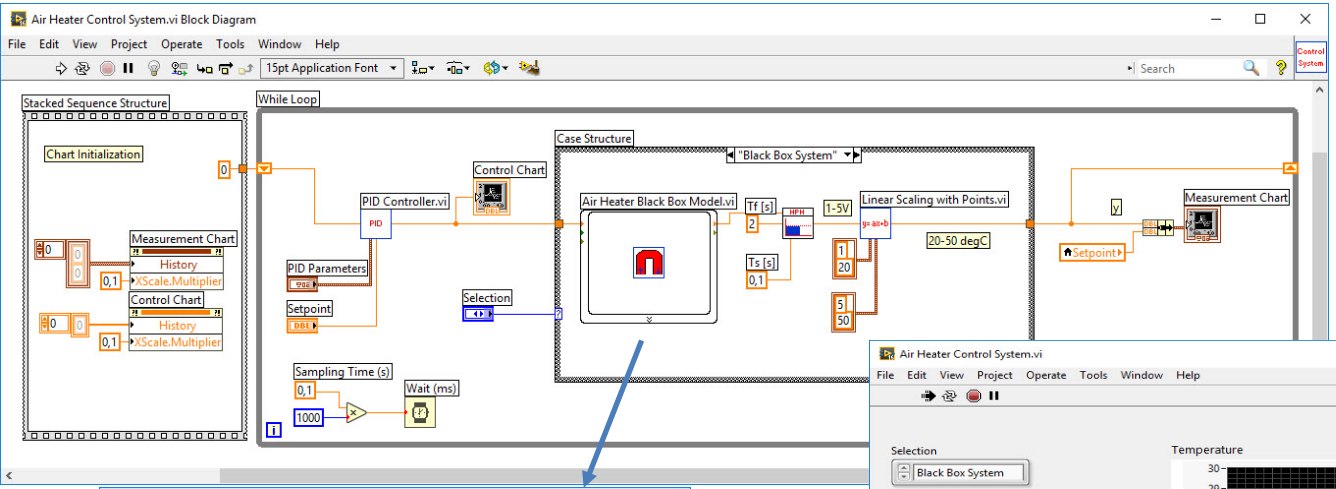
You can assume that the following model is a good representation of the “Black Box Model”:

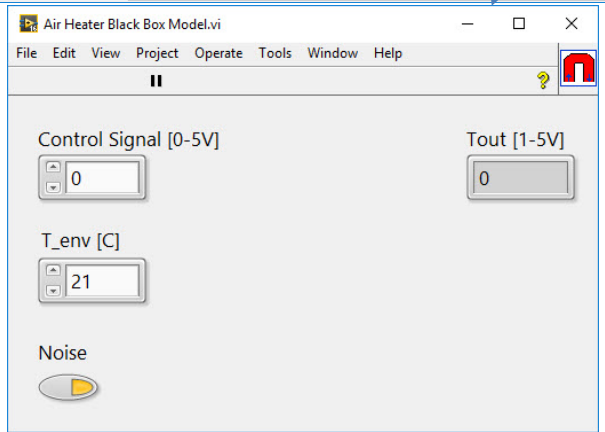
$$\dot{T}_{out} = \frac{1}{\theta_t}\{-T_{out} + [K_h u(t-\theta_d) + T_{env}]\}$$

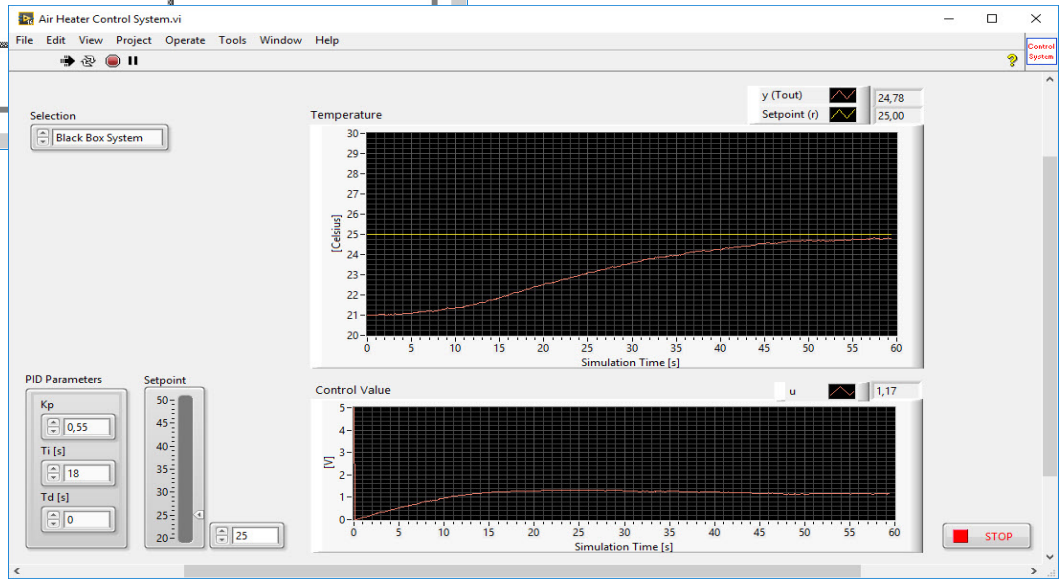
This means you need to need to find $\theta_t, K_h, \theta_d, T_{env}$

T_{env} is the temperature in the room

“Real Process” → “Black Box Simulator”

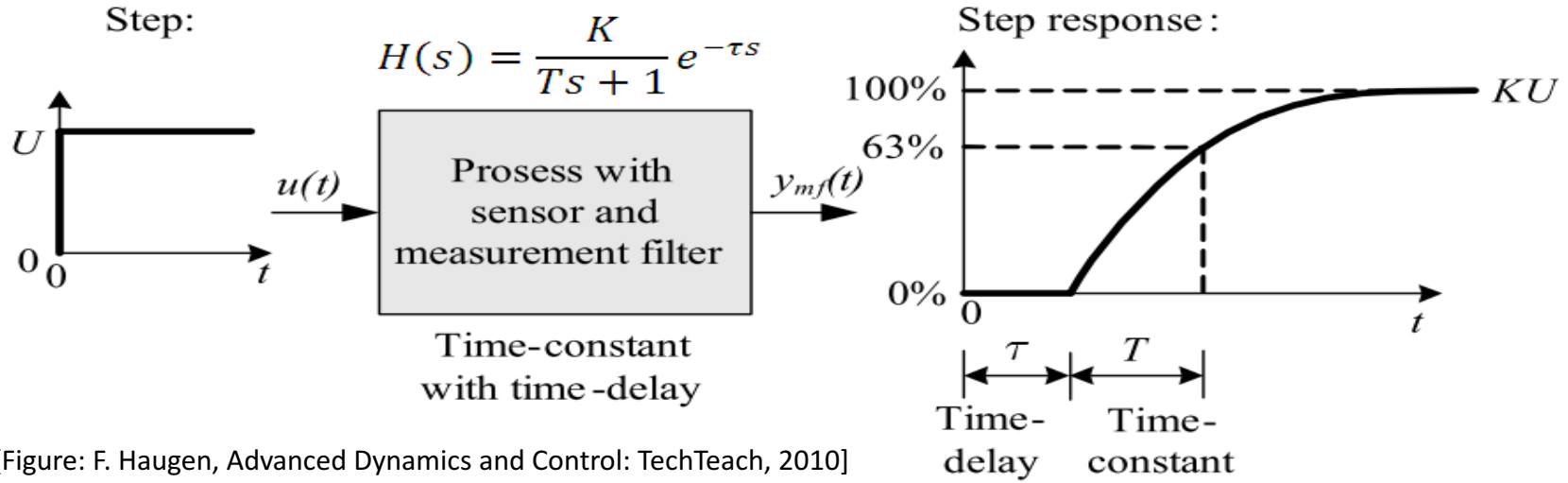






Here we see an example where we control the “Black Box” Model, which we “pretend” is the Real System

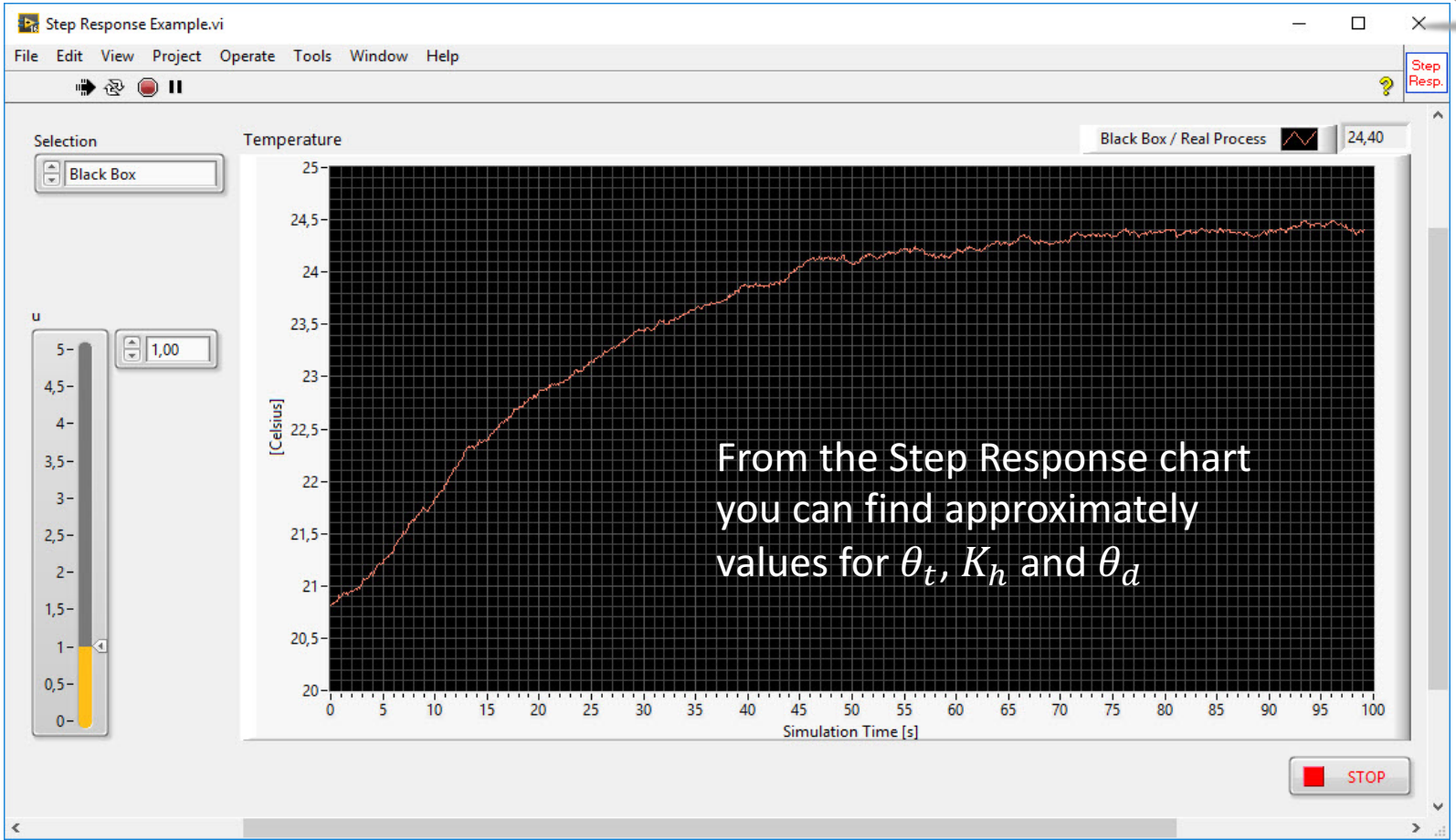
Step Response Method



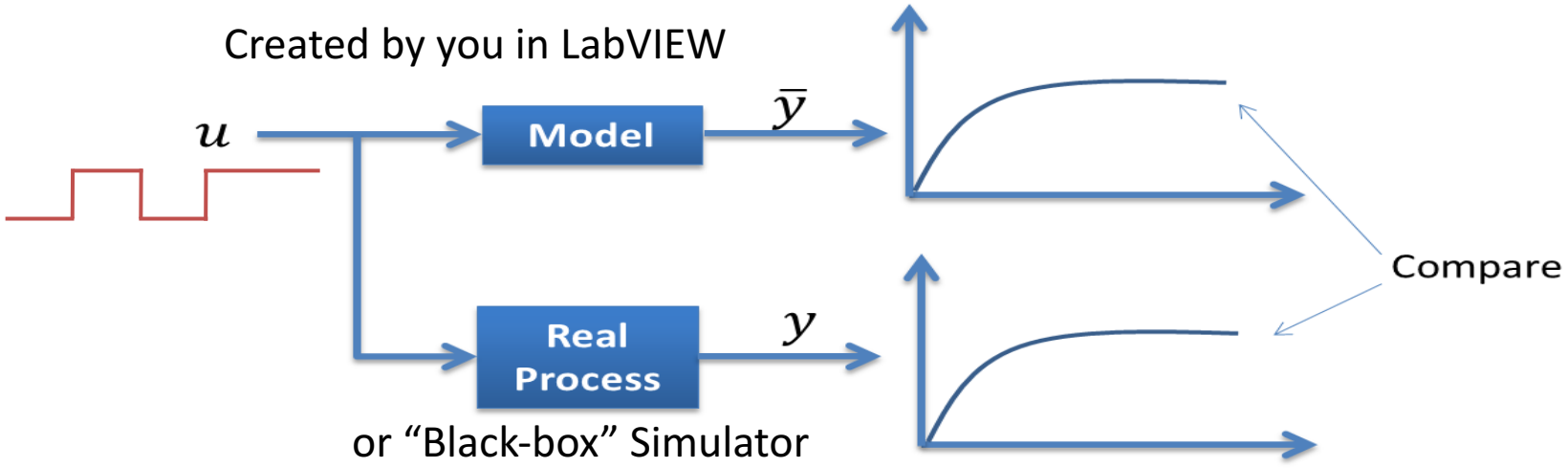
[Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

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)

Step Response Method

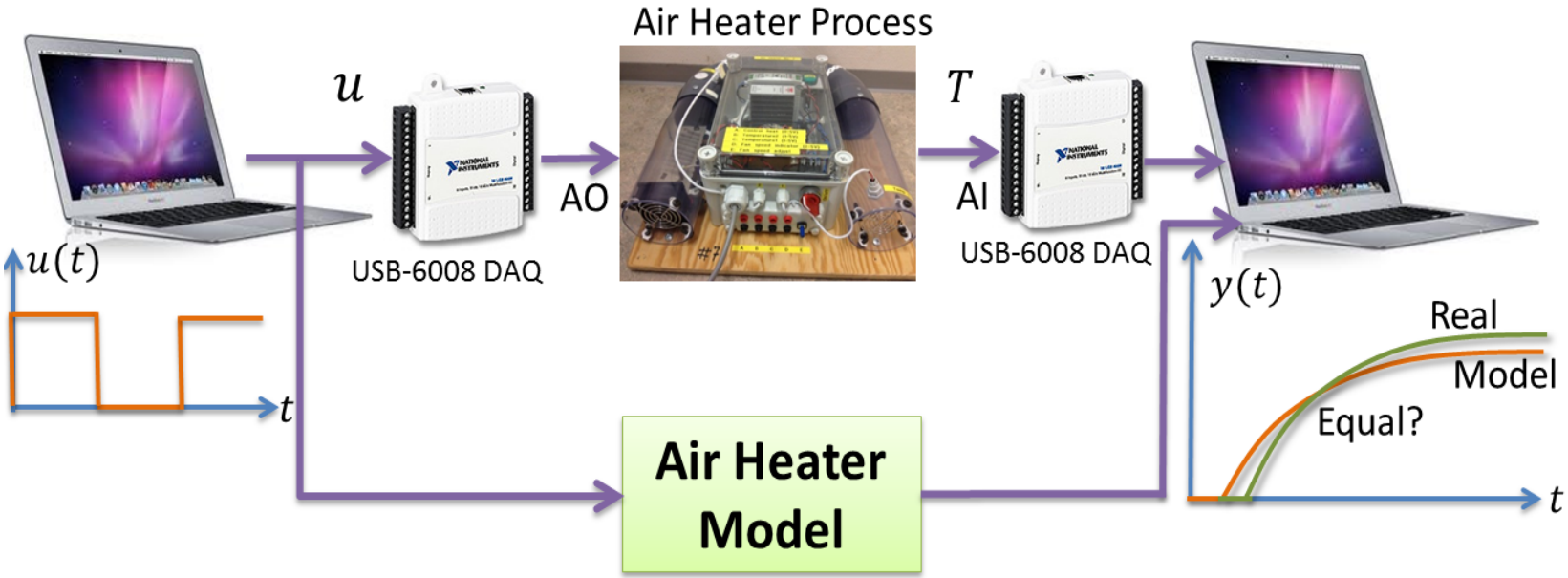


Trial & Error Method



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)

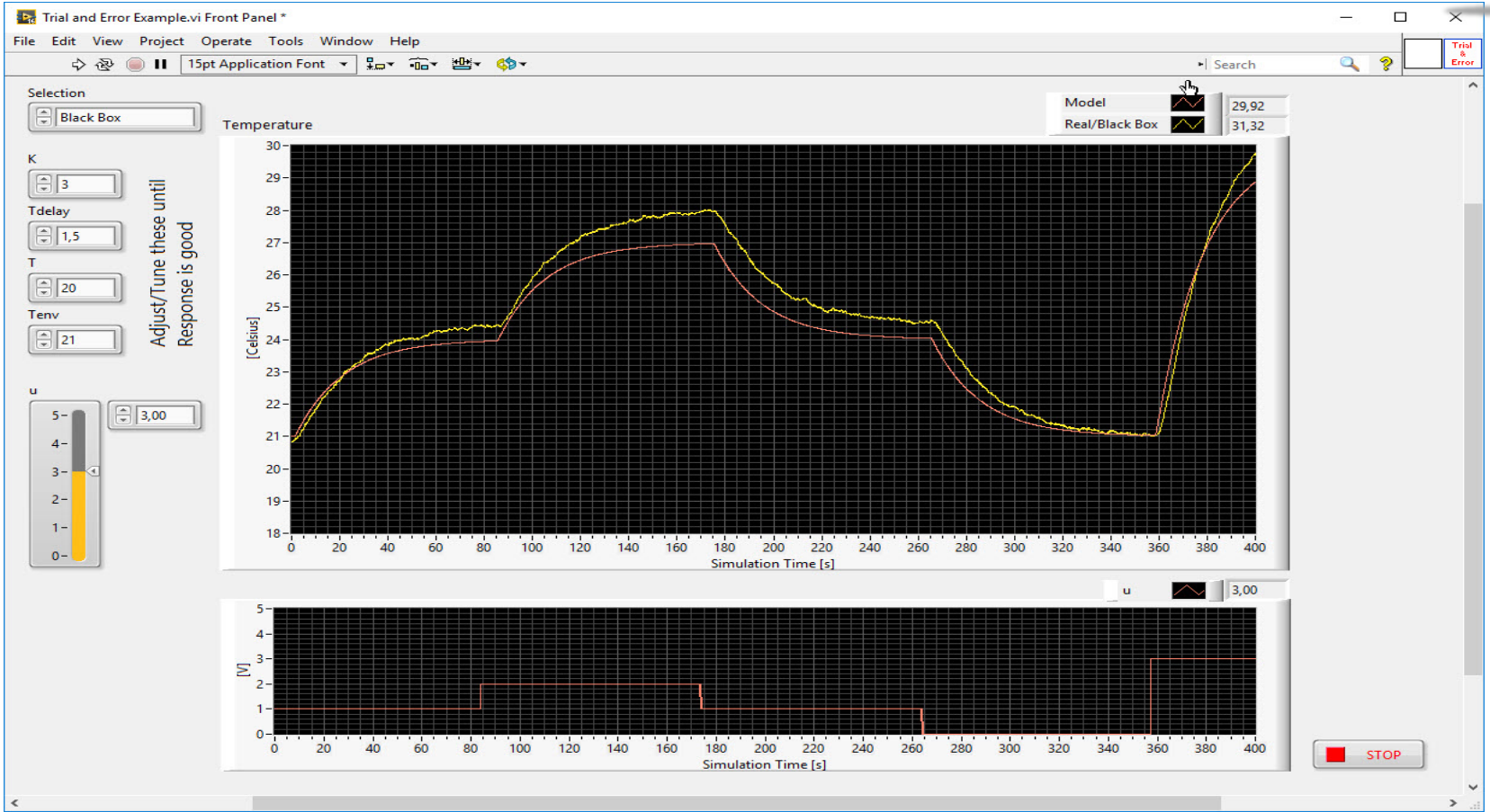
Model Validation



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

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



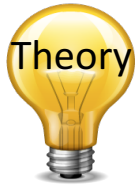
Congratulations! - You are finished with the Task

HIL Simulation and Testing

HIL – Hardware in the Loop

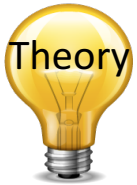
Hans-Petter Halvorsen, M.Sc.

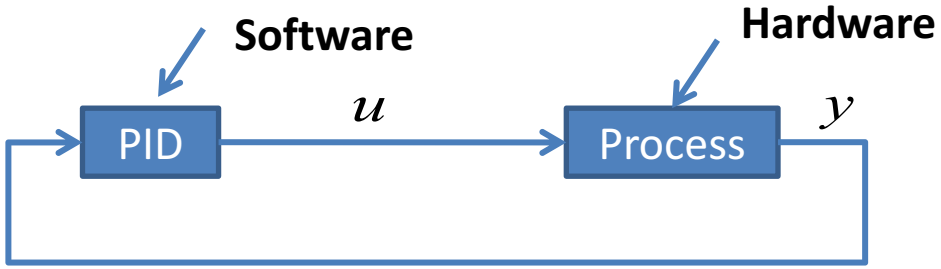
HIL Lab - Background

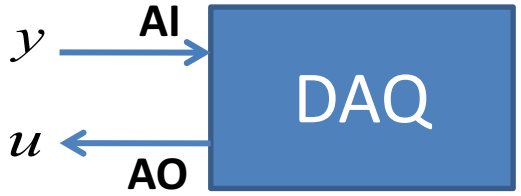


- Typically, a simulator communicates with an “ECU” (“Electronic Control Unit”) via ordinary I/O. Such a system - where the real controller is controlling a simulated process is denoted Hardware-in-the-loop (HIL) simulation.
- **The main purpose of this lab is to test the hardware device on a simulator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

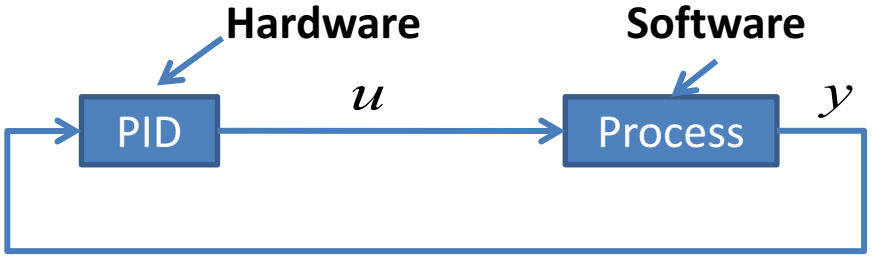
Traditional Process Control using Software for Implementing the Control System

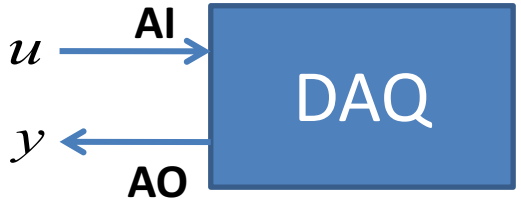




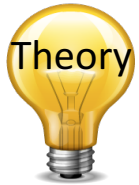


HIL Simulation



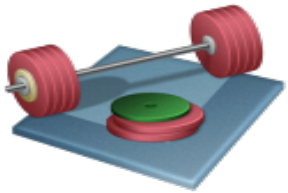


HIL Simulation



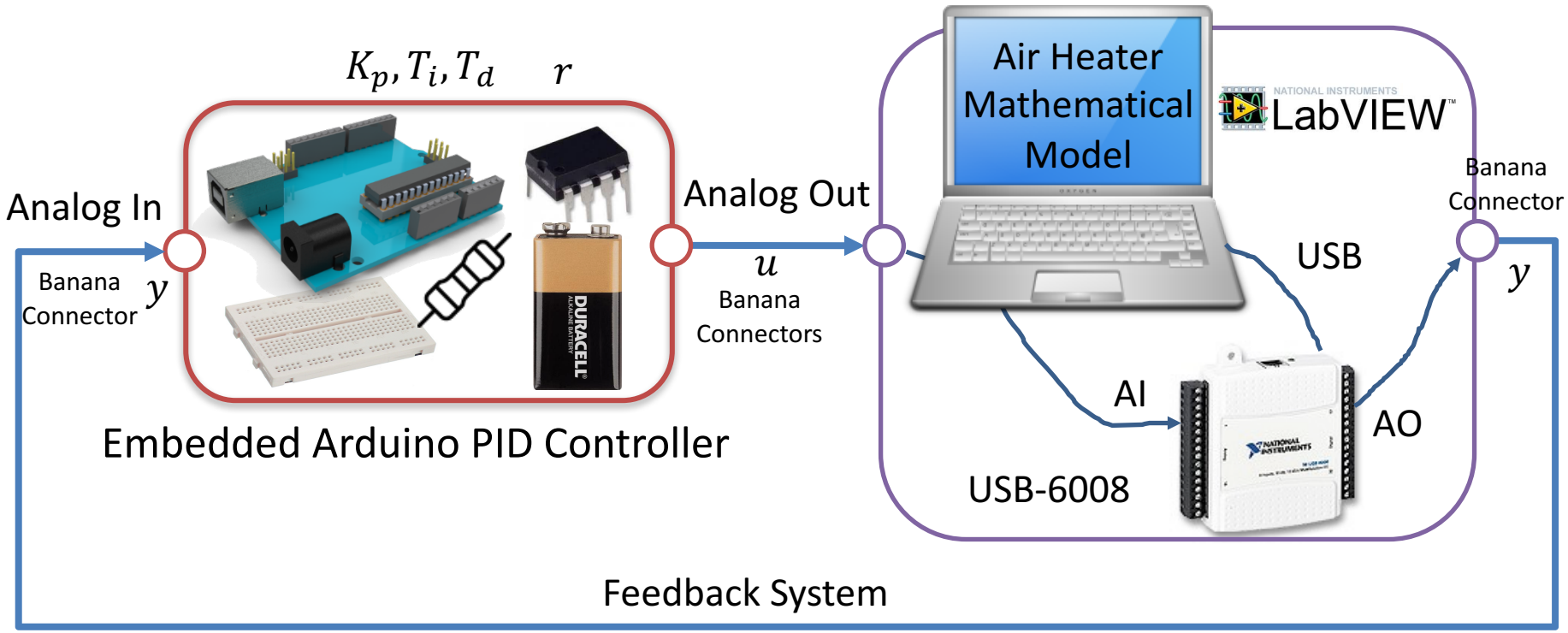
- Hardware-in-the-loop (HIL) simulation is a technique that is used in the development and **test of complex process systems**
- The HIL simulation includes a **mathematical model** of the process and a hardware device/ECU you want to test, e.g. an industrial PID controller we will use in our example. 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 Simulations and Testing

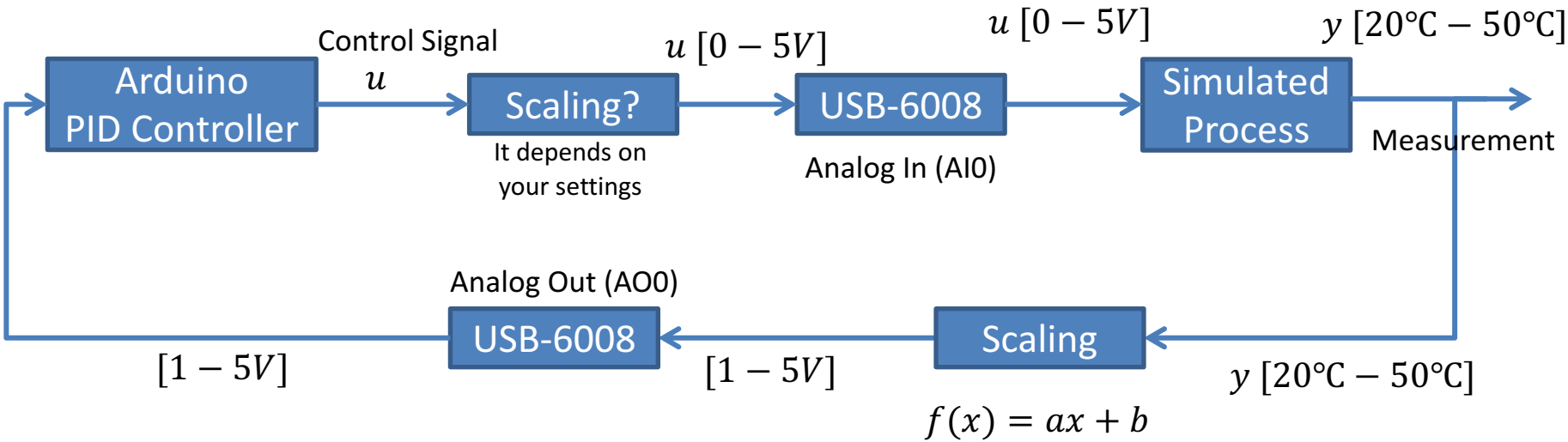


- Make sure to test your Embedded Arduino PID Controller using HIL Simulation and Testing principles before you use it on the Real Air Heater process
- Make sure it works as expected and without risk of damaging the Air Heater process

HIL Simulation and Testing



HIL Simulation using Arduino PID Controller

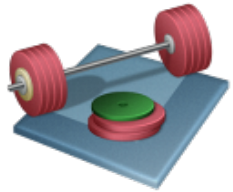


Congratulations! - You are finished with the Task

Controller Design

Hans-Petter Halvorsen, M.Sc.

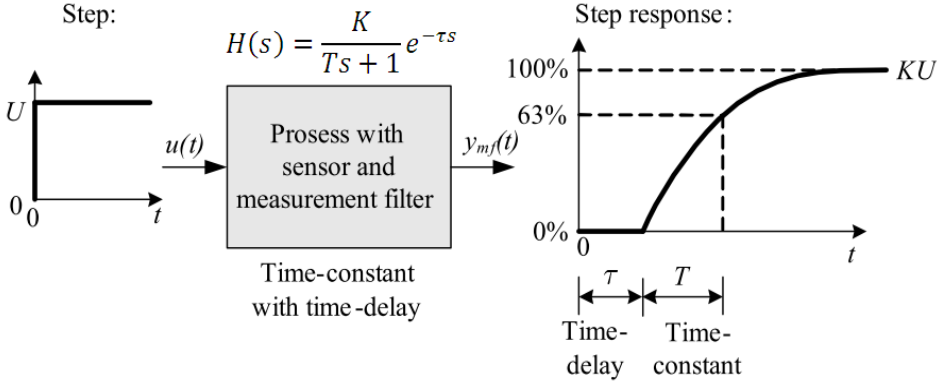
PI(D) Controller Design



- Find Proper PI Parameters
- Use e.g., the Skogestad’s method
- Fine-tune PI Parameters during Simulations and Practical Experiments

Skogestad's method

- 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.



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	$\frac{K}{s^2}e^{-\tau s}$	$\frac{1}{4K(T_C+\tau)^2}$	$4\,(T_C+\tau)$	$4\,(T_C+\tau)$

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).

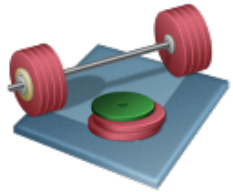
[Figures: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

Congratulations! - You are finished with the Task

Air Heater Control System

Hans-Petter Halvorsen, M.Sc.

Air Heater Control System



- Control the real Air Heater process using your Arduino PID Controller
- Test the Control System and Fine-tune PI(D) Parameters if necessary doing some Practical Experiments
  - Change in Reference
  - Disturbance

System Overview

ARDUINO

Download your Application
and then remove USB cable

K_p, T_i, T_d r

Air Heater

Arduino

DAC

Analog In

y

Banana
Connector

Breadboard

Analog Out

u

Banana
Connector

Banana
Connector

AI

AO

Banana
Connector

y

Embedded Arduino PID Controller

Test first using HIL
Simulation and Testing

Feedback System

Congratulations! - You are finished with the Task

Enhanced Features

Hans-Petter Halvorsen, M.Sc.

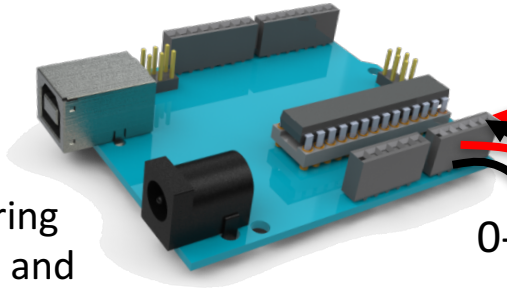
Enhanced Features

Here are some suggestions for those who want to explore more (You don't need to do anything of this!):

- Build a Casing for your Arduino PID controller (make it look like an industrial PID controller)
- Improve PID controller with Anti-windup, etc.
- Possible to remote set the Reference Value (r)
- Possible to remote set the PID Parameters (K_p, T_i, T_d)
- Publish Process Data to a PC (u, y), etc.
- Use a PC for remote Monitoring
- etc.

Arduino PID + Real Air Heater + PC for Monitoring

Arduino PID Controller

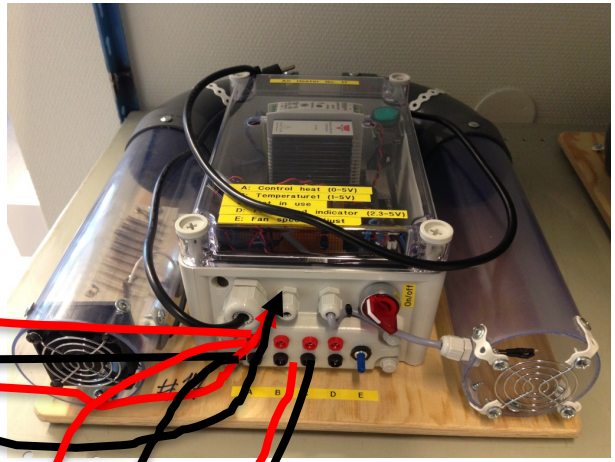


Process Value

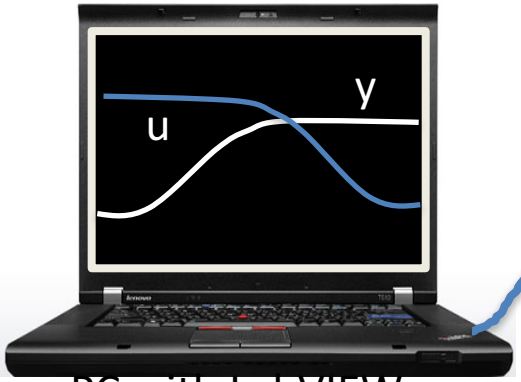
1-5V

0-5V

Control Signal



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



PC with LabVIEW

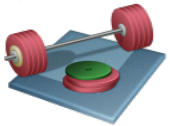
USB

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

Congratulations! - You are finished with the Task

Remote Access and Publishing

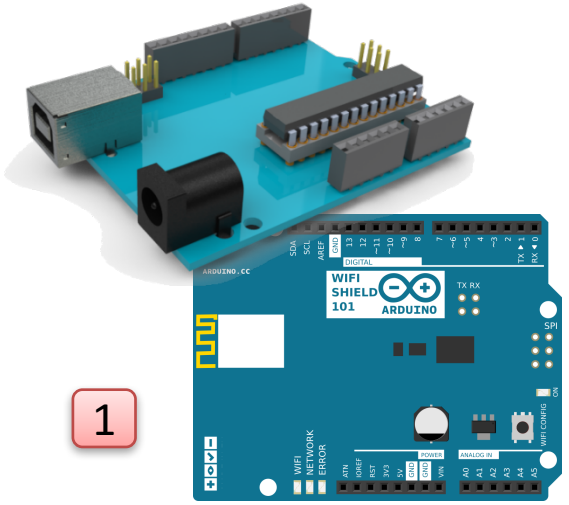
Hans-Petter Halvorsen, M.Sc.

Data Collection

2

ThingSpeak

www.ThingSpeak.com

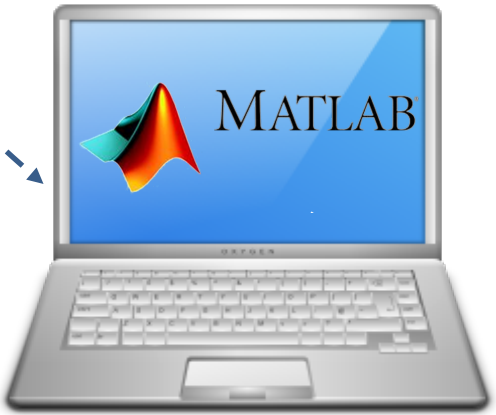


1

u, y

The Cloud

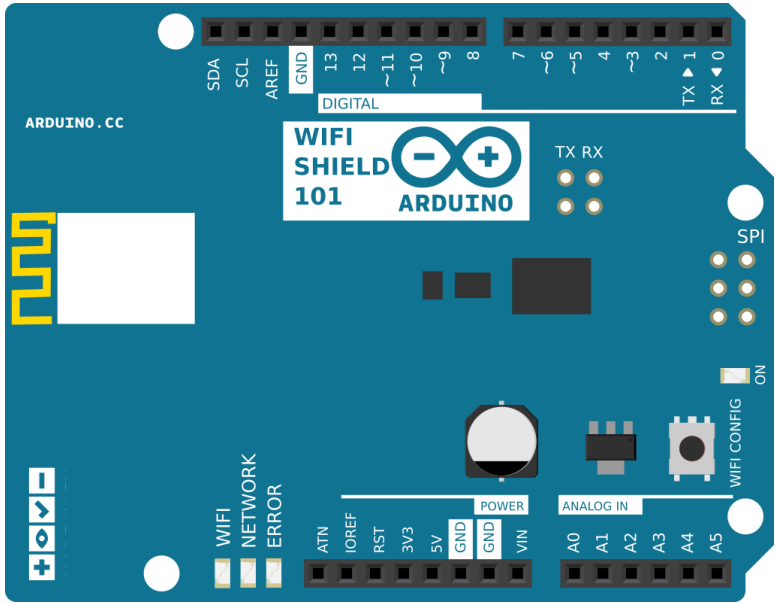
3



Arduino + WiFi Shield

Data Analysis

Arduino Wi-Fi Shield



With a Wi-Fi Shield you can get remote access to your Embedded PID Controller, setting Set-point (r), PID Parameters (K_p, T_i, T_d) and/or you can also publish your Process parameters like Control Values (u) and Measurements (y).

Getting Started with Arduino WiFi Shield 101:
https://www.arduino.cc/en/Guide/ArduinoWiFiShield101

Arduino WiFi Shield + ThingSpeak

- ThingSpeak is a free Web Service (REST API) that lets you collect and store sensor data in the cloud and develop Internet of Things applications.
- It works with Arduino, Raspberry Pi and MATLAB

https://thingspeak.com

Arduino Example:
https://www.arduino.cc/en/Tutorial.WiFi101ThingSpeakDataUploader

ThingSpeak + MATLAB

The “ThingSpeak Support Toolbox” lets you use desktop MATLAB to analyze and visualize data stored on ThingSpeak.com

ThingSpeak Support from Desktop MATLAB:

http://se.mathworks.com/hardware-support/thingspeak.html

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

Hans-Petter Halvorsen, M.Sc.

University College of Southeast Norway

www.usn.no

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

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