https://www.halvorsen.blog

Arduino Control System

Hans-Petter Halvorsen

Table of Contents

Introduction

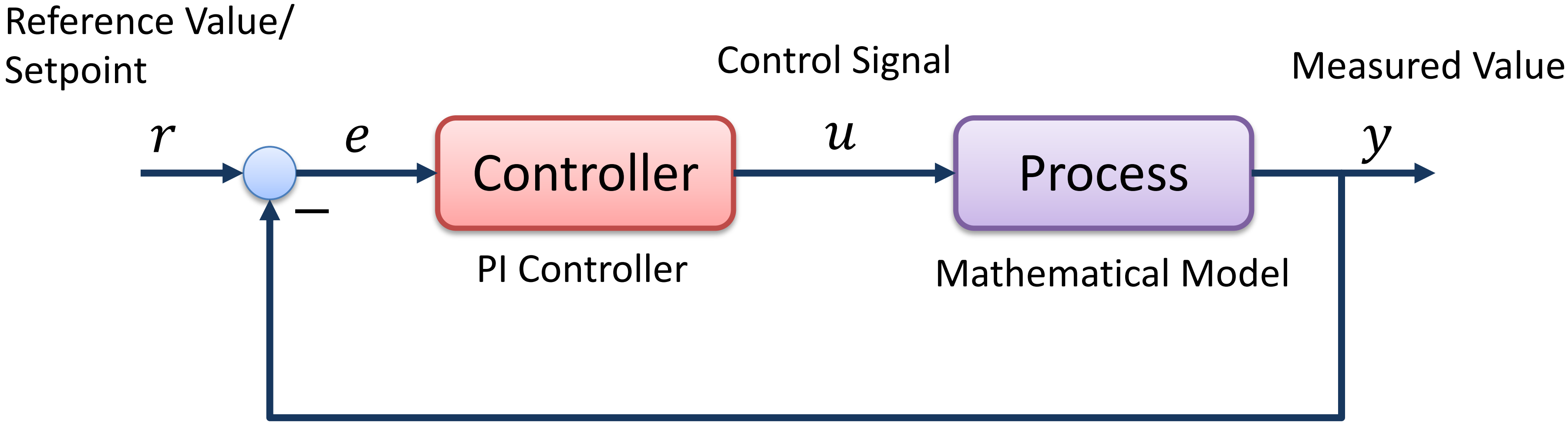
Hans-Petter Halvorsen

Table of Contents

Introduction

- We will create a basic Control System using Arduino
- This Tutorial uses **Arduino UNO**, but other Arduino devices may be used
- We will implement a simple **PI Controller**
- We will implement a **Mathematical Model** which we will **simulate** and control using the PI Controller
- Finally, we will also implement a **Lowpass Filter**

Arduino Control System



Arduino

Hans-Petter Halvorsen

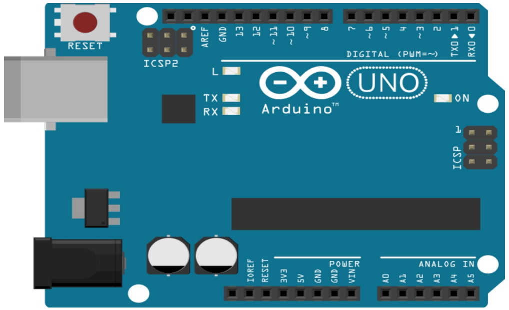
Table of Contents

Arduino

- Arduino is an open-source electronics platform based on easy-to-use hardware and software.
- It's intended for anyone making interactive projects, from kids to grown-ups.
- You can connect different Sensors, like Temperature, etc.
- It is used a lots in Internet of Things projects
- Homepage: https://www.arduino.cc

Arduino

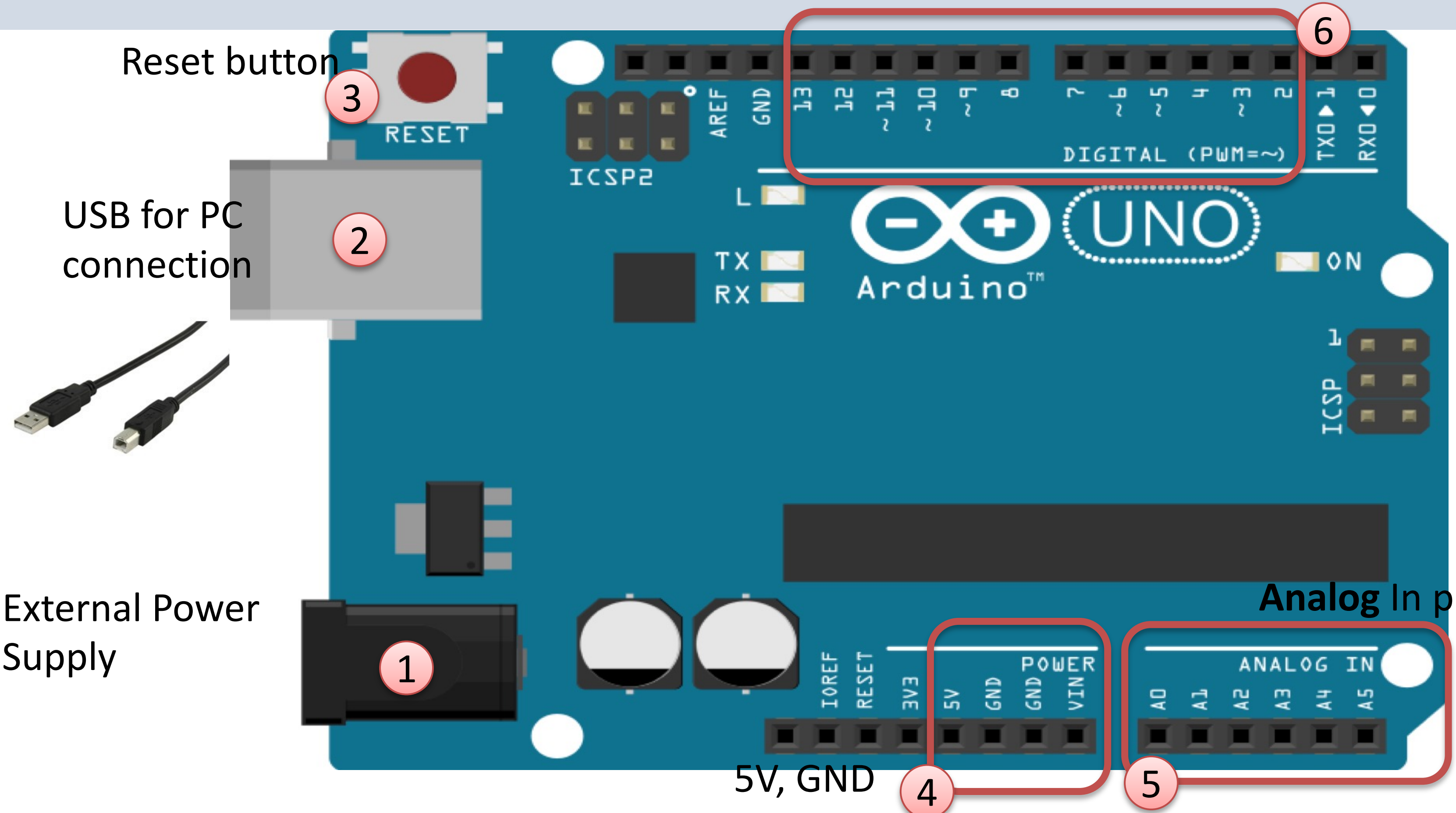
- Arduino is a Microcontroller
- Arduino is an open-source platform with Input/Output Pins (Digital In/Out, Analog In and PWM)
- Price about $20
- Arduino Starter Kit ~$40-80
  with Cables, Wires, Resistors, Sensors, etc.



Arduino

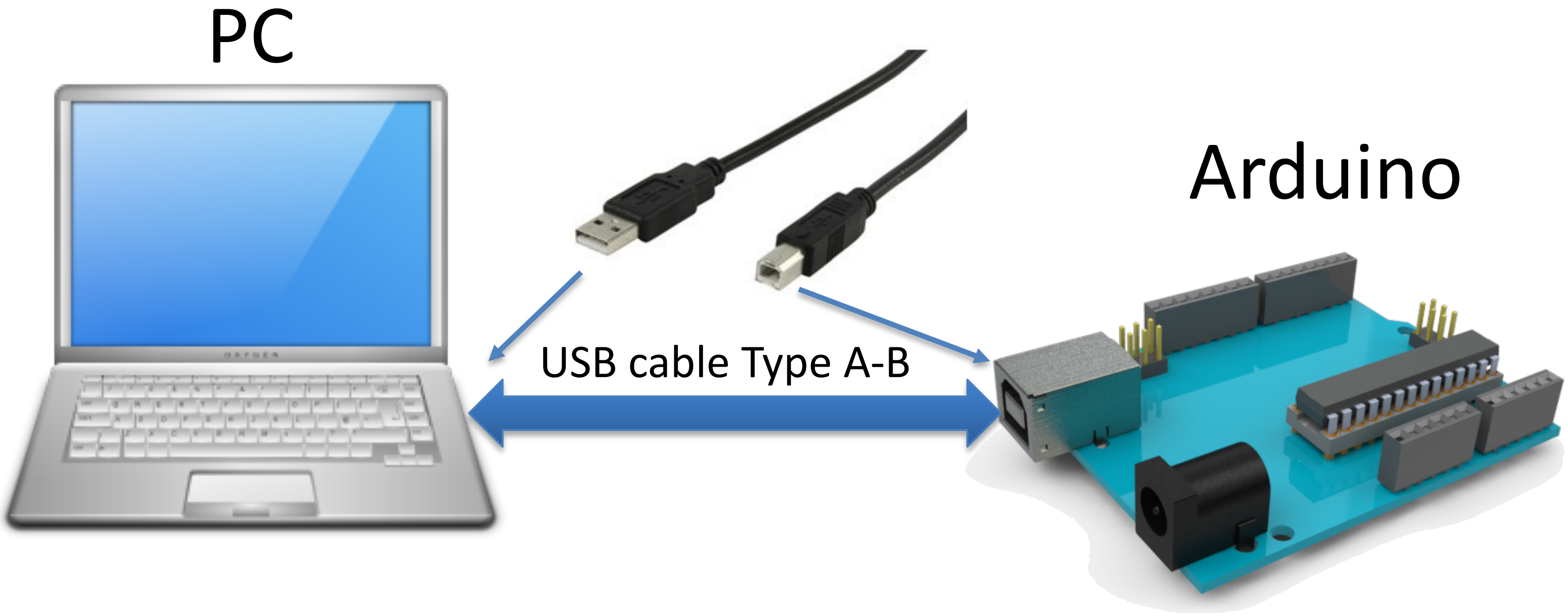
- Lots of different Arduino boards exists
- There are different Arduino boards with different features and boards that are tailormade for different applications
- https://www.arduino.cc/en/Main/Products
- The most common is called “Arduino UNO”

Arduino UNO



Digital ports (2-13)
6
Reset button
3
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX0▶1
RX0◀0
DIGITAL (PWM=~)
L
USB for PC connection
2
TX
RX
UNO
Arduino™
ON
1
ICSP
External Power Supply
1
Analog In ports (0-5)
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
5V, GND
4
5

Connect Arduino to your PC



Arduino Software

Upload Code to Arduino Board

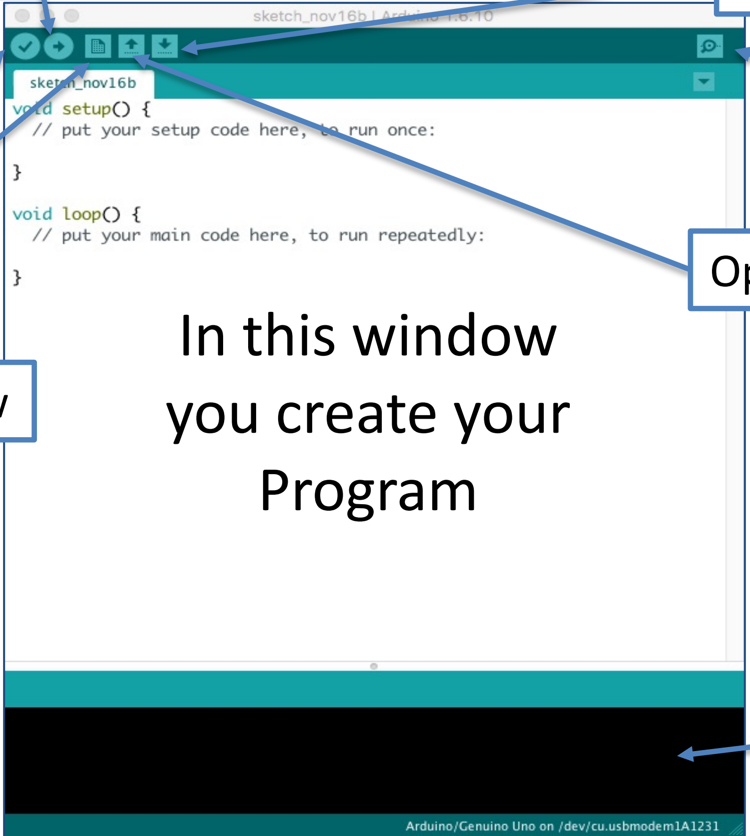
Save

Open Serial Monitor

Compile and Check if Code is OK

Open existing Code

Creates a New Code Window



In this window you create your Program

Error Messages can be seen here

The software can be downloaded for free:

www.arduino.cc

Arduino Programs

All Arduino programs must follow the following main structure:

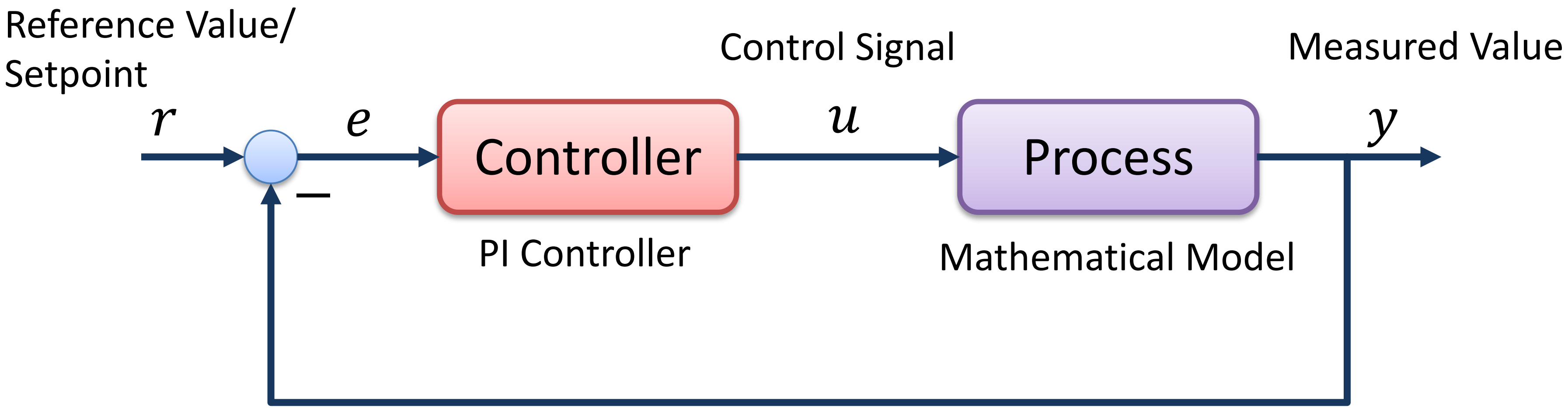
```
// Initialization, define variables, etc.

void setup()
{
      // Initialization
      ...
}

void loop()
{
      //Main Program
      ...
}
```

Control System

Hans-Petter Halvorsen

Table of Contents

Arduino Control System



Code

Here you see an example of the main code structure of your application

The Code for the PI Controller, the Process Model, etc. should be put into separate Functions

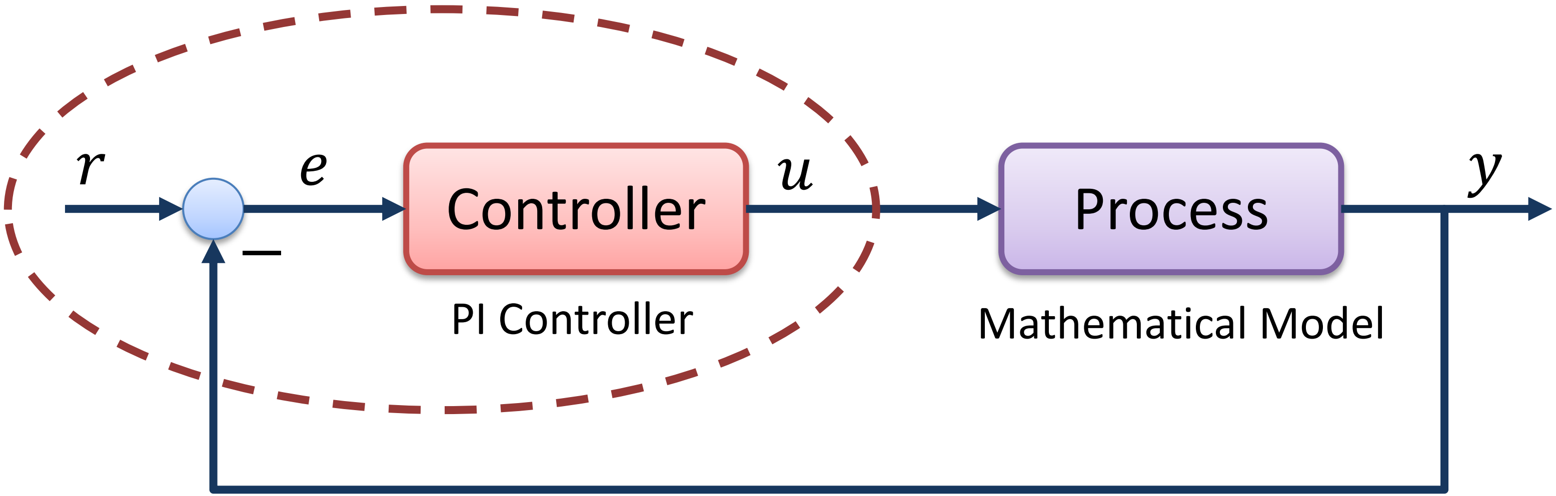
```
// Initialization
..
void setup()
{
    // Initialization
    ..
}

void loop()
{
  PiController();
  ProcessModel();
  delay(wait)
}
```

PI Controller

Hans-Petter Halvorsen

Table of Contents

Arduino Control System



PID Controller

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

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.]

PI Controller

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

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 $[sec.]$

Discrete PI controller

We start with the continuous PI Controller:

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

We derive both sides in order to remove the Integral:

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

We can use the Euler Backward Discretization method:

$$\dot{x} \approx \frac{x(k) - x(k-1)}{T_s}$$

Where T_s is the Sampling Time

Then we get:

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

Finally, we get:

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

Where $e_k = r_k - y_k$

PI Controller Code Example

```
void PiController()
{
  u_prev = u;
  e = r - Tout;
  u = u_prev + Kp*(e - e_prev) + (Kp/Ti)*Ts*e;
  if (u < 0)
    u = 0;
  if (u > 5)
    u = 5;
}
```

Note! This is a very basic example

```
//Controller
float r = 24;
float Kp = 0.8;
float Ti = 20;
float u = 0;
..
```

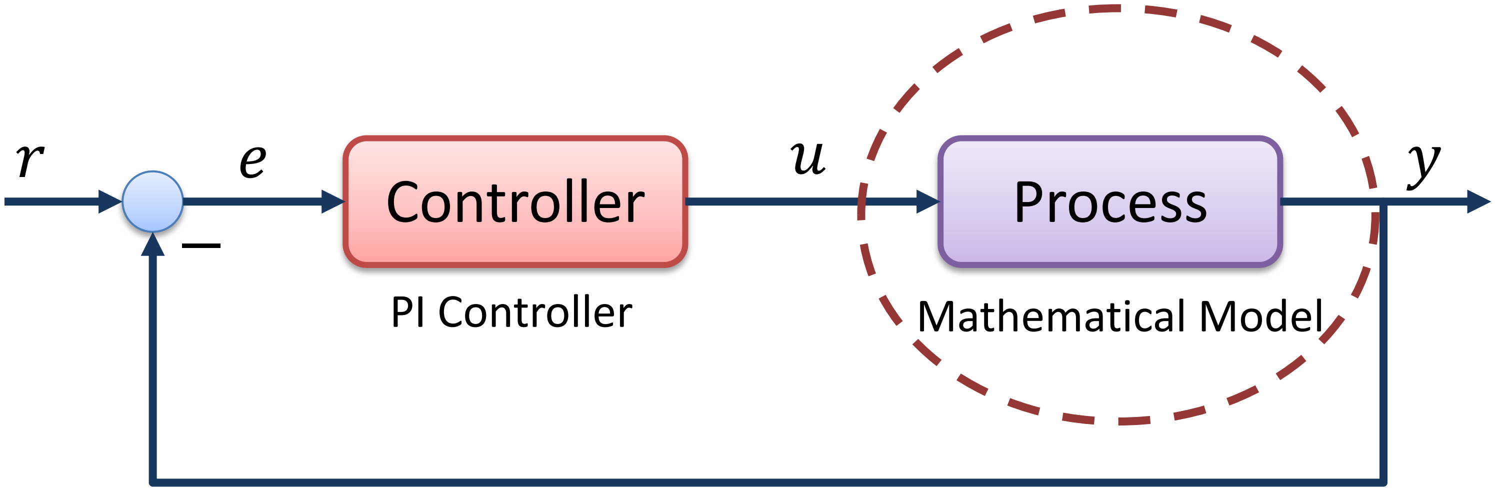
The variables are in this basic example set as global variables on top in the Arduino program

Process and Mathematical Model

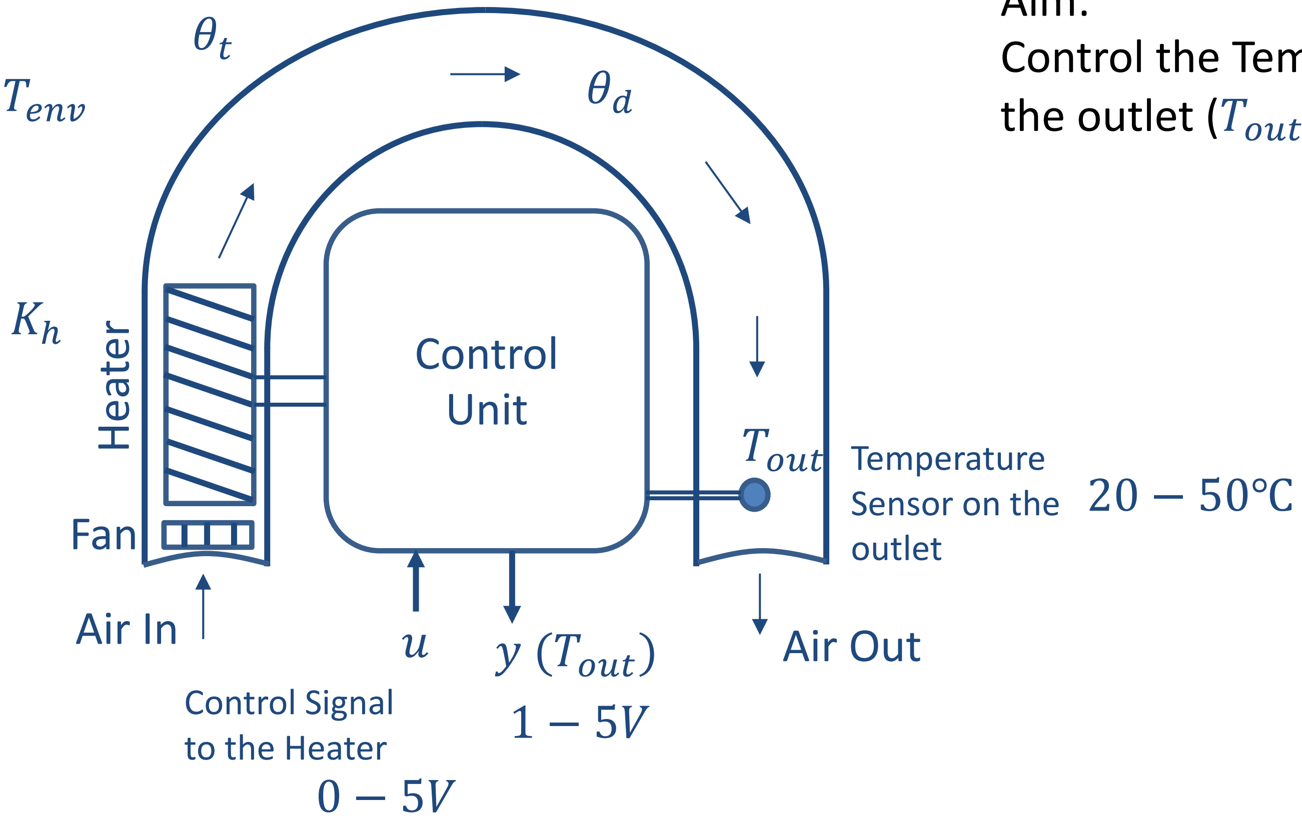
Hans-Petter Halvorsen

Table of Contents

Arduino Control System

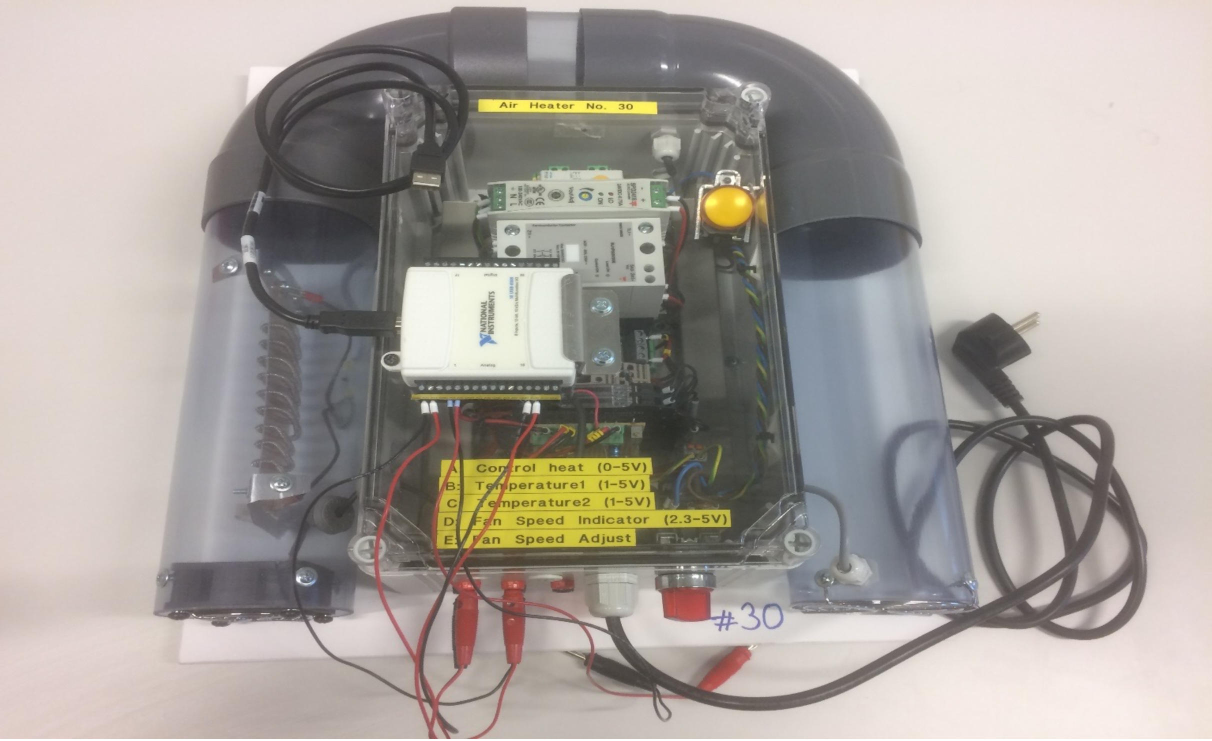


Air Heater System



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

Air Heater System



We can, e.g., use the following values in the simulation:

$\theta_t = 22\ s$

$\theta_d = 2\ s$

$K_h = 3.5\ \frac{°C}{V}$

$T_{env} = 21.5\ °C$

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

Discrete Air Heater

Continuous Model:

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

We can use e.g., the Euler Approximation in order to find the discrete Model:

$$\dot{x} \approx \frac{x(k+1) - x(k)}{T_s}$$

T_s - Sampling Time

$x(k)$ - Present value

$x(k+1)$ - Next (future) value

The discrete Model will then be on the form:

$$x(k+1) = x(k) + \ \ldots$$

We can then implement the discrete model in any programming language

Discrete Air Heater

We make a discrete version:

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

$$\frac{T_{out}(k+1) - T_{out}(k)}{T_s} = \frac{1}{\theta_t}\{-T_{out}(k) + [K_h u(k - \theta_d) + T_{env}]\}$$

This gives the following discrete system:

$$T_{out}(k+1) = T_{out}(k) + \frac{T_s}{\theta_t}\{-T_{out}(k) + [K_h u(k - \theta_d) + T_{env}]\}$$

The Time delay θ_d makes it a little complicated. We can simplify by setting $\theta_d = 0$

$$T_{out}(k+1) = T_{out}(k) + \frac{T_s}{\theta_t}\{-T_{out}(k) + [K_h u(k) + T_{env}]\}$$

Discrete Air Heater (Simplified)

Discrete version with Time delay $\theta_d = 0$

$$T_{out}(k+1) = T_{out}(k) + \frac{T_s}{\theta_t}\{-T_{out}(k) + [K_h u(k) + T_{env}]\}$$

We can use the following values in the simulation:

$$\theta_t = 22s$$
$$K_h = 3.5\,\frac{°C}{V}$$
$$T_{env} = 21.5°C$$

We can set the Sampling Time $T_s = 0.1s$

Process Model

```
void AirHeaterModel()
{
  Tout_prev = Tout;
  Tout = Tout_prev + (Ts/theta_t) * (-Tout_prev + Kh*u + Tenv);
}
```

The variables are in this basic example set as global variables on top in the Arduino program

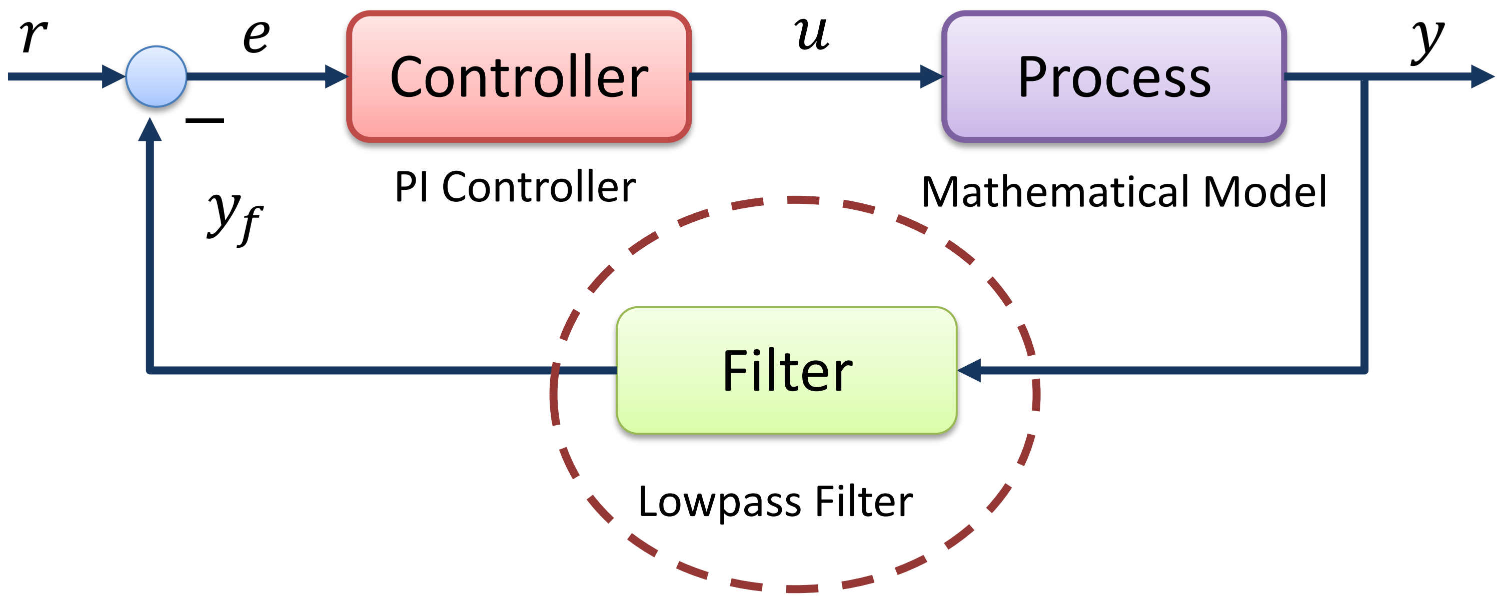
```
// Air Heater Model
float Kh = 3.5;
float theta_t = 22;
float theta_d = 2;
float Tenv = 21.5;
float Tout = Tenv;
float Tout_prev = Tenv;
```

Lowpass Filter

Hans-Petter Halvorsen

Table of Contents

Arduino Control System



Lowpass Filter

The Transfer Function for a Low-pass filter is given by:

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

Where:
y is the Signal from the DAQ device (that contains noise)
y_f is the Filtered Signal
T_f is the Filter Time Constant

Why Lowpass Filter?

- In Measurement systems and Control Systems we typically need to deal with noise
- Noise is something we typically don’t want
- Lowpass Filters are used to remove noise from the measured signals
- Noise is high-frequency signals
- A Lowpass Filter make sure the low frequencies pass (the measurements) and removes the high frequencies (the noise)

Discrete Lowpass Filter

Lowpass Filter:

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

We can find the Differential Equation for this filter using Inverse Laplace:

$$T_f \dot{y}_f + y_f = y$$

We use Euler Backward method: $\dot{x} \approx \frac{x(k) - x(k-1)}{T_s}$

Then we get:

$$T_f \frac{y_f(k) - y_f(k-1)}{T_s} + y_f(k) = y(k)$$

This gives: $y_f(k) = \frac{T_f}{T_f + T_s} y_f(k-1) + \frac{T_s}{T_f + T_s} y(k)$

We define:

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

Finally, we get the following discrete version of the Lowpass Filter:

$$y_f(k) = (1-a) y_f(k-1) + a y(k)$$

This equation can easily be implemented using the Arduino software or another programming language

Golden rule for selecting proper T_f:

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

Lowpass Filter

```
void LowPassFilter()
{
  y = Tout;
  yf = (1-a)*yf_prev + a*y;
  yf_prev = yf;
  Tout = yf;
}
```

Note! This is a very basic example

```
//Filter
float Tf = 5*Ts;
float a = Ts/(Tf+Ts);
float y;
float yf;
float yf_prev = Tout;
```

The variables are in this basic example set as global variables on top in the Arduino program

Final Control System Implementation

Hans-Petter Halvorsen

Table of Contents

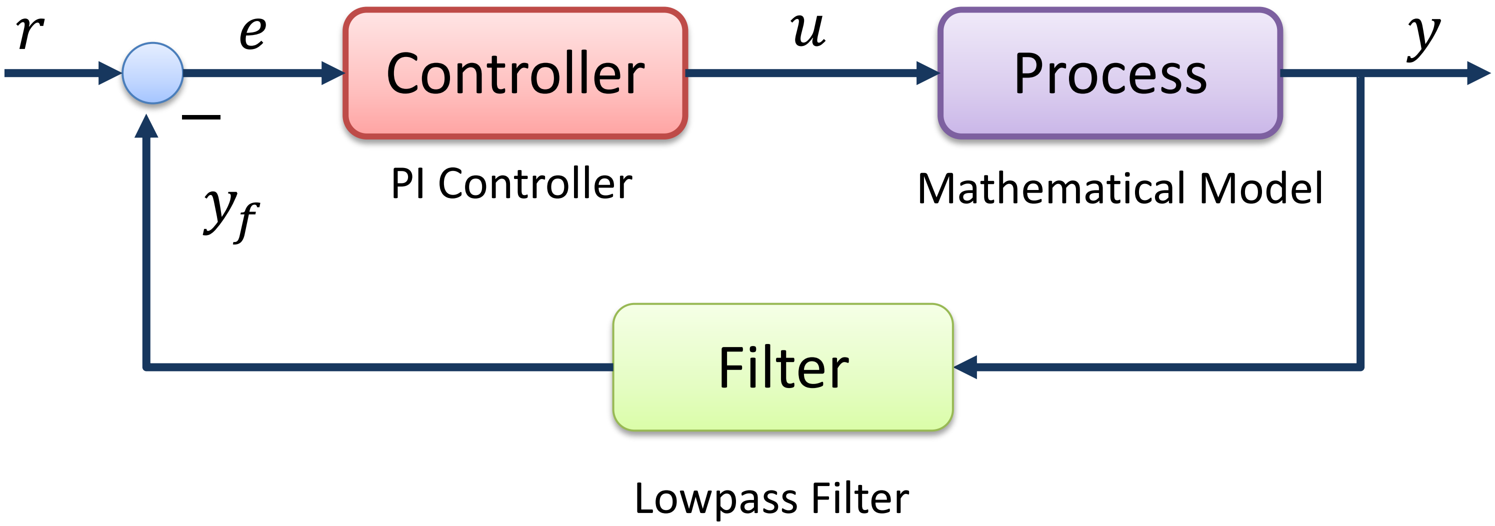
Code

Here you see an example of the main code structure of your application

The Code for the PI Controller, the Process Model and Lowpass Filter have been put into separate Functions

```
// Initialization
..
void setup()
{
    // Initialization
   ..
}

void loop()
{
   PiController();
   ProcessModel();
   LowPassFilter();
   delay(wait)
}
```

Arduino Control System



Control System

air_heater_simulation | Arduino 1.8.13

File Edit Sketch Tools Help

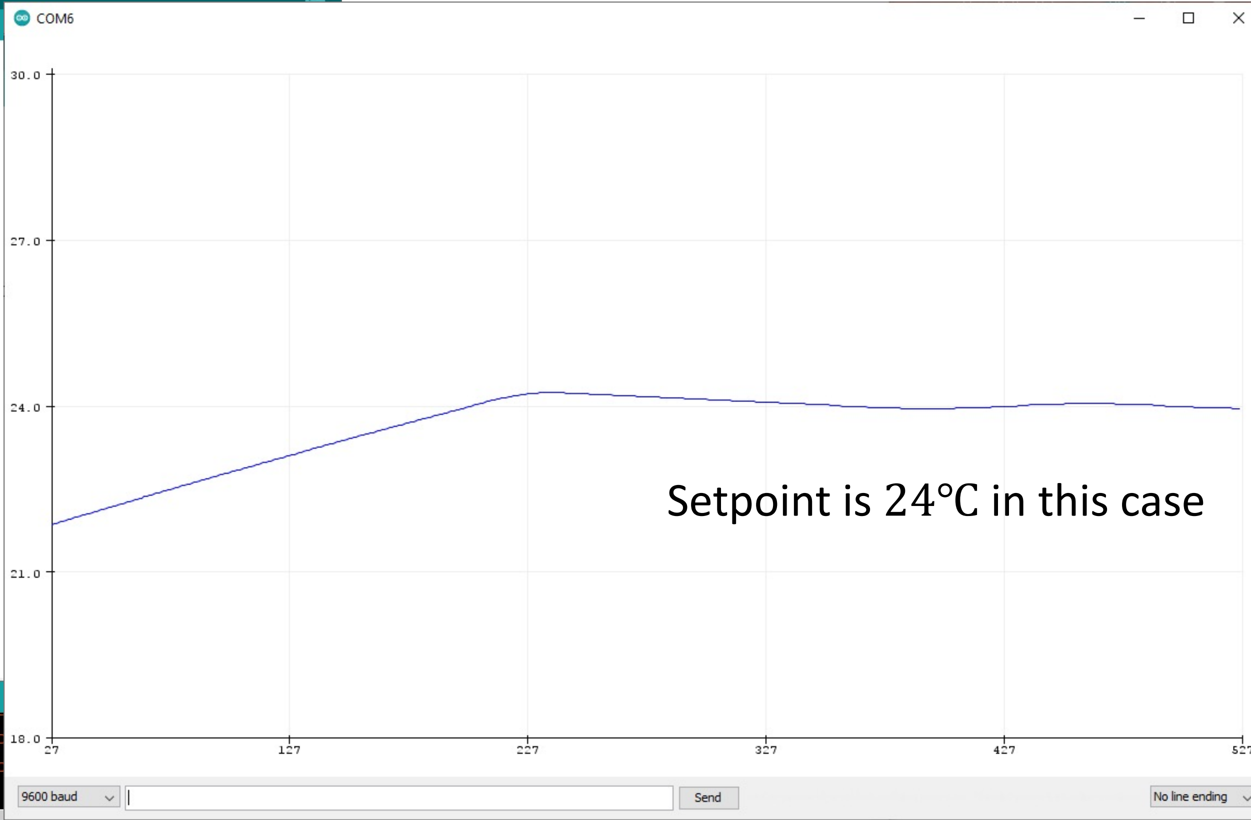
air_heater_simulation §

```
//Filter
float Tf = 5*Ts;
float a = Ts/(Tf+Ts);
float y;
float yf;
float yf_prev = Tout;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  PiController();
  AirHeaterModel();
  LowPassFilter();
  SerialPlotter();
  k = k + 1;
  delay(wait);
}
```

Done Saving.

```
This behaviour is deprecated and will result in an erro
You probably want to use 0xcd instead of 0xc9 (double c
```

Serial Plotter



Setpoint is 24°C in this case

Summary

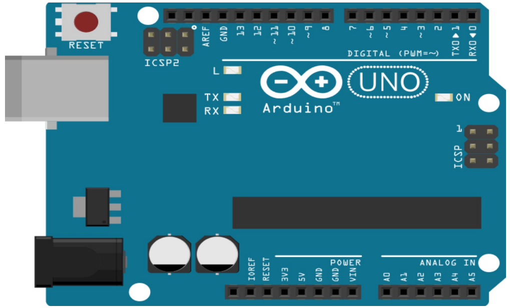
- We have made a simple Control System with Arduino.
- The Code Examples are very simplified and lots of improvements can be made, e.g., reduce the use of global variables, etc.
- You should also structure the code into Classes and make an Arduino Library for the general PI and Lowpass Functions.
- You can add features for storing the data to either an SD card, to a cloud service, etc.
- The Arduino has no Graphical User Interface (GUI), so the user cannot set Setpoint, Kp, Ti, etc. during execution. Here you can use a Cloud Service, create a Web Application, etc.
- The final step is to use Arduino to Control the Real System and not only a simulation where the mathematical model is used.

Real Control System Example

Arduino has NO Analog output pins, so an external DAC is needed

DAC

PI Controller in
Arduino software



Control Signal

u

Real
Process

y

Process Output

The real process could be the Air Heater system shown in this tutorial, a water tank, etc.

Lowpass Filter in
Arduino software

Arduino has Analog Input pins so reading the process value is no problem

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog