https://www.halvorsen.blog

Arduino Control Library

Hans-Petter Halvorsen

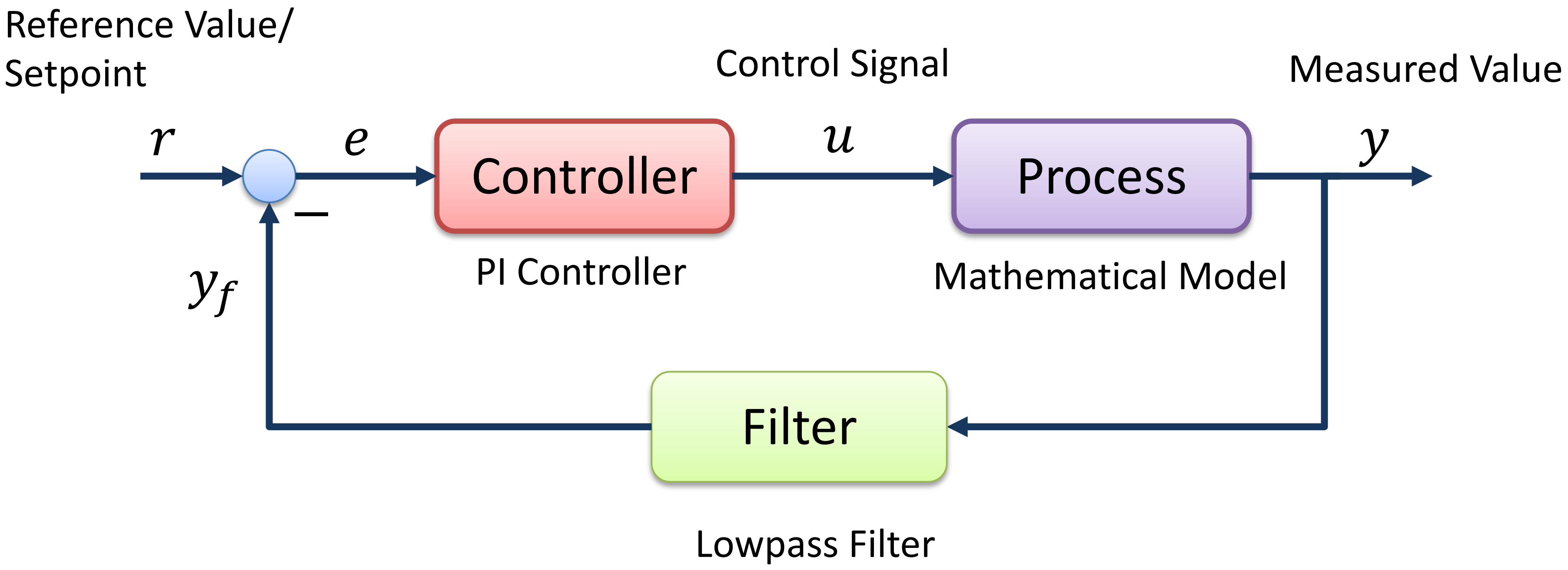
Table of Contents

Introduction

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
- We will also implement a **Lowpass Filter**
- When the code is working properly, we will create an **Arduino Library**
  - Makes it easy to reuse your Code in different Applications
  - Makes it easy to Distribute to others

Arduino Control System



Arduino Control System

We will create an Arduino Control System in 3 steps:

- Step 1: Create and make the Control System work
- Step 2: Make Class and Improve Code Structure
- Step 3: Making an Arduino Library

- Step 1 was part of another Tutorial - Arduino Control System
- YouTube: https://youtu.be/Zvc_I08hXxs
- We will have a short repetition here before we move on to Step 2 and Step 3

Arduino

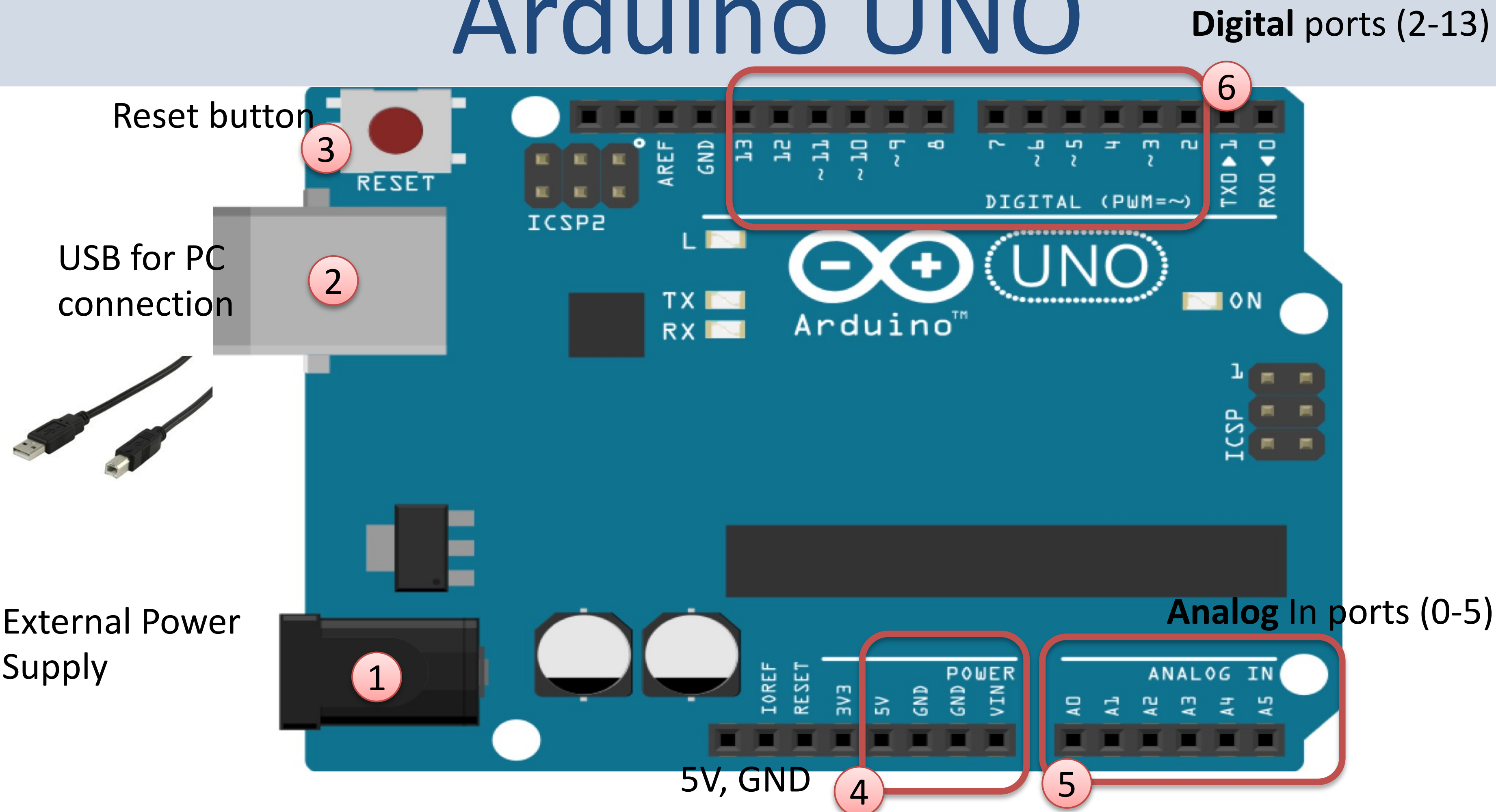
Hans-Petter Halvorsen

Table of Contents

Arduino

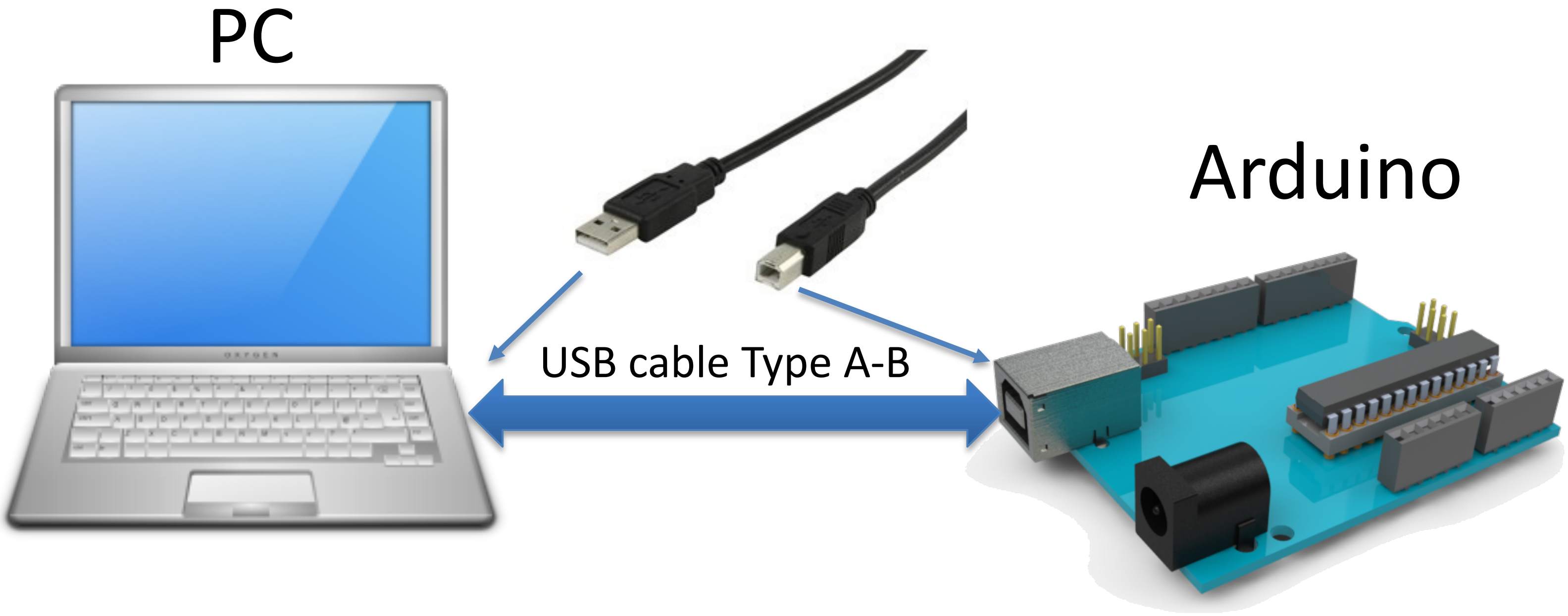
- Arduino is an open-source electronics platform based on easy-to-use hardware and software.
- It's intended for anyone making interactive projects, from kids to grown-ups.
- You can connect different Sensors, like Temperature, etc.
- It is used a lots in Internet of Things projects
- Homepage: https://www.arduino.cc

Arduino 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=~)
USB for PC connection
2
L
TX
RX
UNO
Arduino™
ON
1
ICSP
External Power Supply
1
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
Analog In ports (0-5)
5V, GND
4
5

Connect Arduino to your PC



Arduino Software

Upload Code to Arduino Board

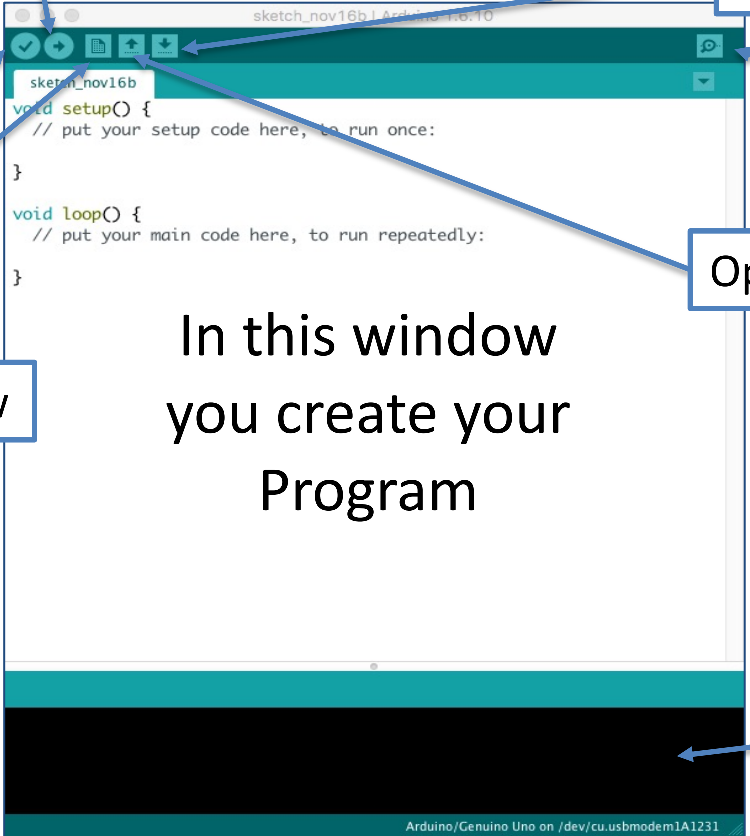
Save

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
       ...
}
```

PI Controller

Hans-Petter Halvorsen

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

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

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

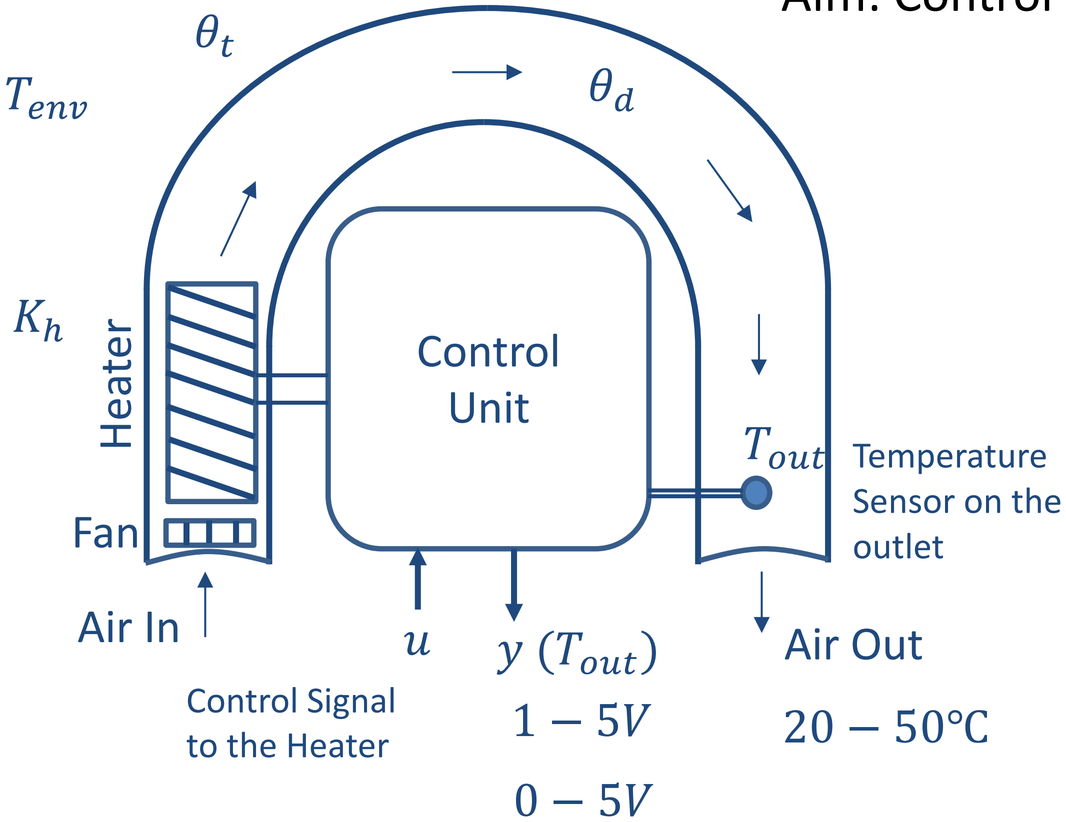
Process and Mathematical Model

Hans-Petter Halvorsen

Table of Contents

Air Heater System

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



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

$$\theta_t = 22\ s$$

$$\theta_d = 2\ s$$

$$K_h = 3.5\ \frac{°\mathrm{C}}{V}$$

$$T_{env} = 21.5\ °\mathrm{C}$$

Discrete Air Heater

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

We make a discrete version:

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

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:

$$\theta_t = 22s$$

$$K_h = 3.5\,\frac{℃}{V}$$

$$T_{env} = 21.5℃$$

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

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:

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

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

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

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

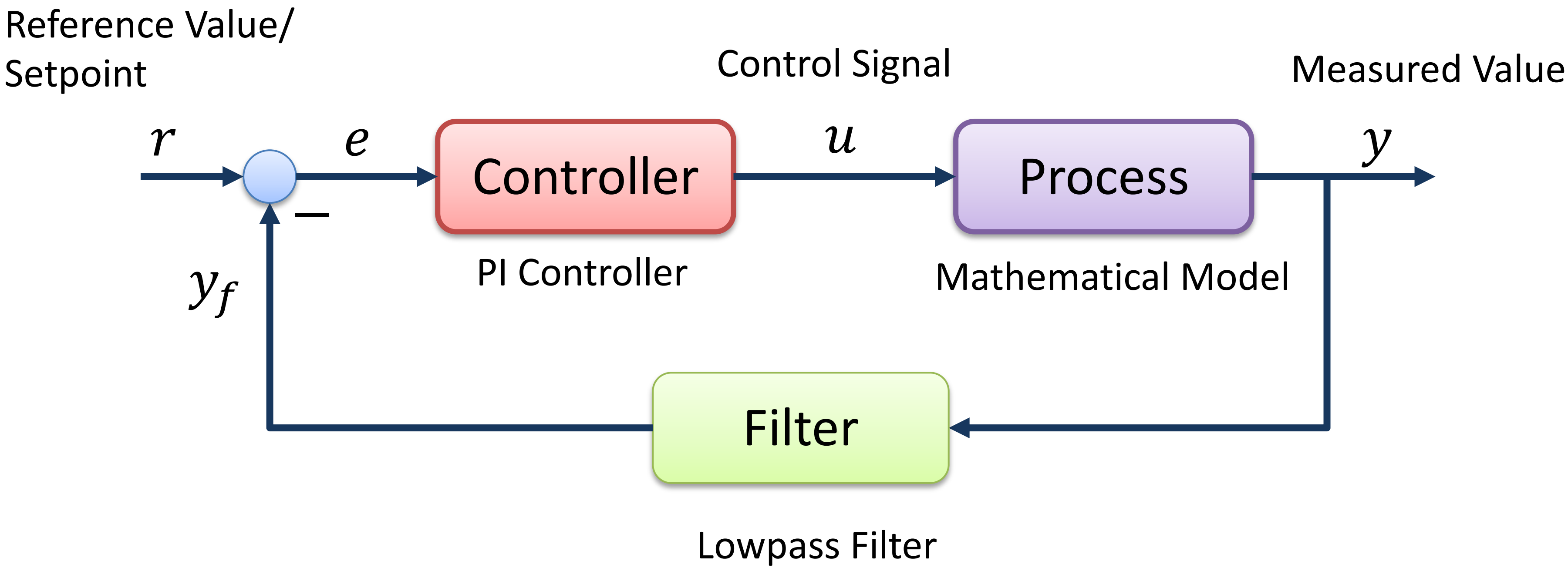
Control System Implementation

Step 1: Create and make the Control System work

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 and Lowpass Filter have been put into separate Functions

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

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

Control System

air_heater_simulation | Arduino 1.8.13

File Edit Sketch Tools Help

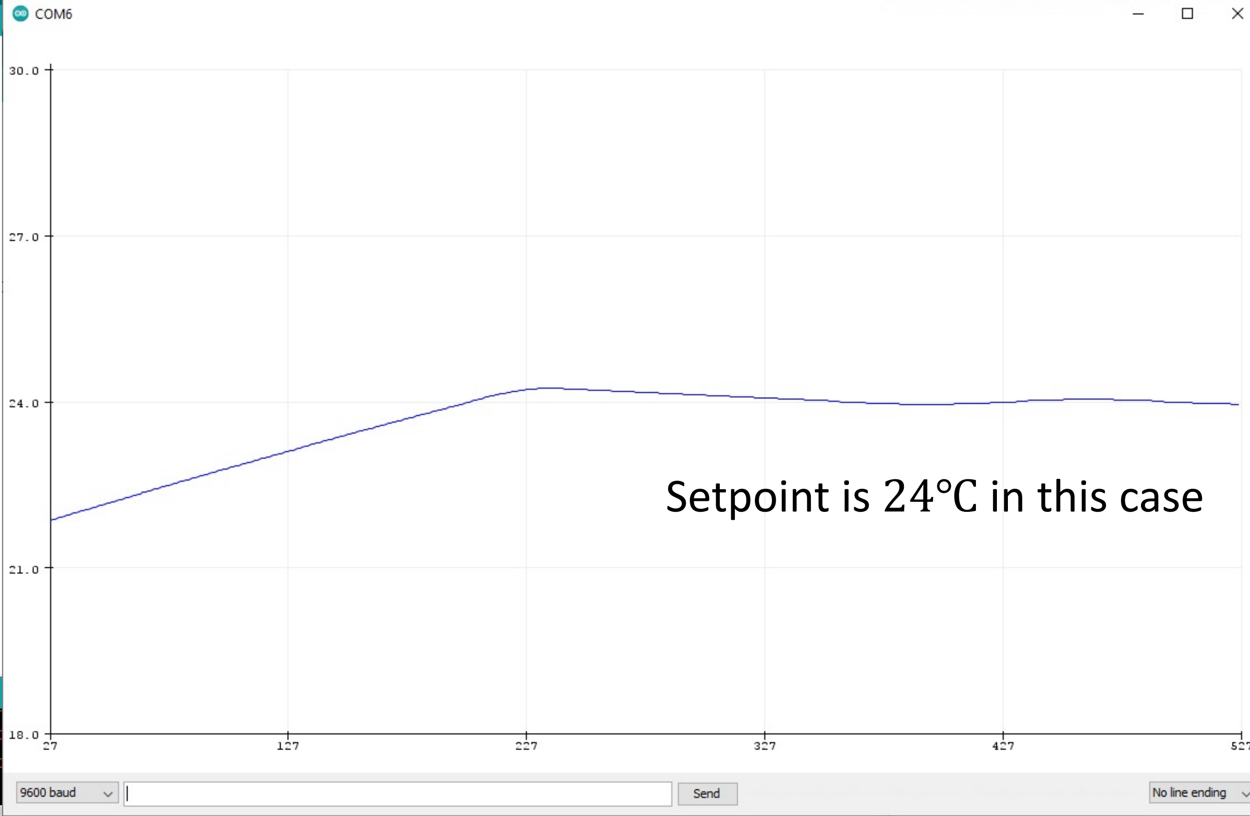
air_heater_simulation §

```
//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



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.

Control System Implementation

Step 2: Make Class and Improve Code Structure

Hans-Petter Halvorsen

Table of Contents

Code (1/2)

A Class called “Control” has been made.

The Class now contains the different functions like the “AirHeaterModel”, “PiController”, “LowassFilter”. Etc.

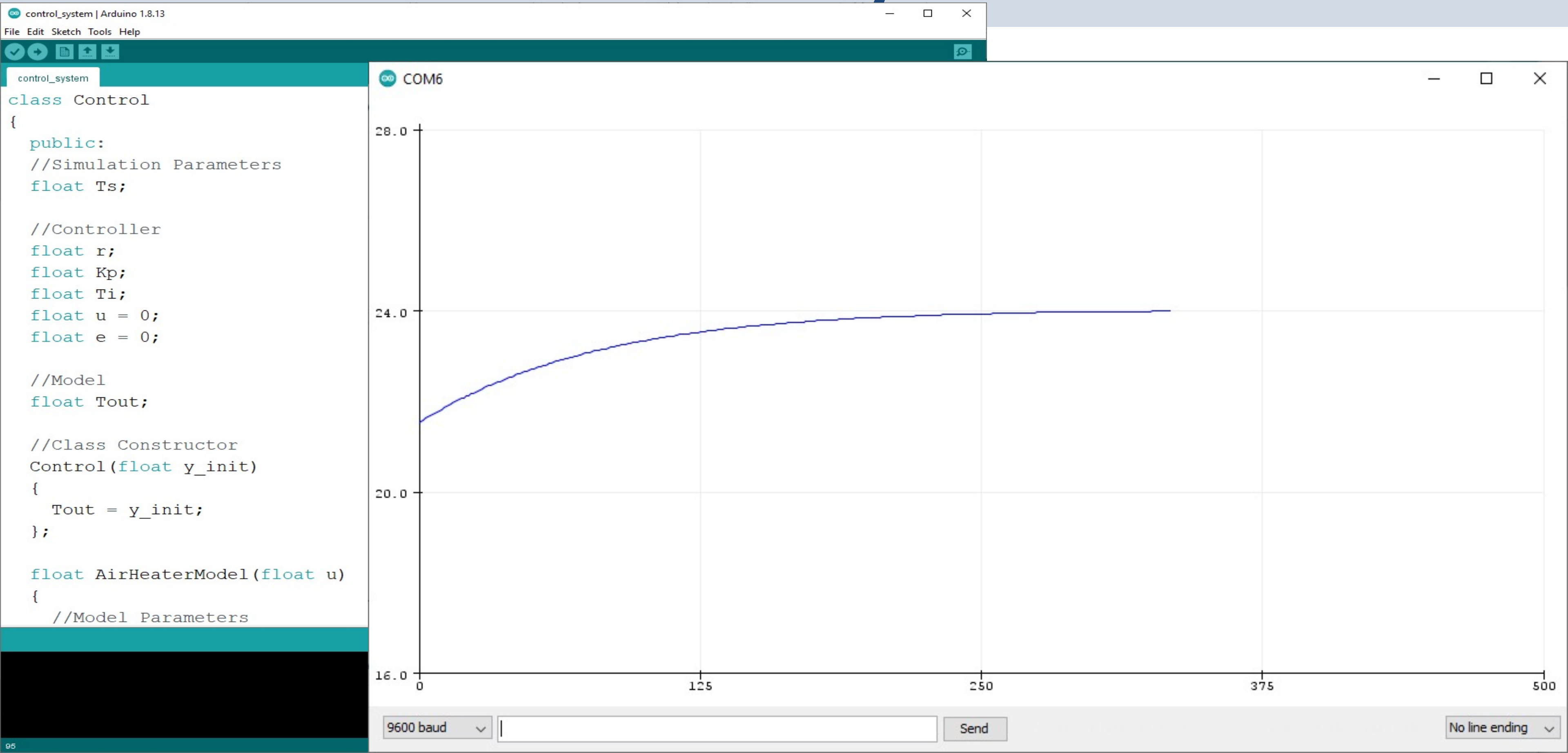
```
class Control
{
  public:

  Control(..) //Constructor
  {
    ..
  };

  float AirHeaterModel(float u)
  {
    ..
    Tout = Tout_prev + (Ts/theta_t) * (-Tout_prev + Kh*u + Tenv);
    return Tout;
  }

  float PiController(float y)
  {
    ..
    e = r - y;
    u = u_prev + Kp*(e - e_prev) + (Kp/Ti)*Ts*e;
    ..
    return u;
  }

  float LowPassFilter(float y)
  {
    ..
    yf = (1-a)*yf_prev + a*y;
    ..
    return yf;
  }

  void SerialPlotter(float y)
  {
    Serial.println(y);
  }
};
```

Code (2/2)

Main Application using the Class

```
// Simulation
float Ts = 0.1;

//Controller
float u = 0;

//Model;
float y_init = 21.5;
float y = y_init;

Control control(y_init);

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

  //Simulation
  control.Ts = Ts; //Sampling Time

  //Controller
  control.Kp = 0.8;
  control.Ti = 20;
  control.r = 24;
}

void loop()
{
  u = control.PiController(y);
  y = control.AirHeaterModel(u);
  y = control.LowPassFilter(y);
  control.SerialPlotter(y);
  delay(1000*Ts);
}
```

Control System



Control System Implementation

Step 3: Making an Arduino Library

Hans-Petter Halvorsen

Table of Contents

Arduino Library

- Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc.
- There are hundreds of additional libraries available on the Internet for download.
- You can also create your own Libraries from scratch – That's what we will show her
- Why create your own Libraries?
  - Better Code structure
  - Reuse your Code in different Applications
  - Distribute to others
- https://www.arduino.cc/en/Hacking/libraryTutorial

Arduino Library

You need at least two files for a library:

- Header file (.h) - The header file has definitions for the library
- Source file (.cpp) – The Functions within the Class

Note the Library Name, Folder name, .h and .cpp files all need to have the same name

Location:

- Windows: C:\Users\<User>\Documents\Arduino\libraries
- macOS: /Users/<User>/Documents/Arduino

Code (1/3)

Control.h

```
#ifndef Control_h
#define Control_h

#include "Arduino.h"

class Control
{
  public:
    //Simulation
    float Ts;
    //Controller
    float r;
    float Kp;
    float Ti;

    //Constructor
    Control(float y_init);

    //Functions
    float AirHeaterModel(float u);
    float PiController(float y);
    float LowPassFilter(float y);
    void SerialPlotter(float y);

  private:
    float u;
    float e;
    float Tout;
};

#endif
```

Code (2/3)

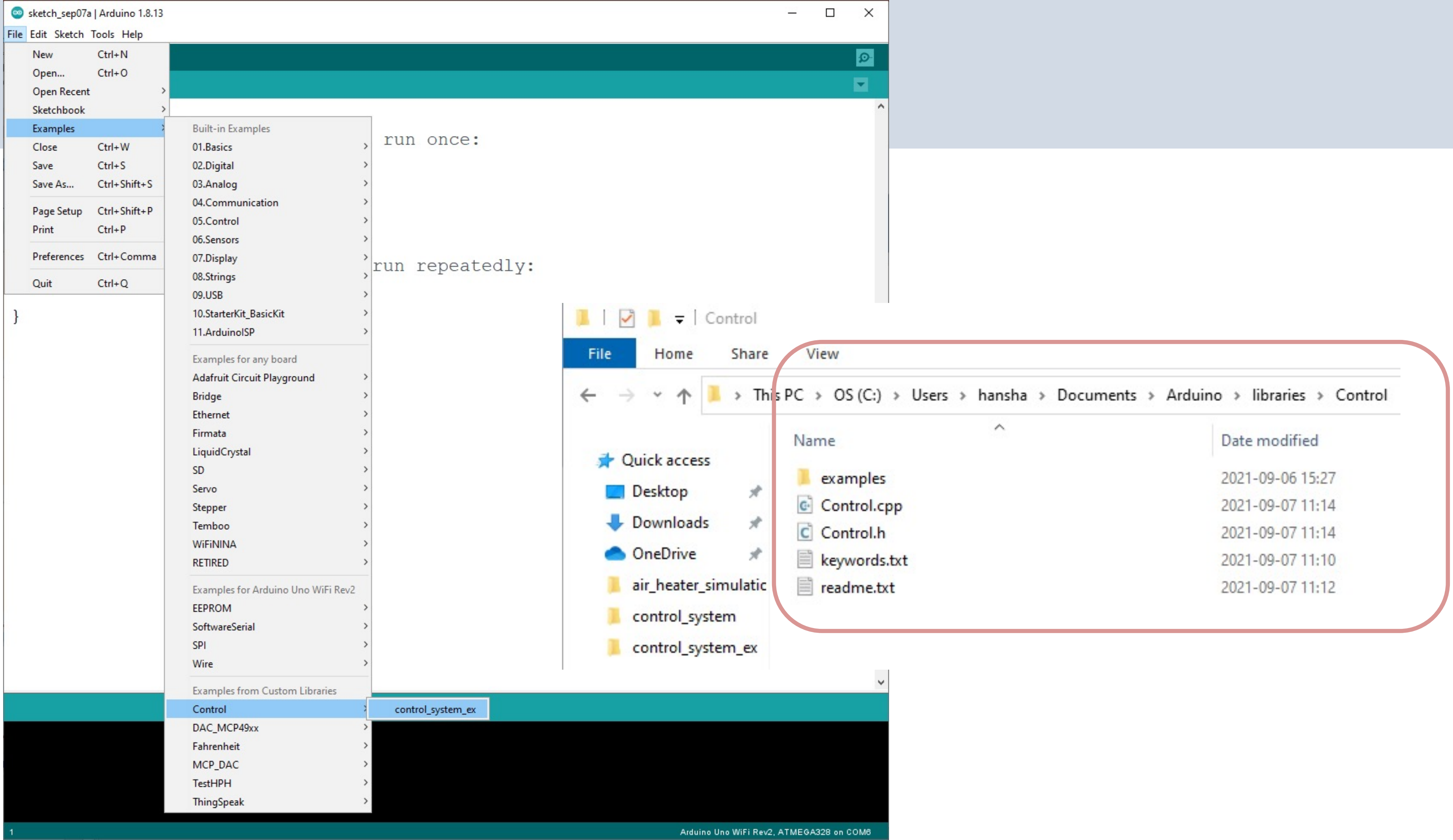
Control.cpp

```
#include "Arduino.h"
#include "Control.h"

Control::Control(float y_init)
{
  Tout = y_init;
}

float Control::AirHeaterModel(float u)
{
  //Model Parameters
  float Kh = 3.5;
  float theta_t = 22;
  float theta_d = 2;
  float Tenv = 21.5;
  float Tout_prev;
  Tout_prev = Tout;

  Tout = Tout_prev + (Ts/theta_t) * (-Tout_prev + Kh*u + Tenv);

  return Tout;
}
float Control::PiController(float y)
{
  float u_prev = u;
  float e_prev = e;

  e = r - y;
  u = u_prev + Kp*(e - e_prev) + (Kp/Ti)*Ts*e;
  if (u < 0)
    u = 0;
  if (u > 5)
    u = 5;

  return u;
}
float Control::LowPassFilter(float y)
{
  float Tf = 5*Ts;
  float a = Ts/(Tf+Ts);
  float yf;
  float yf_prev = y;

  yf = (1-a)*yf_prev + a*y;
  yf_prev = yf;
  return yf;
}
```

Code (3/3)

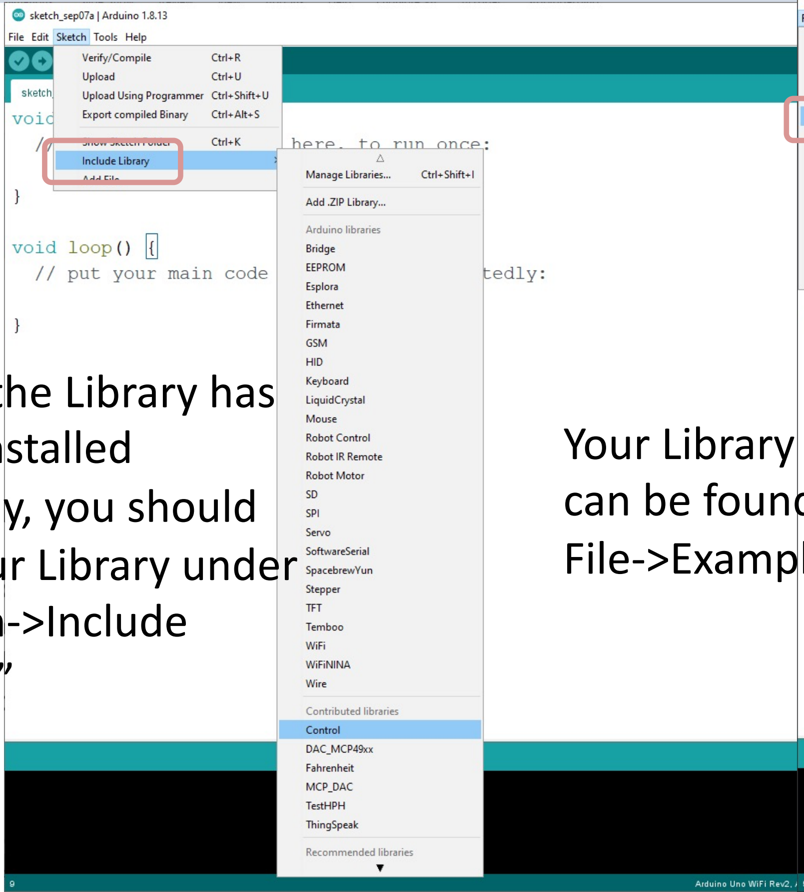
Example Code

```
#include <Control.h>

//Initialization
// Simulation
float Ts = 0.1;
//Controller
float u = 0;
//Model;
float y_init = 21.5;
float y = y_init;

//Constructor
Control control(y_init);

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

  //Simulation
  control.Ts = Ts; //Sampling Time

  //Controller
  control.Kp = 0.8;
  control.Ti = 20;
  control.r = 24; //Setpoint
}

void loop()
{
  u = control.PiController(y);
  y = control.AirHeaterModel(u);
  y = control.LowPassFilter(y);
  control.SerialPlotter(y);
  delay(1000*Ts);
}
```

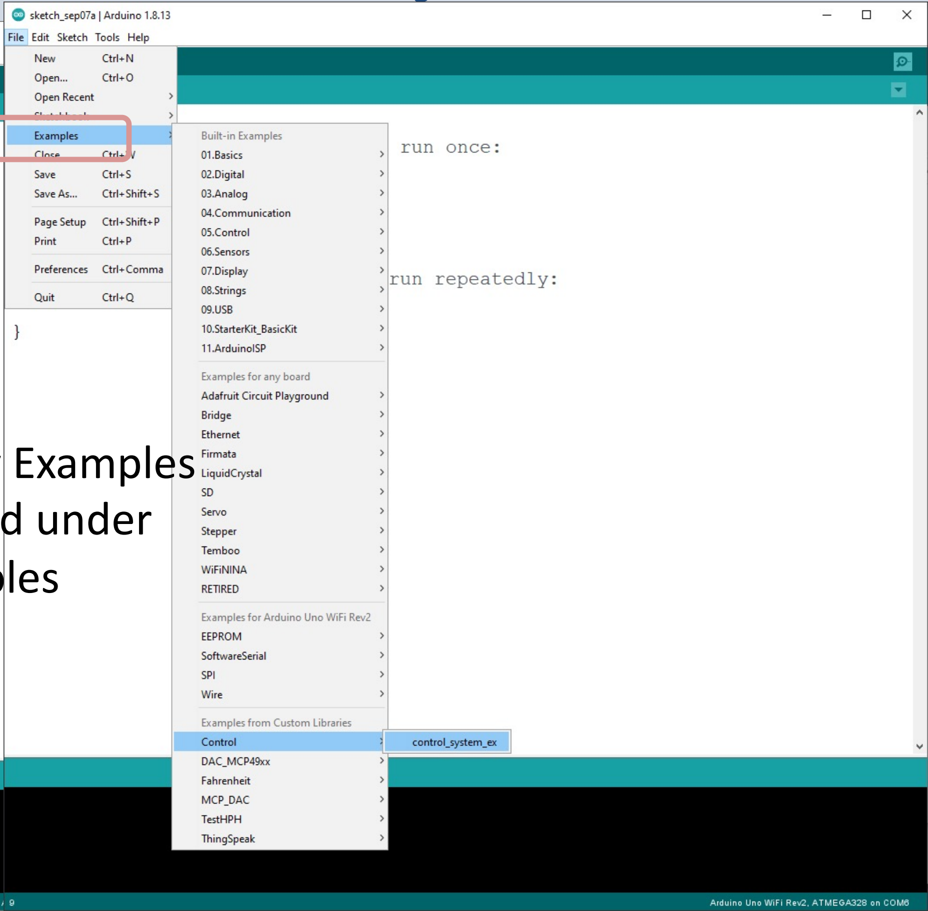

sketch_sep07a | Arduino 1.8.13
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Servo
Stepper
Temboo
WiFiNINA
RETIRED
Examples for Arduino Uno WiFi Rev2
EEPROM
SoftwareSerial
SPI
Wire
Examples from Custom Libraries
Control
control_system_ex
DAC_MCP49xx
Fahrenheit
MCP_DAC
TestHPH
ThingSpeak
run once:
run repeatedly:
}
1
Arduino Uno WiFi Rev2, ATMEGA328 on COM6
Control
File Home Share View
This PC › OS (C:) › Users › hansha › Documents › Arduino › libraries › Control
Quick access
Desktop
Downloads
OneDrive
air_heater_simulatic
control_system
control_system_ex
Name
Date modified
examples 2021-09-06 15:27
Control.cpp 2021-09-07 11:14
Control.h 2021-09-07 11:14
keywords.txt 2021-09-07 11:10
readme.txt 2021-09-07 11:12

Using the Library

When the Library has been installed properly, you should see your Library under “Sketch->Include Library”



Your Library Examples can be found under File->Examples



Control.h

```cpp
/*
  Control.h - Control Engineering Arduino Library
  Created by Hans-Petter Halvorsen
*/
#ifndef Control_h
#define Control_h

#include "Arduino.h"

class Control
{
  public:
    //Simulation
    float Ts;
    //Controller
    float r;
    float Kp;
    float Ti;

    //Constructor
    Control(float y_init);

    //Functions
    float AirHeaterModel(float u);
    float PiController(float y);
    float LowPassFilter(float y);
    void SerialPlotter(float y);

  private:
    float u;
    float e;
    float Tout;
};

#endif
```

Control.cpp

```cpp
/*
Control.h - Control Engineering Arduino Library
Created by Hans-Petter Halvorsen
*/

#include "Arduino.h"
#include "Control.h"

Control::Control(float y_init)
{
  Tout = y_init;
}

float Control::AirHeaterModel(float u)
{
  //Model Parameters
  float Kh = 3.5;
  float theta_t = 22;
  float theta_d = 2;
  float Tenv = 21.5;
  float Tout_prev;
  Tout_prev = Tout;

  Tout = Tout_prev + (Ts/theta_t) * (-Tout_prev + Kh*u + Tenv);

  return Tout;
}

float Control::PiController(float y)
{
  float u_prev = u;
  float e_prev = e;

  e = r - y;
  u = u_prev + Kp*(e - e_prev) + (Kp/Ti)*Ts*e;
  if (u < 0)
    u = 0;
  if (u > 5)
    u = 5;

  return u;
}
```

```cpp
#include <Control.h>

//Initialization

// Simulation
float Ts = 0.1;

//Controller
float u = 0;

//Model;
float y_init = 21.5;
float y = y_init;

//Constructor
Control control(y_init);

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

  //Simulation
  control.Ts = Ts; //Sampling Time

  //Controller
  control.Kp = 0.8;
  control.Ti = 20;
  control.r = 24; //Setpoint
}

void loop()
{
  u = control.PiController(y);
  y = control.AirHeaterModel(u);
  y = control.LowPassFilter(y);
  control.SerialPlotter(y);
  delay(1000*Ts);
}
```

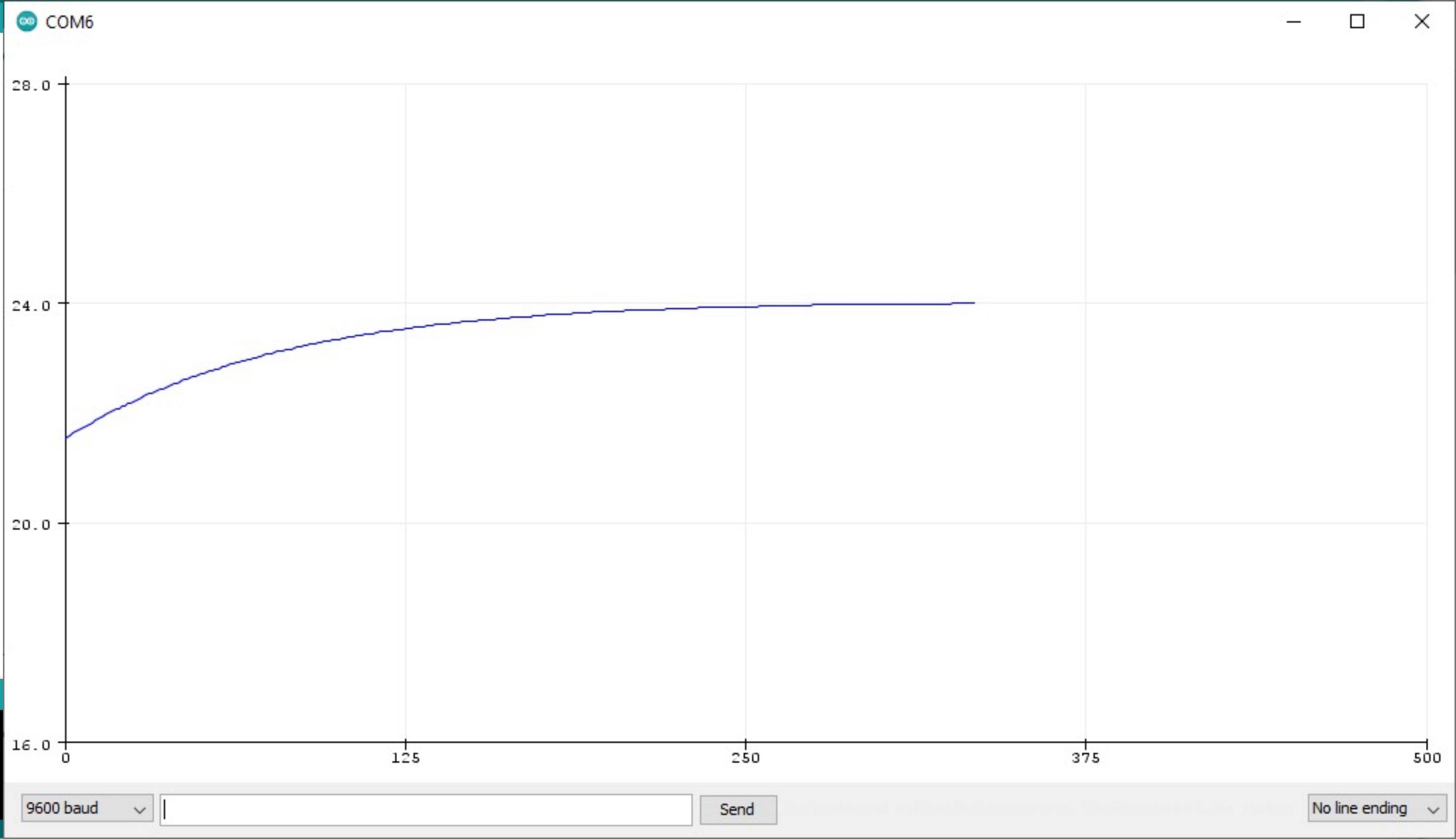
Control System

control_system_ex | Arduino 1.8.13

File Edit Sketch Tools Help

control_system_ex

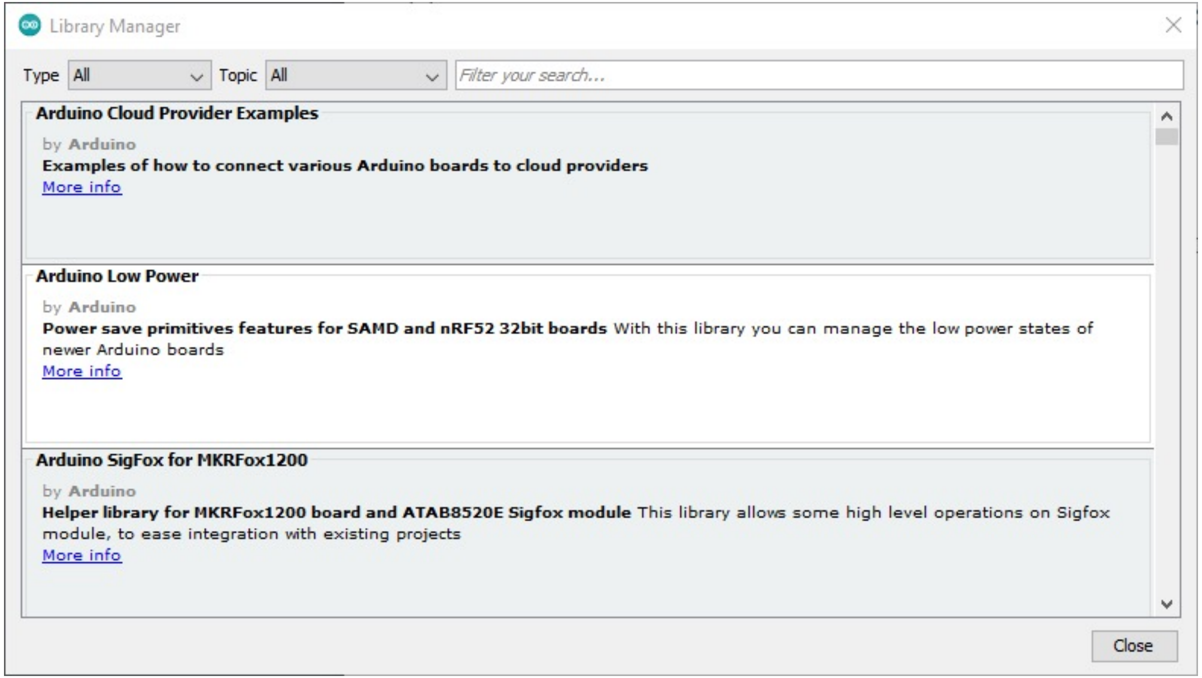
```
#include <Control.h>

//Initialization

// Simulation
float Ts = 0.1;

//Controller
float u = 0;

//Model;
float y_init = 21.5;
float y = y_init;

//Constructor
Control control(y_init);

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

  //Simulation
  control.Ts = Ts; //Sampling Time
```



Library Manager

Hans-Petter Halvorsen

Library Manager



If you are really happy with your Arduino Library and thing others may have use for it, you may distribute it to others via the Library Manager.

For more information how you can do that: https://github.com/arduino/library-registry

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog