https://www.halvorsen.blog

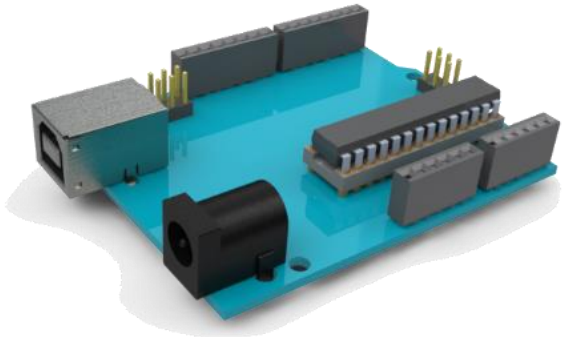
Internet of Things (IoT) Control System

Hans-Petter Halvorsen

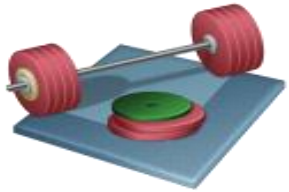
Table of Contents

Introduction

- Cloud services and IoT solutions are becoming increasingly popular.
- Even the industry embrace IoT as Industrial Internet of Things (IIoT), which is part of the next generation Automation Systems
- In this Assignment you will use Internet of Things (IoT) Technology, Devices and Services to create an Embedded Arduino PID Controller
- One of the challenges is that Arduino UNO has no Analog Out
- The Data should be stored in the Cloud



Lab Assignment Overview



1. Create Embedded PI(D) Controller + Lowpass Filter in Arduino code
   - Create “Arduino Library” with functions
   - Create “Analog Out”, using DAC chip and use a simple RC circuit on a breadboard (Arduino UNO 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/Cloud

See next slides for details...

Controlling the Air Heater using Arduino

Embedded PID Controller

PV

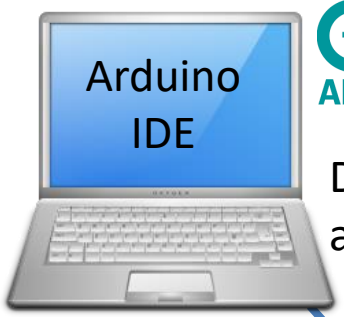
MV

Air Heater (Process)

PC as Controller

Industrial PID Controller

System Overview



ARDUINO

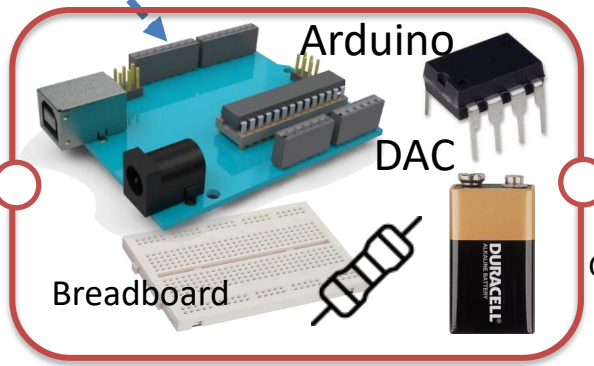
Download your Application
and then remove USB cable

Cloud

ThingSpeak

K_p, T_i, T_d r

Air Heater



Analog In

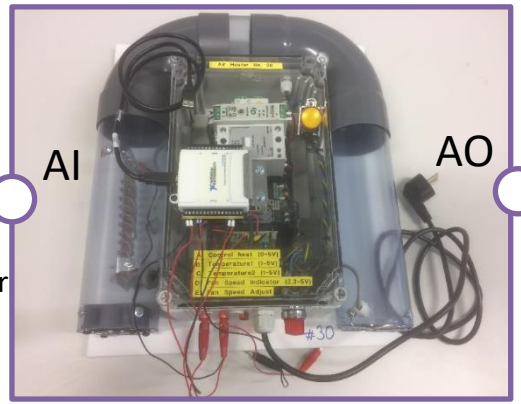
y

Banana
Connector

Analog Out

u

Banana Connector Banana Connector



Banana
Connector

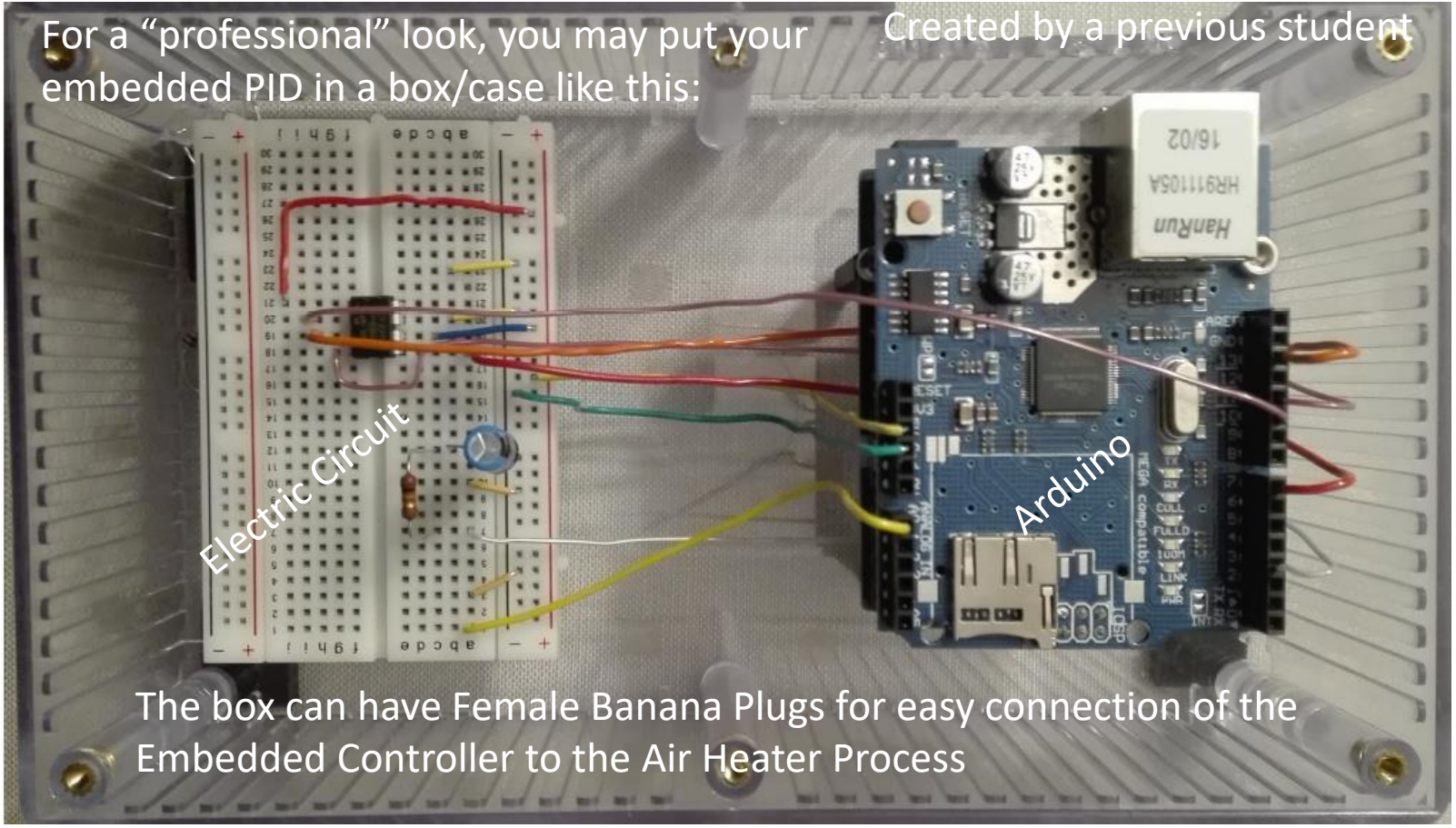
y

Embedded Arduino PID Controller

Test first using HIL
Simulation and Testing

Feedback System

Embedded Arduino PID Example



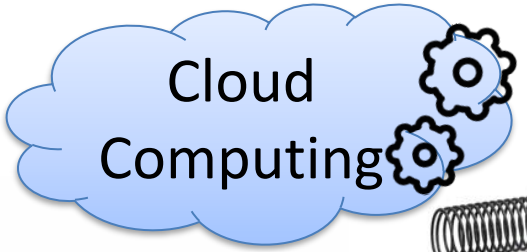
Keywords

- Internet of Things (IoT) and Cyber Security
- Arduino, Electronics, Embedded Systems
- Control Design and Simulations
- Practical Implementation of Control Systems
- PID
- Hardware in the Loop (HIL) Simulation and Testing
- Cloud Services, Publishing and Monitoring
- Data Analysis

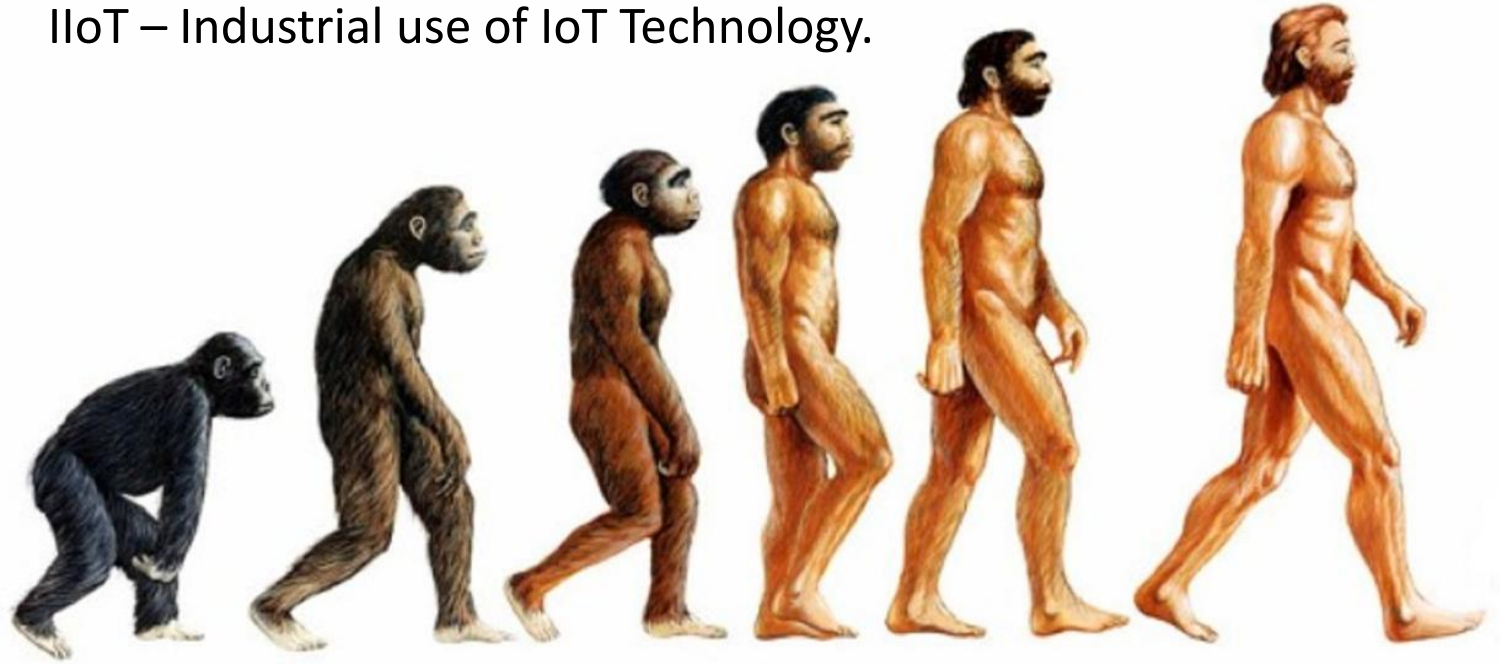
Learning Goals

- Introduction to the term Internet of Things (IoT) and how it affects the next generation Control and Automation Systems
- Learn PID Control, both Control theory and Practical implementations
- Learn Programming, Arduino Programming and LabVIEW
- Learn about Microcontrollers (Arduino)
- Learn about Electronics and Electrical Components
- Learn about Hardware-Software Interactions
- Learn about Digital to Analog Conversion (DAC)
- Learn SPI/I2C Communication
- Learn Hardware-in-the Loop (HIL) Simulation and Testing
- Learn Software Installation, which can be cumbersome with many pitfalls
- Learn to use and create Software
- Internet of Things and Cyber Security

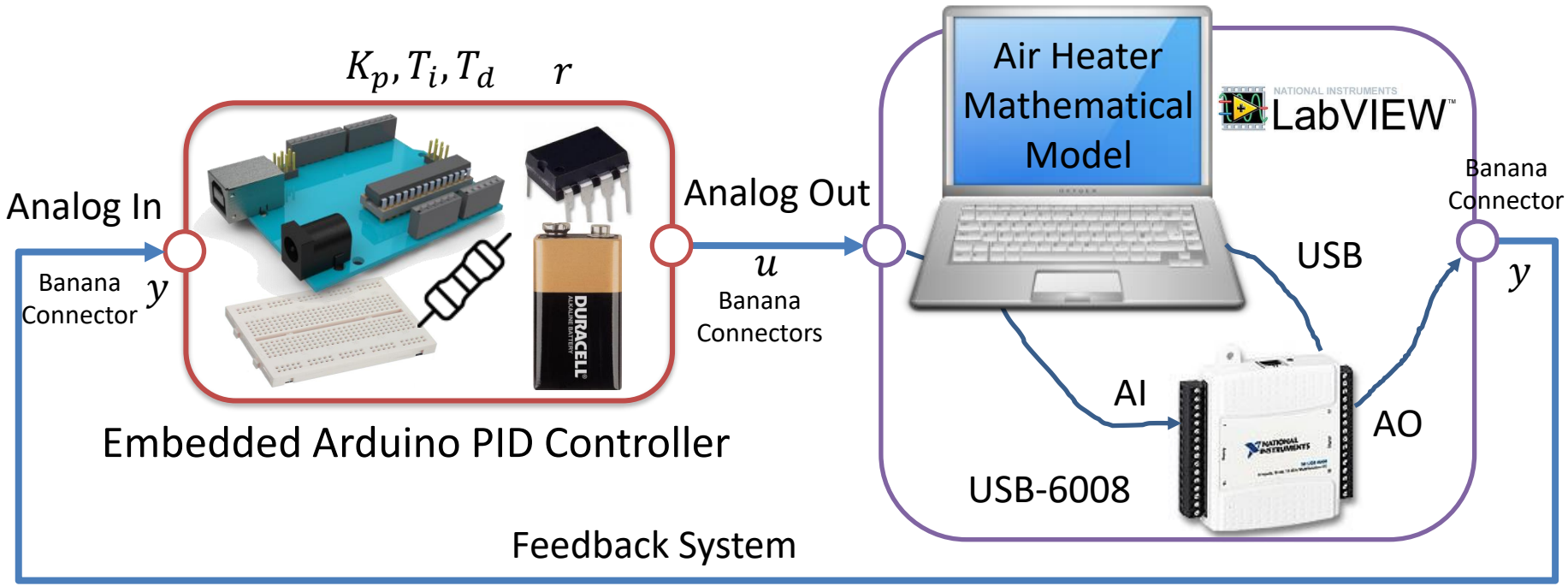
Internet of Things (IoT)



Industrial Internet of Things (IIoT) is another word for Industry 4.0
IoT – Consumer oriented, Smart Home Solutions, etc.
IIoT – Industrial use of IoT Technology.



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.

Software

Arduino IDE

Lowpass Filter

PID Controller

NATIONAL INSTRUMENTS LabVIEW™

Air Heater Simulator

HIL Simulation and Testing

MATLAB

ThingSpeak

Cloud Service

ThingSpeak is a IoT Service that lets you collect and store sensor data in the cloud and develop Internet of Things applications.

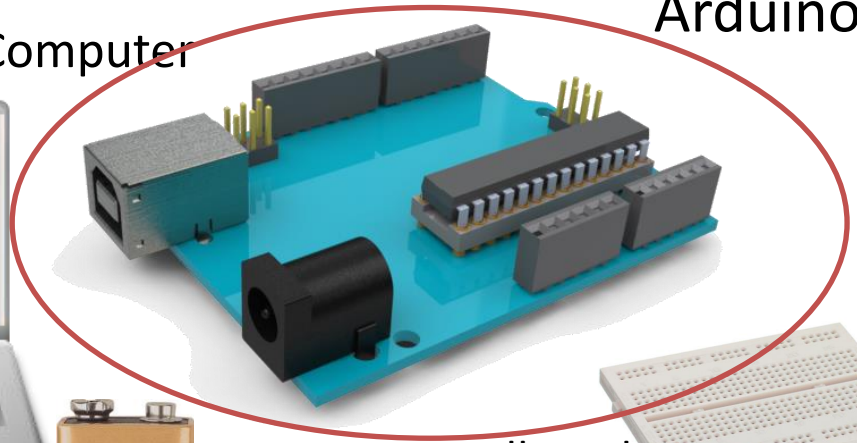
fritzing

Fritzing is an open source software for design of electronics hardware and creating circuit diagrams

Hardware

Your Personal Computer

Arduino



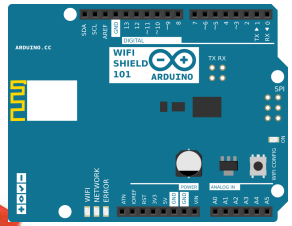
DAQ Device, e.g. USB-6008, USB-6001, myDAQ

Breadboard

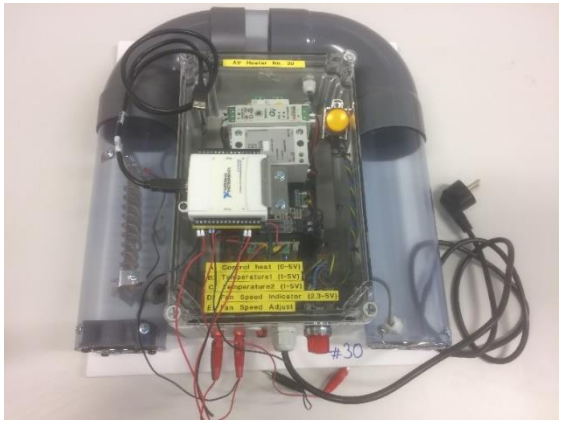
9V Battery

Multi-meter

Banana Plugs/Cables



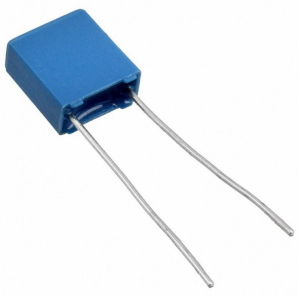
(Arduino Wi-Fi Shield)



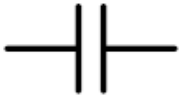
Air Heater

Electrical Components

Capacitor



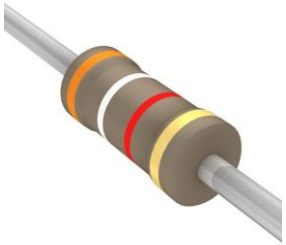
e.g., $C = 10\mu F$



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

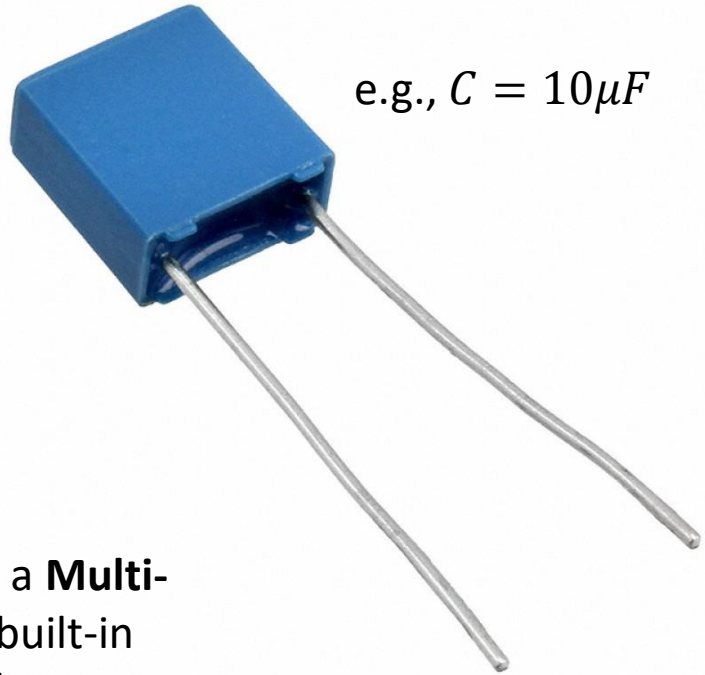


$R = 3.9k\Omega$

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

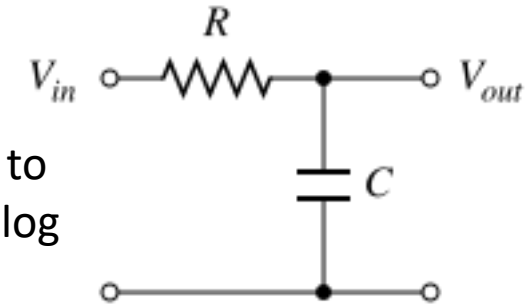


e.g., $C = 10\mu F$

The Capacitor is typically 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).

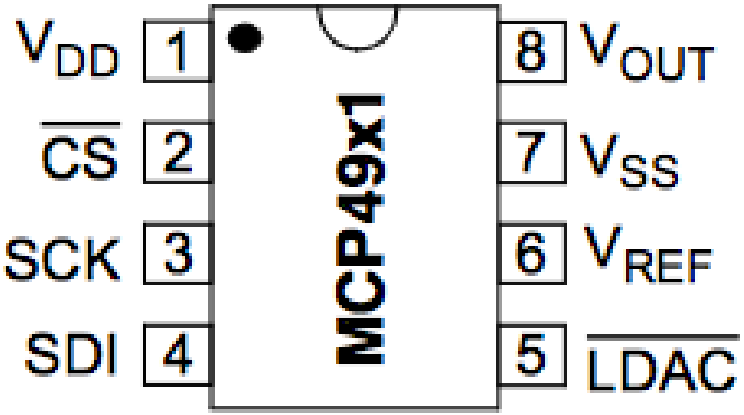


We will use the capacitor to create a RC Lowpass Filter in order to smooth the PWM signal from the Arduino to make a “real” Analog Out Signal

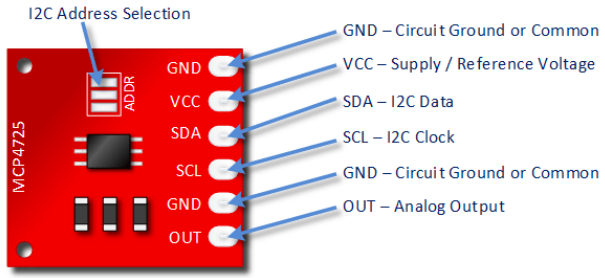
DAC

Arduino UNO has no Analog Output Pins, so we need a DAC such as, e.g., Microchip **MCP4911**, MCP4725 or similar

MCP4911: 10-bit single DAC, SPI Interface



MCP4725



12-bit resolution
I2C Interface

The MCP4725 is a little more expensive, but simpler to use

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!

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

Visual Studio

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

Google

Another person in the world probably had a similar problem

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

My System is not Working??

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.

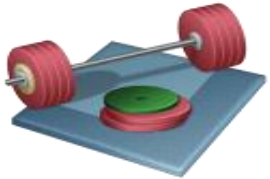
Introduction to Arduino

https://www.halvorsen.blog/documents/technology/iot/arduino

Hans-Petter Halvorsen

Arduino Microcontroller

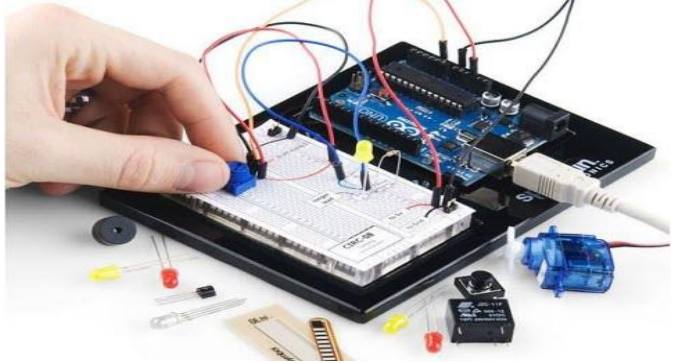
Get Started with Arduino



It is assumed that you are already familiar with basic Arduino (if not, basic Arduino Programming is very simple to learn!). If you feel you need to refresh your knowledge, then it is recommended that you do the small exercises below (if not, you may skip them).

In order to learn basic Arduino, do the following basic Arduino Exercises:

- Make a **LED** blink
- Turn on a LED using a **Switch/Push Button**
- Use a **Potentiometer** in order to decrease/increase the intensity of a LED (“light dimmer”)
- Read Temperature values using **TMP36**
- Read Temperature values using **NTC Thermistor**



- 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

http://www.arduino.cc

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

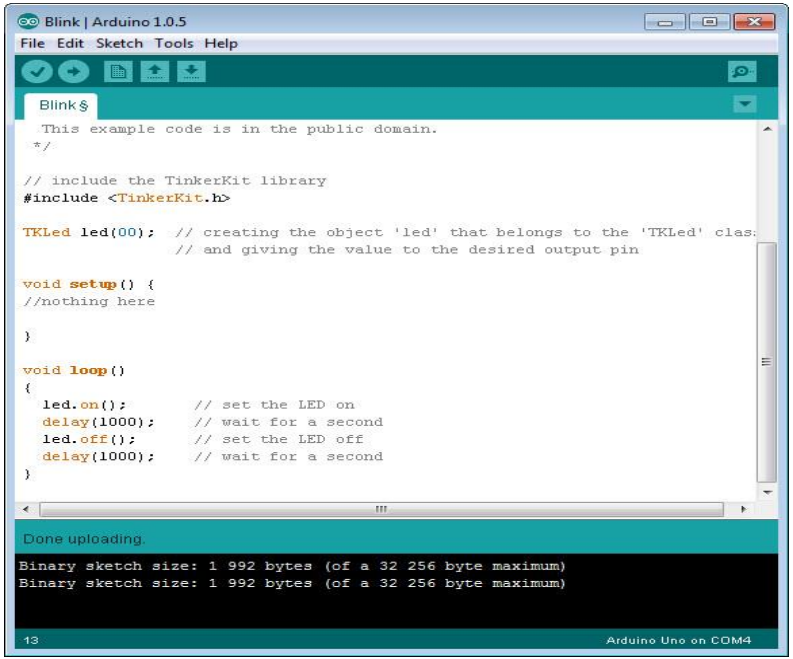
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 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

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

Arduino Software

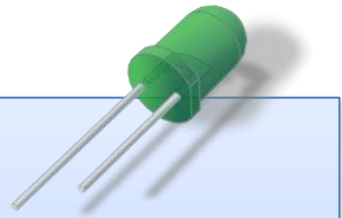
Programming with Arduino is simple and intuitive!

Arduino Sketch IDE



The syntax is similiar to C programming

Example:



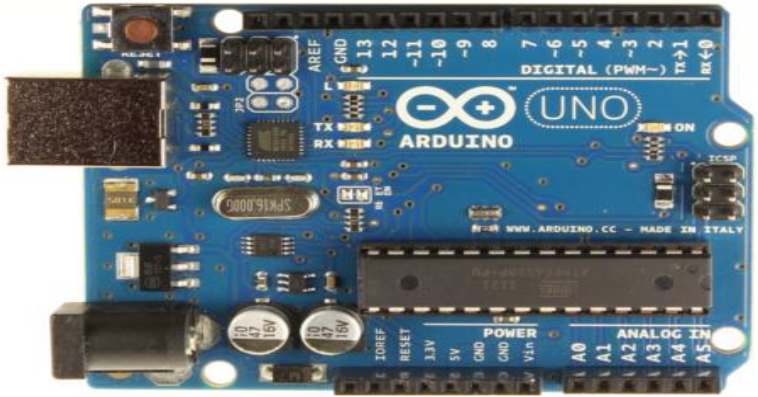
```
void setup()
{
      pinMode(8, OUTPUT);
}

void loop()
{
      digitalWrite(8, HIGH);    // Turn on the LED
      delay(1000);              // Wait for one second
      digitalWrite(8, LOW);     // Turn off the LED
      delay(1000);              // Wait for one second
}
```

This program makes a LED blink

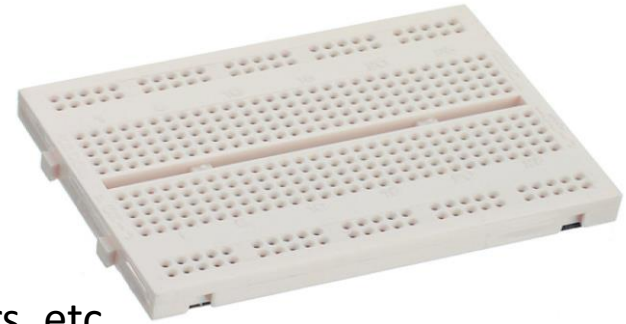
Software Installation: http://arduino.cc/en/Main/Software

Arduino Basics

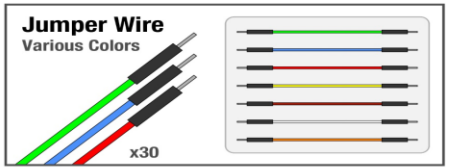
Arduino Uno Board

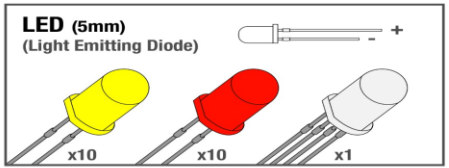


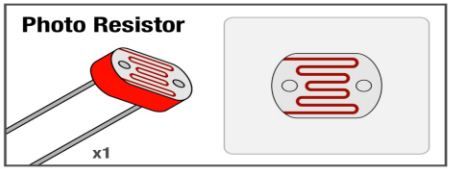
Breadboard

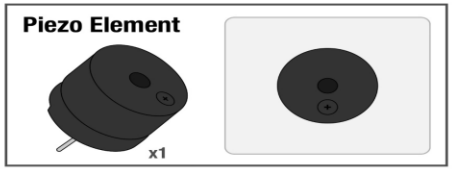


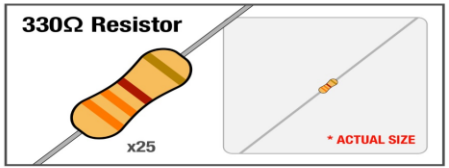
Sensors and Actuators, etc.

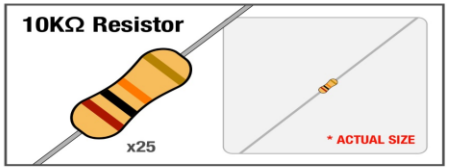


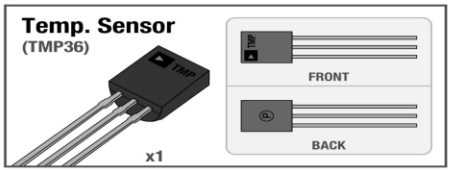


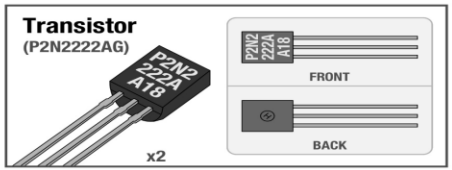


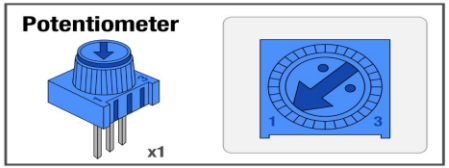


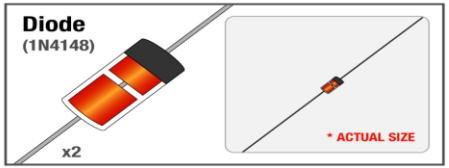


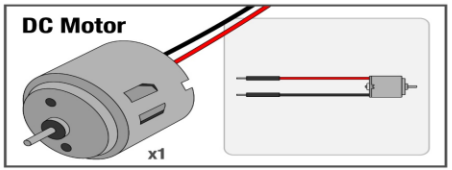


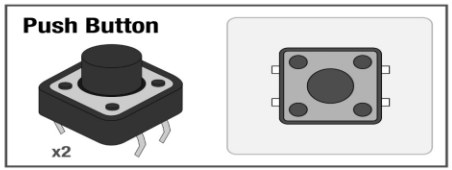




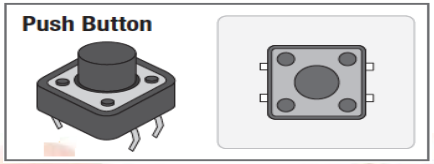


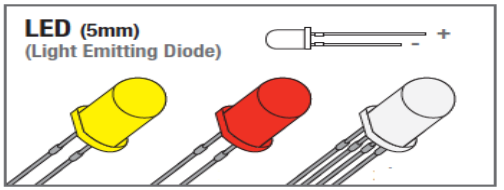


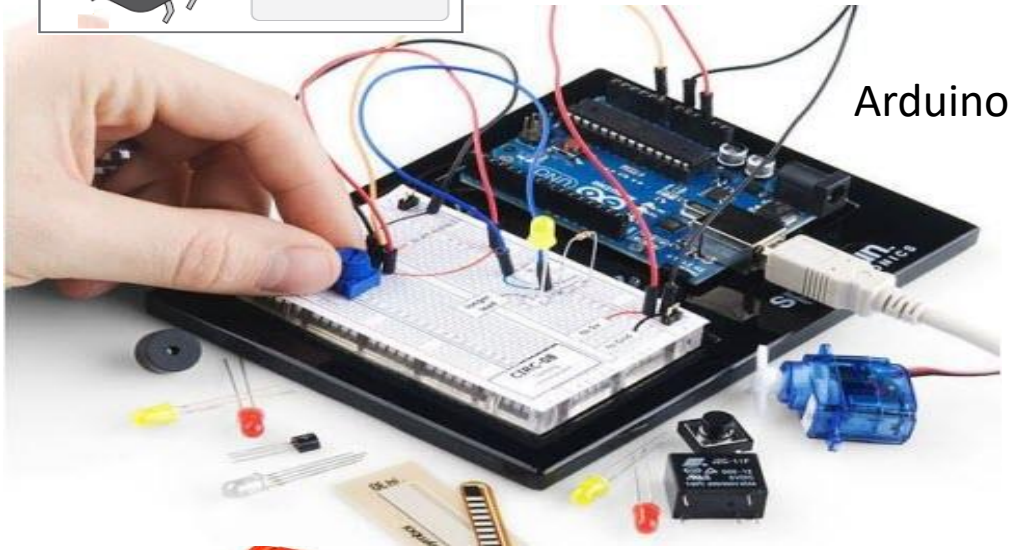




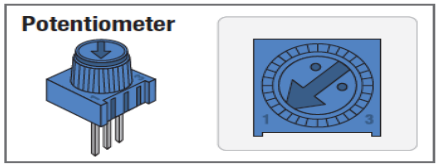
The Arduino Kit



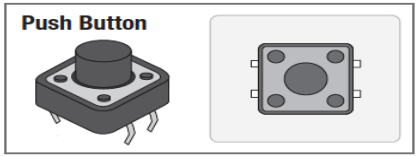


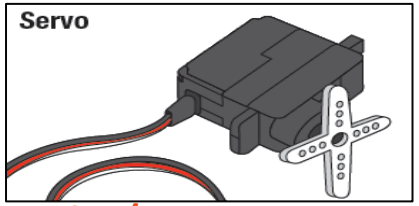


Arduino Uno Board



Small-size Sensors and Actuators





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

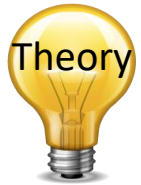
The Arduino Kit

- Arduino 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

Sensors and Actuators

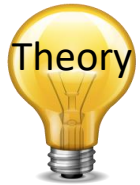


- A **Sensor** is a converter that measures a physical quantity and converts it into a signal which can be read by an observer or by an (today mostly electronic) instrument.
- An **Actuator** is a type of motor for moving or controlling a mechanism or system. It is operated by a source of energy, typically electric current, hydraulic fluid pressure, or pneumatic pressure, and converts that energy into motion. An actuator is the mechanism by which a control system acts upon an environment.

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

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

Sensors

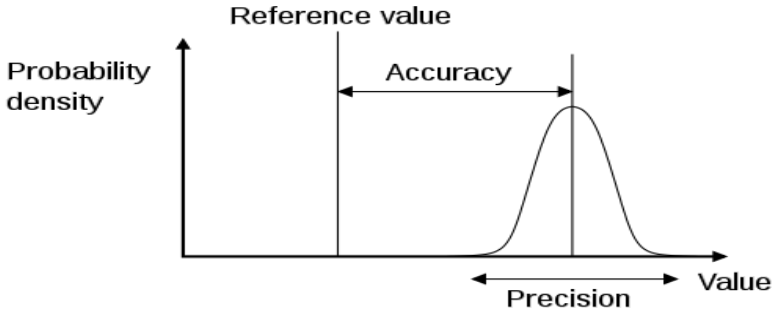


Calibration: A comparison between measurements. One of known magnitude or correctness made or set with one device and another measurement made in as similar a way as possible with a second device. The device with the known or assigned correctness is called the standard. The second device is the unit under test, test instrument, or any of several other names for the device being calibrated.

Accuracy: How close the measured value is the the actual/real value, e.g., ±0.1 %

Resolution: The smallest change it can detect in the quantity that it is measuring. The following formula may be used (where S is the measurement span, e.g., 0-100deg.C):

$$R = \frac{S}{2^n - 1}$$



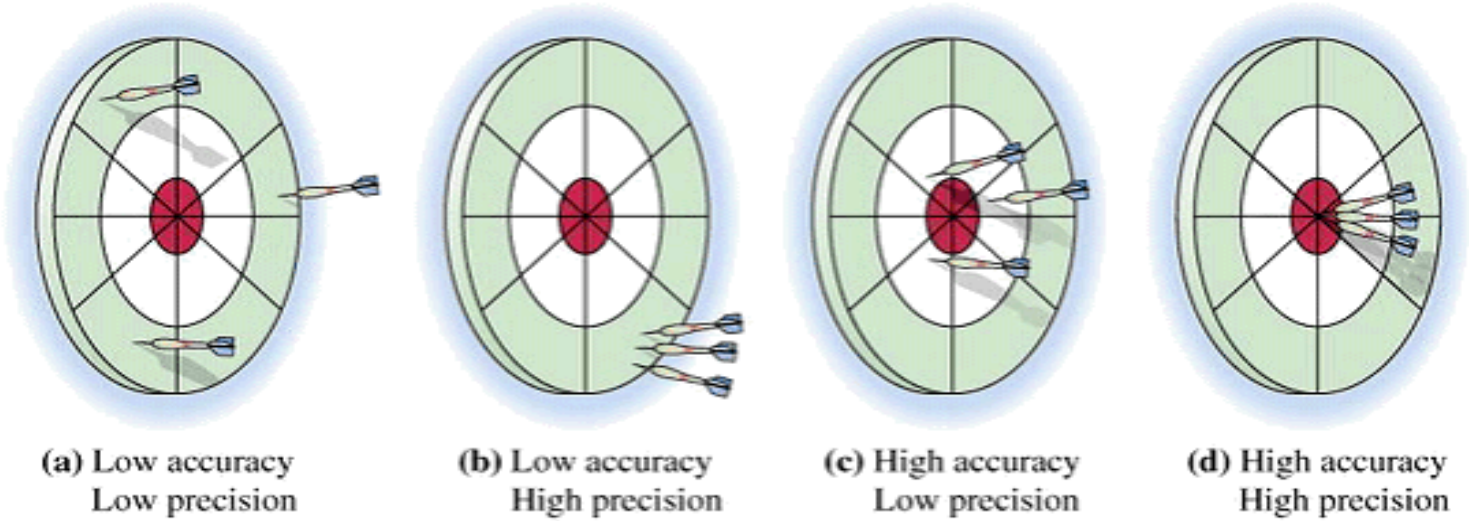
In the assignment you need to deal with these parameters. You find information about these parameters in the Data sheet for your device

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

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

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

Measurements and Sensors



(a) Low accuracy
Low precision

(b) Low accuracy
High precision

(c) High accuracy
Low precision

(d) High accuracy
High precision

Measurement Fundamentals: http://www.ni.com/white-paper/4523/en/

Sensor Fundamentals: http://www.ni.com/white-paper/4045/en/

Sensor Terminology: http://www.ni.com/white-paper/14860/en/

Voltage-based Sensors

TMP36

According to the TMP36 datasheet, the relation of the output voltage to the actual temperature uses this equation:

$$y[°C] = (1/10)*x[mv]-50$$

Where the voltage value is specified in millivolts.

However, before you use that equation, you must convert the integer value that the analogRead function returns into a millivolt value.

10-bit analog to digital converter

You know that for a 5000mV (5V) value span the analogRead function will return 1024 possible values:

$$voltage = (5000 / 1024) * output$$

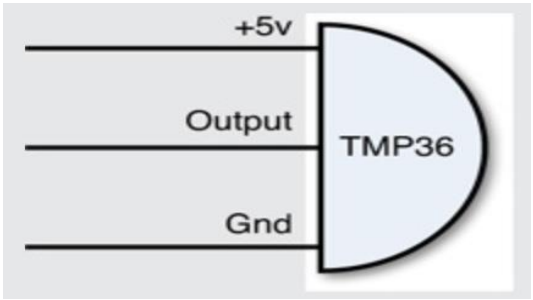
Where

mV

$$output = analogRead(aichannel)$$

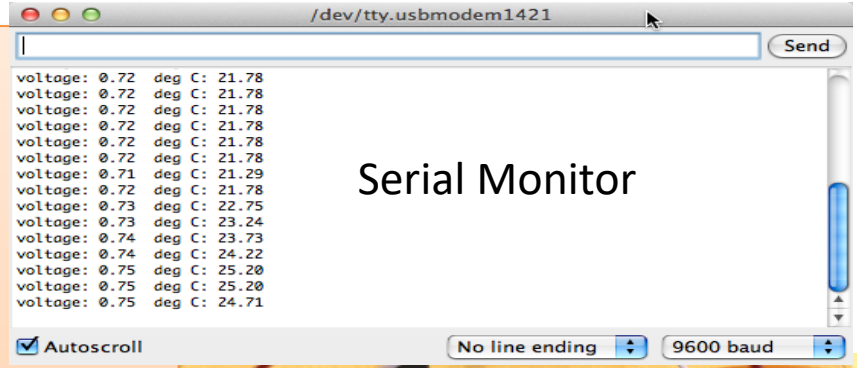
0-1023

A0-A5

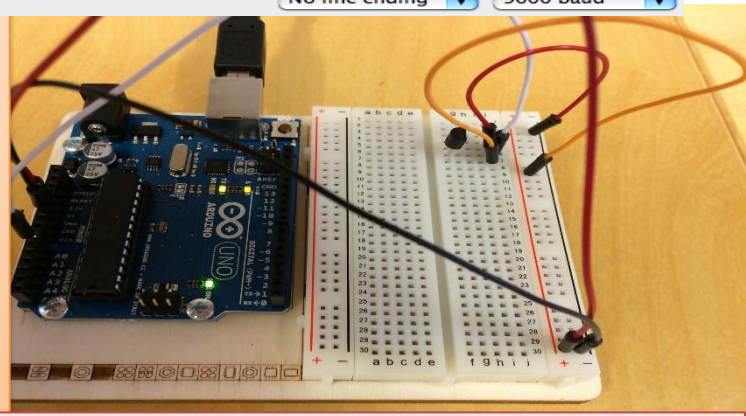


TMP36 Temperature Sensor Example

```
// We'll use analog input 0 to read Temperature Data

const int temperaturePin = 0;

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

void loop()
{
  float voltage, degreesC, degreesF;

  voltage = getVoltage(temperaturePin);

  // Now we'll convert the voltage to degrees Celsius.
  // This formula comes from the temperature sensor datasheet:

  degreesC = (voltage - 0.5) * 100.0;

  // Send data from the Arduino to the serial monitor window
  Serial.print("voltage: ");
  Serial.print(voltage);
  Serial.print("  deg C: ");
  Serial.println(degreesC);

  delay(1000); // repeat once per second (change as you wish!)
}


float getVoltage(int pin)
{
  return (analogRead(pin) * 0.004882814);

  // This equation converts the 0 to 1023 value that analogRead()
  // returns, into a 0.0 to 5.0 value that is the true voltage
  // being read at that pin.
}
```

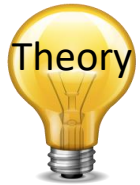


Serial Monitor



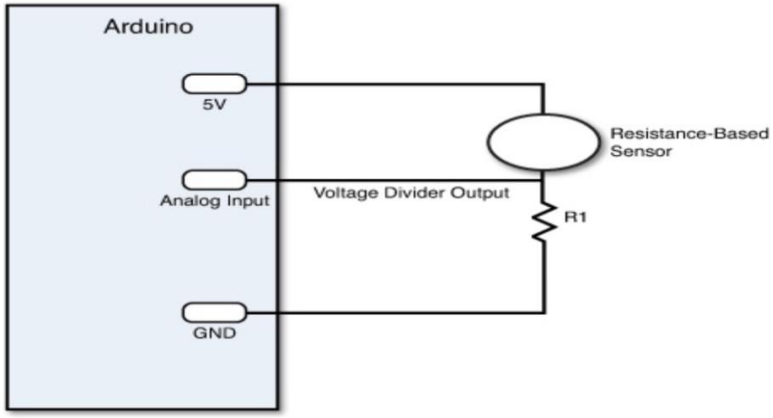
Just don't copy the Example! Make it from scratch in your own way! You need to understand what's happens! Play and Explore! Add Value to your code!

Resistance-based Sensors



The problem with resistance sensors is that the Arduino analog interfaces can't directly detect resistance changes.
This will require some extra electronic components. The easiest way to detect a change in resistance is to convert that change to a voltage change. You do that using a **voltage divider**, as shown below.

Thermistor



By keeping the power source output constant, as the resistance of the sensor changes, the voltage divider circuit changes, and the output voltage changes. The size of resistor you need for the R1 resistor depends on the resistance range generated by the sensor and how sensitive you want the output voltage to change.

E.g., the Steinhart-Hart Equation can be used to find the Temperature:

$$\frac{1}{T} = A + B\ln(R) + C(\ln(R))^3$$

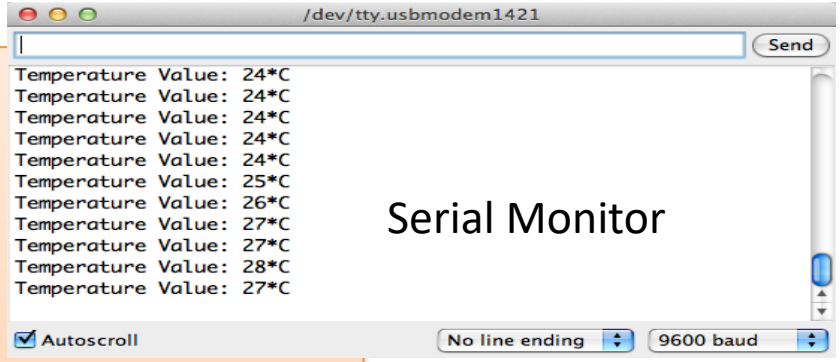
Generally, a value between 1K and 10K ohms works just fine to create a meaningful output voltage that you can detect in your Arduino analog input interface.

NTC Thermistor Example

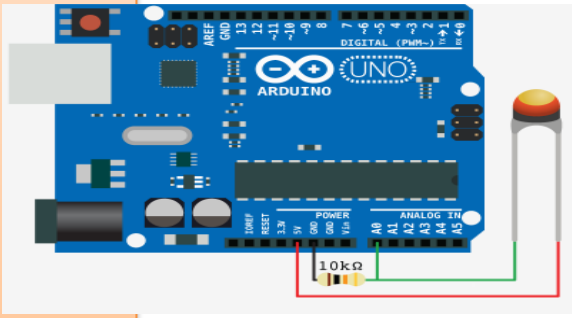
```
// Read Temerature Values from NTC Thermistor
const int temperaturePin = 0;

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

void loop()
{
  int temperature = getTemp();
  Serial.print("Temperature Value: ");
  Serial.print(temperature);
  Serial.println("*C");
  delay(1000);
}

double getTemp()
{
  // Inputs ADC Value from Thermistor and outputs Temperature in Celsius

  int RawADC = analogRead(temperaturePin);
  long Resistance;
  double Temp;

  // Assuming a 10k Thermistor.  Calculation is actually: Resistance = (1024/ADC)
  Resistance=((10240000/RawADC) - 10000);

  // Utilizes the Steinhart-Hart Thermistor Equation:

  // Temperature in Kelvin = 1 / {A + B[ln(R)] + C[ln(R)]^3}
  // where A = 0.001129148, B = 0.000234125 and C = 8.76741E-08

  Temp = log(Resistance);
  Temp = 1 / (0.001129148 + (0.000234125 * Temp) + (0.0000000876741 * Temp * Temp * Temp));
  Temp = Temp - 273.15;  // Convert Kelvin to Celsius
  return Temp;  // Return the Temperature
}
```

Steinhart-Hart Equation:

$$\frac{1}{T} = A + B\ln(R) + C(\ln(R))^3$$



Serial Monitor

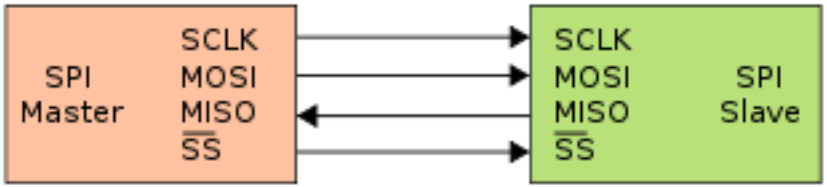


Just don't copy the Example! Make it from scratch in your own way! You need to understand what's happens! Play and Explore! Add Value to your code!

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

I2C Bus

- I^2C (Inter-Integrated Circuit), is a multi-master, multi-slave, single-ended, serial computer bus
- It is typically used for attaching lower-speed peripheral ICs to processors and microcontrollers.
- I^2C is typically spelled I2C (pronounced I-two-C)
- The I^2C bus was developed in 1982 by Philips Semiconductor.
- The I^2C protocol requires only 2 wires for connecting all the peripheral to a microcontroller.

https://learn.sparkfun.com/tutorials/i2c

http://en.wikipedia.org/wiki/I2C

Fritzing

- Fritzing is an open source software for design of electronics hardware and creating circuit diagrams
- Documentation is important!
- Creating professional circuit diagrams is important, so you should consider using a tool like Fritzing for your circuit diagrams.
- http://fritzing.org

Arduino
PWM and Analog Out

Hans-Petter Halvorsen

Arduino Analog Out

- Arduino UNO has no built-in Analog Output Channels
- We need Analog Out for the Control Signal ($0 - 5V$)
- We will use a 2 different options:
  - Create a RC Lowpass Filter that converts PWM to Voltage
  - Use a DAC chip/IC (Digital to Analog Converter)
    - Such a chip uses either the SPI bus or the I2C bus

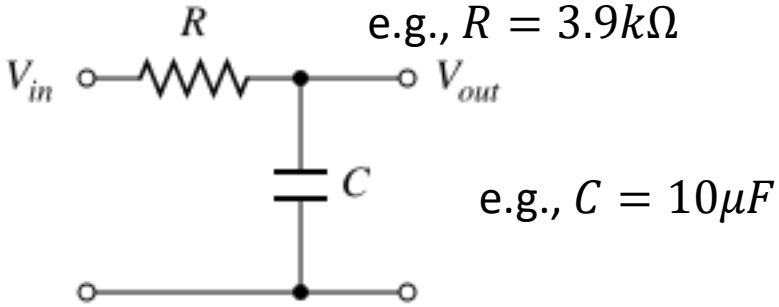
Analog Out

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

Option 1

Option 1: Convert PWM to Voltage

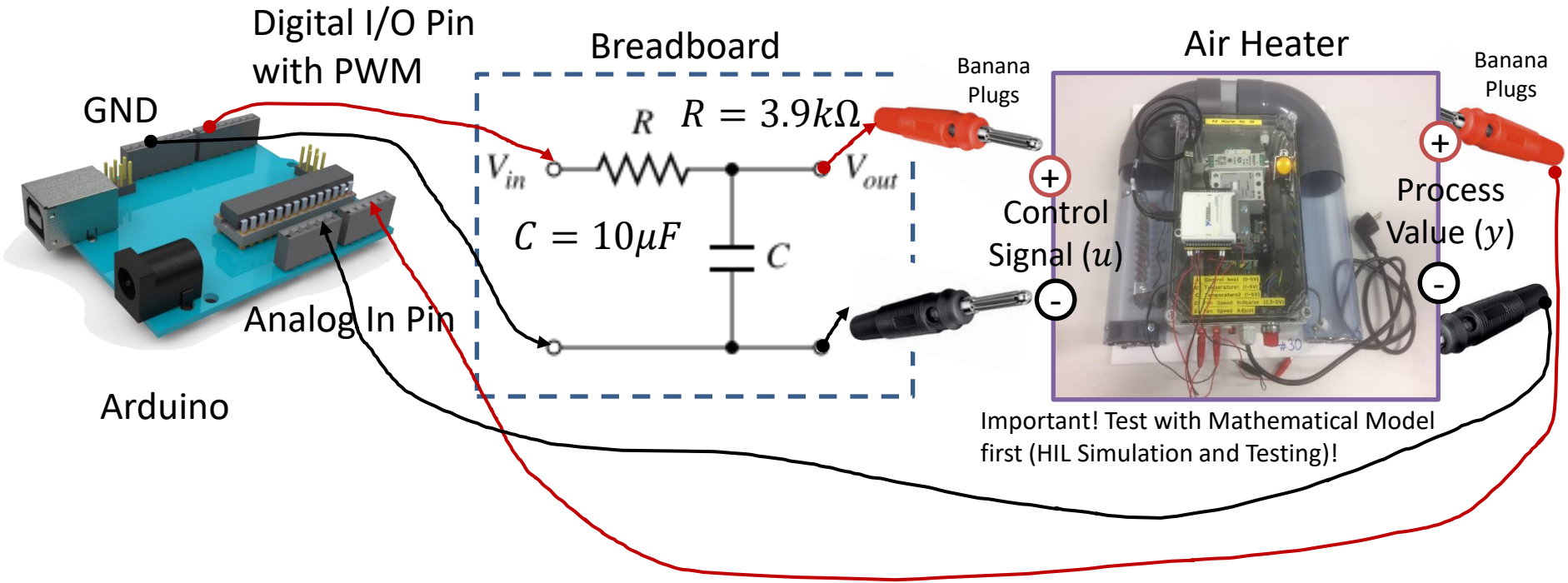
RC Lowpass Filter:



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

Convert PWM to Voltage Example

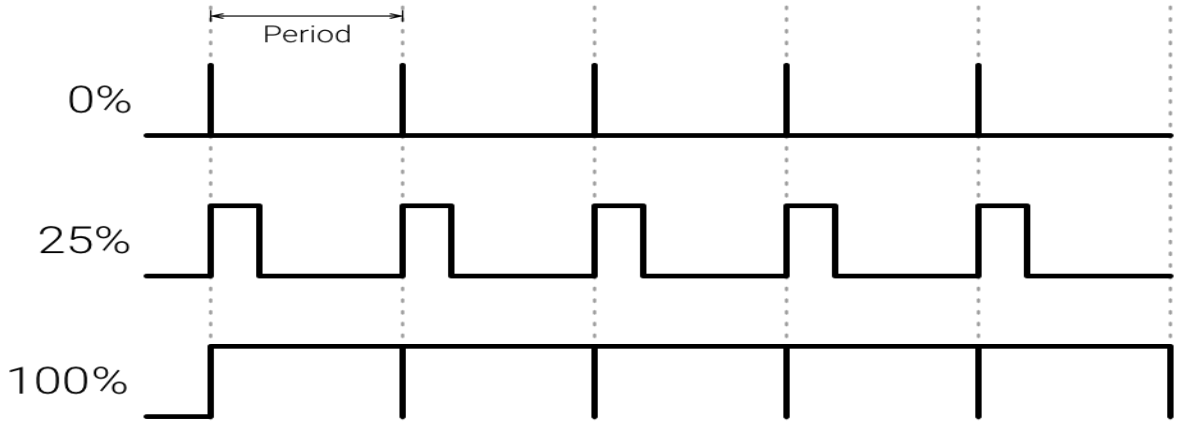
Below you see an example how you can wire:



PWM

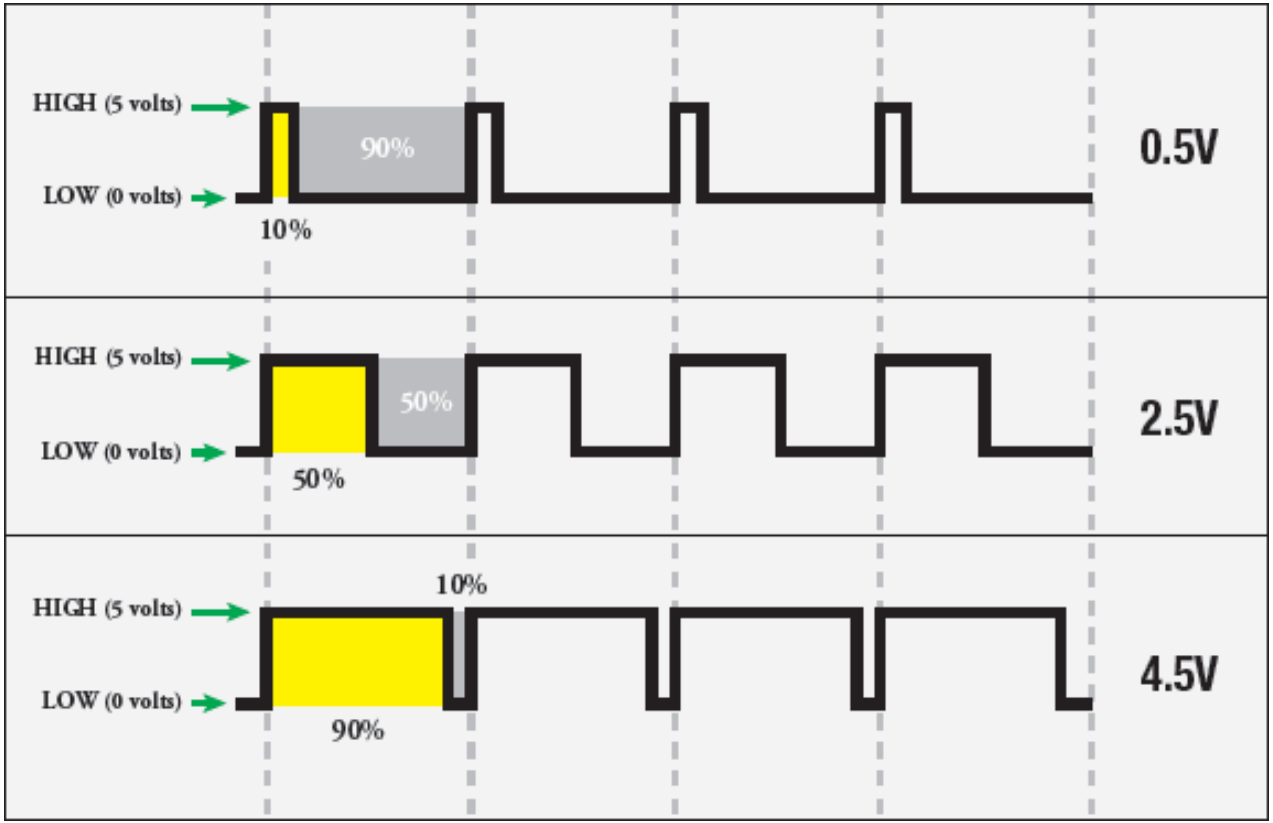
PWM is a digital (i.e. square wave) signal that oscillates according to a given *frequency* and *duty cycle*.
The frequency (expressed in Hz) describes how often the output pulse repeats.
The period is the time each cycle takes and is the inverse of frequency.
The duty cycle (expressed as a percentage) describes the width of the pulse within that frequency window.

You can adjust the duty cycle to increase or decrease the average "on" time of the signal. The following diagram shows pulse trains at 0%, 25%, and 100% duty:



https://developer.android.com/things/sdk/pio/pwm.html

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.)

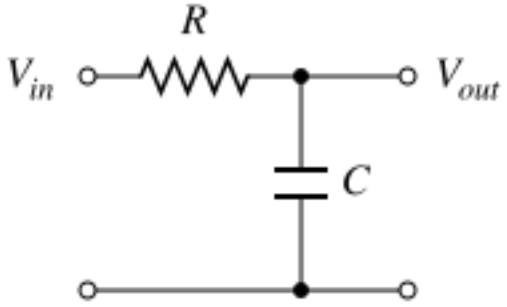
$$0 - 5V \rightarrow 0 - 255 \rightarrow y(x) = 51x$$
$$u = 0V \rightarrow analogWrite(0)$$
$$u = 5V \rightarrow analogWrite(255)$$
$$u = xV \rightarrow analogWrite(51 * x)$$

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

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

Design and Analysis

For a deeper understanding, you could consider doing the following:



- 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.

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

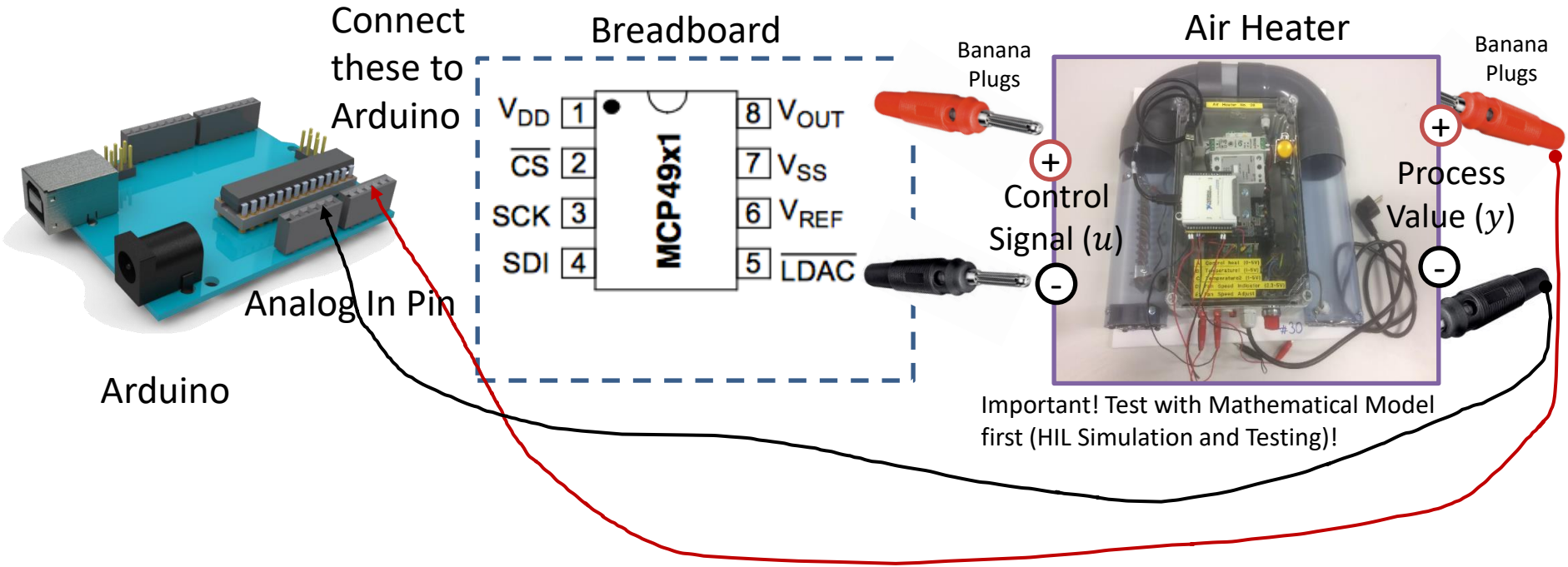
- Create and simulate a PWM signal in LabVIEW.
- Create the RC Lowpass Filter transfer function in LabVIEW.
- Design and Analyze the RC Lowpass Filter based on simulations (Time domain and Frequency domain) in LabVIEW. Find proper values for R and C. Find Bandwidth/Cut-off frequency, etc.

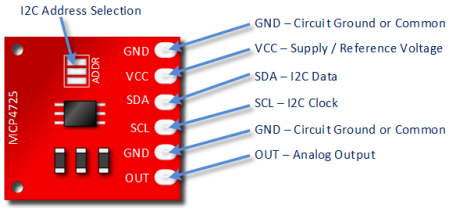
Examples:
http://techteach.no/simview/rc_circuit/
http://techteach.no/simview/pwm_control/

Option 2

Option 2: Use a DAC chip

Below you see an example how you can wire:





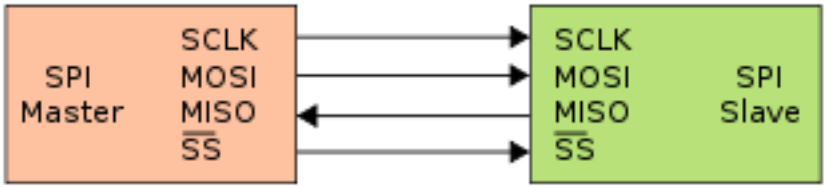
Using a DAC chip

- DAC – Digital to Analog Converter
- Use, e.g., Microchip MCP4911, MCP4725 or similar
- **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

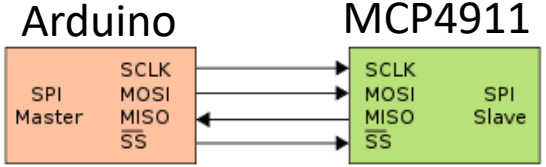
https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi

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

DAC

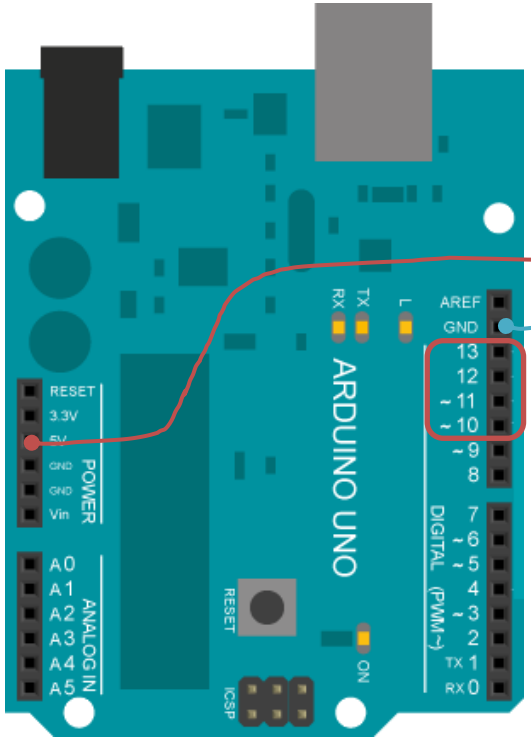
MCP4911: 10-bit single DAC



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.

$V_{SS} = 5V$

$V_{DD} = 0V$

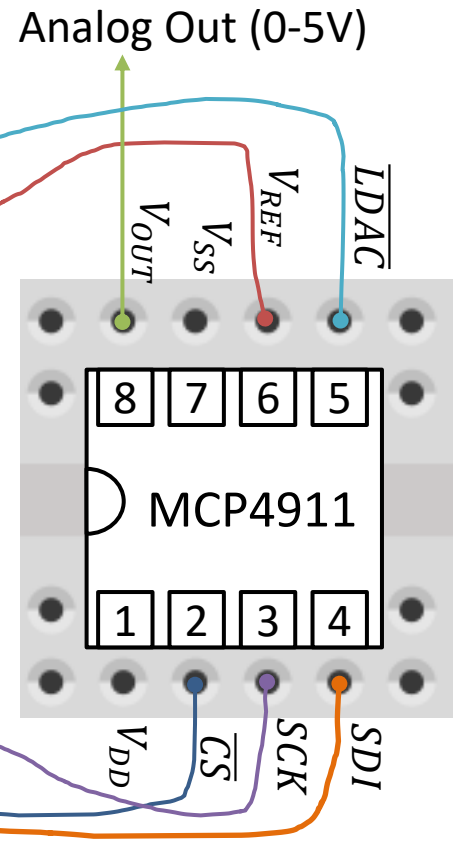


SCK (13)

MISO (12)

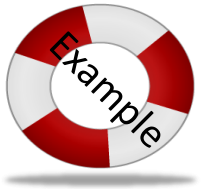
MOSI (11)

SS (10)



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

MCP49xx Arduino Library Example



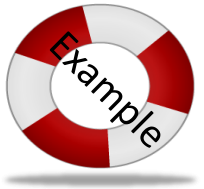

```
#include <SPI.h>            //Include the Arduino SPI Library
#include <DAC_MCP49xx.h>    //Include the MCP49xx Arduino Library

// The Arduino pin used for the slave select / chip select
#define SS_PIN 10

//Set up the DAC DAC MCP4911
DAC_MCP49xx dac(DAC_MCP49xx::MCP4911, SS_PIN);

void setup()
{
}

void loop()
{
  double u; //Control Signal
  // For MCP4911, use values below (but including) 1023 (10 bit)
  u = 255; //Simulating the Control Value
  dac.output(u);
  delay(5000);

  u = 512; //Simulating the Control Value
  dac.output(u);
  delay(5000);
}
```

The control signal (u) should come from the PI/PID controller function.
It need to be converted from 0-5V (or 0-100%) -> 0-1023 before we send it to the DAC

Connect the circuit (Arduino + MCP4911) on a breadboard. Use a multi-meter so see if you get the correct output signal

https://github.com/exscape/electronics/tree/master/Arduino/Libraries

MCP49xx Arduino Library Example




```
#include <SPI.h>           //Include the Arduino SPI Library
#include <DAC_MCP49xx.h>   //Include the MCP49xx Arduino Library

// The Arduino pin used for the slave select / chip select
#define SS_PIN 10

DAC_MCP49xx dac(DAC_MCP49xx::MCP4911, SS_PIN);

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

void loop()
{
  double u; //Control Signal
  int aiPin = 0;
  int aiValue;

  for (int i=0; i<1023; i++)
  {
    u = i;
    dac.output(u);

    aiValue = analogRead(aiPin);
    Serial.print("AIValue=");
    Serial.println(aiValue);

    delay(1000);
  }
}
```

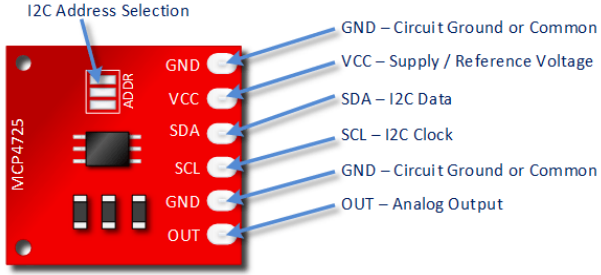
Connect the circuit (Arduino + MCP4911) on a breadboard. Use a multi-meter to see if you get the correct output signal.

On the Multimeter you should see the output slowly increasing from ~0V to ~5V with intervals of 1000ms.

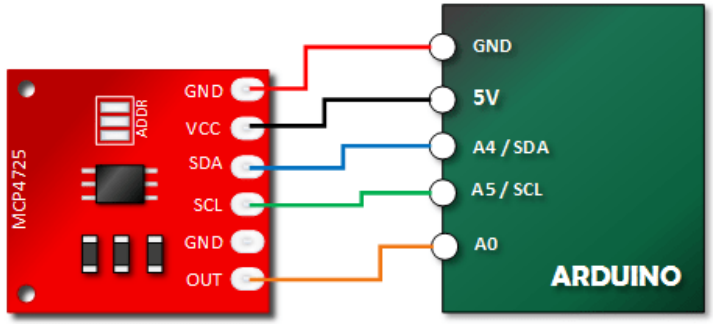
You can also connect the output from the DAC to an Analog Input Pin on the Arduino. Write the value to the Serial Monitor.

Alternative Solution

MCP4725



12-bit resolution
I2C Interface



The MCP4725 is a little more expensive (than MCP49xx), but simpler to use.

http://henrysbench.capnfatz.com/henrys-bench/arduino-output-devices/arduino-mcp4725-digital-to-analog-converter-tutorial/

Congratulations! - You are finished with the Task

Embedded Control System Using Arduino

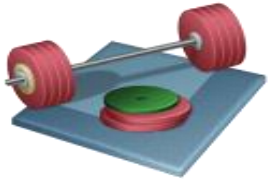
Hans-Petter Halvorsen

PID Control

Theory

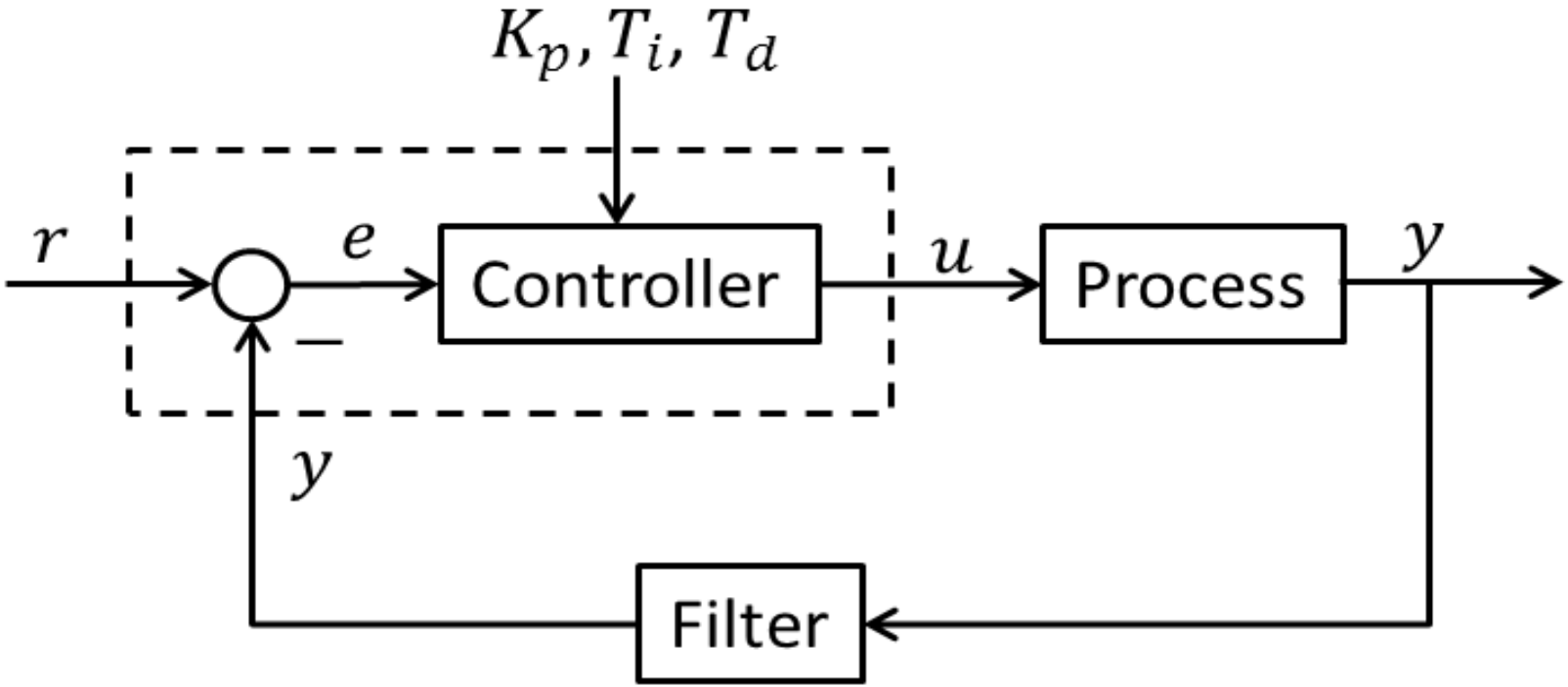
Hans-Petter Halvorsen, M.Sc.

Arduino PID Controller

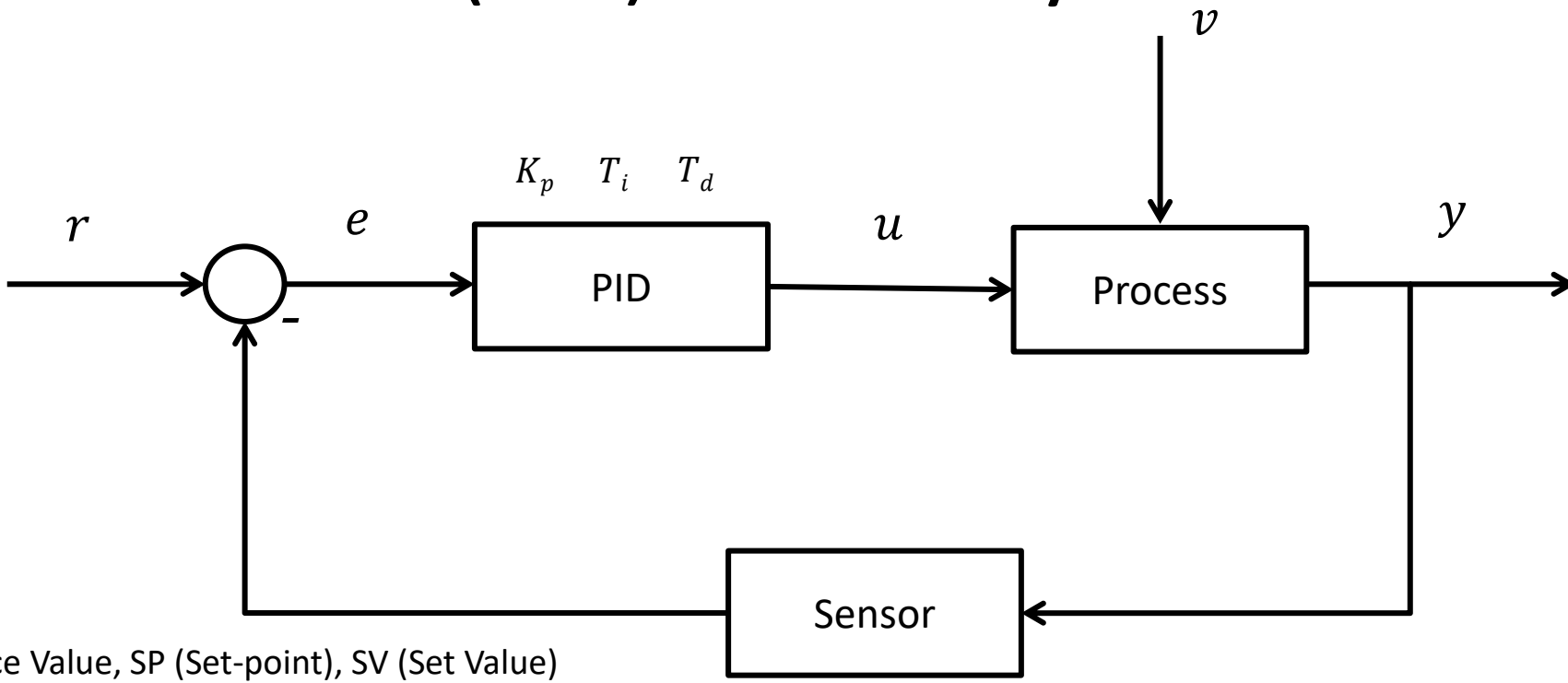


- Find a discrete PID algorithm (using pen and paper) – or you probably already have one you can use
- Implement the PID algorithm using Arduino Programming

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.

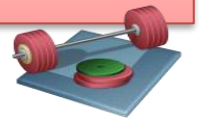
Congratulations! - You are finished with the Task

Lowpass Filter

Hans-Petter Halvorsen

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

Arduino Library

Hans-Petter Halvorsen

What is an Arduino Library?

- The Arduino environment can be extended through the use of libraries, just like most programming platforms.
- You can also create your own libraries
- It is a good way to structure your code
- It is also a good way to share and reuse the code in different applications and projects
- It is pretty easy to create you own library
- Basically, An Arduino Library is just a C++ Class containing one or more functions
- It is recommended that you put your PID controller, Scaling and Lowpass filter into a Arduino Library, i.e., a Class containing 3 functions

Arduino Libraries

It is recommended that you 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

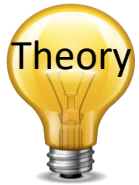
Congratulations! - You are finished with the Task

HIL Simulation and Testing

HIL – Hardware in the Loop

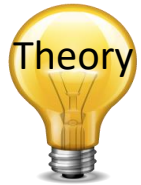
Hans-Petter Halvorsen

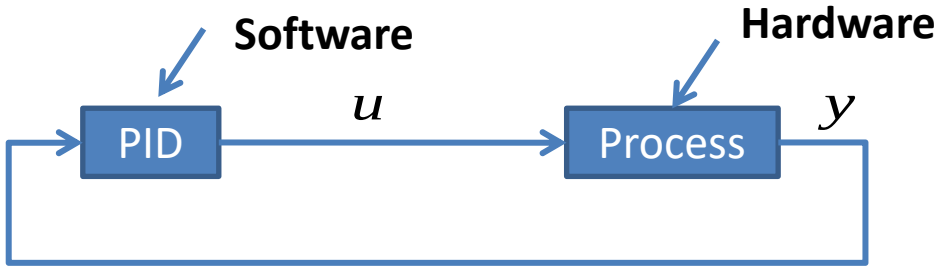
HIL Lab - Background

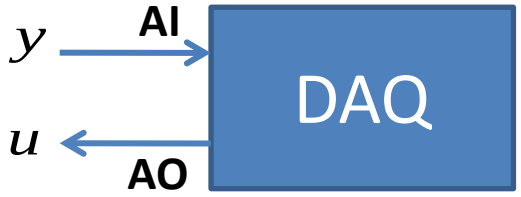


- Typically, a simulator communicates with an “ECU” (“Electronic Control Unit”) via ordinary I/O. Such a system - where the real controller is controlling a simulated process is denoted Hardware-in-the-loop (HIL) simulation.
- It is important to test the hardware device on a simulator before we implement 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 PID controller on a model, and if everything is OK we will implement the controller on the real system.

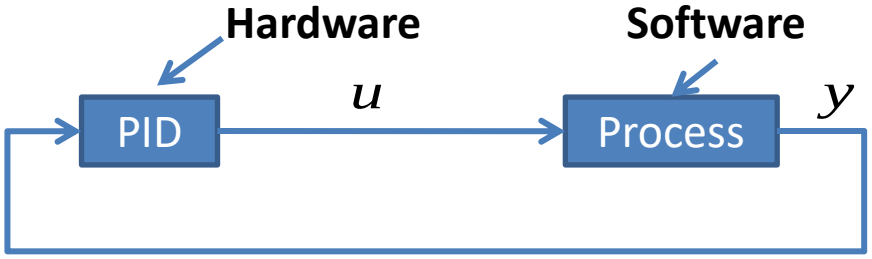
Traditional Process Control using Software for Implementing the Control System

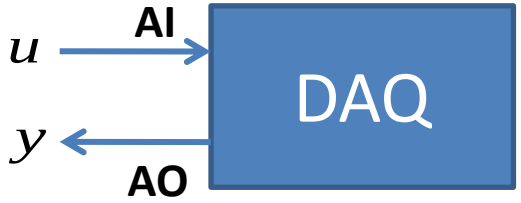




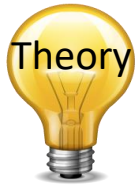


HIL Simulation



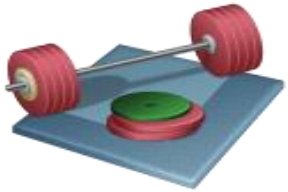


HIL Simulation



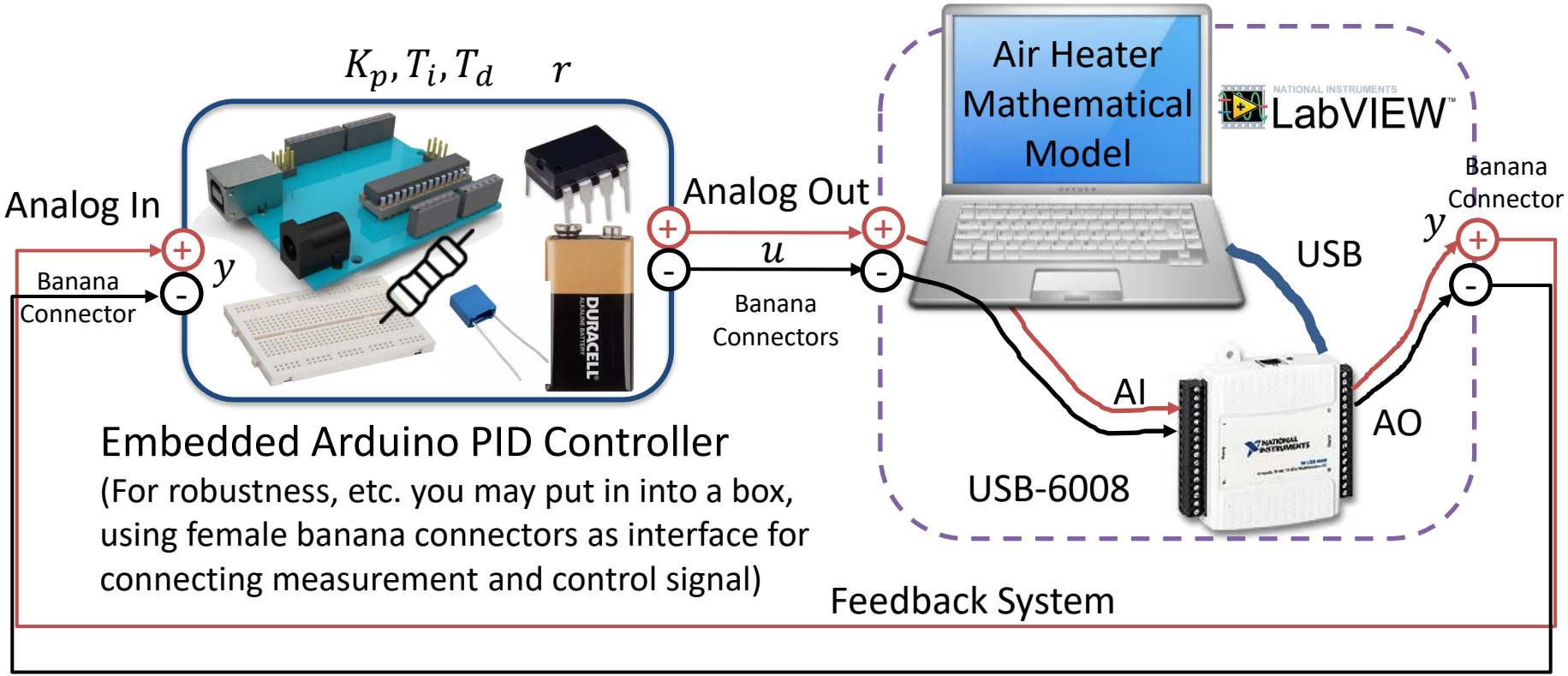
- Hardware-in-the-loop (HIL) simulation is a technique that is used in the development and **test of complex process systems**
- The HIL simulation includes a **mathematical model** of the process and a hardware device/ECU you want to test, e.g. an industrial PID controller we will use in our example.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

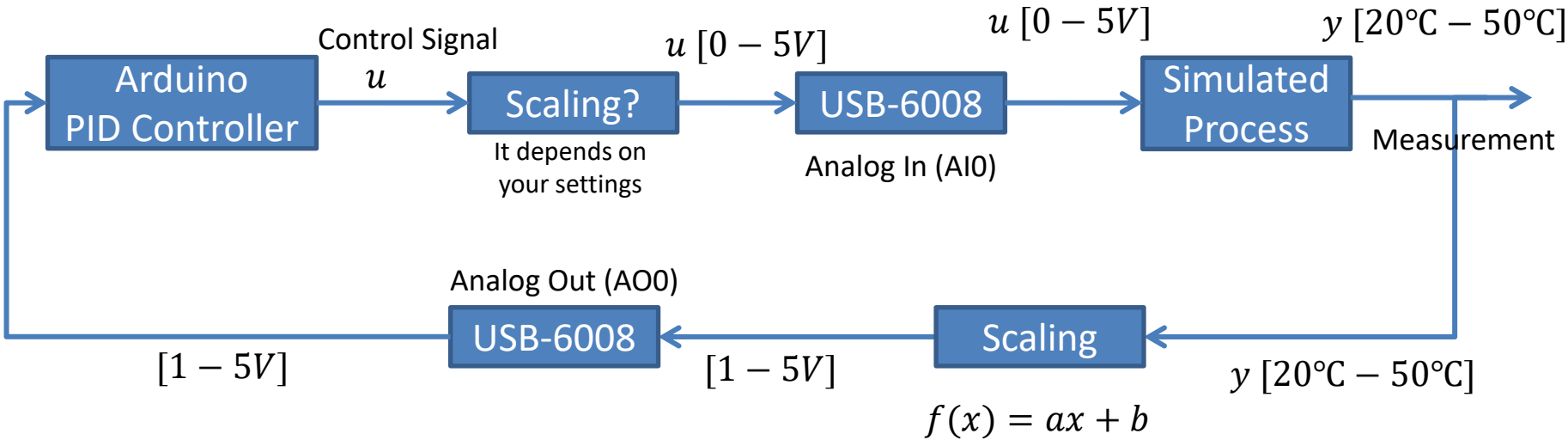


- Make sure to test your Embedded Arduino PID Controller using HIL Simulation and Testing principles before you use it on the Real Air Heater process
- Make sure it works as expected and without risk of damaging the Air Heater process

HIL Simulation and Testing



HIL Simulation using Arduino PID Controller

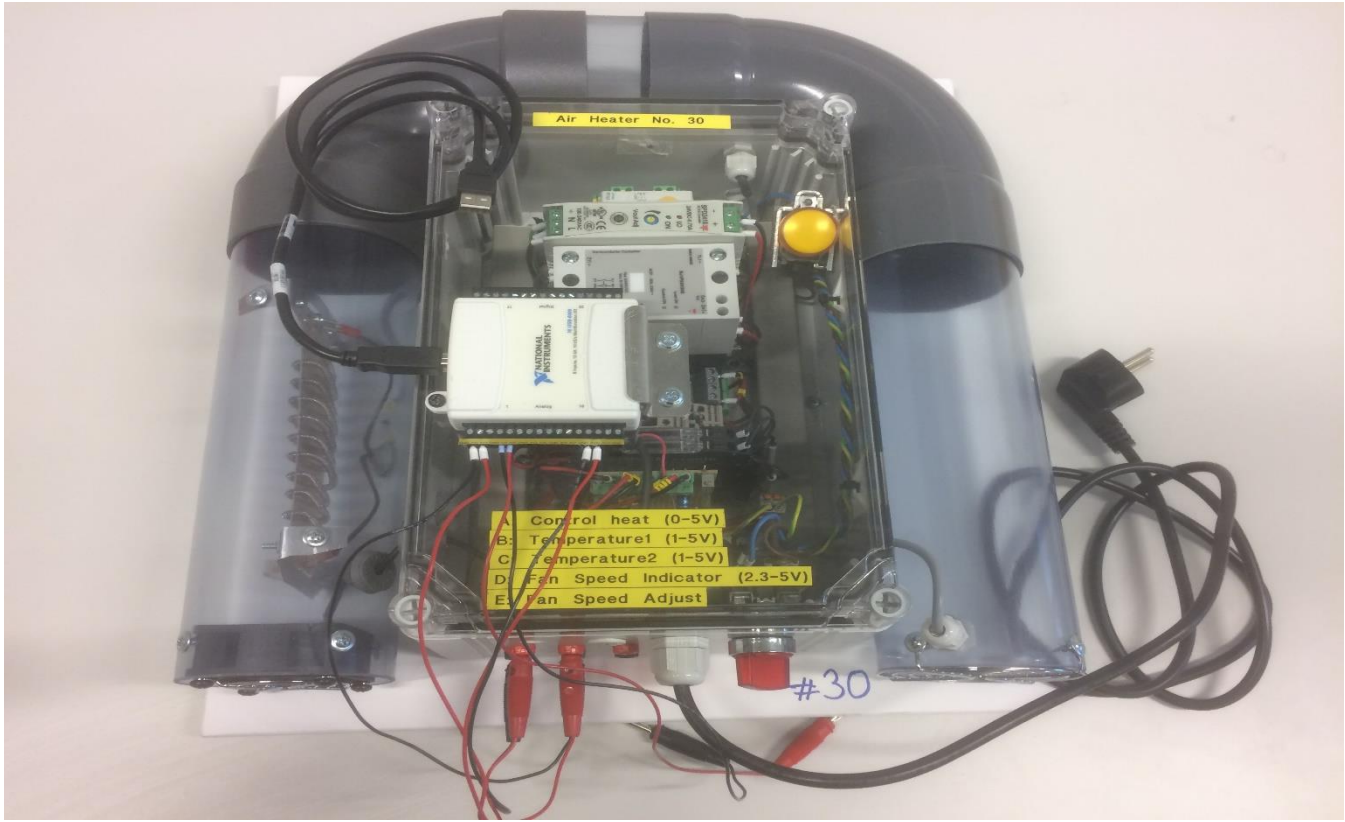


Congratulations! - You are finished with the Task

Modelling and Simulation

Hans-Petter Halvorsen

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

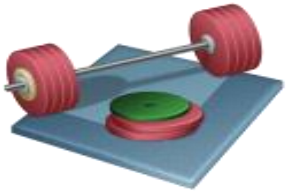
Air Heater Mathematical Model

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

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

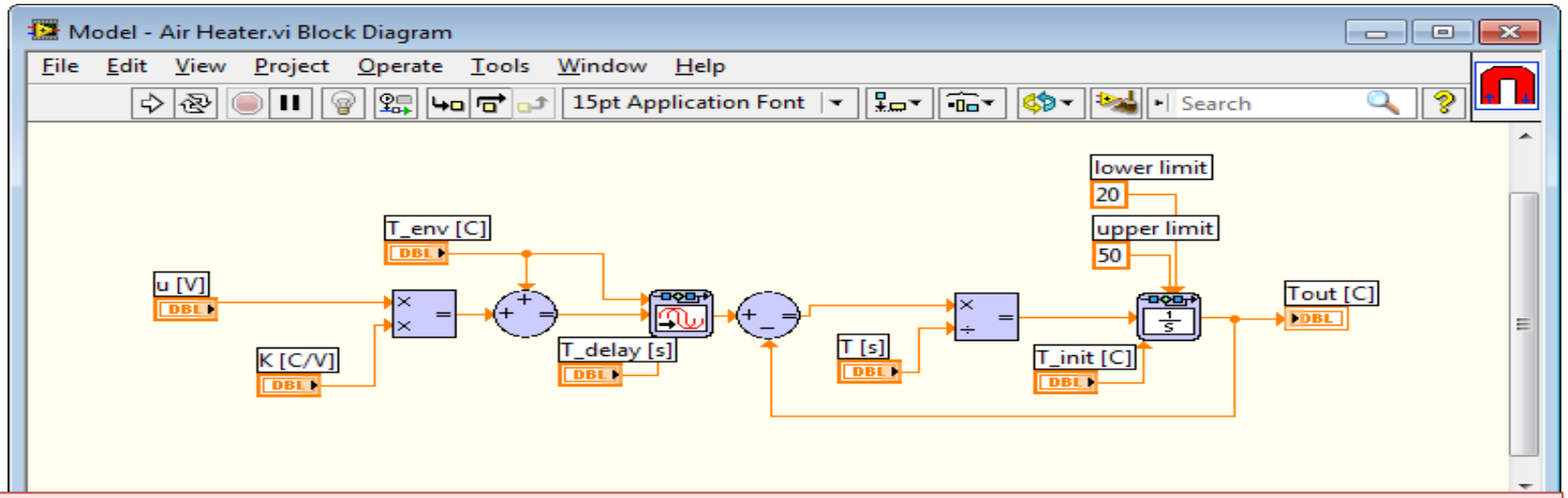
Use the “Black Box Model” when you are not in the laboratory

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:

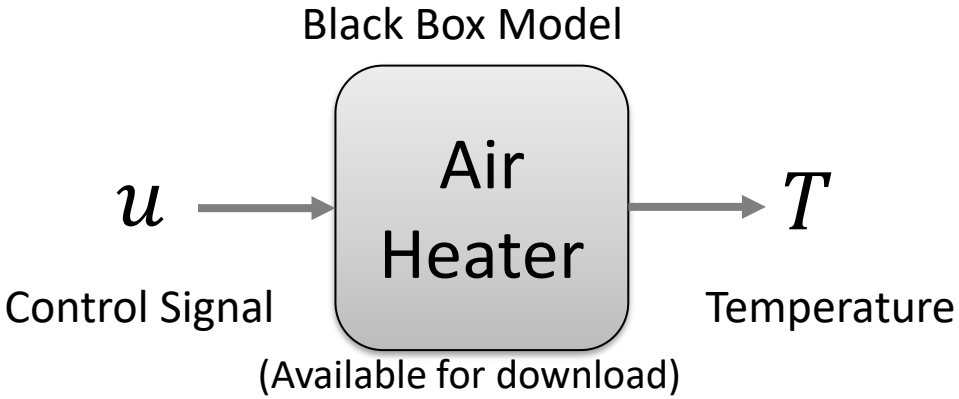


Note! This model is implemented in a so-called "**Simulation Subsystem**" (which is recommended!!!)

“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”



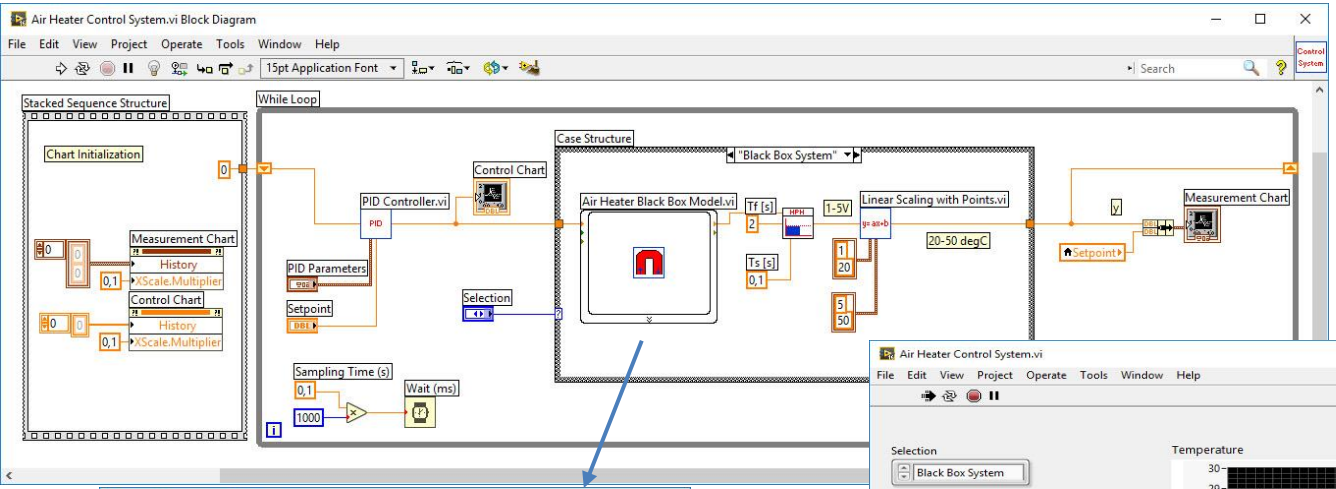
You can assume that the following model is a good representation of the “Black Box Model”:

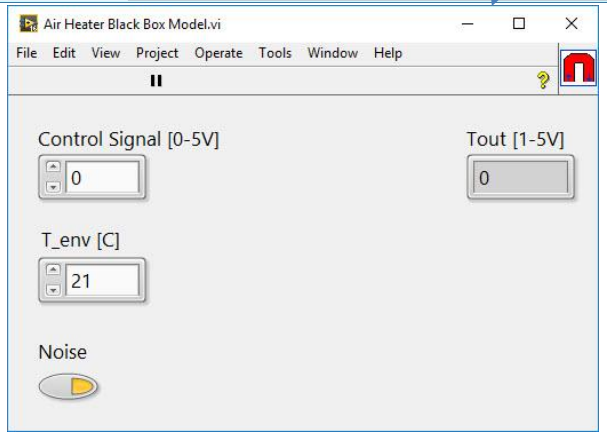
$$\dot{T}_{out} = \frac{1}{\theta_t}\{-T_{out} + [K_h u(t-\theta_d) + T_{env}]\}$$

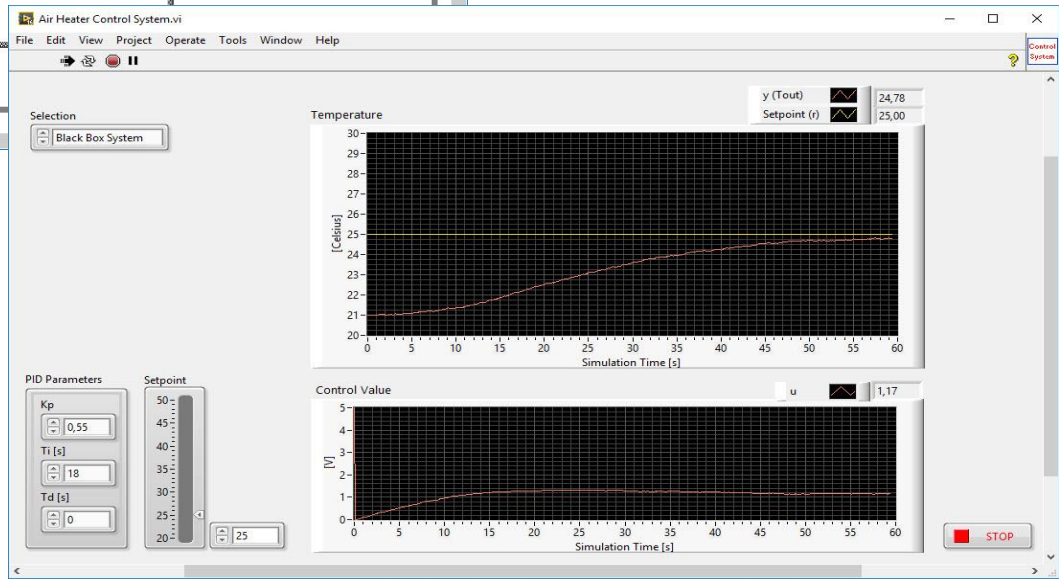
This means you need to need to find $\theta_t, K_h, \theta_d, T_{env}$

T_{env} is the temperature in the room

“Real Process” → “Black Box Simulator”

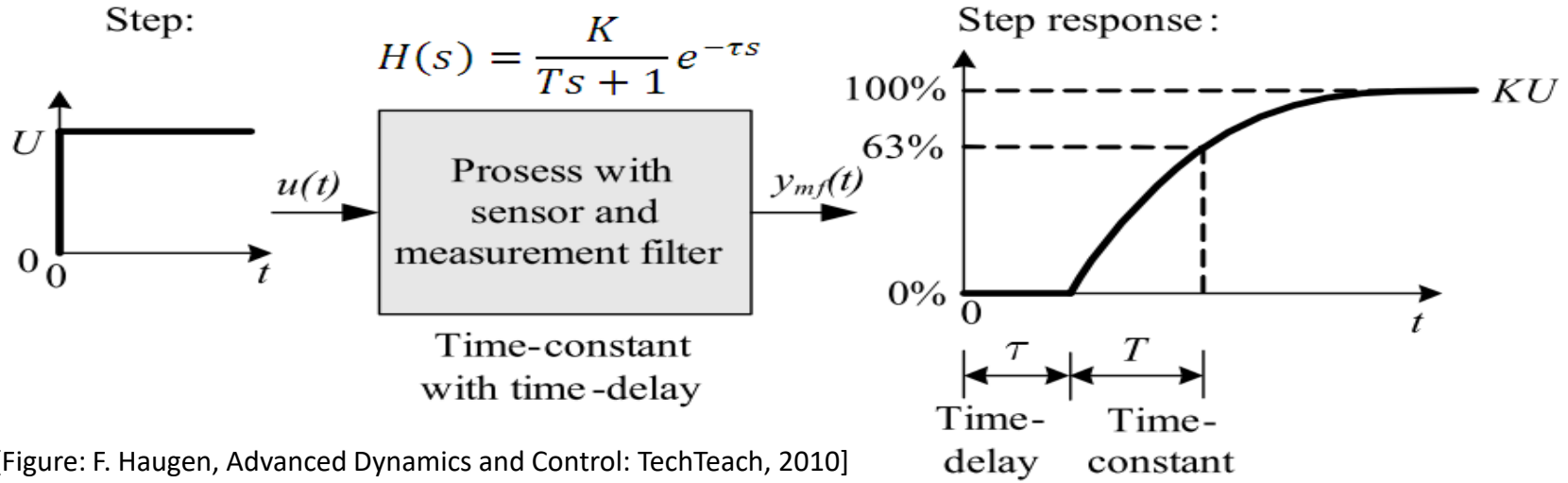






Here we see an example where we control the “Black Box” Model, which we “pretend” is the Real System

Step Response Method

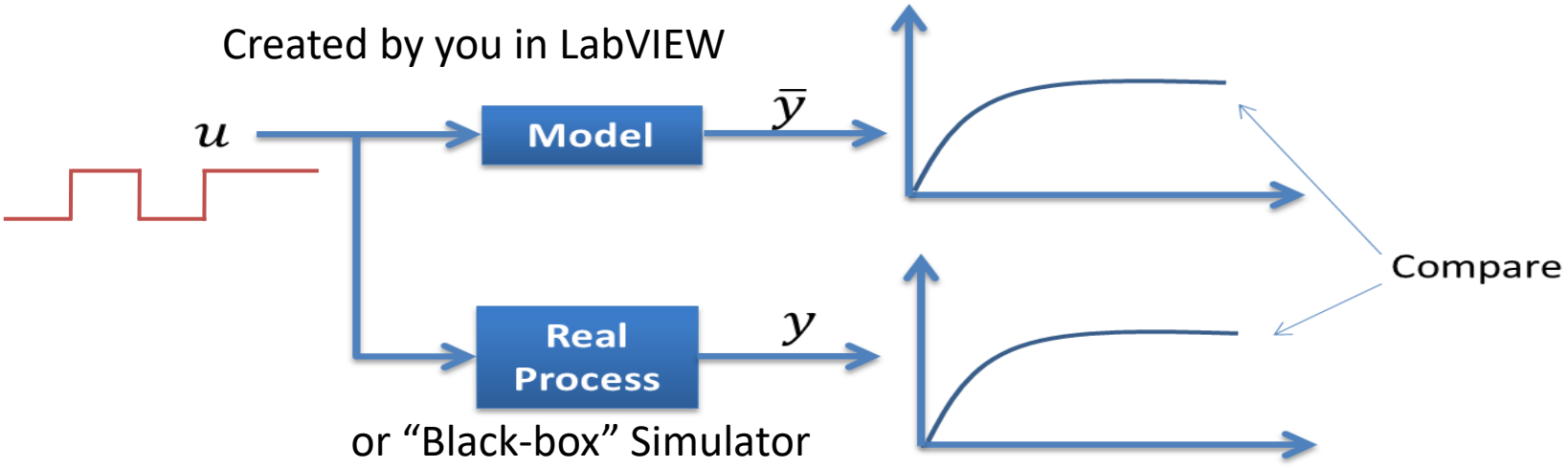


[Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

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)

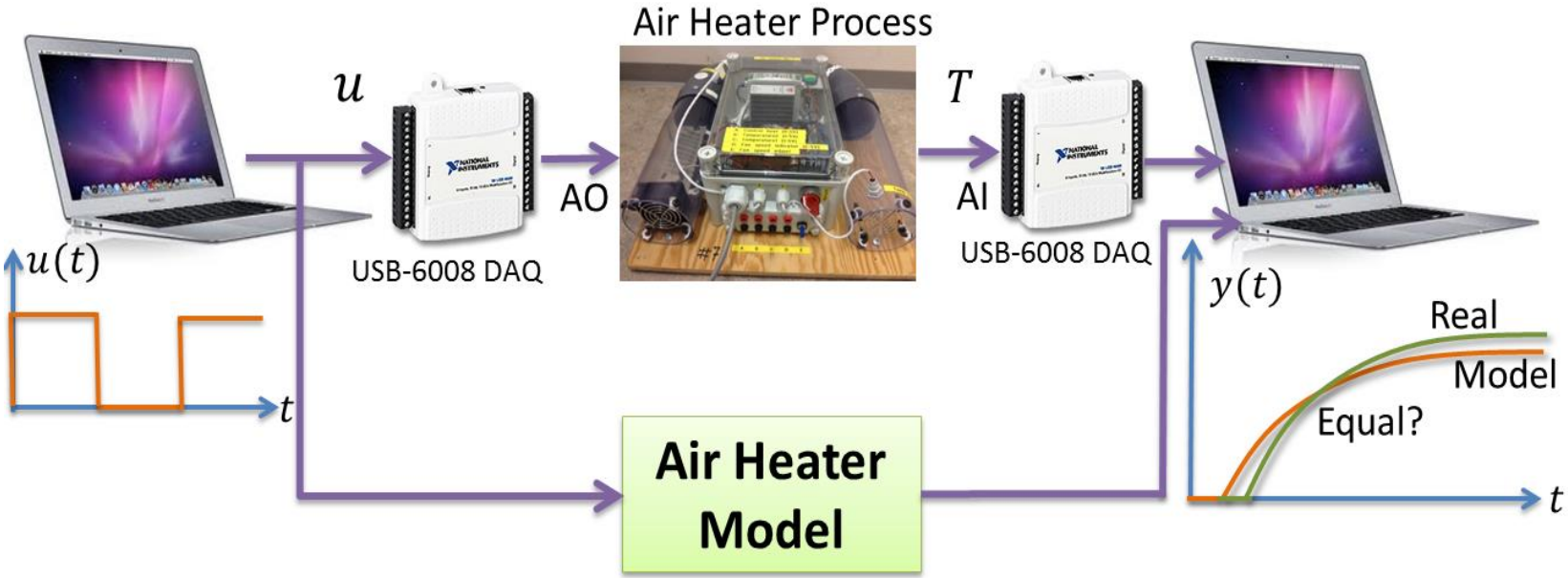
Step Response Method

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)

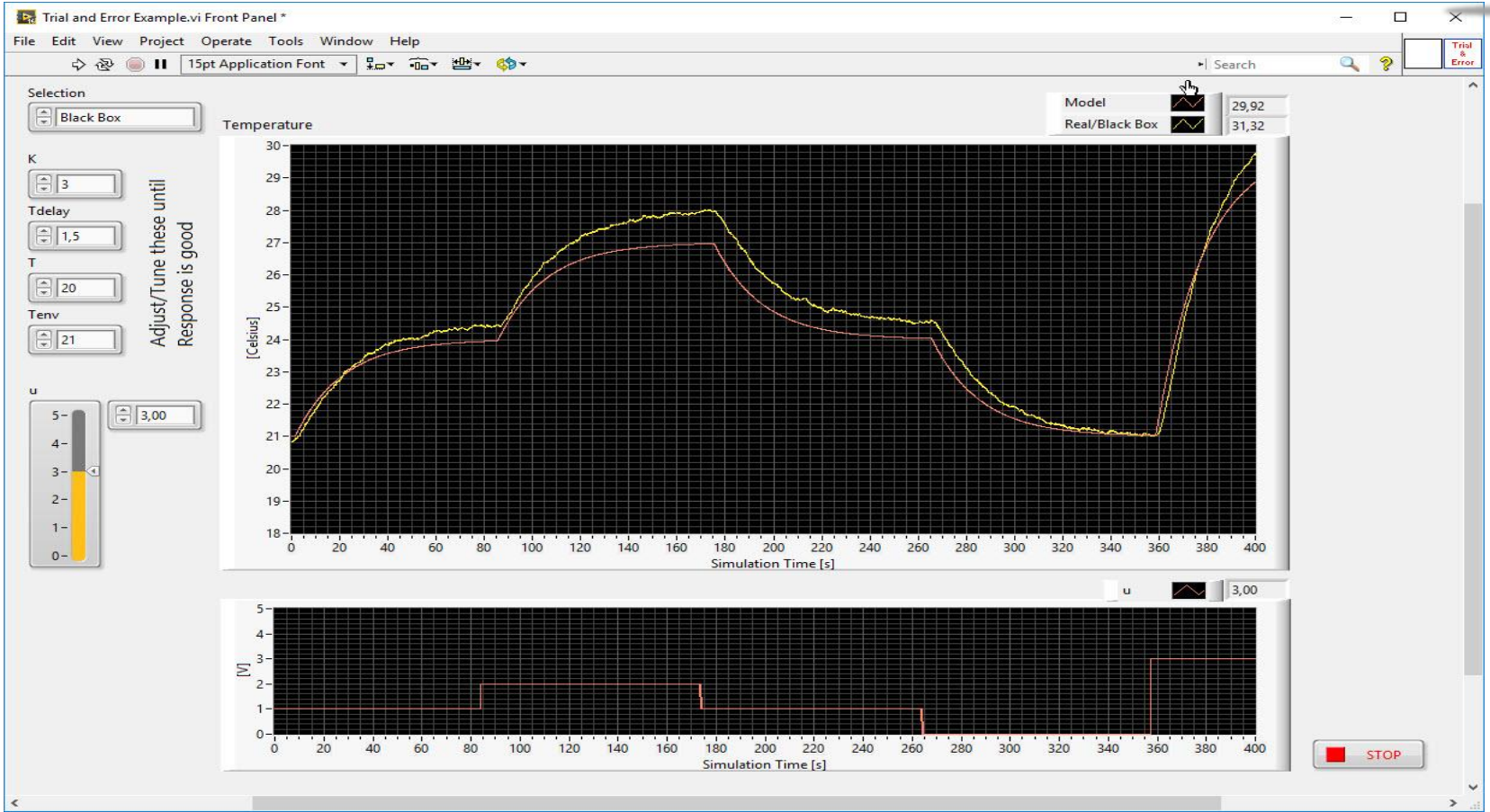
Model Validation



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

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

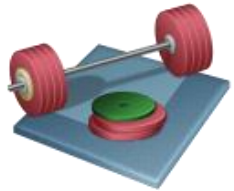


Congratulations! - You are finished with the Task

Air Heater Control System

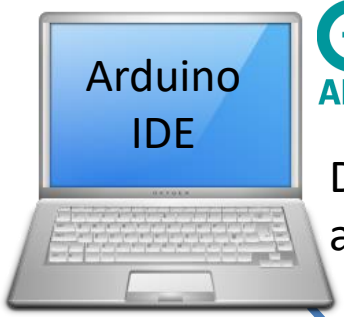
Hans-Petter Halvorsen

Air Heater Control System



- Control the real Air Heater process using your Arduino PID Controller
- You may, e.g., use a Potentiometer to change the Reference value/Setpoint
- 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

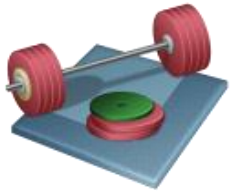
y

Embedded Arduino PID Controller

Test first using HIL
Simulation and Testing

Feedback System

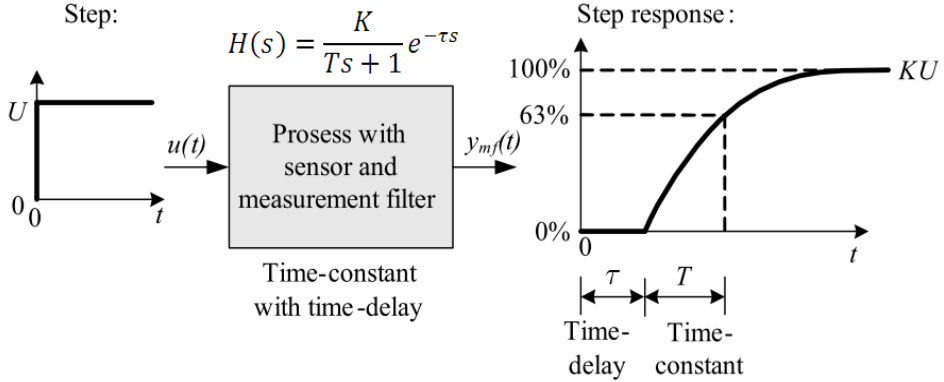
PI(D) Controller Design



- 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

Data Publishing with ThingSpeak

Hans-Petter Halvorsen

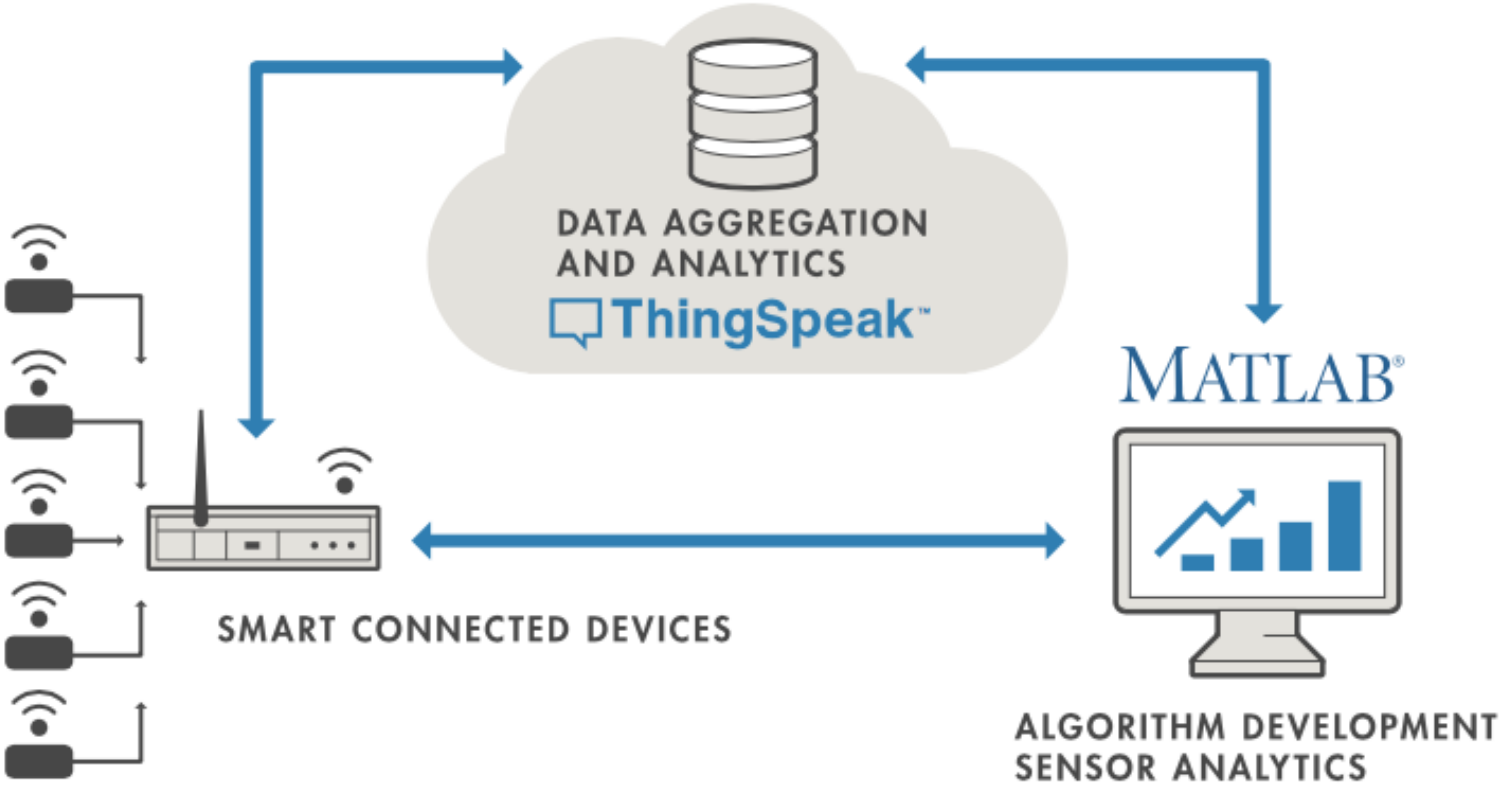
ThingSpeak

- ThingSpeak is a IoT Cloud Service 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


DATA AGGREGATION
AND ANALYTICS
ThingSpeak™
MATLAB®
SMART CONNECTED DEVICES
ALGORITHM DEVELOPMENT
SENSOR ANALYTICS

ThingSpeak

- ThingSpeak is an IoT analytics platform service that lets you collect and store sensor data in the cloud and develop Internet of Things applications.
- The ThingSpeak service also lets you perform online analysis and act on your data. Sensor data can be sent to ThingSpeak from any hardware that can communicate using a REST API
- ThingSpeak has a Web Service (REST API) that lets you collect and store sensor data in the cloud and develop Internet of Things applications (it also has MQTT API).

https://thingspeak.com

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

Remote Access and Publishing

Select one of the following alternatives (Alt 1 is simpler, Alt 2 is more sophisticated and more challenging):

1. Use a PC for remote Monitoring. Publish Process Data to a ThingSpeak (u, y), etc.
2. Publish your Data to ThingSpeak directly from Arduino
   - In this scenario you need a Wi-Fi/Ethernet Shield for Arduino UNO or you need an Arduino with built-in Wi-Fi
   - This alternative is more challenging and it is also more cumbersome to make it work properly
   - You will need to include different Arduino Libraries for Wi-Fi/Ethernet, ThingSpeak, etc. This can be a problem when using Arduino UNO due to limited memory (An alternative is Arduino Mega)

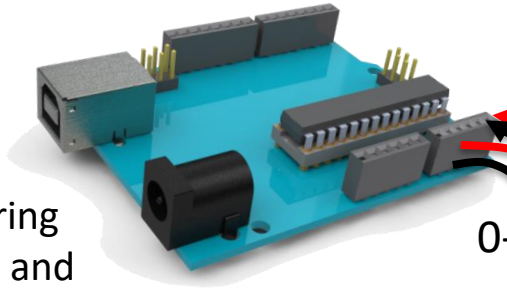
Alternative 1

PC for remote Monitoring

Hans-Petter Halvorsen

Arduino PID + Real Air Heater + PC for Monitoring

Arduino PID Controller

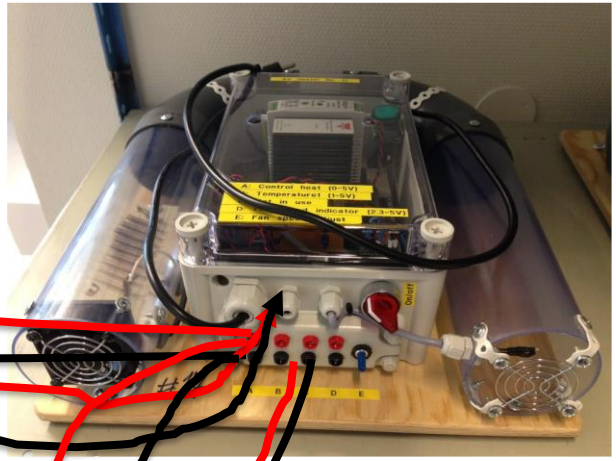


Process Value

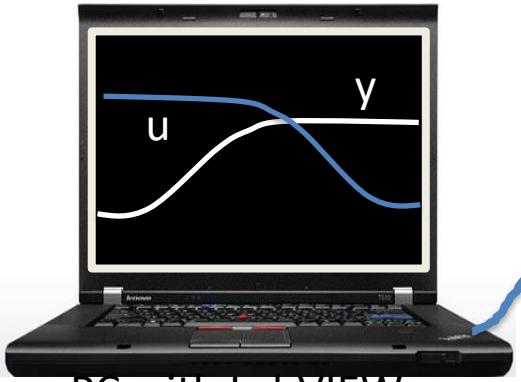
1-5V

0-5V

Control Signal



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



PC with LabVIEW

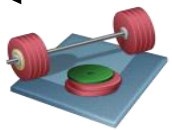
USB

Process Value

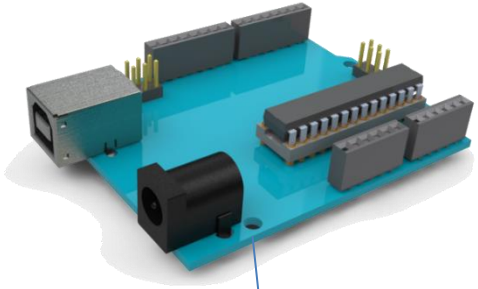
1-5V

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. Send Data to ThingSpeak



USB

1

LabVIEW

The Cloud

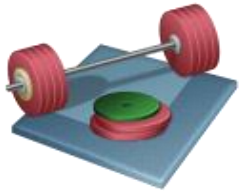
2

Data Collection

ThingSpeak

www.ThingSpeak.com

3 Data Analysis

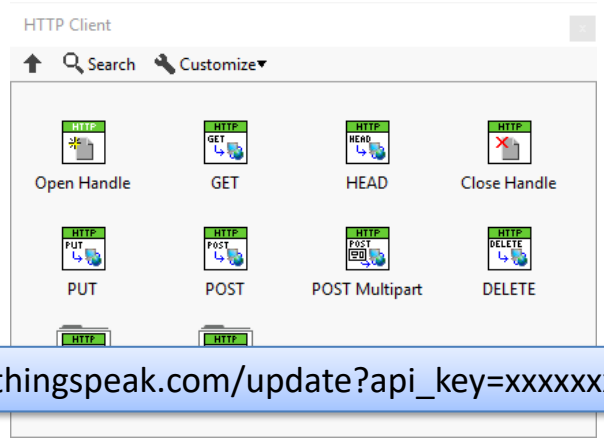


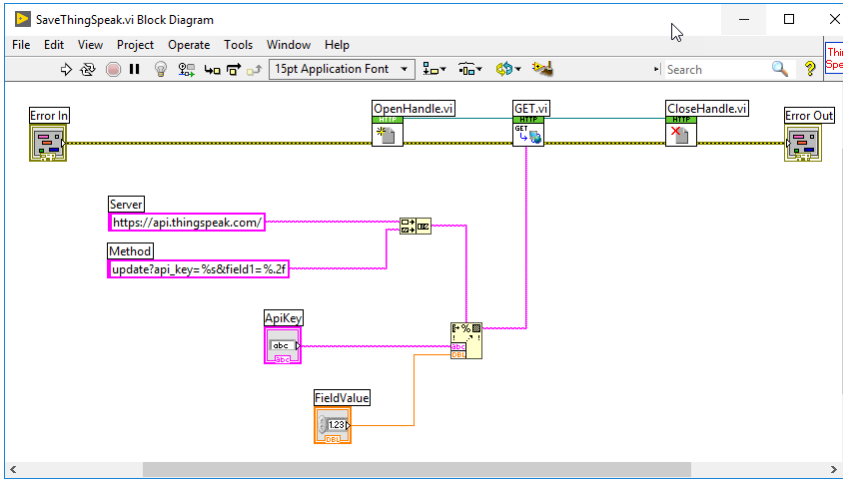
Save Process Data into the Cloud using the ThingSpeak Service.

Do some Analysis of the Data in MATLAB

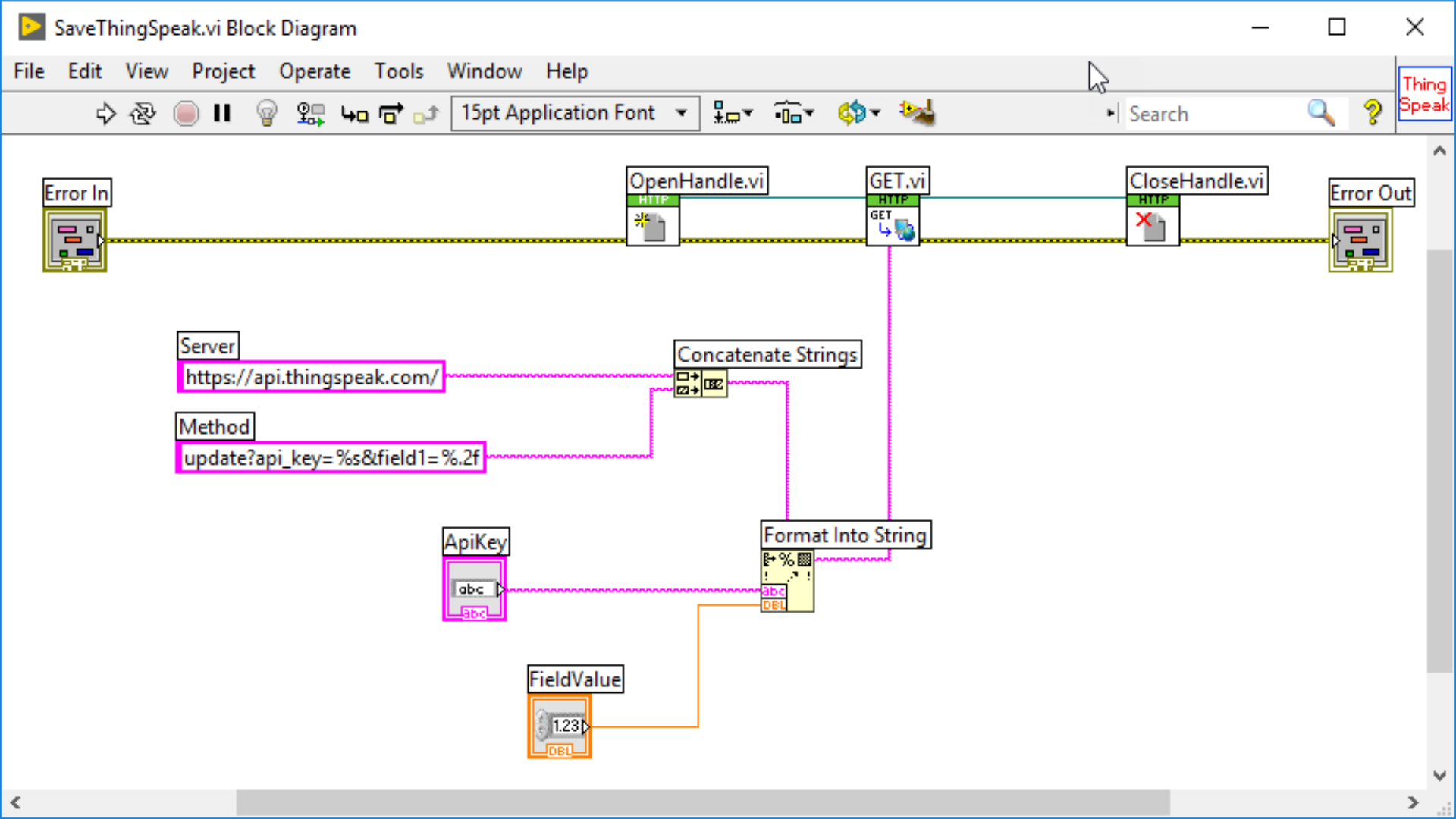
ThingSpeak + LabVIEW

- ThingSpeak uses standard HTTP REST API, which can be used from any kind of Programming Language, including LabVIEW
- In LabVIEW you can use the HTTP client VIs





https://api.thingspeak.com/update?api_key=xxxxxxx&field1=22.5


SaveThingSpeak.vi Block Diagram
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Thing Speak
Error In
OpenHandle.vi
GET.vi
CloseHandle.vi
Error Out
Server
https://api.thingspeak.com/
Concatenate Strings
Method
update?api_key=%s&field1=%.2f
ApiKey
Format Into String
FieldValue
1.23

We can also Set and Read PID Parameters Remotely

Set Kp Remotely Example:

Enter the following in a Web Browser (or from a Programming Language like LabVIEW)

We set Kp=2

https://api.thingspeak.com/update?api_key=<WriteKey>&field3=2

Read Kp Remotely Example:

https://api.thingspeak.com/channels/<ChannelId>/fields/3/last.json?key=<ReadKey>

Response in Browser: {"created_at":"2017-06-26T07:41:54Z","entry_id":1270,"field3":"2"}

We read Kp=2

Alternative 2

Arduino + ThingSpeak

Hans-Petter Halvorsen

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, MATLAB and LabVIEW, etc.

https://thingspeak.com

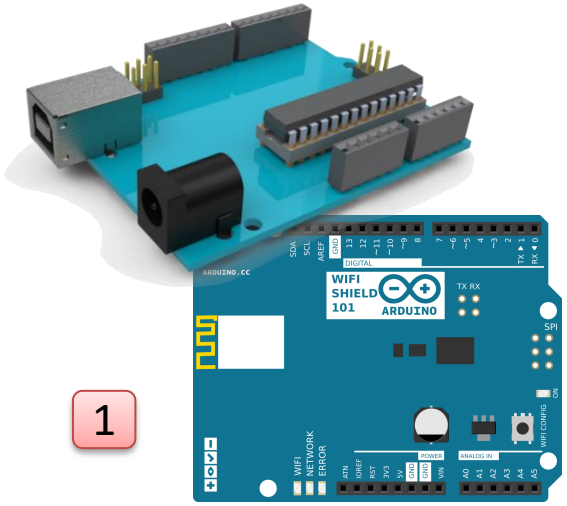
Arduino Example:
https://www.arduino.cc/en/Tutorial.WiFi101ThingSpeakDataUploader

Data Collection

2

ThingSpeak

www.ThingSpeak.com

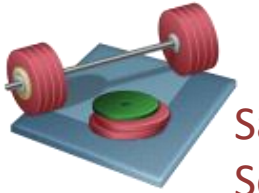


1

Arduino + Wi-Fi Shield

The Cloud

3 Data Analysis

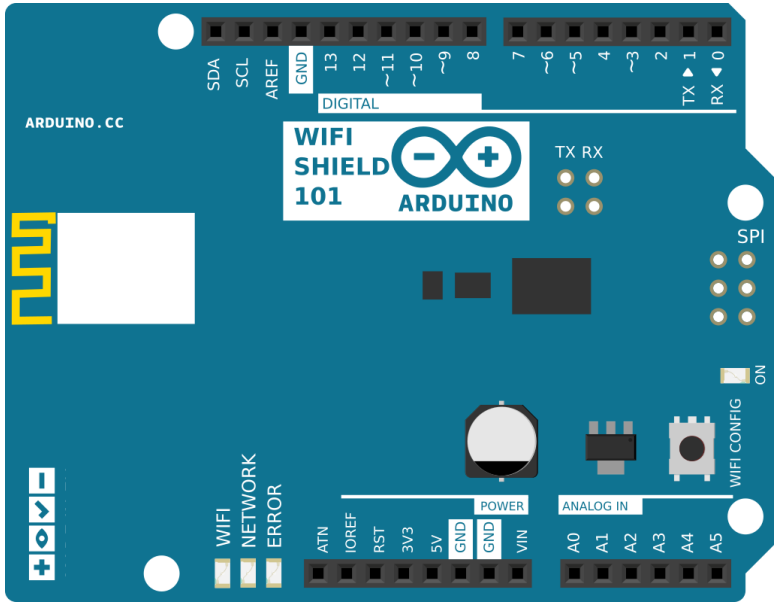


Save Process Data into the Cloud using the ThingSpeak Service.

Do some Analysis of the Data in MATLAB

Can you make your Arduino remotely receive the Controller Parameters as well?

Arduino Wi-Fi/Ethernet Shield or similar



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).

The Arduino Wi-Fi Shield or Ethernet Shield are available in the Laboratory

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

Set and Read PID Parameters Remotely

Set Kp Remotely Example:

Enter the following in a Web Browser (or from a Programming Language like LabVIEW, MATLAB, etc)

We set Kp=2

https://api.thingspeak.com/update?api_key=<WriteKey>&field3=2

Read Kp Remotely Example:

https://api.thingspeak.com/channels/<ChannelId>/fields/3/last.json?key=<ReadKey>

Response in Browser: {"created_at":"2017-06-26T07:41:54Z","entry_id":1270,"field3":"2"}

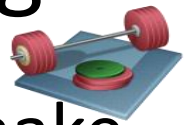
We read Kp=2

Congratulations! - You are finished with the Task

IoT and Cyber Security

Hans-Petter Halvorsen

Cyber Security and the Internet of Things



- IoT solutions and Data Security? How can we make sure our applications and data are safe?
- Security is crucial in IoT/IIoT Applications. Why?
- Give an overview of issues regarding IoT and Cyber Security in in context of your system.
- What can be (or what have you) done to protect the system (and data) you have created?
- How does ThingSpeak handle security?

Congratulations! - You are finished with the Task

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

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog