FeSCADA – Closed Loop Control (PIDs)

Introduction

A new utility called “PIDs” was added to FeSCADA software. With this feature the user can setup and run closed loop controls. The most general form of a closed loop control is the PID controller.

This chapter will cover the following topics:

1. Description – useful situations
2. Basics of Closed Loop Control (PID control)
3. Settings of a PID
4. PID Inputs and Outputs – Analog and PWM
5. Open Loop, Closed Loop and bumpless start
6. PID parameters tuning
7. Application examples
8. Conclusions

1) Description – useful situations

Proportional-Integral-Derivative (PID) controllers are used in most automatic process control applications in industry today to regulate flow, temperature, pressure, level, and many other industrial process variables. It is estimated that more than 95% of the control loops in industry are of PID type.

PID controllers are the workhorse of modern process control systems, as they automate regulation tasks that otherwise would have to be done manually. Much more practical than the typical on/off controller, PID controllers allow for much better adjustments to be made in the system.

FeSCADA can have access to multiple remote terminal units (RTU), each one with digital and analog inputs and outputs. Once a temperature sensor is connected to an analog input and a digital output to a heater’s relay, it is fairly simple to setup and use a PID to control that temperature.

On a computer there is no memory restrain, like it is on a small PLC or an Arduino. The user can setup and run two, three or 100 PIDs without any worry about the CPU speed or the memory limits. The only limit is for the speed of the process that can be controlled by a PID. FeSCADA is using a loop cycle time of 100ms. That means that the control algorithm is run 10 times a second.

A process time constant is a property of a system about how fast its variable can oscillate if stimulated from outside. This time constant has to be about 10 times bigger than the loop cycle time for a good PID control to be achieved. With FeSCADA the user can use PIDs to control processes with a time constant of 1 second or more. For that the communication speed with the remote IOs has to be as fast as (or faster than) the loop cycle time.

The majority of the physical processes in industry have time constants bigger than 1 second: ovens heating, pressure, flow control, and levels in big tanks, soil humidity, buildings heating or cooling, chemical reactions or concentration, etc.

Attention !!

Any tool can be dangerous if used in a wrong way. The first example is the knife. FeSCADA is a tool and using PIDs in FeSCADA software can create some challenges. If the communication with the remote IOs is lost because of any reason (power outage, software crash or cable disconnect) the feedback loop is broken and the control is lost. In these cases the system can runaway to dangerous limits of temperature, pressure, overflow, etc. The user has to think a failsafe scenario if danger can arise. The best engineering practice is to use remote IOs that will reset to zero the outputs when the communication is lost. In the same time a local redundant safety system can be put in place to catch any dangerous runaway situations that can be generated from a communication lost, or by a poor PID tuning in a normal run.

2) Basics of Closed Loop Control (PID control)

An industrial process control involves the measuring of a physical analog property like: temperature [^{o}C], pressure [PSI], tank level [in], flow rate [l/min], or position [mm]. This is called the *process variable* (**PV**). The sensors for these properties will always give back an analog value. To control a physical continuous process the controller is using an actuator: heater, pump, servo-motor. The output command to the actuator can be analog (0…10 VDC; from fully closed for 0 V to fully open for 10V), or digital (on, off). The output command is called *controller output* (**CO**). The process is controlled if the *process variable* (**PV**) is changed by the *controller output* (**CO**) to reach a desired target value called *set point* (**SP**) value.

At any moment of the process control the feedback from the sensors is updated and the output is corrected based on the difference between the *set point* value and the *process variable* (**SP-PV**). This difference is called the control error (**e**). Because of the permanent feedback monitoring, this type of control is called a *closed loop control*.

A PID controller is an electronic device or a software program used in a closed loop control. The basic idea behind a PID controller is to read a sensor, compute the error (**e=SP-PV)**, or the reverse error **(e=PV-SP)**, and from here the desired actuator output (**CO**), by calculating proportional (**P**), integral (**I**), and derivative (**D**) responses of the error, and summing those three components to compute the output.

The following general formula is used to compute the CO for a PID controller:

$$CO(t) = K_p \cdot e(t) + K_i \cdot \int_{t_0}^{t} e(\tau) \cdot d\tau \; + K_d \cdot \frac{de(t)}{dt} + K_{FF} \qquad (10.1)$$

where ***t*** is the actual time from the start time t_0, **e** is the deviation error, $\mathbf{K_p}$, $\mathbf{K_i}$, $\mathbf{K_d}$ are the proportional, integral, and derivative coefficients (parameters) of the PID controller, and $\mathbf{K_{FF}}$ is the feed forward term, usually a constant value.

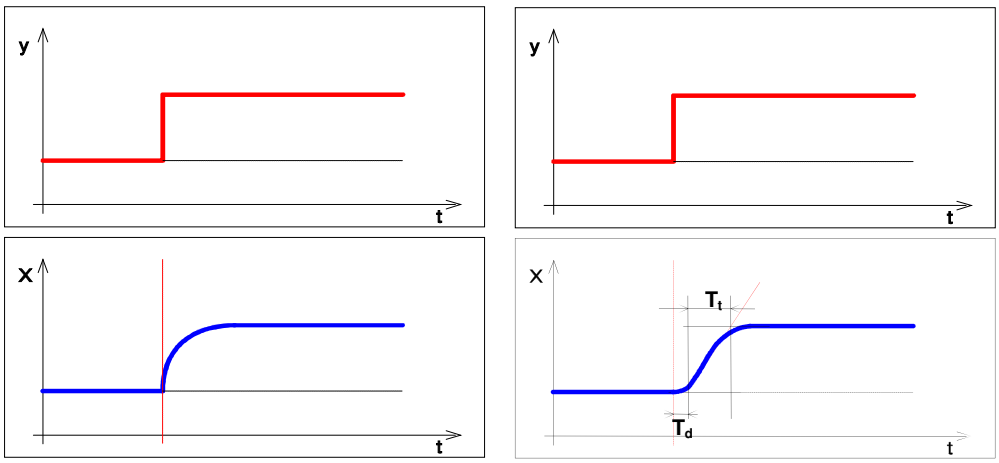
The equation (10.1) is the most general description of a PID controller. A second form used in industry is the equation:

$$CO(t) = K_p \cdot \left(e(t) + \frac{1}{T_i} \cdot \int_{t_0}^{t} e(\tau) \cdot d\tau + T_d \cdot \frac{de(t)}{dt} \right) + K_{FF} \qquad (10.2)$$

Where $\mathbf{T_i}$ and $\mathbf{T_d}$ have some intuitive significance as PID parameters. The equivalence between the 2 equations is as follows:

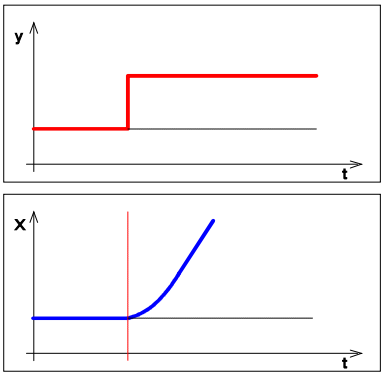
$$K_i = \frac{K_p}{T_i} \qquad K_d = K_p \cdot T_d$$

The equation (10.1) is preferred in many computer PID controller implementations because it is easy to obtain other controllers of type **P**, **I**, **PI**, or **PD** just by choosing a value of 0 (zero) for some of the coefficients $\mathbf{K_p}$, $\mathbf{K_i}$, or $\mathbf{K_d}$.



Typical responses (X) for a step output excitation (Y) for systems with *integral compensation*.
Examples: Temperature, pressure and flow control, positioning control against a spring.

There is another family of controllers related with the PID. It is called PDD^2. This type of controller is often used for motion control and especially for pneumatics and hydraulics where the system does not have *integral compensation*. In these cases, if the valve is opening in one direction, the piston will keep moving to the right (or left) until the end of the cylinder or until the valve will close, see figure in the right. The equation of such a controller is:

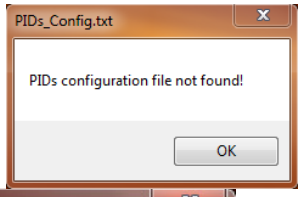


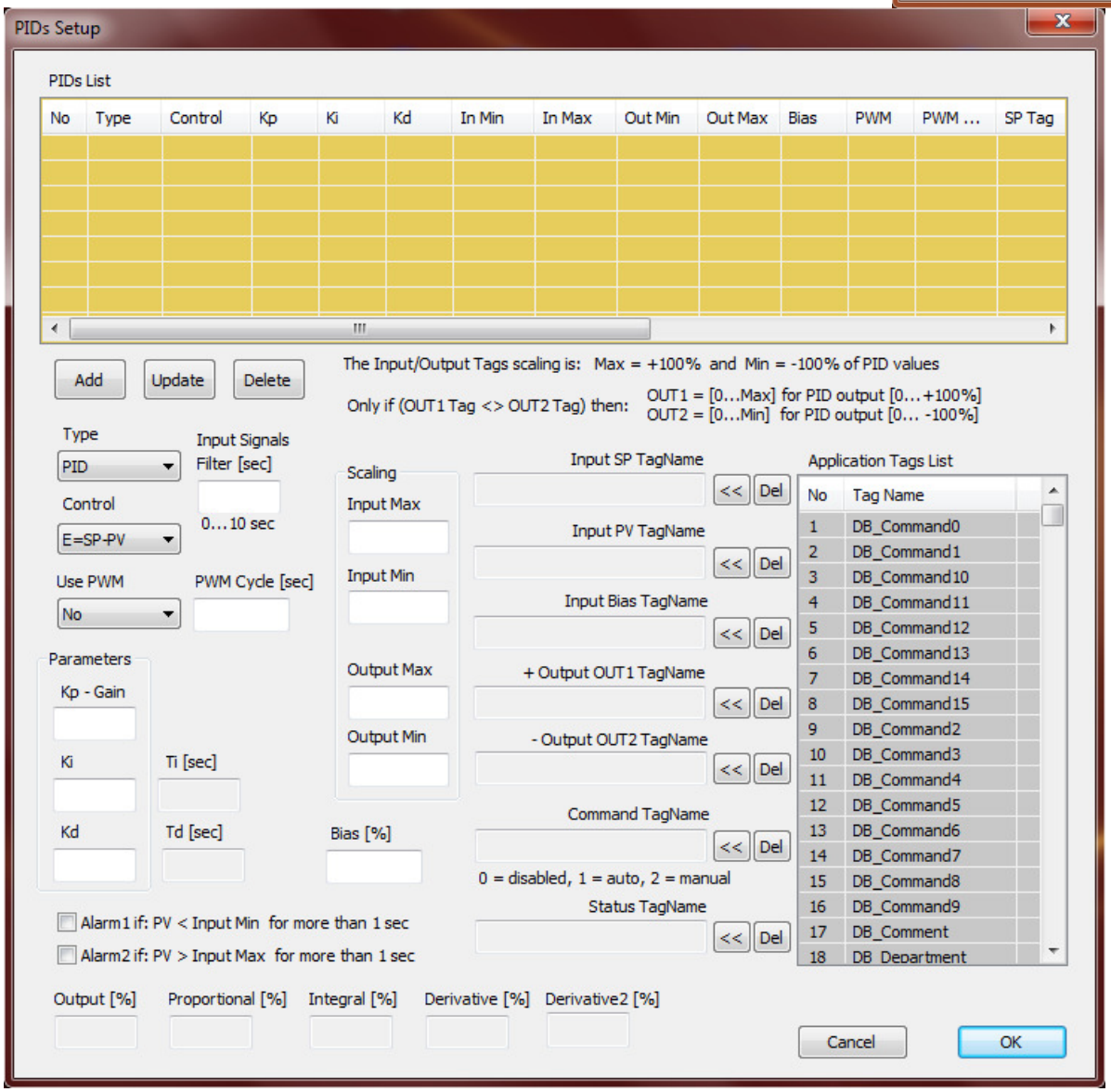
$$CO(t) = K_p \cdot e(t) + K_d \cdot \frac{de(t)}{dt} + K_{d2} \cdot \frac{d^2e(t)}{dt^2} + K_{FF} \qquad (10.3)$$

The first derivative is limiting the speed of correction. The second derivative is limiting the acceleration and deceleration of the move.

3) Settings of a PID

To open the **PIDs Setup** dialog window, select from the menu **Utils->PIDs**. A notification message will inform if no data was previously configured. Press **OK** to open the dialog window. The **PIDs List** is on the top of the window. The **Application Tags List** is in the bottom right corner.



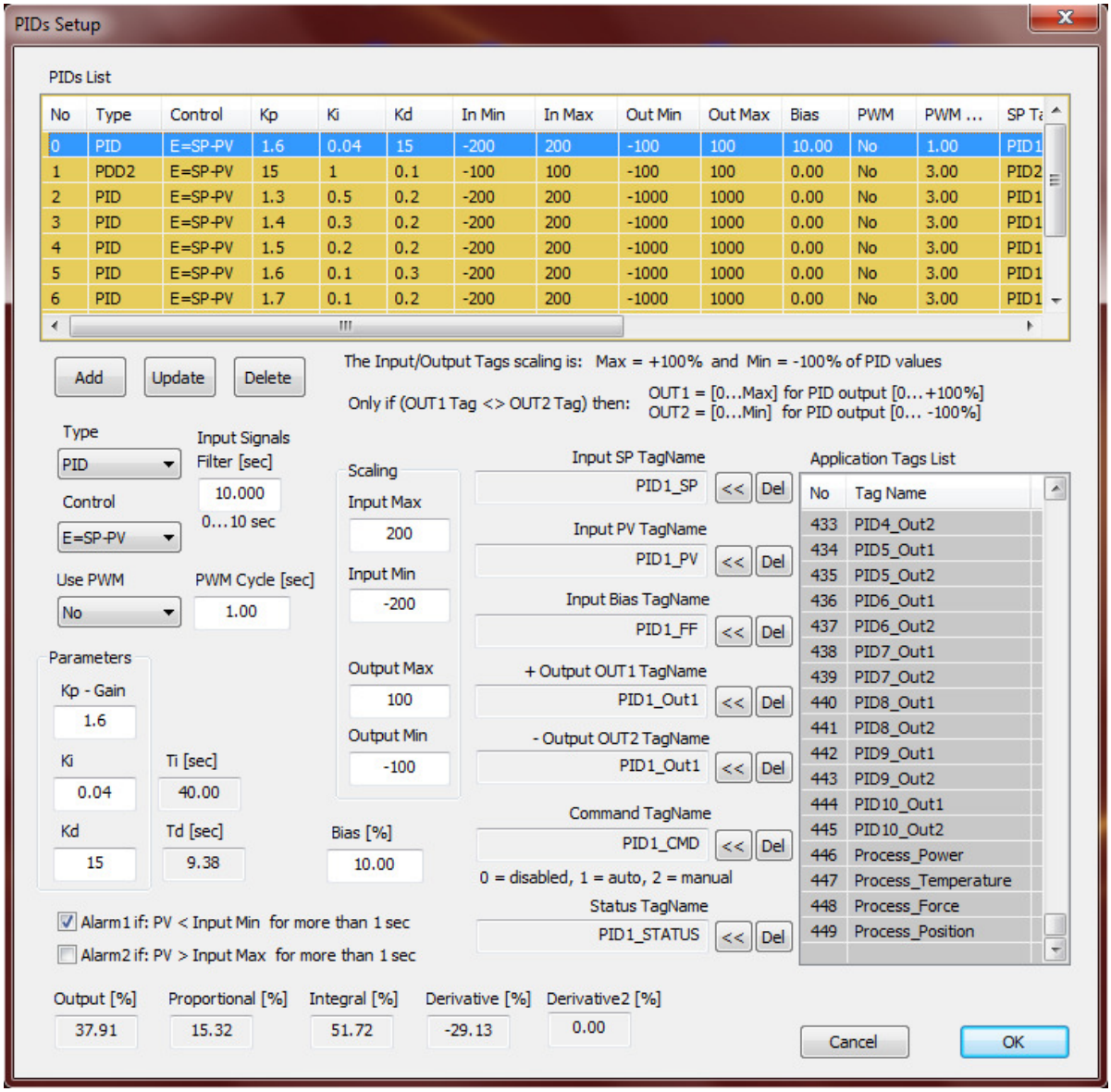


To setup a PID first select the **Type** of the controller: *PID* or *PDD*^{2} and the **Control** way: direct *E=SP-PV*, or reverse *E=PV-SP*. If the process variable signal has noise you can filter the signal by setting a value in **Input Signals Filter [sec]**. If digital outputs are used (on/off control) select to use pulse width modulation **Use PWM** = *Yes* and set the PWM cycle time. Next set the PID parameters **K_p**, **K_i**, and **K_d**.

From the **Application Tags List** select with the buttons **<<** the tags for: Inputs **SP**, **PV** and **bias** (feed forward), and for the outputs **OUT1** and **OUT2**. Also select the tags for **Command** and **Status** of the PID controller. For the inputs and the outputs the scaling limits has to be defined in **Input Max**, **Input Min**,

Output Max and **Output Min** text boxes. A constant value for the PID bias (feed forward) can be setup in **Bias %** text box. This value will be used only if the **Input Bias TagName** is not defined (no tag selection).

After completing all the setting press **Add** button under the **PIDs List**. You can select a PID from the list, change settings and press **Update** button to save them in the list. You can delete an entry in the list with **Delete** button. To save the list in a configuration file press **OK** when you want to leave the setup dialog window. In the picture below it is an example of a PID setup.



The PID algorithm is not limiting the range of the input values SP and PV, but will always limit the output in the range -100…100. This output value can be scaled to the desired engineering value with the **Output Max** and **Output Min** parameters.

Alarm1 and **Alarm2**, if selected, will deactivate the PID and reset the output in case the conditions are true. These safety measures are checking the value of the process sensor (e.g., a lot of sensors give the output in the range 4…20 mA; when the sensor is faulted or it is not connected the input will be 0 mA).

4) PID Inputs and Outputs – Analog and PWM

Scaling the inputs and the outputs of the tags used in a PID is necessary because there can be a variety of ranges for both the inputs as well as for the outputs. The PID algorithm is designed to work with the normalized inputs and outputs, in the range [-100.0 … +100.0].

Examples of input scaling

Physical variable	**Electrical value (analog input)**	**Digital tag value**	**Scaled tag value**	**PID value: -100 …100**
-200 … 300 °C	0 … 20 mA	0…1024	-200 … 300	Input Max = 300 Input Min = -200
0 … 50 PSI	0 … 10 VDC	0…4096	0 … 50	Input Max = 50 Input Min = 0

There are no limitations for the ranges of all three inputs: SP, PV and Bias. But all of them will have the same input scaling: Input Max -> +100.0, and Input Min -> -100.0.

The PID algorithm will limit the PID output value to -100.0 … 100.0. It includes methods to eliminate the windup phenomenon. At the transfer from the PID output to the tag output the output scaling is applied. There can be different options to use the PID output. One can use 1(one) output or 2(two) outputs (heating and cooling). Analog outputs or digital (on/off).

Examples of output scaling

PID output value: -100 …100	**Scaled tag value**	**Tag value (to RTU)**	**Electrical value (analog output)**	**Physical variable**
Output Max = 300 Output Min = 0	0 … 300	0…1024	0…20 mA	0…80 Hz AC motor, VFD speed
Output Max = 50 Output Min = -50	-50 … 50	0…4096	-10…10 VDC	-100% … +100% left to right opening of a proportional valve

The following options can be chosen for PID output.

		Option 1	Option 2	Option 3	Option 4
PID output: -100 …100	**Scaled PID output**	**1 Tag used**	**1 Tag used**	**1 Tag used**	**2 Tags used**
Positive: 0 … +100	X… Output Max	+OUT1 **TagName1** X = 0	---	+OUT1 **TagName1** X= (Max + Min)/2	+OUT1 **TagName1** X = 0
Negative: 0 … -100	X… Output Min	---	-OUT2 **TagName2** X = 0	-OUT2 **TagName1** X= (Max + Min)/2	-OUT2 **TagName2** X = 0

For **Option 3**, if both *Output Min* and *Output Max* are either positive, or both negative, *TagName1* will take the values:

+100	Output Max
0	(Output Max + Output Min)/2
-100	Output Min

For **Option 4**, even if both *Output Min* and *Output Max* are either positive or negative:

0 … +100	0… Output Max	TagName1
0 … -100	0… Output Min	TagName2

The digital tags (on/off) can be used with the PWM option in the same way as the analog tags. The difference is that the PID output, -100%…100%, will be applied directly to the PWM cycle (no scaling).

5) Open Loop, Closed Loop and bumpless start

Open loop control means that the PID is not active but it is in manual mode. In this mode the Bias tag input value (Feed Forward) is manually transferred to the output of the PID controller. To set the PID in manual control write 2(two) in *Command* tag. The status tag will show 2 as acknowledge that the PID is in *Manual mode*.

Closed loop control is active when the *Command* tag has the value 1(one). The Status tag will acknowledge with the same value. In closed loop control the PID is in *Automatic mode*.

If the *Command* has the value 0 the PID is deactivated and the output is set to 0(zero).

The *bumpless start* happens when the user is putting the PID in manual mode (*Command* = 2) and after that is going in automatic mode (*Command* = 1). In a bumpless start the PID algorithm is computing the integral term so that it will counteract the proportional term and the sum of proportional and integral terms will be equal with the manual existing value (Feed Forward value), before the transition.

When switching from automatic mode (*Command* = 1) to manual mode (*Command* = 2), the bias (feed forward) term will be changed to have the value of the CO (controller output), before the transition, and the integral term is reset to zero. Because the bias (feed forward) term is always scaled as an input, with Input Min and Input Max values, the change of the bias term will use a reverse scaling (from -100…100, the value of CO, to Input Min…Input Max).

If the user is transiting from *Command* = 0 to *Command* = 1 the PID will reset the integral term to 0(zero) and it will start the algorithm iterations from there. If the proportional term is big, because the deviation error is big, the controller output will jump significantly.

If the *Command* = 3 the PID will start working in a special mode: $\mathbf{K_p}$ = *Scaled* ($\mathbf{K_{FF}}$), $\mathbf{K_i}$ = 0 and $\mathbf{K_d}$ = 0. The gain can be changed by the user simply by changing the **Input Bias** (feed forward) tag value. The PID will work only as a **P** controller. This mode is useful in testing the controlled system for frequency response tuning. The **Bias** tag value is scaled with the **Input Max** and **Input Min** coefficients. To compute the real $\mathbf{K_p}$ value the scaling has to be applied to the **Bias** tag value.

$$K_p = -100 + \frac{100 - (-100)}{Input\ Max\ -\ Input\ Min} \cdot (K_{FF} - Input\ Min)$$

To bypass this hurdle one can choose the parameters **Input Max** = 100 and **Input Min** = -100. In this case $\mathbf{K_p} = \mathbf{K_{FF}}$.

The table below is showing all the possibilities for the *Command* and *Status* tags values.

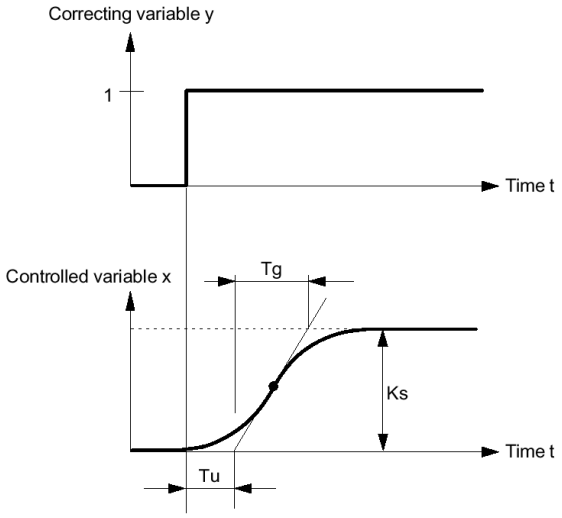
Command	Status	Observations
<0 or >3	-1	Command error
0	0	Deactivated – PID output is set to 0(zero) and the PID is turned off
1	1	Automatic mode
2	2	Manual mode
3	3	Automatic mode as a P controller. The gain is taken from Bias (K_{FF} input)
1 or 3 + Alarm1	-2	Deactivated – PID output is set to 0(zero) and the PID is turned off
1 or 3 + Alarm2	-3	

6) PID parameters tuning

Finding the right PID parameters (K_p, K_i and K_d) is an important technical ability because the system's closed loop performances depend on these values. Every controlled system has its own time constants, which depend on the driven power, inertia, exterior perturbations, etc. The performances of a closed loop controlled system are: fast response, no oscillations around the set point value, small deviation in a steady state situation. The performances of such a controlled system are determined by the PID coefficients. There are a few empirical methods to estimate the good PIDs parameters:

a) ***Manual adjustment*** - Adjust the coefficients little by little, from small values to greater values. After each modification check the system behavior and try to find out if the performances are better. If the system behaves better, continue with the increase of the coefficients. If the system runs worst then decrease the coefficients to the previous values. Usually the K_p coefficient must be increased first, until the system reaches the best performances. After that, increase the K_i and finally the K_d coefficient as long as the system performances became better.

b) ***Step input response curve*** – In this method the user should have the manual control over the system. Set the PID output at a constant value and record the process value response in time. Find the values for time delay (T_u), rise time (T_g) and system gain (K_s). T_u and T_g are time values, in seconds. K_s is the system gain and is measured as the rate of the PID input signal (in percent, after scaling) over the step of the PID output signal (in percent, before scaling) when the system is in steady state. For example, if the output is 50% and the input from the sensor is 75% then the system gain is Ks = 75/50 = 1.5.



c) ***Frequency response*** – In this method the user should have the automatic control over the system. Set the PID in a closed loop control only with the P control. Increase the gain from zero until steady-state oscillations arise. Record the system's response in time. Find the values for critical oscillating period (T_c) and critical gain (K_c). The time is in seconds. K_c is the PID controller's gain when the oscillations started to occur.

Use the values above and the tables below to estimate the PID parameters. After the first evaluation the user can fine tune the PID by applying small parameters changes.

Step response: Ziegler – Nichols

	K_p	T_i	T_d	K_i	K_d
P	$1 \cdot \frac{T_g}{T_u} \cdot \frac{1}{K_s}$				
PI	$0.9 \cdot \frac{T_g}{T_u} \cdot \frac{1}{K_s}$	$3.33 \cdot T_u$		$\frac{K_p}{T_i}$	
PID	$1.2 \cdot \frac{T_g}{T_u} \cdot \frac{1}{K_s}$	$2.0 \cdot T_u$	$0.5 \cdot T_u$	$\frac{K_p}{T_i}$	$K_p \cdot T_d$

Step response: Chien, Hrones and Reswick

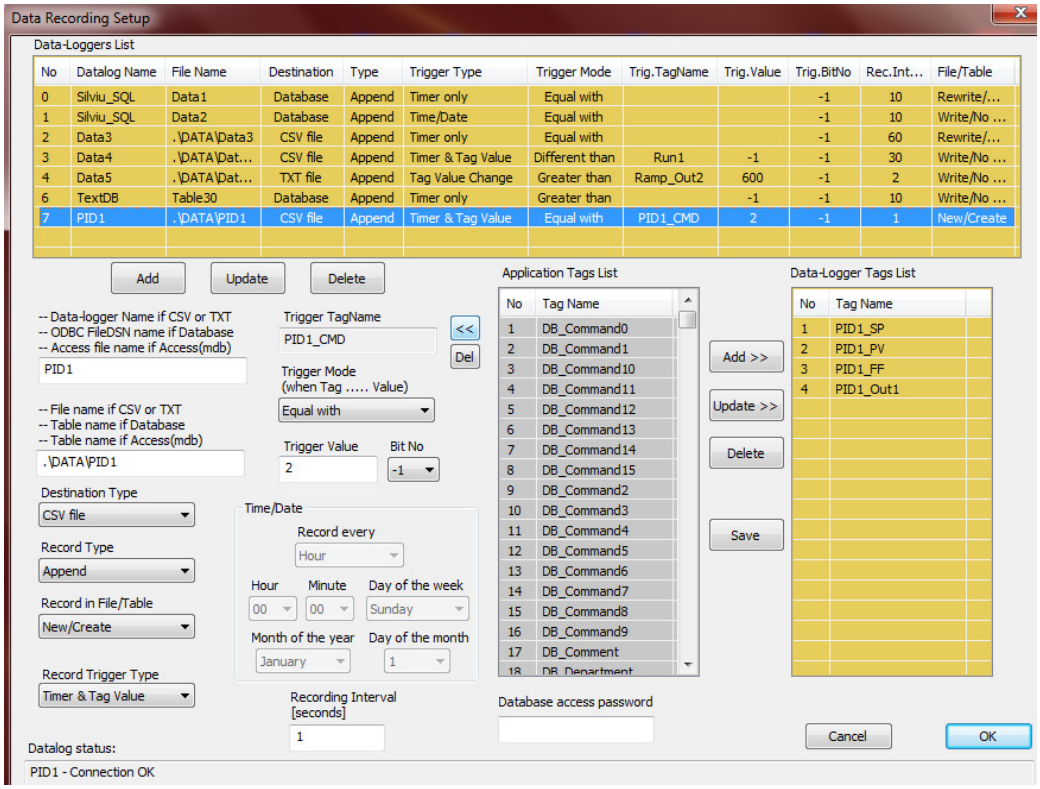
	K_p	T_i	T_d	K_i	K_d
P	$0.3 \cdot \frac{T_g}{T_u} \cdot \frac{1}{K_s}$				
PI	$0.6 \cdot \frac{T_g}{T_u} \cdot \frac{1}{K_s}$	$4 \cdot T_u$		$\frac{K_p}{T_i}$	
PID	$0.95 \cdot \frac{T_g}{T_u} \cdot \frac{1}{K_s}$	$2.4 \cdot T_u$	$0.42 \cdot T_u$	$\frac{K_p}{T_i}$	$K_p \cdot T_d$

Frequency response: Ziegler – Nichols

	K_p	T_i	T_d	K_i	K_d
P	$0.5 \cdot K_c$				
PI	$0.45 \cdot K_c$	$\frac{T_c}{1.2}$		$\frac{K_p}{T_i}$	
PID	$0.6 \cdot K_c$	$\frac{T_c}{2}$	$\frac{T_c}{8}$	$\frac{K_p}{T_i}$	$K_p \cdot T_d$

FeSCADA can be used to setup a step response test, or a frequency response test. The user can record tags with *Data Logger* utility. The data can be imported in Excel spreadsheets and analyzed.

Hereafter is an example of using the step response method. An oven with heat losses is heating a mass of 1 liter of water with a maximum power of 5000 W. For *PID1* we setup a data logging that will record, every second, the values of PV and CO, if the tag *PID1_Cmd* = 2 (manual mode). See picture on the right.



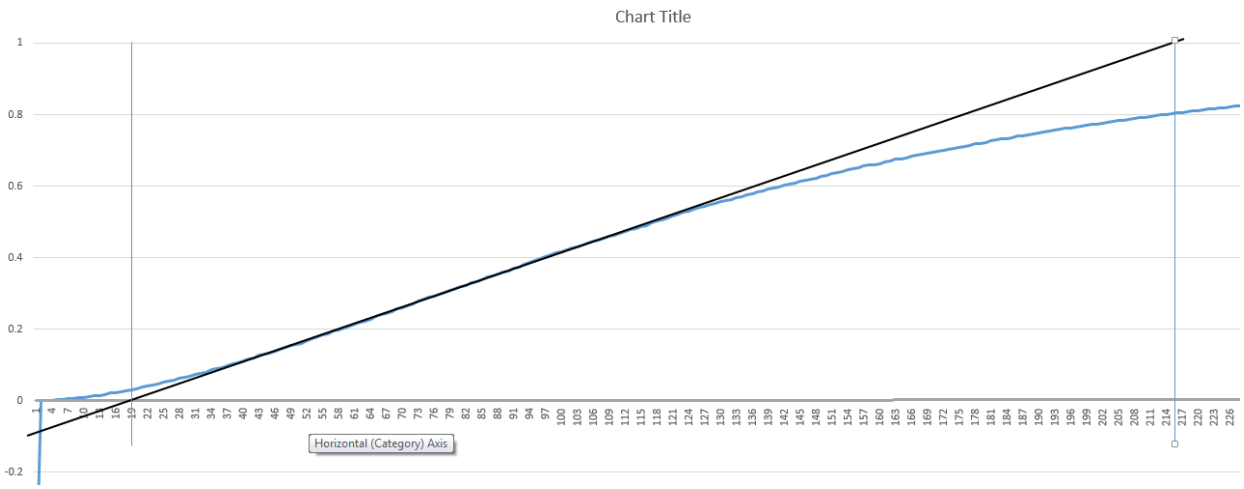
In a screen we monitor the step response with a trend graph. When the temperature is stable enough the test can be stopped.

The data imported in Excel gave the chart from below. The graph was normalized by dividing all the data with the maximum value of 82.88 ^{o}C.

The delay time is
$\mathbf{T_u}$ = 19 seconds.

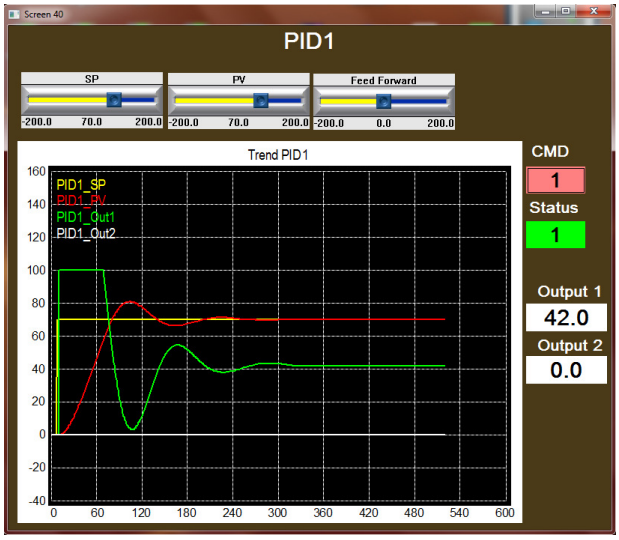
The rise time is
$\mathbf{T_g}$ = 215-19 = 196 seconds.

The system gain is
$\mathbf{K_s}$ = scaled(82.88) / 50 = 41.44 / 50 = 0.828.

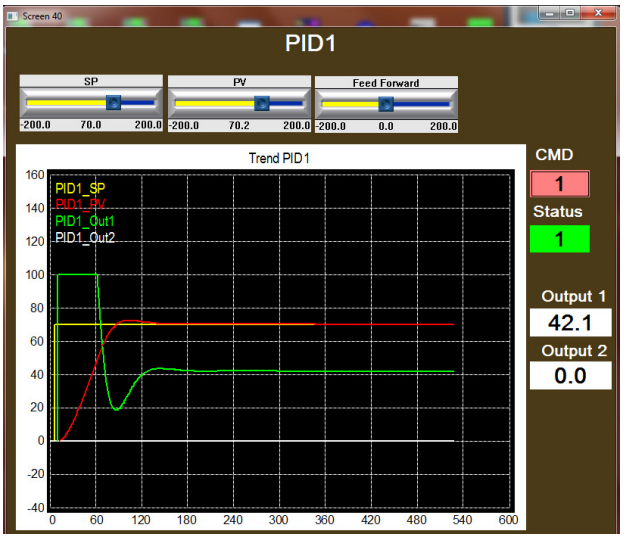


With these values, applying the formulas from "Chien, Hrones and Reswick" table for step response, we get the parameters from the table to the right. Running the PID we get the responses from the pictures below.

	Kp	Ti	Td	Ki	Kd
P	3.73				
PI	7.47	76.00		0.098	
PID	11.83	45.60	7.98	0.26	94.4



PI Control



PID Control

7) Application examples

There are many possible applications that can be done using the **PIDs** utility of FeSCADA software. The temperature control is probably the easiest to accomplish because there are many temperature sensors (Thermocouples, Thermistors, RTDs, etc.) and the output can be a simple relay, or solid state relay (SSR).

Another family of applications covers the controlling of AC motors speeds with variable frequency drives (VFDs). The VFD can take the speed reference from an analog input, old style, or directly from a message

in a communication network (Modbus TCP/IP, Ethernet IP, etc.). The AC motors are used extensively in industry for pumps, which are the actuators to regulate level, flow and pressure.

A third area of applications is related to pneumatic and hydraulic proportional valves. They are widely used to control pressure, force, speed, position. The proportional valves are controlled with analog inputs: 0…10 VDC, - 10…10 VDC, 0…20 mA, or 4…20 mA.

Apart of these classical industrial applications there are many others that can make use of a PID controller.

8) Conclusions

With FeSCADA software it is possible to setup and run PID controllers. Tags are assigned and used for *Command* and *Status* of a PID, for PID inputs and outputs. Scaling is applied for input and output tags. The output of the PID can be used with one or two tags, analog or digital. Digital tags use pulse width modulation (PWM). The PIDs can be run in manual and automatic mode. Two empirical methods for PID tuning are available in FeSCADA: the step response and the frequency response.