Position & Temperature Control in Batch Processing

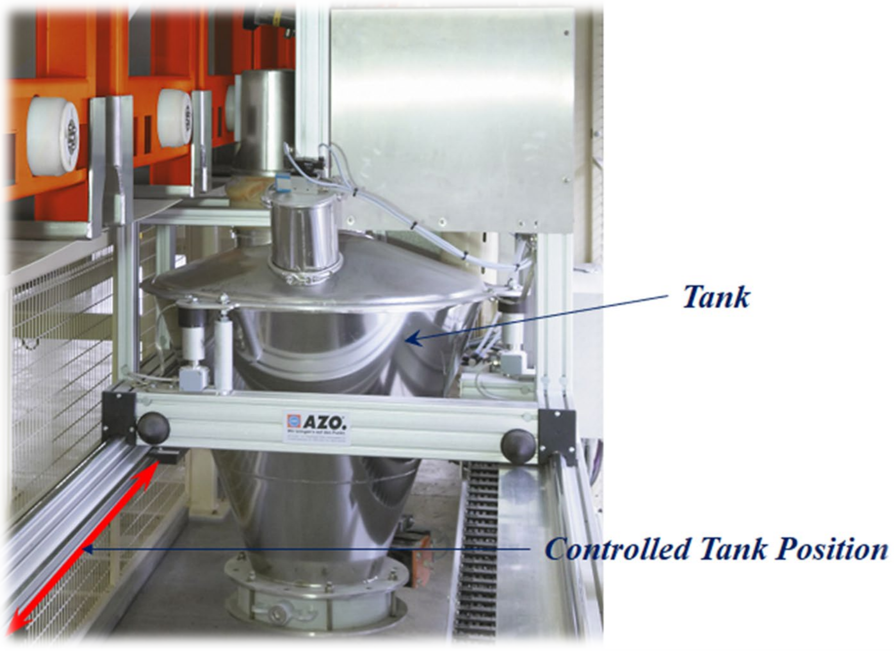


Figure 1 Controlled position tank example.

Figure 1 and figure 2 shows a process system in which two liquids (or chemicals), A and B are mixed together in a moveable tank. S1, S2, S3 are level switches of the NC (normally closed) type that change state when the fluid reaches the level in question. It gives a logical high signal (or TRUE) when it is NOT activated. The system also has (not shown in figure 2) a starting button, **StartP**, a stopping button, **StopP** and there is also a temperature sensor, **Output_Heater,** and a level sensor **TankLevelSensor**.

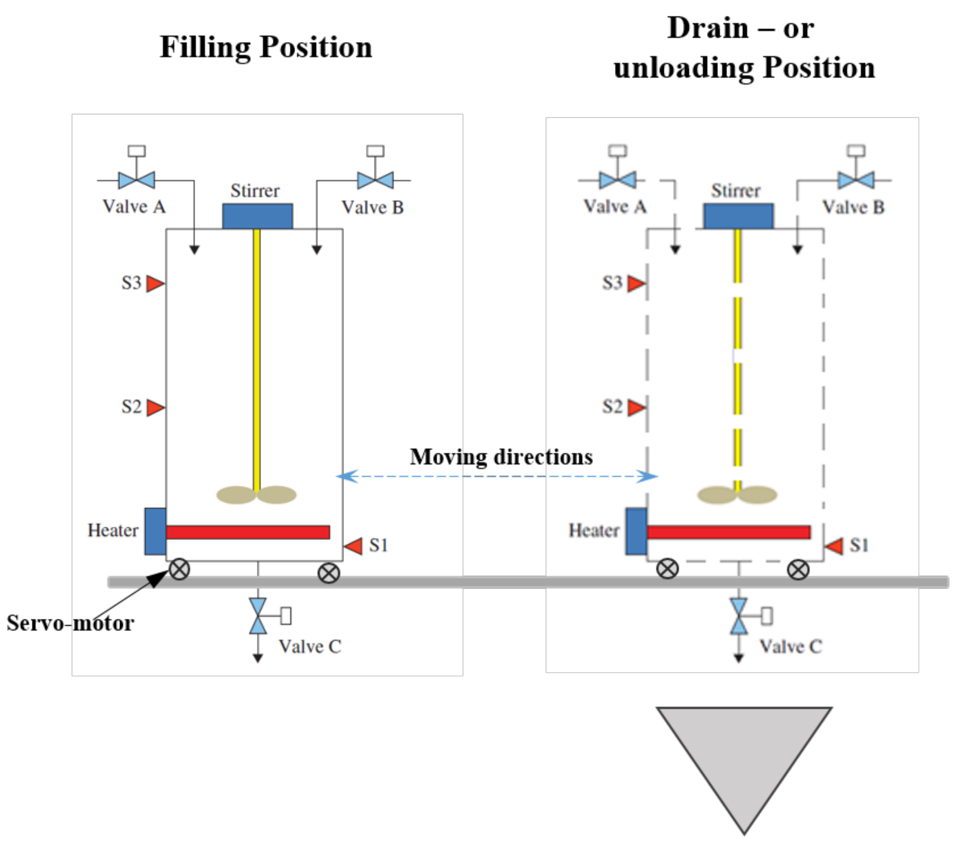


figure 2 Tank with several Sensors and Actuators

Functional description 1: the system is to function as follows: (one sequence)

Assuming that the tank is empty and it is located at the filling position (setpoint = 0) the liquid temperatures are 0 ºC. A batch can be started.
The temperature setpoint for the heater will be set at 75 ºC for this sequence.

- At the signal to start (using **StartP**), valve **A** opens.
- At level **S2**, valve **A** closes, the heating element is turned on, the stirrer(agitator) starts, and valve **B** opens.
- At level **S3**, valve **B** closes. Please note: **S1**, **S2** and **S3** are digital sensors!
- When the temperature of 75 $^\circ$C $\pm$ 2% is reached, a timer is activated and after 20 seconds the mixture will become homogeneous. The tank can now be transported to the unloading position (Setpoint = 30) by the servomotor. When the position of tank, "30 ± 0.6" is reached, at the settling time, valve **C** will open.
- The heater will be turned off.
- The stirrer will stop if the level is below **S2**.
- Below level **S1**, valve **C** closes and the tank will be moved back to the start position (setpoint = 0) and this will be the end of one sequence.

- The sequence will be repeated until the amount of sequences is reached (a batch of 2 sequences) or if the button **StopP** is pressed.
- At the end of each sequence the time duration of that sequence will be registered. When the amount of sequences is reached (batch of 2 sequences) the average time of a sequence will be calculated and registered (in seconds) for further research.
- The level of each liquid (A and B) should be registered during each sequence and these values will be obtained using the **TankLevelSensor** values.
-

Task 1(Normal mode)

Develop (draw) a state diagram of the whole sequence as described as the **Functional description 1** and develop a PLC application using ST (Structured Text) according your state diagram. Create also a simple but acceptable user interface showing all the information needed e.g. states: "heating", "stirring", position, temperature, time values etc. Show your well documented results in a pdf file and comment or draw your **conclusions.****

In order to test your application the project `"Pos_and_Temp_Controller.tnzip"` is available and this project includes a model that simulates the tank with the heater which is a 2nd order process. The "**Heater**" temperature is measured with the "**Output_Heater**" sensor (the process value of the system). The POU name is: "HEATER_PROCESS.PRG".

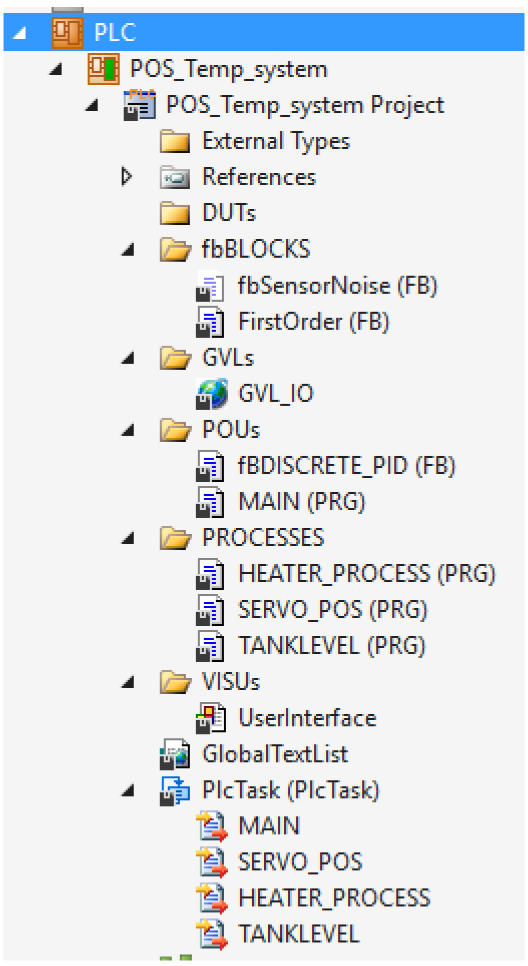
A P-controller for the Heater is already implemented and it is "switched off" by now. (open loop) `"Auto_Mode"` parameter is **FALSE**, see fbDISCRETE_PID.FB, **(Explain why**).**

"TANKLEVEL.PRG" is a first order model that simulates the liquid level and can be controlled only by the valves A, B and C and the states of the level sensors Level_**S1**, Level_**S2** and Level_**S3** sensors!

Also included is a first order process model with an integrator of a servo motor for positioning `"SERVO_POS"` and it is currently controlled by a P-Controller. The `"Auto_mode"` parameter is **TRUE,** see fbDISCRETE_PID.FB **(Explain why**)**

These POU's are located in the Map `"PROCESS"` . These files in this MAP should not be modified without notice! See figure 3.

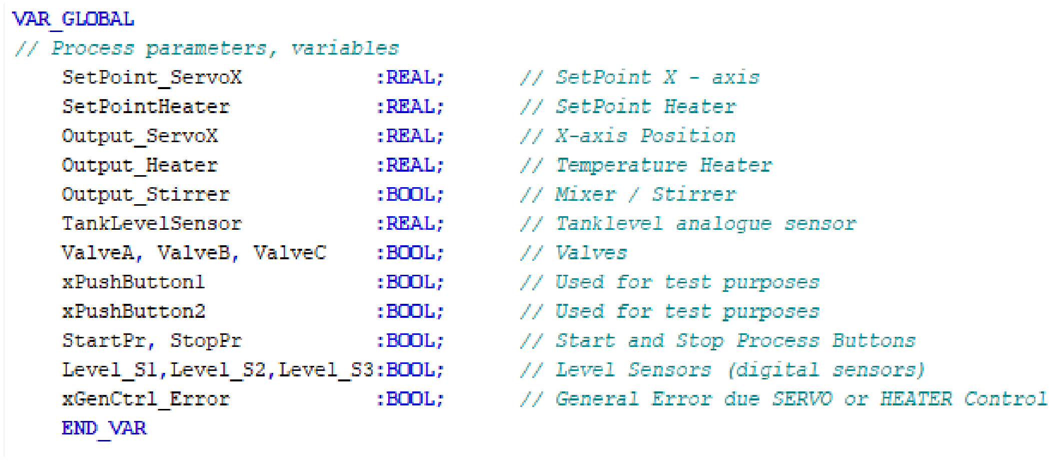
In this task the PID controllers parameters will remain unchanged!



```
PLC
  POS_Temp_system
    POS_Temp_system Project
      External Types
      References
      DUTs
      fbBLOCKS
        fbSensorNoise (FB)
        FirstOrder (FB)
      GVLs
        GVL_IO
      POUs
        fBDISCRETE_PID (FB)
        MAIN (PRG)
      PROCESSES
        HEATER_PROCESS (PRG)
        SERVO_POS (PRG)
        TANKLEVEL (PRG)
      VISUs
        UserInterface
      GlobalTextList
      PlcTask (PlcTask)
        MAIN
        SERVO_POS
        HEATER_PROCESS
        TANKLEVEL
```

figure 3 Overview Solution Explorer (a part)

The GLOBAL variables that represents the sensors and the actuators are listed here also , see figure 4. These variables should be used in your application which will be mainly developed in the “MAIN.PRG”



```
VAR_GLOBAL
// Process parameters, variables
    SetPoint_ServoX          :REAL;      // SetPoint X - axis
    SetPointHeater           :REAL;      // SetPoint Heater
    Output_ServoX            :REAL;      // X-axis Position
    Output_Heater            :REAL;      // Temperature Heater
    Output_Stirrer           :BOOL;      // Mixer / Stirrer
    TankLevelSensor          :REAL;      // Tanklevel analogue sensor
    ValveA, ValveB, ValveC   :BOOL;      // Valves
    xPushButton1             :BOOL;      // Used for test purposes
    xPushButton2             :BOOL;      // Used for test purposes
    StartPr, StopPr          :BOOL;      // Start and Stop Process Buttons
    Level_S1,Level_S2,Level_S3:BOOL;     // Level Sensors (digital sensors)
    xGenCtrl_Error           :BOOL;      // General Error due SERVO or HEATER Control
    END_VAR
```

figure 4 GLOBAL variables with "sensors and Actuators"

Task 2 (Improved mode)

As you noticed earlier in Task 1, the Heater is controlled as an open loop system and the Tank position is controlled by a P-Controller. (see the instances of "`fbDISCRETE_PID.FB`" pidServo and pidHeater)
In order speed up the production and using other setpoint values a PID controller is suggested to control the heater system and for the Servo Motor for positioning of the Tank.
Your task is to develop a PID Controller for both systems, the heater system and the servo positioning system. The POU "`fbDISCRETE_PID.FB`" will be the general function block that should be modified.

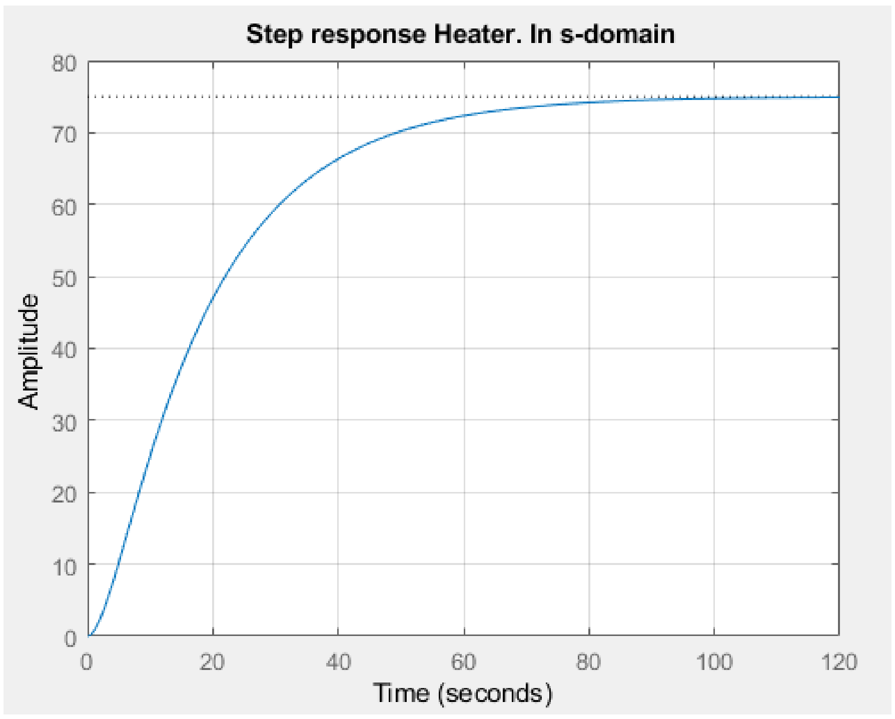
Equation (1.1) shows an example of an ideal PID algorithm.

$$mv(t) = \boldsymbol{K_P} e(t) + \boldsymbol{K_I} \int_0^t e(t)dt + \boldsymbol{K_D} \frac{de(t)}{dt} \qquad (1.1)$$

Your PID controllers should be developed and simulated first using Matlab/Simulink.

With your obtained parameters the PID algorithms will be implemented in Structured Text, without using libraries of TwinCAT 3 IDE.

The transfer function of the Heater system is: $G(s)_{Open} = \dfrac{5}{256s^2 + 100s + 5}$ based on the measurements on the heater system. A step response can be seen also in figure 6.



fiure 5 Step response of Heater system at 75 degrees.

In figure 7 the response of an example of a PID controller for the Heater is shown. An overshoot less than 8 % is required in **Task 2**.

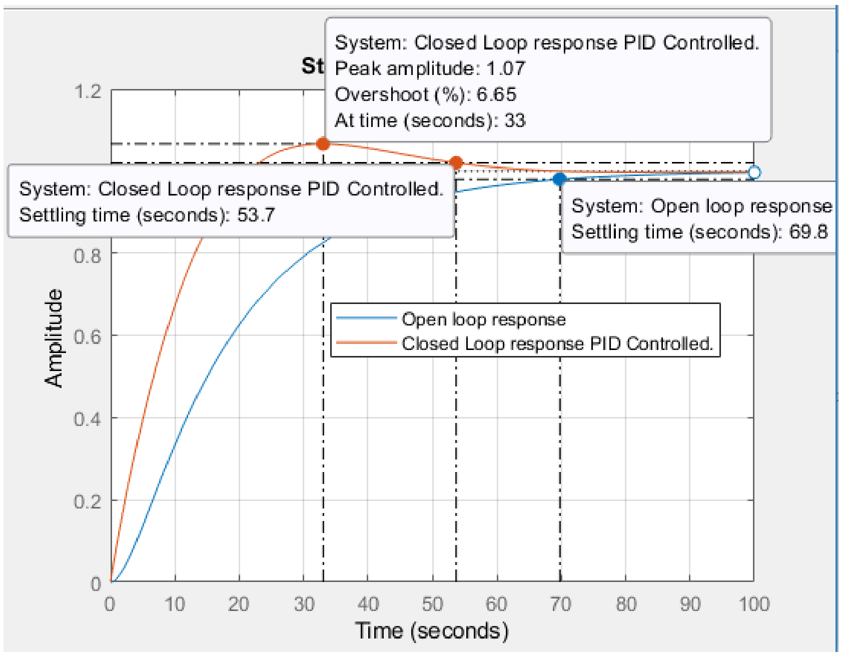


figure 6 Simulated responses on heater system

The transfer function of the Servo_Pos Process is also given: $G_{SERVO} = \dfrac{20}{0.5s^2 + s}$

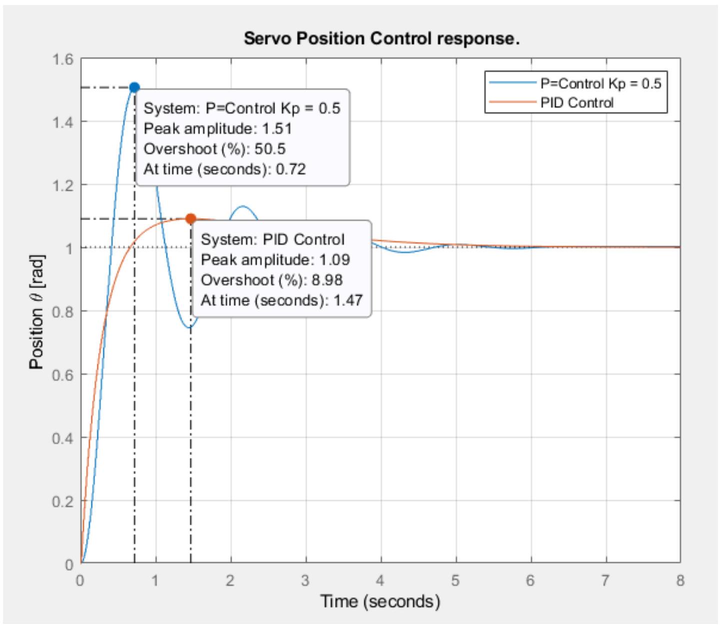


figure 7 Simulated response of Servo system

In figure 8 the responses of the servo system with a P- and with a PID-Controller is shown. An overshoot less than 9% is required in **Task 2**.

- Create a function or a function block in order to measure and visualize the overshoot of both processes to verify the requirements

For the heater process it is not necessary to wait for the settling time of the process output. On the other hand activating valve C on unloading or drain position it is crucial to wait until the desired position is been settled!

Modify your PLC program in a such way that you can run a batch according **Task 1** and **Task 2** by means of a switch selector (Normal/Improved)

Show your simulated Controller in Matlab/Simulink results and compare your results with your implementation. An example of a simulated result using Matlab can be seen also in figure 7 and figure 8.

Show that the production time of a sequence is been improved comparing to **Task 1** after implementing your own PID controllers. Verify also that your PID Controllers also works with other setpoint values!

A demo of your application is mandatory.

This assignment will be graded with extra weight, you may discuss your solutions with other students but you must hand in your own work!

Good luck !

Deliverables:

- A **pdf** report with state diagrams, measurement results, matlab/simulink responses, user interface but also your comments, conclusions and motivations of your choice.
- **tnzip** –file including the PLC project of TwinCAT 3
- **zip** file including all files (e.g. matlab/Simulink, pdf)
- A life demo of your application is mandatory!

The contents of this document may be changed or updated without notice. Make sure you are using an updated version.

Appendices:

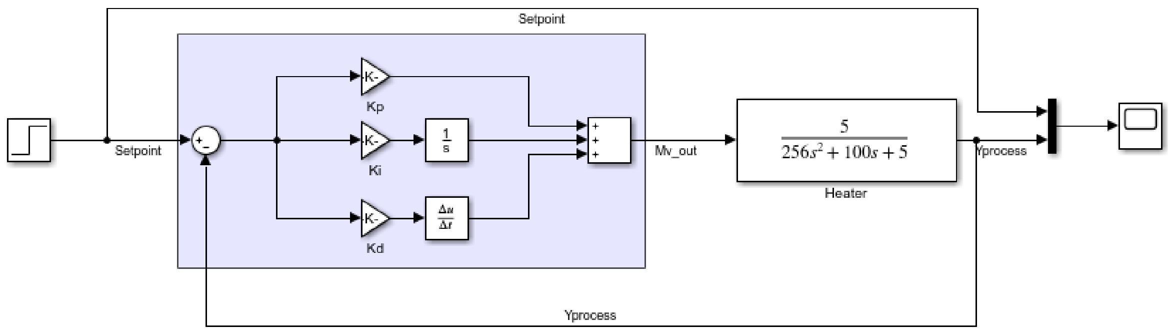


Figure 8 A PID controller example implemented in Simulink