# PID Controller Implemented in Festo CDPX

Primož Podržaj

*Faculty of Mechanical Engineering, University of Ljubljana, Aškerčeva 6, 1000 Ljubljana, Slovenia*

**Abstract.** In this paper, we describe the procedure for the implementation of the PID controller in the Festo CDPX operator unit. These units enable the execution of the control algorithm and human machine interface in a single unit. In our laboratory the unit is used to teach the students about the basics of control systems. For this purpose, one of the most common closed loop control systems for the education purposes was selected. It is a water level control system. In this paper the design of the whole system is presented. The need for a PI control algorithm is also explained. The programming of the operator unit CDPX, both in Festo CoDeSys and Designer Studio is explained. Such a simple system has turned out to be a great educational tool for Control Theory and Programmable Logic Controller related subjects.

## 1 Introduction

Programmable Logical Controllers are a vital part of the industrial environment envisioned within the Industry 4.0 paradigm. This term has been recently widely used to address the current trend of automation and data exchange in industrial and manufacturing technologies. It includes cyber-physical systems, the Internet of things, cloud computing and so on. Sometimes Industry 4.0 is referred to as the fourth industrial revolution. For this reason, more and more manufacturers try to expand their product portfolio into the domain addressed by Industry 4.0. Festo is no exception from that point of view. It is a company founded in 1925 and mainly associated with pneumatic systems' solutions. Gradually they have been also entering the filed of pneumatic and electrical control and drive technology for factory or process automation. Festo CDPX operator units can be seen as part of this effort. They are high-performance processors combined with a widescreen technology. They provide more functions at a higher resolution for the interfaces between man and machine. They operate as PLCs and at the same time as servers by allowing access for external clients both on-site and around the world thanks to the Ethernet interface with integrated switch. They are easily programmed as a PLCs with CoDeSys V2.3 and V3.5 software environment on one side and as HMI using Festo Designer Studio on the other side. As such they are ideal controller tool for the purpose of teaching Control Theory and Programmable Logic Controllers related subjects in the era of Industry 4.0.

When considering different objects to fulfill their role in the close loop control system, our main goal was that the object is inexpensive and easy for comprehension for students. Water level system is a possibility that fits here perfectly. It is also one of the most common (if actually not the most often used) systems in Control Theory related books [1-5]. As it is open-loop stable system, it is easy to visualize, when controller is turned off. In our case the classical PID control algorithm was used. Water level control system is also very good at demonstrating the need for the I component in the PID control algorithm.

In this paper the design of a real water level control system is described. A special care is taken that the system is as close to its linear model as possible. The controller is implemented with the Festo CDPX operator unit using both Festo CoDeSys and Designer Studio.

## 2 System design

The schematic representation of the whole water level control system is shown in Fig. 1. The frame (1) of the system is welded from steel bars. Upper (2) and lower (3) reservoir are used to accumulate water. Two pumps (4) are permanently pumping water from the lower to the upper reservoir. Constant water level in the upper reservoir is maintained by the pipe (5). All the excess water flows through it back to the lower reservoir. The reservoir in which the water level is controlled (9) is shown in Fig. 2. The inflow of water from the upper reservoir is controlled by the valve (6), which is opened/closed by the electric drive (7). The input signal for the electric drive is the output signal of the programmable logic controller (not shown in Fig. 1). The outflow of water is determined by the opening of the tap. As this signal is not under control of the system, it is considered to be a disturbance. It is up to a user to open

the tap as desired. The water level in the reservoir (9) is measured using a float (10).

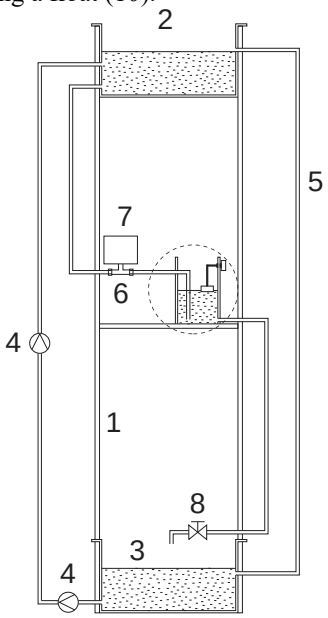


**Figure 1.** Schematic representation of the water level control system (the zoomed in area marked by a dashed circle is shown in Fig. 2).

It is connected to a rotational potentiometer (11) as shown schematically in Fig. 2 and also on the photo in Fig. 3.

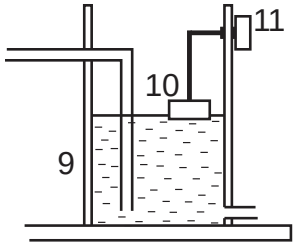


**Figure 2.** Zoomed in area marked by a dashed circle in Fig. 1.

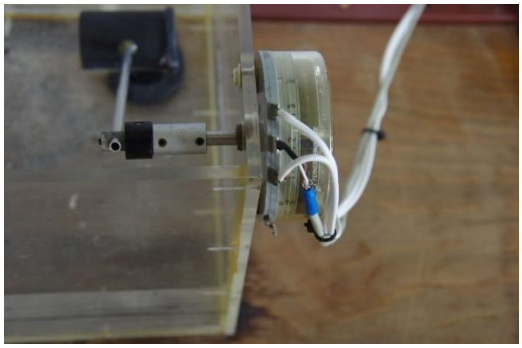

**Figure 3**. The mounting of the float and the rotational potentiometer.

The rotational potentiometer's supply voltage is constant. The output voltage however depends on the position of the float, which of course depends on the level of the water. The output voltage of rotational potentiometer is the input voltage for the programmable logic controller (PLC). The PLC is mounted in a plexiglass holder as shown in Fig. 4.

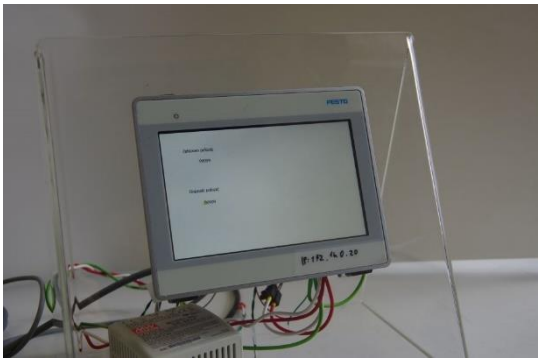

**Figure 4.** PLC (Festo CDPX) in a plexiglass stand.

Festo CDPX functions not only as a PLC but also as a human machine interface (HMI) device. A user can therefore enter the desired water level via a touchscreen. The desired and the actual water level are then compared. In perfect situation these two signals would be the same all the time. Their difference (called the error signal) is the signal on which the control algorithm programmed in the PLC (Fest CDPX in our case) is applied. The resulting signal is the output signal of the PLC. It is also the input signal for the electric drive used to open/close the valve. The electric drive Danfoss AME30 is mounted on the valve as shown on the photo in Fig. 5.

**Figure 5.** The electric drive and valve.

The electric drives output is the movement of small piston, which then presses on the top of the valve (see Fig. 6).

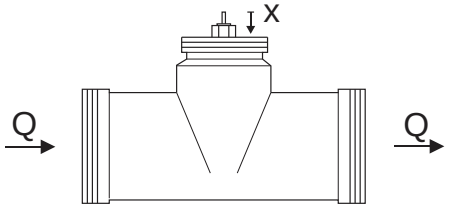


**Figure 6.** The valve with a linear characteristic.

The selected valve has a linear characteristic (the relation between x and Q is linear). This is important if

we want the response of the system to resemble the calculated one as close as possible.

# 3 Theoretical background

## 3.1 Block diagram of the water level control system

In general, there is no agreement about the block diagram of a close-loop control system. Different books use different block diagrams [6-10]. All of them however at least share an object and a controller. For our purpose, the block diagram shown in Fig. 7 seems to be the most appropriate.

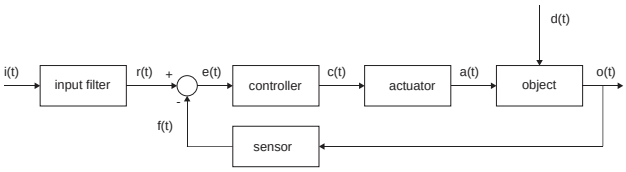


**Figure 7.** Block diagram of a feedback control system

In our case the input filter is actually used to convert the input signal (the desired water level) into the reference signal, which is voltage. This is however done in the Festo CDPX, because we enter the desired level through the touchscreen. The role of the sensor is played by the float and the rotational potentiometer. The feedback signal f(t) is then subtracted from the reference signal r(t). The error signal e(t) is then used in the PID control algorithm:

$$C(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)E(s) \qquad (1)$$

where C(s) and E(s) are the Laplace transforms of the control and error signals. The parameters KP, TI and TD are determined during the parameter tuning process and represent proportional P, integral I and derivative D component of the PID controller (of course not all the components must be present). The role of the actuator block is played by two elements in our case (the electric drive and the valve). The actuating signal a(t) is therefore the inflow of the water. The disturbance d(t) is of course the outflow of the water. The object is the reservoir of water in which the water level is controlled. The output is the actual water level h(t). In order to get to the block diagram (and later the transfer function) of the system, we must get the transfer function of each element. The controller has already been described. The most difficult part is the electric drive. It is itself a control system, and in fact a nonlinear one. Its block diagram is shown in Fig. 8.

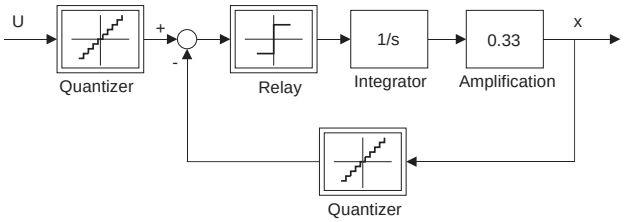


**Figure 8.** Block diagram of the Danfoss AME 30 electric drive.

The electric drive is not supposed to move all the time. In theory it would not move, when desired height is equal to actual height. As there is always noise in the system, it would be moving all the time without quantizers. Their role is such, that the input voltage must vary at least 80mV in order for it to move. When this happens, it needs 3s to move the piston 1mm. The maximum displacement is equal to 12mm, when the input voltage is 10V. If, however the time constant of the water level height response is much larger than the time needed for the piston to reach the desired displacement, the electric drive can be considered a pure proportional element. The valve is proportional element anyway (due to its linear characteristic). The transfer function of the reservoir is given by the following equation:

$$H(s) = \frac{1}{As} \cdot \left(Q_{in}(s) - Q_{out}(s)\right) \qquad (2)$$

The (simplified) block diagram for the presented system can now be designed. It is shown in Fig. 9.

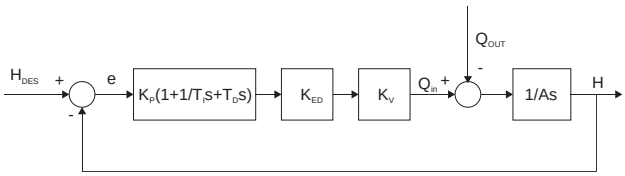


**Figure 9.** Block diagram of the simplified system.

There are some modifications in comparison with the block diagram shown in Fig. 7. First, there is no sensor in Fig. 9. Sensor is of course present, but its output voltage is used by the PLC to calculate the actual height H. This process is not that straightforward, because the relation is nonlinear (the sine of the angle is proportional to the height of the water level). The desired water level is also entered via the touch screen directly into the PLC, so no input filter is needed. The simplified (linearized) version of the electric drive is also used in Fig. 9. In such a case its transfer function is a simple amplification (KED). If the valve has a linear characteristic, it can also be represented by a simple amplification (KV in this case). As the actuating signal and the disturbance are inflow and outflow, the can be the input for the additional summing point in front of the object. There are two reasons for this simplification being made. The first one is that the transfer function of such a system is easily obtained and consequently its output for various inputs can be calculated using the Laplace transform. The second reason is that in this case it is easy to demonstrate what type of controller is needed.

## 3.2 PID type related analysis

The simplest possible controller uses only proportional component (its transfer function is therefore reduced to just KP). It can be demonstrated that such a solution would not be perfect. The proportional controller can

easily be realized by a controller using only mechanical components as shown in Fig. 10.

If for example the system is in a steady state, where the desired water level HDES is equal to the actual water level H, the inflow Qin must also be equal to the outflow $Q_{out}$. Now, let's say that the outflow changes to some new value. As the outflow changes, the inflow should also change accordingly (of course we are only talking about the new steady state). The desired height has not been changed, so we also don't want the actual height to change. The question is however, if this is at all possible. The answer is pretty simple. The inflow can only change if the gate valve is opened/closed. Due to the lever, this can only happen if the water level is changed. This proves that only proportional control will not be sufficient.

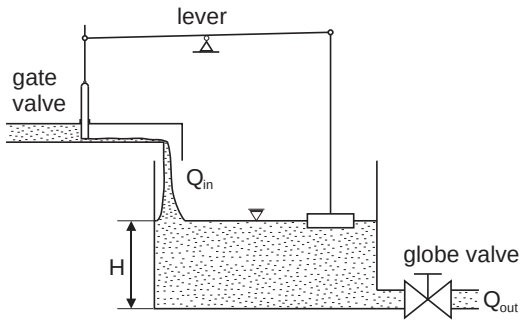


**Figure 10.** Water level control using a mechanical proportional controller.

A more theoretical approach shows the reason. What we are actually looking at is the so called steady state error $e_{SS}$. It is defined in the following way [11]:

$$e_{SS} = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) \qquad (3)$$

where we used the final value theorem of the Laplace transform. In case that the step response for $Q_{OUT}$ is what we are looking for, we can write, the following equality in the case of the P controller being used:

$$e_{SS} = \lim_{s\to 0} s \frac{-\frac{1}{As}}{1+K_P K_{ED} K_V \frac{1}{As}} \frac{1}{s} = \frac{-1}{K_P K_{ED} K_V} \qquad (4)$$

which can never be made equal to zero. In theory, it could be equal to zero by making KP infinitely large, but this is not a practical solution. In case of the system shown in Fig. 10 this would for example mean that the lever support has to exactly over the float. If, however a PI controller is used, we get:

$$e_{SS} = \lim_{s\to 0} s \frac{-\frac{1}{As}}{1+K_P\left(1+\frac{1}{T_I s}\right)K_{ED}K_V\frac{1}{As}} \frac{1}{s} = \lim_{s\to 0} \frac{-T_I s}{T_I s As + K_P(T_I s+1)K_{ED}K_V} = 0 \qquad (5)$$

This gives us a much more convenient way of eliminating the steady state error. The D component was not used in our case. In general, it helps to decrease the overshoot.

# 4 Festo CDPX Programming

## 4.1 Programming in CoDeSys

CoDeSys is an acronym for a controller development system. It is a development environment for programming controller applications according to the international industrial standard IEC 61131-3. Currently the following programming languages are supported: Structured Text (ST), Function Block Diagram (FBD), Ladder Diagram (LD), Instruction List (IL) and Sequential Function Chart (SFC) [12].

When programming the Festo CDPX the first step is to setup the communication between PLC and PC. This is done using the Festo Field Device Tool. The communication is enabled via Ethernet. Many devices can be visible and we just need to select the IP of the CDPX. The next step is the programming in CoDeSys. We open a new project, give it a name and select it as a CDPX project. Several ways of programming languages are available as seen in Fig. 11.

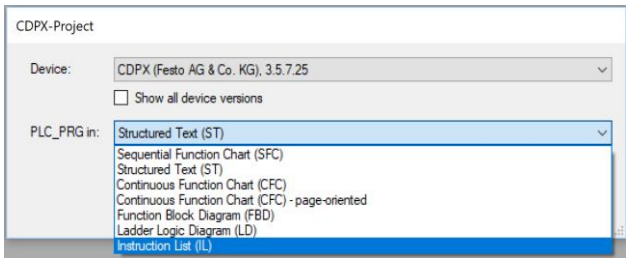


**Figure 11.** Possible programming languages in Festo CoDeSys.

In the next step we add the global variable list (GVL), where all the used variables are declared (see Fig. 12).

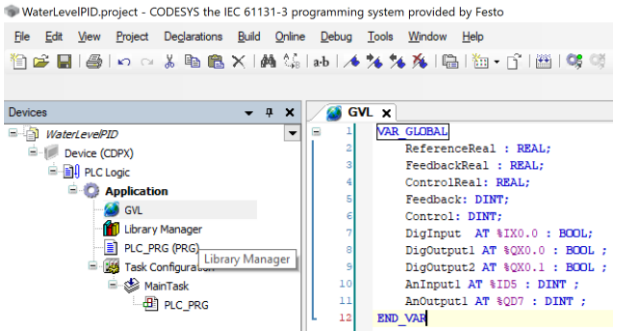


**Figure 12.** Project tree with all the global variables.

The PLC has no problem working with floating point numbers (type REAL), but for analog inputs/outputs the type must be DINT. In order that we can use the feedback and output the control signal the following program in Fig.13 is needed.

```
1  Feedback := AnInput1;
2  FeedbackReal := 1/1000000.0*DINT_TO_REAL(Feedback);
3  Control := REAL_TO_DINT(ControlReal*1000000);
4  AnOutput1 := Control;
5
```

**Figure 13.** The program needed for the PID.

In order to be able to get the PID function a corresponding library must be added first. So we go to Library Manager and with the command Add Library we add the Util library. It has a Controller folder, in which the possible realizations of a PID control algorithm are given. The next step is the addition of the POU in the project tree. It should be added within the Application folder. We name this POU the PID_Algorithm. In the Toolbox menu at the right edge of the user interface we select a box and add it into the program. If we name it PID, a PID block with all the inputs and outputs will be selected and shown. So we just need to add all the signals and parameters as shown in Fig. 14.

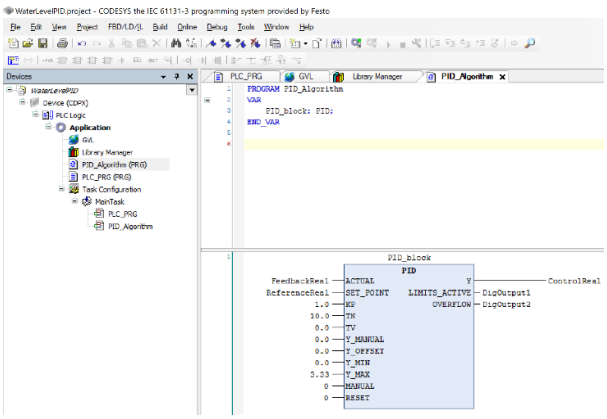


**Figure 14.** PID block in CoDeSys.

I think all the names of the signals and names are clear. The ones that might need explanation are TN and TV. They correspond to TI and TV (Nachstellzeit and Vorhaltezeit in German) [13, 14]. If we want the PLC program to be able to communicate with the, a Symbol Configuration must be added to the Application. Then the code is generated with the GVL checked.

## 4.2 Programming in Designer Studio

In order to create the User Interface on the HMI the Designer Studio programming environment is used. An application can be made by opening a project, naming it and selecting a CDPX-X-A-W-7 in the proper layout (see Fig. 15).

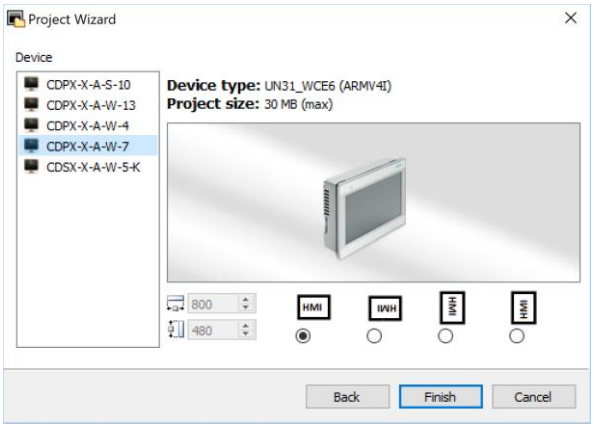


**Figure 15.** Opening of a project in Designer Studio.

Then in the Protocols a protocol CODESYS V3 ETH must be selected. After that in Tags we import the variables from the CoDeSys. We see them as shown in Fig. 16.

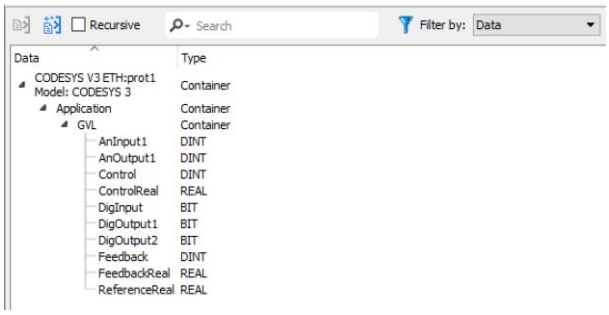


**Figure 16.** Variables imported from CoDeSys into Designer Studio.

After that, the application can be made. One of the simplest possible solutions is shown in Fig. 17.

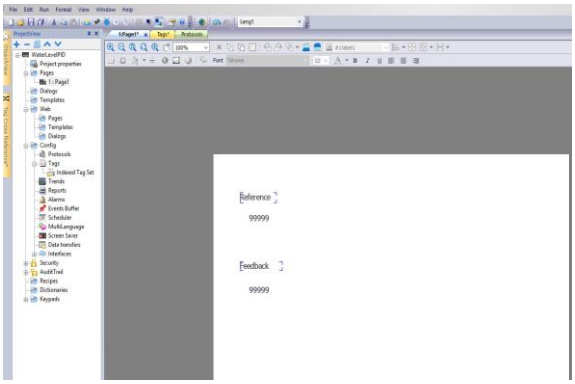


**Figure 17.** Creating a Page in Designer Studio for user interface on the Festo CDPX.

In order to do it, we just need to select the appropriate elements from the Widget Gallery at the right edge of the screen. In our case only labels and numerical fields are needed. When we are adding numerical fields, we must also connect them to the appropriate variables. This can be done as shown in Fig. 18.

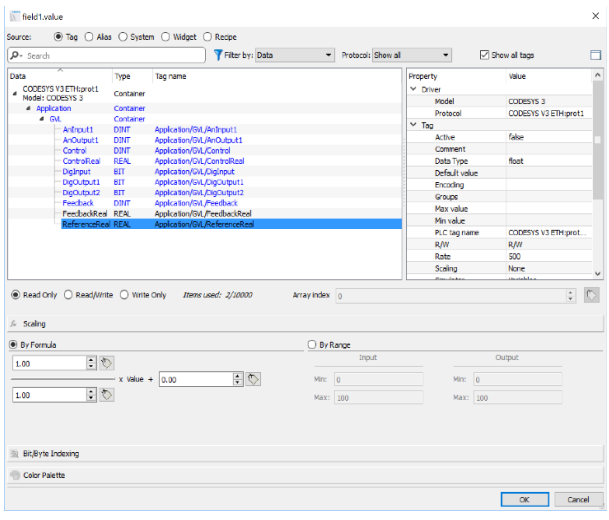

**Figure 18.** Connecting numerical fields to the variables.

After all the programming is done, both programs (the one made in CoDeSys and the one made in Designer Studio) just need to be downloaded to the target. Then the PID controller is functional.

## 5 Results

The PID controller programed in the CDPX was tested in two experiments. The first one is shown in Fig. 19.

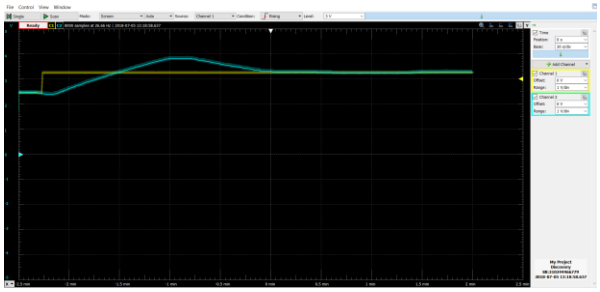

**Figure 19.** Transient response for the reference.

It shows the transient response in case when the reference was increased by 2 cm. The second one (Fig. 20) is the case when the outflow was increased by some appropriate (not quantized) value.

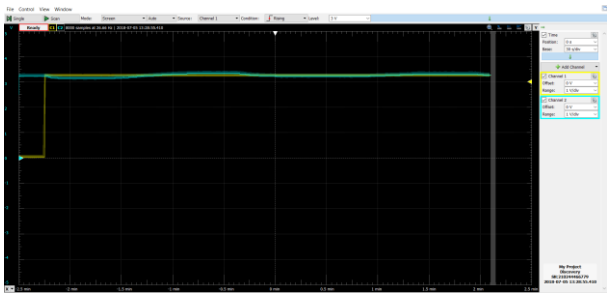

**Figure 20.** Transient response for the disturbance.

We can see that the transient response is satisfactory in both cases. There is no substantial steady state error. The system however doesn't reach the zero-error steady state and stay in it, because the electric drive need 80mV variation in input voltage in order to move. In all the other respects the transients closely resemble the ones calculated with the help of the Laplace transform.

## 6 Conclusions

The presented system is used for lab exercises for students studying the basics of Control Theory. The presented exercise was well received among them. It shows them how to program a PLC on a real application which is very easy to understand as the system is open loop stable.

It can also be seen that Festo CDPX is a good option for implementing PLCs if HMI is needed as well. Their software tools (Festo CoDeSys and Festo Designer Studio) however still lack comfort available from some other environments (Siemens TiaPortal for example). A great advantage of Festo is that their licensing is related to the target device and not to the software. So, it's very convenient for the students to install the software and write the program on their own PC. They just need to test it on one shared target device with the installed license.

## References

1. Dorf, R. C. and Bishop, R. H. 2011. *Modern Control Systems, 12th Edition*. Pearson.
2. Ogata, K. 2010. *Modern control engineering, 5th Edition*. Prentice hall.
3. Hughes, T. A. 1995. *Measurement and Control Basics, 3rd Edition*. The Instrumentation, Systems, and Automation Society – ISA.
4. DiStefano, J. J., Stubberud, A. J. and Williams, I. J. 1990. *Schaum's Outline of Feedback and Control Systems, 2nd Edition*. McGraw-Hill Professional.
5. Bolton, W. 2015. *Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering, 6th Edition*. Pearson Education Limited.
6. Franklin, G. F., Powell, J. D. and Emami-Naeini, A. 2015. *Feedback Control of Dynamic Systems, 7th Edition*. Pearson.
7. Houpis, C. H. and Lamont, G. B. 1991. *Digital Control Systems: Theory, Hardware, Software, 2nd Edition*. McGraw-Hill.
8. Lurie, B., and Enright, P. 2012. *Classical Feedback Control: with MATLAB and Simulink, 2nd Edition*. CRC Press.
9. Nise, N. S. 2015. *Control Systems Engineering, 7th Edition*. John Wiley & Sons.
10. Zak, S. H. 2003. *Systems and Control*. Oxford University Press.
11. Dyke, P. P. 2014. *An Introduction to Laplace Transforms and Fourier series, 2nd Edition*. Springer.
12. Hanssen, D. H. and Lufkin, D. 2015. *Programmable Logic Controllers: A Practical Approach to IEC 61131-3 using CoDeSys*. John Wiley & Sons.
13. Zacher, S. and Reuter, M. 2017. *Regelungstechnik für Ingenieure: Analyse, Simulation und Entwurf von Regelkreisen, 14. Auflage*. Springer Vieweg.
14. Unbehauen, H. 2008. *Regelungstechnik I: Klassische Verfahren zur Analyse und Synthese linearer kontinuierlicher Regelsysteme, Fuzzy-Regelsysteme, 15. Auflage*. Vieweg Teubner.