Comparison of different DC motor positioning control algorithms

N. Baćac*, V. Slukić*, M. Puškarić*, B. Štih*, E. Kamenar**, S. Zelenika**

* University of Rijeka, Faculty of Engineering, Rijeka, Croatia
** University of Rijeka, Faculty of Engineering and Centre for Micro and Nano Sciences and Technologies, Rijeka, Croatia

ekamenar@riteh.hr

Abstract – A comparison between different DC motor positioning control algorithms is performed in this work. Transient responses while employing a PID controller, a cascade controller and a state-space controller are considered. LabVIEW programming environment with a suitable acquisition card and a miniature DC motor with an integrated encoder are used for experimental assessment. Calculations and control system simulations are made using Matlab. The PID controller is implemented via the predefined PID block in LabVIEW. In turn, the state-space controller is modelled by using Matlab while the accuracy of the results is confirmed experimentally using LabVIEW. The cascade controller is developed as a series of two Proportional-Integral (PI) controllers, one representing the positioning and the other the velocity loop. The obtained results allow establishing that positioning control via the state-space controller has the fastest response and the lowest settling times.

I. INTRODUCTION

A widely used actuator in positioning systems is a Direct Current (DC) motor. It finds application in many of today's mechatronics systems such as robots, precision positioning machines or industrial applications. Positioning control of mechatronic devices is also used in situations when there is a need for an accurate response in a predictable and repeatable manner. In a pick-and-place machine for production of Printed Circuit Boards (PCB), for example, components must be placed precisely on the board before the soldering process. Another example is a robot manipulator that uses several motors for 2D or 3D positioning of the robotic arms. Although stepper motors are sometimes employed for these purposes, DC motors can also be a viable solution. However, when DC motors are used, a feedback sensor is needed in order to establish positioning control [1, 2].

A DC motor with an embedded quadrature rotational incremental encoder as a feedback sensor is employed in this work. Three different control algorithms are compared in the terms of simulations in Matlab and experimental results obtained by using the LabVIEW programming environment.

II. DC MOTOR POSITIONING CONTROL ALGORITHMS

A. PID control

One of the most common controllers in industrial applications and control systems is the Proportional-Integral-Derivative (PID) controller. The transfer function of an ideal parallel PID structure can be expressed as (refer to the list of symbols at the end of the paper) [1]:

$$G_R(s) = \frac{U(s)}{E(s)} = K_P\left(1 + \frac{1}{T_I s} + T_D s\right) \tag{1}$$

The PID parameters are defined as:

- K_P proportional gain
- T_I integral time
- T_D derivative time

$E(s)$ can be calculated as the difference between the reference position $Y_0(s)$ and the actual position $Y(s)$:

$$E(s) = Y_0(s) - Y(s) \tag{2}$$

In the time domain, the PID controller can be expressed as:

$$u(t) = K_P \cdot \left(e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau + T_D\frac{de(t)}{dt}\right) \tag{3}$$

Using the finite time values:

$$t \to kT_s \tag{4}$$

a discretized form of PID controller can be obtained [1]:

$$u(kT_s) = K_P \cdot \left(e(kT_s) + \frac{T_s}{T_I}\sum_{i=0}^{k} e(iT_s) + \frac{T_D}{T_s}[e(kT_s) - e(kT_s - T_s)]\right) \tag{5}$$

The PID controller is implemented in this work via the predefined PID block in LabVIEW. The tuning of the PID controller parameters is conducted in two steps. The Ziegler-Nichols method is used first to achieve a rough estimate of the gains; in a second step, an experimental method of fine-tuning of PID parameters is performed.

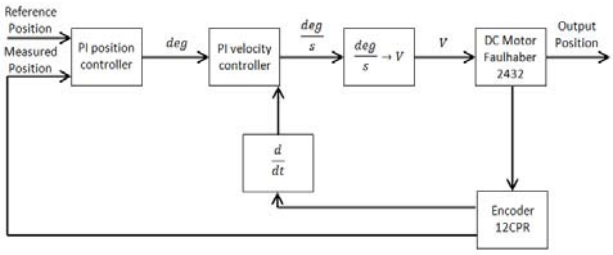

Figure 1. Cascade control scheme

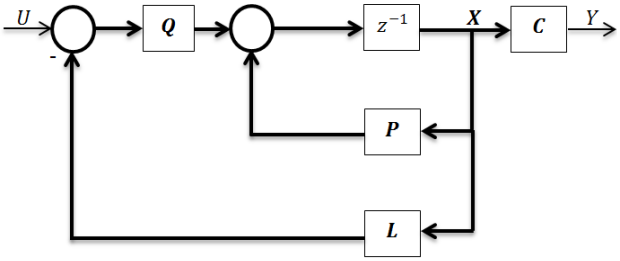

Figure 2. z-transformation of the state-space model

B. Cascade control

Cascade control is composed of two loops: the velocity and the position loop. In the simplest form, an analogue command (set point velocity) is compared with the signal from the feedback sensor (incremental rotational quadrature encoder) to generate voltage that induces rotational motion. The produced torque will speed up or slow down the actuator in order to reach the velocity set point. The most known velocity loop is the Proportional-Integral (PI) loop and it is constituted by two parameters: a proportional gain (K_P), which scales the velocity error, and an integral time constant (T_I), which defines the integration time. A velocity loop itself cannot ensure that the actuator stops in a certain position. Hence, one of the common configurations is to place a positioning loop in cascade (series) with the PI velocity controller (Fig. 1) [3].

Tuning of the regulators in cascade can often prove to be a challenging task since there are four parameters to tune [3]. The parameters are thus tuned in this work by using a custom developed Matlab model. The resulting parameters are then implemented in the LabVIEW environment where additional online fine tuning is performed in order to match better real system response.

C. State-space control

State-space control derives from the state-variable method of representing differential equations. When compared with transfer-function based control, state-space control is designed by working directly with the state-space-variable description of the system [2].

Advantages of state-space design are particularly relevant for Multiple-Input – Multiple-Output (MIMO) systems, although the system considered in this work is a Single-Input – Single-Output (SISO) system. A further advantage of this method is its robustness to dynamic perturbations in the system [2].

A process with one input $u(t)$ and one output $y(t)$ can hence be described in a state-space model as [4-6]:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t)$$
$$y(t) = \mathbf{C}\mathbf{x}(t) \tag{6}$$

where **A** is the system matrix which correlates the current state with its change, **B** is the control matrix and determines how the system input affects the state change and **C** is the output matrix which determines the relationship between the system state and its output. The considered process in a discrete form can thus be described as:

$$\mathbf{x}[k+1] = \mathbf{P}\boldsymbol{x}[k] + \mathbf{Q}u[k]$$
$$y[k] = \mathbf{C}\mathbf{x}[k] \tag{7}$$

The **P** and **Q** matrices can be defined by using the fundamental matrix $\boldsymbol{\Phi}(s)$ [4]:

$$\boldsymbol{\Phi}(s) = (s\mathbf{I} - \mathbf{A})^{-1}$$
$$\mathbf{P} = \mathcal{L}^{-1}\{\boldsymbol{\Phi}(s)\} \tag{8}$$
$$\mathbf{Q} = \mathcal{L}^{-1}\left\{\frac{\boldsymbol{\Phi}(s)}{s}\right\}\mathbf{B}$$

The process can also be described by applying the z-transformation of the equations of the state-space model [4]:

$$\mathbf{X}(z) = (z\mathbf{I} - \mathbf{P})^{-1}\mathbf{Q}U(z)$$
$$Y(z) = \mathbf{C}(z\mathbf{I} - \mathbf{P})^{-1}\mathbf{Q}U(z) \tag{9}$$
$$G(z) = \frac{Y(z)}{U(z)} = \mathbf{C}(z\mathbf{I} - \mathbf{P})^{-1}\mathbf{Q}$$

Now the state-space controller equation can be defined:

$$U(z) = -\mathbf{L}\mathbf{X}(z) \tag{10}$$

The basic principle of the synthesis of the state-space controller is to achieve the desired closed-loop dynamics with proper values of the **L** vector. These values can be obtained by using the following equations:

$$z\mathbf{X}(z) = (\mathbf{P} - \mathbf{Q}\mathbf{L})\mathbf{X}(z)$$
$$|z\mathbf{I} - \mathbf{P} + \mathbf{Q}\mathbf{L}| = 0 \tag{11}$$

where the roots of equation (11) are correlated with the poles of the desired closed-loop dynamics. The state-space controller gain vector **L** can then be calculated by using Ackermann's formula [4]. The block diagram of the resulting state-space model is shown in Fig. 2.

D. DC motor model in state-space

The behaviour of a DC motor can be modelled as [5]:

$$\frac{d\varphi}{dt} = \omega \tag{12}$$

$$J\frac{d\omega}{dt} + b \cdot \omega = \frac{K_m}{R_a}(u - K_e \cdot \omega)$$

For simplicity, the armature time constant is neglected.

In turn, in state-space a DC motor can be described as:

$$s\begin{bmatrix}\varphi\\ \omega\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & -\frac{1}{T_m}\end{bmatrix} \cdot \begin{bmatrix}\varphi\\ \omega\end{bmatrix} + \begin{bmatrix}0\\ \frac{K_m}{J \cdot R_a}\end{bmatrix} \cdot u$$

$$Tm = \frac{J \cdot R_a}{b \cdot R_a + K_m \cdot K_e} \tag{13}$$

$$b = K_m \frac{I_{an}}{\omega}$$

The DC motor transfer function is:

$$G(s) = \frac{\Phi(s)}{\Omega(s)} = \frac{\frac{K_m}{J \cdot R_a}}{s^2 + s \cdot \frac{1}{T_m}} \tag{14}$$

while the respective discretized model in state-space is:

$$z\begin{bmatrix}\varphi\\ \omega\end{bmatrix} = \begin{bmatrix}1 & \alpha \cdot T_m\\ 0 & \beta\end{bmatrix} \cdot \begin{bmatrix}\varphi\\ \omega\end{bmatrix} + \frac{K \cdot T_m}{J \cdot R_a}\begin{bmatrix}\gamma \cdot T_m\\ \alpha\end{bmatrix} \cdot u$$

$$\beta = e^{-\frac{T_s}{T_m}}$$
$$\alpha = 1 - \beta \tag{15}$$
$$\gamma = \frac{T_s}{T_m} - \alpha$$

The requirements for aperiodic closed-loop dynamics imply then:

$$\xi = \frac{1}{\sqrt{2}}$$
$$\omega_n = \frac{1}{T_m\sqrt{2}} \tag{16}$$

while the recommended sample time is [5]:

$$T_s = \frac{0.2}{\omega_n} \tag{17}$$

Vector **L** can hence be approximately calculated as [5]:

$$\mathbf{L} = \frac{J \cdot R_a}{K_m \cdot T_m}\left[\frac{0.4983}{T_m} \quad 0.0672\right] \tag{18}$$

By inserting the characteristic values for the used DC motor (see Table I), vector **L** becomes:

$$\mathbf{L} = [1.1173 \quad 0.0011] \tag{19}$$

To estimate finally the variables used in the control algorithm, the Matlab software is used.

TABLE I. FAULHABER 2342 DC MOTOR CHARACTERISTICS

Moment of inertia	J	$5.7 \cdot 10^{-7}$ kg · m^2
Torque constant	K_m	$13{,}4 \cdot 10^{-3}\ \frac{\text{Nm}}{\text{A}}$
Back-EMF constant	K_e	$1{,}4 \cdot 10^{-3}\ \frac{\text{V}}{\text{rpm}}$
Armature resistance	R_a	1,9 Ω
Nominal voltage	U_n	12 V
Nominal armature current	I_{an}	75 mA
Friction coefficient	b	$1\ \frac{\text{mNm}}{\text{rpm}}$

III. EXPERIMENTAL SET-UP

The assessment of the characteristics of the described positioning control algorithms is performed on an experimental set-up, whose block-scheme is shown in Fig. 3. The system is composed of a DC motor coupled with a gearbox and an encoder, the operational amplifier and the control system based on the National Instruments (NI) hardware and the LabVIEW software.

A. Actuator and encoder

The main component of the experimental set-up is the Faulhaber 2342 DC motor with an embedded planetary gearbox and an incremental rotational encoder [7]. The main characteristics of the actuator are given in Table I. As shown in Fig. 4, the shaft of the actuator is connected to the gearbox unit (1) with a reduction ratio 1:64. On the opposite side, an encoder (2) is installed.

The used encoder is an incremental encoder with 2 channels (A and B) with a 90 degree phase delay and 12 cycles per revolution (cpr). Two encoder channels enable not only position and speed but also the direction of movement to be determined. A very important parameter for encoder measurements is the encoding type, which directly affects encoder resolution. In this application, rising and falling edges on both channels are counted (Fig. 5). This type of encoding is usually referred as X4 encoding [8-10].

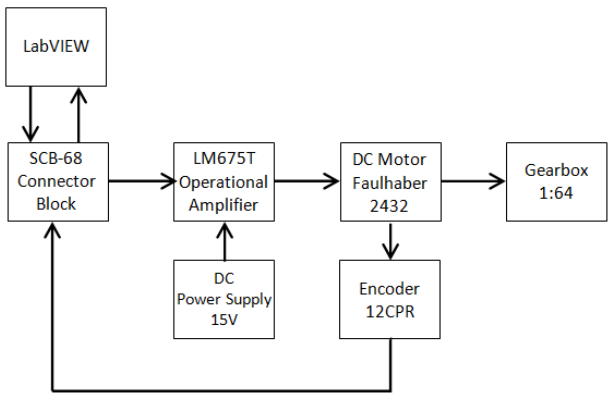

Figure 3. Experimental set-up scheme

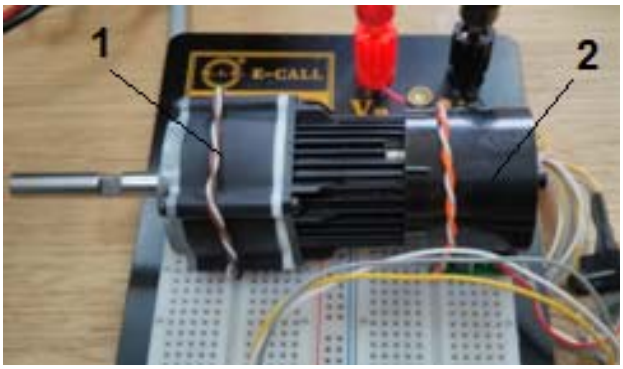

Figure 4. DC motor with gearbox and encoder

B. *National Instruments Hardware*

The electronics of the experimental set-up is based on the National Instruments PXI-1050 chassis, including a PXI-8196 embedded controller and a PXI-6221 Data Acquisition Card (DAQ) which is connected to a SCB-68 connector block [11]. The SCB-68 has 68 terminals with digital and analogue inputs and outputs. Counter inputs are used for connecting the encoder while the analogue output is used for driving the DC motor.

IV. LABVIEW CONTROL IMPLEMENTATION

The LabVIEW development environment is used for implementing the control algorithms and storing measurement data. This configuration meets the requirements for embedded system design such as real-time computing, wide choice of peripheral equipment and extensive support for accessing instrumentation hardware. Moreover, its library contains predefined mathematical blocks that process acquired data to enable appropriate outputs. In the considered case, the DAQmx *Start Task block* is used to trigger the used channels. The acquired data is then passed to the DAQmx *Read block* that, due to continuous position readings, is located within the time loop. Similarly, voltage output is driven using the DAQmx *Write block*. On the other hand, measurements are stored in a CSV file. This file is created by using a *Write-to-spread sheet file block* which enables to capture the data that can subsequently be used for further analysis in Matlab.

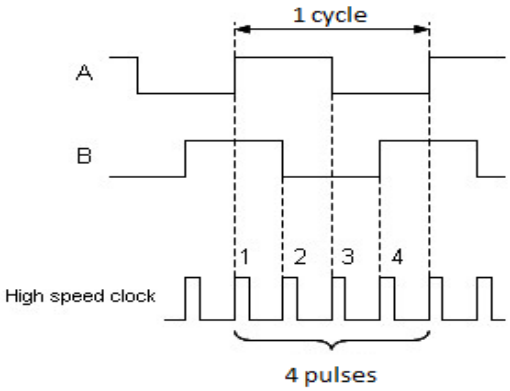

Figure 5. Reading and encoding signals from the encoder

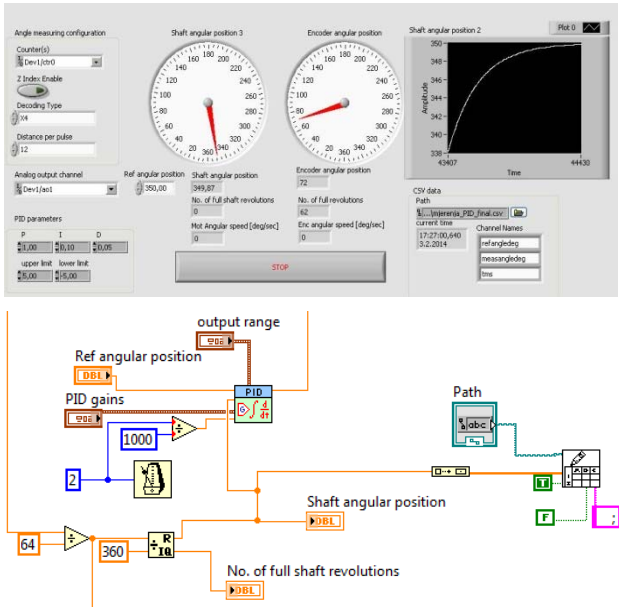

Figure 6. PID regulation VI: front panel (up), block diagram (down)

A. *PID control*

PID control is implemented in LabVIEW by using the predefined PID block from the *LabVIEW Control, Design and Simulation Module*. The respective front panel as well as the main elements of the block diagram are shown in Fig. 6.

Input and output channels have to be selected first.

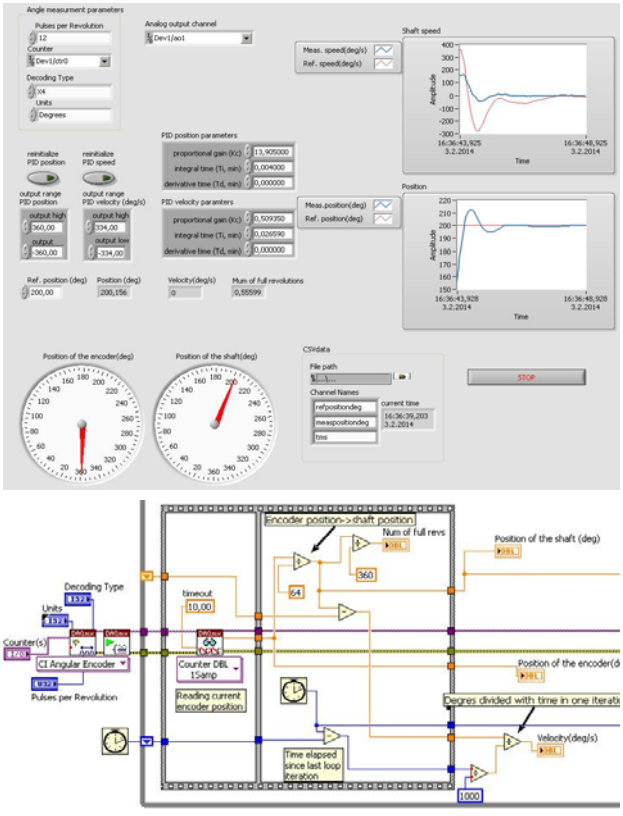

Figure 7. Cascade control VI: front panel (up), block diagram (down)

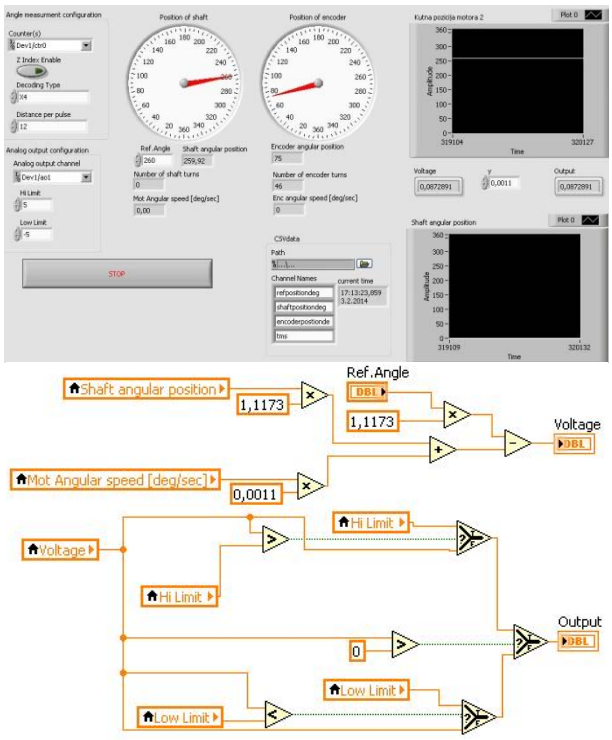

Figure 8. State-space control VI: front panel (up), block diagram (down)

Using the DAQmx counter as input, information about position is obtained from the channels of the encoder. This is compared to the defined reference position so as to determine the analogue input to the actuator.

B. Cascade control

The front panel and the block diagram of the cascade control scheme are shown in Fig. 7. The front panel allows the following input parameters to be defined: number of pulses per revolution, input and output channels, decoding type, position measurement units, parameters for the PI regulators as well as the file path. For a given reference position, the dynamic response of the system can be visualised via a waveform chart and/or a circular gauge. The number of full motor shaft revolutions is also shown. The inferred velocity is hence calculated as shown in the block diagram in Fig. 7. Experiments allowed establishing that a disadvantage of the used method is that, for sample times shorter than 50 ms, the velocity signal results in errors because the computational routines are slower than motor's response.

C. State-space control

The state-space regulator is implemented in LabVIEW using equations (12-19) – Fig. 8. The measured position and velocity are scaled by the gain vector **L** using equation (19). These values are then compared to the desired reference values. Since there are no additional time constants (no additional poles) added to the system, the state-space control algorithm has faster response times than the PID and the cascade controller. On the other hand, since state-space control has only proportional gain, a residual steady-state error is present. To minimize this error, the reference position is multiplied with a compensation gain.

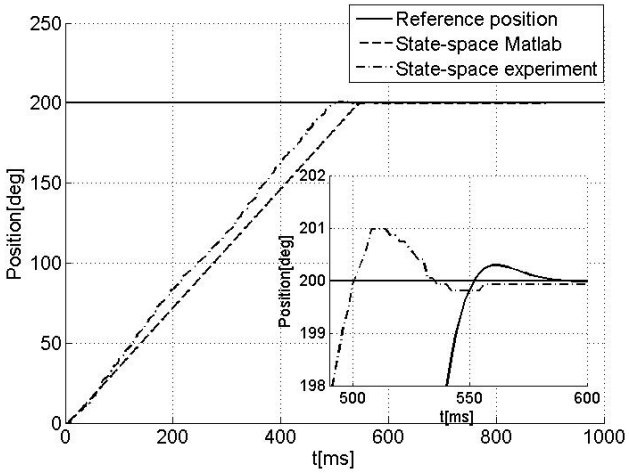

Figure 9. State-space transient response

V. EXPERIMENTAL RESULTS AND DISCUSSION

To test the performances of the described DC motor position control systems, a set of experiments using PID, cascade and state-space control typologies is performed. All tests are made with an input step command from 0° to 200°. DC motor input voltage is limited to ± 5V. The parameters of the PID and the cascade controllers are given in Tables II and III both for the simulations and the experiments. It is interesting to notice here that the simulations did not allow the correct integral time of the positioning loop of the cascade control to be determined. This value will thus be obtained in future investigations.

The comparison of simulated and experimentally obtained transient responses in the case of the state-space controller is shown in Fig. 9 and Table IV. From the magnified detailed view of the system dynamic response during settling, it can be seen that an aperiodic transient response occurs with a negligible difference in rise times between the simulated and actual results. In the case of the PID and the cascade controllers, due to simplified modelling assumption, some discrepancies of the simulated and experimental dynamic results occur. Future investigations will be done to obtain better matching of the simulated and experimental results.

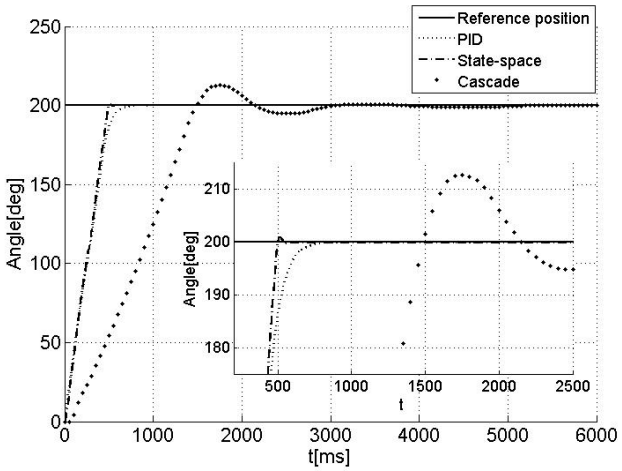

Figure 10 Combined experimental results

TABLE II PARAMETERS FOR THE PID CONTROL METHOD

	MATLAB	Experiment
Proportional gain K_p	0.576	0.6
Integral time T_I[ms]	0.313	0.3
Derivate time T_D [ms]	0.021	0.02

TABLE III PARAMETERS FOR THE CASCADE CONTROL METHOD

		MATLAB	Experiment
Positioning loop	Proportional gain K_p	13.56	13.905
	Integral time T_I [ms]	?	4
Velocity loop	Proportional gain K_p	0.508	0.508
	Integral time T_I [ms]	5	5

TABLE IV DYNAMIC RESULTS FOR THE STATE-SPACE CONTROL METHOD

	MATLAB	Experiment
Rise time τ_r [ms]	430	390
Percent overshot σ [%]	0.35	0.5
Settling time τ_s [ms]	580	550
Steady-state error [%]	0	0.04

TABLE V COMPARISON OF DYNAMICS FOR DIFFERENT CONTROL METHODS (EXPERIMENTS)

	PID	Cascade	S.S.
Rise time τ_r [ms]	425	1120	390
Percent overshot σ [%]	0	6.35	0.5
Settling time τ_s [ms]	800	3500	550
Steady-state error [%]	0	0.1	0.04

The comparison of transient responses obtained experimentally for all three used controllers is given in Fig. 10. Rise time, percent overshoot, settling time and steady-state error for each control algorithm are in turn given in Table V. It can be concluded that the fastest response is obtained by using the state-space control algorithm. The PID control algorithm results, however, in the smallest steady-state error.

VI. CONCLUSIONS AND OUTLOOK

An overview of different DC motor control approaches is given in this work. A conventional PID controller is employed first. The respective parameters are obtained using the Zieger-Nichols method and fine tuning. A cascade controller is developed next. Finally, the state-space controller with positioning and velocity loop is employed. A Matlab model of the used actuator is established in order to simulate different control approaches. Controllers are then implemented in the LabVIEW environment and experiments are conducted.

By comparing experimental results (Table V), it is concluded that positioning control via the state-space controller has the fastest response and the lowest settling times. Cascade control can be efficiently used, although the tuning of its parameters can often be cumbersome and computationally more intensive due to the presence of two PI blocks and the needed velocity calculation. This all limits the execution time which directly affects system's dynamic response. Improvements of cascade control could be achieved by using real-time hardware (e.g. the NI FPGA module) or if direct measurement of velocity would be possible. PID control results in negligible steady-state errors and acceptable rise and settling times.

LIST OF SYMBOLS

A	system matrix
B	control matrix
b	friction coefficient [mNm/rpm]
C	output matrix
$E(s)$, $e(t)$	difference between the reference and the process value
$G_R(s)$	PID transfer function
I_{an}	nominal armature current [mA]
J	motor inertia [kg·m^2]
k	finite time values factor (k=1, 2, 3, …)
K_e	back EMF constant [V/rpm]
K_m	torque constant [Nm/A]
K_P	proportional gain
L	state-space controller gain vector
P	system matrix in discrete domain
Q	control matrix in discrete domain
R_a	armature resistance [Ω]
s	Laplace variable
t	time variable
T_I, T_D	integral and derivative time constants
T_m	mechanical time constant
T_S	sample period
$U(s)$, $u(t)$	output from the controller
U_n	nominal voltage [V]
Y(s), y(t)	process value
Y_0(s)	reference position
z	discrete (z) domain variable
Δt	time change [ms]
$\Delta\varphi$	motor shaft angle change [°]
ξ	attenuation coefficient
σ	percent overshot [%]
τ_r	rise time [ms]
τ_s	settling time [ms]
φ	motor shaft position [°]
ω	velocity of the motor shaft [rpm]
Φ	fundamental matrix

REFERENCES

[1] E. Kamenar and S. Zelenika, "Micropositioning mechatronics system based on FPGA architecture," Proc. 36th International Convention on Information and Communication Technology, Electronics and Microelectronics MIPRO, pp. 138-143, 2013.

[2] G. F. Franklin, J. D. Powel and A. Emami-Naeini, "Feedback control of dynamic systems – 2nd ed.," Addison-Wesley, 1991.

[3] URL: www.machsupport.com

[4] D. Matika, "Sustavi digitalnog upravljanja (Digital control systems)," University of Rijeka, 2005.

[5] R. Cupec, "Diskretni sustavi upravljanja," University of Osijek, 2008. URL: http://www.etfos.unios.hr/upload/OBAVIJESTI/obavijesti_diplomski/diskretni_sustavi_upravljanja_19-11-2009.pdf

[6] K. J. Astrom and B. Wittenmark, "Computer controlled systems: theory and design – 3rd ed.," Prentice Hall, 1996.

[7] URL: www.faulhaber.com/uploadpk/EN_2342_CR_DFF.pdf

[8] URL: www.gurley.com/Encoders/Understanding_Quadrature.pdf

[9] G. S. Gordon, "Square Waves And Pulses: A Clarification," Measurements & Control, 1988.

[10] URL: www.dynapar.com/uploadedFiles/Downloads/Encoder_Glossary_of_Terminology.pdf

[11] URL: www.ni.com