Real-time Implementation of a Full State Feedback Controller for a DC Motor

K. Love*, C. Kriger, N. Tshemese-Mvandaba
Department of Electrical, Electronics and Computer Engineering, Faculty of Engineering and the Built Environment, Cape Peninsula University of Technology, Cape Town, South Africa
*corresponding author's email: kevinluffylove@gmail.com

Abstract *– This research work presents the real-time implementation of a Full State Feedback (FSFB) controller, which controls the angular position of a Direct Current (DC) motor. The designed Simulink controller is transformed into an object that is used in Beckhoff's TwinCAT programming environment to interface with an actual DC motor test rig. The MATLAB command code required for the transformation is described with all other software required. The test rig used for the implementation is also shown and discussed. Lastly, a case study is done to test the control system's response and limitations to various positional set point changes. This research contributes to the study of FSFB control for DC motors by providing a practical method to implement the simulated controller on a hardware device that is used in industry. The implemented control system can be used in industry and not just for testing purposes. This method can be used to implement any designed FSFB controlled DC motor model.*

Keywords: *DC motor, Full State Feedback Controller, MATLAB Simulink, State Space*

Article History
Received 31 July 2024
Received in revised form 1 September 2024
Accepted 1 October 2024

I. Introduction

Angular position control of a DC motor is a well-versed topic that has been studied by multiple authors. The research field entails adding a type of controller to change the response of a motor as it changes position to a set point. The responses simulated and studied include settling time, rise time, overshoot, and steady-state error. Input step responses are usually compared between the open-loop DC motor control system and the DC motor control system with added controller. Many different controllers have been modelled and simulated, for example Proportional-Integral-Derivative (PID) controllers in [1] - [4] and Full State Feedback (FSFB) controllers in [5] - [12]. These authors went as far as simulating the control systems but never implemented the controllers in real-time using actual DC motors.

References [13] – [20] have implemented their developed controllers on actual DC motors using microcontrollers with a pulse width modulated output.

Reference [13] uses an ATMega 2560 series microcontroller to implement a fuzzy logic controller for a DC motor. The implementation is done to compare the reliability of the model when implemented as the simulation does not consider noise and other disturbances. The results are successful, and no overshoot is evident in the output response.

Reference [14] uses an Arduino microcontroller to implement a sliding mode controller to prove that the simulated results can be implemented. The results are compared with a PID controller, but it is seen that oscillations are present in the rotation of the motor and therefore improvement is required.

References [15] and [16] use Arduino microcontrollers to control a DC motor with PID control. The results in [15] show that the controller successfully controls the motor driver's voltage, speed, position and velocity and compares the open-loop and closed-loop system responses. Results with low error oscillations are recorded in [16] but errors do occur when trying to turn the motor at smaller rotations due to friction in the gearbox.

A PIC microcontroller is used in [17] to implement the simulated mathematical model of the DC motor control system. The author's results show that there is a relationship between the overshoot and speed of response and that a sacrifice of one parameter must be made to improve the other.

Reference [18] implements the developed control system using LabVIEW and an Arduino Uno microcontroller. The results show that the system is robust and that the PID controller is satisfactory for the required implementation.

Reference [19] controlled a DC motor with a PID and FSFB controller using an Arduino microcontroller and motor driver. The results showed that the FSFB controller has a better system response than the PID controller. The FSFB controller showed good performance with robust characteristics as the same response of the DC motor occurred for different input setpoints. The state space approach using pole placement is a more modern control technique and will be used in this research work to control a DC motor in real-time.

Reference [20] also uses an FSFB controller with pole placement technique due to the simplicity of control. The author also investigates multiple journals which use FSFB control but only do theoretical modeling and simulations. The author states that the practical constraints of simulated controllers aren't often tested as the price of sensors required to build test rigs are often too high. This results in difficulty when trying to implement a modelled system in an actual industrial application.

The use of microcontrollers in [13] – [20] is fine for testing responses for research purposes but not appropriate for industrial applications. This research work considers the implementation of a DC motor that can be used in real-time industrial applications such as CNC machines, robotic arm control, and radio antenna positioning. This is done by implementing the control system using a Programmable Logic Controller (PLC) instead of a microcontroller.

The implementation work in this research paper is an extension of the designed and simulated controller in [12] where the author has gone one step further than just using a single step response by using random position input set points to test the robustness of the controller. In [12] the controller is developed and simulated using MATLAB/Simulink. The results show that the FSFB controller is fast and consistent, but no implementation is done to prove that the control system will work in a real-time application.

In this research paper, the controller from [12] is transformed from a Simulink model into a The Windows Control and Automation Technology (TwinCAT) 3.1 software object to allow the model to be used in the PLC programming environment. The transformed simulation model interacts with the DC motor from a PLC using an EtherCAT network to a remote motor controller. The motor controller outputs a voltage to the DC motor to turn it, and an incremental encoder is used to feedback the angular position of the motor and close the control loop. A test rig is built to create the closed-loop system that interfaces with the transformed Simulink model. The test rig consists of a Beckhoff C4015 IPC, Beckhoff EK1100 EtherCAT coupler, Beckhoff EL7342, 12V DC motor and an OMRON incremental encoder for feedback.

The closed-loop control system is tested with various position set point changes to prove the modelling of the system is accurate and that the model can be used in real-time. The successful implementation also shows that Simulink models can be transformed to TwinCAT objects to allow for implementation on actual hardware and not just simulation. This research contributes to the field of real-time positional control of DC motors using a state space approach by showing methods used to convert any DC motor state space model into a TwinCAT object that can be used for controlling an actual DC motor. The research also shows methods on how to test the limitations of the closed-loop control system to make the controller industry ready.

II. Simulink Model

The state space model of the DC motor control system with full state feedback and integral control from [12] is shown in (1) and (2). Fig. 1 shows the Simulink block diagram of the state space equations.

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \theta_i \end{bmatrix} = \begin{bmatrix} -5.26 & -2.2862 & 0 \\ 1 & 0 & 0 \\ 0 & -1.1431 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \theta_i \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} r(t) \quad (1)$$

$$y(t) = \begin{bmatrix} 0 & 1.1431 & 0 \end{bmatrix} \begin{bmatrix} x \\ x_n \end{bmatrix} \quad (2)$$

This controller robustness has been simulated and tested against random positional set point changes and will therefore be used as the example control system to implement. The results showed that the addition of the full state feedback controller to control position of the DC motor reduced the rise time of the response from 4.013s to 0.966s. With added integral control, the steady state error of 9.84 is reduced to zero.

For this work, the model needs to be ported from Simulink to the TwinCAT environment, as a Beckhoff PLC is used to integrate with the actual DC motor. This transformation requires adjusting the current Simulink model to make it useable in TwinCAT. The model needs to include parameters that are required by TwinCAT to control the DC motor as a servo with encoder feedback. These parameters include encoder feedback, velocity, deceleration, acceleration, and jerk. A set point position and position output are already present in the existing model and are also required. The new Simulink model is shown in Fig 2.

The DC motor angular position is measured in millimeters; therefore, it is necessary to convert all the parameters from radians per second to millimeters per second. This is necessary as the current Simulink model uses radians per second. This conversion is possible by multiplying the output parameters by 0.02m, which is the radius of the DC motor shaft, and then by 1000 to convert from meters to millimeters. The reverse is done to convert the input parameters.

The output velocity is found using the C matrix. The output of the modelled DC motor is multiplied by 1.1431 to get the value of the two states, position and velocity. Position uses the second entry of the C matrix and velocity uses the second entry. Therefore, velocity is found by multiplying the output of the modelled DC motor by matrix [1.1431 0]. The output velocity is limited to 565.5 mm/s as this is the limit of the actual DC motor.

The acceleration is found by taking the derivative of the output velocity. This is done in the Simulink model by using a derivative block before converting to millimeters per second. The same is done for deceleration but the velocity is first multiplied by -1. The value for the jerk parameter is found by deriving the acceleration output.

Now that all parameters required for servo control are present in the Simulink model it is ready to be converted to a TwinCAT object.

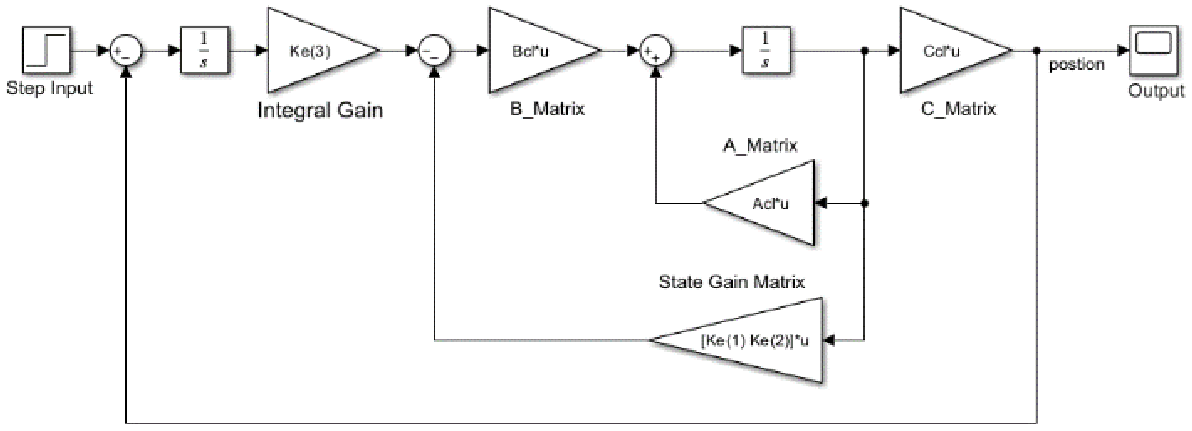


Fig. 1. DC Motor closed-loop control system with FSFB and integral control

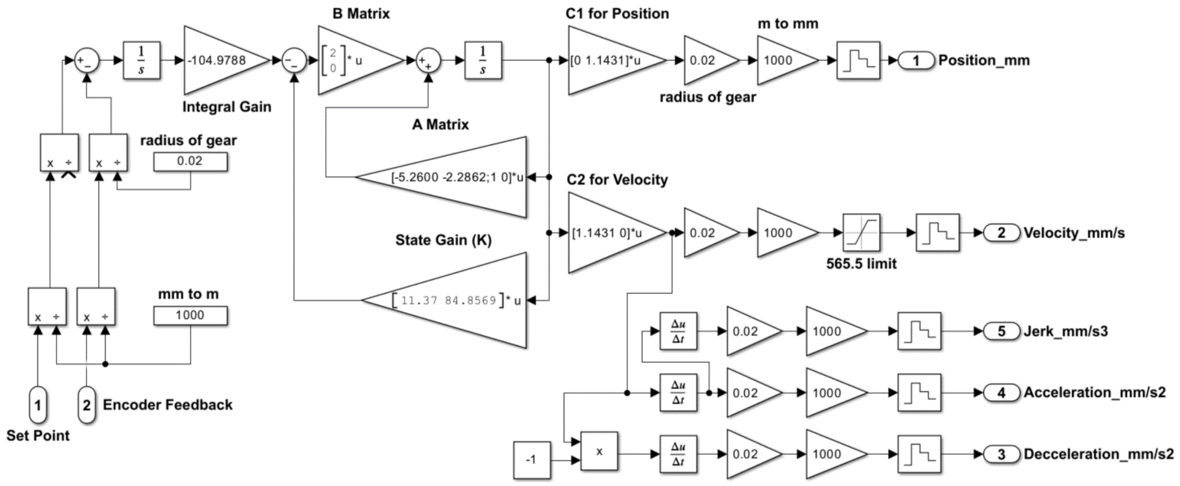


Fig. 2. Updated Simulink model which includes additional input and output parameters

III. Transformed TwinCAT Object

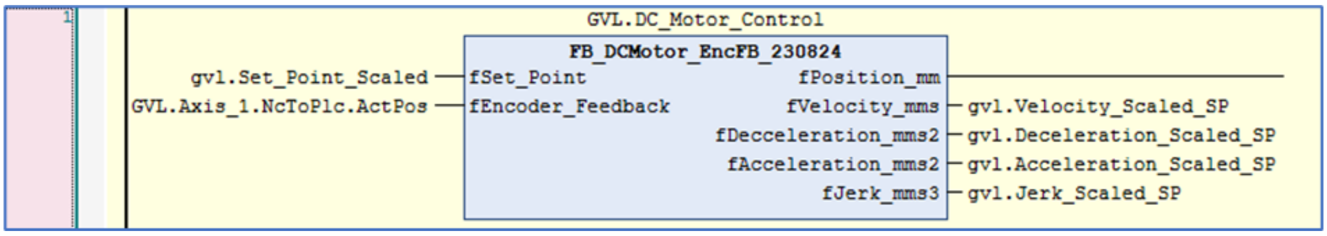
Transforming a Simulink model into a TwinCAT object requires the installation of Beckhoff’s TE1400 target software which allows for seamless transitions between two software development environments [21]. This target specifically converts Simulink block diagrams into C++ code that can be used by the TwinCAT programming environment. The software is licensed but models with less than 100 blocks, 5 input signals, and 5 output signals can be transformed without license.

The target software automatically converts and imports the code into TwinCAT by running a series of MATLAB commands that automatically configures and transfers the model to the TwinCAT programming environment. The first command opens the Simulink model which needs to be transformed. The second command changes the solver type from continuous to fixed step mode. This is necessary as the PLC run time is cyclic and therefore the model will be executed every cycle and not continuously.

The third and fourth MATLAB commands change the formatting of the model’s files name to suit the TwinCAT naming conventions. Lastly, the Simulink Coder build command is run to build and compile the model.

After the commands are run, the transformed model can be used as a function block in the Controller Development System (CODESYS) environment. Fig 3 shows the function blocks in ladder logic with all the inputs and output pins which match the parameters required by the Simulink model.

The global variables connected to the function block pinouts are linked to the actual DC motor through the EtherCAT network and motor terminal controller. When a set point change occurs, the velocity, deceleration, acceleration and jerk are sent to the motor causing it to turn to the set position. As the position changes, the encoder feedback changes and manipulates the outputs of the function block to allow the motor to reach the desired set point at the same response times of the simulated models. This function block is a template that can be used multiple times if more DC motor need to be controlled. For example, a robotic arm with 3 axes could use a function block for each DC motor connected to each axis. The set point for each axis can be controlled by G-code commands with each motor having separate encoder feedback. If a different sized DC motor is used, the state space model parameters must be updated in Simulink before transforming the model into the TwinCAT object.



```
                                  GVL.DC_Motor_Control
                                 FB_DCMotor_EncFB_230824
    gvl.Set_Point_Scaled ─fSet_Point                fPosition_mm─
GVL.Axis_1.NcToPlc.ActPos ─fEncoder_Feedback        fVelocity_mms─ gvl.Velocity_Scaled_SP
                                          fDecceleration_mms2─ gvl.Deceleration_Scaled_SP
                                           fAcceleration_mms2─ gvl.Acceleration_Scaled_SP
                                                  fJerk_mms3─ gvl.Jerk_Scaled_SP
```

Fig. 3. CodeSYS function block of transformed Simulink Model

IV. Hardware Test Rig

A test rig is built to do real-time testing of the developed controller with an actual DC motor. The test rig is shown in Fig 4 and consists of AC voltage distribution breakers (1), a 24V DC power supply (2), a 12V DC power supply (3), a Beckhoff C6015 PLC (4), Beckhoff EK1100 EtherCAT remote module with Beckhoff EL7342 terminal motor controller card (5), a DC motor (6), an Omron incremental encoder (7), and two push buttons (8). Each item plays a crucial role in the closed loop system.

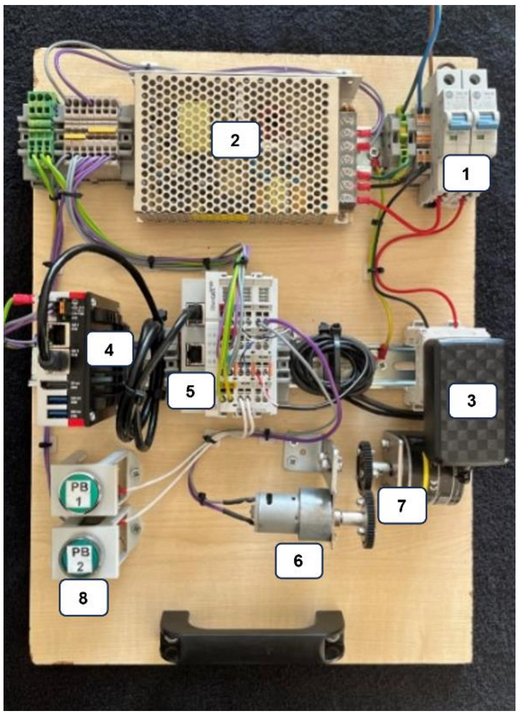


Fig. 4. DC motor with encoder feedback test rig

Two different voltage rated power supplies are used to power the components on the test rig. The 24V DC power supply, powers the PLC, EtherCAT modules, and digital inputs via the push buttons. The 12V DC power supply is used to supply the GB37Y360 12V DC motor. The voltage is regulated through the motor control terminal to change the speed of the motor according to the required response of the controller.

The Beckhoff PLC cyclically executes the transformed Simulink model in real-time. When a change in position set point is registered, the object sends the velocity, acceleration, deceleration, and jerk parameters to the remote EtherCAT module which converts the EtherCAT signals to a voltage that is sent to the DC motor via the motor control terminals. As the motor turns, the incremental encoder feeds the rotations back into the motor terminal controller via pulses. The Omron E6C2-CWZ5B incremental encoder outputs 400 pulses per revolution. The gear ratio is 1:1 between the motors shaft and the encoder, therefore one rotation of the motor is equal to 400 pulses.

The push buttons are added to easily send a position set point change to the DC motor. PB1 is used to turn the motor forward, and PB2 is used to turn reverse the motor by sending a negative positional set point change. The case studies done will alter the magnitude of the position set point to see the motors response.

V. Case Studies

The real-time implementation of the closed loop DC motor control system is tested with input step responses. This is done by sending a set point change to the model's input and recording the response of the actual DC motor. The case studies have four variables to consider to accurately describe the response. These variables are the position step input, the model actual position, the motor set position, and the motor actual position.

The position step input is the value sent to the transformed Simulink model to trigger the velocity output that will allow the motor to start turning. The set point is triggered by either of the push buttons to turn the motor forward or reverse. The model actual position is the position output from the TwinCAT object that is sent to the axis to start turning the motor. The motor set position is the position sent to the actual DC motor by the EtherCAT remote motor terminal. Lastly, the motor actual position is the actual motor position feedback from the incremental encoder via the remote motor terminal.

The motor's response and setpoints are monitored using Beckhoff's Scopeviewer software within the TwinCAT programming environment. Three case studies are done to test the robustness and limitations of the DC motor used. Case study 1 shows the effects of network delays on the control system and how to eliminate them. Case study 2 and 3 tests the limitations of the control system.

A. Case Study 1

A step input response is used to test the real-time application. A single rotation of the DC motor is 10mm therefore a step response of 10mm is used. The first step response is shown in Fig 5. Due to using an EtherCAT network, the effects of network delays can be seen by the reaction time of the actual motor position to the motor set position. A delay of almost 80ms is present, shown in Fig 6, which is caused by the delayed feedback of the encoder position over the EtherCAT network. This delay is also known as a sensor-to-controller delay.

This delay causes a decrease in performance in the system as the motor tries to turn faster to catch up to the initial position set point. This causes the motor to turn too fast and overshoot the initial position step input as seen in Fig 5. The model's actual position and set position also does not reach the 10mm mark as the encoder value fed back into the model is already at this position.

Delays such as sensor-to-controller and controller-to-sensor delays can be dealt with when modeling the control system but for this research Beckhoff's time delay compensation feature is used. This feature uses specific algorithms to predict the desired output of the DC motor. The algorithms use the actual position, set position and following error of the axis with respect to time compared to the physical axis position at the same time, to determine what the position of the motor should be. By monitoring the control signal and corresponding feedback, the time delay compensation feature can control the signals to account for dead time in the system.

The new response with time delay compensation is shown in Fig 7. The motor actual position reaches the position set point without overshoot. The ripple effect of the encoder feedback is caused by running the motor at low speeds to get to the position set point. This is due to the control system consisting of a low efficiency motor, low pulse encoder, and low current output controller. The motor terminal controller is rated at 3.5A and therefore the torque required to run the motor at slower speeds is not possible. This causes the motor to jerk when reaching the set point. To overcome this jerking, a higher efficiency motor can be used to run at lower speeds.

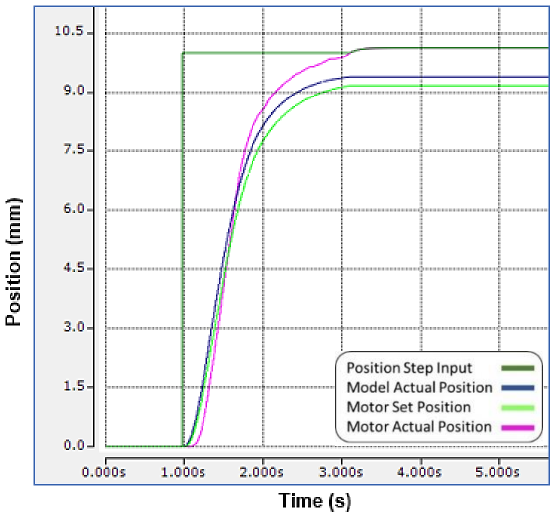


Fig. 5. Step response of 10mm without time delay compensation

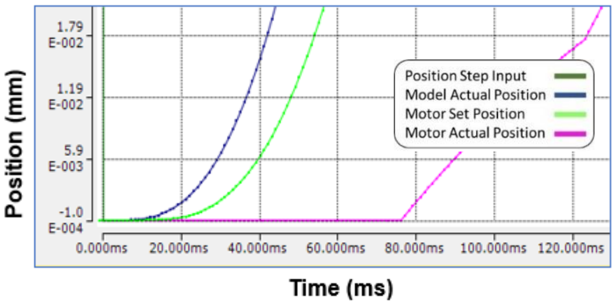


Fig. 6. Panned in X and Y Axis of the response in Fig 5

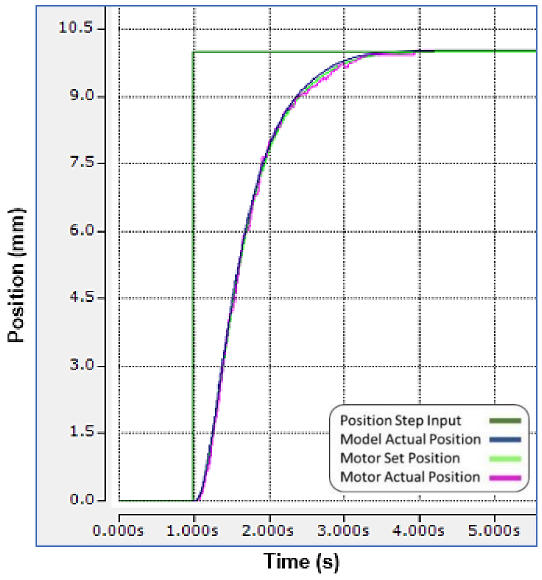


Fig. 7. Step response of 10mm with time delay compensation

B. Case Study 2

Depending on the application, the need to turn the motor to a position less or more than 10mm could be required. A quarter or half turn might be required for smaller movements, whereas three or four turns might be required for a bigger change in position. Therefore, it is necessary to test the limitations of the DC motor control system.

Fig 8 shows the system response to a half turn of the motor of 5mm. Due to factors mentioned in Case Study 1 such as low efficiency motor and low pulse encoder being used, the DC motor jerks while moving to the desired set point. Because the same rise time, settling time, and overshoot is expected due to a FSFB controller being used, the DC motor needs to turn at a very slow velocity at small positional set point changes.

Fig 9 shows the response of the motor to a 40mm change in positional set point, which is four rotations of the motor. The system response is stable and reaches the desired set point at the same response times as Fig 7 where one rotation is set. Because the system is stable at 4 rotations, it is possible to rotate the motor twice for small changes in position set point by using a 2:1 gearbox. This will prevent oscillations in the motor by running the motor at a faster speed while turning the motor at smaller position changes.

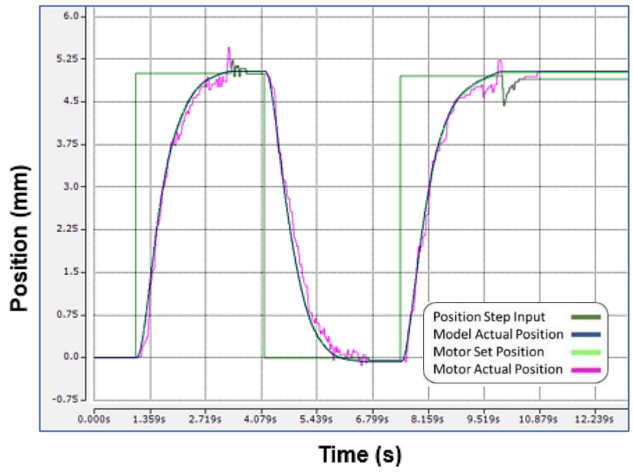


Fig. 8. Step response of 5mm forward and reverse

C. Case Study 3

The maximum positional change of the motor needs to be known before the control system can be used for any application. Fig. 10 shows the response of the DC motor to a positional change of 60mm. As shown, the motor is not able to accelerate fast enough to keep up with the output of the controller. Therefore, the motor tries to speed up as fast as possible and overshoots the input set point. The motor eventually settles and reaches steady state, but this overshoot is not ideal for real-life implementations.

A maximum positional set point of 40mm should be used when applying the controller to an application. This value could change if a higher efficiency motor, or higher count encoder, or higher current rate motor control terminal, is used.

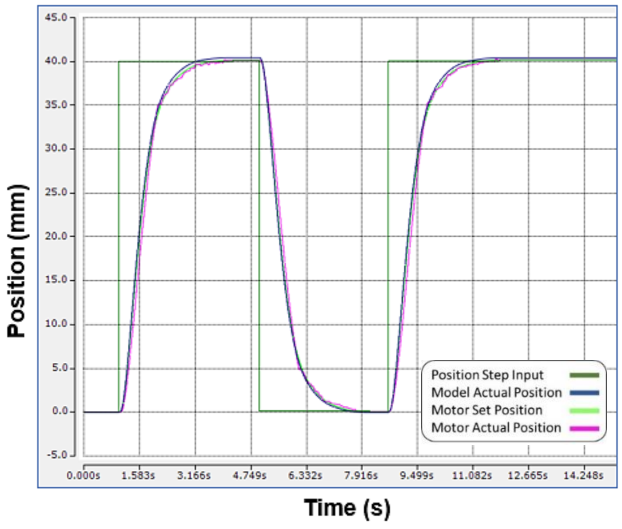


Fig. 9. Step response of 40mm forward and reverse

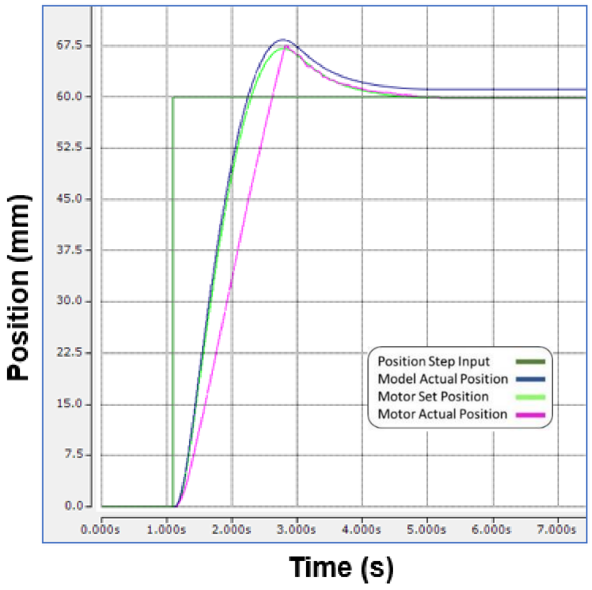


Fig. 10. Step response of 60mm

VI. Conclusion

The Simulink model of the FSFB controller for a DC motor controller in [12] is successfully implemented on a real-time test rig. The model is converted to a TwinCAT object which is used as a function block to interface with the actual DC motor. A method is described to convert the current Simulink model to have the required parameters such as velocity, acceleration, deceleration, accessible before the transformation. MATLAB commands are described and used to complete this transformation before implementing. The methods described can be used for any FSFB controller for a DC Motor.

The test rig hardware is described in detail, explaining the connections between each electrical components as well as the flow of information. The transformed model is downloaded to the Beckhoff PLC which interfaces with the remote terminal motor controller to control the DC motor. The encoder feedback allows the controller to know the position of the DC motor in real-time.

Once all hardware and software is communicating successfully, the control system response is tested. The system is subjected to various position set points to test the limitations and robustness to changes in position.

The first position set point in Case Study 1 is set to allow the motor to turn once. The effects of network delays due to an EtherCAT network present in the feedback loop are shown immediately in Beckhoff's Scopeviewer software. Beckhoff's TwinCAT function time delay compensation is used to remove these delays by estimating the motors feedback once a change in position is commanded, minimizing the effects of network delays.

Case Study 2 tests turning the motor at half a turn as well as a double rotation. A half turn of the motor is possible, but the motor oscillates at low speeds. This is due to the motor not being very efficient and the remote motor terminal card not having a high enough output current to allow the motor to turn as a low torque. Turning the motor twice gives a good response, with the motor not oscillating and the set point is reached without overshoot. This result shows that if the industrial application requires smaller rotations of the motor, a gearbox can be used to allow the motor to turn more times than the required output, depending on the gearbox ratio.

The limitations of the FSFB controller are tested in Case Study 3. A set point of three rotations of the motor is tested and the results show the motor overshooting the set point and reversing back to the set point. This is not ideal in a real-life implementation and therefore the limitation should be considered when designing the application.

This research work contributes to the study of FSFB controllers for DC motors by implementing the designed models on equipment suited for real life industrial applications. The methods use allow for control of any DC motor depending on the application.

Future work can be done to decrease the effects of network delays by adding filters to the Simulink model therefore time delay compensation does not have to be used to predict the controller response. This might make the controller more accurate.

More future work can be done by using higher efficiency motors. This will increase accuracy and prevent oscillations occurring especially at lower speeds. Smaller positional changes will also be possible allowing more opportunity for the controller to be used in industrial applications that require higher accuracy.

VII. Conflict of Interest

The authors declare no conflict of interest in the publication process of the research article.

Author Contributions

Kevin Love conducted the research, drafted the paper, and analyzed the data; Nomzamo Tshemese-Mvandaba, reviewed, corrected, and edited the paper, and Carl Kriger reviewed and edited the paper; all authors approved the final version.

References

[1] Hammoodi, S. J. Flayyih, K. S. Hamad, A. R. (2020) 'Design and implementation speed control system of DC Motor based on PID control and Matlab Simulink', International Journal of Power Electronics and Drive Systems (IJPEDS), 11(1), pp. 127-134, DOI:10.11591/ijpeds.v11.i1.

[2] Eze, P. C. Ugoh, A. C. (2021) 'Positioning Control of DC Servomotor-Based Antenna Using PID Tuned Compensator', Journal of Engineering Sciences, 8(1), pp. E9-E16. DOI: 10.21272/jes.2021.8(1). e2

[3] Venugopal, K Manoharan, S. K. Megalingam, R. K. (2022) 'Position Estimation in DC Motor using Strain Gauge Closed Loop Control for Robotic Grippers', 2022 IEEE IAS Global Conference on Emerging Technologies (GlobConET), pp. 355-360, DOI:10.1109/GlobConET53749.2022.9872356.

[4] Mezher, L. (2019) 'Position Control for Dynamic DC Motor with Robust PID Controller using MATLAB', International Journal of Advanced Trends in Computer Science and Engineering, 8(3), pp. 936-942. DOI: 10.30534/ijatcse/2019/92832019.

[5] Iswanto. Raharja, N. M. Ma'arif, A. Ramadhan, Y. Rosyady, P. A. (2021) 'Pole Placement Based State Feedback for DC Motor Position Control', Annual Conference on Science and Technology Research (ACOSTER) 2020. DOI: 10.1088/1742-6596/1783/1/012057.

[6] Pal, D. (2016) 'Modeling, Analysis and Design of a DC Motor based on State Space Approach', International Journal of Engineering Research & Technology (IJERT), 5(2), pp. 293-296. DOI: 10.17577/ijertv5is020332.

[7] Ahmad, M. Khan, A. Raza, M, A. Ullah, S. (2018). 'A Study of State Feedback controllers for Pole Placement', 5th International Multi-Topic ICT Conference (IMTIC). DOI: 10.1109/IMTIC.2018.8467276.

[8] Venugopal, K Manoharan, S. K. Megalingam, R. K. (2022) 'Position Estimation in DC Motor using Strain Gauge Closed Loop Control for Robotic Grippers', 2022 IEEE IAS Global Conference on Emerging Technologies (GlobConET), pp. 355-360, DOI:10.1109/GlobConET53749.2022.9872356.

[9] Shafi, S. Hamid, P. S. Nahvi, S. A. (2023) 'Observer Based State Feedback Controller Design of a DC Servo Motor Using Identified Motor Model: An Experimental Study', 2023 International Conference on Power, Instrumentation, Energy and Control (PIECON), pp. 1-6, DOI:10.1109/PIECON56912.2023.10085764.

[10] Nur, A. A. Rahim, A. H. M. A. (2022) 'State feedback for DC Motor Position Control Based on Pole Position'.

[11] Chowdhury, A. Debnath, A (2021) 'DC Motor Position Control using State Space Technique' International Journal of Advances in Science and Technology (IJAST).

[12] Love, K. Kriger, C. Nomzamo, T. (2024) 'Design and Simulation of Full State Feedback Controller for DC Motor', International Journal of Electrical Engineering and Applied Sciences, 7(1).

[13] Martin. Margana, D. B. Habinuddin, E. (2023) 'Fuzzy Logic Controller Implementation for Motor DC Control Position with Real-Time Operating System', International Journal of Information System & Technology, 6(5), pp. 654-663. DOI: 10.30645/ijistech.v6i5.283.

[14] Ma'arif, A. Cakan, A. (2021) 'Simulation and Arduino Implementation of DC Motor Control Using Sliding Mode Controller', Journal of Robotics and Control (JRC), 2(6), p.p 582-587, DOI: 10.18196/jrc.26140.

[15] Thein, M. M. Lwin, K. S. (2019) 'Implementation of DC Motor Controlling Techniques', International Journal of Science, Engineering and Technology Research (IJSETR), 8(7), pp. 345-251.

[16] Moulahcene, F. Laib, H. Merazga, A. (2022) ‘Angular Position Control of DC Gear-Motor Using PID Controllers for Robotic Arm’, International Conference on Electrical, Computer and Energy Technologies (ICECET 2022), pp. 1-6, DOI:10.1109/ICECET55527.2022.9872821.

[17] Moradi, S. Y. Saeedi, E. (2016) ‘Controlling DC Motor Position, Using PID Controller Made by PIC Microcontroller’, ZANCO Journal of Pure and Applied Sciences, p.p 29-36, DOI: 10.21271/zjpas.v28i2.807.

[18] Saaf, M. (2021) ‘Real Time DC Motor Position Control Using PID Controller in LabVIEW’, Journal of Robotics and Control (JRC), 2(5), pp. 342-348, DOI:10.18196/jrc.25104.

[19] Ma’arif, A. Setiawan, N. R. (2021) ‘Control of DC Motor Using Integral State Feedback and Comparison with PID: Simulation and Arduino Implementation’, Journal of Robotics and Control (JRC), 2(5), pp. 456-461. DOI: 10.181196/jrc.25122.

[20] Ohemu, M. Onuche, A. D. F. Kachalla, I. A. Zuleihat, Z. K. (2022) ‘State-feedback Control for a DC motor Using Pole Placement Technique’, International Conference on Electrical Engineering Applications (ICEEA 2020).

[21] Tshemese-Mvandaba, N. Mnguni, M. E.S. (2023) ‘Decentralized proportional-integral controller based on dynamic decoupling technique using Beckhoff TwinCAT-3.1. International Journal of Electrical and Computer Engineering (IJECE)’, vol. 13, no. 3, pp. 2721-2733, ISSN: 2088-8708, DOI: 10.11591/ijece.v13i3.pp2721-2733