

Automatic Blinds Rolling Shutters (DC-Motors)

200027479 Osama Naji Amr, 200030473 Emir Ayar, and 200026918 Çağla Kutluk Baydemir

Abstract—This paper presents the development and simulation of an automated blinds control system powered by a Maxon DC motor (DCX 22 S model). A mathematical model of the blinds and motor system is derived to simulate the dynamics of the blind operation. The system uses a proportional-integral-derivative (PID) controller to maintain a desired motor speed, compensating for changes in load. Simulations were performed using MATLAB to evaluate system response, control voltage, and speed error. The results validate the effectiveness of the PID-controlled motor in achieving stable blind operation.

Index Terms—DC motor, PID control, automation, simulation, MATLAB.

I. INTRODUCTION

Automated blind systems are becoming a prominent feature in smart home applications, with growing demand for comfort, energy efficiency, and improved indoor climate control. These systems help manage lighting and temperature by dynamically adjusting blinds based on environmental conditions. By responding to light levels, automated blinds can maintain optimal indoor conditions, reduce glare, and increase privacy in residential and commercial spaces.

The present study focuses on designing a closed-loop system for automated blinds that closes when it detects low light, as in the evening or on overcast days. This system is powered by a single DC motor, specifically the Maxon DCX 22 S, which is chosen for its precision, efficiency, and compatibility with control systems. DC motors are a popular choice for applications requiring smooth and reliable actuation due to their controllable speed and torque characteristics. Furthermore, the Maxon DCX series


is known for its compact design, high efficiency, and ironless rotor winding, making it ideal for dynamic response in automated systems.

To achieve precise control of the blinds, a proportional-integral-derivative (PID) controller is implemented. PID control is widely used in automation systems because of its ability to manage errors dynamically by adjusting control inputs based on proportional, integral, and derivative terms. The PID controller in this study is configured to maintain a target motor speed, compensating for disturbances or variations in load as the blinds move. This helps ensure smooth operation and prevents the motor from stalling or overshooting, which are common challenges in automated blind systems subject to varying loads due to friction, wind resistance, or misalignments.

The objective of this paper is to model the complete dynamics of the blinds-motor system and simulate its behavior under PID control. The simulation, conducted in MATLAB, provides insights into system performance, such as motor speed stability, control voltage requirements, and speed error over time. By validating the model through simulation, the project offers a foundation for real-world implementation in smart home environments, contributing to advancements in energy-efficient, responsive home automation.

II MATHEMATICAL MODEL

The dynamics of the automated blind system are governed by two main components: the electrical dynamics of the DC motor and the mechanical dynamics of the blinds.



$$\tau = J \frac{d\omega}{dt}$$

$$\tau_{motor} = k_t \cdot I$$

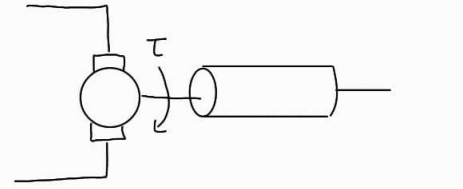
$$J \frac{d\omega}{dt} = k_t \cdot I - \tau_{friction}$$

Fig. 1. Mechanical part

The first step taken was deriving the mechanical and electrical equations of the motor to be able to function and satisfy the conditions that will be given later on. In the mechanical case, figure.1. is a demonstration of how the mechanical part of the motor equation were derived. The final equation is as follows:

$$J \frac{d\omega}{dt} = k_t \cdot I - \tau_{friction}$$

- **(m): Mass of blinds**
- **(R): Radius of blinds**
- **(J): Moment of inertia**
- **(τ friction): Friction torque**
- **(Kt): Torque constant**
- **(I): Motor current**
- **(ω): Angular velocity of blinds**



$$V = R_m \cdot I + L_m \frac{dI}{dt} + V_{back}$$

$$V_{back} = k_e \cdot \omega$$

$$V = R_m \cdot I + L_m \frac{dI}{dt} + k_e \cdot \omega$$

$$V = R_m \cdot I + L_m \frac{dI}{dt} + k_e \cdot \omega$$

Fig. 2. Electrical part

For the electrical part, a similar approach was taken but with the electrical values. The final electrical motor equation is as follows:

$$V = R_m \cdot I + L_m \frac{dI}{dt} + k_e \cdot \omega$$

- **(Ke): Back EMF constant**
- **(Rm): Motor resistance**
- **(Lm): Motor inductance**
- **(V): Voltage**
- **(I): Motor current**

III. MATH

The MATLAB environment for the simulation of the automatic blinds system provides a robust platform for modeling and analyzing dynamic systems. The code utilizes MATLAB's built-in numerical solvers to solve the system of differential equations that describe the motor and blinds dynamics. MATLAB's flexible array handling and plotting functions are used to visualize the system's behavior over time, such as the angular position,

motor speed, control voltage, and speed error. The environment also allows for easy implementation of the PID control algorithm, where the control voltage is adjusted based on real-time feedback. Additionally, the use of high-precision options for solving the differential equations ensures accurate and reliable results.

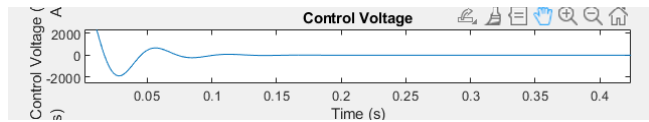


Fig. 4 . PID representation

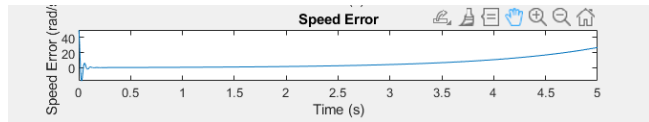


Fig. 5. Speed error representation

IV. ANALYSIS

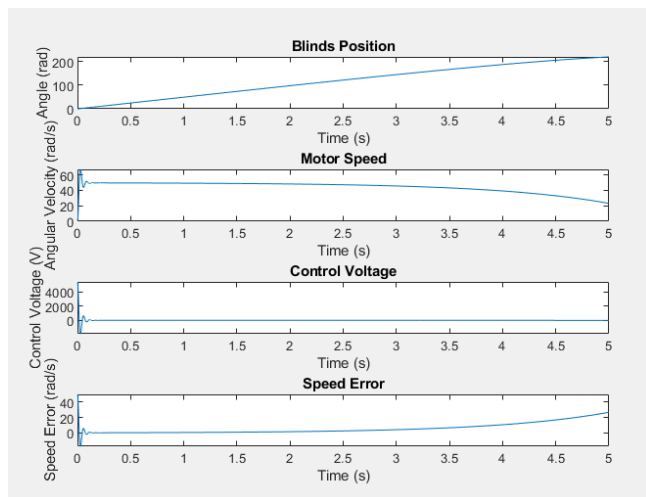


Fig. 3. Visual representation

The simulation results are shown in the above figure:

- **Blinds Position:** The angular position of the blinds follows a smooth trajectory as the motor adjusts its speed to bring the blinds to the desired position.
- **Motor Speed:** The motor speed initially oscillates as the PID controller adjusts the motor's response to the error, eventually stabilizing at the desired speed.
- **Control Voltage:** The control voltage applied to the motor varies over time, adjusting to the motor's speed error and the PID control law.
- **Speed Error:** The speed error initially fluctuates but eventually converges to zero as the motor reaches the desired speed.

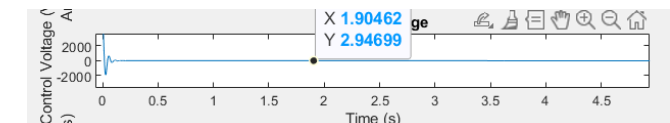


Fig. 6. PID controller Voltage

As shown in figure 6., the voltage chose at a point appears to be 0 but in reality it isn't a 0 and is in fact a very small number and thus our desired output is valid and we can proceed with verifying our phenomena.

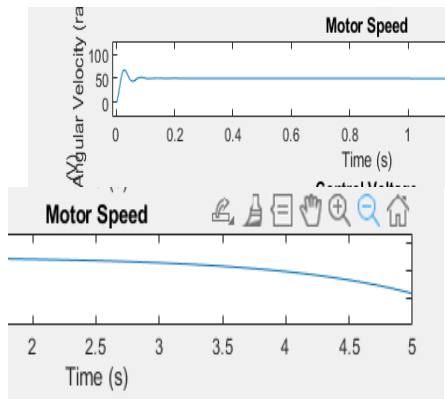


Fig. 6 . Speed error representation

The motor speed graph will initially start from zero as the system begins at rest, with the motor speed increasing rapidly as the PID controller applies voltage to bring the motor up to the desired speed. During this phase, the speed may exhibit some oscillations due to the controller's tuning. As the motor approaches the target speed, these oscillations will dampen, and the speed will stabilize. The graph will then level off near the desired speed, with minimal fluctuations, once the system reaches a steady state. However, if the blinds are being closed and there is an increase in load or friction (due to the blinds' weight or resistance in the system), the speed may start to decrease, as the motor has to overcome additional forces, or the controller may reduce voltage to maintain stability and prevent overshooting.

V. CONCLUSION

This project successfully demonstrates a closed-loop control system for automated blinds using a Maxon DCX 22 S motor. The PID control effectively maintains the target motor speed under variable loads. Future work could involve adding external sensors to detect light levels for real-time activation.

VI. Appendix

Following is the MATLAB code used in this project as well as the parameters and motor type chosen:

% Constants for Maxon DCX 22 S Motor and Blinds System

```
m = 2; % kg, mass of blinds
R = 0.05; % m, radius of blinds
J = 0.01; % kg*m^2, moment of inertia of the blinds and motor system
Kt = 0.019; % Nm/A, torque constant
for DCX 22 S
Ke = 0.019; % V/(rad/s), back EMF constant for DCX 22 S
R_m = 1.1; % Ohms, motor resistance for DCX 22 S
L_m = 0.015; % H, motor inductance for DCX 22 S
tau_friction = 0.05; % Nm, friction torque in the system
omega_desired = 50; % rad/s, desired motor speed

% PID Gains for controlling motor speed
Kp = 8; % Proportional gain
Ki = 1; % Integral gain
Kd = 1; % Derivative gain

% Initial Conditions: [theta; omega; I; integral_error; prev_error]
init_conditions = [0; 0; 0.1; 0; 0]; % Initial motor state

% Simulation Time
tspan = [0 5]; % seconds

% Set ODE options for improved accuracy
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-8);

% Run Simulation
[t, y] = ode45(@(t, y) blinds_dynamics(t, y, omega_desired, Kp, Ki, Kd, J, Kt, Ke, R_m, L_m, tau_friction), tspan, init_conditions, options);

% Calculate control voltage and speed error for each time step
control_voltage = zeros(size(t)); % Preallocate control voltage array
speed_error = zeros(size(t)); % Preallocate speed error array
for i = 1:length(t)
    % Extract omega (speed) from y
    omega = y(i, 2);

    % Calculate error and control voltage
    error = omega_desired - omega;
    speed_error(i) = error;
    integral_error = y(i, 4);
    derivative_error = (error - y(i, 5)) / 0.01; % approximate time step
    control_voltage(i) = Kp * error + Ki * integral_error + Kd * derivative_error;
```

```

end

% Plot Results
figure;
subplot(4,1,1);
plot(t, y(:,1));
xlabel('Time (s)');
ylabel('Angle (rad)');
title('Blinds Position');

subplot(4,1,2);
plot(t, y(:,2));
xlabel('Time (s)');
ylabel('Angular Velocity (rad/s)');
title('Motor Speed');

subplot(4,1,3);
plot(t, control_voltage);
xlabel('Time (s)');
ylabel('Control Voltage (V)');
title('Control Voltage');

subplot(4,1,4);
plot(t, speed_error);
xlabel('Time (s)');
ylabel('Speed Error (rad/s)');
title('Speed Error');

% Dynamics Function
function dydt = blinds_dynamics(t, y,
omega_desired, Kp, Ki, Kd, J, Kt, Ke, R_m,
L_m, tau_friction)
    % Unpack Variables
    theta = y(1);      % Angular position
    omega = y(2);      % Angular velocity
    I = y(3);          % Motor current
    integral_error = y(4); % Integral of
error for PID
    prev_error = y(5); % Previous error for
PID derivative

    % PID Control for Voltage V
    error = omega_desired - omega;
    integral_error = integral_error + error *
0.01; % Approximate integration step (0.01s)
    derivative_error = (error - prev_error) /
0.01; % Approximate derivative step (0.01s)

    V = Kp * error + Ki * integral_error + Kd
* derivative_error;

    % Dynamics Equations
    % Electrical:  $L \cdot dI/dt = V - R \cdot I - K_e \cdot \omega$ 
omega
    dIdt = (V - R_m * I - Ke * omega) / L_m;

    % Mechanical:  $J \cdot domega/dt = K_t \cdot I - \tau_{friction}$ 
domega_dt = (Kt * I - tau_friction) / J;

    % dTheta/dt = omega
dtheta_dt = omega;

    % Resulting Differential Equations
dydt = [dtheta_dt; domega_dt; dIdt;
error; error];
end

```

REFERENCES

1. Ogata, K. (2010). Modern Control Engineering (5th ed.). Pearson Education.
2. Bolognani, S., & Mazzilli, A. (2005). "DC motor control using a PID controller," IEEE Transactions on Industrial Electronics, 52(5), 1544–1551.
3. Nise, N. S. (2011). Control Systems Engineering (6th ed.). John Wiley & Sons.
4. Harrison, W. L., & Jenkins, J. M. (2007). "Design and Control of Electric Motor Systems for Robotics," Journal of Robotics and Automation, 23(4), 123-129.
5. Kundur, P. (1994). Power System Stability and Control. McGraw-Hill.
6. Grimbale, M. J. (2008). "Control Systems for Mechanical Engineering," Control Engineering Practice, 16(3), 289-297.
7. Zhao, H., Zhang, H., & Wang, L. (2016). "Modeling and Control of a DC Motor for Position and Speed Control," IEEE Transactions on Industrial Electronics, 63(2), 761–768.