FEDERAL UNIVERSITY OF TECHNOLOGY, AKURE
JOURNAL OF ENGINEERING AND ENGINEERING TECHNOLOGY

DESIGN AND IMPLEMENTATION OF A PID CONTROL FOR A DC MOTOR-DRIVEN BELT CONVEYOR SYSTEM

[1]Olubosede, O. A., [1]Adedeji, K. B.*, [2]Odeyemi, C. S. and [3]Shabangu, T. H.

[1]Department of Electrical and Electronics Engineering, Federal University of Technology, Akure, Ondo State, Nigeria
[2]Department of Computer Engineering, Federal University of Technology, Akure, Ondo State Nigeria
[3]Department of Electrical Engineering, Tshwane University of Technology, Pretoria South Africa
**Correspondence author e-mail: kbadedeji@futa.edu.ng*

Olubosede, O. A., Adedeji, K. B., Odeyemi, C. S. and Shabangu, T. H. (2025): Design and Implementation of a PID Control for a DC Motor-Driven Belt Conveyor System. *FUTA Journal of Engineering and Engineering Technology /19(2), 1-12*

Received Date: 05.08.25 **Accepted Date: 12.09.25**

Abstract
In this study, a Proportional-Integral-Derivative (PID) control system was designed and implemented on a DC motor-driven belt conveyor system. In addition, the effects of other control strategies, such as open-loop, Proportional (P), Proportional-Integral (PI), and the PID control on the loading conditions of a DC motor-driven belt conveyor system were investigated. The DC motor's speed control is critical to ensuring the conveyor's stability and efficiency. The effectiveness of the system was analysed using steady-state error, maximum overshoot, peak time, and rise time. In the open-loop configuration, the system exhibits high steady-state errors and slow responses, making it unsuitable for precise speed control. Proportional control provides a faster response than open-loop but a high steady-state error and relatively high overshoot. PI control balances steady-state error reduction with moderate overshoot, peak time, and rise time. PID control, however, provides the best performance with no steady-state error, fast rise and peak times, and moderate overshoot. In terms of the steady state error, the PID control has 100% improvement than the PI and P control respectively. Also, a 2% reduction in the maximum overshoot is observed when compared to the PI control and 41% reduction when compared to the P control. The comparative analysis demonstrates that PID control is the most effective strategy for optimising the DC motor belt conveyor system, offering a balanced improvement across all performance metrics.

Keywords: *belt conveyor, DC motor, microcontroller, overshoot, PID control*

Introduction

In the modern world, the development of machinery and robotic systems has reached a very advanced level of precision and automation. The industrial sector could undergo a transformation thanks to robotics and automated systems. The construction sector and other labour-intensive fields have benefited greatly from it (Davila *et al*., 2019). The rapid advancement in automated systems is accompanied by higher demands for efficiency, precision, and automatic error correction. As these systems become more complex, there is an increasing need for effective control mechanisms. The majority of electromechanical systems consist of a power supply system, a number of sensors, actuators, and embedded controllers like computers, microcontrollers, or programmable chips (Taghavi *et al*., 2020). Some of these have been utilized in belt conveyor systems. A belt conveyor system is an electromechanical system that moves materials from one place to another. It is made up of an electric motor-driven closed loop of flexible belt stretched across rollers.

These faults need to be detected, located and cleared in due time, not by the traditional way of receiving trouble calls from the affected customers but by use of advanced technological methods such as remote sensing and fault location technologies through the use of advance software to monitor, control and analyze real-time data from the network to quickly locate fault and isolate the faulty system.

Due of its affordability and versatility, belt conveyors are the most widely utilized powered conveyors (Kumar, 2023). The performance of a conveyor belt system is often measured by its ability to maintain a consistent speed and accurately position items at specific points along the belt. Belt conveyor systems enable the continuous movement of goods and materials and have found applications in many industrial settings such as mining (Szrek *et al*., 2022), food processing (González *et al*., 2023), bottling and canning (Nangare and Sonawane, 2022), electronics, and assembly lines (Varna and Abromavičius, 2022). In these applications, it is important to ensure smooth and accurate operation of conveyor belts. This requires robust control mechanisms.

When a belt conveyor system is powered by a motor, the type of load attached to the machine's axis determines the motor's speed and torque. The motor achieves a comparatively high speed and low torque when the load is light. A heavy load will cause the motor to run slower and produce more torque (Hammoodi *et al*., 2020). Controlling the belt's position and speed is essential in industrial conveyor belt systems, particularly in applications where accuracy is crucial, such as assembly lines and food processing. There is also the need to maintain the same speed under different loading conditions (Hammoodi *et al*., 2020). Different applications of belt conveyor systems have different requirements. For example, the conveyor at an airport baggage claim may not require as much precision as the heavy duty conveyor at an assembly plant. Traditional methods of control often struggle to meet the requirements of precision, constancy, and varying operating conditions, leading to inefficiencies such as inconsistent speed and misalignment (Somefun *et al*., 2021). The challenge is designing a control strategy that can effectively regulate the speed of the conveyor belt, keeping it constant under different loading conditions (Somefun *et al*., 2021). This study designs a motor-driven belt conveyor system and implements a PID control system to regulate the system stability under load conditions. Also, effect of load conditions on other control strategies was investigated. The goal is to achieve improved speed regulation, precise positioning, and enhanced overall system reliability, contributing to increased productivity and operational efficiency.

A PID control system is a closed loop system where the plant output (controlled variable) is continuously measured and compared with the desired output (He *et al*., 2017). The disparity between them gives the error (or offset). PID control employs three control actions, that is., proportional, integral and derivative. The P term applies an actuating signal that is proportional to the control output. This indicates a direct correlation between the actuating signal and the error size. The stronger the control action, the greater the error. The integral term applies a control action based on the cumulative sum of past errors, which helps to eliminate residual steady-state errors by adjusting the controller output until error free scenario is observed. The D term provides a predictive action that helps to limit oscillations and overshoot in the system by applying a control action proportional to the rate of change of the error. By addressing both current error and trends in error changes, these three control actions enable the PID controller to offer a balanced approach to control, resulting in accurate and reliable system control. Figure 1 displays a PID controller's block diagram representation. In Figure 1, Kp, K_i and K_D denote the proportional, integral and derivative gains respectively. The resulting actuating signal $U(t)$ after the error is expressed by Equation (1) (Kumar *et al*., 2020).

$$U(t) = K_C\left(e(t) + \frac{1}{T_i}\int e(t)dt + T_d\frac{de(t)}{dt}\right) \tag{1}$$

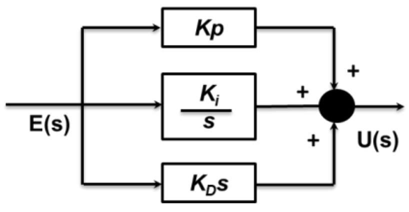


Figure 1: A conventional block of PID controller (Kumar *et al*., 2020)

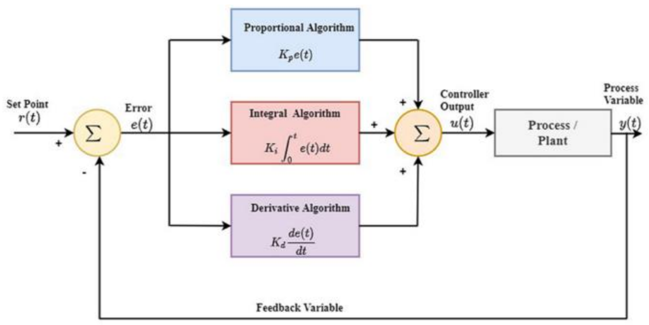


Figure 2: A PID-based process control (Borase et al., 2021)

where the integral and derivative times are T_i, and T_d, $e(t)$ is the error signal. The block diagram of a PID-based process control system is displayed in Figure 2.

Controller tuning is the process of choosing the controller parameters to satisfy specified performance requirements. Adjusting the PID controller's parameters—proportional gain (K_p), integral gain (K_i), and derivative gain (K_d)—to their ideal values in order to attain the system's intended performance is known as PID controller tuning. PID's prolonged tuning time is one of its drawbacks (Ogata, 2020; Maghfiroh *et al*., 2020). A PID controller can be tuned using a variety of techniques, such as the Cohen-Coon approach, Ziegler-Nichols method, adaptive tuning, and manual trial and error.

The Ziegler-Nichols method is a simple and effective PID controller tuning method and it is widely used especially when the plant transfer function is not known. However, it requires a time-consuming experiment and the initial results may need to be fine-tuned for optimal performance. Also, more advanced algorithms have been developed such as the fuzzy logic PID tuning (Madebo, 2025) and neural network PID tuning (El-Rifaie *et al*., 2025).

The purpose of a belt conveyor system is to move materials from one location to another. Figure 3 illustrates a simple belt conveyor. As can be observed, an electrical motor drives a loop of flexible material that is stretched between rollers. A typical belt conveyor system consists of a head pulley, tail pulley, belt, frame, and idler rollers. There are different types of belt conveyors including Roller Bed Belt Conveyor, Flat Belt Conveyor, Modular Belt Conveyor, Cleated Belt Conveyor, Curved Belt Conveyor, Incline/Decline Belt Conveyor, Sanitary Washdown Conveyor, Troughed Conveyors, and Magnetic Belt Conveyor (IQS Directory, 2024). The main factors to be taken into account while building a conveyor belt are the choice of motor and gearbox, belt speed, tension and take-up, material to be delivered, distance to be moved, and working conditions (IQS Directory, 2024).

Much study has been done on the design, simulation, and use of PID controllers for belt conveyor systems powered by DC motors. Chen and Wu (2011) presented a novel approach to PID controller design that relies on FPGA implementation, VHDL description, and tuning parameters for evolutionary algorithms. In this study, the FPGA is used to realise the PID controller's parameters after they have been optimised using a genetic algorithm. The top-down design approach is used to split the system's functional units throughout the realisation phase. The VHDL language is used to describe each system module. Lastly, Matlab/Simulink and DSP Builder are used to implement the controller's closed loop testing. The design method's flexibility, online self-tuning, high reliability, quick technical development cycle, fast running speed, and other attributes have been demonstrated by simulation results.

An enhanced PID controller for DC motor control was introduced by Ahmed *et al*. (2021). The proposed controller's performance was compared with that of the conventional PID controller. The proposed controller outperforms the PID, according to the results. The controller's characteristics included an 8% overshoot, a rise time of 0.07s, settling time of 0.2 and also a steady-state error of about 1.5% lapsing up to 2.5s response duration. The authors demonstrated that the PID controller has room for further development. Using LabVIEW 2011, Kumar *et al*. (2020) created a PID controller to regulate a DC motor's speed. The DC motor transfer function was computed in this work. The PID controller was also adjusted and put into use. In order to maintain a steady speed for a brushed DC

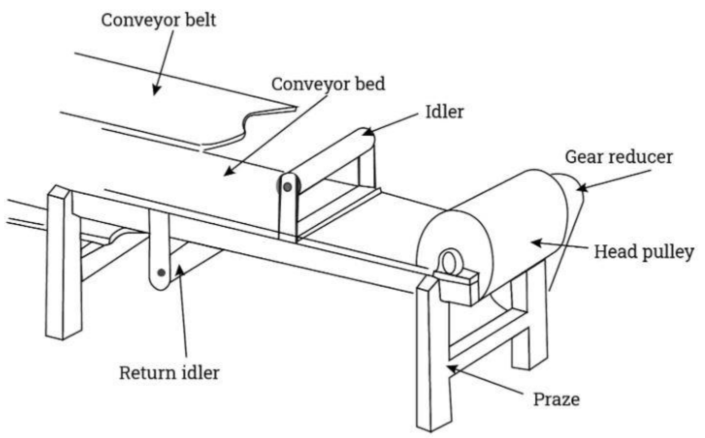


Figure 3: Simple conveyor components (IQS Directory, 2024)

motor in the face of load fluctuations (disturbances), Hammoodi *et al*. (2020) constructed and simulated a closed-loop speed control system. The authors achieved this by modelling PID control using MATLAB/Simulink. In this study, a chopper transistor circuit is used to independently excite a dc motor through the speed control circuit. Consequently, a chopper transistor circuit was used to supply a DC motor with a DC source voltage. The chopper circuit is made up of a single GTO-1 thyristor with a freewheeling diode and a complete control circuit. According to the simulation results, PID control has fast response. A robust belt conveyor system was created by Todkar *et al*. (2018) for application in the coal processing sector. Stresses on the pulley caused by belt tensions at the head side, tail/take up, and snub side were calculated in this study's conveyor design. The belt conveyor was modelled using a 3D CAD mode and designed in accordance with Indian standards (IS11592). Additionally, ANSYS software was used to perform finite element analysis while taking precise loading conditions into account. The belt conveyor is prone to failure, according to the findings of the stress study done to identify its most stressed components. Also, Balamurugan and Umarani (2020) built a digital control system that integrates a discrete PID controller with DC motor; the authors used CAD software to generate an exact 3D computer-aided design (CAD) model of the system. The dynamics of the system was determined and controller parameters are obtained by root locus method.

Methodology

Figure 4 displays the activity diagram for the study. The initial step is to design and construct the physical conveyor belt system. The Arduino Uno microcontroller was used in this study. The second step is to obtain the open loop response of the motor. The next step is to design and implement a Low Pass Filter (LPF) to remove the high frequency oscillations in the response. Then the proportional, proportional-integral, and proportional-integral-derivative control mechanism is implemented and the responses are obtained. Finally, the performances of the various control mechanisms are compared under load conditions. Plates 1 and 2 show the circuit diagram and the pictorial view of the system. The circuit was designed and simulated on Fritzing software. Thereafter, the designed was transferred to a breadboard.

As shown in Plate 1, the Arduino Uno was connected to an L298N motor driver, which powers a DC motor. The motor gets its power from a 12 V DC source, and the Arduino sends control signals to the motor driver to adjust speed and direction. An optical slot sensor sits near the motor, measuring its speed. It does this by detecting interruptions in light as a part of the motor spins. The sensor sends data back to the Arduino, which then displays the speed on a 16 × 2 LCD. The LCD uses an I2C interface to reduce the number of connections. Plate 2 shows the developed system. After the belt conveyor system have been developed, the other steps shown in Figure 4 follows.

Gearbox and Motor Selection

Determining the effective pulling force needed for the conveyor will help with the motor selection. Equation (2) provides the effective pulling force (F_U) for a basic horizontal conveyor.

$$F_U = \mu_R \times g(m + m_b + m_g) \qquad (2)$$

where m is the mass of items transported over the entire conveyor's length, m_b is the mass of the belt, m_R is the mass of all rotating rollers less the mass of the drive roller, μ_R is the friction coefficient when going over a roller, and g is the acceleration caused by gravity. Equation (3) (IQS Directory, 2024) provides the effective pulling force for a system on an incline.

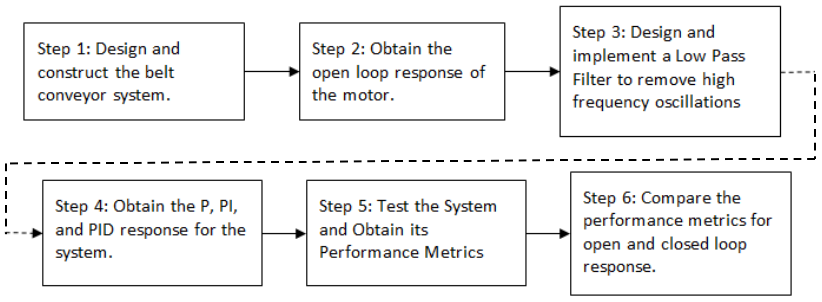

Step 1: Design and construct the belt conveyor system.
Step 2: Obtain the open loop response of the motor.
Step 3: Design and implement a Low Pass Filter to remove high frequency oscillations
Step 4: Obtain the P, PI, and PID response for the system.
Step 5: Test the System and Obtain its Performance Metrics
Step 6: Compare the performance metrics for open and closed loop response.

Figure 4: Activity flow diagram for the study

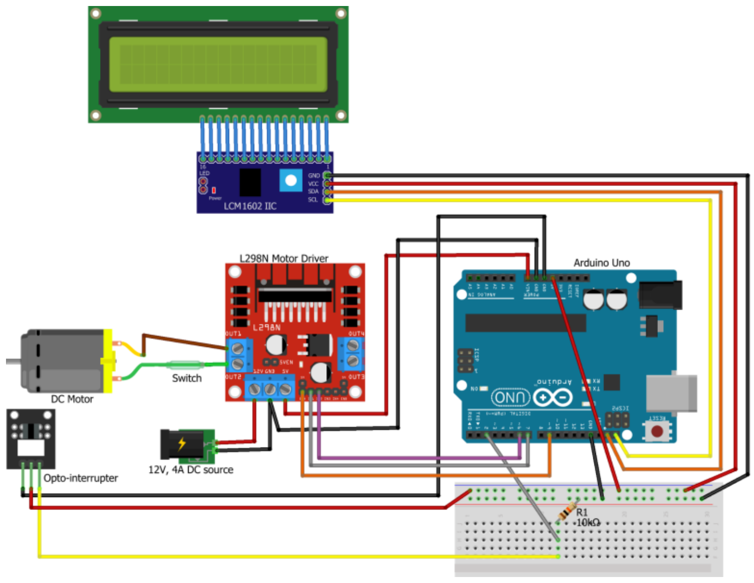
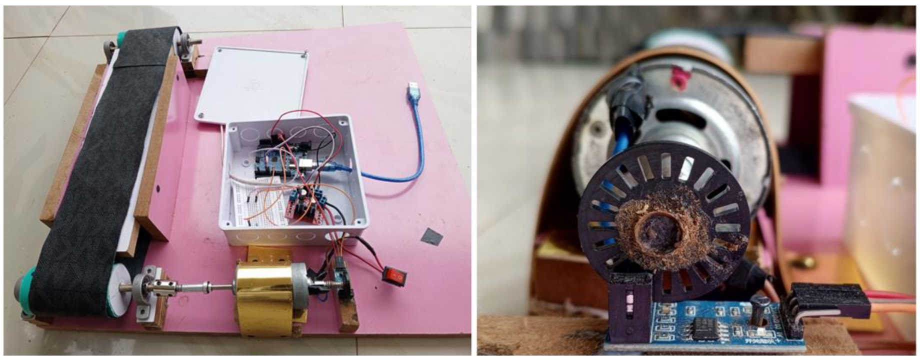


Plate 1: Circuit diagram of the motor-driven belt conveyor system



(a) (b)

Plate 2: Pictorial view (a) conveyor belt system (b) close-up of the speed measurement mechanism

$$F_U = \mu_g \times g \times (m + m_b + m_g) + (g \times m \times sin(a)) \quad (3)$$

where a is the angle of inclination.

The Conveyor Speed

Using Equation (4), one may assess a conveyor's speed.

$$V_c = D \times F \quad (4)$$

where V_c is the speed of the conveyor belt, D is the diameter of the drive pulley, and F is the revolutions of the drive pulley per second.

Determination of Open Loop Response of the Motor and Speed Calculation

The Arduino Uno was programmed with Arduino IDE. The speed was calculated using the optical slot sensor and the 20 grid speed counting disk. Since the speed counting disk does not physically touch the sensor, there was minimal wear and tear, ensuring longevity and reliability. In the Arduino, an interrupt was attached to the sensor via the "attach Interrupt" function, which triggers every time the optical sensor detects a rising edge (when an object like a hole or slot passes through the sensor). Each time the sensor detects this event, it sends a counter variable. This counter stores the number of interruptions that have occurred. The 20 grid speed counting disk is attached to the shaft of the motor and it is positioned such that it fits into the slot of the optical slot sensor. In this arrangement, one complete revolution of the motor shaft produced 20 pulses from the sensor pin.

To measure the speed of rotation using the Arduino, the time in microseconds (for very high precision) is logged on every loop pass. The velocity is estimated using Equation (5).

$$V = \left(\frac{c}{20}\right) \times \frac{1}{\Delta T} \quad (5)$$

where V is the velocity, c is the number of pulses detected since the last reading. The pulses per revolution (determined by the number of slots in the grid speed counting disk) is taken as 20, and ΔT is the time interval between the current and previous speed calculation. The resulting speed in rotations per second, is then converted to rotations per minute.

Designing the Low Pass Filter (LPF)

Since Infinite Impulse Response (IIR) filters typically require fewer coefficients to perform equivalent filtering operations (Soltani *et al*., 2024; Yadav *et al*., 2025), IIR LPFs were constructed in this study. Additionally, it uses less memory and operates more quickly (Kumar et al., 2024). The initial speed response graph derived from this study is coarse and there is a high frequency oscillation between values. There is a need to filter the response to make it smoother. A LPF is required to attenuate the strength of the high frequency component. The transfer function of the designed filter is shown in Equation (6) (Roonizi, 2024).

$$H(z) = \frac{0.0728z + 0.0728}{z - 0.854} \qquad (6)$$

The discrete update equation for the designed LPF is shown in Equation (7).

$$v\_filt[n] = 0.854 * v\_filt[n-1] + 0.0728 * v[n] + 0.0728 * v[n-1] \qquad (7)$$

where *v_filt[n]* is the filtered velocity for the n th iteration, *v_filt[n - 1]* is the filtered velocity for the (n - 1)th iteration, *v[n]* is the unfiltered velocity for the n th iteration, and *v[n-1]* is the unfiltered velocity for the (n - 1)th iteration.

Implementing a Proportional Control

Proportional Control is a simple and effective control strategy employed in many different applications, including position control, temperature regulation, and motor speed control. The error, or the discrepancy between a measured process variable and a desired set point, is directly proportional to the controller's output in P control. Equation (8) expresses the control output for a P control system.

$$u(t) = K_p \cdot e(t) \qquad (8)$$

Implementing Proportional-Integral Control

A PI control was also implemented in this study. PI control is a widely used control strategy in industrial applications for systems requiring precise regulation. A PI controller helps in the situation where proportional control is necessary. Speeding up the settling and integral control is necessary to reduce the error that is constant over time (Sayed *et al*., 2024). It combines the benefits of proportional control with an integral action to eliminate steady-state error. The integral component addresses accumulated error over time. It integrates the error to eliminate steady-state error, improving accuracy. The control output for PI control is given in Equation (9).

$$u(t) = K_p \cdot e(t) + K_i \cdot \int e(t)dt \qquad (9)$$

The *integral_error* variable accumulates the error over time. By summing the error multiplied by the elapsed time (the time since the last update), the integral term helps account for the accumulated error that has occurred over time. The *elapsedTime* variable stores the time interval since the last control update. Multiplying the error by the elapsed time gives the integral term's contribution based on how long the error has persisted. The integral term is essential in eliminating steady-state error, helping to ensure the system reaches and maintains the target speed.

Implementing PID Control

A PID control is the major focus of this study. The derivative component of PID control uses the error's rate of change to forecast future errors. It improves stability and lessens overshoot by dampening the system reaction. The control output *u*(*t*) for a PID controller is expressed in Equation (10).

$$u(t) = Kp \cdot e(t) + Ki \cdot \int e(t)dt + Kd \cdot \frac{de(t)}{dt} \qquad (10)$$

where *Kd* is the derivative gain. The error value from the previous iteration of the control loop is stored. It is then used to compute how much the error has changed over time. So the derivative error is the rate of change of the error over time. It is obtained by subtracting the previous error from the current error and dividing by the time that has elapsed. This helps predict the future behaviour of the error by examining how quickly it is changing. If the error is increasing rapidly, the derivative term will provide a larger corrective action to counteract this trend. The load response can then be obtained. Finally, the effectiveness of the different control techniques using was analysed and compared based on the steady state error, maximum overshoot, peak and rise times.

Results and Discussion

Open Loop Response of the Motor

Figure 5 shows the motor's open loop response. The motor's speed increases sharply at the beginning, indicating a rapid response to the applied voltage. The rise time is short, showing that the motor reacts quickly. It was observed that there are high frequency bounces between levels due to the discrete measurements. To alleviate this effect, a LPF was applied to smoothen the graph.

Low Pass Filter Response

After the implementation of the LPF, the response of the system becomes much smoother as illustrated in Figure 6. The unfiltered response exhibited significant oscillations and noise, particularly after the motor speed stabilised. The filtered response demonstrates a much smoother curve, with the oscillations from the raw speed data significantly reduced. The filtering process has effectively minimised the noise, providing a cleaner representation of the motor's speed. The filtered data better represents the actual motor performance since it eliminates high-frequency noise and provides a clearer picture of the motor's speed over time. Using filtered data would lead to more stable and reliable motor control.

Proportional Control Response

The response with proportional control with the set point set to 600 and the proportional gain set to 0.72 is shown in Figure 7. It can be observed that the system experiences oscillations about a steady value. The average steady-state value is 448, which gives a steady–state error of 152 (25% error).

The PI Control Response

The response of the PI control with the set point set to 600, the proportional gain to 0.72, and the integral gain to 1.6, is shown in Figure 8. It can be observed that the oscillations are reduced in amplitude and frequency compared to the proportional control response.

The PID Control Response

The response of the PID control with the set point set to 600, the proportional gain to 0.72, the integral gain to 1.6, and derivative gain to 0.05, is shown in Figure 9. The oscillations about the steady state value are much reduced and the overall trend can be easily seen. The rise time and overshoots are also reduced.

Load Response

The response of the system when it is loaded was also investigated. Figure 10 shows the system response to the load. The speed and actuating signal u(t) are plotted with time as a load is added and later removed. During the initial response (0 - 500 ms), the system starts by attempting to reach the set point of 600 rpm. There is an initial oscillation in the speed as the system stabilises. The actuating signal of the controller begins at a very high value of over 1,000,000 when the speed is at 0. This is typical behaviour of a PID controller as it is reacting aggressively to the initial large error (the difference between the desired speed of 600 rpm and the actual speed of 0 rpm). The actual applied value of the actuating signal is capped in the code to a maximum of 255 since that is the highest PWM value the Arduino can deliver.

Under steady-state operation (500 - 4000 ms), before the load was added, the system settled at 600 rpm, showing only minor oscillations. This suggests that the PID controller has brought the system to a fairly stable operation with minimal steady-state error. The actuating signal also stabilises at a value of 95 during this period, indicating that the control effort has also stabilised once the system is near the set point.

When the load was added (~4200 ms), a significant drop in speed was observed, indicating that the added load causes a large disturbance in the system. The speed initially dropped to below 500 rpm as the system reacts to the sudden change. In response to the increased error, the actuating signal spiked upward to counteract the speed drop, applying more power to the motor to bring the system back to the set point. Considering the recovery after load addition (~4200 - 5500 ms), the system gradually recovers from the disturbance. The actuating signal decreases correspondingly as the PID controller adjusts to reduce the power supplied to the motor.

When a new steady-state was reached (~5500 - 8500 ms), the system settles into a new steady-state balance where the speed is still 600 rpm but the steady actuating signal is increased from 95 to 184. The increase is what is required to keep the speed steady at 600 rpm while driving the load. When the load was removed (~8500 ms), the system experiences another disturbance. The speed increased briefly above the set point, but the system quickly returned to 600 rpm. The actuating signal shows a dip as the PID controller reduces the control effort to compensate for the sudden drop in load. During the final steady-state (9000 ms and beyond), after the load was removed, the system returns to initial steady-state operation.

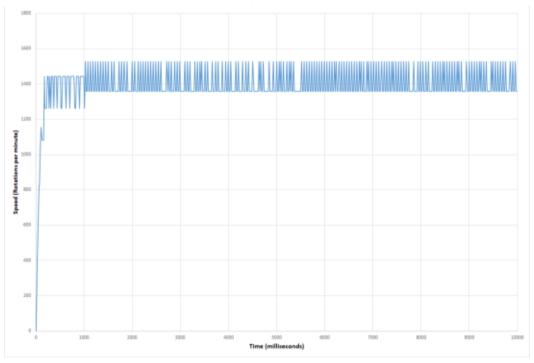


Figure 5: Open loop response of the motor

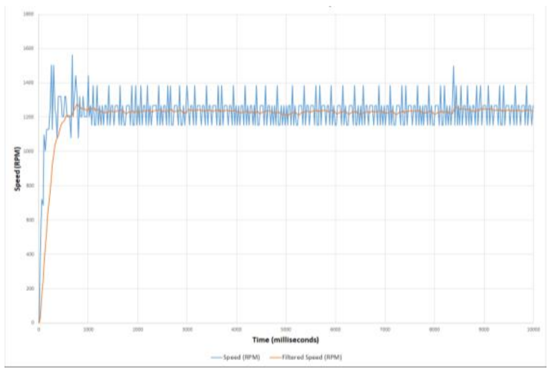


Figure 6: Filtered and unfiltered response of the motor

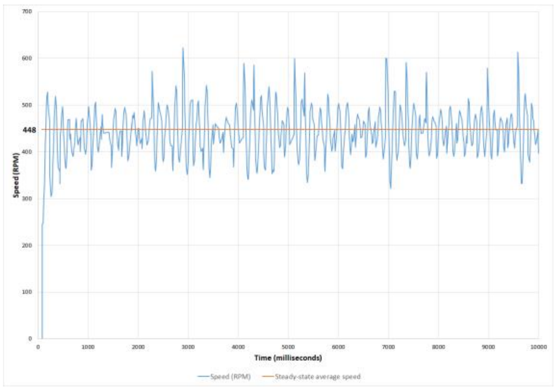


Figure 7: Response with P control

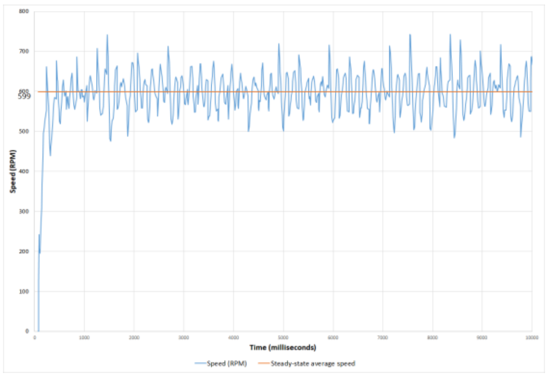


Figure 8: Response with PI control

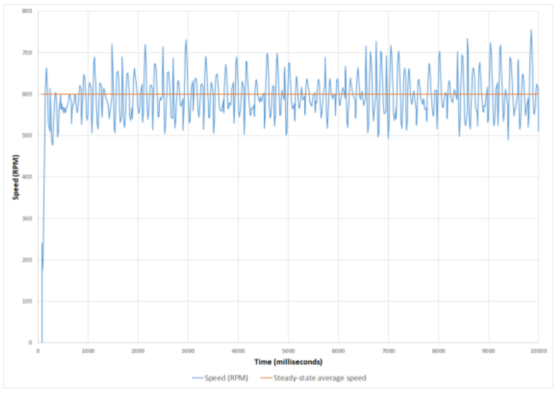


Figure 9: Response with PID control

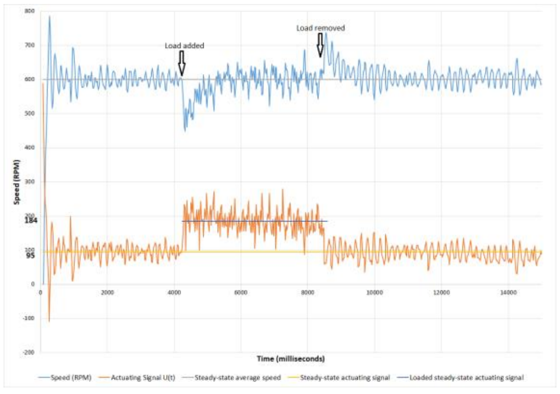


Figure 10: Load response with PID control

Overall, the system performs reasonably well, but improvements in tuning (e.g., fine-tuning PID parameters or adding more advanced filtering) could enhance its response to load changes and reduce oscillations.

Comparing the Performance of the System

Table 1 summarizes the performance of the different control strategies using the set point values for the proportional, integral, and derivative gain. A smaller steady-state error signifies better accuracy and performance, while a larger error indicates that the system may not reach the desired output precisely. A smaller steady-state error is preferred, indicating that the system is accurately tracking the desired input. The maximum overshoot reflects how well the system can control or limit oscillations before reaching a steady state. High overshoot may indicate an underdamped system, while low or no overshoot suggests better control and stability. A shorter peak time indicates that the system reaches its maximum output quickly, while a longer peak time suggests a slower response. It is determined by the system's damping ratio and natural frequency. The rise time of a control system is the time the system's output rise from a specified low percentage to a specified high percentage of its final steady-state value when subjected to a step input.

As can be observed in Table 1, for the steady state error, no target value is provided as there is no feedback for the open loop. In the open-loop control

systems, steady-state error typically depends on system dynamics and external disturbances. For the P Control, there is a 25% steady-state error, indicating that proportional control alone cannot fully eliminate the steady-state error. The PI control action minimizes the steady-state error to 0.17%, as the integral action compensates for the error over time. Additionally, the PID control action is the most accurate in maintaining the intended set point because it totally removes the steady-state error (0%).

Considering the maximum overshoot, the open loop has a minimal overshoot of 3.75%, which is due to the system's natural dynamics. The P control action, on the other hand has the highest overshoot of 17.8%, indicating that proportional control leads to more aggressive responses. The PI control action exhibits lower overshoot (10.3%) since the integral term helps control the error. while PID control reduces overshoot to 10.5% which is comparable to PI due to system dynamics. The PID control also reduces the oscillations in the response.
From the analysis of the peak time, the open loop control has the longest peak time (772 milliseconds), showing the slow response of the system without feedback control. The P control significantly reduces the peak time to 188 milliseconds, resulting in a faster response. The PI control slows down slightly compared to P control with a peak time of 240 milliseconds, due to the effect of the integral term. The PID control, on the other hand has a peak time of 169 milliseconds, longer than P and PI controls but faster than the open-loop system. The derivative action helps anticipate the error, speeding up the response. For the rise time as shown in Table 1, the open loop has a moderate rise time of 310 milliseconds. For the P control, the rise time decreases to 148 milliseconds, improving responsiveness. Considering the PI Control, the rise time increases to 207 ms due to the integral component which introduces a bit of delay. Meanwhile, the PID control achieves the fastest rise time (136 ms), as the derivative term provides a damping effect that helps the system respond quickly without excessive overshoot.

Conclusion

This study demonstrates the effectiveness of PID control in managing the speed and stability of a conveyor belt system. By integrating a PID controller with a DC motor and utilising the Arduino Uno as the control platform, the system was able to achieve precise regulation of conveyor speed with zero steady-state error, even in spite of disruptions and load fluctuations. Throughout the study, the key objectives of minimising the steady-state error, reducing overshoot, and ensuring fast response times were met through careful tuning of the PID gains. The system's performance was evaluated based on critical performance indices, such as rise time, peak time, and steady-state error. The results revealed that the PID controller provided significant improvements over other control methods like proportional and PI control. The results presented show that open loop control lacks feedback, leading to large steady-state errors and slow responses. Proportional control provides a faster response than open-loop but a high steady-state error and relatively high overshoot. PI control balances steady-state error reduction with moderate overshoot, peak time, and rise time, making it a good option. PID control offers the best performance with no steady-state error, fastest rise and peak times, and moderate

Table 1: Performance comparison of open loop, P, PI, and PID control for set point 600 rpm

Performance Indices	Open Loop	P Control	PI Control	PID Control
Steady-state error	-	25%	0.17%	0%
Maximum overshoot	3.75%	17.8%	10.3%	10.5%
Peak time (Milliseconds)	772	188	240	169
Rise time (Milliseconds)	310	148	207	136
Kp = 0.72, Ki = 1.6, Kd = 0.05, Set point = 600 rpm				

overshoot. For the belt conveyor system, the PID control is the optimal choice due to its ability to eliminate steady-state error, provide a fast response (short rise and peak times), and keep overshoot within acceptable limits.

The following points were noticed during the operation of the belt conveyor system;

i. **Damping effect of the load**: The addition of load influences the oscillatory behaviour of the system by minimising the amplitude and frequency of oscillations in the speed of the system. In control systems, damping is a critical factor that affects system stability and responsiveness. Loading increases this factor. Based on this, more controller parameter adjustment can be used to increase the basic system's damping factor.

ii. **Load handling**: The system can recover from load disturbances but experiences notable speed drops and overshoots when the load was added or removed. This indicates that the PID controller is effective but could be fine-tuned to minimise the impact of such disturbances.

iii. **System Stability**: Once the load was applied, the system becomes more stable, as the additional mass dampens any rapid fluctuations in speed. However, while the load increases stability, it may also cause a longer settling time, where the system takes longer to reach steady state after a disturbance. This trade-off between stability and response time is typical of systems with significant damping.

References

Ahmed, M., Tahir, N. M., Zimit, A. Y., Idi, M., Abubakar, K. A. and Jalo, S.A. (2021). Improved PID controller for DC motor control. *IOP Conference Series: Materials Science and Engineering,* 1052(1), 1-9.

Balamurugan, S. and Umarani, A. (2020). Study of discrete PID controller for DC motor speed control using MATLAB. *International Conference on Computing and Information Technology,* Tabuk, Saudi Arabia, 1-6.

Borase, R.P., Maghade, D.K., Sondkar, S.Y. and Pawar, S.N. (2021). A review of PID control, tuning methods and applications. *International Journal of Dynamics and Control,* 9, 818–827.

Chen, Y. and Wu, Q. (2011). Design and implementation of PID controller based on FPGA and genetic algorithm. *Proceedings of the International Conference on Electronics and Optoelectronics*, 29-31 July, Dalian, China, V4-308–V4-311.

Davila, D. J. M., Oyedele, L., Ajayi, A., Akanbi, L., Akinade, O., Bilal, M. and Owolabi, H. (2019). Robotics and automated systems in construction: Understanding industry-specific challenges for adoption. *Journal of Building Engineering*, 26, 1-11.

El-Rifaie, A.M., Abid, S., Ginidi, A.R. and Shaheen, A.M. (2025). Fractional order PID controller based-neural network algorithm for LFC in multi-area power systems. *Engineering Reports*, *7*(2), 1-14.

González, J.J.D., Jiménez, M.Y.O., Vega, J.E.P. and Caro, E.O.M. (2023). Conveyor belt for cleaning, sanitizing and drying unpackaged food. *Visión electrónica*, 17(2), 284-292.

Hammoodi S. J., Flayyih K. S. and 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 System,* 11(1), 127-134.

He, X., Cui, T., Zhang, D., Wei, J., Wang, M., Yu, Y., Liu, Q., Yan, B., Zhao, D. and Yang, L. (2017). Development of an electric-driven control system for a precision planter based on a closed-loop PID algorithm. *Computers and Electronics in Agriculture*, 136, 184-192.

IQS Directory. (2024). *Belt Conveyors.* Online. Available from: https://www.iqsdirectory.com/articles/conveyors/belt-conveyors.html, [Accessed 1/10/2024].

Kumar, N. (2023). Fundamentals of conveyors. In *Transporting operations of food materials within food factories* (pp. 221-251). Woodhead Publishing.

Kumar, D., Hussain, M., Tyagi, S. V., Gupta, R. and Salim, P. (2020). LabVIEW based speed control of DC motor using PID controller. *International Journal of Electrical and Electronics Research,* 3(2), 293-298.

Kumar, V., Arya, M., Kumar, A. and Jhariya, D.K. (2024). Design and comparison between IIR Butterworth and Chebyshev digital filters using MATLAB. In: *Proceedings of the IEEE 4th International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies*, 11-12 January, India, 1-7.

Madebo, N.W. (2025). Enhancing Intelligent Control Strategies for UAVs: A Comparative Analysis of Fuzzy Logic, Fuzzy PID, and GA-Optimized Fuzzy PID Controllers. *IEEE* 13, 16548-16563.

Maghfiroh, H., Nizam, M., and Praptodiyono, S. (2020). PID optimal control to reduce energy consumption in DC-drive system. *International Journal of Power Electronics and Drive Systems*, 11(4), 2165–2172.

Nangare, V.A. and Sonawane, P.R. (2022). Design, Analysis and Weight Optimization of Roller Conveyor System by using Glass Fiber Composite Material. *International Journal for Research in Applied Science and Engineering Technology*, 10, 1681-1687.

Ogata, K. (2010). PID controllers and modified PID controllers. In: Ogata, K. Modern Control Engineering. (5th ed., pp. 567-647), Prentice Hall.

Roonizi, A.K. (2024). Digital IIR filters: Effective in edge preservation? *Signal Processing*, 221, 109492.

Sayed, K., El-Zohri, H.H., Ahmed, A. and Khamies, M. (2024). Application of tilt integral derivative for efficient speed control and operation of BLDC motor drive for electric vehicles. *Fractal and Fractional*, *8*(1), 1-25.

Soltani, S., Pakniat, H. and Yasrebi, N. (2024). A wide stopband, high selectivity microstrip low-pass filter for wireless communications. *AEU-International Journal of Electronics and Communications*, 185(12), 155443.

Somefun, O. A., Akingbade, K., and Dahunsi, F. (2021). The dilemma of PID tuning. *Annual Reviews in Control*, 52, 65–74.

Szrek, J., Jakubiak, J. and Zimroz, R. (2022). A mobile robot-based system for automatic inspection of belt conveyors in mining industry. *Energies*, 15(1), 1-16.

Taghavi, N., Luecke, G., and Jeffery, N. (2020). A technical review on development of an advanced electromechanical system. *Computers,* 9(1), 1-16.

Todkar, S., Ramgir, M., and Tathwade, J. R. (2018). Design of belt conveyor system. *International Journal of Science, Engineering and Technology Research*, 7(7), 458-462.

Varna, D. and Abromavičius, V. (2022). A system for a real-time electronic component detection and classification on a conveyor belt. *Applied Sciences*, 12(11), 1-17.

Yadav, N.K., Dhawan, A., Tiwari, M. and Jha, S.K. (2025). A state-of-the-art survey on noise removal in a non-stationary signal using adaptive finite impulse response filtering: challenges, techniques, and applications. *International Journal of Systems Science*, 56(4), 885-918.