

Hardware-In-The-Loop Pid-Based Speed Regulation of A Dc Motor Under Variable Mechanical Load

Hamand Srekand¹, Suman T², Dr. Saravanakumar R³

¹Department Of Electronics And Instrumentation Engineering
Vellore Institute Of Technology, Vellore

²Department Of Electronics And Instrumentation Engineering Vellore Institute Of Technology, Vellore

³Department Of Control And Automation Vellore Institute Of Technology, Vellore

ABSTRACT- This study presents a Hardware-in-the-Loop (HIL) closed-loop speed regulation of a physical DC motor using a LabVIEW-based PID controller. In DC motors, maintaining the desired speed under changing mechanical loading is an important control problem in electromechanical systems, where variations in load torque result in speed deviations and transient disturbances. To overcome these issues, mathematical modelling and software simulation are useful for controller development; experimental evaluation with a hardware setup is necessary to account for practical effects associated with motor dynamics, sensing, actuation, and mechanical loading. An optical quadrature encoder provides real-time shaft speed feedback, and a variable friction brake is employed to introduce controlled mechanical load disturbances. A discrete-time mathematical motor model is incorporated as a supporting simulation tool for controller development and analysis, while the physical DC motor remains the experimental plant. The system is evaluated under various mechanical-load conditions with speed references. Motor-speed response, tracking error, controller output, and actual performance metrics are analysed to assess reference tracking and disturbance-rejection behaviour. The proposed setup provides an experimentally oriented platform for investigating closed-loop DC motor speed regulation under controlled physical loading and establishes a basis for systematic controller tuning, experimental validation, and further development of HIL-based electromechanical control systems.

Keywords: Hardware-in-the-Loop, DC motor, PID control, speed regulation, quadrature encoder, mechanical load disturbance, LabVIEW, closed-loop control.

1 INTRODUCTION

DC motors are valued for their high starting torque and near-linear torque-speed relationship. Since the establishment of controlled rotational motion is fundamental to many electromechanical systems, and the DC motor has a straightforward dynamic representation and controllability, making them suitable for both experimental investigation and controller development. These characteristics make them “workhorses” in industry and education, easily exceeding heavy start-up loads and allowing straightforward speed control [3]. In ideal cases, a DC drive would maintain a desired speed perfectly, but in practice, real systems rarely achieve this. Variations in mechanical load, parameter uncertainty, and sensor noise cause deviations from the setpoint. For instance, if a sudden load torque is applied, a purely proportional controller alone would let the speed

drop, and even a well-tuned PID controller shows a transient error before regulation restores the setpoint [2], [4]. In a normal system without disturbances, closed-loop feedback would hold a constant speed, but any lag can lead to overshoot or steady-state error. Recent research investigates PID-based DC motor regulation, system identification, and controller optimisation, indicating that speed tracking and controller tuning remain active problems [3]–[6].

An ideal closed-loop system has to maintain the desired speed despite changes in conditions. In reality, applying an additional mechanical load increases the opposing load torque and causes the shaft speed to deviate from its reference. PID controllers are the industry standard for DC motor speed regulation due to proven performance [3]; however, PID tuning does not adapt to changing load or unmodeled friction. The resulting error is then detected through the feedback sensor and supplied to the controller:

$$\dot{y}(t) = \dot{r}(t) - \dot{y}(t)$$

where $r(t)$ is the reference speed and $y(t)$ is the measured motor speed.

(1)

In our system, the variable friction brake provides a controlled way of producing this disturbance. Thus, it must be tested on physical hardware to validate performance under realistic disturbances, and our experiment is not simply asking whether the motor reaches a target speed. It asks whether the complete feedback system can detect and compensate for a physical change in mechanical loading. Sensor-assisted PID research has similarly emphasised the importance of feedback and disturbance rejection in DC motor applications [7]. This has motivated recent research based on PID tuning and alternative control approaches Ekinici et al. [4]

Mathematical models play an efficient role in analysing motor dynamics and developing an initial controller. They result in providing rapid controller evaluation without repeatedly subjecting physical hardware to experimental conditions. The limitation is only in representing the dynamics and parameters included in its formulation. Physical motors introduce effects of friction, inertia, sensing, actuation, and parameter variation that may not be completely represented in a simple model.

Hardware-in-the-loop (HIL) acts as a bridge by filling the gap by integrating simulation models with physical hardware components, so that a controller can be tested and evaluated under any controlled conditions. Romdlony and Irsyadi [1] presented an HIL-based DC motor control platform, while Mufhaddhal et al. [2] integrated encoder-based DC motor speed measurement with HIL and adaptive control. In other work, they deployed an HIL experiment with a DC motor and a Data Acquisition (DAQ) board. They used MATLAB Stateflow for PID tuning on the real system, showing improved disturbance rejection compared to an internal model control PID [4]. Most recent work has been used to investigate modelling, estimation, and controller evaluation using HIL setups [8]. However, HIL configurations differ in what is represented as a hardware setup and what is simulated. Some use a simulated motor with hardware interfaces; others use

an actual motor and sensing hardware. Consequently, the term HIL not only represents how realistically mechanical disturbances are represented. Our system is an integration of a physical DC motor, encoder, variable mechanical load, and a software controller.

The recent literature indicates that most published HIL DC motor studies are either fully simulated or fixed experimental setups. Like Romdlony's HIL system used a virtual system [1], while Angel's work uses a fixed motor-generator system [4]. Similarly, Mufhaddhal [2] has included encoder feedback and MRAC control, but did not explore a user-adjustable load. Shah and Sands [9] compared PID methods on a DC drive for an underwater vehicle, but they implemented it for a specific controller application, not on systematic load variation. Across these strands, the least clearly addressed is the integration of such elements in a compact experimental platform where a physical mechanical load can be deliberately varied while the controller operates through real-time encoder feedback. The distinction is important because changing the physical load directly alters the motor's torque balance rather than modifying a parameter in a simulation.

The present study focuses on the experimental evaluation of a Hardware-in-the-loop speed control system for a physical DC motor under variable mechanical loading. Our system is an integration of a physical DC motor, an optical quadrature encoder for real-time speed feedback, a variable friction brake for mechanical load disturbances, and a LabVIEW-based PID controller. A discrete-time mathematical motor model and SubVIs are combined for controller development and analysis; however, the hardware DC motor remains as the experimental plant during testing. This setup enables the controller response to be tested and examined under changing load conditions while providing the flexibility of software-based implementation and monitoring.

For experimentally evaluating a controller under physical disturbances, the controller provides a control behaviour to be tested against the dynamics of an actual electromechanical system and is accessible for controller development, tuning, validation, and loading conditions. The integration of LabVIEW allows the

control algorithm, feedback signals, monitoring functions, and supporting simulation.

2 System Architecture

3 Overall HIL System Architecture

The hardware-in-the-loop (HIL) system evaluates the speed control of a DC motor under varying mechanical loading conditions. The experimental setup is an integration of LabVIEW, an Arduino Nano, an L298N motor driver, a DC motor equipped with a two-channel magnetic quadrature encoder, and a variable friction mechanism for mechanical disturbances. The low-level interface between the computer and the motor is handled by the Arduino Nano, and LabVIEW performs the control and visualization functions.

The system functions as a closed-loop feedback system where a reference motor speed is entered through the LabVIEW interface and compared with the original motor speed from the encoder. The resulting error is rectified and processed by the controller to generate a motor command and transmitted from LabVIEW to the Arduino Nano through the USB port.

The Arduino converts the command to a pulse-width-modulated signal and directs it to the L298N motor driver, as it supplies the required electrical power to the DC motor. The motor shaft has a controllable mechanical load using a friction mechanism, resulting in a change in rotational speed which is detected by the magnetic quadrature encoder. The encoder generates two phase shifted digital signals which are acquired by the Arduino Nano and processed to determine the motor speed and the measured speed is transmitted back to LabVIEW forming a feedback loop.

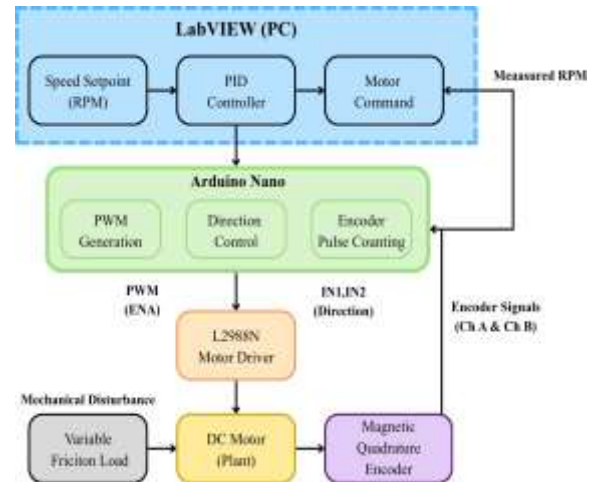


Fig. 1. Overall architecture of the proposed HIL-based DC motor speed-control system.

This setup is similar to DC motor HIL implementations, in which communication between the microcontroller and the motor setup happens; for example, Mufhaddhal et al. made a DC motor HIL system using an encoder, Arduino, and a computational controller for speed tracing [6], and many other studies also investigated experimental DC motor control using the controller optimisation method [2]-[5].

4 Arduino Nano as Hardware Interface

The Arduino Nano acts as a low-level interface between LabVIEW and the physical motor, where LabVIEW is responsible for the high-level computation, and Arduino provides the hardware-level timing and digital signal interfacing for motor actuation and encoder acquisition. The communication happens between LabVIEW and Arduino through a USB serial port at a baud rate of 115200 baud. LabVIEW sends the calculated motor command to the Arduino to return the measured motor speed received from the encoder. The main function of the Arduino is to convert the received control command to a PWM signal for controlling the motor driver enable input, generate a digital logic signal for controlling the direction of the motor through the H-bridge, and acquire the encoder signal to determine the rotation speed information for feedback.

Table I

5 Arduino Nano Signal Assignment

Arduino Pin	Connected Signal	Function
D9	ENA	PWM motor command
D8	IN1	Motor direction control
D7	IN2	Motor direction control
D2	C1 / Channel A	Encoder pulse input
D3	C2 / Channel B	Quadrature channel input
USB	UART	LabVIEW communication

6 L298N Motor Driver

The power interface between the Arduino Nano and the DC motor is provided by the L298N motor driver. The output pins of the Arduino are for logic-level control and can't directly provide the current required by the motor; therefore, the signal generated by the Arduino is supplied to the L298N, which switches the motor supply power to the motor. The motor driver mainly receives three control signals from the Arduino: the ENA input receives a PWM signal that determines the effective motor drive level, and IN1 and IN2 determine the direction of motor rotation.

The L298N therefore separates the control signal path from the motor power path, allowing the LabVIEW and Arduino to manage the physical motor without directly supplying the motor current from the microcontroller.

7 DC Motor and Variable Mechanical Load

The DC motor is the main physical part of the experimental HIL setup they are widely used for speed control applications because of its regulatable speed through electrical input and a feedback controller. Much research has tested the PID-based approach for DC motor speed under different operating conditions [2]-[5].

A variable friction mechanism is used to provide controllable mechanical disturbance to the motor shaft.

Under normal conditions, the motor functions toward the specified reference speed; when the friction load is increased, there is an opposing torque acts, resulting in a reduction in rotation speed. This deviation is measured by an encoder and supplied to the feedback controller.

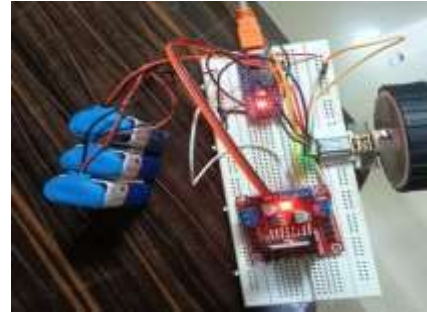


Fig. 2. Experimental hardware setup of the DC motor HIL platform.

8 Magnetic Quadrature Encoder

The magnetic quadrature encoder with two channels is used to provide rotational feedback from the motor shaft they generate two digital signals which are phase-shifted by approx. 90° , allowing the rotational motion to be represented as a sequence of digital pulses. They are designated as channel A (C2) and channel B (C2). The main purpose is the measurement of rotational speed and determination of rotational direction, where the speed is measured in a way that the generated pulses during the measurement interval are counted and converted to revolutions per minute (RPM), and the direction of rotation is determined by the phase relationship between two channels

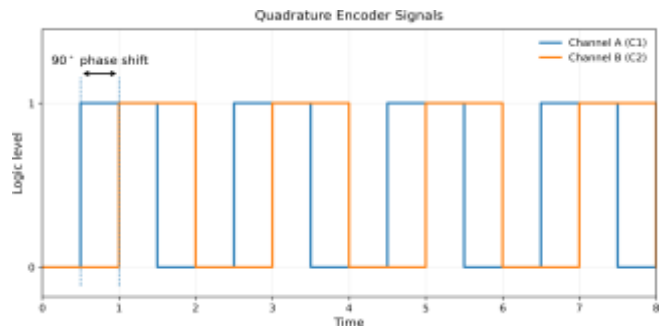


Fig. 3. Quadrature encoder signals and phase relationship between Channels A and B.

9 LabVIEW-Based Supervisory Controller

The high-level control and computational process is done by LabVIEW; its front panel provides the controls for desired motor speed control parameters, displays measured rpm speed error, motor command, and response waveform

LabVIEW has several advantages for experimental setup because the controller parameters can be modified without changing the motor control hardware, the motor response can be viewed simultaneously, and the measured data can be recorded for evaluation of transient and steady-state characteristics.

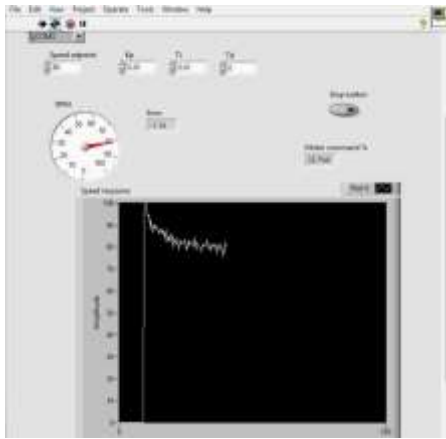


Fig. 4. LabVIEW front panel for closed-loop DC motor speed control.

10 DC Motor Mathematical Model

The electrical dynamics of the armature circuit can be represented by

$$V_a(t) = L_a \frac{di_a(t)}{dt} + R_a i_a(t) + K_e \omega(t)$$

(2)

In this equation,

$$J \frac{d\omega(t)}{dt} = K_t i_a(t) - B\omega(t) - T_L(t)$$

In this equation,

$V_a(t)$
 a

denotes the armature voltage,

L_a denotes the armature inductance, R_a

denotes the armature resistance, $i_a(t)$ denotes the armature current, K_e denotes the back-EMF constant, and $\omega(t)$ denotes the velocity.
 $V_a(t)$ denotes the armature voltage,

The mechanical dynamics of the motor can be expressed by

$$J \frac{d\omega(t)}{dt} = K_t i_a(t) - B\omega(t) - T_L(t)$$

(3)

In this equation, J denotes the rotor moment of inertia, K_t denotes the torque constant, B denotes the viscous-friction coefficient, and $T_L(t)$ denotes the applied load torque.

L_a

Equations (1) and (2) describe the coupling between the dynamics of the motor and the mechanical dynamics of the motor. In the proposed experiment, the $T_L(t)$ term is particularly

L_a

important because the mechanical friction mechanism is used to introduce controlled disturbances to the motor.

Recent work has employed modelling and system-identification techniques to represent DC motor dynamics before controller implementation [7]. Experimental PID studies have also combined modelling with physical motor testing to assess speed-control performance [2].

11 PI/PID Controller

The general PID control law is given by

$$u(t) = K_p e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt}$$

(4)

where K_p , T_i and T_d are the proportional gain, integral time and derivative time, respectively.

A summary of the controller and implementation parameters used in the developed LabVIEW environment is shown in Table II. The proportional and integral parameters were set to

$K_p = 0.001$ and $T_i = 0.001$, respectively, and the derivative parameter was set to $T_d = 0$.

$$u(t) = K_p e(t) + K_p / T_i \int_0^t e(\tau) d\tau$$

Thus, the configuration of the controller is equivalent to the PI control in the general PID structure. The controller output is limited to the 0–100% motor command range and transmitted to the Arduino Nano using serial communication at a baud rate of 115200. The supporting discrete-time motor model uses an update interval of 0.02 s.

TABLE II

I. CONTROLLER AND IMPLEMENTATION PARAMETERS

Parameter	Symbol	Value	Unit / Description
Proportional gain	K_p	0.01	—
Integral time	T_i	0.01	s
Derivative time	T_d	0	s
Controller structure	—	PI	$T_d = 0$
Controller output range	u	0–100	% motor command
Serial communication rate	—	115200	baud
Motor command interface	—	PWM	Arduino D9 → L298N ENA
Encoder feedback	—	Quadrature A/B	Arduino D2/D3
Speed feedback variable	y(t)	RPM	Encoder-derived speed
Speed reference	r	80	RPM

The proportional part gives a reaction to the current error, while the integral part adds up the error over time and helps reduce long-term steady-state error. The derivative part reacts to how the error is changing and can make the response faster. PID control and the problem of setting the parameters are still studied a lot for controlling the speed of a DC motor. Recent papers have looked at ways to set the parameters using math

and optimization methods, and comparing PI, PD, and PID types [2]–[5] [8] [9].

In the LabVIEW setup, the controller allows changing the values of K_p , T_i , and T_d . For the setup shown in the test interface, T_d is set to zero, so that specific setup works as a PI controller inside the PID system.

1.1 Supporting Discrete-Time Motor Model

In addition to the motor experiment, a simplified discrete-time motor model was created in LabVIEW to help with early controller development and response analysis. The model used in the LabVIEW implementation can be written as

$$\omega_{k+1} = \omega_k + 0.02(15u_k - \omega_k)$$

(6)

where ω is the current model speed, ω is the next model state, and u is the normalized motor command.

The number 15 is the model gain, and 0.02 is the discrete model update interval. This model gives a way to represent what the motor should do during controller development. It is important to note that Eq. (6) is the supporting simulation model and not the real motor dynamics. The experimental work discussed later in this paper uses the DC motor with encoder feedback, the Arduino interface, and the mechanical load.

This difference between model-based development and real testing is important because earlier studies have shown that mathematical or virtual models can help with controller design, while testing with hardware gives more details about how the real hardware behaves [1] [7] [10].

2. HIL IMPLEMENTATION AND EXPERIMENTAL METHODOLOGY

This HIL platform joins the controller to a real DC motor using an Arduino Nano and an L298N motor driver. With this setup, the controller can read the motor speed from the encoder, compute the difference from the target speed, and send a new command to the motor. Unlike a simulation test, this arrangement lets the HIL controller experience the real electrical, mechanical, sensing, and actuation behavior of the motor.

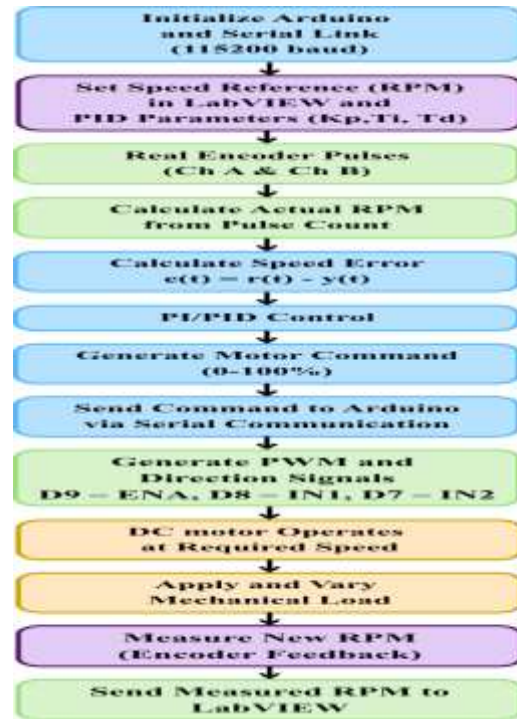


Fig. 5. LabVIEW block diagram of the implemented HIL speed-control system.

3. HIL Control Algorithm

First, set the desired motor speed on the LabVIEW front panel. The Arduino Nano reads the speed from the magnetic quadrature encoder. The measured speed is compared to the target to find the tracking error. The controller then uses this error to generate the motor command.

The command travels over USB serial to the Arduino Nano. The Arduino turns the command into a PWM signal and direction signals for the motor driver. The driver supplies the needed power to the real DC motor.

If a mechanical disturbance is added through the friction mechanism, the motor speed changes. The encoder detects this change. Sends the new speed back to the controller. This continuous feedback lets the controller adjust the motor command on the fly.

4. LabVIEW–Arduino Communication

LabVIEW acts as the control and visualization platform. The Arduino Nano takes care of the low-level hardware

interface tasks. Serial communication is set up through the USB cable at a baud rate of 115200.

The communication is two-way. LabVIEW sends the motor command to the Arduino. The Arduino replies with the measured motor speed back to LabVIEW. By keeping the control and visualisation in LabVIEW, the Arduino can focus on PWM generation. Reading the encoder signals. The LabVIEW block diagram that we built includes a communication interface, a PID controller, a speed-error calculation, a motor-command generator, a response visualisation, and elements for loop timing.

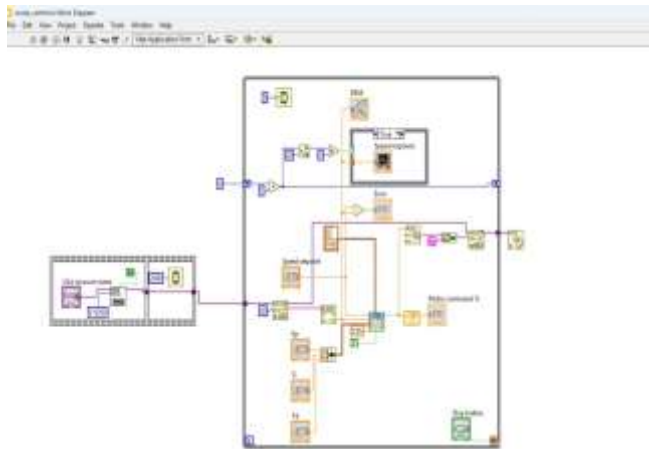


Fig. 6. LabVIEW block diagram of the implemented HIL speed-control system.

5. RESULTS AND DISCUSSION

The experimental evaluation has two parts: the model response and the physical HIL system response. This separation is necessary because the simplified mathematical model in LabVIEW shows the expected motor behaviour, while the HIL experiment uses the motor, encoder, driver, friction, and communication interface.

6. LabVIEW Model Response

The model-based implementation provides an environment for observing controller response before evaluating the physical motor. The model response waveform can be monitored through the speed-response graph while controller parameters are varied.

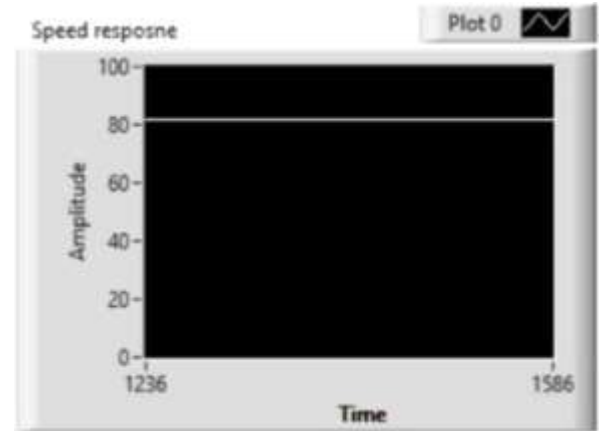


Fig. 7. LabVIEW model-based speed response.

The LabVIEW configuration shows a reference speed of 90 RPM. The controller parameters for the shown run are $K_p = 0.001$, $K_i = 0.001$ and $K_d = 0$. Since the derivative term is zero, this

configuration corresponds to PI control within the PID framework.

7. Experimental HIL Speed Response

The physical HIL experiment was conducted using the assembled DC motor, Arduino Nano, L298N driver, magnetic quadrature encoder, and variable friction mechanism.

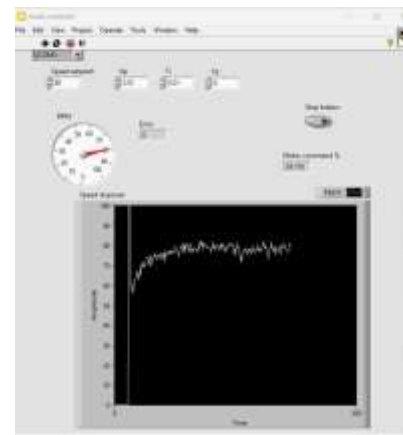


Fig. 8. Recorded LabVIEW closed-loop speed response of the DC motor under nominal operating conditions.

Fig. 8 presents the speed response of the physical DC motor under the nominal operating condition.

Following application of the reference speed, the motor speed increases toward the reference value. The encoder provides feedback to the controller, allowing the motor command to be adjusted according to the instantaneous speed error. After the transient, the measured speed approaches the reference value, and the remaining deviation can be evaluated using the steady-state error.

1.8 Response Under Mechanical Load

The principal experimental test involves introducing a friction disturbance to the motor shaft. When the friction load is applied, the motor experiences an opposing torque. Consequently, the measured motor speed temporarily deviates from the reference value. The resulting error is detected by the feedback system. Supplied to the controller.

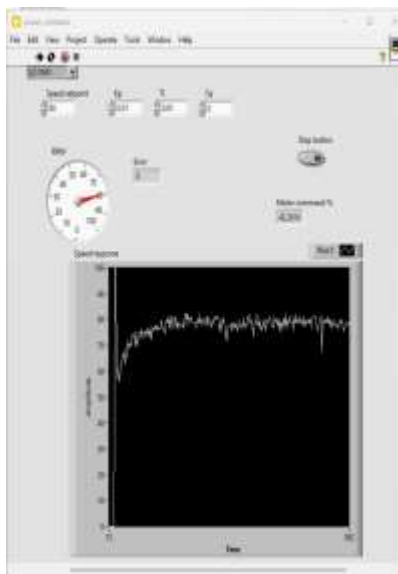


Fig. 9. Recorded LabVIEW closed-loop speed response of the DC motor during application of load.

As shown in Fig. 9, the application of the load produces a transient reduction in motor speed. The magnitude of this deviation depends on the applied load and the operating condition of the motor. Following the disturbance, the controller modifies the motor command to compensate for the increased load torque. The motor speed subsequently moves toward the reference value, demonstrating the closed-loop response of the system to a disturbance.

1.9 Controller Response to Disturbance

The disturbance response can also be evaluated through the controller output. When the load increases, the measured motor speed. Consequently, the speed error increases. The controller responds by increasing the motor command. This increase in command provides electrical actuation to compensate for the increased mechanical load.

The relationship observed during the experiment can therefore be represented as:

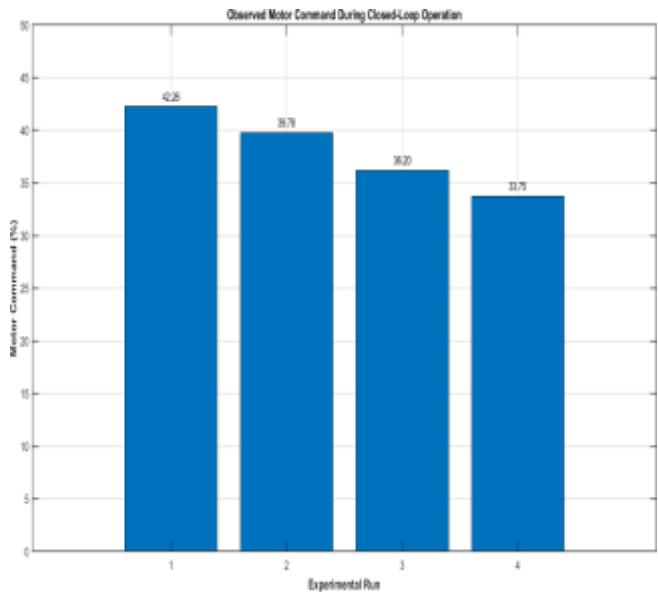


Fig. 10. Motor-command values observed during the recorded closed-loop operating conditions.

The increase in motor command following the application of the disturbance indicates that the controller responds to the speed deviation rather than operating solely according to a predefined open-loop command.

1.10 Model and Experimental Response Comparison

The simplified LabVIEW model gives a picture of how control should behave, while the HIL experiment brings in real motor physics. By comparing the two responses, we can spot gaps between Model behaviour and the real Experimental Response implementation.

Table Iii

li. Observed Closed-Loop Performance

Ru n	Reference Speed (RPM)	Error (RPM)	Absolute Error (%)	Motor Command (%)
1	80	0.00	0.00	42.2614
2	80	0.00	0.00	39.7780
3	80	+1.14	1.43	36.1989
4	80	-1.14	1.43	33.7541

The Model and Experimental Response both aim to follow the reference speed. Still, differences appear because the simplified Model does not capture every detail of the Experimental Response setup.

The physical implementation brings in motor friction, mechanical loading, encoder measurement quirks, motor-driver traits, PWM actuation, and the link between Arduino and LabVIEW. Because of these factors, deviations between Model and Experimental Response are normal. Push us toward HIL-based evaluation.

III. DISCUSSION

The experimental results show how the controller talks to the motor as mechanical conditions change. When conditions stay normal, the controller works to shrink the gap between the reference speed and the speed seen by the encoder. If the mechanical load rises, the motor speed drifts from the reference and the controller output shifts accordingly. That behaviour matches the core idea of feedback speed control.

Recent DC motor studies explored PID control, fine-tuned controller parameters, sensor-assisted control, and HIL evaluation. This work adds to that view by adding a friction device to create a repeatable disturbance while keeping the controller and monitoring in LabVIEW. The key practical value of this platform is that we can watch the controller act on the motor and sensor instead of only a math model of the plant. These experimental data let us assess tracking quality and how fast the system recovers from disturbances in real-world conditions.

IV. CONCLUSION

This paper shows how we built a hardware-in-the-loop platform that controls the speed of a DC motor while the load changes. The system brings together a DC motor, a magnetic quadrature encoder, an Arduino Nano, an L298N driver, a variable friction device, and LabVIEW PI/PID control. The Arduino Nano acts as the bridge between the computer control world and the real motor system. It creates the PWM and direction signals needed to drive the motor. It collects the quadrature encoder signals that give speed data. LabVIEW handles the reference speed, computes the error, runs the controller, shows the response, and logs the data.

LabVIEW also holds a math model of the DC motor to help build and test the controller early. Still, the real DC motor stayed as the plant during HIL tests. This choice lets us see how real motor dynamics, load changes, sensor quirks, actuation, and hardware links affect the controller. In the experiments, we add a friction load to shake up the motor speed. The encoder spots the speed shift. Sends that back to the controller, which then tweaks the motor command to counter the disturbance.

These measurements let us measure error, how the system behaves when it starts, and how it recovers from disturbances. In short, this platform gives a lab setup to study closed-loop DC motor speed control and to compare the simple model with the real motor response.

1 FUTURE WORK

Future work could aim for repeatable load disturbances, systematic tuning of controller settings, and testing across more motor speeds and loads. We might also try adaptive or model-based control. Use finer-resolution feedback. Adding auto data capture and performance analysis inside LabVIEW could make the controller testing repeatable and faster.

REFERENCES

1. M. Z. Romdlony and F. Irsyadi "Hardware-in-the-loop simulation of DC motor as a media for control system design and testing " Journal of

- Mechatronics, Electrical Power and Vehicular Technology vol. 12 No. 2 Pp. 81–86, 2021.
2. I. Cojuhari, I. Fiodorov, B. Izvoreanu and D. Moraru "Synthesis of PID Algorithm for Speed Control of the DC Motor " in Proc. 2022 Int. Conf. Development and Application Systems (DAS) 2022.
 3. S. Ekinici, B. Hekimoğlu and D. Izci "Opposition based Henry gas solubility optimization as an algorithm for PID control of DC motor " Engineering Science and Technology an International Journal, vol. 24, No. 2 Pp. 331–342, 2021.
 4. S. Ekinici, D. Izci and M. Yilmaz "Efficient Speed Control for DC Motors Using Novel Gazelle Simplex Optimizer " IEEE Access, vol. 11 Pp. 105830–105842, 2023.
 5. D. Baidya, S. Dhopte and M. Bhattacharjee "Sensing System Assisted Novel PID Controller for Efficient Speed Control of DC Motors in Electric Vehicles " IEEE Sensors Letters, vol. 7 No. 1 2023.
 6. M. A. Mufhaddhal, E. Susanto and M. R. Rosa "Model Reference Adaptive Control of a DC Motor Speed with Encoder using Hardware in the Loop " Indonesian Journal of Electronics and Instrumentation Systems vol. 14 No. 2 Pp. 151–162, 2024.
 7. "Mathematical Model Construction of DC Motor by Closed-Loop System Identification Technique Using MATLAB/Simulink " in 2021 9th International Electrical Engineering Congress (iEECON) 2021.
 8. "Optimized PI/PD/PID Controller Design for DC Motor Speed Control: A Comparative Study of Random Forest-Based Tuning & Classical Methods " in 2025 Devices for Integrated Circuit (DevIC) 2025.
 9. "Optimization of PID Controller Parameters for Speed Control of PMDC Motor Using Simulated Annealing " in 2025 International Conference on Electrical, Computer and Energy Technologies (ICECET) 2025.
 10. "Modeling, Estimation and Control of DC Motors: An Optimization Approach Using Hardware-In-The-Loop (HIL) " in 2025 IEEE 8th International Conference, on Electrical, Control and Computer Engineering (InECCE) 2025.