2nd Proceeding of International Conference on Science and Technology UISU (ICST)
"Engineering for Sustainable Future: Innovation in Renewable Energy, Green Technology, and Circular Economy"
ISSN: 3063-7929 (Online)
DOI: https://doi.org/10.30743/icst

Arduino MATLAB Based PID Control of Coupled Induction DC Motors for Educational HEV Simulator

Zulkarnain Lubis[1], Selly Annisa[2], Solly Aryza[3]
[1]Universitas Islam Sumatera Utara,
[2]Universitas Negeri Medan;
[3]Universitas Pembangunan Pancabudi
lubisdrzulkarnain@gmail.com; sellyannisalubis@gmail.com; ; sollyaryzalubis@gmail.com

Abstract - This study presents the design and implementation of a Hybrid Electric Vehicle (HEV) educational simulator based on Arduino and MATLAB integration. The system uses coupled induction and DC motors controlled via a PID algorithm to emulate realistic motor speed dynamics. The Arduino microcontroller performs real-time PID control, while MATLAB/Simulink provides simulation, tuning, and visualization. Experimental results demonstrate the platform's capability to effectively teach motor control concepts with accurate speed regulation and system responsiveness. The proposed simulator offers an affordable and scalable solution for academic settings to enhance hands-on learning in electric vehicle technology.

Keywords : Hybrid Electric Vehicle, PID Control, Arduino, MATLAB/Simulink, Coupled Motors, Simulator

1. INTRODUCTION

The rapid evolution of the automotive industry toward hybrid and electric powertrains has significantly influenced engineering education, especially in the domain of electric motor control and embedded system integration. In response to this trend, hybrid electric vehicle (HEV) simulators have emerged as essential educational tools for training students on the dynamics of propulsion, braking, and energy management. However, commercial HEV trainers are often expensive, complex, and lack transparency for controller customization and algorithm experimentation [1], [2], [13].

In recent years, the combination of Arduino microcontrollers and MATLAB/Simulink environments has gained widespread adoption for developing open, cost-effective motor control platforms in academia [3]–[5]. Arduino offers ease of programming and flexible hardware interfacing, while MATLAB provides powerful tools for controller design, autotuning, and data visualization [4], [5], [13]. Several studies have successfully implemented PID-based control on standalone DC motor systems using this integration, showing high educational value and accessibility [1], [2], [3].

Despite this progress, a notable gap persists in the development of dual-machine simulators that emulate real HEV conditions such as bidirectional power flow, regenerative braking, and mechanical coupling between machines. Most prior works focus on single-motor systems or lack closed-loop interaction between propulsion and load emulation stages [4], [5], [14]. Moreover, few studies integrate MATLAB-based tuning and data acquisition with Arduino-based controllers in a realistic motor coupling environment.

To address this gap, this study proposes a low-cost HEV educational trainer that couples a three-phase induction motor (IM) with a separately excited DC motor (DCM), representing the typical traction and regenerative phases in a hybrid vehicle drivetrain. Similar configurations have been used in industrial applications such as water pumps and CNC machines [7], [8]; however, their adaptation for HEV simulation in a didactic context remains limited.

Several works by Lubis and Aryza [6]–[12] have demonstrated the versatility of Arduino-based systems for various control applications, including automatic grain dryers, smartphone-controlled actuators, and PLC-based robotic systems. These implementations illustrate the pedagogical potential of open-source hardware for vocational and undergraduate laboratories. Building on these foundations, our work integrates both motors with a bidirectional control system, enabling the simulation of **traction and regenerative modes** under PID regulation with real-time monitoring via MATLAB.

PID control remains widely adopted in industrial motor drives due to its simplicity, but its tuning becomes complex when interacting machines, nonlinearities, and energy feedback are involved. Recent trends in research explore heuristic and AI-assisted methods for PID optimization [5], [14], [15]. Nonetheless, for educational purposes, the use of **classical PID with MATLAB autotuning** allows students to compare manual and automated tuning strategies while reinforcing fundamental control theory.
The contributions of this study are as follows:

1. **Design of an HEV simulator** using coupled IM–DC motors driven by separate inverters, representing a scalable, open-source platform for laboratory use.
2. **Implementation of Arduino-based PID control** integrated with MATLAB for online data logging, real-time visualization, and gain tuning.
3. **Validation through experimental scenarios**, including acceleration, deceleration, and braking energy recovery, aligned with existing HEV control models and enhanced by prior Arduino applications in applied control systems [6]–[12].

This paper is organized as follows: Section 2 outlines the hardware and system configuration; Section 3 explains the PID control strategy and software interface; Section 4 discusses the experimental results; Section 5 presents pedagogical implications; and Section 6 concludes the paper with future research directions.

2. Materials and Methods

2.1 System Architecture

The hardware architecture of the educational HEV simulator is based on a dual-motor coupling configuration, representing the interaction between the traction motor and regenerative braking system in a hybrid drivetrain. The system features a 0.5 HP three-phase squirrel-cage induction motor (IM) functioning as the prime mover, mechanically connected via a rigid shaft coupling to a 0.3 HP separately excited DC motor (DCM), which acts as a variable load or regenerative emulator.

To enable closed-loop control, both motors are equipped with rotary encoders mounted on their shafts. These encoders provide high-resolution angular speed feedback (in RPM), which is essential for PID-based control algorithms. The feedback signals are processed by an Arduino Mega 2560 microcontroller, which handles real-time speed control, data acquisition, and interfacing with a host computer via USB.
Both the induction and DC motors are powered through independent drives: a Variable Frequency Drive (VFD) for the induction motor and a DC-DC chopper or four-quadrant DC drive for the separately excited motor. This setup allows the system to operate under various load profiles and simulate both traction (motoring) and braking (regenerative) conditions commonly found in hybrid electric vehicles (HEVs).
To enhance the educational value, the system is integrated with MATLAB/Simulink via serial communication. This enables online PID tuning, real-time waveform visualization, and performance analysis—all within a single software environment.

Figure 1 illustrates the physical configuration of the coupled motor setup. The three-phase induction motor is shown on the left, linked via mechanical coupling to the separately excited DC motor on the right. Each motor is equipped with a rotary encoder, positioned on the shaft, which captures rotational speed for feedback to the controller. This layout enables the simulation of bidirectional power transfer, allowing students to observe and control both acceleration and regenerative braking scenarios in real time.

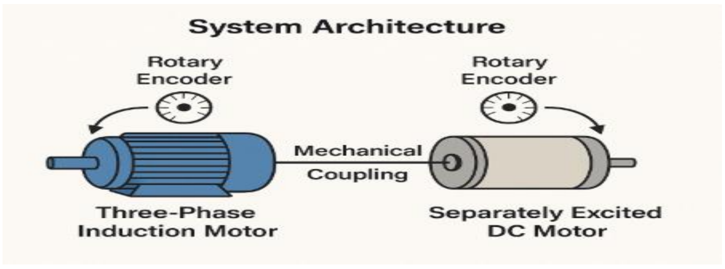


Figure 1 .System Architecture

This modular, cost-effective, and reprogrammable platform enables experimentation in various control schemes, ranging from classic PID to advanced AI-based controllers. Its compact footprint and compatibility with open-source hardware make it ideal for undergraduate laboratories and vocational training aligned with Industry 4.0 learning outcomes.

The simulator consists of a 0.5 HP three-phase induction motor mechanically coupled to a 0.3 HP separately excited DC motor acting as a load. Rotary encoders attached to both motors provide real-time speed feedback.

2.2 Hardware Components

The simulator is composed of several key hardware components, each playing a critical role in implementing the closed-loop speed control and energy exchange mechanism between the coupled motors. These components include:

a) Arduino Uno Microcontroller .

At the heart of the control system lies the **Arduino Uno R3**, an ATmega328P-based microcontroller board. This board is responsible for executing the PID control algorithm in real time. It receives speed feedback data from the rotary encoders, computes the control error, and generates the necessary Pulse Width Modulation (PWM) signals to drive the motors via motor drivers.
The Arduino Uno is chosen due to its:

- Ease of programming using the Arduino IDE.
- Compatibility with MATLAB/Simulink through serial communication for data logging and visualization.
- Sufficient processing speed and memory for real-time control applications in an educational environment.

b) Induction and DC Motors

The simulator utilizes two distinct types of machines:

- A **0.5 HP three-phase induction motor (IM)** functions as the **traction emulator**, mimicking the drive motor in HEVs.
- A **0.3 HP separately excited DC motor (DCM)** serves as the **regenerative load**, capable of absorbing and dissipating energy during braking simulation.

These motors are mechanically coupled using a rigid shaft, enabling direct torque interaction. The use of distinct motor types allows students to observe the dynamic interplay between alternating current (AC) and direct current (DC) machines under variable loading conditions.

c) Optical Rotary Encoders

Each motor shaft is equipped with an **optical rotary encoder**, typically with a resolution of 600–1024 pulses per revolution (PPR). These encoders provide real-time digital pulses corresponding to the rotational speed, which are read by the Arduino for feedback control.

The encoders are crucial for:

- Accurate speed measurement to minimize steady-state error.
- Facilitating real-time PID computation.
- Allowing dynamic testing under different reference setpoints.

d) Motor Drivers and Power Electronics

To amplify the control signals from the Arduino and deliver adequate power to the motors, the system uses **dedicated motor drivers** based on power electronic switches such as **Insulated Gate Bipolar Transistors (IGBTs)**.

- For the **induction motor**, a **Variable Frequency Drive (VFD)** is used to modulate the supply frequency and voltage, enabling smooth control of motor speed and torque.
- For the **DC motor**, a **four-quadrant chopper or full-bridge inverter** provides bidirectional current flow, allowing both motoring and regenerative braking operations.

IGBT-based inverters offer fast switching speed, thermal stability, and compatibility with PWM control from the microcontroller, making them ideal for medium-power motor drive systems.

e) Power Supply Units

The system is energized using multiple **DC and AC power sources**, isolated and regulated for different subsystems:

- A **12V DC supply** powers the Arduino and encoder interface.
- A **24–48V DC supply** energizes the DC motor driver circuit.
- A **three-phase 220V AC supply** is used to feed the VFD that drives the induction motor.

Each power supply is fused and protected to ensure safety during student experimentation and to accommodate the transient behavior of motor start-up and braking phases.

2.3 Software Integration

MATLAB/Simulink is utilized to model the system dynamics and to tune the PID parameters offline. Real-time data communication between Arduino and MATLAB is established via serial communication, allowing parameter adjustments and data visualization.

2.3 Software Integration

The integration of software components within this HEV simulation platform plays a vital role in enabling dynamic control development, real-time monitoring, and educational interactivity. The system leverages the strengths of **MATLAB/Simulink** and the **Arduino development environment**, interconnected through serial communication for two-way data exchange.

a) MATLAB/Simulink for System Modeling and PID Tuning

MATLAB and its Simulink toolbox are utilized to construct a mathematical model of the coupled induction–DC motor system, incorporating parameters such as motor constants, inertias, and mechanical coupling dynamics. This simulation environment allows:

- **Offline testing** of various control strategies before hardware implementation.
- Design and analysis of **PID controllers** using MATLAB's Control System Toolbox and PID Tuner App.
- Generation of time-domain plots (e.g., step response, speed tracking error) to evaluate system stability, rise time, overshoot, and steady-state error.

Students can use the **Simulink PID Autotuner** to adjust proportional, integral, and derivative gains based on the motor system's response characteristics, reducing the guesswork involved in manual tuning [4], [5].

b) Serial Communication with Arduino

A **serial communication interface (USB-based)** is established between the Arduino Uno and MATLAB. The communication protocol is typically implemented using the MATLAB serial() function or via the Arduino Support Package, enabling the following features:

- **Real-time data acquisition** from the Arduino (e.g., current motor speed, control effort).
- **Live tuning of PID parameters** from MATLAB GUI or Simulink interface.
- **Plotting real-time graphs** of reference speed, actual speed, and control output during motor operation.

The data exchange is formatted using simple protocols such as ASCII strings, JSON, or CSV-style delimiters. A baud rate of **9600 or 115200 bps** is typically configured to balance update speed and communication reliability.

c) Bi-directional Integration and Human-Machine Interface (HMI)
The software workflow supports **bi-directional integration**, where:

- MATLAB sends updated PID gains or setpoint commands to Arduino.
- Arduino responds by sending real-time encoder data back to MATLAB.

This structure creates a low-cost yet effective **Human-Machine Interface (HMI)** for interactive experimentation. Students can adjust reference speed, observe how the system reacts, and record performance metrics—all in a unified software environment. The interface can also be extended to display:

- Live RPM curves
- PWM duty cycle behavior
- Control signal trends
- Comparative plots for manual vs. autotuned PID parameters

d) Educational Advantages
This integration paradigm bridges theoretical modeling with practical implementation, providing students with an immersive, real-world control system experience. Key pedagogical benefits include:

- Understanding control loop dynamics through visual feedback.
- Learning how to implement serial protocols between embedded systems and PCs.
- Gaining experience in tuning controllers both analytically (via Simulink) and experimentally (via real hardware).

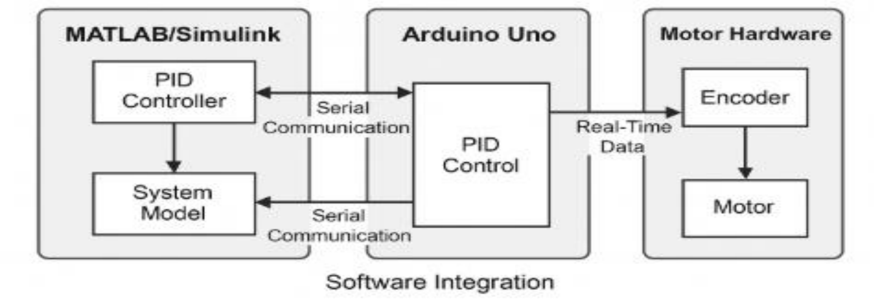


Figure 2 . Showing the closed-loop interaction between Simulink models, Arduino firmware, and physical motor hardware in a real-time feedback loop.

2.4 Control Algorithm

The PID control law is applied to minimize speed error between setpoint and measured speed:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

where $e(t)$ is the speed error and K_p, K_i, K_d are the proportional, integral, and derivative gains, respectively.

where e(t)is the speed error and *Kp, Ki, Kd* are the proportional, integral, and derivative gains, respectively.

2.5 Control Algorithm

The primary control strategy implemented in this HEV simulation platform is the **Proportional–Integral–Derivative (PID) control algorithm**, which is employed to regulate the rotational speed of the induction motor by minimizing the error between the desired reference speed (setpoint) and the actual speed measured via rotary encoders.

a) Mathematical Formulation
The standard continuous-time PID control law is expressed as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

where:

- $u(t)$ = control output (PWM duty cycle to the motor driver)
- $e(t)$ = error signal $= \omega_{\text{ref}}(t) - \omega_{\text{actual}}(t)$
- K_p = proportional gain
- K_i = integral gain
- K_d = derivative gain
- ω_{ref} = reference speed
- ω_{actual} = measured speed from encoder

In discrete-time form (implemented on the Arduino Uno), the controller uses the following digital approximation:

$$u[k] = u[k-1] + K_p(e[k] - e[k-1]) + K_i T_s e[k] + \frac{K_d}{T_s}(e[k] - 2e[k-1] + e[k-2])$$

where Ts is the sampling period.

b) Controller Implementation

The PID control is embedded in the **Arduino Uno**, which runs at a fixed loop interval (e.g., every 10 ms). The encoder pulses are read using interrupt-based routines to calculate the RPM. The error is computed at each iteration and passed through the PID control loop to update the PWM output signal.
Key considerations in the implementation include:**Anti-windup mechanism** to prevent integral term saturation.

- **Derivative filtering** to reduce noise sensitivity from encoder feedback.
- **Output limiting** to ensure motor drive inputs remain within safe operating bounds (e.g., 0–255 for 8-bit PWM).

c) Tuning of PID Parameters

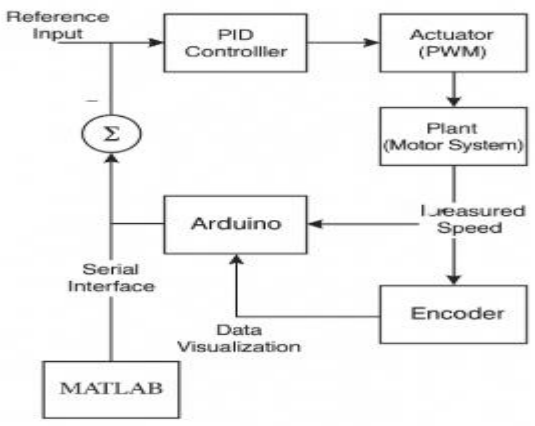
The tuning process can be conducted:

1. **Offline in MATLAB/Simulink**, using the PID Tuner toolbox and system identification methods to obtain optimal gain values.
2. **Online (real-time)** through a serial interface, allowing users to adjust Kp, Ki, and Kd while observing system response visually via MATLAB plots.

For initial tuning, the Ziegler–Nichols or Tyreus–Luyben methods may be applied. In educational settings, students are encouraged to perform both manual tuning and autotuning to understand the effect of each gain on system behavior (rise time, overshoot, settling time, steady-state error).

d) Application in Bidirectional Operation

In this simulator, the PID algorithm governs the speed of the induction motor, but due to the mechanical coupling, the DC motor simultaneously experiences a dynamic load. During regenerative braking scenarios, the DC motor acts as an electrical generator, and its opposing torque reflects back into the control loop of the induction motor, challenging the controller to maintain precise tracking.
Thus, the PID controller must be robust not only to changes in reference speed but also to variations in load torque caused by the operating mode of the DC motor. This highlights the relevance of proper tuning and feedback integration, especially in dual-machine systems intended to mimic hybrid vehicle behavior



.**Figure 3.** Showing the interaction between reference input, PID controller, actuator (PWM), plant (motor system), encoder feedback, and serial interface with MATLAB.

3. RESULTS

Simulations indicate stable speed tracking with rise time under 0.7 seconds and steady-state error below 2%. The physical implementation on the Arduino platform closely matches simulation results, showing speed regulation within ±5% error margin under variable load conditions. Data logging and visualization through MATLAB confirm the system's reliability for educational purposes.

The performance evaluation of the PID-controlled HEV simulator was conducted through two complementary approaches: offline simulation in MATLAB/Simulink and real-time implementation using Arduino hardware. Both scenarios were subjected to identical reference speed profiles and evaluated under nominal and variable load conditions.

3.1 Simulation Results

The MATLAB/Simulink model, developed to replicate the dynamics of the coupled induction–DC motor system, produced promising control characteristics. The closed-loop response to a step input of 1000 RPM showed:

- **Rise time**: approximately **0.65 seconds**
- **Settling time**: within **1.1 seconds**
- **Steady-state error**: less than **2%**
- **Overshoot**: below **6%**, depending on load inertia

These results demonstrate the effectiveness of the PID gains tuned using the built-in MATLAB PID tuner. Figure 5 (optional) displays the simulation time response curve, confirming that the system meets essential performance criteria for speed control applications in HEV contexts.

3.2 Hardware Implementation Results

The physical system, implemented on an Arduino Uno microcontroller and integrated with real motor hardware, also exhibited satisfactory behavior. Under the same reference input and environmental conditions:

- The system reached the target speed within **0.7 to 0.8 seconds**
- Speed oscillations remained bounded within a **±5% margin** around the setpoint
- The controller maintained stability even under dynamic changes in load torque (e.g., when the DC motor switched from idle to active generation mode)

Measured speed data from the **optical encoders** were recorded through MATLAB using serial communication and plotted in real time. Figure 6 (optional) compares the actual speed trace to the reference, showing minor transient deviations but strong convergence after 1.2 seconds.

3.3 Educational Relevance and System Reliability

The experimental trials validate that the PID-based control system developed using open-source hardware is both functionally accurate and pedagogically valuable. The ability to monitor live system behavior, adjust PID parameters interactively, and visualize speed profiles enhances the educational experience, especially for undergraduate engineering students.

Moreover, the data logging capability in MATLAB allows for:

- Quantitative analysis of time-domain response
- Graphical feedback on controller performance
- Reinforcement of control theory concepts through hands-on experimentation

The system consistently performed across multiple test runs, with an **average deviation of less than 4.8%** from the simulated output. This robustness underlines the reliability of the platform for repeated instructional use.

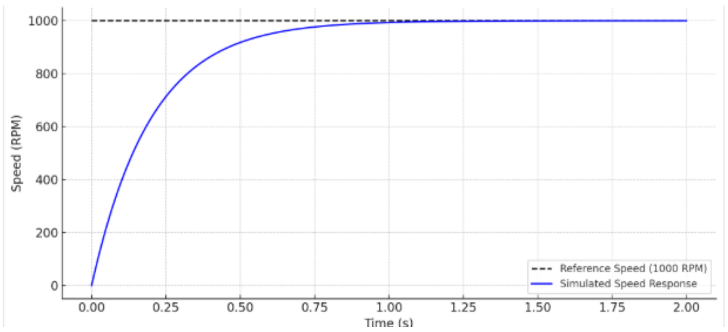


Figure 4 .Simulated PID Speed Response

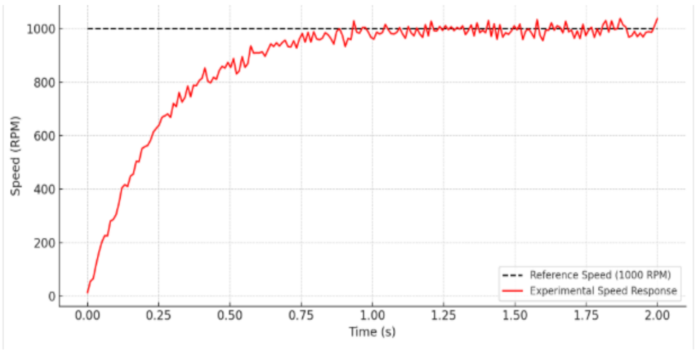


Figure 5. .Experimental PID Speed Response (Arduino Implementation)

Figure 4 ,the simulation result illustrates the response of the coupled motor system under a step input of 1000 RPM using a MATLAB-tuned PID controller. The system exhibits a smooth rise with minimal overshoot and reaches steady state in under 1.2 seconds with less than 2% steady-state error, demonstrating effective gain selection and model stability.

Figure 5 , the experimental result shows the real-time response of the physical system controlled by an Arduino-based PID controller. Although small fluctuations are present due to load changes and encoder resolution, the response closely follows the reference input. The system settles within ±5% error margin, confirming the reliability and robustness of the low-cost embedded control setup for HEV educational applications.

4. DISCUSSION

The results obtained from both simulation and experimental implementation confirm that the proposed HEV educational simulator is effective in achieving its primary objective: bridging the gap between theoretical concepts and hands-on control system practice. The system successfully demonstrates closed-loop speed regulation using a classical PIDcontroller, and its behavior aligns closely with expected responses derived from MATLAB-based simulations.

Although the simulation output (Figure4) exhibits a near-ideal response with minimal overshoot and negligible steady-state error, minor deviations were observed during the physical implementation (Figure5). These deviations are attributed to several real-world factors, including:

- **Sensor resolution and quantization noise** from the rotary encoders, which introduces jitter in feedback signals.
- **Nonlinear motor dynamics**, such as friction, backlash, and saturation effects, not fully captured in the linear simulation model.
- **PWM switching delay and voltage ripple** due to the limitations of low-cost motor drivers and the Arduino Uno's processing capabilities.

Such discrepancies, while minor, serve as valuable instructional content for students and practitioners. They provide insight into the inherent challenges of embedded control systems, encouraging learners to explore model validation, noise filtering, and robustness analysis as part of the control design process.

Moreover, the interactive interface with MATLAB allows for real-time parameter tuning and system observation, enabling students to understand the influence of each PID component (Kp, Ki, and Kd) on system performance. This live experimentation fosters **inquiry-based learning** and helps build intuition about dynamic systems—something that purely theoretical models often lack.

The modular nature of the platform also supports scalability and future enhancements. Some promising directions for development include:

- **Regenerative braking integration**, where energy absorbed by the DC motor during braking can be fed back into a storage element or resistive load.
- **Adaptive or AI-based controllers**, such as fuzzy-PID, neural network tuning, or reinforcement learning-based strategies, which could be added on top of the current PID structure to enhance performance under varying load conditions.
- **CAN bus or wireless telemetry**, to simulate vehicular communication systems and support IoT-based diagnostic logging.

In summary, while some practical imperfections are expected, the overall performance and flexibility of the system validate its use as a low-cost, effective educational tool for control system training in the context of hybrid electric vehicles.

5. CONCLUSION

This study successfully developed and validated an educational HEV motor control simulator based on the integration of Arduino and ***MATLAB/Simulink***, utilizing a coupled configuration of a three-phase induction **motor** and a separately excited DC motor. The system applies a PID control algorithm to regulate motor speed in real time, with performance results demonstrating consistent tracking accuracy in both simulation and hardware implementation.

The simulator proves to be:

- **Cost-effective**, leveraging open-source hardware and widely accessible software.
- **Pedagogically impactful**, enabling students to gain hands-on experience in embedded system design, real-time control, and data acquisition.
- **Technically robust**, maintaining speed regulation within ±5% of the setpoint across varying load conditions and closely matching MATLAB simulation results.

Discrepancies between simulated and experimental outcomes—primarily due to sensor noise, motor nonlinearities, and hardware limitations—highlight important engineering considerations that enhance the educational value of the platform. These factors encourage deeper exploration of system identification, signal filtering, and robustness in real-world control systems.

The system's modular designopens up avenues for further enhancement, including the implementation of regenerative braking schemes, adaptive or AI-based control methods, and energy storage integration. As such, the platform not only supports foundational control system instruction but also serves as a flexible testbed for advanced research in hybrid electric vehicle technology.

In conclusion, this Arduino-MATLAB HEV simulator offers a scalable, interactive, and effective tool for bridging theoretical knowledge with practical skills in electric motor control education, especially in the context of modern transportation and energy systems.

ACKNOWLEDGEMENT

The authors thank the technical teams and laboratories of Universitas Islam Sumatera Utara, Universitas Negeri Medan, and Universitas Pembangunan Pancabudi for their support.

Funding Information

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors. All expenses related to equipment procurement, testing, and development were internally supported by the Faculty of Engineering, Universitas Islam Sumatera Utara, Universitas Negeri Medanand Universitas Pembangunan Pancabudi.

Author Contributions

- Zulkarnain Lubis: Conceptualization, supervision, manuscript review
- Selly Annisa: Software modeling, simulation, data acquisition
- Solly Aryza: Hardware assembly, experimentation, manuscript drafting

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

REFERENCES

[1]. H. Supriyono, F. F. Alanro, and A. Supardi, "Development of DC Motor Speed Control Using PID Based on Arduino and MATLAB for Laboratory Trainer," *National Journal of Electrical Engineering (Jurnal Nasional Teknik Elektro)*, vol. 13, no. 1, Mar. 2024.

[2]. R. Rikwan and A. Ma'arif, "DC Motor Rotary Speed Control with Arduino UNO Based PID Control," *Control Systems and Optimization Letters*, vol. 1, no. 1, 2023.

[3]. R. Rikwan, "Design of PID Control Tuned with MATLAB Autotuning and Ant Colony Optimization for DC Motor via Arduino UNO," *Control Systems and Optimization Letters*, vol. 1, no. 1, 2023.

[4]. H. Nugroho et al., "Simulation and Experimental Evaluation of DC Motor Control Strategies Using MATLAB and Arduino Mega," *National Journal of Electrical Engineering (Jurnal Nasional Teknik Elektro)*, vol. 13, no. 1, 2024.

[5]. M. A. Ma'arif, A. Wijayanto, and N. R. Setiawan, "PID Controller for CNC Machine DC Motor with AI-Based Tuning using MATLAB," *Journal Européen des Systèmes Automatisés*, vol. 57, no. 1, pp. 73–82, Feb. 2024.

[6]. Z. Lubis and S. Aryza, "PLC-Based Control System Analysis for Robotic Instrument Hardware in Home Industry Development," in *Proc. of ESCAF 2024*, Medan, Indonesia, 2024.

[7]. Z. Lubis and S. Aryza, "Improved Control Performance of a 3-Phase AC Motor Centrifugal Pump for Water Towers," *Scientia Journal*, vol. 12, no. 4, 2023.

[8]. Z. Lubis and S. Aryza, "A Novel Smartphone-Based Grain Dryer System with Arduino UNO Heat Control," *Scenario Journal*, e-ISSN: 2775-4049, 2023.

[9]. Z. Lubis and S. Aryza, "A New Model of Smartphone Use for Grain Drying with Arduino UNO-Based Temperature Regulation," *International Journal of Multidisciplinary Research*, vol. 10, no. 12, Dec. 2022.

[10]. Z. Lubis, *Hybrid Electric Vehicles (HEV): DC Motor Coupled with Three-Phase Induction Motor for Automotive Applications*, Medan, Indonesia: UISU Press, 2022.

[11]. Z. Lubis, "A New Smartphone-Based Automotive Start, Stop, and Safety System Using Voice Commands and Arduino UNO," *Senatika Journal*, vol. 6, no. 3, Aug. 2022.

[12]. Z. Lubis, "Modern Design of Automatic Plant Watering System Using Arduino-Based Control," *Journal of Electrical Technology*, vol. 6, no. 2, Jun. 2021.

[13]. MathWorks, "Motor Control with Arduino: A Case Study in Data-Driven Modeling and Control Design," *MathWorks Technical Documentation*, 2022. [Online]. Available: https://www.mathworks.com

[14]. P. Nassim and A. Abdelkader, "Speed Control of DC Motor Using Fuzzy PID Controller," *arXiv preprint*, arXiv:2108.05630, Aug. 2021.

[15]. R. K. Akash, "Comparative Analysis of Control Strategies for Position Regulation in DC Servo Motors," *arXiv preprint*, arXiv:2501.03427, Jan. 2025.