International Journal of Robotics and Control Systems
Vol. 4, No. 3, 2024, pp. 1186-1206
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

Simulation and Arduino Hardware Implementation of ACO, PSO, and FPA Optimization Algorithms for Speed Control of a DC Motor

Adil Najem [a,1,*], Ahmed Moutabir [a,2], Abderrahmane Ouchatti [a,3]

[a] G.E.I.T.I.I.L laboratory, Aïn chok Faculty of Sciences Casablanca, Hassan 2nd University, Morocco
[1] adilnajem2@gmail.com; [2] a.moutabir65@gmail.com; [3] ouchatti_a@yahoo.fr
* Corresponding Author

ARTICLE INFO

Article history
Received May 14, 2024
Revised June 29, 2024
Accepted July 07, 2024

Keywords
Ant Colony Optimization;
Particle Swarm Optimization;
Flower Pollination Algorithm;
PID Controller

ABSTRACT

This article proposes implementing and comparing the effectiveness of three optimization algorithms (ACO, PSO, and FPA) for tuning a proportional-integral-derivative (PID) controller on an Arduino Mega 2560 board. This relatively unexplored approach aims to evaluate these algorithms through practical experiments. The choice of PID control is due to its design simplicity and widespread industrial use. Similarly, the permanent magnet DC motor (PMDC) was selected because of its crucial role in various industrial sectors. Tuning PID parameters using optimization algorithms has garnered increasing interest due to its demonstrated efficiency. Several studies have validated the stability of ACO, PSO, and FPA algorithms, justifying their selection. In this article, simulation results showed that ACO, with a response time of 0.322s and an overshoot of 0.68%, was more effective than PSO, which had a response time of 0.768s and an overshoot of 13%. FPA had a response time of 0.347s, close to ACO, but a higher overshoot of 6%. In practice, several factors come into play, such as speed ripples caused by the speed sensor, and machine saturation, which must be considered to ensure practical implementation. After adjusting the PID parameters and integrating a low-pass filter in the feedback loop, ACO, with a response time of 0.596s and an overshoot of 1.68%, was very close to FPA, which had a response time of 0.644s and an overshoot of 0.81%. This comparison highlighted the advantages of the FPA algorithm, which is simple to use, requires fewer parameters to adjust, and takes less time than ACO. This study suggests the potential for implementing a hybrid FPA-ACO algorithm, leveraging the strengths of both algorithms.

1. Introduction

The PID controller remains one of the most widespread and effective tools in the field of automation and control of dynamic systems [1], [2]. Its popularity endures due to its simple design, ease of implementation, and ability to provide robust control performance across a wide range of applications such as : Variable Speed Pumping Systems [3], robotic arm [4], Magnetic Levitation System [5], [6], Converter [7], [8], Inverted Pendulum [9], Quadcopter [10], [11]. When estimating PID controller coefficients, two main approaches are commonly used:

- Trial and Error Method: This involves manual adjustments of coefficients, which can be time-consuming and prone to human error, and does not always guarantee an optimal solution [12], [13].
- Optimization Algorithms: These algorithms aim to minimize a cost function that measures the difference between the system response and the desired behavior by progressively adjusting the PID coefficients using mathematical techniques [14]-[16]. They offer the advantage of converging towards an optimal solution, especially for complex or nonlinear systems, and can be automated to save time and resources. However, they require accurate system modeling, and the choice of algorithm can impact solution quality.

Recent research has explored a number of optimization algorithms for adjusting PID controller parameters such as Artificial Bee Colony Algorithm(ABC) [17], [18], Flower Pollination Algorithm (FPA) [19], [20], Sine Cosine Algorithm (SCA) [21], Grey Wolf Optimizer (GWO) [22], [23], Genetic Algorithm (GA) [24], Water Wave Optimization (WWO) [25], Firefly Algorithm(FA) [26], [27], Bat Algorithm (BA) [28], [29], Harris Hawk Optimization (HHA) [30], [31], Ant Colony Optimization(ACO) [32], [33], Particle Swarm Optimization (PSO) [34], [35], Reinforcement Learning (RL) [36], [37]. In addition, this article [38] presents a synthesis of the algorithms used for controlling DC motors via PID controllers, along with recent publications in renowned journals addressing this subject. The article explores various methods of optimizing the control of DC motors, focusing on the use of PID controllers.

Given the variety of optimization algorithms, comparing these algorithms is essential for several reasons. First, it allows evaluating their efficiency and performance, as different algorithms may converge to different solutions in terms of speed and quality [39]. Second, it considers how sensitive these algorithms are to specific system characteristics, such as nonlinearities or constraints, to choose the most suitable one. Additionally, it assesses algorithm robustness, i.e., their ability to provide quality solutions under various problem conditions and configurations. Finally, it takes into account computational complexity by evaluating computation time and required resources to strike a balance between accuracy and efficiency [40].

Several studies have compared these approaches, including GA_ACO_PSO [41], ABC, Bat_GWO_PSO [42], GA_ABC_PSO [43]. Further research has confirmed the stability of the results obtained by the ACO, PSO, and FPA algorithms compared to other methods. To confirm the stability of these algorithms and justify our choice, we propose to give the most important comparisons:

FPA, in particular, is inspired by the natural process of flower pollination, using biological concepts to solve optimization problems. This allows for the effective modeling of solutions for complex systems. FPA effectively balances exploration (global search) and exploitation (local search), which is crucial for avoiding local minima and converging to optimal solutions. Additionally, FPA is flexible and can be applied to a wide range of optimization problems. In comparison with other algorithms:

- Genetic Algorithms (GA) [44] are powerful but can be slow and suffer from premature convergence. FPA, with its unique pollination mechanism, often overcomes these limitations.

PSO is easy to understand and implement, requiring fewer parameters to adjust compared to other algorithms like GA. It is known for its rapid convergence, especially in problems where the objective function is relatively smooth. PSO is inspired by the social behavior of birds, where each particle adjusts its position based on its own experience and that of other particles, facilitating the discovery of optimal solutions. PSO is robust and can effectively handle multi-dimensional and nonlinear problems. In comparison with other algorithms:

- Differential Evolution (DE) [45] is powerful but can take more time to converge. PSO tends to converge faster with fewer parameters to adjust.

- Artificial Bee Colony (ABC) [46], inspired by the behavior of bees, is often more complex to implement than PSO, which is easier to adapt due to its simple particle dynamics.

ACO is capable of finding optimal solutions in vast and complex search spaces by effectively exploiting the best solutions found so far. It can be easily adapted to various constraints and specific objectives, making it useful for a variety of industrial and research problems. In comparison with other algorithms:

- GA can sometimes struggle to find optimal solutions for combinatorial problems due to the stochastic nature of crossover and mutation, whereas ACO is often better suited [47].
- Bacterial Foraging Optimization (BFO) [48] is powerful but can be complex to implement and adjust. ACO, with its mechanisms of pheromone deposition and evaporation, is often more intuitive and direct for pathfinding problems.

The choice of FPA, PSO, and ACO algorithms for a comparative study on Arduino is justified by their unique and complementary characteristics:

- FPA: Offers an excellent balance between exploration and exploitation, with high adaptability for various optimization problems.
- PSO: Known for its simplicity and rapid convergence, particularly effective for continuous and nonlinear problems.
- ACO**:** Particularly suited for combinatorial and discrete optimization problems, with effective reinforcement mechanisms.

These three algorithms provide a broad and robust coverage of optimization problem types, allowing for an exhaustive and relevant comparison in various industrial and academic contexts. To this end, we plan to implement a PID controller whose coefficients will be optimized by the FPA algorithm, using an Arduino Mega 2560 board. Subsequently, we will compare the performances of the three algorithms (ACO, PSO, and FPA) based on experimental results applied to PMDC. This approach, still relatively unexplored on hardware controlling a PMDC, is relevant.

Despite the emergence of newer technologies such as induction AC motors and stepper motors, PMDC motors retain an important role in many industrial and commercial sectors [49]-[51]. Indeed, PMDC motors are particularly useful for conveyor systems, industrial robots, and machine tools requiring variable speeds and precise movements, while enabling precise torque control [52], [53]. Thus, the results of simulations and experiments can be more easily applied to real industrial scenarios, such as robotic arms used in assembly, welding, or other repetitive tasks. Arduino controls the PMDC motors responsible for the articulated movements of the robot.

The PID controller, optimized by the optimization algorithm, ensures smooth and precise movements by correcting position and speed errors in real-time. This process aims to improve the accuracy and repeatability of the tasks performed by the robot, thereby increasing the quality and speed of production. PID controllers are used in a wide range of industrial applications, making this study a solid foundation for understanding and improving the performance of practical control systems. Indeed, the use of PID controllers and PMDC motors in a simulation study on Arduino to compare the ACO, PSO, and FPA algorithms is justified by their simplicity, availability, and representativeness of real industrial systems. PMDC motors, in particular, offer considerable advantages in terms of precise control, reduced maintenance, and quick response, making them ideal for a multitude of industrial and experimental applications. To successfully complete this experimental section, Section 2 of the article, which corresponds to the methodology used, first describes the ACO, PSO, and FPA optimization algorithms to provide a clear understanding of these algorithms. Next, it outlines the basic principles for implementing these algorithms and controlling a PMDC motor through the L298n driver on an Arduino Mega 2560 board. The following step involves using MATLAB/Simulink for identifying the transfer function of the PMDC motor.

To better understand the other sections and the importance of these algorithms in the control of PMDC motors, our article highlights their utility by organizing our work into four sections: Section 2 describes the methods used, Section 3 simulates the algorithms with MATLAB/Simulink, Section 3 implements the algorithms on Arduino Mega, and Section 4 presents the conclusion of our study.

2. Method

2.1. Ant Colony Optimizations

[32] ACO is an optimization algorithm based on the behavior of ants searching for optimal paths between their colony and a food source. Inspired by real ant behavior, ACO uses an iterative process where artificial ants deposit "pheromones" along explored paths. Paths with stronger pheromones become more attractive to other ants, promoting the exploration of promising solutions. Over time, this algorithm converges towards optimal or near-optimal solutions.

For an ant k located at node i, the probability of choosing to move towards another node in the network is given by the following relationship:

$$P_{ij}^{k}\begin{cases}\dfrac{(\tau_{ij}^{k})^{\alpha}\quad(\eta_{ij}^{k})^{\beta}}{\sum_{l\epsilon N_i^k}(\tau_{il}^{k})^{\alpha}\quad(\eta_{il}^{k})^{\beta}} & if\ j\epsilon N_i^k \\ 0 & if\ j\notin N_i^k\end{cases} \tag{1}$$

Where τ_{ij}^{k} represents the pheromone levels for ant k at node i, where the denominator is a summation to encompass all potential paths, with N_i^k denoting the set of possible trails for ant k when located at node i.

Parameters α and β influence the pheromone evaporation process and the behavior of ants. Specifically, α regulates the significance of pheromone quantity in ant decision-making for their subsequent steps. Conversely, β governs the importance of heuristic information in the decision-making process. A higher α accentuates the impact of pheromone quantity, while a higher β emphasizes the significance of heuristic information. Conversely, lower α values reduce the influence of pheromone, promoting exploration of new paths, whereas lower β values increase reliance on heuristic information.

The selection of α and β is contingent upon the desired convergence speed of ants toward an optimal solution. These values aim to strike a balance between exploration (seeking new paths) and exploitation (utilizing known information).

$$\tau_{ij}\leftarrow(1-\rho)\tau_{ij}\ avec\ 0\leq\rho\leq1 \tag{2}$$

Where η_{ij}^{k} heuristic information (visibility), ρ is the percentage of pheromone vaporization (evaporate rate). When ants traverse a path, the pheromone levels are updated as follows:

$$\tau_y\leftarrow\tau_y+\sum_{k=1}^{m}\Delta\tau_y^k \tag{3}$$

With

$$\Delta\tau_{ij}^{k}=\frac{1}{C^k} \tag{4}$$

The reward for ant k is linked to C^k for choosing this path.

The optimal selection of K_i, K_P and K_d ensures that the objective function ITAE (Integral of Time-weighted Absolute Error) is minimized:

$$ITAE = \int_0^t t|e|dt \tag{5}$$

Where ζ is a parameter introduced to reinforce pheromone levels as ants approach optimal solutions. To use ACO, we must proceed with the following steps:

- Step 1: Define a search space that includes the possible solutions for the optimization parameters K_i, K_P and K_d.
- Step 2: Initialize the number of ants and iterations, and set all discrete values in the search space to the same initial pheromone value τ.
- Step 3: Calculate the probability (1). Choose α and β; in our case, $\alpha = 1$ and $\beta = 0$ provide a good balance between exploration and exploitation.
- Step 4: For each ant, determine K_i, K_P and K_d for this we must identify the best value f_{best} the worst value f_{worst} based on the objective function ITAE.
- Step 5: Repeat the same procedure for several iterations

Pheromone should be added for f_{best}

$$\tau_j^{new} = \tau_j^{old} + \sum_k \Delta\tau_j^{(k)} \tag{6}$$

Reduced for f_{worst}.

$$\tau_j^{new} = (1-\rho)\tau_j^{old} \tag{7}$$

Fig. 1 outlines the steps to obtain the optimal solution in the form of a flowchart.

The parameter adjustment phase is very complex and requires several attempts to find the right settings. For ACO, choosing a larger population N (number of ants in the colony) can explore more of the search space but increases computation time. It is wise to start with a moderate number and adjust based on the complexity of the problem and available computational resources.

Similarly, increasing the number of iterations allows for more thorough exploration and refinement but requires more time. To adjust this parameter, it is best to determine it based on the convergence rate observed during initial trials. If improvements slow significantly, consider stopping earlier.

Start with $\alpha = 1$. Adjust upwards if the algorithm converges too slowly and downwards if it converges too quickly and lacks exploration. Next, start with $\beta = 2$. Increase it if the problem has strong heuristic information and decrease it if the heuristic information is less reliable. Typical values for the evaporation rate (ρ) range from 0.1 to 0.7. Adjust based on the required balance between exploration and exploitation.

For the pheromone reinforcement coefficient (ζ), generally use small reinforcement values initially and adjust based on observed performance.

2.2. Particle Swarm Optimization

PSO is an optimization algorithm based on the behavior of flocks of birds or swarms of insects. In PSO, a population of potential solutions, called "particles," is iteratively improved by moving through the search space based on their own experience and that of their neighbors. Each particle has a position and a velocity in the search space. The particles adjust their velocity and position based on the performance of the best solutions found by themselves and their neighbors. PSO is often used for

continuous or discrete optimization problems and is known for its conceptual simplicity and ease of implementation. To use PSO, we must proceed with the following steps:

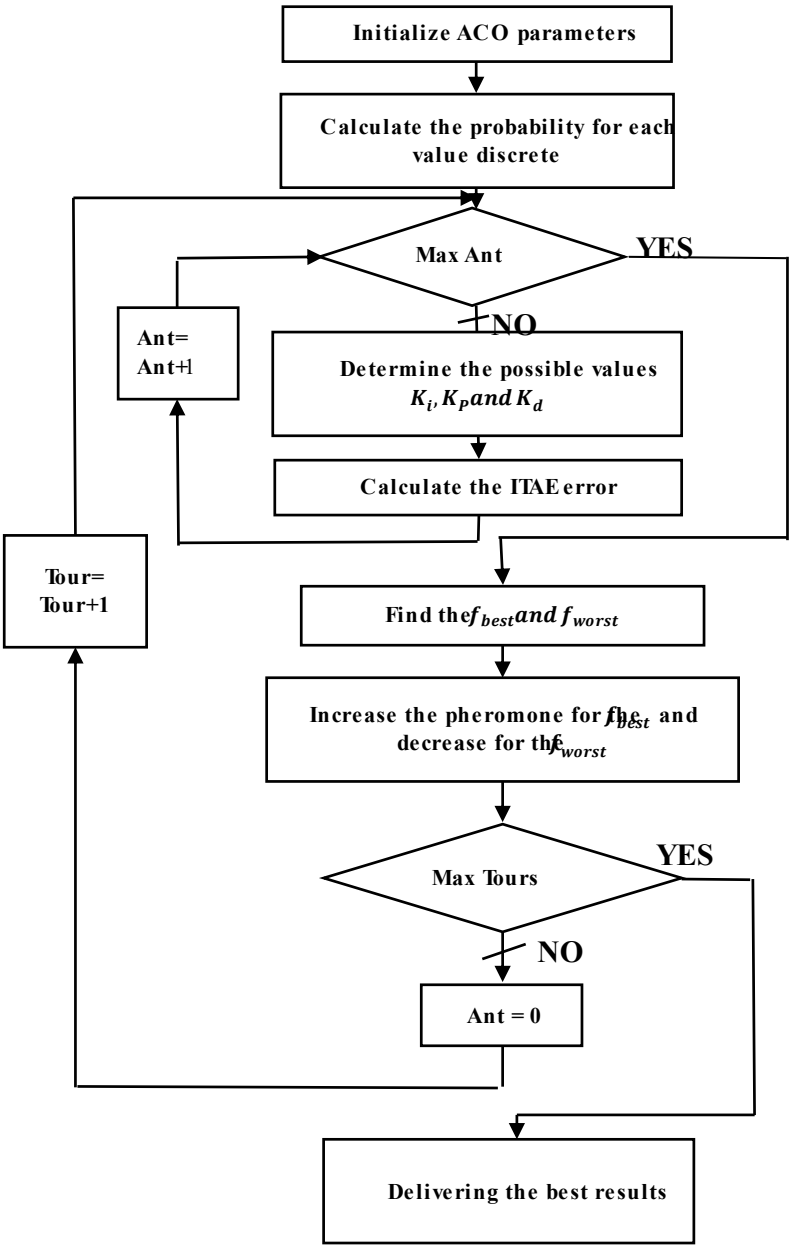

Fig. 1. Flowchart for ACO algorithm

Step 1:

- Initial population (number of particles N)
- Initial position (x) and velocity (v)
- Assign p_{best} and g_{best}

With p_{best}: local optimal solution and g_{best}: global optimal solution.

Step 2: Update velocity and position of each particle

$$v_i(t) = \theta v_i(t-1) + c_1 r_1 \left(p_{best,i} - x_i(t-1)\right) + c_2 r_2 \left(g_{best,i} - x_i(t-1)\right) \quad (8)$$

$$x_i(t) = x_i(t-1) + v_i(t) \quad (9)$$

With: θ is inertia weight, $c_1 and\ c_2$ are individual and social cognitive,

$r_1\ and\ r_2$ are uniformly distributed random numbers in the rang (0,1).

Step 3: Evaluate the objective function ITAE and update p_{best} and $g_{best.}$

Step 4: Repeat the same procedure for several iterations.

[34] The flowchart provides an overview of the steps to optimize using PSO Fig. 2.

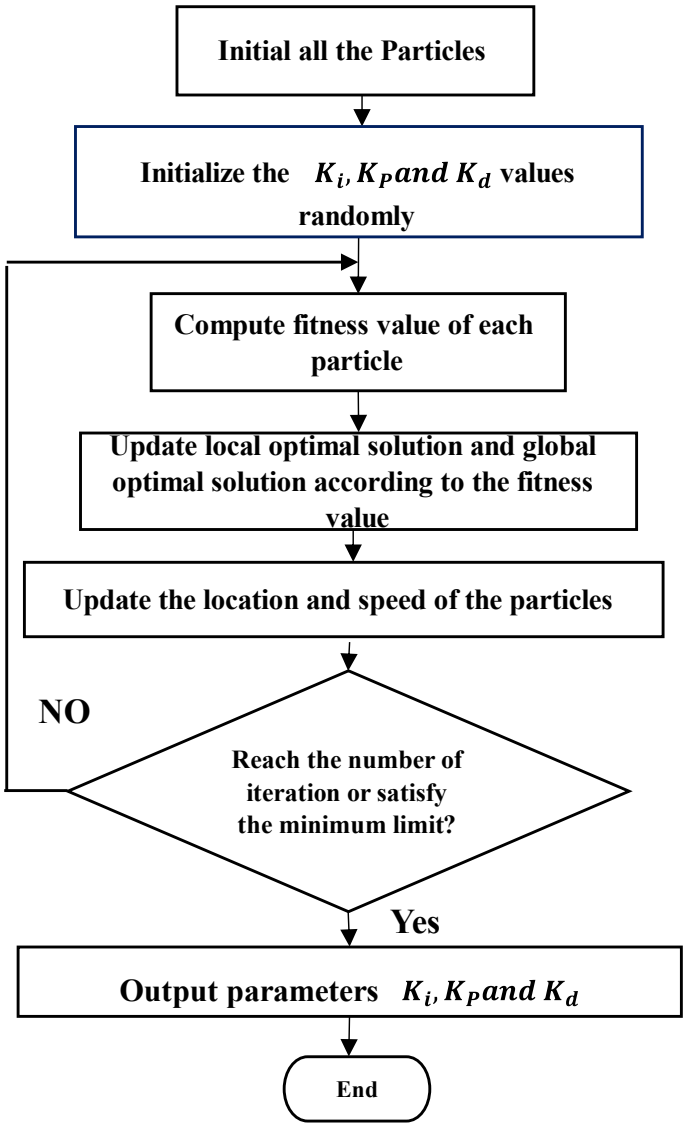

Fig. 2. Flowchart for PSO algorithm

To adjust the PSO parameters, the number of particles and iterations should be moderate at the beginning.

Start with the cognitive parameter (C1), which influences the personal experience of each particle in updating its position. Higher values increase the importance of the individual component, favoring exploration. Adjust (C1) based on the need for individual exploration.

Next, adjust the social parameter (C2), which influences the collective experience of the group of particles in updating their positions. Adjust (C2) based on the need for collective exploration.

Finally, the inertia weight (θ) influences the previous velocity of a particle on its current velocity. Typical values range from 0.4 to 0.9. A decreasing inertia weight, which diminishes over iterations, is often used to balance exploration and exploitation.

2.3. Flower Pollination Algorithm

FPA is a bio-inspired optimization algorithm based on the process of flower pollination in nature. In FPA, each potential solution is considered as a "flower," and the pollination processes between flowers are simulated to search for optimal solutions. Flowers attract pollinators (such as bees, butterflies, etc.) by releasing attractive chemical substances. These pollinators carry pollen between flowers, facilitating reproduction and diversification of plant species. In FPA, solutions are updated based on the quality of flowers and interactions between them. FPA is often used for continuous and discrete optimization problems and is appreciated for its simplicity and robustness. [54] The following steps describe the FPA algorithm:

a) Step1:

- Initialize the population of flowers (N) based on the following equation:

$$X_i = L_b + Rand \times (U_b - L_b) \tag{10}$$

Where $[L_b, U_b]$is the search space for optimal solutions.

- Determine the objective function for each flower, then select the flower for the best solution X_g.

b) Step 2: Select a random number (Rand) for each flower, if the number is less than the probability ρ, then generate a step σ and apply global pollination.

$$X_i^{t+1} = X_i^t + \sigma(X_i^t - X_g) \tag{11}$$

Where X_i^t is the pollen i or the solution vector X_i at the t^{th} iteration.

X_g: the current best solution among the current options in the current iteration.

c) Step 3: For numbers greater than the probability ρ, apply local pollination.

$$X_i^{t+1} = X_i^t + \epsilon(X_j^t - X_k^t) \tag{12}$$

Where X_j^t, X_k^t represent the pollens of different flowers of the same species. ϵ belongs to the interval [0,1].

d) Step 4: In case the new solutions are more promising, they should be replaced and repeat the steps for all populations to find the current solution.

To adjust the FPA parameters, start with moderate values for the population size and the number of iterations. Then, adjust according to the need for global exploration (p) or local exploration (1-p). If the algorithm converges too slowly, increase (p). If the algorithm converges too quickly without finding good solutions, decrease (p). Fig. 3 represents the flowchart that describes the steps to follow to achieve an optimal solution.

2.4. PID Controller

PID Controller One of the key objectives of this study is to optimize the performance of the PID controller by determining coefficients that ensure optimal rise time, settling time, overshoot, and accuracy. To achieve this, we have chosen to use three optimization algorithms: ACO, PSO, and FPA. These algorithms aim to adjust the parameters of the PID, whose transfer function is as follows:

$$\frac{U(s)}{E(s)} = k_p + \frac{k_i}{s} + k_d s \tag{13}$$

With k_p representing the proportional coefficient, k_i the integral coefficient, k_d the derivative coefficient, U the control signal, and E the error between the actual value and the setpoint.

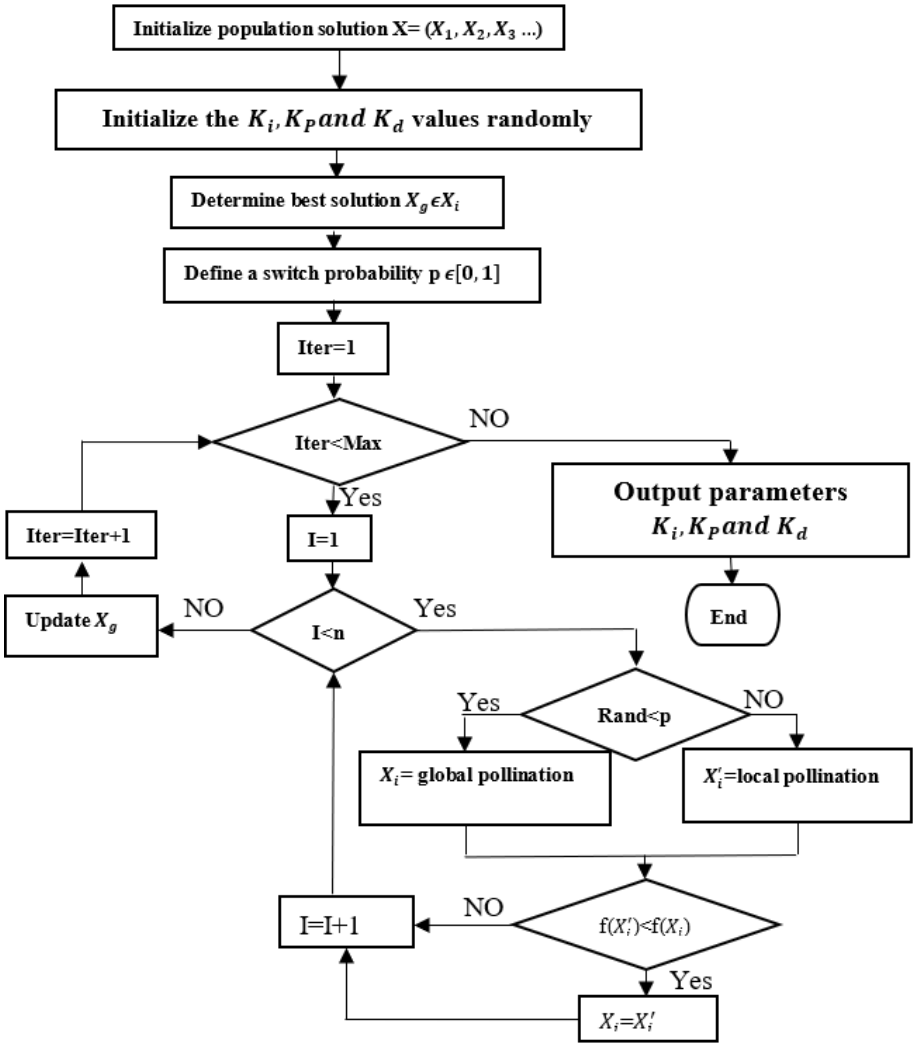

Fig. 3. Flowchart for FPA algorithm [47]

Fig. 4 depicts the functional diagram of a speed control system for a PMDC motor using an Arduino Mega 2560 board. In this system, a PID controller optimized by optimization algorithms is implemented on real hardware. The Arduino board sends a Pulse Width Modulation (PWM) signal to the L298n driver, which in turn provides an optimal voltage to the PMDC motor to track the speed setpoint. A Hall effect speed sensor sends pulses to the Arduino board via a feedback loop, allowing the board to convert them into a numeric value expressing the speed in revolutions per minute (RPM). The error is deduced by subtracting the speed expressed in RPM from the setpoint.

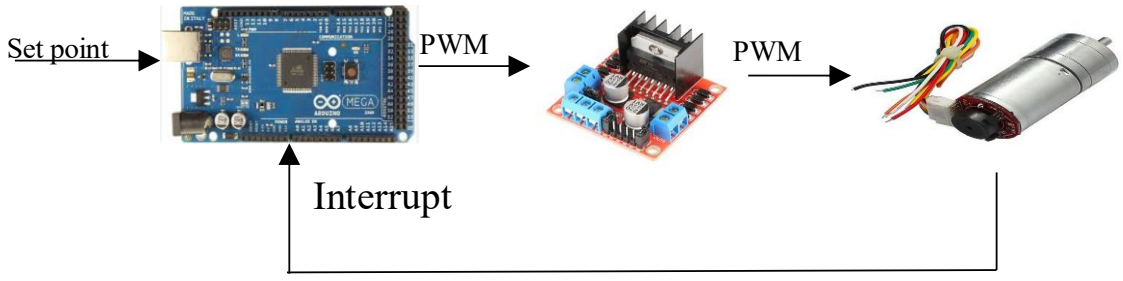

Fig. 4. Block diagram of a speed control

2.5. Identification

To validate these optimization algorithms and determine the coefficients of the PID controller, we need to go through the model identification process using the MATLAB Identification Toolbox, which provides a series of powerful tools for identifying models of dynamic systems from experimental data. This process involves estimating the parameters of a mathematical model using measurements or observations from the real system.

We will model the transfer function of the PMDC motor and the L298N driver. To do this, we will create a model in MATLAB/Simulink that will be implemented on Arduino. This model will take input from the Hall effect speed sensor pulse and produce a PWM signal as output. Subsequently, measurements of different speed values for various duty cycle settings are sent to MATLAB, as shown in Fig. 5 and Fig. 6.

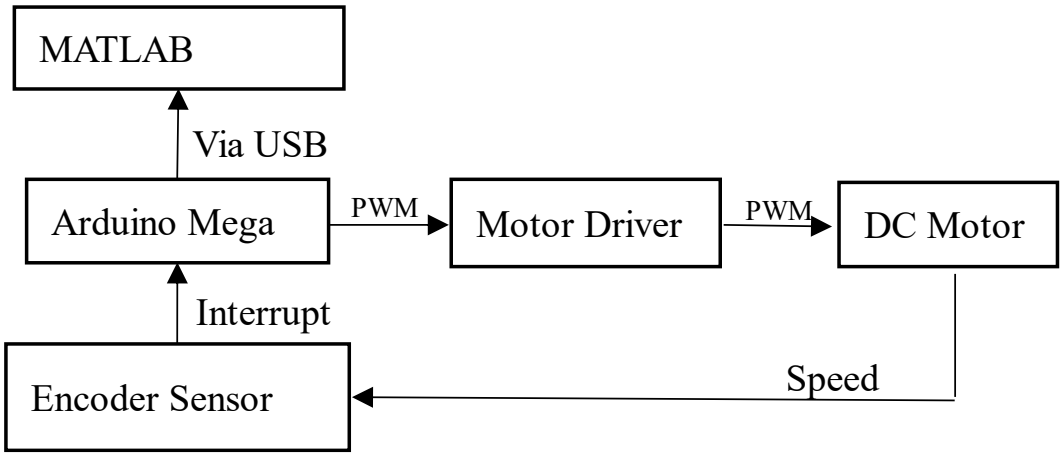

Fig. 5. Block diagram of the identification function

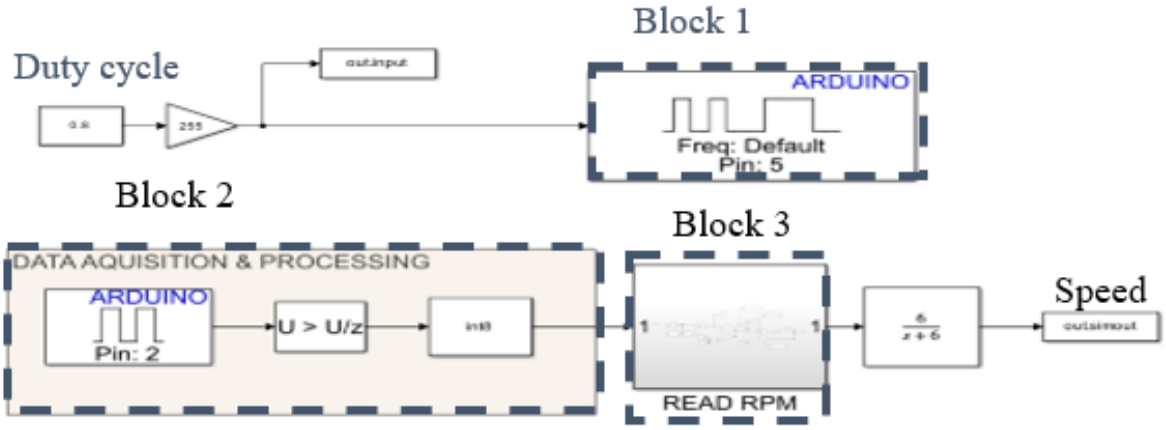

Fig. 6. Block diagram of the identification function in MATLAB/Simulink

Block 1 sends a PWM signal through pin 5 of the Arduino board to the L298n driver, which in turn sends a PWM signal to the PMDC motor. Block 2 enables the acquisition of pulses generated by the speed sensor, connected to interrupt pin 2 of the Arduino. This block also detects rising edges and converts them into integer values. Block 3, illustrated in Fig. 7, calculates the speed in RPM. To do this, we first count the detected pulses by incrementing a counter at each detected rising edge. We then consider only the last 100 samples. At this stage, we have the number of pulses, but to convert it to RPM, we use the following formula:

$$RPM = \frac{N \times 1000 \times 60}{T \times P} \tag{14}$$

Where N represents the number of counted pulses, T is the time required to count the 100 samples, and P is the number of points per revolution of the sensor. This formula is represented as a block in Fig. 7.

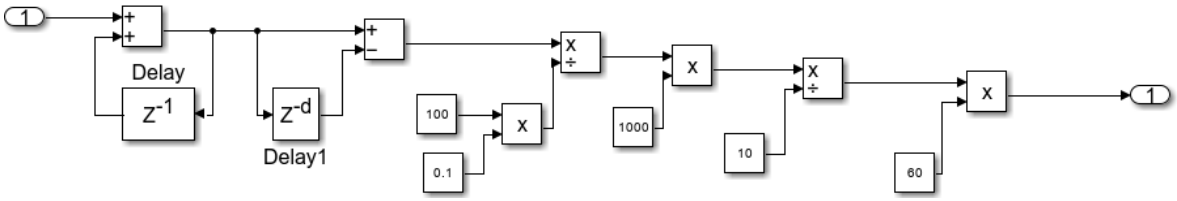

Fig. 7. Diagram for reading speed in RPM in MATLAB/Simulink

The core principle of system identification revolves around deriving an approximate mathematical pattern based on collected experimental data, representing the system's input and output. In MATLAB, this procedure can be facilitated using one of the software's built-in toolboxes, as depicted in Fig. 8. This toolbox comprises various functions tailored for the system identification process, including preprocessing and estimation. Users can import experimental data directly into the toolbox's interface and choose the mathematical operations required for the system identification algorithm. Additionally, the toolbox offers functions to analyze the performance of the estimated system model, rendering it a comprehensive solution for system identification tasks.

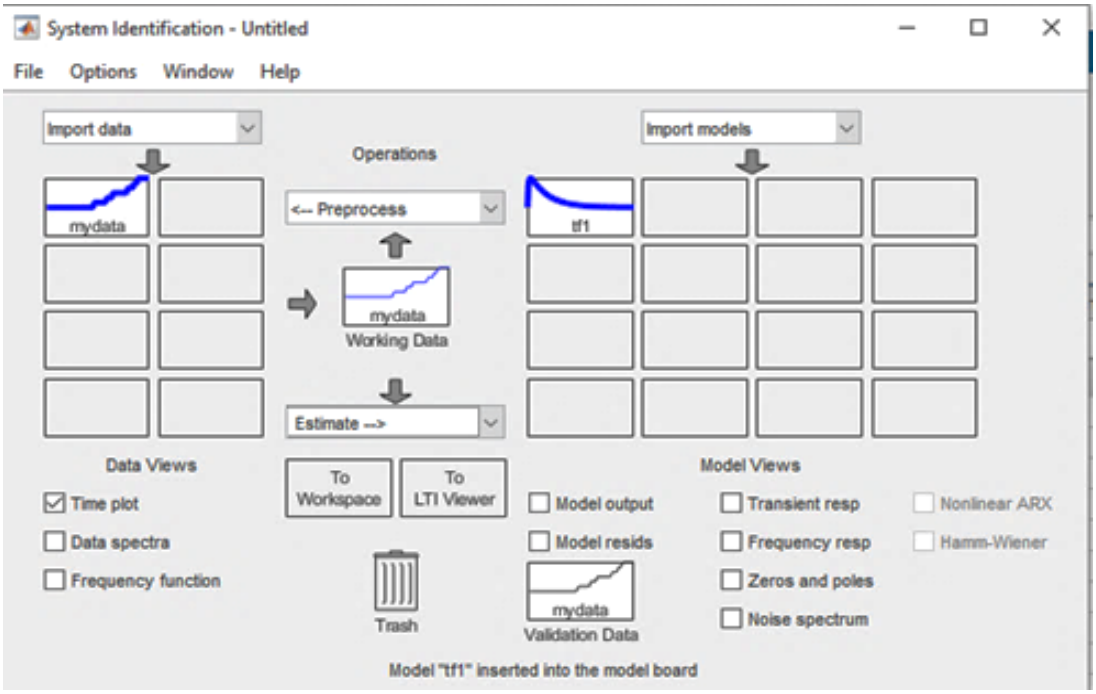

Fig. 8. MATLAB system identification toolbox

After collecting the data (duty cycle and speed) and storing it in MATLAB's Workspace, we use the System Identification Toolbox to estimate the transfer function of the system to be controlled. Multiple attempts are necessary to achieve a good estimation of the transfer function. The results of this estimated transfer function are then compared in a closed-loop simulation against real-world operations using the actual system. Adjusting the transfer function derived from the toolbox is essential to closely match the real-world behavior. The transfer function of the PMDC deduced from the identification box is:

$$\frac{700.3}{s^2 + 25.11s + 12} \tag{15}$$

3. Simulation

The simulation in MATLAB aims to determine the optimal coefficients of the PID controller using optimization algorithms (ACO, PSO, and FPA) based on the transfer function deduced from the identification tool. Fig. 9 illustrates the work carried out in MATLAB/Simulink.

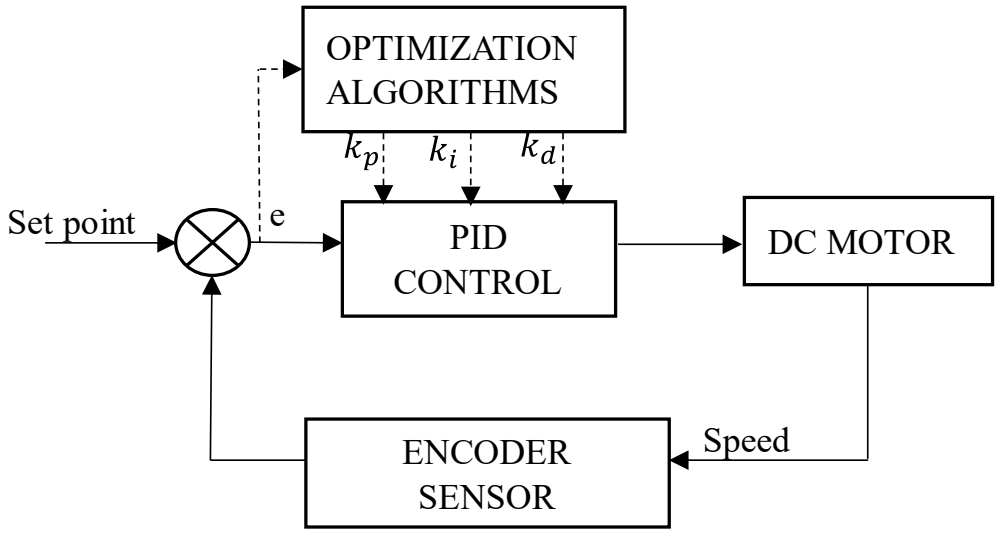

Fig. 9. Block diagram of PID control with optimization algorithms

Table 1, Table 2, and Table 3 elucidate the parameters of the algorithms (ACO, PSO, and FPA) used to determine the optimized coefficients k_p, k_i *and* k_d to achieve a minimal ITAE objective function.

Table 1. ACO Parameter values

Description and Symbol	Values
Population size(N)	100
Maximum number of iteration (Iter)	120
Alpha(α)	1
Beta(β)	0
Evaporation rate(ρ)	0.6
Step to obtain an optimal solution(h)	0.01
Reinforce pheromone(ζ)	2

Table 2. PSO Parameter values

Description and Symbol	Values
Population size(N)	80
Maximum number of iteration (Iter)	90
Cognitive Parameter(C1)	2
Social parameter(C2)	2

Table 3. FPA Parameter values

Description and Symbol	Values
Population size(N)	80
Maximum number of iteration (Iter)	100
Probability of switching (p)	0.7

Fig. 10 compares the performance of three algorithms: ACO, FPA, and PSO. The PSO algorithm shows a response time of 0.768s, a rise time of 0.102s, and an overshoot of 13%. In comparison, the FPA algorithm exhibits a faster response time at 0.347s with an overshoot of 6.4%, which is better than PSO in certain aspects. On the other hand, ACO achieves a response time of 0.322s with an

overshoot of only 0.68%, demonstrating its efficiency compared to the other two algorithms (PSO and FPA).

Although ACO has proven effective in optimization compared to PSO and FPA, it requires more time and its numerous parameters make the choice of values more complex. In comparison, PSO and FPA are easier to manage due to their simplicity and fewer parameters.

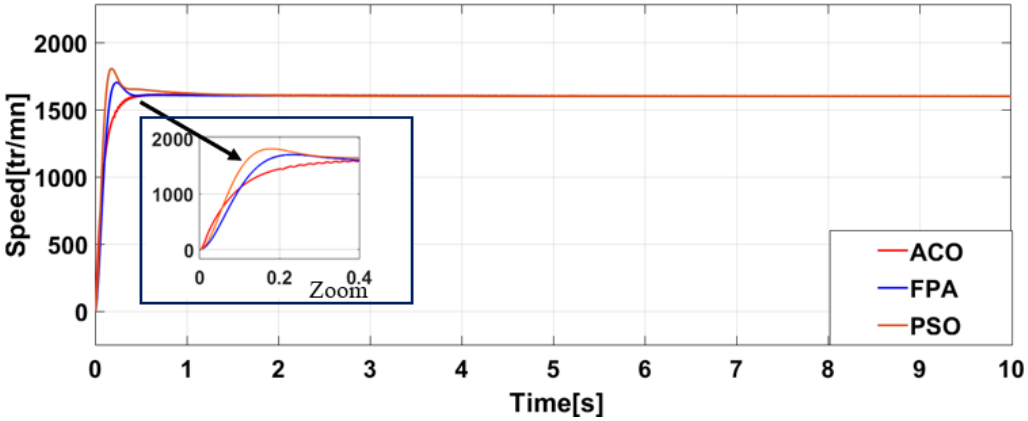

Fig. 10. Speed response by MATLAB/Simulink

Table 4 summarizes the performance obtained using the three optimization algorithms.

Table 4. Results of simulation of PSO, FPA and ACO

Method	PID Controller Parameters			Rise Time	Settling Time	Overshoot
	k_p	k_i	k_d	tr (sec)	ts (sec)	MP (%)
PSO	4.5	6.48	0.03	0.102	0.768	13
FPA	3	1.758	0.000127	0.138	0.347	6
ACO	8	2	0.4	0.196	0.322	0.68

4. Experimentation

In this practical phase, we will test these optimization coefficients on real hardware by integrating them into the embedded controller of an Arduino Mega board. The part implemented on Arduino is illustrated in Fig. 11 and Fig. 12. as well as the PMDC motor with characteristics mentioned in Table 5.

Fig. 11. Experimental work illustration

Table 5. Parameters and corresponding values of PMDC

Parameter and Symbol	Values
Moment of Inertia (J)	19e-7$kg.m^2$
Friction coefficient (f)	52.77e-6N.ms
Back EMFconstant(k_e)	0.0229V/rad. s^{-1}
Torque constant (k_t)	0.0214N.m/A
Electric resistance(R)	8Ω
Armature inductance(L)	10e-03H

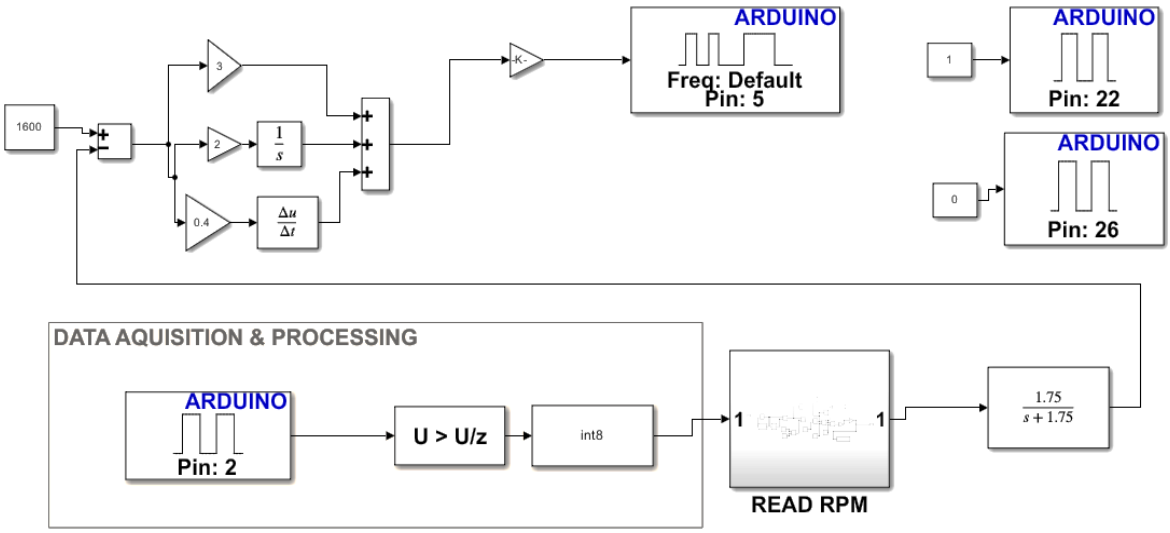

Fig. 12. Block diagram of a speed control in MATLAB/Simulink

This practical phase will allow us to effectively evaluate the comparison between the three algorithms (ACO, PSO, and FPA), taking into account all nonlinearities and uncertainties of all components in the control chain.

To start with, we tested the following optimization coefficients: k_p=2.5, k_i=2, and k_d=0.4, which are very close to those of ACO with a set point of 1600 rpm. Fig. 13 shows the motor speed, which exhibits fluctuations that could affect speed control. Therefore, we decided to use a low-pass filter, whose cutoff frequency was adjusted through trial and error. Additionally, we made slight modifications to the optimization coefficients (k_p=3, k_i=2, and k_d=0.4) since the identified function represents an approximation of the system. Fig. 14 illustrates the adjustments made to the control system and the positive effect of the low-pass filter on motor speed. Speed sensors can be affected by electrical noise and other disturbances, creating rapid and erratic fluctuations in the speed signal. The low-pass filter attenuates the high-frequency components of the speed measurement signal, reducing noise and providing a cleaner signal for the PID controller. A more stable speed measurement allows the PID controller to operate more effectively, improving the accuracy and stability of speed control.

Integrating a low-pass filter in the speed control of a PMDC motor offers significant advantages in terms of noise reduction, improved stability, and reduced mechanical wear. However, it is crucial to manage the delays introduced by the filter and choose an appropriate cutoff frequency to balance noise reduction and system responsiveness. By adopting an experimental and iterative approach, it is possible to leverage the benefits of filtering while minimizing its drawbacks.

Based on the optimization coefficients obtained through the iterations using the algorithms (ACO, PSO, and FPA) in MATLAB/Simulink, we were able to control the speed of the PMDC on real hardware, with some adjustments to kp, ki, and kd considering that the transfer function in MATLAB is an approximation of the real system. Fig. 15 highlights this control, and Table 4 summarizes the control performance.

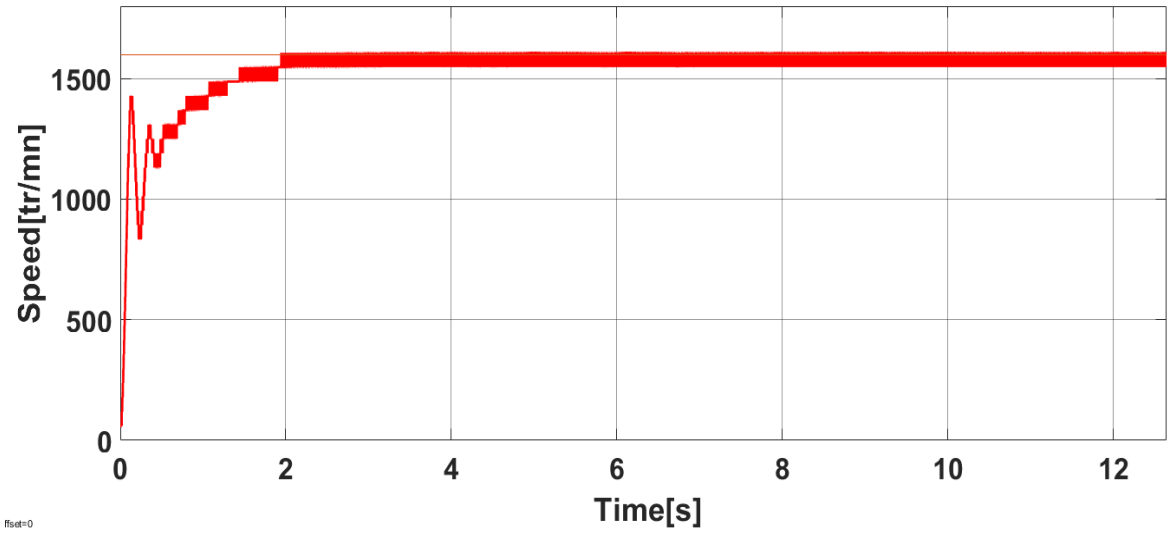

Fig. 13. Speed response by hardware without low-pass filter

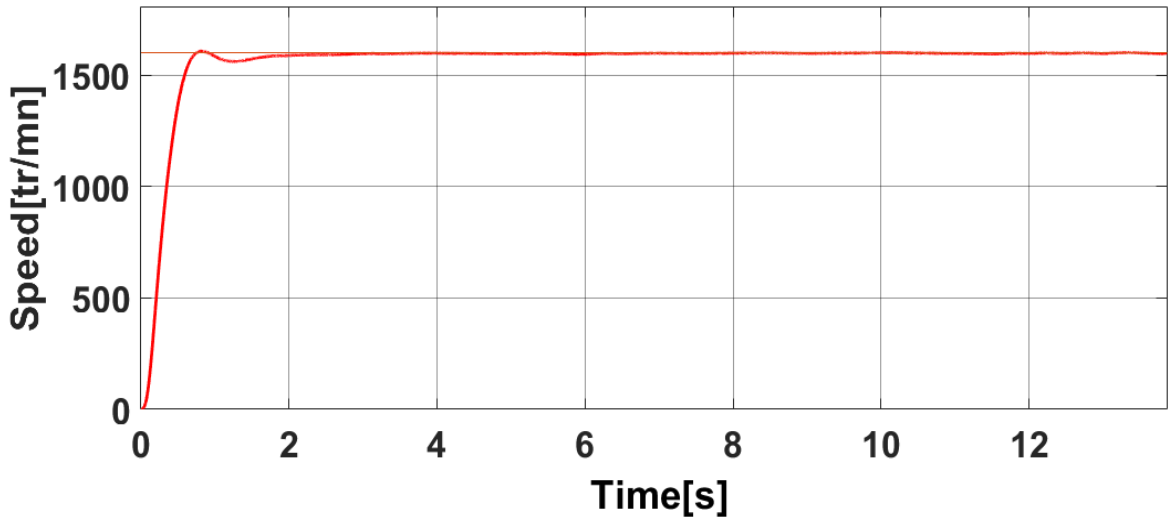

Fig. 14. Speed response by hardware with low-pass filter

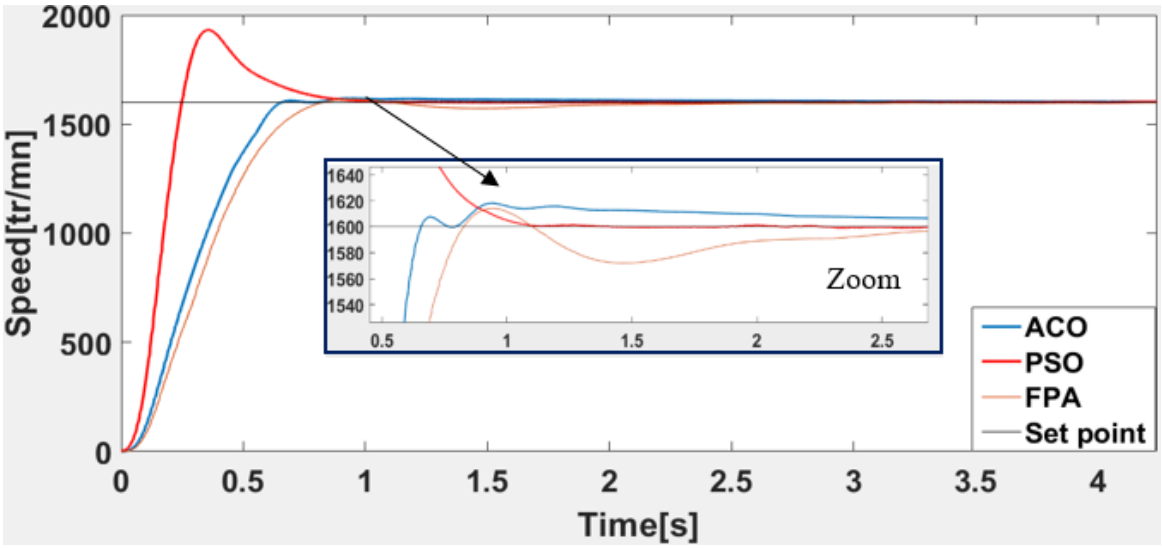

Fig. 15. Speed response by hardware of ACO, PSO and FPA

The results of the experiment show a close correlation between the simulation results and those obtained through PSO optimization. The system optimized by PSO demonstrates a response time of 0.753 seconds, an overshoot of 19%, and a rise time of 0.204 seconds. In comparison, using FPA yields a response time of 0.644 seconds with an overshoot of only 0.81%. These performances are remarkably close to those achieved with ACO, which shows a response time of 0.596 seconds and an overshoot of 1.68%. Table 6 shows these results.

Table 6. Comparative performance of PSO, FPA and ACO

Method	PID Controller Parameters			Rise Time tr (sec)	Settling Time ts (sec)	Overshoot MP (%)
	k_p	k_i	k_d			
PSO	1.6	6.48	0.1	0.204	0.753	19
FPA	0.95	1.758	0.000127	0.520	0.644	0.81
ACO	4.2	2	0.4	0.534	0.596	1.68

The ACO algorithm exhibits lower overshoot and a reduced response time compared to PSO, but is relatively close to FPA. A possible explanation is that ACO is known for its good balance between exploration and exploitation. The ants deposit pheromones that guide the search for solutions while exploring new possibilities. This balance allows ACO to find optimal solutions quickly and stably, which can explain the low overshoot and rapid response time. Additionally, ACO enables dynamic adaptation based on pheromones, helping to avoid suboptimal solutions and converge more quickly to an effective solution.

PSO focuses heavily on exploiting the best solutions found by the particles, which can result in higher overshoot if the particles prematurely converge on a local suboptimal solution. This can also lengthen the response time when adjustments are needed to move away from these local solutions.

FPA uses both global and local pollination strategies, allowing it to balance exploration and exploitation well. The similar results between ACO and FPA suggest that both algorithms effectively manage the balance between exploring new solutions and exploiting known ones.

Table 7 consolidates the simulation and experimental sections. This comparison helps to better understand the performance differences between the simulation and experimental setups and to closely examine the causes of these result discrepancies.

Table 7. Simulation and experimentation results

Method	PID Controller Parameters			Rise Time tr (sec)	Settling Time ts (sec)	Overshoot MP (%)
	k_p	k_i	k_d			
PSO (experimental)	1.6	6.48	0.1	0.204	0.753	19
PSO (simulation)	4.5	6.48	0.03	0.102	0.768	13
FPA (experimental)	0.95	1.758	0.000127	0.520	0.644	0.81
FPA (simulation)	3	1.758	0.000127	0.138	0.347	6
ACO (experimental)	4.2	2	0.4	0.534	0.596	1.68
ACO (simulation)	8	2	0.4	0.196	0.322	0.68

Several factors explain the differences between simulation and experimental results:

- Latency and Response Time: Calculation and response times in a simulation environment are often ideal and without latency. On an Arduino board, processing times can introduce delays that affect the PID control performance.
- Noise and Disturbances: Simulations may neglect or minimize the impact of noise and disturbances on the sensors and the L298n driver. In practice, electrical noise, mechanical vibrations, and other disturbances can degrade control accuracy.

- Sensor Resolution and Accuracy: Sensors in simulations can be modeled with ideal precision. In reality, sensors have limitations in terms of resolution and accuracy, which can affect PID performance.
- Hardware Limitations: Simulations do not always account for the hardware limitations of the Arduino and peripheral components. Power constraints, processing capacity, and limited memory can impact PID control performance.
- Non-linearities and Saturation: Simulations may not capture all system non-linearities or the effects of saturation in switches. In practice, PMDC motors may exhibit non-linear behaviors and reach saturation limits that degrade performance.

Based on this experiment, the choice of the FPA and ACO algorithms is justified over the PSO approaches. Additionally, FPA has the advantage of being simpler to use, with fewer parameters to adjust.

This article can lead us towards a potential research avenue, such as a hybrid FPA-ACO algorithm, by leveraging the strengths of both algorithms. The global search capability of FPA to avoid local minima, combined with ACO's ability to refine solutions through pheromone-guided search, potentially leads to faster and more precise convergence, making it a promising solution for complex control tasks.

5. Conclusion

This study highlights the importance of comparing optimization algorithms to help researchers select the most suitable one for a given situation. The experimental results show that the ACO algorithm, with a response time of 0.596s and an overshoot of 1.68%, and the FPA algorithm, with a response time of 0.644s and an overshoot of 0.81%, are more effective in tuning PID parameters to achieve reduced response times, minimal overshoot, and increased accuracy in controlling the speed of the PMDC compared to the PSO algorithm. Although the performance obtained with PSO-optimized PID parameters is acceptable, with a response time of 0.768s and an overshoot of 19%, the results of FPA and ACO are similar. However, FPA is preferable to ACO due to its simplicity and faster implementation. These findings can be beneficial for applications similar to our study, such as robotic arms used in assembly, welding, or other repetitive tasks, and industrial pumps used for fluid transfer in manufacturing processes.

The practical part addressed the speed ripples, which have very direct consequences on the motor and control accuracy, by mitigating them with a low-pass filter in the feedback loop, while managing the delays introduced by the filter and choosing the appropriate cutoff frequency to balance noise reduction and system responsiveness.

We plan to extend this study to more complex nonlinear systems to evaluate the limits and effectiveness of each algorithm, also incorporating a practical component into our approach.

Author Contribution: Adil Najem: Conceptualization, methodology, experimentation. Ahmed Moutabir: Writing, preparation of the original version. Abderrahmane Ouchatt: Supervision.

Funding: No external funding was allocated to this research.

Acknowledgments: We express our gratitude to the members and leaders of the “Laboratory G.E.I.T.I.I.L” for their insightful comments and clear-sighted recommendations.

Conflicts of Interest: The authors certify the absence of known conflicting financial interests or personal relationships that could have influenced their work presented in this article.

Link to the Video Showing Part of the Experiment:
youtu.be/MNiM9PGHX1s?si=Jnk7FGNCSR0FWrJt

References

[1] V. Kumarasamy, V. KarumanchettyThottam Ramasamy, G. Chandrasekaran, G. Chinnaraj, P. Sivalingam, and N. S. Kumar, "A review of integer order PID and fractional order PID controllers using optimization techniques for speed control of brushless DC motor drive," *International Journal of System Assurance Engineering and Management*, vol. 14, pp. 1139-1150, 2023, https://doi.org/10.1007/s13198-023-01952-x.

[2] S. B. Joseph, E. Gbenga Dada, A. Abidemi, D. Opeoluwa Oyewola, and M. Khammas, "Metaheuristic algorithms for PID controller parameters tuning: review, approaches and open problems," *Heliyon*, vol. 8, no. 5, p. E09399, 2022, https://doi.org/10.1016/j.heliyon.2022.e09399.

[3] L. Kumar and N. P. Mandal, "Pressure control of fixed displacement variable speed radial piston pump using PID controller," *Materials Today Proceedings*, vol. 56, pp. 1840-1846, 2022, https://doi.org/10.1016/j.matpr.2021.11.034.

[4] M. Z. B. A. Karim, and N. M. Thamrin, "Servo Motor Controller using PID and Graphical User Interface on Raspberry Pi for Robotic Arm," *Journal of Physics: Conference Series*, vol. 2319, no. 1, p. 012015, 2022, https://doi.org/10.1088/1742-6596/2319/1/012015.

[5] A. Wahyudie, T. B. Susilo, C. S. A. Nandar, S. Fayez, and R. Errouissi, "Simple Robust PID Tuning for Magnetic Levitation Systems Using Model-free Control and $\mathscr{H}\infty$ Control Strategies," *International Journal of Control, Automation and Systems*, vol. 19, pp. 3956-3966, 2021, https://doi.org/10.1007/s12555-020-0253-8.

[6] S. Kadry and V. Rajinikanth, "Design of PID Controller for Magnetic Levitation System using Harris Hawks Optimization," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, vol. 6, no. 2, pp. 70-78, 2020, http://dx.doi.org/10.26555/jiteki.v6i2.19167.

[7] D. Bakria, M. Azzouzi, and D. Gozim, "Chaos Control and Stabilization of a PID Controlled Buck Converter Using the Spotted Hyena Optimizer," *Engineering, Technology and Applied Science Research*, vol. 11, no. 6, pp. 7922-7926, 2021, https://doi.org/10.48084/etasr.4585.

[8] D. Izci, B. Hekimoğlu, and S. Ekinci, "A new artificial ecosystem-based optimization integrated with Nelder-Mead method for PID controller design of buck converter," *Alexandria Engineering Journal*, vol. 61, no. 3, pp. 2030-2044, 2022, https://doi.org/10.1016/j.aej.2021.07.037.

[9] M. Waszak and R. Łangowski, "An Automatic Self-Tuning Control System Design for an Inverted Pendulum," *IEEE Access*, vol. 8, pp. 26726-26738, 2020, https://doi.org/10.1109/ACCESS.2020.2971788.

[10] M. F. Q. Say, E. Sybingco, A. A. Bandala, R. R. P. Vicerra and A. Y. Chua, "A Genetic Algorithm Approach to PID Tuning of a Quadcopter UAV Model," *2021 IEEE/SICE International Symposium on System Integration (SII)*, pp. 675-678, 2021, https://doi.org/10.1109/IEEECONF49454.2021.9382697.

[11] R. Khandait, V. Kumar, V. Bhurse, V. Tiwari and S. Khubalkar, "Quadcopter Control using Different Controllers," *2022 International Conference on Intelligent Controller and Computing for Smart Power (ICICCSP)*, pp. 1-6, 2022, https://doi.org/10.1109/ICICCSP53532.2022.9862416.

[12] T. Ahmmed, I. Akhter, S. M. R. Karim, and F. A. S. Ahamed, "Genetic Algorithm Based PID Parameter Optimization," *American Journal of Intelligent Systems*, vol. 10, no. 1, pp. 8-13, 2020, https://doi.org/10.5923/j.ajis.20201001.02.

[13] Y. Hong, C. Fu, and B. Merci, "Optimization and determination of the parameters for a PID based ventilation system for smoke control in tunnel fires: Comparative study between a genetic algorithm and an analytical trial-and-error method," *Tunnelling and Underground Space Technology*, vol. 136, p. 105088, 2023, https://doi.org/10.1016/j.tust.2023.105088.

[14] 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, pp. 1-4, 2023, https://doi.org/10.1109/LSENS.2023.3234400.

[15] A. A. Abd Samat, M. A. Subani, N. F. Ab Aziz, N. A. Salim, K. Daud and A. I. Tajudin, "PSO-Based PI Controller for Speed Control Of DC Motor," *2022 IEEE International Conference on Power and Energy (PECon)*, pp. 481-486, 2022, https://doi.org/10.1109/PECon54459.2022.9988840.

[16] F. Z. Baghli, Y. Lakhal, and Y. A. E. Kadi, "The Efficiency of an Optimized PID Controller Based on Ant Colony Algorithm (ACO-PID) for the Position Control of a Multi-articulated System," *Journal of Robotics and Control*, vol. 4, no. 3, pp. 289-298, 2023, https://doi.org/10.18196/jrc.v4i3.17709.

[17] H. Du, P. Liu, Q. Cui, X. Ma, and H. Wang, "PID Controller Parameter Optimized by Reformative Artificial Bee Colony Algorithm," *Journal of Mathematics*, vol. 2022, no. 1, pp. 1-16, 2022, https://doi.org/10.1155/2022/3826702.

[18] A. I. Tajudin, M. A. D. Izani, A. A. A. Samat, S. Omar and M. A. M. Idin, "Design a Speed Control for DC Motor Using an Optimal PID Controller Implementation of ABC Algorithm," *2022 IEEE 12th International Conference on Control System, Computing and Engineering (ICCSCE)*, pp. 97-102, 2022, https://doi.org/10.1109/ICCSCE54767.2022.9935644.

[19] Y. Nuthalapati and R. S. R. K. Naidu, "A Novel Modulating PID Controller for a Speed Control of BLDC Motor Adopting Flower Pollination Algorithm," *Intelligent Computing in Control and Communication*, vol. 702, pp. 159-168, 2021, https://doi.org/10.1007/978-981-15-8439-8_14.

[20] T. Chiranjeevi *et al.*, "Control of electric machines using flower pollination algorithm based fractional order PID controller," *Global Transitions Proceedings*, vol. 2, no. 2, pp. 227–232, 2021, https://doi.org/10.1016/j.gltp.2021.08.057.

[21] D. Guha, P. K. Roy, S. Banerjee, S. Padmanaban, F. Blaabjerg and D. Chittathuru, "Small-Signal Stability Analysis of Hybrid Power System With Quasi-Oppositional Sine Cosine Algorithm Optimized Fractional Order PID Controller," *IEEE Access*, vol. 8, pp. 155971-155986, 2020, https://doi.org/10.1109/ACCESS.2020.3018620.

[22] P. Dutta and S. K. Nayak, "Grey Wolf Optimizer Based PID Controller for Speed Control of BLDC Motor," *Journal of Electrical Engineering & Technology*, vol. 16, no. 2, pp. 955-961, 2021, https://doi.org/10.1007/s42835-021-00660-5.

[23] J. Bhookya, M. Vijaya Kumar, J. Ravi Kumar, and A. Seshagiri Rao, "Implementation of PID controller for liquid level system using mGWO and integration of IoT application," *Journal of Industrial Information Integration*, vol. 28, p. 100368, 2022, https://doi.org/10.1016/j.jii.2022.100368.

[24] M. M. Gani, M. Saiful Islam, · Muhammad, and A. Ullah, "Optimal PID tuning for controlling the temperature of electric furnace by genetic algorithm," *SN Applied Sciences*, vol. 1, no. 880, 2019, https://doi.org/10.1007/s42452-019-0929-y.

[25] Y. Zhou, J. Zhang, X. Yang, and Y. Ling, "Optimization of PID Controller Based on Water Wave Optimization for an Automatic Voltage Regulator System," *Information Technology and Control*, vol. 48, no. 1, pp. 160-171, 2019, https://doi.org/10.5755/j01.itc.48.1.20296.

[26] B. N. Kommula and V. R. Kota, "Direct instantaneous torque control of Brushless DC motor using firefly Algorithm based fractional order PID controller," *Journal of King Saud University: Engineering Sciences*,vol. 32, no. 2, pp.133-140, 2020, https://doi.org/10.1016/j.jksues.2018.04.007.

[27] M. Ali, H. Suyono, M. A. Muslim, M. R. Djalal, Y. M. Safarudin, A. A. Firdaus, "Determination of the parameters of the firefly method for PID parameters in solar panel applications," *Sinergi*, vol. 26, no. 2, pp. 265-272, 2022, https://dx.doi.org/10.22441/sinergi.2022.2.016.

[28] D. F. U. Putra, A. A. Firdaus, H. Arof, N. P. U. Putra, and V. A. Kusuma, "Improved load frequency control performance by tuning parameters of PID controller and BESS using Bat algorithm," *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 5, pp. 2624-2634, 2023, https://doi.org/10.11591/eei.v12i5.4548.

[29] S. Tiacharoen, "Optimal Tuning of 2DOF-PID Controllers Using Bat Algorithm," *2023 7th International Conference on Information Technology (InCIT)*, pp. 388-391, 2023, https://doi.org/10.1109/InCIT60207.2023.10412900.

[30] A. Loganathan and N. S. Ahmad, "Robot Path Planning via Harris Hawks Optimization: A Comparative Assessment," *2023 International Conference on Energy, Power, Environment, Control, and Computing*

(ICEPECC), pp. 1-4, 2023, https://doi.org/10.1109/ICEPECC57281.2023.10209484.

[31] S. Ekinci, D. Izci and B. Hekimoğlu, "PID Speed Control of DC Motor Using Harris Hawks Optimization Algorithm," *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pp. 1-6, 2020, https://doi.org/10.1109/ICECCE49384.2020.9179308.

[32] A. Najem, A. Moutabir, A. Ouchatti, and M. El Haissouf, "Experimental Validation of the Generation of Direct and Quadratic Reference Currents by Combining the Ant Colony Optimization Algorithm and Sliding Mode Control in PMSM using the Process PIL," *International Journal of Robotics and Control Systems*, vol. 4, no. 1, pp. 188-216, 2024, https://doi.org/10.31763/ijrcs.v4i1.1286.

[33] S. Chauhan, B. Singh, and M. Singh, "Modified ant colony optimization based PID controller design for coupled tank system," *Engineering Research Express*, vol. 3, no. 4, p. 045005, 2021, https://doi.org/10.1088/2631-8695/ac2bf3.

[34] E. S. Rahayu, A. Ma'arif, and A. Cakan, "Particle Swarm Optimization (PSO) Tuning of PID Control on DC Motor," *International Journal of Robotics and Control Systems*, vol. 2, no. 2, pp. 435-447, 2022, https://doi.org/10.31763/ijrcs.v2i2.476.

[35] A. K. Kashyap and D. R. Parhi, "Particle Swarm Optimization aided PID gait controller design for a humanoid robot," *ISA Transactions*, vol. 114, pp. 306-330, 2021, https://doi.org/10.1016/j.isatra.2020.12.033.

[36] Z. Guan and T. Yamamoto, "Design of a Reinforcement Learning PID Controller," *IEEJ transactions on electrical and electronic engineering*, vol. 16, no. 10, pp. 1354-1360, 2021, https://doi.org/10.1002/tee.23430.

[37] A. Najem, A. Moutabir, M. Rafik and A. Ouchatti, "Comparative Study of PMSM Control Using Reinforcement Learning and PID Control," *2023 3rd International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, pp. 1-5, 2023, https://doi.org/10.1109/IRASET57153.2023.10153024.

[38] S. Oladipo, Y. Sun, Z. Wang, "Optimization of PID Controller with Metaheuristic Algorithms for DC Motor Drives: Review," *International Review of Electrical Engineering*, vol. 15, no. 5, pp. 352-381, 2020, https://doi.org/10.15866/iree.v15i5.18688.

[39] X. Zhang and Q. Zhang, "Optimization of PID Parameters Based on Ant Colony Algorithm," *2021 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, pp. 850-853, 2021, https://doi.org/10.1109/ICITBS53129.2021.00211.

[40] A. Abushawish, M. Hamadeh, A. B. Nassif, "PID Controller Gains Tuning Using Metaheuristic Optimization Methods: A survey," *International Journal of Computers*, vol. 14, pp. 87-95, 2020, https://doi.org/10.46300/9108.2020.14.14.

[41] M. Sreejeth, R. Kumar, N. Tripathi and P. Garg, "Tuning A PID Controller using Metaheuristic Algorithms," *2023 8th International Conference on Communication and Electronics Systems (ICCES)*, pp. 276-282, 2023, https://doi.org/10.1109/ICCES57224.2023.10192687.

[42] M. V. D. Rocha, L. P. Sampaio and S. A. O. D. Silva, "Comparative Analysis of ABC, Bat, GWO and PSO Algorithms for MPPT in PV Systems," *2019 8th International Conference on Renewable Energy Research and Applications (ICRERA)*, pp. 347-352, 2019, https://doi.org/10.1109/ICRERA47325.2019.8996520.

[43] T. O. Ajewole, O. Oladepo, K. A. Hassan, A. A. Olawuyi, O. Onarinde, "Comparative study of the performances of three metaheuristic algorithms in sizing hybrid-source power system," *Turkish Journal of Electrical Power and Energy Systems*, vol. 2, no 2, pp. 134-146, 2022, https://doi.org/10.5152/tepes.2022.22012.

[44] A. Mateen, M. Wasim, A. Ahad, T. Ashfaq, M. Iqbal, and A. Ali, "Smart energy management system for minimizing electricity cost and peak to average ratio in residential areas with hybrid genetic flower pollination algorithm," *Alexandria Engineering Journal*, vol. 77, pp. 593-611, 2023, https://doi.org/10.1016/j.aej.2023.06.053.

[45] M. Dadvar, H. Navidi, H. H. S. Javadi, and M. Mirzarezaee, "A cooperative approach for combining particle swarm optimization and differential evolution algorithms to solve single-objective optimization

problems," *Applied Intelligence*, vol. 52, no. 4, pp. 4089-4108, 2022, https://doi.org/10.1007/s10489-021-02605-x.

[46] G. Zhou, H. Moayedi, M. Bahiraei, and Z. Lyu, "Employing artificial bee colony and particle swarm techniques for optimizing a neural network in prediction of heating and cooling loads of residential buildings," *Journal of Cleaner Production*, vol. 254, p. 120082, 2020, https://doi.org/10.1016/j.jclepro.2020.120082.

[47] C. Yin, Q. Fang, H. Li, Y. Peng, X. Xu, and D. Tang, "An optimized resource scheduling algorithm based on GA and ACO algorithm in fog computing," *The Journal of Supercomputing*, vol. 80, pp. 4248-4285, 2024, https://doi.org/10.1007/s11227-023-05571-y.

[48] A. I. A. Raof, M. S. Hadi, A. Jamali, H. M. Yatim, M. H. A. Talib and I. Z. M. Darus, "Intelligent PID Controller Tuned by Bacterial Foraging Optimization Algorithm for Vibration Suppression of Horizontal Flexible Structure," *2022 IEEE 8th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)*, pp. 237-241, 2022, https://doi.org/10.1109/ICSIMA55652.2022.9928906.

[49] I. S. Okoro and C. O. Enwerem, "Robust control of a DC motor," *Heliyon*, vol. 6, no. 12, p. E05777, 2020, https://doi.org/10.1016/j.heliyon.2020.e05777.

[50] A. Maarif and N. R. Setiawan, "Control of DC Motor Using Integral State Feedback and Comparison with PID: Simulation and Arduino Implementation," *Journal of Robotics and Control*, vol. 2, no. 5, pp. 456-461, 2021, https://doi.org/10.18196/jrc.25122.

[51] A. Ma'arif and A. Çakan, "Simulation and Arduino Hardware Implementation of DC Motor Control Using Sliding Mode Controller," *Journal of Robotics and Control*, vol. 2, no. 6, pp. 582-587, 2021, https://doi.org/10.18196/jrc.26140.

[52] K. G. Abdulhussein, N. M. Yasin, I. J. Hasan, and K. G. Abdulhussein, "Comparison between butterfly optimization algorithm and particle swarm optimization for tuning cascade PID control system of PMDC motor," *International Journal of Power Electronics and Drive Systems*, vol. 12, no. 2, pp. 736-744, 2021, http://doi.org/10.11591/ijpeds.v12.i2.pp736-744.

[53] N. Razmjooy, Z. Vahedi, V. V. Estrela, R. Padilha, and A. C. B. Monteiro, "Speed Control of a DC Motor Using PID Controller Based on Improved Whale Optimization Algorithm," *Metaheuristics and Optimization in Computer and Electrical Engineering*, vol. 696, pp. 153–167, 2021, https://doi.org/10.1007/978-3-030-56689-0_8.

[54] S. Nadweh, O. Khaddam, G. Hayek, B. Atieh, and H. Haes Alhelou, "Optimization of P& PI controller parameters for variable speed drive systems using a flower pollination algorithm," *Heliyon*, vol. 6, no. 8, p. E04648, 2017, https://doi.org/10.1016/j.heliyon.2020.e04648.