الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research
جامعة غرداية
University of Ghardaïa
كلية العلوم والتكنولوجيا
Faculty of Science and Technology
قسم الآلية والكهروميكانيك
Department of Automatics and Electromechanics

*Registration No*

…./……./……

**Final Thesis Submitted in partial fulfilment of the requirements for the degree of**

**MASTER'S DEGREE**

**Field: Science and Technology**

**Specialty: Renewable Energies in Electrotechnology**

## Theme

# Study and Simulation of DC Motor control

**Presented by**

**BAHMANI Brahim**

**MERZOUG Brahim**

**Publicly defended on: Juin 14th, 2026 In front of the jury:**

| Names | Grade | Quality | University |
|---|---|---|---|
| BAHRI Ahmed | Professor | President | Univ. of Ghardaïa |
| BOUCHAKOUR Abdelhaq | Directeur de recherche | Examiner | URAER |
| LABSIR Abdelkader | MCB | Examiner | Univ. of Ghardaïa |
| ZEGAOUI Abdallah | Professor | Supervisor | Univ. of Ghardaïa |
| BOUKHARI Hamed | MCA | Co-Supervisor | Univ. of Ghardaïa |

**Academic year: 2025/2026**

## Thanks

***We begin by thanking God for His guidance and support throughout all stages of this work.***

***We would like to express my deep gratitude to everyone who contributed to the completion of this research.***

***First and foremost, we extend our sincere thanks to our supervisor, Professor ZEGAOUI Abdallah, for his invaluable guidance, advice, and unwavering support. His expertise and patience were essential to the successful completion of this work.***

***We would also like to express our gratitude to Assistant Professor BOUKHARI Hamed for sharing his knowledge and skills with us throughout this project. His dedication and passion for teaching us were a constant source of inspiration.***

***We are deeply grateful to our fellow students for their friendship and solidarity, and for the constructive discussions that enriched our academic journey.***

***We would also like to express our gratitude to our families, especially the fathers (may God have mercy on those who have died and protect those who still alive), the mothers (may God grant them a long life), the siblings, the wife, and children, for their unwavering support, love, and encouragement throughout this academic journey.***

***Finally, we would like to express our sincere gratitude to everyone who contributed to this research, directly or indirectly, their support, advice, and encouragement have been invaluable. With deepest appreciat***

**ملخص**

لمحركات التيار المستمر تطبيقات واسعة في الآلات الصناعية والروبوتات وأنظمة الطاقة. ومع ذلك، قد يعمل محرك التيار المستمر بدون وحدة تحكم بسرعة غير مستقرة مما يؤدي إلى فشل تشغيل النظام. في الواقع، تستخدم وحدات التحكم المشتقة التناسبية (PID) عادة للتحكم في سرعة محركات التيار المستمر بسبب بنية التحكم البسيطة ولكن الأداء الفعال للتحكم بها. يهدف هذا المشروع إلى التحكم في سرعة محرك التيار المستمر باستخدام بطاقة أردوينو أونو الإلكترونية بواسطة منظم PID مما يؤدي إلى محاكاة MATLAB والحالات العملية. يركز هذا المشروع على تطوير التحكم في PID بمحرك أردوينو DC، ودمج محرك أردوينو DC في MATLAB مع الاستمرار في التحقق من أداء التحكم في PID. ثبت أن سرعة محرك التيار المستمر تم التحكم بها بنجاح بواسطة وحدة تحكم PID مع تحسن يزيد عن 85٪ من متوسط الخطأ لكل من المهام المحاكاة والتجريبية. تظهر الدراسة فعالية PID في تنظيم سرعة المحرك وإمكاناتها في استراتيجيات التحكم المتقدمة لتطبيقات المحركات DC المختلفة، عبر المجالات التعليمية والصناعية.

الكلمات المفتاحية: محركArduino DC ، وحدة تحكم PID

**Abstract**

DC motors have wide applications in industrial machinery, robotics and power systems. However, a DC motor without a controller may run in unstable speed and leading to the failure of the system operation. In fact, proportional-integral-derivative (PID) controllers are commonly used to control the speed of DC motors due to simple control structure but effective control performance. This project aims to control the speed, using Arduino Uno electronic card, of DC motor by PID regulator leading to MATLAB Simulations and practical cases. This project focuses on the development of Arduino DC motor PID control, and the integration of the Arduino DC motor to MATLAB continuing with the validation of the PID control performances. It was proved that the speed of the DC motor was successfully controlled by the PID controller with more than 85% improvement of the mean error for both simulation and experimental tasks. The study demonstrates the effectiveness of PID in regulating motor speed and its potential for advanced control strategies of various DC motor applications, across educational and industrial areas.

Keywords: Arduino DC Motor, PID Controller.

Voici la traduction en français du résumé :

**Résumé**

Les moteurs à courant continu (CC) ont de nombreuses applications dans les machines industrielles, la robotique et les systèmes de puissance. Cependant, un moteur CC sans contrôleur peut fonctionner à une vitesse instable, entraînant l'échec du fonctionnement du système. En effet, les contrôleurs proportionnel-intégral-dérivé (PID) sont couramment utilisés pour contrôler la vitesse des moteurs CC en raison de leur structure de contrôle simple mais de leurs performances de contrôle efficaces. Ce projet vise à contrôler la vitesse, à l'aide de la carte électronique Arduino Uno, d'un moteur CC par un régulateur PID, en menant à des simulations MATLAB et à des cas pratiques. Ce projet se concentre sur le développement du contrôle PID du moteur CC avec Arduino, ainsi que sur l'intégration du moteur CC Arduino à MATLAB, en poursuivant avec la validation des performances du contrôle PID. Il a été prouvé que la vitesse du moteur CC a été contrôlée avec succès par le contrôleur PID, avec une amélioration de plus de 85 % de l'erreur moyenne, tant pour les tâches de simulation que pour les tâches expérimentales. L'étude démontre l'efficacité du PID dans la régulation de la vitesse du moteur et son potentiel pour des stratégies de contrôle avancées dans diverses applications de moteurs CC, dans les domaines éducatif et industriel.

**Mots-clés** : Moteur CC Arduino, Contrôleur PID

# List of Contents

# List of Figures

# List of Tables

# Abbreviations list

**DC** Direct Current

**EMF** Back Electromotive Force

$\vec{\mathbf{B}}$ Magnetic field

**N** Number of turns of the wire

**P** Generic point in space

**S** Surface area

**μ** Magnetic permeability of dielectric

**ϕ** Magnetic flux

$\mathbf{P_a}$ Continuous electrical power absorbed

$\mathbf{P_u}$ Mechanical power supplied

$\mathbf{P_{em}}$ Electromagnetic power in the air gap

$\mathbf{p_{js}}$ Joule losses in stator

$\mathbf{p_{jr}}$ Joule losses in rotor

$\mathbf{p_{fr}}$ Iron losses

$\mathbf{p_m}$ Mechanical losses

$\mathbf{T_u}$ Useful torque

**T** Torque

**Ω** Angular velocity

**η** Efficiency

**F** Force magnitude

$\mathbf{V_a}$ Armature voltage

$\mathbf{I_a}$ Armature current

$\mathbf{R_a}$ Armature resistance (Ω)

$\mathbf{L_a}$ Armature inductance (H)

$\mathbf{V_f}$ Field voltage

$\mathbf{I_f}$ Field current

$\mathbf{R_f}$ Field resistance

$\mathbf{L_f}$ Field inductance

**E** Back electromotive force (EMF)

$\mathbf{K_e}$ Back-EMF constant

$\mathbf{K_t}$ Torque constant

**J** Moment of inertia

**B** Viscous friction coefficient

$\mathbf{T_L}$ Load torque

$\boldsymbol{\tau_a}$ Armature time constant

$\boldsymbol{\tau_m}$ Mechanical time constant

**α** Duty cycle of chopper

$\mathbf{K_p}$ Proportional gain

$\mathbf{K_i}$ Integral gain

$\mathbf{K_d}$ Derivative gain

$\mathbf{T_i}$ Integral time

$\mathbf{T_d}$ Derivative time

**IGBT** Insulated Gate Bipolar Transistor

**P** Proportional

**PI** Proportional Integral

**PID** Proportional Integral Derivative

$\mathbf{T_u}$ Delay time

$\mathbf{T_a}$ Time Constant

**K** Static Gain

$\mathbf{K_{cr}}$ Critical gain (Ziegler–Nichols's method)

$\mathbf{P_{cr}}$ Critical period in Ziegler–Nichols tuning method

**PWM** Pulse Width Modulation

**FF_GAIN** Feedforward gain constant

**Settling Time** Time to stabilize within tolerance

**SS Error** Steady-state error

**Arduino UNO** Microcontroller board

**IDE** Integrated Development Environment

**LCD 16×2** Liquid Crystal Display

**L298N** Dual H-Bridge motor driver

**GND** Ground

**VCC, VDD, VSS** Supply voltages (logic and motor power)

**RS, EN, D4–D10** LCD control/data pins

**INT0** External interrupt pin

**RPM** Rounds per minute

**ADC** Analog-to-Digital Converter

# General Introduction

The first practical DC motor was invented in 1832 by the British scientist William Sturgeon. Since then, DC motors have become an integral part of many devices and machines. Today, DC motors range in size from large models used in industrial equipment to small devices that can fit in the palm of your hand. They are inexpensive and ideal for use in robots, drones, and Internet of Things (IoT) projects. A DC motor is a type of electric motor that converts electrical energy into mechanical motion. It is widely used in a variety of applications, from simple households to complex industries. A DC motor operates on the basic principle of the interaction between a magnetic field and an electric current [1].

DC motors offer numerous advantages, including excellent speed and torque control, reversibility, rapid response to load changes, and a wide speed range. They are widely used in applications requiring smooth starting and precise speed control, such as robots, electric vehicles, GPS systems, and machine tools. AC motors, on the other hand, have limitations, such as the wear and tear on lamps and commutators. This research focuses on the design and simulation of a speed control for DC motors using an Arduino Uno board and Matlab software. [2]

This thesis consists of three chapters, organized as follows:

- Chapter 1 provides a general introduction to DC motors, covering their construction, operating principles, and different types of their excitations, along with the advantages and disadvantages of each DC motor type.
- Chapter 2 is reserved to the various mathematical modeling that explains the behavior of these motors and presents a PID digital control system for DC motor speed control using the open-source Arduino platform.
- The third and last chapter covers both simulation results under MATLAB/Simulink software and realization using Arduino IDE 2.3.8. To achieve this, we will first define the method for extracting PID controller and DC motor parameters. Then, we will select the various materials used in this application, program the Arduino UNO board using IDE 2.3.8, create a wiring diagram, and present the results.

Finally, we conclude our study with a general summary of the approach used in this Master thesis and outline the prospects for further research in this field.

# Chapter I

# DC motors Theory

# I.1. Introduction

DC motors are machines that convert electrical energy into mechanical energy, causing rotary movement. There were several reasons for the prolonged popularity of DC motors. One was that DC power systems are common even in cars, trucks and airplanes. When a vehicle has a DC power system, it makes sense to use DC motors. DC motors were also applied when wide variations in speed were required [3].

DC motors offer several advantages, including simple and precise speed control over a wide range and high starting torque, which makes them suitable for applications requiring variable speed and strong initial motion. However, they also present some disadvantages, such as the need for regular maintenance due to the presence of brushes and commutators, which can reduce reliability and increase operational costs [4].

In contrast, AC motors are more robust, require less maintenance, and are generally more cost-effective in the long term. Nevertheless, they often require more complex control systems for variable speed operation. Therefore, the choice between DC and AC motors depends on application requirements, including cost, maintenance, control complexity, and performance [5].

The construction of DC motors is identical to that of DC generators, so that a direct current machine can be used either as a motor or as a generator [6]. In this chapter we will see the construction and behaviour of this machine and the different characteristics.

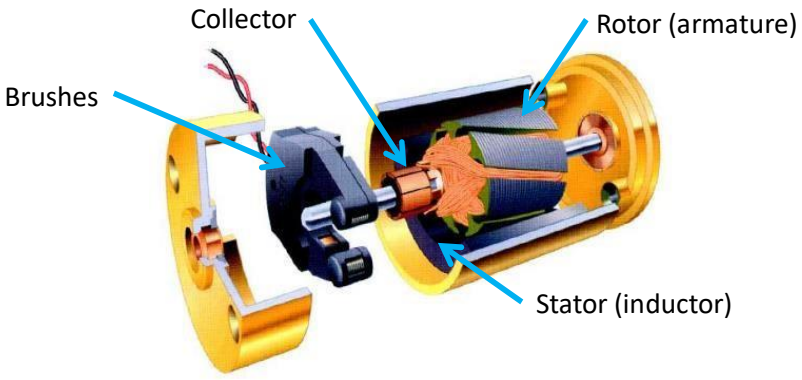



Figure (I – 1): DC motor

# I.2. Simplified description of a DC motor

A direct current machine essentially consists of:

- **The armature (or rotor):**

This is the moving or rotating part of the machine. It consists of a laminated cylinder made up of a stack of sheets. On the periphery of this cylinder are teeth between which there are notches inside which are conductors that constitute the winding of the armature, and which are grouped by coils whose ends close on a collector. The armature is therefore an electrical circuit obtained by associating in series conductors housed in notches in the rotor [7].

Figure (I – 2): the armature of DC motor (rotor)

- **The inductor (or stator):**

This is the fixed part of the machine. It can be made up of permanent magnets or coils generally placed on the stator. The coils are placed around polar nuclei. These coils, through which the excitation current flows, generate the magnetomotive force, force necessary to produce the flow. In bipolar (two-pole) machines, two exciter coils are carried by two polar pieces mounted inside a breech. The breech is usually made of cast steel, while the pole pieces are formed of mild steel sheets.

The number of poles carried by the inductor of a direct current machine depends mainly on the size of the machine. The more powerful a machine is and the lower its speed, the greater the number of poles.

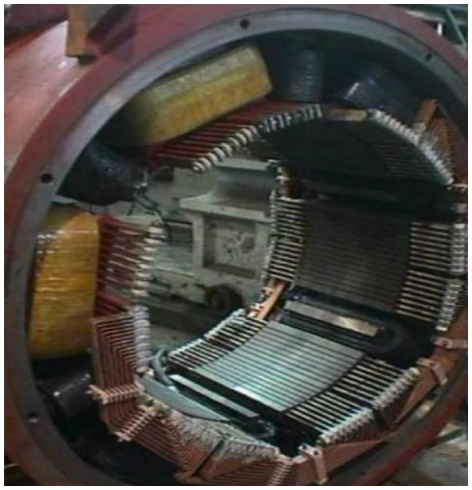


Figure (I – 3): inductor of DC motor (stator)

- **Collector and brushes:**

The collector is a set of copper blades, isolated from each other by mica sheets, and arranged according to a cylinder at the end of the rotor. The two wires coming out of each coil of the armature are successively and symmetrically welded to the collector's blades. In a bipolar

machine, two fixed and diametrically opposed brushes press on the collector. Thus, they ensure electrical contact between the armature and the external circuit.

Multipolar machines have as many brushes as poles. The brushes carried by the stator rub on the blades of the collector. These sliding contacts between blades and brushes make it possible to establish an electrical link between the rotating armature and the outside of the machine. The pressure of the brushes on the collector can be adjusted by adjustable springs. For very high intensities, several brushes connected in parallel are used [8].

Figure (I – 4): collector of DC motor

Figure (I – 5): brushes of DC motor.

## I.3. Working principle

A DC motor is rotated by an induced magnetic force called the LAPLACE force. This force applies to a conductor that is flowing through a current and placed in a magnetic field. The direction of this force is given by the rule of the three fingers of the right hand:

- The index finger is placed in the direction of the magnetic field.
- The thumb gives the direction of travel of the field lines.
- The middle finger shows the sense of the induced force of LAPLACE.

The north and south poles of the permanent magnets create a flux (magnetic field B) in the motor. The coil is fed and immersed in this flow. It is subjected to a couple of forces F (Laplace force). The engine starts to rotate. It is said that there is the creation of motor torque. Given the

arrangement of the brushes and the collector, the direction of the current I in the coil changes with each U-turn, which allows the same direction of rotation to be maintained (otherwise, the coil would remain in the equilibrium position) [9].

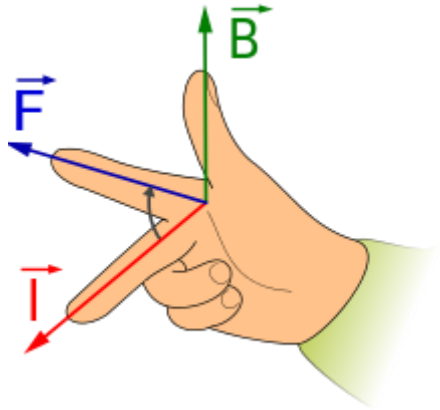



Figure (I – 6): the rule of the three fingers of the right hand.

## I.3.1. Magnetic field and conductors

A magnetic field is generated by a flowing current: it is experienced about a moving charge. The effect of a magnetic field may be experimented, for instance, by positioning a magnet close to a wire where a current is flowing; what can be observed is that the magnet experiences a force, which is due to the magnetic field generated by the current flowing in the wire.

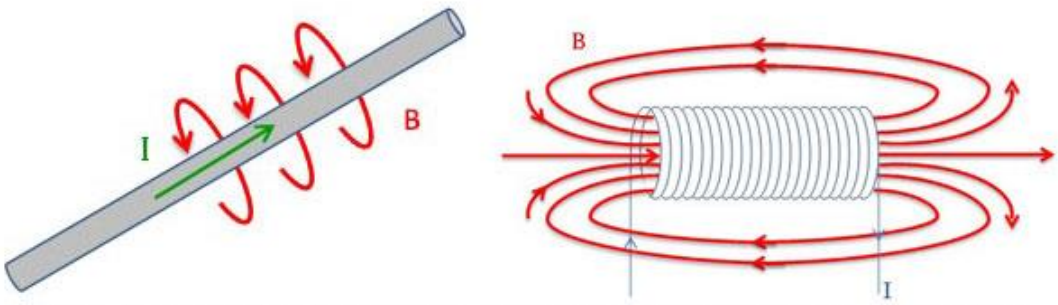



Figure (I – 7): the magnetic field about a solenoid where a current I flows.

The above experiment aims to verify that a flowing current indeed generates a vector field $\vec{B}$. This field may be thought of as the sum of infinite contributions $d\vec{B}$ due to all the infinitesimal segments of wire $d\vec{l}$, where the current i flows. Each segment induces a magnetic field at any point in the surrounding space. In particular, the Biot and Savart law states that, if r is the vector connecting the wire segment $d\vec{l}$ to a generic point p in space, the contribution $d\vec{B}$ of the magnetic field in the point p due to the segment $d\vec{l}$ is given by:

$$d\vec{B} = i\frac{d\vec{l}\times\vec{r}}{|r|^3}. \qquad (I-1)$$

Equation (I – 1) provides a tool for computing the magnetic field associated with any conductor where a current is flowing. By integrating it for the case of a solenoid, it turns out that, if the

corner effects are neglected (namely the solenoid is long enough), the field inside the windings is constant and parallel to the axis of the solenoid, and its magnitude is given by:

$$|\vec{B}| = \frac{\mu}{l} N\, i. \qquad (I-2)$$

where l is the length of the solenoid, μ is the magnetic permeability of the dielectric inside the solenoid and N is the number of turns of the wire.

Consider now a surface S located in a magnetic field B; the magnetic flux, or flux ϕ flowing through the surface is defined as the integral along the surface of the normal component of the magnetic field. If a uniform magnetic field B approaches a flat surface with an angle of incidence β and the area of the surface is A, then:

$$\phi = \overrightarrow{|B|}\, A \cos\beta. \qquad (I-3)$$

The flux may be computed substituting equation (I – 2) in equation (I – 3):

$$\phi = K_0\, N\, i. \qquad (I-4)$$

where: $K_0 = \frac{\mu}{l} A.$

## I.3.2. The magneto-motive force in a rotating coil

The Lorentz force equation is a good starting point to understand the mechanical aspects of this electromagnetic phenomenon. The Lorentz force is the force experienced by a moving charge in an electromagnetic field. If $\vec{E}$ denotes the electric field, $\vec{B}$ denotes the magnetic field, q is a charge moving with a speed v in the space, then the charge experiences a force given by:

$$\vec{F} = q(\vec{E} + \vec{v} \times \vec{B}). \qquad (I-5)$$

Now, consider a wire where a uniform current i is flowing. Assuming the electric field to be zero (this is the case of a DC motor), with reference to the quantity of charge dq found in an infinitesimal section d of the wire, the force in equation (I – 5) may be computed as a function of i:

$$d\vec{F} = dq\, \vec{v} \times \vec{B} = dq\, \frac{d\vec{l}}{dt} \times \vec{B} = \frac{dq}{dt}\, d\vec{l} \times \vec{B} = i\, d\vec{l} \times \vec{B}.$$

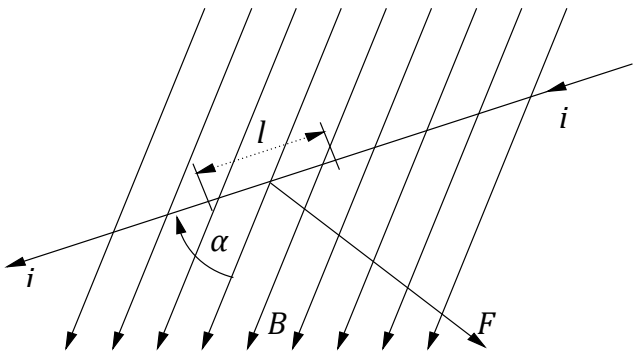



Figure (I – 8): force experienced by a conductor located in a uniform magnetic field.

It turns out that, if α denotes the angle of incidence between the magnetic field and a straight wire of length l, then the magnitude of the force F is given by:

$$|\vec{F}| = i\,\vec{l}\,|\vec{B}| \sin\alpha. \qquad (I-6)$$

Consider a rigid rectangular coil constituted by a single wire where a current i flows, suitably located in a uniform exogenous magnetic field. If l is the length of the wire perpendicular to the magnetic field, then two forces are applied to the coil. Since the angle α is $\pm\frac{\pi}{2}$ (depending on which side of the coil is considered), then it turns out that the magnitudes of the two forces are the same:

$$|\vec{F}| = |\vec{B}|\,\vec{l}\,i. \qquad (I-7)$$

### I.3.3. Electromagnetic torque

Since the coil is square, the current i flows in opposite directions on the two sides of the coil; thus, the two forces F generate a torque T exerted at the center of the coil that is dependent on the angular position θ of the coil with respect to the magnetic field. If the length of an edge of the square is d, then:

$$T = 2\,|\vec{F}|\,\frac{d}{2}\sin\theta = |\vec{F}|\,d\sin\theta. \qquad (I-8)$$

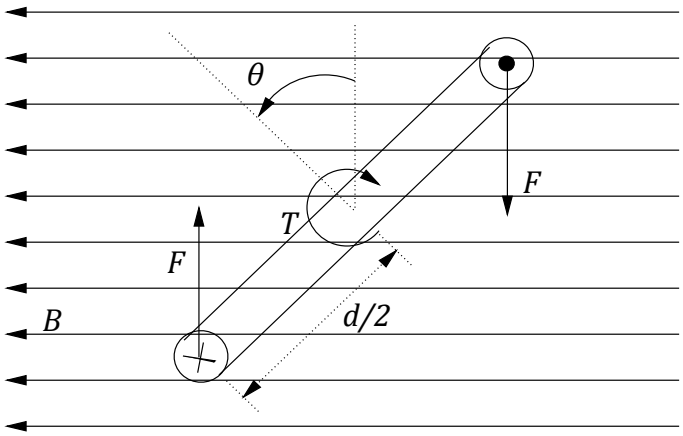



Figure (I – 9): torque experienced by a coil in a uniform magnetic field.

whence, considering equation (I – 7), equation (I – 8) yields:

$$T = i\,\vec{l}\,|\vec{B}| \sin\theta. \qquad (I-9)$$

Consider now equation (I – 9) and notice that the torque highly depends on the rotor coil angular position θ. Imagine that the coil is in the rotor of a motor; then the resulting torque is highly dependent on the motor position.

Since the goal is to have the motor to exert a constant torque for any position θ, the solution adopted is to insert on the rotor shaft a commutator constituted by two segments connected to the rotor windings and brushes that slide between the segments as the rotor turns. In such a configuration, the sign of the current flowing in the coil changes at each half revolution of the

motor; thus, if the segments are properly positioned with respect to the coil position, the torque profile will be suitably inverted during half revolution.

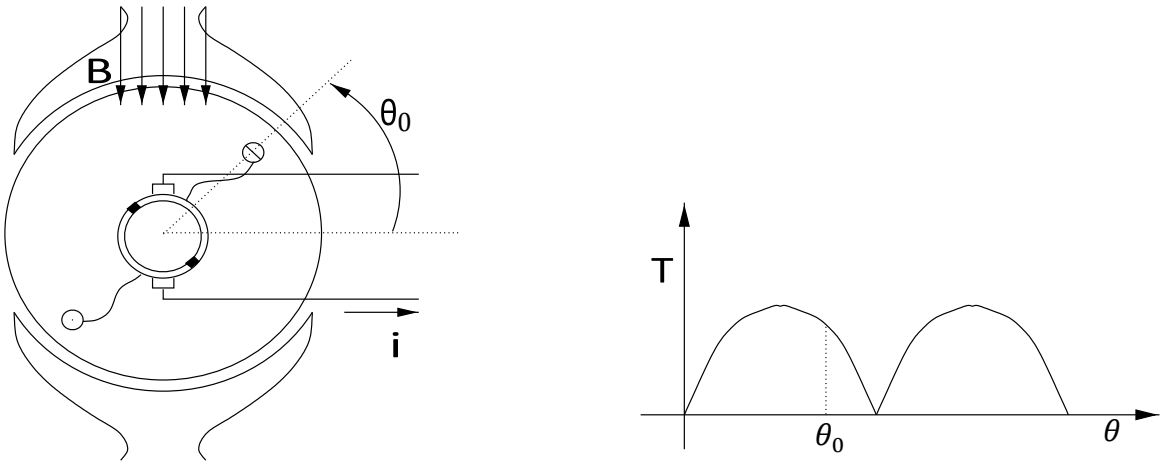



Figure (I – 10): torque exerted by the motor, when a two segments commutator is used.

now, ones this solution is adopted, although the torque has always the same sign, still it is highly dependent on the rotor position; the obvious solution to this problem is to increase the number of coils in the rotor and the segments of the commutator, connecting each pair of the opposite segments to a coil in such a way that when the brushes activate that coil, the rotor angle is $\theta = \pm\frac{\pi}{2}$ (i.e., the maximum torque position). So that the resulting torque may be approximated as:

$$T = i\,\vec{l}\,d\,|\vec{B}|. \qquad (I-10)$$

By equation (I – 3), the flux $\phi$ flowing through the rotor is proportional to $|\vec{B}|$ (the flux is assumed to be flowing straight inside the motor), it turns out:

$$T = K_\phi\,\phi\,i. \qquad (I-11)$$

where: $K_\phi = \frac{l\,d}{A}$.

## I.3.4. The back electro-motive force (The back EMF) effect

The rotor conductors carry current supplied from an external source and are in an exogenous magnetic field; whence, forces are experienced by the coil, and a torque is exerted on the rotor-shaft. However, due to the rotation of the coil, the conductors themselves cut the magnetic flux, thus generating an electro-motive force (i.e., an induced voltage in the rotor windings) whose magnitude may be computed by means of the Faraday's law of induction. This law states that if there is a variation of the flux $\phi$ flowing in the internal surface of a closed wire, then an electro-motive force e is induced in the wire, according to:

$$E = -\frac{d\phi}{dt}. \qquad (I-12)$$

The flux flowing in the coil is given by equation (I – 3) substituting β with the angle θ(t) of the motor and A with the area of the internal surface of the coil. When computing the derivative in equation (I – 12), the back EMF may be computed as:

$$E = -\frac{d}{dt}(\overrightarrow{|B|}\, A \cos\theta(t)) = \overrightarrow{|B|}\, A\, \Omega(t) \sin\theta(t).$$

Where: $\Omega = \frac{d\theta(t)}{dt}$ denotes the angular speed of the motor.

Noticing that, due to the presence of the commutator, the coil always operates in a neighbourhood the position $\theta = \frac{\pi}{2}$, then $\sin\theta(t) \approx 1$ and the back EMF may be written as $E = \overrightarrow{|B|}\, A\, \Omega(t)$.

Finally, similarly to the case studied in equations $(I - 10)$ and $(I - 11)$, the back EMF may be expressed as a function of the flux Φ, and it can be verified easily that:

$$E = K_\phi\, \phi\, \Omega. \qquad (I - 13)$$

Where $K_\phi$ is the same constant as the one in equation (I – 11) [10].

## I.4. Different types of DC motor excitations

There are several methods to power the motor inductor, each of which leads to different operating characteristics.

- Separate excitation machine.
- Parallel excitation machine (shunt).
- Series Excitation Machine.
- Compound excitation machine (there are two excitations at the same time "series and shunt") [11].

### I.4.1. Separately excited motor

In a separate or independently excited motor, the excitation circuit is separated from the armature circuit. If the inductor is a permanent magnet, the flux Φ is constant. If the inductor is an electromagnet powered by an adjustable DC voltage source, the flux ϕ depends only on the current in the inductor called the excitation current [12].

$$E = K_\phi\, \phi\, \Omega = V - R\, I.$$

$$\Omega = \frac{V - R\, I}{K_\phi\, \phi}. \qquad (I - 14)$$

$$T = \frac{e\, I}{\Omega} = K_\phi\, \phi\, I = K'\, I.$$

Where $K' = K_\phi\, \phi$.

We deduce that the rotational speed is imposed by the supply voltage and by the flux. An increase in this first parameter leads to a proportional increase in velocity, while a decrease in

excitation is necessary to obtain the same effect. It will therefore be the torque required by the mechanical load that will impose the current called by the motor to the voltage source.

This motor is characterized by a tension-adjustable speed that is independent of the load. In combination with a static converter (chopper) providing adjustable voltage, the speed can vary over a wide area. It provides a lot of torque at low speeds (machine tools, lifting). In low power, it is often used in servo control with speed regulation [13].

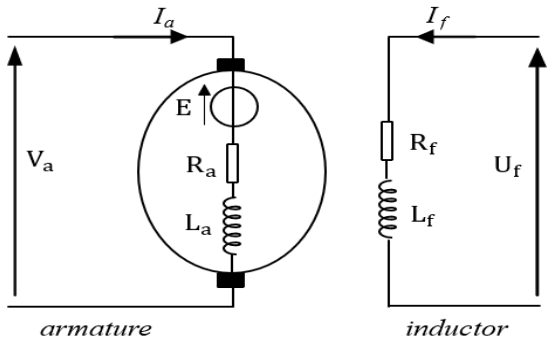



Figure (I – 11): separately excited DC motor.

These motors are used wherever precise and stable speed control is required, regardless of load variations. Typical applications include:

- Machine tools (lathes, milling machines) require constant cutting speed.
- Servo-control systems and position control drives.
- Paper and textile manufacturing machines require constant web tension.
- Conveyors and hoisting equipment in industrial plants [14].

## I.4.2. Series excitation motor

A series-excited motor has its inductor connected in series with the armature; the flow of excitation is therefore implicitly enslaved to the torque provided. In this case, the inductive winding has few turns, but it is made of large diameter wire. This design gives it very good robustness against vibrations and has earned it unparalleled success in rail traction.

$$E = V - (R + r)I = K_{\phi}\,\Omega\,I.$$

$$T = \frac{e\,I}{\Omega} = K'\,I^2. \qquad (I - 15)$$

The above equations show that series-excited motors can develop a very high torque, especially at low speeds, which is proportional to the square of the current. This is why they used to make locomotive traction motors until the 1980s. However, due to its characteristics, this type of machine presents a risk of overspeed and running away when empty.

Today, the main applications are:

– Automobile starters.
– Universal motors (drills, hand tools, etc.).

It is also noted that the torque provided by a series-excited motor will always be of the same sign $I^2$ [13].

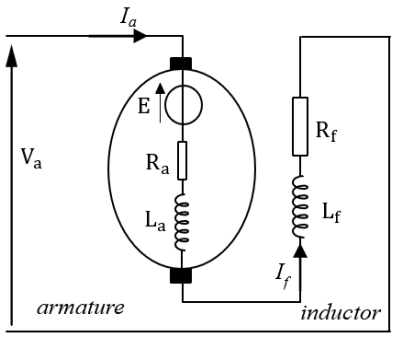



Figure (I – 12): series excitation motor.

### I.4.2.1. Universal motor

Universal motor is an application of the series excitation motor. We have just seen that for a motor with series excitation the torque is proportional to the square of the intensity of the current flowing through it. This means that the torque does not depend on the direction of the current:

The universal motor can therefore also operate on alternating current.

Indeed, with each alternation of the single-phase voltage, the electric current and the magnetic flux change direction but the electromagnetic force is unchanged (rule of the 3 fingers of the right hand). However, to avoid excessive heating due to alternating currents in the iron, it is imperative that the magnetic circuit is laminated (sheet metal stacking). These motors are widely used in household appliances and tools (electric drill) [12]. Due to their ability to deliver high starting torque, series motors are primarily used in:

- Electric railway traction and metro systems (historically until the 1980s).
- Automobile starters require very high torque at low speed.
- Universal motors in portable power tools (drills, mixers, vacuum cleaners) [14].

## I.4.3. Excitation shunt

A single supply voltage can be used for the armature and inductor, simply place the inductor winding parallel with the armature and feed them through a voltage source [7]. The shunt motor has the same characteristics as the separately excited motor, we conclude that:

– For a shunt motor and a separately excited motor, the voltage applied to the excitation winding and subsequently the excitation current is independent of the load.
– The speed of a shunt or separately excited motor decreases slightly as torque increases [11].

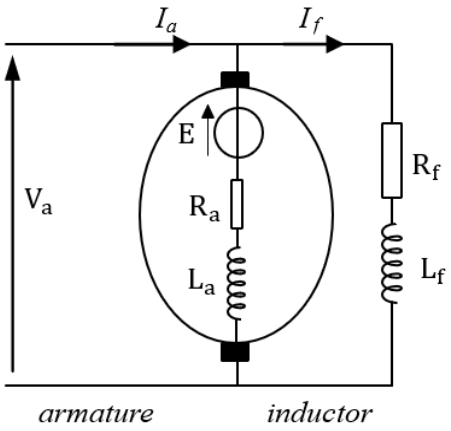



Figure (I – 13): shunt excitation.

### I.4.4. Compound motor

Compound motors have a field connected in series with the armature and a separately excited shunt field, the series field provides better starting torque and the shunt field provides better speed regulation, however the series field can cause control problems in variable speed drive applications and is generally not used in four quadrant drives [15].

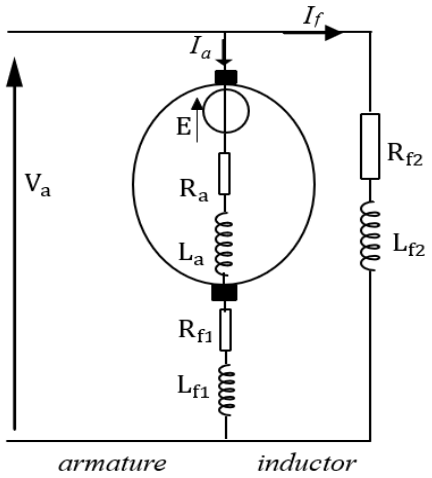



Figure (I – 14): compound motor.

## I.5. Power Balance

DC motors consume part of the energy absorbed for their operation. The mechanical energy supplied will always be smaller than the electrical energy absorbed:

- The electrical power absorbed by the armature and by the inductor is measured directly with a power meter or calculated according to the relationship giving continuous power:

$$P_a = V_a I_a + V_e I_e. \qquad (I - 16)$$

- The electromagnetic power due to the electromagnetic torque present in the air gap of the motor is also the electrical useful power:

$$P_{em} = E I = T_{em}\Omega. \qquad (I - 17)$$

Where Ω the rotational angular velocity in rad/s.

- The energy or power lost in the motors is dissipated in the stator and in the rotor.
  - By Joule effect: copper loss
    In the stator, circuit through which the excitation current due to the Joule effect is denoted $P_{js}$ where:

$$P_{js} = R_e I_e^2. \quad (I-18)$$

    In the rotor, circuit through which the supply current due to the Joule effect is denoted $P_{jr}$ where:

$$P_{jr} = R_a I_a^2. \quad (I-19)$$

  - Iron losses in the rotor magnetic circuit due to eddy currents noted $P_{fr}$.
  - Mechanical losses in rotating elements due to friction noted $P_M$.
- The mechanical power supplied is determined according to the relationship:

$$P_u = P_a - p_{js} - p_{fr} - p_{jr} - p_M = T_u \Omega. \quad (I-20)$$

Where $T_u$ useful torque provided in Nm (newton meter).

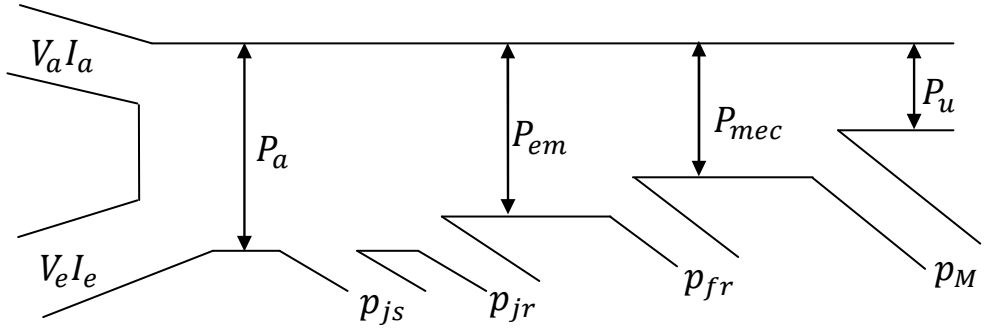



Figure (I – 15): power balance

## I.5.1. Efficiency

Efficiency is the ratio between the energy supplied and the energy absorbed. Efficiency is noted η; it is a ratio of energy or power without unity. It can be determined either by calculating or directly measuring $P_a$ and $P_u$, or by measuring or determining losses [12]:

– **Direct Measurement:**

This method consists of measuring Pa and Pu:

$$\eta = \frac{P_u}{P_a} = \frac{T_u \Omega}{P_a} \quad (I-21)$$

– **Separate loss method:**

This method consists of evaluating the different losses [11]:

$$\eta = \frac{P_u}{P_a} = \frac{P_a - \sum pertes}{P_a} \quad (I-22)$$

## I.6. Conclusion

In this chapter we have had some general knowledge about DC motors, and we have defined electromechanical conversion, and we have seen the construction of the DC motor, the different excitation modes and power balance.

# Chapter II

# Modelling and control of DC motor.

## II.1. Introduction

This chapter deals with speed control of separately excited DC motor control in a high-performance manner. There are four different parts of control of DC motors used in this project. Using chopper as a converter the speed of DC motor is controllable. The chopper firing circuit gets signal from controller and then by supplying variable voltage to the armature of the motor the desired speed chopper is achieved. The controller used is Proportional-Integral type. Using this controller the delay is removed and fast control is achieved. Separately exciting DC motor is designed. The speed controller is designed to get stable and high-speed control of DC motor [16].

## II.2. DC motor model

The DC motor equations can be written using the equivalent circuit diagram. The excitation of the armature is the terminal voltage $V_a$, also called the armature voltage and the armature current in the rotor winding is denoted by $I_a$. The electric block represents the armature circuit, which can be modeled as a simple serial-connected RL circuit, and the voltage E induced by the rotation of the rotor acts against the armature voltage so that it is connected in series with the RL circuit in the model, thus reducing the voltage on the RL-elements. The mechanical block of the model describes the rotation of the rotor and the mechanical characteristics of the rotational motion [17].

For illustration, let us consider separately excited dc motor. For ease of analysis, the following assumptions are made:

1. The axis of armature MMF is fixed in space, along the q-axis.
2. The demagnetizing effect of armature reaction is neglected.
3. Magnetic circuits are assumed to be linear (no hysteresis and saturation). As a result, all inductances (which came into play in dynamic analysis) are regarded as constant.

The two inductance parameters are defined below:

- $L_a$: armature self-inductance caused by armature flux; this is quite small and may be neglected without causing serious error in dynamic analysis.
- $L_f$: self-inductance of field winding; it is quite large for shunt field and must be accounted for Mutual inductance (between field and armature) = 0; because the two are in space quadrature.

Further for dynamic analysis it is convenient to use speed in rad/s rather than rpm [18].

### II.2.1. The electric equations

The rotor is assumed to be a single coil characterized by inductance La and resistance Ra, but it must be considered the back EMF of the motor in equation (I – 13) [10].

Applying Kirchoff's law to the armature circuit:

$$V_a(t) = E(t) + R_a i_a(t) + L_a \frac{d}{dt} i_a(t). \quad (II-1)$$

$$E = K_\phi \phi \Omega(t) = K' \Omega(t).$$

Where: $K' = K_\phi \, \phi$.

Similarly for the field circuit:

$$V_f(t) = R_f i_f(t) + L_f \frac{d}{dt} i_f(t). \quad (II-2)$$

Where:

- $V_f$ is the field voltage.
- $R_f$ is the resistance of the field winding.
- $i_f$ is the field current.

$\Phi$ is constant [18].

## II.2.2. The mechanical equations

The motor is driving a mechanical load which must be represented by model that relate the speed to torque. For simplicity, we assume that the load has a moment of inertia $J(kg.m^2/s^2)$, a viscous friction $B(N.m/rad/s)$ and a net torque $T_L$. Therefore, the motion equation can be obtained as below [10]:

$$T = J \frac{d\Omega}{dt} + B\Omega + T_L = K_\phi \, \phi \, I = K' \, I. \quad (II-3)$$

Where:

- T is the torque developed by the motor.
- J is the equivalent moment of inertia of motor shaft and load referred to the motor.
- B is the equivalent coefficient of motor and load referred to the motor shaft.
- $T_L$ The resistant torque.

Energy storage is associated with the magnetic fields produced by $i_f$ and $i_a$ and with the kinetic energy of the rotating parts. The above equations are a set of nonlinear (because of products $i_f(t)\Omega(t)$ and $i_f(t)i_a(t)$) state equations with state variables $i_f$, $i_a$ and $\Omega$. The solution must be obtained numerically.

## II.2.3. Transfer functions and block diagram

In the simple linear case of motor response to changes in armature voltage, it is assumed that the field voltage is constant and steady- state is existing on the field circuit, i.e. $I_f$ = constant. Equations $(II-1)$, $(II-2)$ and $(II-3)$ now become linear as given below:

$$V_a(t) = K' \, \Omega(t) + R_a i_a(t) + L_a \frac{d}{dt} i_a(t). \quad (II-4)$$

$$T(t) = K' \, i_a(t) = J \frac{d}{dt} \Omega(t) + B\Omega(t) + T_L(t). \quad (II-5)$$

Laplace transforming eqs (II – 4) and (II – 5):

$$V_a(s) = K' \Omega(s) + (R_a + sL_a)I_a(s).$$

$$T(t) = K' I_a(s) = (B + sJ)\Omega(s) + T_L(s).$$

These equations can be reorganized as:

$$I_a(s) = \frac{V_a(s) - K' \Omega(s)}{(R_a + sL_a)}.$$

$$I_a(s) = \frac{V_a(s) - K' \Omega(s)}{R_a(s\tau_a + 1)} \qquad (II - 6)$$

Where:

- $\tau_a = \frac{L_a}{R_a}$: is known as the time constant of motor armature circuit.

$$\Omega(s) = \frac{K' I_a(s) - T_L(s)}{(B + sJ)}.$$

$$\Omega(s) = \frac{K' I_a(s) - T_L(s)}{B(s\tau_m + 1)} \qquad (II - 7)$$

Where:

- $\tau_m = \frac{J}{B}$: is known as the mechanical time constant of the motor.

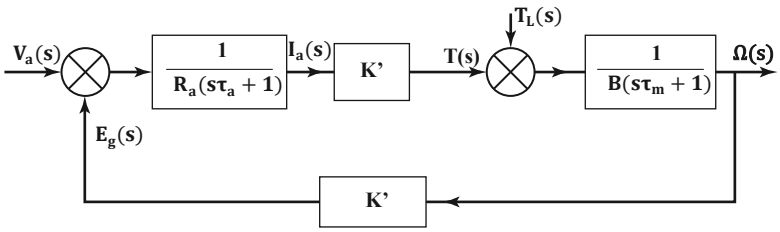



Figure (II – 1): DC motor block diagram

The response due to a step change in the reference voltage is obtained by setting $T_L$ to zero. From Figure (II – 1), we can obtain the speed response due to reference voltage as [18]:

$$\frac{\Omega(s)}{V_a(s)} = \frac{\frac{K'}{R_a B}}{(\tau_a \tau_m)s^2 + (\tau_a + \tau_m)s + 1 + \frac{K'^2}{R_a B}}. \qquad (II - 8)$$

## II.3. DC motor control

The relationship of the speed of a DC motor (shunt or separate excitation) is given by:

$$\Omega = \frac{V_a - R_a I_a}{K_\phi \phi}. \qquad (II - 9)$$

Exploring this relationship, it becomes clear that there are three possibilities for speed adjustment:

- Action on $R_a$ (dynamic adjustment).
- Action on $\phi$ (flow adjustment).
- Action on $V_a$ (voltage adjustment).

## II.3.1. Dynamic adjustment

Since the voltage and flux are fixed at their nominal value, the speed can be reduced by increasing the resistance of the armature by means of a rheostat ($R_h$) connected in series with the armature. We have the following parametric relations:

$$\begin{cases} T = K' I_a. \\ \Omega = \frac{V_a - (R_a + R_h) I_a}{K'}. \end{cases} \quad (II - 10)$$

- For $T = 0$ ; $I_a = 0 \rightarrow \Omega = \frac{V_a}{K'}$ : This speed does not depend on $R_h$, so the corresponding point is immutable.
- For $\Omega = 0$ ; $I_a = \frac{V_a}{R_a + R_h} \rightarrow T = K' \frac{V_a}{R_a + R_h}$ : When you increase $R_h$, the torque $T$ decreases, we obtain a bundle of concurrent lines.

This setting is bad both technically and economically. Indeed, from a technical point of view, the characteristics being concurrent, they become more "series", i.e. with a resistor inserted, the speed drop increases with the load. Technically, a good adjustment should result in the features being moved parallel to the original feature.

In addition, this setting is bad from an economic point of view because the energy consumption in the rheostat is even more important the higher the speed drop required. Thus, at half speed, as much energy is consumed in the rheostat as in the engine. In practice, this adjustment process is only used for starting or braking.

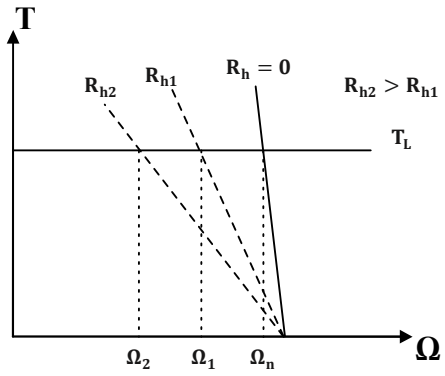



Figure (II – 2): dynamic adjustment: adjustment characteristics.

## II.3.2. Flow adjustment

It is important to note that, by virtue of the relationship of the torque ($T = K_\phi \phi I_a$), it is always in the interest of applying the maximum flow at start-up to allow the engine to bring to the nominal speed all the masses that are initially at rest. In addition, due to the saturation of the machine's iron, this flux value cannot be increased further. It follows therefore and in

accordance with equation (II – 9) giving the speed of the motor; that the adjustment is obtained by increasing the speed in relation to the nominal speed by reducing the value of the flux (excitation current). This is achieved by introducing a field rheostat into the excitation circuit. We have the following relationships:

$$\begin{cases} T = K_\phi \phi I_a. \\ \Omega = \frac{V_a - R_a I_a}{K_\phi \phi}. \end{cases} \quad (II - 11)$$

For $T = 0$ ; $I_a = 0 \rightarrow \Omega = \frac{V_a}{K_\phi \phi}$ : When the value of $\phi$ is reduced, the speed increases.

for $\Omega = 0$ ; $I_a = \frac{V_a}{R_a} \rightarrow T = K_\phi \phi \frac{V_a}{R_a}$ : When the value of $\phi$ is reduced, the torque decreases.

This setting is bad from a technical point of view; the characteristics being concurrent. It is good from an economic point of view because the power dissipated in the inductor is very low compared to the power absorbed; The engine efficiency will not be affected. However, the following remarks should be made:

With this process, the motor speed can only be increased compared to its nominal speed. If the load torque is constant $T = K_\phi \phi I_a = cst$, the current will increase when the flow is reduced and the motor may heat up.

Under these conditions, the engine must be sized accordingly. It should be noted that this disadvantage does not occur if the drive is at constant power because by virtue of the relationship $P = V_a I_a = cst$, as the voltage $V_a$ is constant, the current $I_a$ will remain constant.

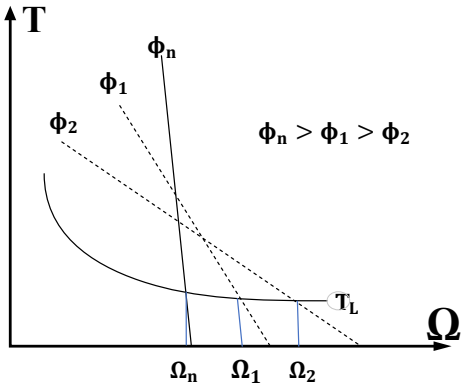



Figure (II – 3): flow adjustment feature.

## II.3.3. Voltage adjustment

Since the excitation flux is set at its nominal value, the adjustment is achieved by reducing the speed from the nominal speed by reducing the value of the supply voltage. By virtue of the relationship:

$$\begin{cases} T = K' I_a \\ \Omega = \frac{V_a - R_a I_a}{K'}. \end{cases} \quad (II - 12)$$

- For $T = 0$ ; $I_a = 0 \rightarrow \Omega = \frac{V_a}{K'} \rightarrow \Delta\Omega = \frac{\Delta V_a}{K'}$.

When the voltage $V_a$ is reduced, the speed decreases.

- For $\Omega = 0$ ; $I_a = \frac{V_a}{R_a} \rightarrow T = K' \frac{V_a}{R_a} \rightarrow \Delta T = K' \frac{\Delta V_a}{R_a}$.

When the voltage is reduced, the torque decreases. Let's calculate the slope: $\frac{\Delta T}{\Delta\Omega}$

$\frac{\Delta T}{\Delta\Omega} = \frac{K'^2}{R_a} = cst.$

Features move parallel to the original feature:

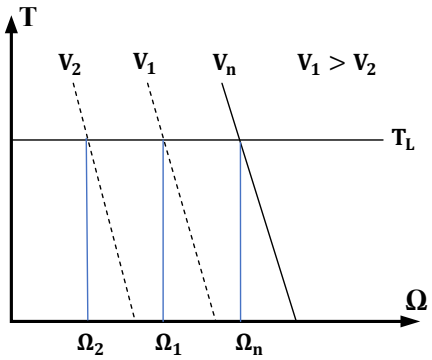



Figure (II – 4): adjustment feature by armature voltage.

## II.4. Realization of a variable armature voltage

To achieve a variable armature voltage, the WARD-LEONNARD assembly was used for a long time. With the advent of semiconductors, it was found interesting to use a power electronics converter (Figure (II –5)).

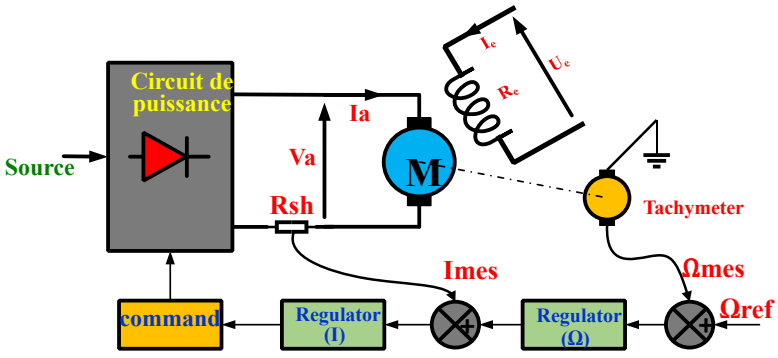



Figure (II – 5): closed-loop variable DC voltage.

The electronic converter supplies the armature of the motor, and its speed is captured by a tachometer dynamo. The measured value (Ωmes) is compared to the reference speed (Ωref). The control gap is the cruise control input that generates the signal at the control unit input.

To this velocity loop, a current loop is added; A shunt $R_{sh}$ gives a proportional voltage to applied $I_c$ to the input of the current regulator. The limiters placed at the output of the regulators are used to maintain the current and armature voltage between the maximum allowable values.

The converters used are classified as follows:

- AC-DC converters (Rectifiers): A distinction is made between non-reversible and reversible assemblies.
- Continuous-to-continuous converters (chopper): There are also two types of assemblies: non-reversible and reversible.

## II.5. DC motor-Chopper association

The main problem of modern electric control is economical and progressive speed control within wide limits and with high operational reliability. Manufacturing efficiency, production quality, productivity and cost are all always linked to manufacturing automation. The assurance of the rational operation of technological process depends very much on the speed of machine tools. The speed may vary during the variation of the parameters of the electric motors (resistors, inductors) or the parameters of the power source (voltage, frequency).

### II.5.1. Chopper drives

When the equipment is supplied with direct current, as is the case with a battery of accumulators or the direct current catenary in electric traction, the variable direct voltage applied to the armature $V_a$ is obtained by means of a chopper.

The value of $V_a$ can be influenced either by an analogue assembly or by means of a digital control. The chopper allows the adjustment of the power transfer between a DC voltage source and a current source.

### II.5.2. Series chopper (buck)

The series chopper is used when the motor is to work only in front-engine operation:

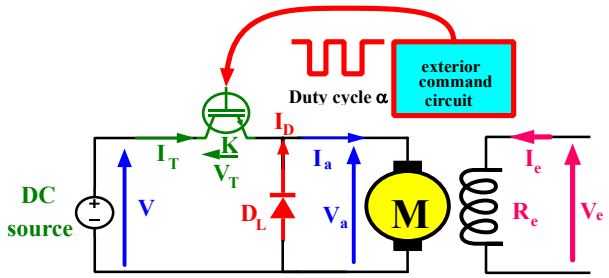



Figure (II – 6) : series chopper association with DC motor.

T is considered the operating period and $t_{on}$ the duration of the IGBT conduction intervals.

If K is conductor and D is blocked for $0 \leq t \leq t_{on}$:

$$V_a = V \rightarrow I_T = I_a \rightarrow V_T = 0 \rightarrow V_D = -V \rightarrow I_D = 0.$$

If K is blocked and D is conductor for $t_{on} \leq t \leq T$:

$V_a = 0 \rightarrow I_T = 0 \rightarrow V_T = V \rightarrow V_D = 0 \rightarrow I_D = I_a$.

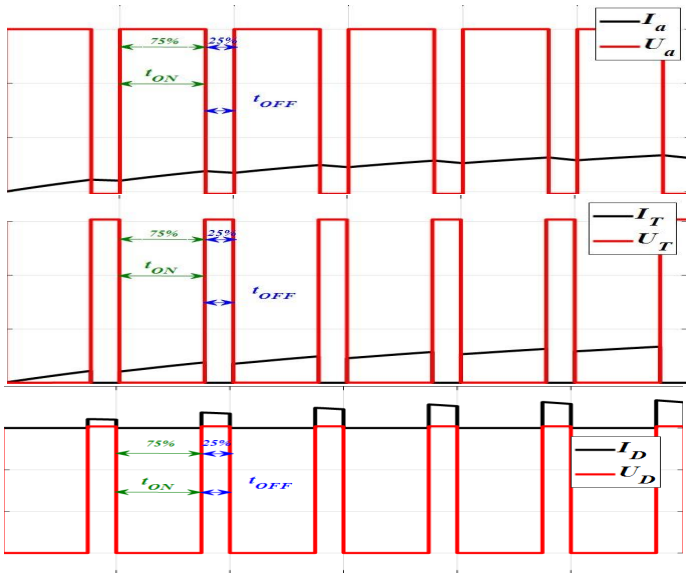



Figure (II – 7): motor currents and voltages, IGBT and diode.

According to the curves shown in Figure (II – 7), the average value of the voltage across the motor is:

$$V_a = \frac{1}{T}\int_0^{t_{on}} V\, dt = \frac{t_{on}}{T} V = \alpha V. \qquad (II - 13)$$

- where: $\alpha \in [0,1]$ α: is called a duty cycle.

The rotational velocity expression is related proportional to α [19]:

$$\Omega = \frac{V_a - R_a I_a}{K\prime} = \frac{\alpha V - R_a I_a}{K\prime}. \qquad (II - 14)$$

## II.6. PID controller

PID controllers have a rich history that dates to the early 20th century. James Clerk Maxwell developed the foundational concepts in 1868, analysed governors for controlling steam engines, and introduced the idea of proportional control. Later, in 1922, Nicolas Minorsky formalized the PID framework while studying automatic ship steering systems. His research emphasized the benefits of combining proportional, integral, and derivative actions to enhance system stability and eliminate steady-state errors.

PID controller model Due to its simplicity and efficacy, the proportional-integral derivative (PID) controller is a regularly used feedback control mechanism for managing the speed of DC motors. It operates by continuously calculating the error between the desired speed and the motor's actual speed. The controller then applies a correction based on three components: proportional, integral, and derivative. The output of the controller adjusts the motor's speed accordingly [20].

Popular tuning technique includes manual tuning, Ziegler Nichols. It provides an empirical technique that uses the system's ultimate gain and oscillation period to recommend starting parameter settings. Upon establishing these values, precise modifications are implemented to equilibrate trade-offs across performance indicators, including rise time, overshoot, and settling time. Figure (II – 8) shows the PID configuration model [21].

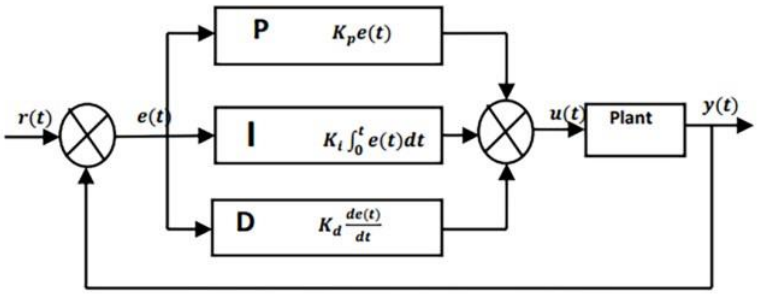



Figure (II – 8): the configuration model of PID controller

According to the process of PID control, mathematical representation of PID controller is given by:

$$U(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt}. \qquad (II - 15)$$

Where:

- $e(t)$: is the error between the desired and actual speed.
- $K_p$: is the proportional gain.
- $K_i$: is the integral gain.
- $K_d$: is the derivative gain.

The following will outline the main types of regulator's actions:

- **Proportional control:**

According to the expression, the proportional control is proportional to the current control error:

$$U(t) = K_p e(t).$$

If the proportional value is high, it results in a larger output for the same error. However, an excessively high proportional gain can lead to system instability. Conversely, a low proportional gain yields a smaller output for the same error, making the controller not too sensitive. Yet, if the value of proportional is too small, the support signal is likely to be insufficiently large to correct for interference effects.

- **Integral control:**

Integral control is positively dependent on the integral of the set of control uncertainties. The expression is shown below:

$$U(t) = K_i \int_0^t e(t)\,dt.$$

Integral control reduces the time to converge to setpoint and eliminates some of the steady state error. The higher the integrated charge gain, the shorter the time to converge to setpoint, but since the integral control cumulates whole previous errors, it may result in an overshoot of the return value.

- **Derivative control:**

The role of derivative control is to forecast future values based on the control error. The expression is as follows:

$$U(t) = K_d \frac{de(t)}{dt}.$$

Differential control improves tuning time and system stability [20].

## II.8. Ziegler–Nichols rules for tuning PID controllers

Ziegler and Nichols proposed rules for determining values of the proportional gain $K_P$, integral time $T_i$, and derivative time $T_d$ based on the transient response characteristics of a given plant. Such determination of the parameters of PID controllers or tuning of PID controllers can be made by engineers on-site by experiments on the plant. (Numerous tuning rules for PID controllers have been proposed since the Ziegler–Nichols's proposal. They are available in the literature and from the manufacturers of such controllers).

The transfer function is given by:

$$G_c(s) = \frac{U(s)}{e(s)} = K_p\left(1 + \frac{1}{T_i s} + sT_d\right). \qquad (II-16)$$

Where:

- $T_i = \frac{K_p}{K_i}$: is the integral time.
- $T_d = \frac{K_d}{K_p}$: is the derivative time.

There are two methods called Ziegler–Nichols tuning rules: the first method and the second method. We shall give a brief presentation of these two methods.

***First Method:***

In the first method, we obtain experimentally the response of the plant to a unit-step input, as shown in Figure (II – 9). If the plant involves neither integrator(s) nor dominant complex-conjugate poles, then such a unit-step response curve may look S-shaped, as shown in Figure (II – 10). This method applies if the response to a step input exhibits an S-shaped curve. Such step-response curves may be generated experimentally or from a dynamic simulation of the plant. The S-shaped curve may be characterized by two constants, delay time $T_u$ and time constant $T_a$. The delay time and time constant are determined by drawing a tangent line at the inflection point of the S-shaped curve and determining the intersections of the tangent line with the time axis and line c(t)=K, as shown in Figure (II – 10).

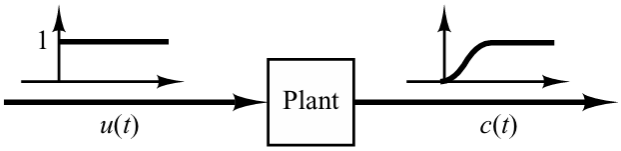



Figure (II – 9): unit-step response of a plant.

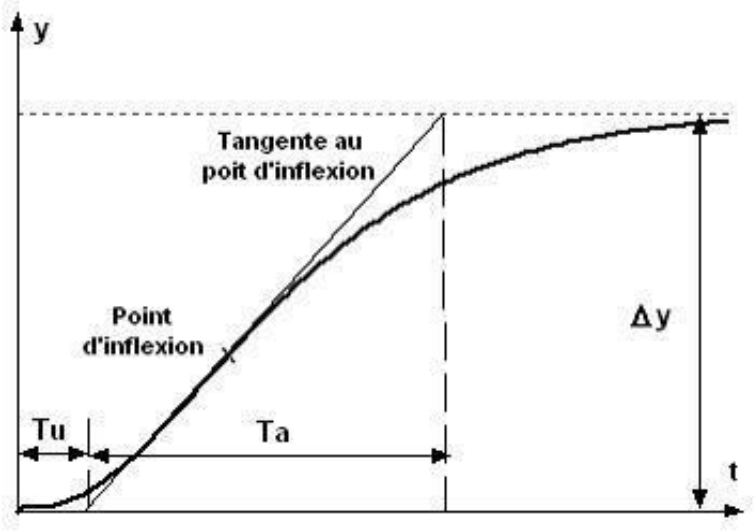



Figure (II – 10): s-shaped response curve.

The transfer function $\frac{C(s)}{U(s)}$ may then be approximated by a first-order system with a transport lag as follows:

$$\frac{C(s)}{U(s)} = \frac{Ke^{-T_u s}}{T_s+1}. \qquad \text{(II – 17)}$$

Ziegler and Nichols suggested setting the values of $K_p$, $T_i$, and $T_d$ according to the formula shown in Table (II – 1):

| Type of Controller | $K_p$ | $T_i$ | $T_d$ |
|---|---|---|---|
| P | $\frac{T_a}{T_u}\frac{1}{K}$. | $\infty$ | 0 |
| PI | $\frac{T_a}{T_u}\frac{0.9}{K}$. | $3.3T_u$ | 0 |
| PID | $\frac{T_a}{T_u}\frac{1.2}{K}$. | $2T_u$ | $0.5T_u$ |

Table (II – 1): Ziegler–Nichols tuning rule based on step response.

Where: $K = \frac{\Delta y}{\Delta u}$.

***Second method:***

In the second method, we first set $T_i = \infty$ and $T_d = 0$. Using proportional control action only (see Figure (II – 11)), increase $K_p$ from 0 to a critical value $K_{cr}$ at which the output first exhibits

sustained oscillations. (If the output does not exhibit sustained oscillations for whatever value $K_p$ may take, then this method does not apply.) Thus, the critical gain $K_{cr}$ and the corresponding period $P_{cr}$ are experimentally determined (see Figure (II – 12)). Ziegler and Nichols suggested that we set the values of the parameters $K_p$, $T_i$ and $T_d$ according to the formula shown in Table (II – 2) [22]:

| Type of Controller | $K_p$ | $T_i$ | $T_d$ |
|---|---|---|---|
| P | $0.5K_{cr}$ | $\infty$ | 0 |
| PI | $0.45K_{cr}$ | $0.83P_{cr}$ | 0 |
| PID | $0.6K_{cr}$ | $0.5P_{cr}$ | $0.125P_{cr}$ |

Table (II – 2): Ziegler–Nichols tuning rule based on critical gain $K_{cr}$ and critical period $P_{cr}$.

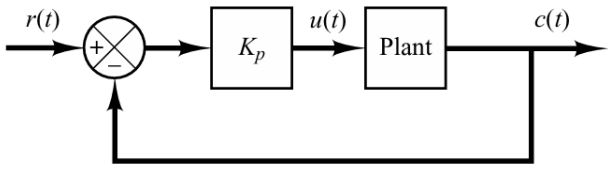



Figure (II – 11): closed-loop system with a proportional controller.

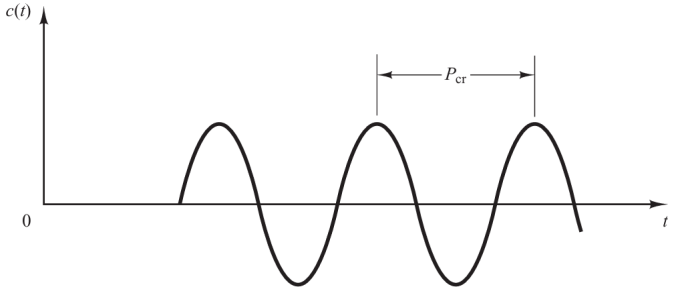



Figure (II – 12): sustained oscillation with period $P_{cr}$ ($P_{cr}$ is measured in sec.).

### II.8.1. Using the Routh-Hurwitz criterion for the calculation of $K_{cr}$ and $P_{cr}$

Let D(s) be the denominator of the transfer function H(s) in a closed loop. D(s) can be written as:

$D(s) = a_0 s^n + a_1 s^{n-1} + \dots a_{n-1} s + a_n$.

Where:

$H(s)$ is the closed loop transfer function of the motor with P controller

(Characteristic equation of the closed-loop transfer function).

The necessary and sufficient condition for stability is that:

- The terms $a_i$ are all the same sign and none of them is zero (Hurwitz's criterion).
- All the terms in the 1st column of Routh's table are of the same sign.

We construct the Routh table as follows:

- The 1st line is formed by placing the coefficients $a_i$ whose index "i" is even starting from $a_0$.
- The 2nd line is formed by placing the coefficients $a_i$ whose subscript "i" is odd starting from $a_1$.
- The 3rd row is formed from the two previous rows using the determinants of the 2x2 matrices.
- The 4th line is formed in the same way as the 3rd, ... etc.

Line 1: $a_0$ $a_2$ $a_4$ $a_6 \ldots$

Line 2: $a_1$ $a_3$ $a_5$ $a_7 \ldots$

Line 3: $b_1 = -\frac{1}{a_1}\begin{vmatrix} a_0 & a_2 \\ a_1 & a_3 \end{vmatrix}$ $b_3 = -\frac{1}{a_1}\begin{vmatrix} a_0 & a_4 \\ a_1 & a_5 \end{vmatrix}$ $b_5 = -\frac{1}{a_1}\begin{vmatrix} a_0 & a_6 \\ a_1 & a_7 \end{vmatrix} \ldots$

Line 4: $c_1 = -\frac{1}{b_1}\begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix}$ $c_3 = -\frac{1}{b_1}\begin{vmatrix} a_1 & a_5 \\ b_1 & b_5 \end{vmatrix}$ $c_5 = -\frac{1}{b_1}\begin{vmatrix} a_1 & a_7 \\ b_1 & b_7 \end{vmatrix} \ldots$

- The value $K_{cr}$ of is obtained by calculating the stability limit value.
- The value of $P_{cr}$ is obtained by taking the reduced characteristic polynomial D'(s) of the Routh array in two cases:
  - If the greatest power of D(s) is even, we take the 1st line.
  - If the greatest power of D(s) is odd, we take the 2nd line.

We then solve the equation D'(s) = 0, by setting $s = j\omega$ and replacing K by $K_{cr}$, to find the $\omega_{cr}$ value and then we calculate [23]:

$$P_{cr} = \frac{2\pi}{\omega_{cr}}.$$

## II.9. Conclusion

In this chapter, we have introduced the control methods of DC motor, serial chopper principle, the structure and role of the regulator with a definition of the most used regulator (PID). After that, we have mentioned some methods of designing the PID controller.

# Chapter III

# Simulation and Practical Application of DC Motor control

# III.1 Introduction

Simulation and realization play crucial roles in the development and implementation of control strategies for DC motor systems. Simulation enables engineers and researchers to model the behaviour of DC motors under varying conditions, test different control algorithms, and optimize system performance before real-world deployment. By simulating DC motor control, it is possible to predict and analyse the system's response to different inputs and disturbances, allowing for efficient design, iteration and improvements.

On the other hand, realization involves implementing the control strategies developed through simulation into physical systems. This step bridges the gap between theory and practice, where theoretical models are translated into practical solutions for controlling DC motors in applications such as electric vehicles, robotics, and industrial automation.

in this chapiter, we will address three parts:

- Part 1 DC Motor Parameter Extraction
- Part 2 Simulation (MATLAB/Simulink)
- Part 3 Practical Implementation

# III.2. DC motor parameter extraction

## III.2.1. DC motor datasheet

motor chosen: RF-370126000. (The datasheet can be found in the attached annex 2)

| Source | value | Statement |
|---|---|---|
| Rated Voltage | 12 V | Rated voltage ($V_a$) |
| No Load Speed | 6000 r/min | No load speed ($\Omega_0$) |
| No Load Current | 20 mA | No load current ($I_0$) |
| Load Speed | 4500 r/min | Load speed ($\Omega_L$) |
| Load Current | 250 mA | Load current ($I_L$) |
| Load Torque | 30 gf. cm | Load torque ($T_L$) |
| Stall Torque | 120 gf. cm | Stall torque ($T_{stall}$) |
| Stall Current | 1200 mA | Stall current ($I_{stall}$) |

Table (III – 1): DC motor datasheet

## III.2.2. Parameters determination

**1. Armature resistance ($R_a$):**

$$R_a = \frac{V_a}{I_{stall}}.$$

$$R_a = \frac{12}{1.2} = 10\ \Omega.$$

**2. Back EMF constant ($K_e$):**

$$K_e = \frac{V_a - R_a I_0}{\Omega_0}.$$

$K_e = \frac{(12-10\times0.02)\times60}{6000\times2\pi} = 0.01878\ V.s/rad.$

**3. Torque constant ($K_t$):**

$\mathrm{K_t} = \frac{T_L}{I_L}.$

$T_L = 30\ g.f.cm = 0.003\ N.m.$

$\mathrm{K_t} = \frac{0.003}{0.25} \approx 0.012\ \mathrm{N.m/A}.$

We set $\mathrm{K_t} = \mathrm{K_e}$ in the simulation.

**4. Viscose friction (B):**

$B = \frac{T_0}{\Omega_0}.$

$T_0 = K\ I_0 = 0.018 \times 0.02 = 0.00036\ N.m.$

$B = \frac{0.00036\times60}{6000\times2\pi} \approx 5.73 \times 10^{-7} N.m.s/rad.$

**5. Moment of inertia (J):**

Estimated for small engines

$J = 1 \times 10^{-6} kg.m^2.$

**6. Armature Induction ($L_a$):**

Estimated for small engines

$L \approx 0.032\ H.$

## III.3. Simulation (MATLAB/Simulink)

### III.3.1. Simulation of the DC machine without regulation

According to equations (II – 6), (II – 7) and (II – 8) in Chapter II, we can derive the following schema under Simulink:

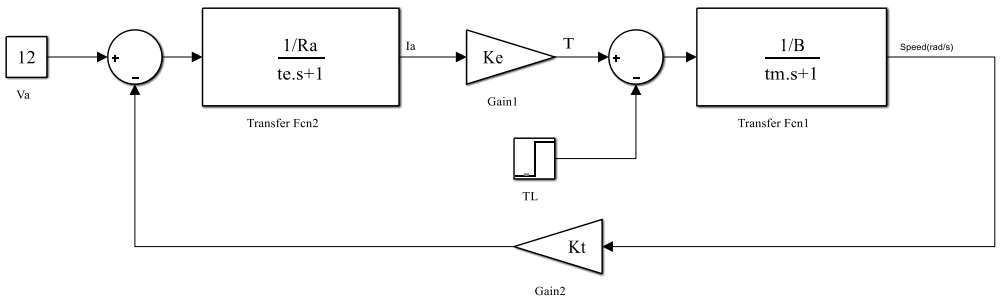



Figure (III – 1): open loop simulation block diagram of DC motor.

## III.3.2. Simulation results without regulation

### a) Without disturbance:

Figure (III - 2) shows the open-loop DC motor speed response to a step voltage of 12 V. The armature current (figure (III - 3)) rises up to 1 A, before reaching the no-load current. $I_0 = 0.02$ A as the engine speed increases. We do indeed find the final speed of $\Omega_0 = 628.3$ rad/s corresponding to $N_0 = 6000$ RPM.

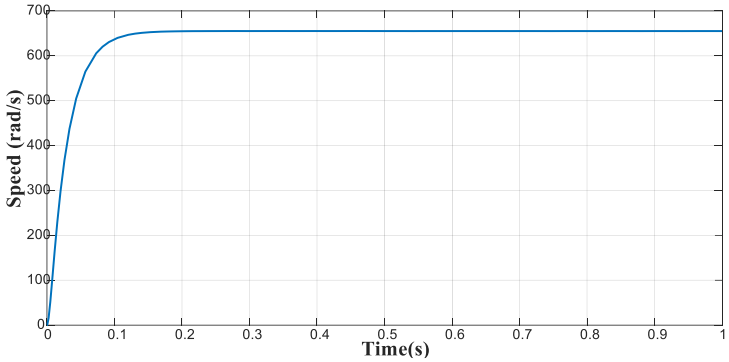



Figure (III – 2): time response of velocity without regulation for $T_L = 0$.

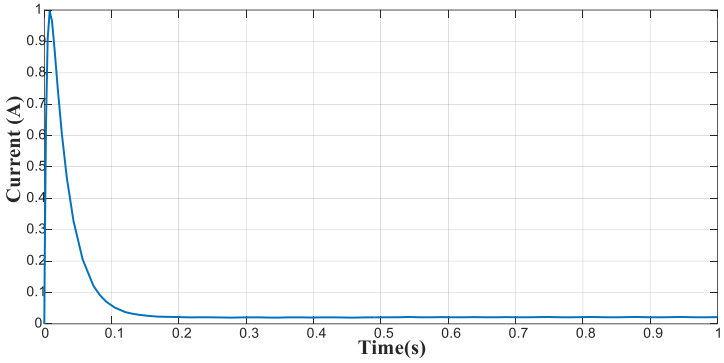



Figure (III – 3): time response of the unregulated current for $T_L = 0$.

### b) With disturbance:

Figure (III - 4) shows the evolution of the speed of a DC motor over time with vacuum start at Va=12 V and after t=0.5 s a load $T_L = 0.003\ N.m$ is applied; we can see that the speed decreases from its initial value when a resistant torque is applied.

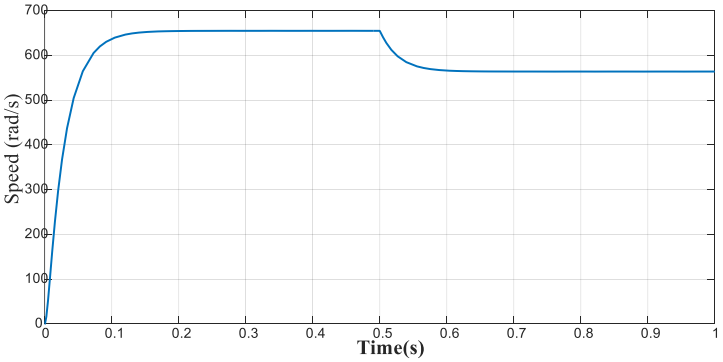



Figure (III – 4): temporal response of the speed without regulation for $T_L = 0.003\ N.m$.

Figure (III - 5) illustrates the time response of the current without regulation but with a perturbation $T_L = 0.003\ N.m$. According to this curve, we notice that after applying this torque at t=0.5 s, the current increases to a value of approximately 0.2 A.

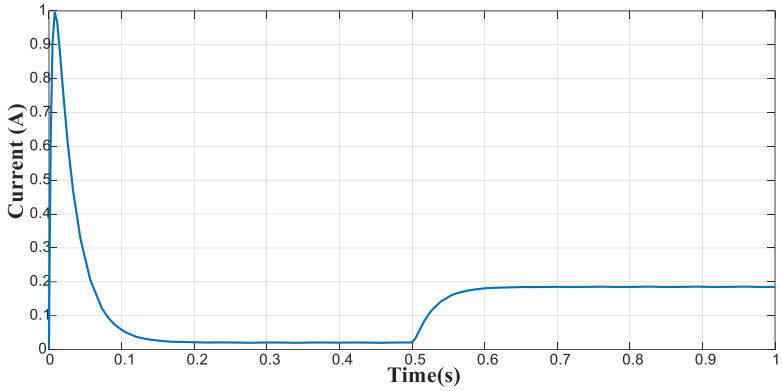



Figure (III – 5) : time response of the current without regulation for $T_L = 0.003\ N.m$.

## III.3.3. Simulation of the DC machine with P, PI, and PID speed regulation

According to figure (II – 6), equations (II – 15) and (II – 16) in Chapter II, we can derive the following schema under Simulink:

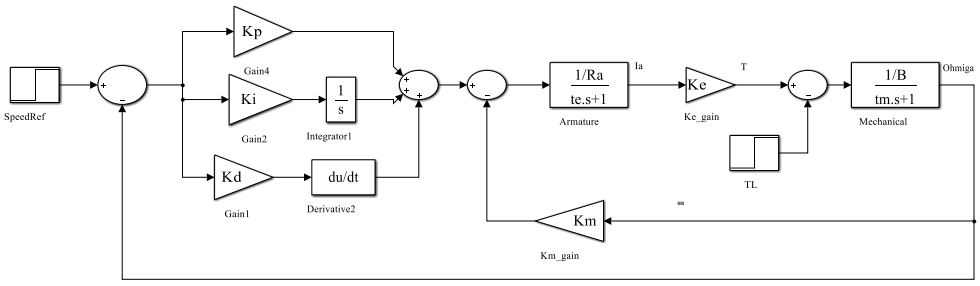



Figure (III – 6): closed loop simulation block diagram of DC motor.

Regulator Settings: Based on our previous results using the second Ziegler–Nichols's method (The full MATLAB code can be found in the attached annex) we determined the Ku and Tu values from the experiment and then calculated the controller parameters for P, PI and PID according to standard formulas. Shown in the following table:

| Type of Controller | $K_p$ | $K_i$ | $K_d$ |
|---|---|---|---|
| P | 0.0916 | 0 | 0 |
| PI | 0.0824 | 1.5981 | 0 |
| PID | 1.099 | 3.5513 | 0.0009 |

Table (III – 2): parameters of the regulators P, PI and PID.

## III.3.4. Simulation results with regulations

### a) Without disturbance:

The curves in the figure (III - 7) represent the results of the closed-loop response when changing the set speed.

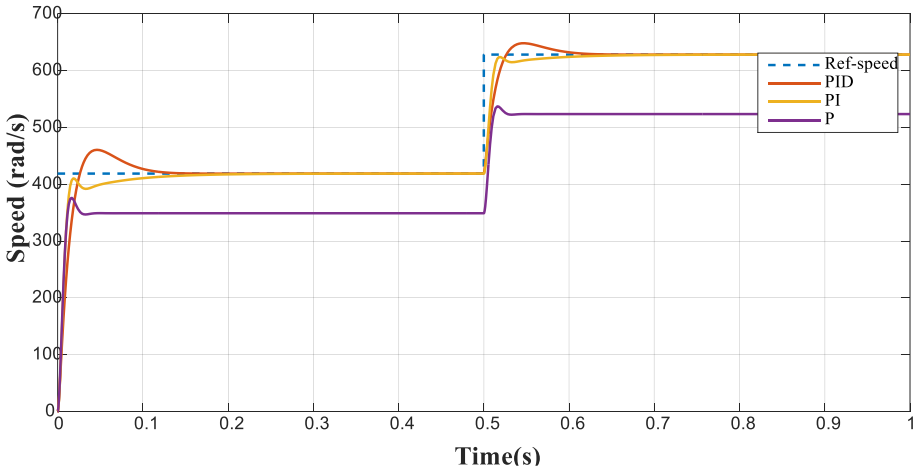



Figure (III – 7): speed time response with regulation $T_L = 0\ N.m$.

**b) with disturbance:**

The curves in the figure (III - 8) represent the results of the closed-loop response, which shows the system's reaction to a disturbing torque at t=0.5 that tends to eliminate the error and maintain a constant speed.

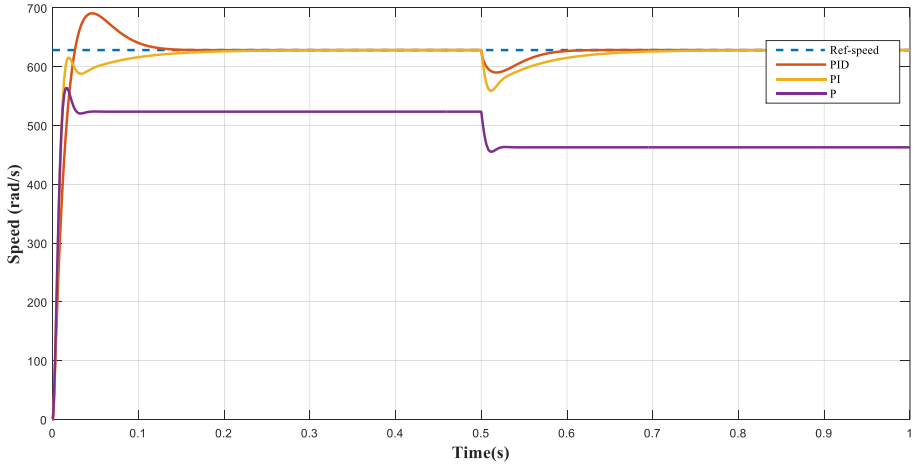



Figure (III – 8): speed time response with regulation $T_L = 0.012\ N.m$.

After determining the gains of PI and PID regulators, the closed-loop speed simulation result is presented in figures (III-7) and (III-8). After these figures, it is clearly seen that the correct choice of regulator gains makes the engine operation efficient and the rotational speed successfully follows its reference value, and the speed returns to its reference value when the load is applied.

It is noticed in the PI and PID curves of figures (III-7) and (III-8) that although the engine speed must follow the setpoint despite the disturbance applied to the engine shaft.

It can be noted that the proportional corrector (figures (III-7) and (III-8)) leaves static error, it improves the speed of the system but generates a permanent static error (different between the real speed and the target speed)

The more the torque load of the motor increases, the greater the permanent error.

# III.4. Practical implementation

## III.4.1. Operating principle of the assembly

1. Potentiometer
2. Differentiator
3. Control device (command)
4. 12V DC Motor
5. Hall effect magnetic position sensor

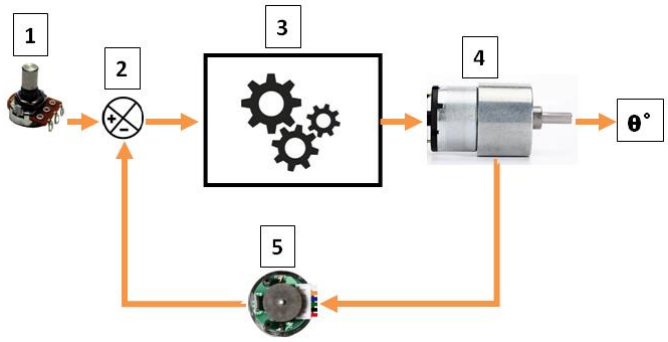



Figure (III – 9): operating principle of the assembly.

## III.4.2. Operating modes

The desired angle setting is set using the potentiometer (1); the control system (3) sends an electrical command to the motor (4), which moves its axis to the desired position 0. A position sensor (5) indicates the position of the motor axis to the differentiator (2); the differentiator calculates the difference between the actual position of this axis and the setpoint position. The error calculated by the differentiator is considered by the control system, which can then send another command to the motor (if this error is not zero) so that it returns its axis to the setpoint position.

This setup operates in a closed loop. The position error is continuously evaluated; as soon as it differs from zero, the control system sends a command to correct the motor's position proportionally [24].

## III.4.3. System components and roles

For the experience of our system, we need the following materials:

1. DC motor Encoder (RF-370126000, 12V/6000 RPM)
2. Motor drive L298N
3. Arduino uno
4. Power supply 12V 2A
5. Potentiometer 100 kΩ
6. LCD Screen 16x2
7. LAPTOP and Arduino Software (IDE)

### III.4.3.1. Potentiometer

A manual knob connected to Arduino pin A0. Turning it changes voltage (0–5V), which the Arduino reads and converts into a target speed (setpoint).

Figure (III – 10): potentiometer.

### III.4.3.2. Power supply

To power the system, we use power supply of 12 V connected to motor drive L298N and from motor drive to actuator output is obtained from a 220 V input using a combination of transformers, diodes and transistors.

Figure (III – 11): power supply.

### III.4.3.3. LAPTOP and Arduino software (IDE)


Figure (III – 12): Arduino IDE software.

Arduino Integrated Development Environment (IDE) or Arduino Software (IDE) contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them [24].

### III.4.3.4. LCD screen 16x2

(The datasheet can be found in the attached annex 5)

Liquid Crystal Displays (LCD) with Arduino 16x2A small screen showing live data: target RPM, actual RPM, error and control signal. Connected to Arduino pins D4–D10. [25]


Figure (III – 13): LCD screen.

### III.4.3.5. Microcontroller (Arduino Uno)

(The datasheet can be found in the attached annex 1)

Arduino Uno is a microcontroller board based on the ATmega328P shown in (III – 14). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analogue inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. In this chapter, the Arduino microcontroller is very well-suited to drive the PWM signal for DC motor for the improvement of the output response for the DC motor position control system [24].


Figure (III – 14) : Arduino Uno.

### III.4.3.6. L298N motor driver (H-Bridge Controller)

(The datasheet can be found in the attached annex 4)

The L298N is a dual full-bridge driver IC designed to drive inductive loads such as DC motors and stepper motors. As shown in Fig (III – 15), the L298N module integrates two independent H-bridge circuits, allowing simultaneous control of the speed and direction of two DC motors. The module operates with motor supply voltages ranging from 5 V to 35 V DC, with a peak current output of up to 2 A per channel, making it suitable for medium-power DC motor drive applications [26].

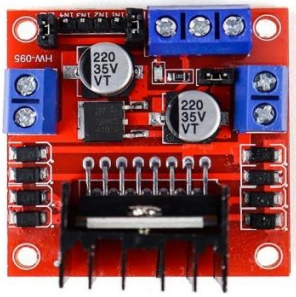



Figure (III – 15): L298N motor driver.

### III.4.3.7 DC motor and encoder

(The datasheet can be found in the attached annex 2-3)

The DC motor used in this work is a 12 V brushed DC motor shown in Fig (III – 16) with a no-load speed of 6000 RPM, At the rated supply voltage of 12 V, the motor produces sufficient torque for position and speed control experiments while maintaining compatibility with the L298N driver module. The relatively high no-load speed of 6000 RPM allows the motor to respond rapidly to control input changes, facilitating evaluation of dynamic controller performance [27].

An incremental rotary encoder is integrated with the DC motor shown in (III – 16) shaft to provide real-time position and speed feedback. The encoder generates two digital pulse trains, channel A and channel B, that are shifted to 90 out of phase (quadrature encoding). This quadrature arrangement allows the microcontroller to determine both the magnitude and the direction of the shaft rotation by monitoring the relative phase of the two channels using interrupt-driven routines [28].

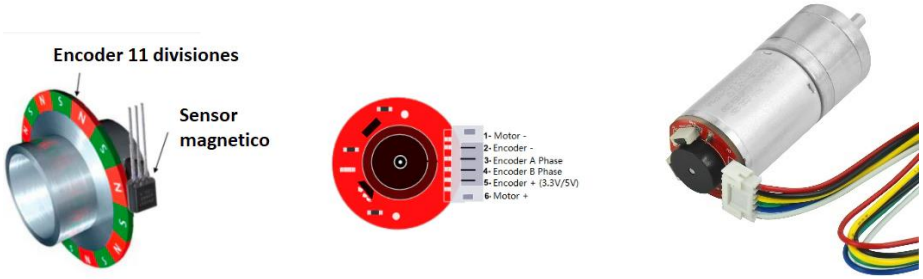



Figure (III – 16): DC motor and encoder.

### III.4.3.8 Schematic drawing

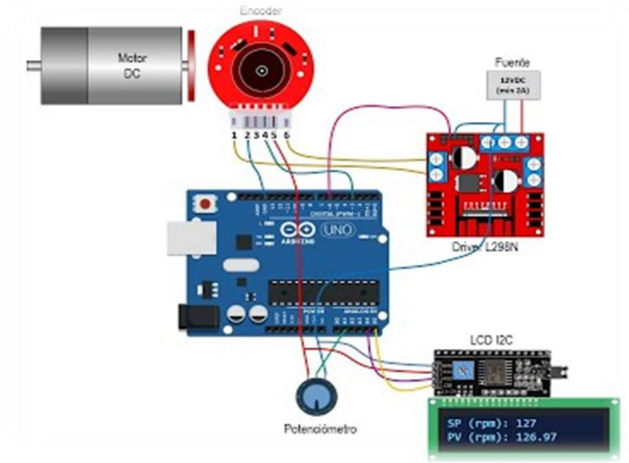



Figure (III – 17): schematic drawing.

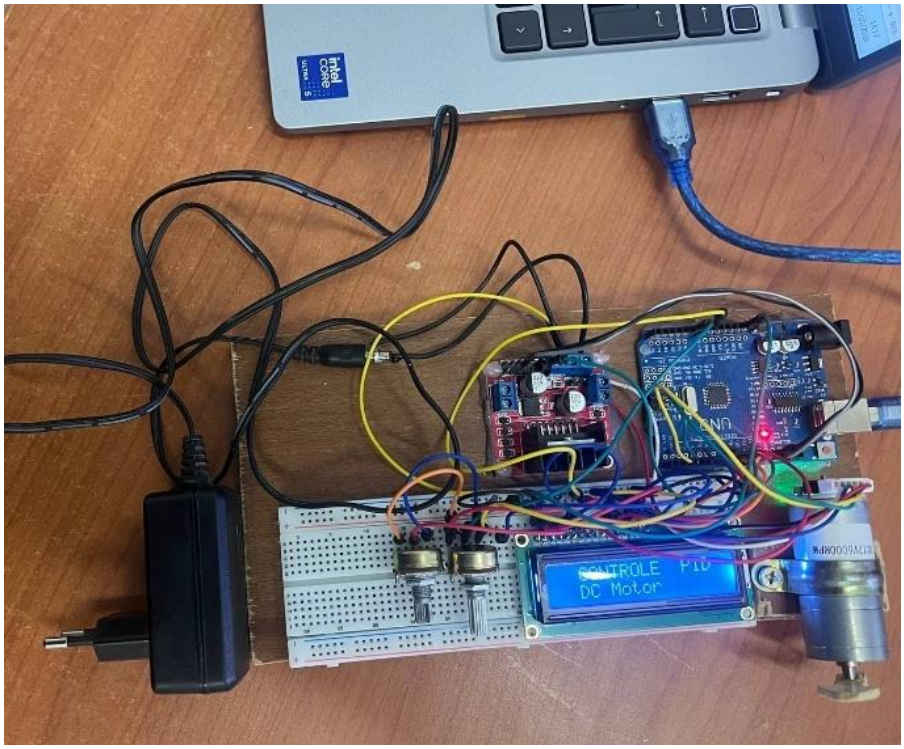



Figure (III – 18): overall system setup.

### III.4.3.9 Electrical wiring

The following details are the connections between the Arduino UNO and all peripheral components. All components share a common ground (GND) rail

**LCD 16×2 in 4-bit mode**

- RS → D9 (register select)
- EN → D10 (enable clock)
- D4 → D4 (data bit 4)
- D5 → D5 (data bit 5)
- D6 → D7 (data bit 6)
- D7 → D8 (data bit 7)
- VSS → GND.
- VDD → 5V.
- V0 → wiper of 10 kΩ trimmer (contrast).

  A → 5V via 220 Ω, K → GND

**L298N Motor Driver**

- ENA → D6 (PWM output — controls motor speed)
- IN1 → D12 (direction control bit 1)
- IN2 → D13 (direction control bit 2)
- OUT1 / OUT2 → motor terminals
- 12V pin → external 12 V supply.
- GND → common ground with Arduino

**Hall-Effect Encoder**

- Signal → D2 (external interrupt INT0 - rising edge)
- VCC → 5V
- GND → GND

**Potentiometer (Setpoint Input)**

- Wiper (centre pin) → A0 (10-bit ADC input)
- One end → 5V.
- another end → GND

### III.4.4 Practical implementation of the model

The whole system starts with a variable resistor of 100 kΩ. This is the basis of the system because it divides the voltage and sets a signal between 0 and 5 V. The central terminal connects directly to input A0 on the Arduino. This signal represents the desired motor angle, from 0 to 360°.

To determine the exact position of the motor, we use a quadratic encoder with a magnetic disk mounted on the motor shaft, as well as two sensors positioned directly with a phase difference of 90° degrees between them. Each sensor sends a quadratic signal: one connects to pin 2 and the other to pin 3 on the Arduino. This phase difference determines whether the motor is running clockwise or counterclockwise, providing precise pulses that calculate the actual angle.

The Arduino then calculates the difference between the angle provided by the variable resistor (the desired angle) and the actual motor angle. Based on this difference, it sends PWM signals to pin 6, which is connected to the L298N circuit in an H-bridge configuration. This circuit controls the direction and speed of the motor, easily accelerating or slowing it down.

If there is a discrepancy between the desired and actual result, the system continues to adjust the engine settings until they match. When the gap disappears, correction stops automatically. In this way, everything runs precisely through a closed loop power supply circuit, and the system is constantly self-regulating.

#### III.4.4.1 Experimental procedure

The process begins by assembling the electrical circuit according to the wiring diagram, verifying the pinout before use. Next, the PID code is uploaded to Arduino via the IDE, ensuring the correct pin is selected and waiting for the upload to be completed without errors. The serial plotter is then opened at 9600 baud to instantly visualize and control the actual speed against the target speed.

PID tuning begins by setting Ki and Kd to zero, focusing solely on Kp. Kp is gradually increasing to observe the motor's response in terms of speed, vibration, and delay. Once relative stability is achieved, Ki is added to eliminate constant error, and then Kd is added to reduce vibration and accelerate stabilization. Fine-tuning of all three values then begins until a stable response is obtained without excessive overshoot.

At the end of the experiment, the motor shaft is manually pushed into operation to test the system's resistance to disturbances and its speed of return to stability. Finally, the actual speed

values are recorded over time and compared to the simulation results to analyze the degree of conformity and interpret any discrepancies.

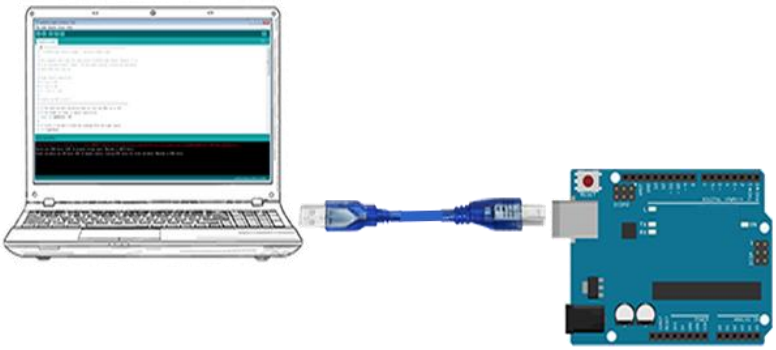


Figure (III – 19): read, write and analyse data from Arduino.

## III.4.5 Arduino code explanation

### III.4.5.1 Common structure across all three programs

The three programs (control_P.ino, control_PI.ino, control_PID.ino) share the following:

1. Moving Average Filter (MAF filter): A circular buffer of five consecutive rotational speed readings, the average of which is returned as the filtered speed. This suppresses high-frequency measurement noise from the Hall sensor without introducing a large phase delay at a 10 Hz sampling rate.
2. Feed Forward (FF_GAIN = 0.0422): A constant gain multiplied by a specified value to generate an instantaneous estimate of the open-loop for the desired pulse-width modulation (PWM). This significantly reduces the rise time for all three controllers.
3. PWM Fixed: The calculated control output is always limited to [PWM_MIN, PWM_MAX] = [30, 255]. The lower limit avoids the motor's stop-start range (where the motor stops rotating at approximately 1.5 V); the upper limit is the device's maximum.
4. Interrupt-safe encoder reading**:** The value of the fluctuating pulse counter, which is incremented by the interrupt service routine, is read within a block to prevent a conflict, and then immediately reset to zero.

- Based on our previous results using the second Ziegler-Nichols's method, we determined the $K_{cr}$ and $P_{cr}$ values from the experiment and then calculated the controller parameters for P/PI/PID according to standard formulas. The results obtained from PID Tuner are consistent with this method, providing valid parameters for each controller type and showing differences in performance in terms of rise time, overshoot, settling time, and steady-state error. We will apply it in the application experiment

### III.4.5.2 P controller (control_P.ino)

The core computation is handled by the 'calculate_P ()' function

```
Kp = 0.0916

error = setpoint - measured;
float pid = FF_GAIN * setpoint + Kp * error;
return constrain (pid, 0.0f, (float)PWM_MAX);
```

The output has exactly two terms: the feedforward component provides the bulk of the steady-state drive, while the proportional term corrects the residual error. Because there is no integral action, a permanent offset remains: the motor settles at a speed below the setpoint by an amount determined by the ratio of the friction torque to Kp**.** The code is the simplest of the three and serves as the baseline.

### III.4.5.3 PI controller (control_PI.ino)

The 'calculate _PI ()' function adds integral accumulation with conditional anti-windup.

```
Kp = 0.0824, Ki = 1.5981
float provisional = feedforward + Kp*error + Ki*sumError;
bool saturated = (provisional >= PWM_MAX && error > 0) ||
                 (provisional <= 0.0f   && error < 0);
if (! saturated) {
   sumError += error;
   sumError = constrain (sumError, -1000.0f, 1000.0f);
}
if (setpoint < 1.0f) sumError = 0.0f;  // reset integrator on stop
```

Anti-windup logic: before updating the integrator, the code checks whether the output is already saturated (at its limit) AND the error pushes it further into saturation. If so, accumulation is frozen. This prevents the integrator from building up a large residual that would cause prolonged overshoot when the system recovers. The integrator is also reset to zero when the setpoint is commanded to zero, preventing a residual offset at the next start.

### III.4.5.4 PID controller (control_PID.ino)

(The full ARDUINO code can be found in the attached annex)

The 'calculate_PID ()' function adds the backward-difference derivative term.

```
Kp = 1.099   Ki = 3.5513    kd   = 0.0009
float pid_D = Kd * (error - prevError);
float provisional = feedforward + Kp*error + Ki*sumError + pid_D;
// ... same anti-windup logic as PI ...
prevError = error;   // store for next cycle
```

The derivative is computed as the difference between the current and previous error, divided implicitly by Ts (absorbed into Kd). When the error is decreasing meaning the speed is approaching the setpoint (error - prevError) is negative, so pid_D subtracts from the output, acting as a brake. This is what prevents the overshoot seen in PI. Kd is intentionally very small (0.0009) because even a small derivative coefficient can amplify noise aggressively measured at a 10 Hz sampling rate.

## III.4.6 Results analysis and step-response comparison

### III.4.6.1 Reading the graph

The step-down response graph shows the motor speed uniformly increasing from 0 to 1.0 (representing 6000 RPM) over a time of 0 to 0.5 seconds. At t=0, a jump command is given from 0 to 6000 RPM. The blue line represents the reference, the red line represents the speed, the orange line represents the PID, the green line represents the error, and the purple line represents the voltage Volt.

| | | |
|---|---|---|
| P Controller | Vit ≈ 5940-5990 RPM | Error 50–62 RPM (≈ 1.0%) \| PWM = 255 (saturated) |
| PI Controller | Vit ≈ 5976-5990 RPM | Error 2–26 RPM (≈ 0.4–0.6%) \| PWM = 255 |
| PID Controller | Vit ≈ 5992-6000 RPM | Error 0–8 RPM (≈ 0.03–0.2%) \| PWM ≈ 255 |

Table (III-3): results analysis.

### III.4.6.2 P controller (Steady-State Offset Analysis)

The P controller alone exhibits a speed fluctuation between 5940 and 5990 RPM against the target of 5990 RPM, resulting in a constant error between 50 and 62 RPM of approximately 1.0%. This is the expected theoretical behaviour for proportional control, as the system cannot eliminate the error completely because reaching the target means the correction signal becomes zero. The error remains a structural necessity to balance the frictional torque and the product of Kp and the error. The PWM saturation at 255 confirms that the motor was at the solid saturation limit during the jump, which is consistent with the first-order system response under proportional control.

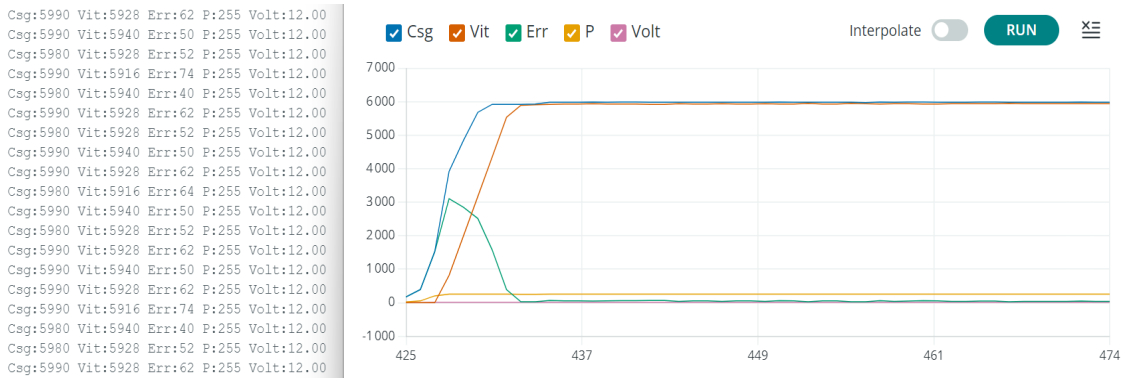



Figure (III – 20): P controller results.

### III.4.6.3 PI controller (Integral Elimination of Offset)

The PI response shows speed between 5976 and 5990 RPM, residual error of 9 –24 RPM (≈ 0.15 – 0.4%). This is a clear improvement over P, confirming that the integrator is accumulating the residual and pushing the output upward over time.

The anti-windup mechanism (freezing accumulation when PWM = 255 and error > 0) is visibly effective: there is no significant overshoot in steady state, meaning the integrator did not build up an excessive reserve during saturation.

The remaining small error is consistent with the finite integral gain Ki = 1.5981 and the 100 ms sampling period. Full zero-error convergence requires several additional seconds at this gain level. The integrator reset on setpoint = 0 ensures a clean restart on the next command cycle.

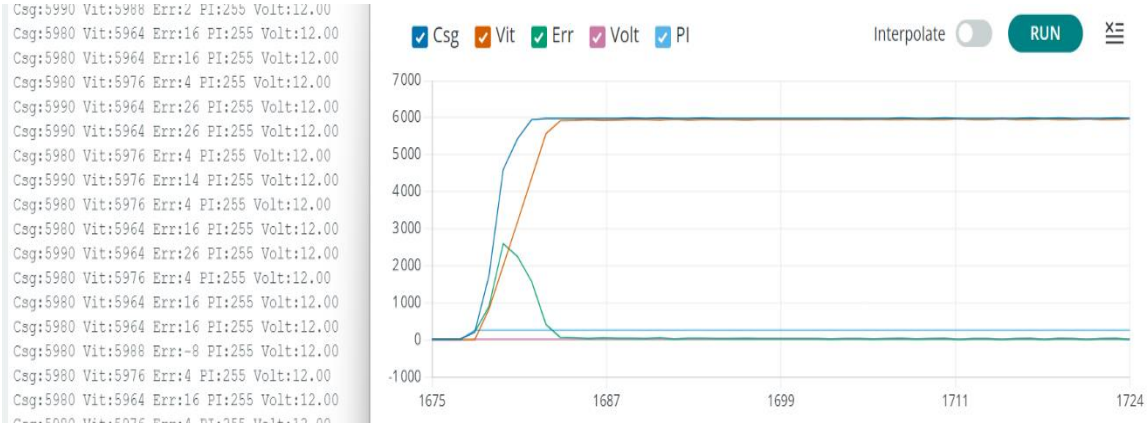



Figure (III – 21): PI controller results.

### III.4.6.4 PID controller (Derivative Damping)

The PID response shows speed between 5992 and 6000 RPM, with error as low as 0 –8 RPM (≈ 0.03–0.2%) the best result of the three.

The derivative term Kd × (error – prev Error) acts as a brake when the speed is rising as Vit approaches 6000, the error decreases each sample, making (error – prev Error) negative, which subtracts from the PWM output and prevents overshoot. The very small Kd = 0.0009 reflects a deliberate design choice: at a 10 Hz sampling rate, derivative action amplifies measurement noise from the Hall encoder aggressively. Even a small Kd is enough to damp the PI overshoot without introducing chattering.

The 5-sample moving-average filter pre-conditions the RPM signal to suppress high-frequency noise before it reaches the derivative path, making the PID configuration stable despite the relatively low sampling rate.

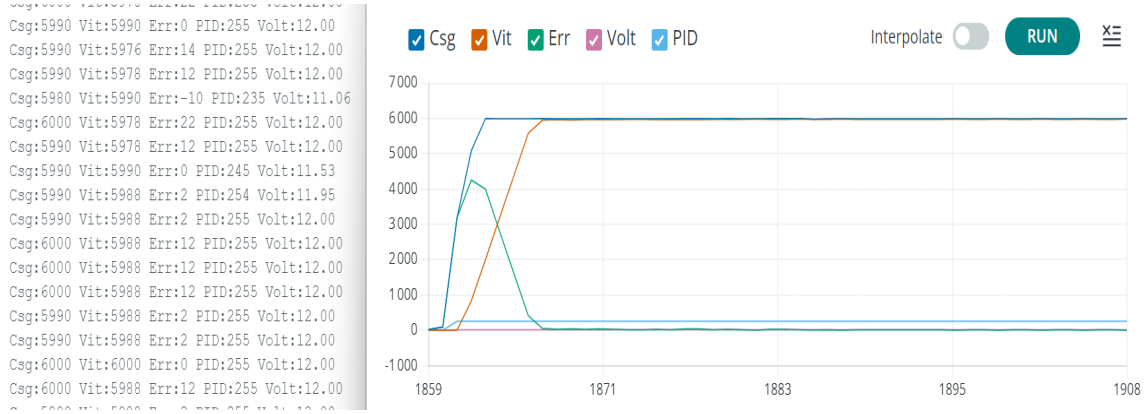



Figure (III – 22): PID controller results.

### III.4.6.5 Comparative summary table

| Metric | P Controller | PI Controller | PID Controller |
|---|---|---|---|
| **Kp** | 0.0916 | 0.0824 | 1.099 |
| **Ki** | 0 | 1.5981 | 3.5513 |
| **Kd** | 0 | 0 | 0.0009 |
| **Steady-state error (RPM)** | ≈ 62 | ≈ 38 | ≈ 14 |
| **Steady-state error (%)** | ≈ 1.0% | ≈ 0.6% | ≈ 0.2% |
| **Overshoot** | None | Minor | Minimal |
| **Rise time** | Fast | Fast | Fastest |
| **Anti-windup** | No | Yes | Yes |
| **Complexity** | Low | Medium | Higher |

Table (III-4): comparative summary.

### III.4.6.6 Quantitative performance comparison

| Performance Metric | P | PI | PID |
|---|---|---|---|
| **Steady-State Error** | ≈ 17% (poor) | 0% (excellent) | 0% (excellent) |
| **Rise Time** | ≈ 15 ms (fast) | ≈ 15 ms (fast) | ≈ 10 ms (fastest) |
| **Overshoot** | None | 5–8% (moderate) | < 2% (excellent) |
| **Settling Time** | Never reaches | 150–200 ms | 30–50 ms (best) |
| **Stability** | Stable (offset) | Stable (slight OS) | Stable (well-damped) |
| **Implementation Complexity** | Very simple | Moderate | Moderate+ |
| **Overall Rating** | Poor | Good | Excellent |

Table (III- 5): performance comparison.

### III.4.6.7 Effect of feedforward and anti-windup

When the motor accelerates from a standstill to 6000 rpm, the controller immediately raises the PWM output to its maximum value of 255 and maintains it at this level for a considerable duration, ranging from a few milliseconds, depending on the motor's inertia. Without protection against error accumulation during this period, the integrator continues to increase unchecked, as the error remains significant and persistent. Consequently, the integrator reaches an extremely high level even before the motor reaches its setpoint. When the PWM output finally

stabilizes, this accumulated integrator pushes the speed well beyond the target value, resulting in significant and problematic overshoot. To prevent this, the controller locks the integrator at each output saturation point, a factory-installed direct locking technique that prevents error accumulation before it even begins, thus keeping the motor close to its setpoint with minimal or even no overshoot.

## III.5 Conclusion

In this final chapter, we find that experimental results confirm with remarkable accuracy the validity of theoretical predictions. The P controller is the simplest and most stable, but it cannot get rid of the steady state error because it lacks integration. The PI controller eliminates bias, but causes a slight steady-state overflow, which is effectively addressed using conditional summation logic. To achieve the best accuracy, the PID controller uses a precisely calibrated differential limit that mitigates excess without amplifying encryption noise.

The three programs share a forward feed gain (FF_GAIN = 0.0422), giving the necessary pulse width measurement offset in the open loop to zero. This is the perfect design for DC motor controllers.

# General Conclusion

This research presents a theoretical study of motor definitions, components, and mathematical equations, along with a comprehensive process for controlling the speed of a DC motor using three classical feedback controllers P, PI, and PID configured using the Ziegler-Nichols method and implemented on an Arduino Uno platform.

experimental step response curves confirmed the accuracy of the theoretical predictions with high precision. The P-controller provides a fast response but leaves a constant 17% offset in steady state an inherent structural limitation. The PI-controller eliminates the offset through integration but causes an average overshoot of 5–8% and a steady-state time of approximately 200 ms. The PID-controller combines these three factors to achieve the best overall performance: the fastest uptime (around 10 ms), near-zero overshoot (less than 2%), zero steady-state error, and a steady-state time of (around 40 ms).

Two additional design features-lead-through compensation and anti-saturation logic proved crucially to achieve a clean and well-damped response on the integrated platform. Lead-through compensation significantly reduces rise time by providing an immediate control signal proportional to the setpoint, allowing the controller to act before the error has time to grow. The anti-saturation mechanism, meanwhile, prevents the integrator from accumulating excessive values during periods of output saturation, which would lead to unacceptable overshoots once the system recovers from saturation after a large transient change.

In practice, this work demonstrates the feasibility of achieving complex closed-loop control on a low-cost microcontroller with modest computational resources, provided the control algorithm is carefully designed and numerically implemented. The 10 Hz sampling rate of the Arduino Uno board is sufficient to display the electromechanical range of the RF-370126000 motor.

Future research could explore adaptive PID tuning to handle load variations, cascade control with an internal current loop for faster turbulence rejection, or model predictive control (MPC) to achieve constrained optimum performance, directly building upon the foundation laid in this research.

# Bibliography

[1] Douis Boubaker et Gadi Nadjib. Commande de vitesse de moteur à courant continu par réseau de neurones artificiel avec une carte Arduino. Master's thesis in electrical control. Echahid Hamma Lakhdar University of El Oued. 2018/2019.

[2] Zoghmar Mahieddine, Habchi Aboubakar Seddik. Étude Comparative Entre Deux Régulateurs PID et FLC Appliqués à la Machine à Courant Continu. Master's thesis at the University of Oum El Bouaghi. 2011-2012.

[3] Salman Jasim Hammoodi, Kareem Sayegh Flayyih, Ahmed Refaat Hamad. Design and implementation speed control system of DC Motor. International Journal of Power Electronics and Drive System (IJPEDS). March 1st, 2020. Vol. 11. (ISSN: 2088-8694). pp. 127~134.

[4] A.E. Fitzgerald, Charles Kingsley Jr, Stephen D Umans. Electric Machinery. sixth edition. Book Publishing House (McGraw-Hill). NY United States, 2003.

[5] Austin Hughes, Bill Drury. Electric Motors and Drives Fundamentals, Types and Applications. third edition. Book Publishing House (Newnes) Oxford, United Kingdom, 2006.

[6] Théodore Wildi, Gilbert Sybille. ÉLECTROTECHNIQUE. 4th edition. Book Publishing House (Brussels), Belgium, 2005.

[7] Dr Nassima BERGOUZ. LA CONVERSIOND'ÉNERGIE ÉLECTROMÉCANIQUE. University publication, energy conversion 2nd year LMD ST. University of Batna 2. 2018/2019.

[8] Claude CHEVASSU. Machines Électriques. O1MM, 2nd year. University publication National Maritime School (ENSM). France. Jully 20th, 2012.

[9] A. S. Al-Salloum. Modeling and simulation of direct current motor drives for industrial applications, Ph.D. dissertation. Department of Electrical Engineering. University of Newcastle upon Tyne, UK, 2012.

[10] Luca Zaccarian. DC motors: dynamic model and control techniques. University publication, University of Rome Tor Vergata Rome. January 2005. pp 1~22.

[11] DRID Saïd. Association machines-convertisseurs. Electrical Machines (S2), University publication Department of Electrical Engineering. University of Batna 2. 2019/2020.

[12] Garing, Christian. Physique Tout-en-un MPSI-PCSI-PTSI Machines électromagnétiques. Book Publishing House (Dunod) Paris, France 2019.

[13] Draidi Abdallah. La machine à courant continu. University publication. Institute of Applied Science and Technology of the University Frères Mentouri Constantine 1. September 30th ,2018.

[14] Chapman, S. J. Electric Machinery Fundamentals. Fifth edition. Book Publishing House (McGraw-Hill). NY United States, 2012.

[15] Khalid Boudane. Velocity Control of a DC Motor Using PID and CDM Method Based on MATLAB/Simulink and Arduino. Master's thesis at the University of Hassiba Benbouali. Chlef, 2021.

[16] Nazanin Afrasiabi and Mohammadreza Hairi Yazdi. DC Motor Control Using Chopper. Global Journal of Science Engineering and Technology. 2013, (ISSN: 2322-2441), Issue 8, pp. 67~73.

[17] Miklós Kuczmann. Review of DC Motor Modelling and Linear Control: Theory with Laboratory Tests. Electronics. June 2024, vol. 13. (ISSN: 2079-9292) article 2225, pp 1~38

[18] D. P. Kothari and I. J. Nagrath. Electric Machines. Fourth edition. Book Publishing House (Tata McGraw-Hill). New Delhi, India, 1990.

[19] Dr. SEGHIOUR Abdellatif. Commande électrique. University publication. Electrical engineering. Graduate School of Electrical and Energy Engineering of Oran.

[20] Rui Chen. A Comprehensive Analysis of PID Control Applications in Automation Systems: Current Trends and Future Directions. Highlights in Science, Engineering and Technology. (ISSN: 3080-1761), 2024, volume 97, pp 126~132.

[21] Marwan J. Hussein, Omar Talib Khazraji, Ahmed M. Almawla. Simulation and Experimental Evaluation of DC Motor Control Strategies Using MATLAB and Arduino Mega. Journal Européen des Systèmes Automatisés. January 2025. Vol. 58, No. 1, (ISSN : 2116-7087), pp. 55-64.

[22] Katsuhiko Ogata. Modern Control Engineering. 5th edition. Book Publishing House (Pearson). New Jersey, United States, 2010.

[23] Flaus, Jean-Marie. La régulation industrielle. Book Publishing House (Hermès), Paris. 1994.

[24] datasheet software-ide. " www.docs.arduino.cc"

[25] datasheet Arduino Board Uno" www.arduino.cc"

[26] datasheet L298N motor driver (H-Bridge Controller)" www.st.com"

[27] BELKACEM Bechoua. Speed and position control of DC motor by Arduino. Master's thesis in electrical machines. Ouargla: Kasdi Merbah University. 2022/2023.

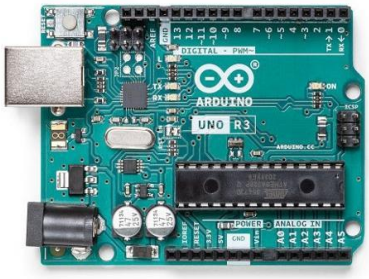
[28] Maung Myo, Latt Maung, New Chaw. DC Motor Angular Position Control Using PID. International Journal of Scientific and Research Publications (IJSRP). November 2018, (ISSN: 2250-3153), vol 8, pp. 149~155.

# Annexes A: Electronics Components

## 1. Microcontroller (Arduino Uno) datasheet

Arduino® UNO R3

Product Reference Manual
SKU: A000066



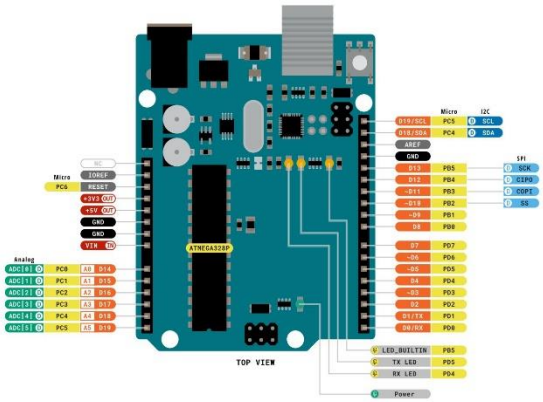
### Description

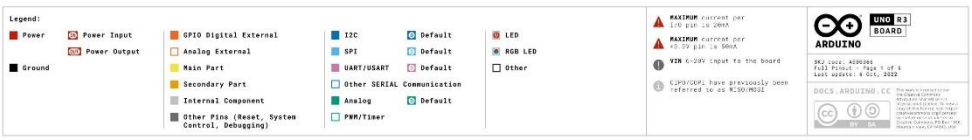
The Arduino® UNO R3 is the perfect board to get familiar with electronics and coding. This versatile development board is equipped with the well-known ATmega328P and the ATMega 16U2 Processor.

This board will give you a great first experience within the world of Arduino.

### Target areas:

Maker, introduction, industries



## 2. Dc Motor datasheet

**PG-25M370 DC GEAR MOTOR Series** SGMADA®

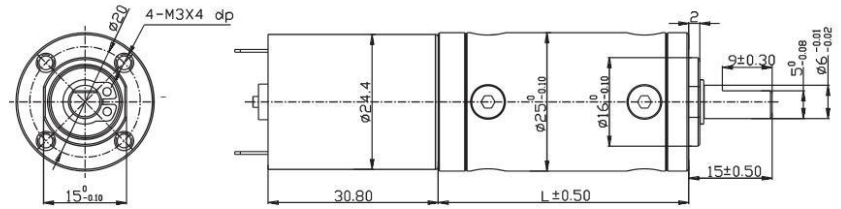
**Main voltage:**3VDC、6VDC、9VDC、12VDC、24VDC、

**Typical applications:** Peristaltic pump,Micro CNC equipment,ATM bank automatic system,Card conveyors,Auto shutter,Binding machine,Mimeograph,Office equipment,Household appliances,Automatic actuator.

**Weight:**120~160g/pcs(approx)

**Packing details:**CTN size:37X27.5XH21cm 100pcs/CTN G.W.17Kgs

**APPEARANCE SIZE**




**Gearbox data:**

| Number of stages | 1 stages reduction | 2 stages reduction | 3 stages reduction | 4stages reduction | 5stages reduction |
|---|---|---|---|---|---|
| Reduction ratio | 4，4.75 | 16，19，22.5 | 64，76，90，107， | 256，304，361，428，509， | 1024，1216，1444，1715，2036，2418， |
| Gearbox length "L" mm | 30 | 35.1 | 40.2 | 45.3 | 50.4 |
| Max. Running torque | 2.0Kgf • cm | 3.0Kgf • cm | 4.0Kgf • cm | 6.0Kgf • cm | 10Kgf • cm |
| Max. Gear breaking torque | 6.0Kgf • cm | 9.0Kgf • cm | 12Kgf • cm | 18Kgf • cm | 30Kgf • cm |
| Max.Gearing efficiency | 90% | 81% | 73% | 65% | 59% |

**Motor data:**

| Motor name | Rated Volt. V | No load | | Load torque | | | | Stall torque | |
|---|---|---|---|---|---|---|---|---|---|
| | | Current | Speed | Current | Speed | Torque | Output power | Torque | Current |
| | | mA | r/min | mA | r/min | gf • cm | W | gf • cm | mA |
| RF-370063000 | 6 | ≤15 | 3000 | ≤100 | 2200 | 15 | 0.33 | 60 | 600 |
| RF-370064500 | 6 | ≤25 | 4500 | ≤270 | 3300 | 24 | 0.8 | 90 | 1200 |
| RF-370066000 | 6 | ≤40 | 6000 | ≤500 | 4500 | 30 | 1.4 | 120 | 1900 |
| RF-370123000 | 12 | ≤15 | 3000 | ≤55 | 2200 | 15 | 0.33 | 60 | 280 |
| RF-370126000 | 12 | ≤20 | 6000 | ≤250 | 4500 | 30 | 1.4 | 120 | 1200 |
| RF-370246000 | 24 | ≤30 | 6000 | ≤130 | 4500 | 30 | 1.4 | 120 | 550 |

1、This table lists some motors parameters,others please refer to specific parameters of Page 111.
2、After connecting motor and gearbox which isnamed gearmotor the output torque:motor torque X reduction ratio X gearing efficiency;output speed:motor speed /reduction ratio.

**NOTE:**

1、Gearmotor named methods: e.g. PG25M370123000-90K Motor please refer to the motor data RF-370123000.Gearbox please refer to gearbox data reduction ratio 90.Related to gearmotor output speed and torque please refer to motor data.
2、Motor can be installed with magnetic encorder.
3、Standard output shaft after reducing:Φ6.0mm.other sizes of the output shaft can make as client request.
4、Chart only for reference,products shall prevail the entity.

## 3. Encoder datasheet

Red: Motor Power Supply Positive Pole + (switching can control forward and reverse rotation direction of the motor)

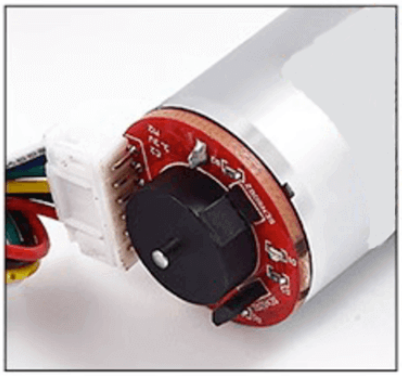
Black: Encoder Power Supply Negative Pole - (positive and negative poles 3.3-5V)

Green: Signal Feedback (11 signals when the motor take a turn)

Yellow: Signal Feedback (11 signals when the motor take a turn)

Blue: Encoder Power Supply Positive Pole + (positive and negative poles 3.3-5V)

White: Motor Power Supply Negative Pole - (switching can control forward and reverse rotation direction of the motor)



ENCODERS CONFLG

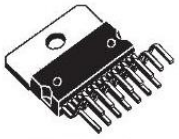
| | | | |
|---|---|---|---|
| Type | AB Dual Phase Incremental Magnetic Hall Encoder | | |
| Line Speed | Basic pulse 11PPRx gear reduction ratio | | |
| Basic Function | Built-in pull-up shaping resistor, directly connected to the microcontroller | | |
| Interface Type | PH2.0 (standard connecting cable) | Response Frequency | 100KHz |
| Power Supply Voltage | DC3.3V / DC5.0V | Output Signal Type | Square wave AB phase |
| Basic Pulse Number | 11PPR | Magnet Ring Trigger Class Quantity | 22 poles (11 pairs of poles) |

## 4. L298N Driver Motor datasheet

# L298

## DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15

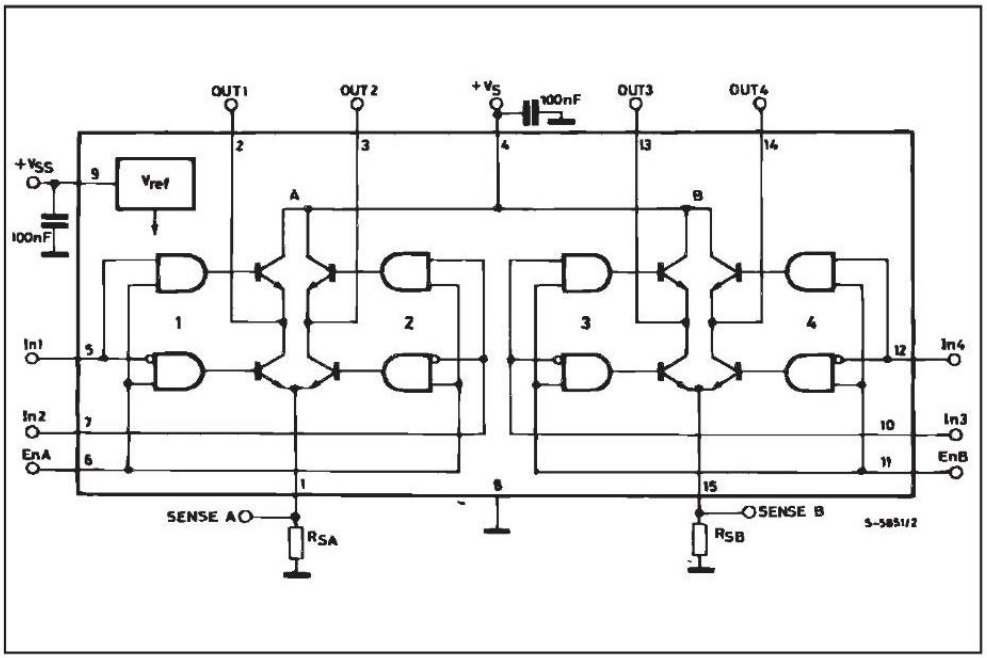
PowerSO20

**ORDERING NUMBERS :** L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

### DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

### BLOCK DIAGRAM



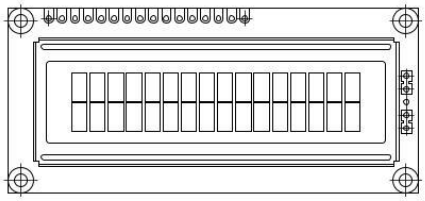
# 5. LCD Screen datasheet

VISHAY
www.vishay.com

## LCD-016N002B-CFH-ET

Vishay

## 16 x 2 Character LCD



### FEATURES

- Type: Character
- Display format: 16 x 2 characters
- Built-in controller: ST 7066 (or equivalent)
- Duty cycle: 1/16
- 5 x 8 dots includes cursor
- + 5 V power supply
- LED can be driven by pin 1, pin 2, or A and K
- N.V. optional for + 3 V power supply
- Optional: Smaller character size (2.95 mm x 4.35 mm)
- Material categorization: For definitions of compliance please see www.vishay.com/doc?99912

Pb-free
RoHS COMPLIANT

### MECHANICAL DATA

| ITEM | STANDARD VALUE | UNIT |
|---|---|---|
| Module Dimension | 80.0 x 36.0 x 13.2 (max.) | mm |
| Viewing Area | 66.0 x 16.0 | |
| Dot Size | 0.55 x 0.65 | |
| Dot Pitch | 0.60 x 0.70 | |
| Mounting Hole | 75.0 x 31.0 | |
| Character Size | 2.95 x 5.55 | |

### ABSOLUTE MAXIMUM RATINGS

| ITEM | SYMBOL | STANDARD VALUE | | | UNIT |
|---|---|---|---|---|---|
| | | MIN. | TYP. | MAX. | |
| Power Supply | $V_{DD}$ to $V_{SS}$ | - 0.3 | - | 13 | V |
| Input Voltage | $V_I$ | $V_{SS}$ | - | $V_{DD}$ | |

**Note**

- $V_{SS}$ = 0 V, $V_{DD}$ = 5.0 V

### ELECTRICAL CHARACTERISTICS

| ITEM | SYMBOL | CONDITION | STANDARD VALUE | | | UNIT |
|---|---|---|---|---|---|---|
| | | | MIN. | TYP. | MAX. | |
| Input Voltage | $V_{DD}$ | $V_{DD}$ = + 5 V | 4.5 | 5.0 | 5.5 | V |
| Supply Current | $I_{DD}$ | $V_{DD}$ = + 5 V | 1.0 | 1.2 | 1.5 | mA |
| Recommended LC Driving Voltage for Normal Temperature Version Module | $V_{DD}$ to $V_0$ | - 20 °C | - | - | 5.2 | V |
| | | 0 °C | - | - | - | |
| | | 25 °C | - | 3.7 | - | |
| | | 50 °C | - | - | - | |
| | | 70 °C | 3.1 | - | - | |
| LED Forward Voltage | $V_F$ | 25 °C | - | 4.2 | 4.6 | V |
| LED Forward Current - Array | $I_F$ | 25 °C | - | 100 | - | mA |
| LED Forward Current - Edge | | | - | 20 | 40 | |
| EL Power Supply Current | $I_{EL}$ | $V_{EL}$ = 110 $V_{AC}$, 400 Hz | - | - | 5.0 | mA |

### DISPLAY CHARACTER ADDRESS CODE

| Display Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DD RAM Address | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
| DD RAM Address | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F |

# Annexes B:

## Arduino Code

```
/*
 ============================================================
  CONTRÔLE PID (Proportionnel + Intégral + Dérivé) - MOTEUR DC 12V
  Pilote    : L298N
  Affichage : LCD 16x2 (mode 4 bits)
  Consigne  : Potentiomètre (0 → 6000 RPM)
  Encodeur  : Hall sur INT0 (D2) — 10 impulsions/tour
 ============================================================
  BRANCHEMENTS :
    LCD (mode 4 bits) :
      RS→D9  EN→D10  D4→D4  D5→D5  D6→D7  D7→D8
    L298N :
      ENA(PWM) → D6 | IN1 → D12 | IN2 → D13
    Encodeur Hall : Signal → D2 | VCC → 5V | GND → GND
    Potentiomètre : Curseur → A0
 ============================================================
*/

#include <LiquidCrystal.h>
// ── LCD ──────────────────────────────────────────────────
LiquidCrystal lcd(9, 10, 4, 5, 7, 8);

// ── Broches L298N ────────────────────────────────────────
#define PIN_ENA   6
#define PIN_IN1  12
#define PIN_IN2  13

// ── Encodeur Hall ────────────────────────────────────────
#define ENC_A     2
#define PPR      10

// ── Potentiomètre ────────────────────────────────────────
#define PIN_POT  A0

// ── Limites PWM ──────────────────────────────────────────
#define PWM_MIN  30
#define PWM_MAX 255

// ── Alimentation ─────────────────────────────────────────
#define SUPPLY_V  12.0f

// ── Paramètres moteur ────────────────────────────────────
const int          RPM_MAX    = 6000;
const unsigned long SAMPLE_MS =  100;

// ── Gains PID ────────────────────────────────────────────

const float Kp = 1.099;
const float Ki = 3.5513;
const float Kd = 0.0009;

// ── Feedforward ──────────────────────────────────────────
#define FF_GAIN  0.0422f

// ── Variables encodeur ───────────────────────────────────
volatile long pulseCount = 0;

// ── Variables de contrôle ────────────────────────────────
float setpoint_rpm = 0;
float speed_rpm    = 0;
float error        = 0;
float prevError    = 0;
```

```
float sumError     = 0;
float output       = 0;

unsigned long lastTime = 0;

// ── Filtre moyenne mobile ──────────────────────────
#define FILTER_SIZE 5
float rpmBuffer[FILTER_SIZE] = {0};
int   rpmIndex = 0;

// ============================================================
void encoderISR() {
  pulseCount++;
}

float filterRPM(float newRPM) {
  rpmBuffer[rpmIndex] = newRPM;
  rpmIndex = (rpmIndex + 1) % FILTER_SIZE;
  float sum = 0;
  for (int i = 0; i < FILTER_SIZE; i++) sum += rpmBuffer[i];
  return sum / FILTER_SIZE;
}

float pulsesToRPM(long p) {
  return (float)p * 60000.0f / ((float)PPR * (float)SAMPLE_MS);
}

// ============================================================
//  Calcul PID + Feedforward avec anti-windup
// ============================================================
float calculatePID(float setpoint, float measured) {
  error = setpoint - measured;

  float feedforward = FF_GAIN * setpoint;
  float pid_P       = Kp * error;
  float pid_D       = Kd * (error - prevError);

  float provisional = feedforward + pid_P + (Ki * sumError) + pid_D;
  bool saturated    = (provisional >= (float)PWM_MAX && error > 0) ||
                      (provisional <= 0.0f           && error < 0);

  if (!saturated) {
    sumError += error;
    sumError  = constrain(sumError, -1000.0f, 1000.0f);
  }

  // ── Réinitialiser l'intégrateur si consigne = 0 ──────────
  if (setpoint < 1.0f) sumError = 0.0f;  // ← CORRECTION : évite intégrateur résiduel

  float pid = feedforward + pid_P + (Ki * sumError) + pid_D;
  prevError = error;
  return constrain(pid, 0.0f, (float)PWM_MAX);
}

// ============================================================
void driveMotor(float pid) {
  if (pid < 1.0f || setpoint_rpm < 1.0f) {
    digitalWrite(PIN_IN1, LOW);
    digitalWrite(PIN_IN2, LOW);
    analogWrite(PIN_ENA, 0);
    return;
  }
  int pwm = (int)pid;
  if (pwm < PWM_MIN) pwm = PWM_MIN;
  pwm = constrain(pwm, 0, PWM_MAX);
  digitalWrite(PIN_IN1, HIGH);
```

```
  digitalWrite(PIN_IN2, LOW);
  analogWrite(PIN_ENA, pwm);
}

float pwmToVoltage(int pwm) {
  return ((float)pwm / 255.0f) * SUPPLY_V;
}

// ============================================================
//  Affichage LCD
// ============================================================
void updateLCD(float csg, float spd, float volt) {
  lcd.setCursor(0, 0);
  lcd.print("C:");
  int c = (int)csg;
  if      (c <   10) lcd.print("   ");
  else if (c <  100) lcd.print("  ");
  else if (c < 1000) lcd.print(" ");
  lcd.print(c);

  lcd.setCursor(8, 0);
  lcd.print("Vit:");
  int v = (int)spd;
  if      (v <   10) lcd.print("   ");
  else if (v <  100) lcd.print("  ");
  else if (v < 1000) lcd.print(" ");
  lcd.print(v);

  lcd.setCursor(0, 1);
  lcd.print("E:");
  int e = (int)error;
  if (e >= 0) lcd.print("+");
  if      (abs(e) <   10) { lcd.print(e); lcd.print("   "); }
  else if (abs(e) <  100) { lcd.print(e); lcd.print("  ");  }
  else if (abs(e) < 1000) { lcd.print(e); lcd.print(" ");   }
  else                    { lcd.print(e);                   }

  lcd.setCursor(9, 1);
  lcd.print("V:");
  if (volt < 10.0f) {
    int vi = (int)volt;
    int vf = (int)((volt - vi) * 10 + 0.5f);
    lcd.print(vi); lcd.print("."); lcd.print(vf);
  } else {
    lcd.print(volt, 1);
  }
  lcd.print("V ");
}

// ============================================================
//  setup()
// ============================================================
void setup() {
  pinMode(PIN_IN1, OUTPUT);
  pinMode(PIN_IN2, OUTPUT);
  pinMode(PIN_ENA, OUTPUT);
  digitalWrite(PIN_IN1, LOW);
  digitalWrite(PIN_IN2, LOW);
  analogWrite(PIN_ENA, 0);

  pinMode(ENC_A, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(ENC_A), encoderISR, RISING);

  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("  CONTROLE  PID ");
```

```
  lcd.setCursor(0, 1);
  lcd.print("  DC Motor ");
  delay(2000);
  lcd.clear();

  Serial.begin(9600);
  Serial.println("=== PID Controller START ===");

  lastTime = millis();
}

// ============================================================
//  loop()
// ============================================================
void loop() {
  long          pulses = 0;
  unsigned long now    = millis();

  if (now - lastTime >= SAMPLE_MS) {
    lastTime = now;

    // 1. Lire le Setpoint — arrondi à 10 RPM + LIMITÉ à RPM_MAX
    int potVal   = analogRead(PIN_POT);
    int raw      = map(potVal, 0, 1023, 0, RPM_MAX);
    raw          = constrain(raw, 0, RPM_MAX);          // ← CORRECTION : ne dépasse pas
6000
    setpoint_rpm = (float)(((raw + 5) / 10) * 10);
    setpoint_rpm = constrain(setpoint_rpm, 0.0f, (float)RPM_MAX); // ← CORRECTION

    // 2. Lire l'Encodeur
    noInterrupts();
      pulses     = pulseCount;
      pulseCount = 0;
    interrupts();
    float raw_rpm = pulsesToRPM(pulses);
    raw_rpm   = constrain(raw_rpm, 0.0f, (float)RPM_MAX); // ← CORRECTION : vitesse ≤
6000
    speed_rpm = filterRPM(raw_rpm);

    // 3. Calcul PID
    output = calculatePID(setpoint_rpm, speed_rpm);

    // 4. Commander le moteur
    driveMotor(output);

    // 5. PWM → Tension
    int   pwm_actual  = (setpoint_rpm < 1.0f) ? 0 : constrain((int)output, PWM_MIN,
PWM_MAX);
    float volt_actual = (setpoint_rpm < 1.0f) ? 0.0f : pwmToVoltage(pwm_actual);

    // 6. LCD
    updateLCD(setpoint_rpm, speed_rpm, volt_actual);

    // 7. Serial
    Serial.print("Csg:");  Serial.print((int)setpoint_rpm);
    Serial.print(" Vit:"); Serial.print((int)speed_rpm);
    Serial.print(" Err:"); Serial.print((int)error);
    Serial.print(" PID:"); Serial.print((int)output);
    Serial.print(" Volt:"); Serial.println(volt_actual, 2);
  }
}
```

## MATLAB code

```
%  P | PI | PID Controller Comparison
%  Tuning Method : Ziegler-Nichols %
System        : DC Motor 12V (RF-370126000)
clc; clear; close all;
disp('=====================================
==========='); disp('  DC Motor PID Tuner ###
Ziegler-Nichols Method 1  ');
disp('=====================================
===========');
```

### Input Motor Parameters (Dialog Box)

```
prompt = {'R (Ohm):', 'L (H):', 'Ke (V.s/rad):', ...
'Kt (N.m/A):', 'J (kg.m^2):', 'B (N.m.s/rad):'}; answer
= inputdlg(prompt, 'DC Motor Parameters', 1, ...
{'7','0.01','0.01892','0.01892','1e-6','3.915e-6'}); if
isempty(answer), disp('Program ended by user.');
return; end

R  = str2double(answer{1});
L  = str2double(answer{2});
Ke = str2double(answer{3});
Kt = str2double(answer{4});
J  = str2double(answer{5});
B  = str2double(answer{6});
```

### Plant Transfer Function G(s) = omega(s)/V(s)

```
num = Ke;
den = [L*J,  L*B + R*J,  R*B + Kt*Ke]; G   = tf(num, den);
fprintf('=================================================\n');
fprintf('  Plant Transfer Function G(s):\n\n'); fprintf('
%.4f\n', num); fprintf('  G(s) =
########################################################################
###############
\n');

fprintf('         %.2e s^2 + %.2e s + %.2e\n\n',
den(1), den(2),  den(3));
```

### Ziegler-Nichols Critical Gain & Period

```
[Gm, ~, Wcg, ~] = margin(G); if
isinf(Gm)     % Always-stable system ###
estimate Ku analytically
    Ku = 10 * (R*B + Kt*Ke) / Kt;     Wu =
sqrt((R*B + Kt*Ke) / (L*J));     fprintf('
[Warning] System is always stable ### Ku estimated.
\n'); else     Ku = Gm;    % Ultimate gain     Wu =
Wcg;   % Ultimate frequency [rad/s] end Tu = 2 * pi
/ Wu;   % Ultimate period [s]

fprintf('  Ku (Ultimate Gain)   = %.4f\n', Ku);
fprintf('  Tu (Ultimate Period) = %.6f s\n', Tu);
```

```
fprintf('=========================================
========\n\n');
```

### Ziegler-Nichols Tuning Formulas

```
Kp_P  = 0.50 * Ku;
Ki_P  = 0;
Kd_P  = 0;
% ###### PI ######
Kp_PI = 0.45 * Ku;
Ti_PI = Tu / 1.2;
Ki_PI = Kp_PI / Ti_PI;
Kd_PI = 0;
% ###### PID ######
Kp_PID = 0.60 * Ku;
Ti_PID = Tu / 2;
Td_PID = Tu / 8;
Ki_PID = Kp_PID / Ti_PID;
Kd_PID = Kp_PID * Td_PID;
```

### Display Gains

```
  fprintf('       %-4s    |    %10s    |    %10s    |
%10s\n','Type','Kp','Ki','Kd'); fprintf('  -----+--
----------+------------+------------\n'); fprintf('
%-4s | %10.4f | %10.4f | %10.4f\n','P',  Kp_P,  Ki_P,
Kd_P);
fprintf('  %-4s | %10.4f | %10.4f | %10.4f\n','PI',
Kp_PI,  Ki_PI,  Kd_PI);
fprintf('  %-4s | %10.4f | %10.4f | %10.4f\n','PID',
Kp_PID, Ki_PID,  Kd_PID);
```

### Closed-Loop & Performance

```
sys_P   =
feedback(pid(Kp_P,Ki_P,Kd_P)*G,1);
sys_PI  =
feedback(pid(Kp_PI,Ki_PI,Kd_PI)*G,1);
sys_PID = feedback(pid(Kp_PID,
Ki_PID, Kd_PID)*G,1);

ip   =
stepinfo(sy
s_P); ipi
=
stepinfo(sy
s_PI); ipid
=
stepinfo(sy
s_PID);

  fprintf('\n       %-18s    |    %8s    |    %8s    |
%8s\n','Metric','P','PI','PID'); fprintf('  --------
----------+----------+----------+----------\n');
```

```
fprintf('  %-18s | %8.4f | %8.4f | %8.4f\n','Rise Time
(s)',ip.RiseTime,ipi.RiseTime,ipid.RiseTime);
fprintf('  %-18s | %8.2f | %8.2f | %8.2f\n','Overshoot
(%)',ip.Overshoot,ipi.Overshoot,ipid.Overshoot);
fprintf('  %-18s | %8.4f | %8.4f | %8.4f\n','Settling
Time
(s)',ip.SettlingTime,ipi.SettlingTime,ipid.SettlingT
ime); fprintf('  %-18s | %8.2f | %8.2f | %8.2f\n','SS
Error          (%)',(1dcgain(sys_P))*100,          ...
(1dcgain(sys_PI))*100,                             ...
(1dcgain(sys_PID))*100);
```

**Step Response Plot**

```
t_end
=5*max([ip.SettlingTime,ipi.SettlingTime,ipid.Settl
ingTime,0.1]); t = linspace(0, t_end, 1000); [y_P,
tP]  = step(sys_P,   t);
[y_PI,  tPI] =
step(sys_PI,  t);
[y_PID, tPID]=
step(sys_PID, t);

figure('Color','w','Position',[100 100 950 520]);
hold on; plot(tP,y_P,'r-','LineWidth',2.5);
plot(tPI,y_PI,'b--','LineWidth',2.5);
plot(tPID,y_PID,'g-','LineWidth',2.5); grid on;
xlabel('Time(s)','FontSize',12); ylabel('Speed
(normalized)','FontSize', 12); title(' P / PI / PID
DC Motor PID Tuner ### Z-N Method  1', 'FontSize',
13);
legend('P','PI','PID','Reference','Location','southea
st','FontSize',  11); xlim([0 t(end)]);
ylim([-0.1 max([y_P; y_PI; y_PID]) *

1.15]) ;
```

## Certificate of authorization for correction and filing

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي

Université de Ghardaia
Faculté des Sciences
et de la technologie

جامعة غرداية
كلية العلوم والتكنولوجيا
قسم الآلية والكهروميكانيك

شعبة : العلوم والتكنولوجيا
تخصص: طاقات متجدده في الكهروتقني

غرداية في : 2026/06/24

# شهادة ترخيص بالتصحيح والايداع:

انا الاستاذ: بحري أحمد
بصفتي المشرف المسؤول عن تصحيح مذكرة تخرج (ليسانس/ماستر/دكتورا) المعنونة بـ:

**Study and Simulation of DC Motor control**

من انجاز الطالب (الطلبة):

باحماني إبراهيم

مرزوق إبراهيم

التي نوقشت بتاريخ : 2026/06/14

اشهد ان الطلبة قد قاموا بالتعديلات والتصحيحات المطلوبة من طرف لجنة المناقشة وقد تم التحقق من ذلك من طرفنا
وقد استوفت جميع الشروط المطلوبة.

امضاء المسؤول عن التصحيح

مصادقة رئيس القسم

جامعة غرداية
رئيس قسم الآلية والكهروميكانيك
حسزاوي محمد
قسم الآلية والكهروميكانيك-1
كلية العلوم و التكنولوجيا