الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Projet de Fin d'Études

présenté par

BOUMESHED Mohamed amine

pour l'obtention du diplôme de master en Électronique option signaux en ingénierie des systèmes informatique industrielle

Thème

Réalisation d'un PID à base ARDUINO

Proposé par : Mr. BENSLAMA Zoubir

Année Universitaire 2013-2014

Dédicace

Je dédie ce modeste travail

A mes chers parents.

A mes sœurs, mes oncles, mes tantes, mes cousins et tous les membres de la famille ***BOUMESHED****, chacun par son nom.*

La famille MESKINE.

A tous mes amis et surtout : LARBI, ABDOU, HAMZA, SOFIANE et TAHER.

A tous mes amis de la promotion

Master SISII 2013-2014.

A tous ceux que j'aime et qui m'aiment

MOHAMED AMINE

Remerciements

Tout d'abord, je tiens à remercier Allah, le clément et le miséricordieux de m'avoir donné la force et le courage de mener à bien ce modeste travail.

Je remercie mes parents pour leur sacrifice, leurs encouragements, durant toute ma scolarité et soucieux de mon avenir.

Je tiens exprimer mes vifs remerciements à mon encadreur ***Benslama zoubir,*** *pour sa confiance, ces orientations et ces conseils.*

Je remercie tous mes collègues, et toutes les personnes, qui ont contribué de près ou de loin, directement ou indirectement à ce travail.

Je remercie nos enseignants de ***la Faculté de Technologie,*** *pour leurs efforts voués à nous transmettre le savoir, en particulier le* ***département d'Electronique.***

Que tous les membres du jury trouvent ici l'expression de mon profond respect pour avoir pris la peine d'examiner le manuscrit.

ملخص

يهدف هذا المشروع لنهاية الدراسة الى تنفيذ التحكم من نوع PID بحيث سيتم تحقيق هذا التحكم بناءا على بطاقة أردوينو« Carte Arduino » من أجل مراقبة سرعة المحرك ذات التيار المستمر.
سيتم ارسال تعليمات و معايير PID بفضل واجهة بيانية على "MATLAB" تُظهر كذلك رسما بيانيا لتطور السرعة في الزمن الحقيقي.

Résumé

Ce projet de fin d'étude a pour but d'implémenter un asservissement de type PID. Ce régulateur sera réalisé à base d'une carte Arduino pour le contrôle de la vitesse d'un moteur à courant continu.La transmission des consignes ainsi que les paramètres du PID se fera grâce à une interface graphique sur MATLAB, qui affichera aussi le graphe de l'évolution de la vitesse en temps réel.

Abstract

This Final project assignment aims to implement a PID servo. This controller will be implemented based on an Arduino board to control the speed of a DC motor .the setpoints and PID parameters will be done through a graphical interface on MATLAB, which also display the graph of the evolution of the speed on real-time.

Table des matières

Liste des figures

Introduction générale

La technologie moderne à permis le développement des sciences tout en imposant l'exploration de domaines théoriques de plus en plus complexes. Parmi ces sciences en pleine expansion et intégrant rapidement l'apport des technologies moderne, on compte l'automatique. Le substantif « automatique » à été utilisé pour la première fois en 1914 dans un article « essai sur l'automatique » publié dans une revue scientifique [13].

L'automatique est la science qui traite de l'analyse et de la commande des systèmes dynamiques évoluant avec le temps. En d'autres mots, de l'automatisation de taches par des machines fonctionnant sans intervention humain [21]. Le système de la commande peut fonctionner en boucle ouverte à partir d'un signal d'entrée. Cependant, c'est uniquement en boucle fermée qu'on est capable de stabiliser, d'améliorer les performances et de rejeter les perturbations externes des systèmes dynamiques. La loi de commande est générée par un système de commande qu'on appelle correcteur ou régulateur.

Plusieurs méthodes de synthèse d'un régulateur, offrant une complexité et des performances très variable, sont disponibles dans la littérature. Cependant, le régulateur PID reste l'algorithme de commande le plus utilisé dans l'industrie. Sa prédominance incontestée provient, outre de la simplicité de sa structure et le nombre restreint de paramètres à ajuster, des performances qu'il peut offrir aux systèmes en boucle fermée, satisfaisant très souvent les cahiers des charges, si ses paramètres sont choisis judicieusement.

L'implémentation d'un algorithme de commande PID sur un ordinateur ou un microcontrôleur à beaucoup d'avantages par rapport à une réalisation analogique. Ces avantages se traduisent par :

- coût faible.
- précision élevée.
- insensibilité au bruit.
- facilité d'implémentation.
- souplesse par rapport aux modifications.

L'objectif de notre projet de fin d'étude, est de réaliser un régulateur PID à base d'une carte Arduino, pour contrôler et commander un moteur à courant continu.

Le travail présenté dans ce mémoire est composé de quatre chapitres :

- Le premier chapitre est consacré aux moteurs à courant continu, à leurs commandes ainsi qu'à l'étude des codeurs optiques.
- Dans le deuxième chapitre, représente une étude sur la régulation et la commande numérique des systèmes.
- Le troisième chapitre concerne la partie électronique et la partie software de notre régulateur PID.
- Le quatrième chapitre est réservé à la réalisation des différents modules et aux tests.
- Dans la dernière partie de ce mémoire, nous présentons la conclusion générale qui récapitule notre travaille.

Conclusion générale

L'objectif qui a été donné pour notre projet, est l'utilisation d'une carte Arduino. Dont le but est de mettre au point la commande en vitesse d'un moteur à courant continu en utilisant un régulateur PID.

Par conséquent nous avons :

- Présenté les caractéristiques d'un moteur à courant continu.
- Etudié la commande de régulation qui se présente par le correcteur PID.
- Bien détaillé les caractéristiques et le fonctionnement de la carte Arduino.
- Réalisé une carte additionnelle pour l'interface Arduino pour contrôler le moteur. Une autre carte de puissance commandée par le signal PWM de la carte Arduino. Aussi une carte d'alimentation stabilisée de 12V pour le moteur.
- Crée une interface graphique sur MATLAB pour commander la vitesse du moteur, et constater l'évolution de cette vitesse en temps réel. Et aussi pour régler les paramètres Kp, Ki et Kd du correcteur PID.

Finalement, ce travail a été une occasion de concrétiser et de mettre en évidence nos compétences acquises au cours de notre formation académique. Ce travail nous à permis d'acquérir une expérience enrichissante dans plusieurs domaines : l'asservissement, les microcontrôleurs, l'électronique générale, les moteurs à courant continue et la programmation.

[1] Françis Milsant, « Electronique de base », tome -2-Edition: Eyrolles, 1983.

[2] R.Bouallaga, H.Benlaoeur, implémentation d'un contrôleur PID pour la régulation de la vitesse d'un robot mobile, thèse de fin d'étude, université de Blida, 2007

[3] Y.Bettou, H.Ahcène, régulation de la vitesse d'un moteur à courant continu par PIC 16F877, thèse de fin d'étude, université saad dahleb de Blida, octobre 2011.

[4] M.Arhoujdam, K.Dahi, mise en œuvre d'une régulation de vitesse d'un moteur à courant continu, thèse de fin d'étude, université Mohamed V souissi, 2012-2013

[5] O.Ait sahed, A.Benachour, implémentation sur carte DSP de la commande PID auto-ajustable d'un moteur à courant continu, thèse de fin d'étude, université saad dahleb de Blida, septembre 2010.

[6] projet de fin d'étude : étude et réalisation d'une alimentation à thyristors pilotée par PIC 16F877A en vue de la commande d'un moteur DC, université saad dahleb de Blida, juillet 2011.

[7] A.Kebaili, M.Izri, réalisation d'une boucle d'asservissement de la vitesse d'un moteur DC utilisant la technique PLL, thèse de fin d'étude, université saad dahleb de Blida, 2011-2012.

[8] J. Pierre Feste, le codeur optique, http://www.lmdindustrie.com/content/guides /achat/ Codeurs_optiques_201003.pdf

[9] codeurs optiques, http://ww2.ac-poitiers.fr/electrotechnique/img/pdf/codeurs optiques _prof.pdf

[10] S. Sekhsokh, K. Oukili, étude d'une boucle de régulation de niveau : implémentation du régulateur et réglage du procédé, thèse de fin d'étude, 2010-2011

[12] A.Bensaadi, S.Abbas, régulation de niveau d'eau dans un réservoir assisté par le logiciel LabVIEW, thèse de fin d'étude, université saad dahleb de Blida, 2010/2012

LISTE DES ABREVIATIONS

FEM	**F**orce **E**lectro**m**otrice
PWM	**P**ulse **W**idth **M**odulation
PID	**P**roportionnel **I**ntégral **D**érivé
PC	**P**ersonal **C**omputer
LCD	**L**iquid **C**rystal **D**isplay
USB	**U**niversal **S**erial **B**us
LED	**L**ight **E**mitting **D**iode
RAM	**R**andom **A**ccess **M**emory
EEPROM	**E**lectrical **E**rasable **P**rogrammable **R**ead **O**nly **M**emory
CPU	**C**entral **P**rocessing **U**nit
TWI	**T**wo **W**ire **I**nterface
I2C	**I**nterface à deux **C**onducteur
MLI	**M**odulation de **L**argeur d'**I**mpulsion
IHM	**I**nterfaces **H**omme **M**achine
GUI	**G**raphical **U**ser **I**nterfaces
MATLAB	**MAT**rix **LAB**oratory
GUIDE	**G**raphical **U**ser **I**nterfaces **D**evelopment **E**nvironment
UART	**U**niversal **A**synchrones **R**eceiver **T**ransmitter
PCB	**P**rinted **C**ircuit **B**oard
EAGLE	**E**asily **A**pplicable **G**raphical **L**ayout **E**ditor

1.1 Introduction

Les moteurs et les générateurs à courant continu furent les premières machines électrique utilisées par les ingénieurs au milieu du 19ième siècle pour produire de la puissance motrice en usine ou en transport.

Le moteur électrique transforme de l'énergie électrique en énergie mécanique et le générateur électrique converti l'énergie mécanique en l'énergie électrique.

Le but de cette partie est d'élaborer une représentation mentale aussi claire et fidèle que possible du moteur à courant continu. Là est l'objet de la commande dans notre réalisation.

Les moteurs à courant continu sont les plus utilisés dans le domaine de faible puissance. En effet, la mise en œuvre de leurs commandes est parfaitement au point [1].

1.2 Constitution d'un moteur à courant continu

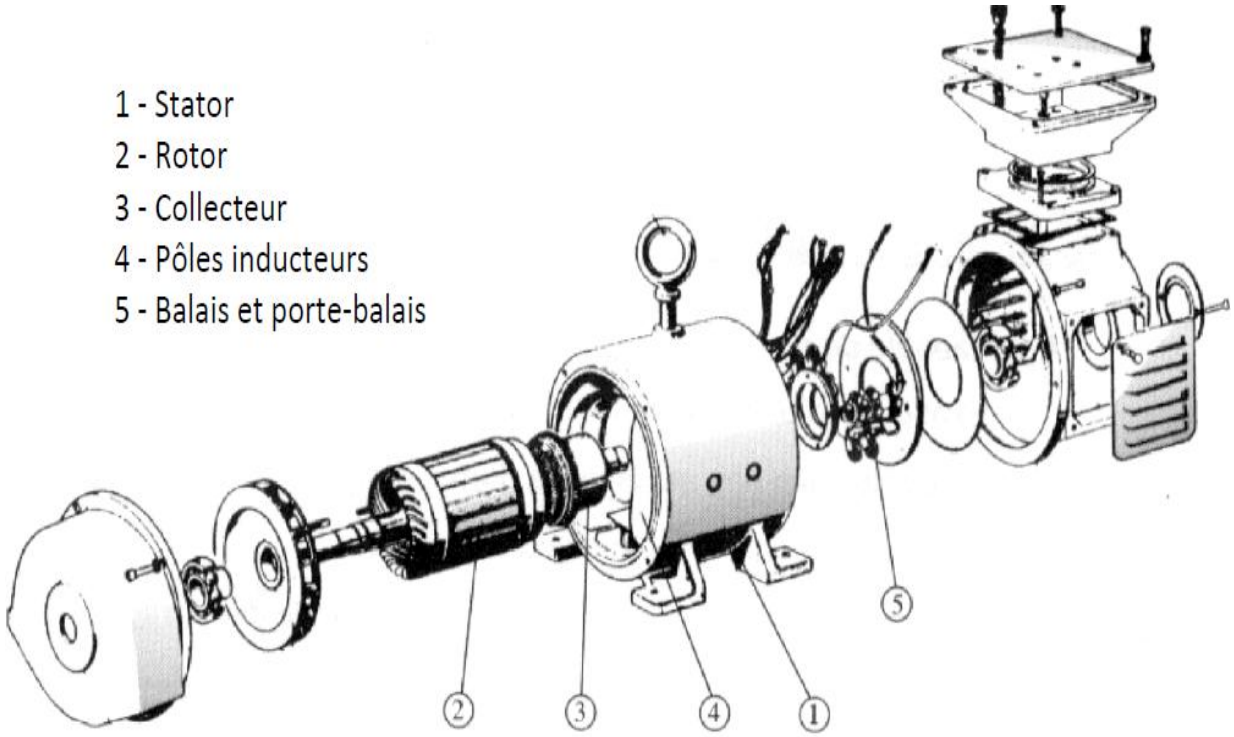


Figure.1.1: schéma d'un moteur à courant continu

1.2.1 Stator ou inducteur

Le stator est constitué de la carcasse du moteur et du circuit magnétique, il génère le champ magnétique nécessaire pour faire tourner le moteur, l'intensité du champ magnétique est proportionnelle au courant dans l'inducteur, le champ d'induction magnétique produit soit par des bobinages parcourus par un courant (électroaimant), soit par des aimants permanents.

Figure.1.2: stator ou inducteur

1.2.2 Rotor ou induit

Le rotor, partie mobile, appelé aussi induit, composé de tôle isolées entre elles et munies d'encoches dans lesquelles sont réparties les conducteur. Parcourus par un courant, ceux-ci créent le champ magnétique dit rotorique.

Figure.1.3: rotor ou induit

1.2.3 Collecteur et balais

Le collecteur est un ensemble de lames de cuivre ou sont reliées les extrémités du bobinage de l'induit. Les balais (ou charbons) sont situés au stator et frottent sur le collecteur en rotation. Le dispositif collecteur /balais permet donc de faire circuler un courant dans l'induit.

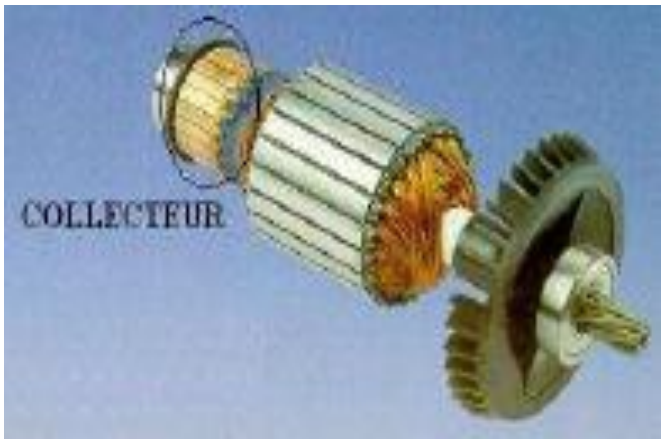


Figure.1.4: collecteur

1.3 Principe de fonctionnement d'un moteur à courant continu

Lorsque l'on place une spire parcourue par un courant (grâce aux balais et au collecteur) dans un champ magnétique, il apparait un couple de force. Ce couple de force crée un couple de rotation qui fait dériver la spire de plus ou moins 90 degrés par rapport au plan vertical, le sens du courant restant inchangé dans la spire, au cours de ce déplacement, le couple de rotation diminue constamment jusqu'à s'annuler après rotation de la bobine de plus ou moins 90 degrés(zone neutre, la spire se trouve à l'horizontale et perpendiculaire aux aimants naturels).

A fin d'obtenir une rotation sans à coup, l'enroulement d'induit doit être constitué d'un nombre élevé de spire similaires. Celles-ci seront réparties de façon régulière sur le pourtour du rotor (induit), de manière à obtenir un couple indépendant de l'angle de rotation. Après le passage de zone neutre, le sens du courant doit être inversé simultanément dans chacune de ces spires.

L'inversion du courant est opérée par l'inverseur ou commutateur (collecteur) qui associé au balai, constitue l'élément assurant la transmission du courant de la partie fixe à la partie tournante du moteur.

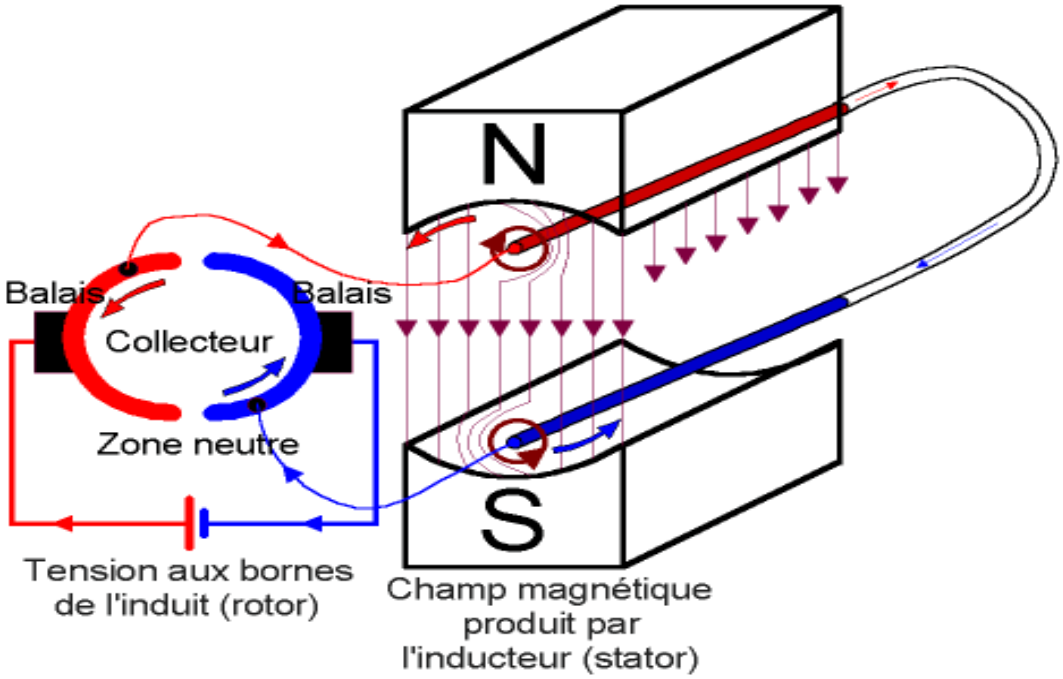


Figure.1.5: principe de fonctionnement d'un moteur DC

1.4 Caractéristiques des moteurs à courant continu

La qualité du moteur (régularité de l'entrainement, couple, vitesse…) est donc directement lié à sa constitution :

- Nombre de pôles,
- Nombre de faisceaux (et donc de lames du collecteur),
- Nombre de spires dans un faisceau,
- Choix des matériaux constituant l'ensemble.

Le moteur se comporte comme une résistance en série avec un générateur de tension (fem : force électromotrice)

$$U = E + R \times I \quad \text{..... (1.1)}$$

U : tension d'alimentation du moteur

E : force électromotrice

R : résistance interne du bobinage

I : courant consommé par le moteur

$$E = Ke \times \boldsymbol{\omega} \quad \text{..... (1.2)}$$

Ke : constante

$\boldsymbol{\omega}$: Fréquence de rotation (rad/s)

La tension fem est proportionnelle à la fréquence de rotation.

$$I = Kc \times C \quad \text{..... (1.3)}$$

Kc : constante

C : couple moteur (m.N)

Le courant consommé par le moteur est directement lié au couple résistant sur l'arbre.

$$E' = \frac{P}{a} nN\emptyset \quad \text{..... (1.4)}$$

E': Force contre électromotrice (f.c.e.m)

$2P$: nombre total des pôles de l'inducteur.

2a : nombre de vois de l'enroulement induit.

n : nombre de brin actifs.

$\emptyset$: Flux utile par pole en weber.

N : vitesse en tr/s.

La vitesse de rotation du moteur est directement liée à la tension d'alimentation, et le sens de rotation dépend de la polarité de l'alimentation du bobinage de l'induit (du rotor) ou du stator lorsque celui-ci est constitué d'électroaimants.

1.5 Classification des moteurs à courant continu

Les moteurs à courant continu sont souvent classes par le type de champ statorique utilisé. Les différents cas possibles sont :

1.5.1 Moteur à excitation indépendante

Ce mode d'excitation nécessite deux sources d'alimentation distinctes. L'alimentation de l'enroulement inducteur est prise sur une source indépendante de la source principale. On change le sens de rotation en permutant les bornes de l'induit ou de l'inducteur [3].

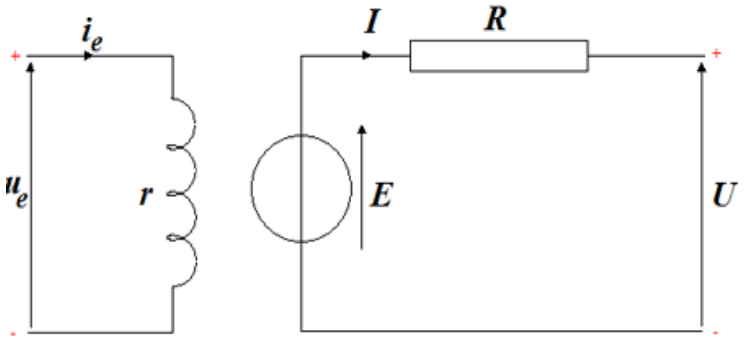


Figure.1.6: moteur à excitation indépendante

Ce moteur à une grande souplesse de commande et une large gamme de vitesse, il est utilisé en milieu industriel associé avec un variateur électrique [4].

1.5.2 Moteur à excitation série

Le circuit d'excitation est placé avec l'induit du moteur. Sa particularité est d'avoir un inducteur qui est traversé par le même courant, l'inducteur possède donc une résistance plus faible que celle des autres types de machines. L'inducteur est en série avec l'induit : une seule source d'alimentation suffit [3].

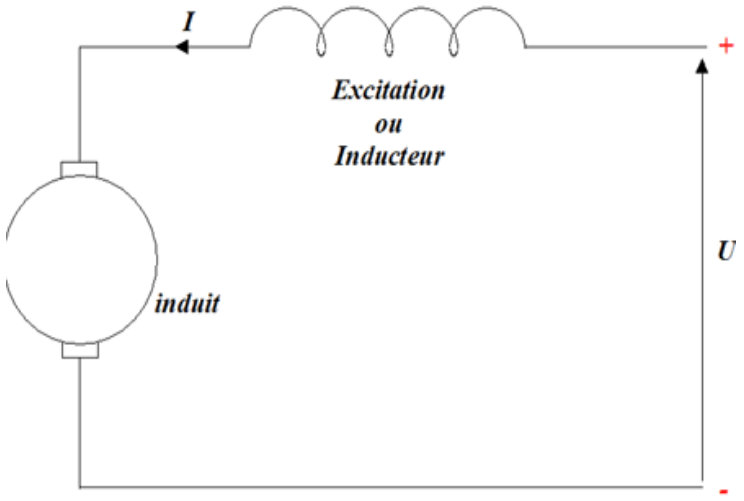


Figure.1.7: moteur à excitation série

Ce moteur à un démarrage fréquent avec un couple élevé, et le couple diminue avec la vitesse [4].

1.5.3 Moteur à excitation shunt

Dans le moteur shunt, le stator est câblé en parallèle avec le rotor, la tension aux bornes du rotor est la même que celle aux bornes du stator.

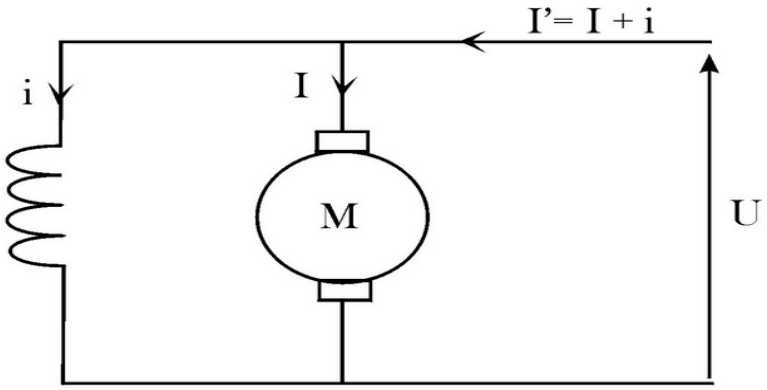


Figure.1.8: moteur à excitation shunt

Ce type de moteur possède une vitesse sensiblement constante quelque soit la charge, ne s'emballe pas à vide et s'emballe si rupture d'inducteur.

1.5.4 Moteur à excitation compound

Dans le moteur compound une partie du stator est raccordé en série avec le rotor et une autre est de type parallèle ou shunt.

Ce moteur réunit les avantages des deux types de moteur : le fort couple à basse vitesse du moteur série et l'absence d'emballement du moteur shunt.

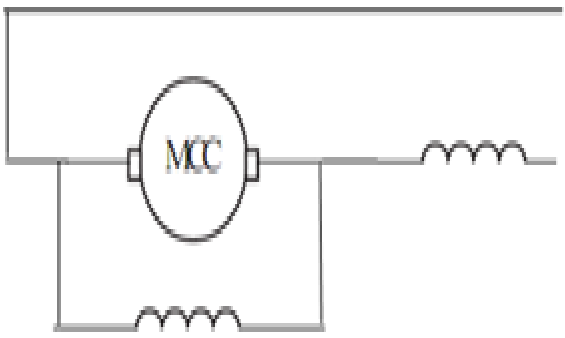


Figure.1.9: moteur à excitation compound

1.6 Différents types de moteur à courant continu

Il existe plusieurs types de moteur à courant continu, suivant leurs constitutions, montages et caractéristique. Nous citons les deux types les plus utilisé :

1.6.1 Moteur avec balais

Ce type de moteurs est largement utilisé dans la pratique à cause de son prix économique et la diversité de ces modèles présents sur le marché, mais il est de plus en plus remplacé par le moteur sans balais.

Les moteurs à aimant permanents sont les plus utilisés pour les applications à faible puissance. Car il est plus économique d'utiliser des aimants permanents qu'un bobinage (pour la création du champ magnétique du stator). Leur inconvénient, est le fait qu'au passage du temps, ils commencent à perdre leurs propriétés.

La relation vitesse/tension de ce type de moteur est linéaire, un changement en tension engendre une de la tension, est l'une des propriétés importantes des moteurs avec balais. Pour la relation du couple avec le courant, elle est aussi linéaire [5].

1.6.2 Moteur sans balais

Il existe également des moteurs à courant continu sans balais, ils utilisent des aimants permanents pour obtenir le champ magnétique, mais à la place de la rotation de l'induit provoqué par le champ magnétique de l'aimant, l'aimant permanent tourne dans la bobine fixe. Avec un moteur à courant continu classique il est nécessaire d'utiliser un commutateur pour inverser le courant dans les bobines à chaque demi-rotation de manière à pour suivre la rotation de l'induit dans le même sens, avec les moteurs sans balais à aimant permanent un circuit électronique permet d'inverser le courant. Le moteur peut être démarré et arrêté en contrôlant le courant dans la bobine fixe.

L'inversion du sens de rotation du moteur est plus complexe, car inverser le courant n'est pas si simple, en raison du circuit électronique qui met en œuvre le commutateur. Une méthode consiste à incorporer des capteurs pour détecter l'emplacement des pôles nord et sud, ces capteurs peuvent ensuit provoquer la commutation du courant dans les bobines au bon moment à fin d'inverser les forces appliquées à l'aimant. La vitesse de rotation peut être commandée par PWM (Pulse Width Modulation).

Figure.1.10: moteur sans balais

1.7 Réglage de la vitesse d'un moteur à courant continu

1.7.1 Réglage rhéostatique

Il est possible de réduire la vitesse en augmentant la résistance de l'induit avec un rhéostat branchée en série avec l'induit tout en fixant la tension et le flux à leurs valeurs nominales [4].

Ce réglage est mauvais sur le plan technique, et aussi sur le plan économique car plus la chut de vitesse réclamée est grande plus la consommation d'énergie dans le rhéostat augmente. Dans la pratique on utilise ce procédé de réglage pour le démarrage et le freinage du moteur [6].

1.7.2 Réglage par le flux

Lors du démarrage on applique le flux maximal, pour obtenir un couple maximal, des que la vitesse nominale sera atteinte il sera possible de réduire le flux par un rhéostat de champ, insérer dans le circuit de l'inducteur.

Ce réglage est bon du point de vue technique, et aussi du point de vue économique, car la puissance dissipé dans l'inducteur étant très faible par rapport à la puissance absorbée, avec ce procédé on ne peut qu'augmenter la vitesse du moteur par rapport à sa vitesse nominale [6].

1.7.3 Réglage par tension

On applique une tension de valeur moyenne variable entre zéro et une valeur maximale tout en réglant le flux à sa valeur maximale. Ce procédé de réglage de vitesse est excellent du point de vue économique car aucune énergie n'est dissipe dans les rhéostats [4].

Quand on veut régler la vitesse d'un moteur à courant continu, il suffit de faire varier la tension à ses bornes, et pour maintenir une vitesse invariable pour une tension de consigne donnée, il faut donc munir le moteur d'un asservissement de vitesse, et pour surveiller l'évolution de cette vitesse il faut utiliser un capteur.

1.8 Le codeur

La mesure des déplacements, des positions et des vitesses du machines rotatives est un problème régulièrement rencontré dans l'industrie: robots, cisailles, machines-outils, bobineuses, ... Les systèmes de détection tout ou rien conventionnels (détecteurs inductifs ou capacitifs, interrupteurs de position, capteurs photoélectriques, ...) apportent souvent une solution suffisante dans la plupart des applications mais à partir du moment où il est important d'effectuer un nombre important de mesures de positions, ces systèmes arrivent très rapidement à saturation (au bout du rouleau). Il est important de résoudre tous ces problèmes à l'aide de capteurs dont le positionnement n'est plus maîtrisé par le capteur physique proprement dit mais bien par le système de traitement numérique qui leur est associé [7]. Une des solutions consiste à utiliser un codeur optique.

Le capteur ou codeur optique rotatif, introduit dans les années 1970, est aujourd'hui très répandu et remplace souvent les génératrices tachymétriques, les résolveurs et les capteurs résistifs [8].

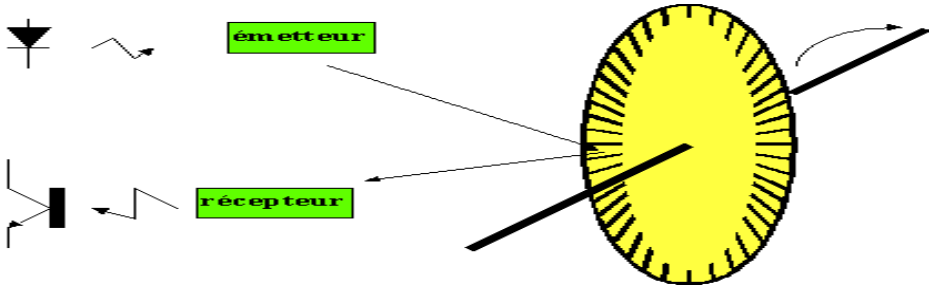


Figure.1.11: codeur optique

1.8.1 Principe de fonctionnement de codeur optique

Le codeur optique, le plus répandu, est lié mécaniquement à un arbre qui l'entraine. Il est constitué d'un système mécanique qui comporte un disque en verre portant des gravures opaques dont l'écartement est fonction du pas angulaire que l'on veut obtenir. Le faisceau lumineux, traversant le disque, est la plupart du temps généré par une diode électroluminescente et le faisceau modulé est capté par une photodiode ou un phototransistor [8].

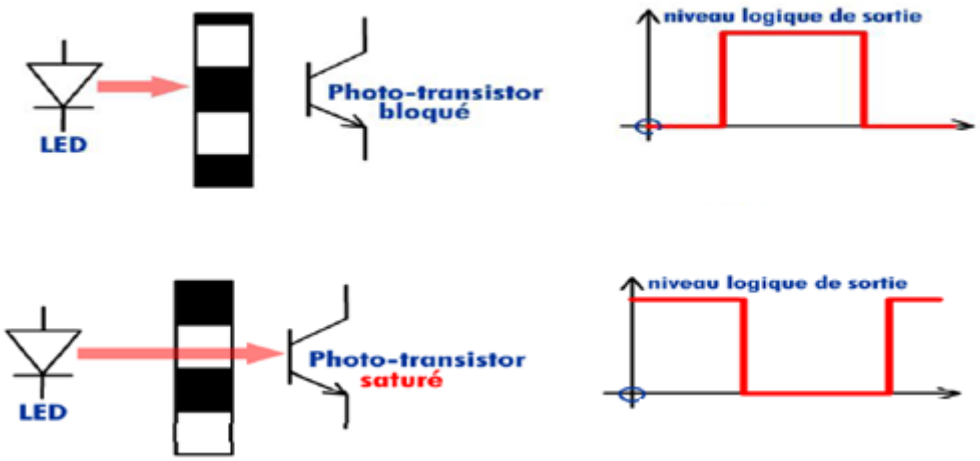


Figure.1.12: principe de fonctionnement de codeur optique

1.8.2 Les types de codeurs optiques

Il existe deux principaux types de codeurs optiques :

- Les codeurs incrémentaux
- Les codeurs numériques de position (codeurs absolus)

1.8.2.1 les codeurs incrémentaux

Le codeur incrémental est destiné à des applications ou l'information de position est obtenue par mesure du déplacement de l'objet. Le codeur délivre un train d'impulsions dont le nombre permet de déduire la valeur du déplacement ainsi que la vitesse car cette

dernière est proportionnelle à la fréquence des impulsions. Il est constitué d'un disque comportant deux à trois pistes :

- Piste extérieure : divisée en intervalles d'angles égaux, alternativement opaques et transparents. C'est le nombre de fenêtres ainsi créées qui détermine la résolution du capteur.
- Piste intérieure : qui ne comporte qu'une seule fenêtre et qui délivre qu'un signal par tour du disque. Ce « top zéro » permet de réinitialiser la partie commande et de connaitre une position d'origine.

Pour un tour complet de l'axe du codeur, la partie commande reçoit autant d'impulsions électriques qu'il ya de fenêtre, dont la durée dépend de la vitesse de rotation du disque [9].

1.8.2.2 les codeurs numériques (codeurs absolus)

Le codeur numérique de position est destiné à des applications pour lesquelles on souhaite obtenir l'information de position sans traitement par la partie commande. Il est constitué d'un disque comportant plusieurs pistes concentriques et d'une tète de lecture par piste. Le nombre de piste détermine le nombre de positions différentes qui peuvent être définies à l'intérieur d'un tour de disque. La partie commande reçoit directement un code numérique sur n bits (n étant le nombre de pistes), image de la position du disque à un instant donné [9].

Le codeur absolu comporte beaucoup d'avantages pour le codage précis et sans erreur de la position angulaire, puisque l'information de position est disponible dés la mise sous tension. Lorsqu'une information est parasitée, la position du codeur n'est pas perdue puisque les suivantes sont inhérentes à sa complexité [8].

Souvent il n'est pas nécessaire de coder une position de manière absolue, par exemple dans les asservissements. Dans ce cas, le codeur incrémental s'impose par sa conception simple à un seul disque, ce qui rend plus fiable et moins cher qu'un codeur absolu.

1.9 Conclusion

Dans ce chapitre, nous avons présenté une vue globale sur les moteur à courant continu, leurs caractéristiques, leurs différents types, et leurs réglages de vitesse. Nous avons présenté aussi les codeurs optiques qui peuvent surveiller l'évolution de vitesse de ces moteurs, a fin de réaliser un asservissement. Et pour cela nous avons besoin d'un régulateur, ce qu'on verra au prochain chapitre.

2.1 Introduction

Le contrôle des procédés est une discipline étudiée et utilisée dans plusieurs domaines de l'ingénierie. L'objectif d'une régulation ou d'un asservissement automatique d'un procédé est de le maintenir le plus près possible de son optimum de fonctionnement, prédéfini par un cahier des charges (conditions ou performances imposées). Le cahier des charges définit des critères qualitatifs à imposer qui sont traduits le plus souvent par des critères quantitatifs, comme par exemple, de stabilité, de précision, de rapidité ou de lois d'évolution [10].

Dans ce chapitre, on va essayer de comprendre la notion de régulation ou d'asservissement, puis la structure et le comportement des processus. Enfin les caractéristiques de régulateur PID (Proportionnel Intégral Dérivé) qui est le plus utilisé dans l'industrie. Même les systèmes les plus complexes de contrôle industriel peuvent comporter un réseau de contrôle dont le principal élément de control est un module de contrôle PID.

2.2 Définition de la régulation et de l'asservissement

- **L'asservissement :** la sortie doit suivre le plus fidèlement possible la consigne qui est variable et les grandeurs perturbatrices n'existent pas ou sont très peu influentes sur la grandeur à maitriser.
- **La régulation :** on cherche à maintenir constante la sortie conformément à une consigne qui est constante malgré l'action des perturbations.
- La régulation est un cas particulier de l'asservissement qui correspond tout simplement au cas d'une consigne constante.

2.3 Eléments constitutifs d'une boucle de régulation

Une boucle de régulation doit comporter au minimum les éléments suivants :

- Une variable de sortie: qui doit être contrôlée.
- Un capteur pour mesurer la variable de sortie.

- Un régulateur pour calculer l'action de contrôle. C'est-à-dire la valeur de la variable manipulée à partir de l'erreur.
- Un actionneur pour appliquer physiquement l'action de contrôle dont le but de changer ou de modifier le procédé, et de le ramener à son point de consigne.

2.4 Principe générale de la régulation

Dans l'étude classique des systèmes, on admet le principe de causalité qui signifie que la cause (entrée) agit sur un système pour produire un effet sur la sortie. Dans le cas des systèmes asservis, l'effet, à son tour, agit et influence la cause. Cette interdépendance entre cause et effet est le mécanisme fondamental qui permet à des systèmes de contrôler et de régler des grandeurs automatiquement.

Les automatismes à chaine fermée présentent un degré de complexité supplémentaire par le fait qu'ils sont capables d'agir sur la grandeur d'entrée en fonction des informations qu'ils mesurent sur la grandeur de sortie. A cet égard, cette possibilité lorsqu'elle s'offre à un système le conduit à une manifestation d'une forme d'intelligence. L'automatisme à chaine fermée le plus courant est le régulateur, dont le rôle est de maintenir constante la grandeur réglée, en fonction d'une valeur désirée appelée consigne ou référence. On a, dans ce cas, constitué une boucle de régulation et plus généralement une boucle d'asservissement. L'aspect régulation est considéré comme le plus important dans le milieu industriel, car les valeurs des consignes sont souvent fixes.

Toute chaine de régulation (ou d'asservissement) comprend trois maillons indispensables : l'organe de mesure, l'organe de régulation et l'organe de contrôle. Il faut donc commencer par mesurer les principales grandeurs servant à contrôler le processus. L'organe de régulation récupère ces mesures et les compare aux valeurs souhaitées, plus communément appelées valeurs de consigne. En cas de non-concordance des valeurs de mesure et des valeurs de consigne, l'organe de régulation envoie un signal de commande à l'organe de contrôle (vanne, moteur, etc.), afin que celui-ci agisse sur le processus [12].

Le choix des éléments de la chaine de régulation est dicté par les caractéristiques du processus à contrôler, ce qui nécessite de bien connaitre le processus en question et son comportement [12].

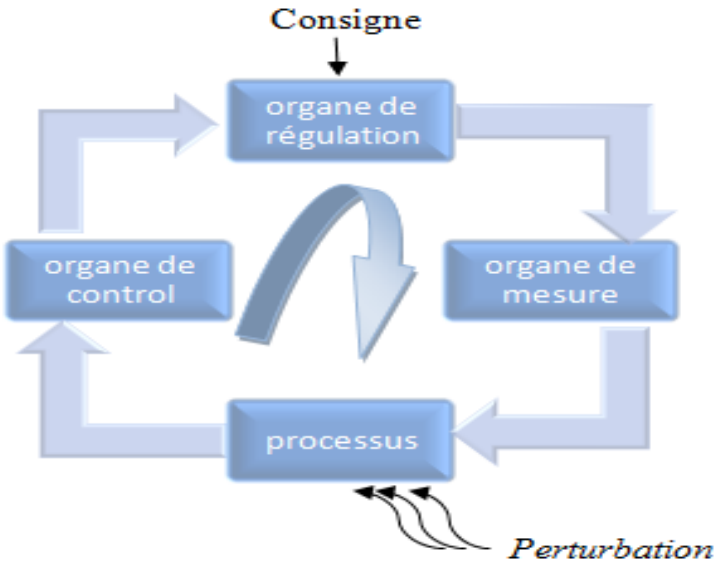


Figure 2.1.shéma de principe d'une chaine de régulation

2.5 Les différents types de boucles de régulation

2.5.1 Régulation en boucle ouverte

Dans un asservissement en boucle ouverte, l'organe de contrôle ne réagit pas à travers le processus sur la grandeur mesurée (celle-ci n'est pas contrôlée). Une régulation en boucle ouverte ne peut être mise en œuvre que si l'on connait la loi régissant le fonctionnement du processus. Contrairement à un asservissement en boucle fermée, un asservissement en boucle ouverte permet d'anticiper les phénomènes et d'obtenir des temps de très courts. Enfin, l'asservissement en boucle ouverte est la seule solution envisageable lorsqu'il n'y a pas de contrôle final possible.

Au niveau des inconvénients, la régulation en boucle ouverte impose de connaitre la loi régissant le fonctionnement du processus. Et aussi, il n'y a aucun moyen de contrôler, à plus forte raison de compenser, les erreurs, les dérives, les accidents qui peuvent intervenir à l'intérieur de la boucle, autrement dit, il n'y a pas de précision ni surtout de fidélité qui

dépendent de la qualité intrinsèque des composants. Enfin, la régulation en boucle ouverte ne compense pas les facteurs perturbateurs [12].

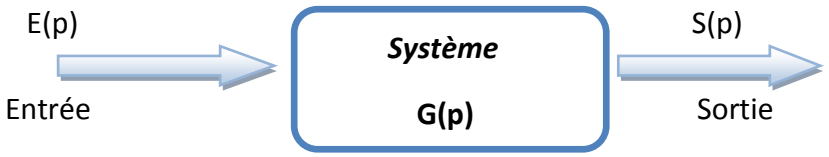


Figure 2.2.schéma bloc d'un système en boucle ouverte

Le comportement du processus est décrit par la relation :

$$S(p)=G(p).E(p) \qquad (2.1)$$

S :grandeur réglée (sorite)

E :grandeur réglante (entrée)

G(p) :fonction de transfert

2.5.2 Régulation en boucle fermée

Dans ce qui vient d'être dit, la variable de sortie (de la chaine de régulation) exerce une influence sur la valeur de la variable d'entrée (de la chaine de régulation) ou variable contrôlée, pour la maintenir dans des limites définies : il s'agit d'une régulation sur la variable ou d'asservissement en boucle fermée. L'action de la grandeur réglante sur la variable contrôlée s'opère à travers le processus qui boucle la chaine.

Dans une régulation en boucle fermée, une bonne partie des facteurs perturbateurs sont automatiquement compensés par la contre-réaction à travers le procédé. Autre avantage, il n'est pas nécessaire de connaitre avec précision les lois, le comportement des différentes composants de la boucle, et notamment du processus, bien que la connaissance des allures statistiques et dynamique des divers phénomènes rencontrés soit utile pour le choix des composants.

Parmi les inconvénients d'une régulation en boucle fermée, il faut citer le fait que la précision et la fidélité de la régulation dépend de la fidélité et de la précision sur les valeurs mesurées et sur la consigne. Autre inconvénient, sans doute plus important, le comportement dynamique de la boucle dépend des caractéristiques des différents

composants de la boucle, et notamment du processus, enfaite un mauvais choix de certaines composants peut amener la boucle à entrer en oscillation [12].

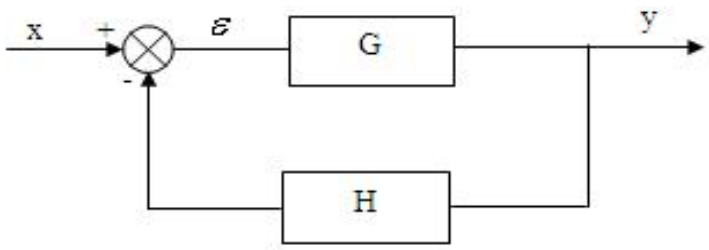


Figure 2.3.schéma bloc d'un système en boucle fermée

Le comportement du processus est décrit par la relation :

$$H_{BF} = \frac{Y}{X} = \frac{G}{1+G.H} \qquad (2.2)$$

Avec Y=G.ε

Et ε =X−H.Y (2.3)

H_{BF} : fonction de transfert en boucle fermée

X : grandeur réglante (consigne)

Y : grandeur réglée (sortie)

ε : l'erreur.

2.5.3 Régulation en cascade

La régulation en cascade se base sur l'utilisation de deux ou trois régulateurs. Elle est mise en place principalement pour réduire les effets des perturbations.

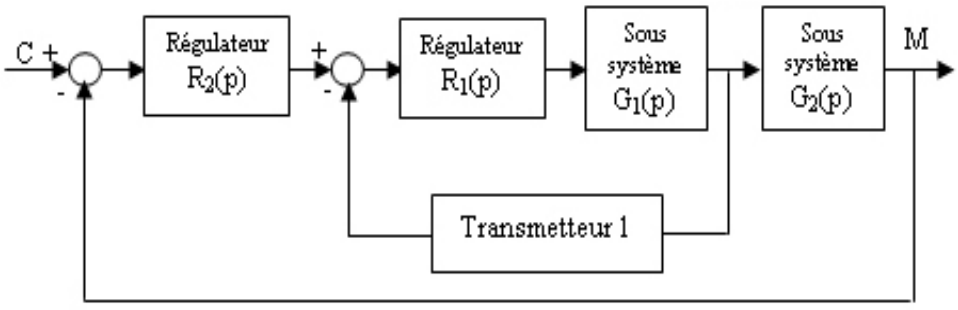


Figure 2.4.exemple de régulation en cascade

2.5.4 Régulation mixte

Ce type de régulation est l'association d'une régulation en boucle fermée et d'une régulation en boucle ouverte. Les deux boucles sont complémentaires et elles associent leurs actions pour améliorer la stabilité globale. Ce type de régulation est à mettre en œuvre lorsqu'une perturbation effecte directement la grandeur à regler [12].

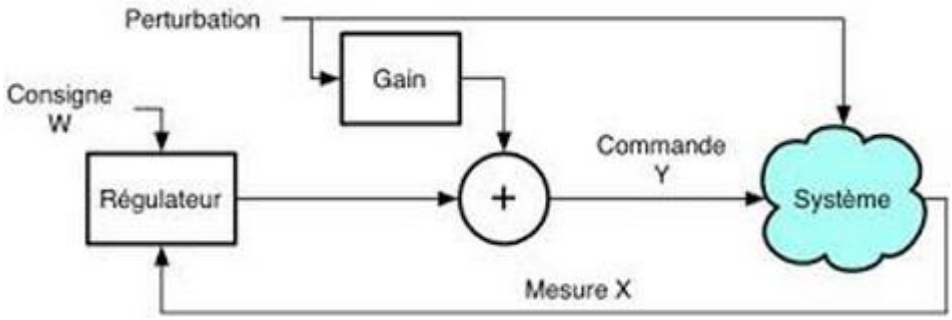


Figure 2.5.exemple de régulation mixte

2.5.5 Régulation de rapport

On utilise une régulation de rapport quand on veut un rapport constant entre deux grandeurs réglées S_1 et S_2 (S1/S2=constant).

Dans l'exemple ci-dessous, la grandeur piloté S_1 est utilisée pour calculer la consigne de la boucle de régulation S_2 (figure 2.6).

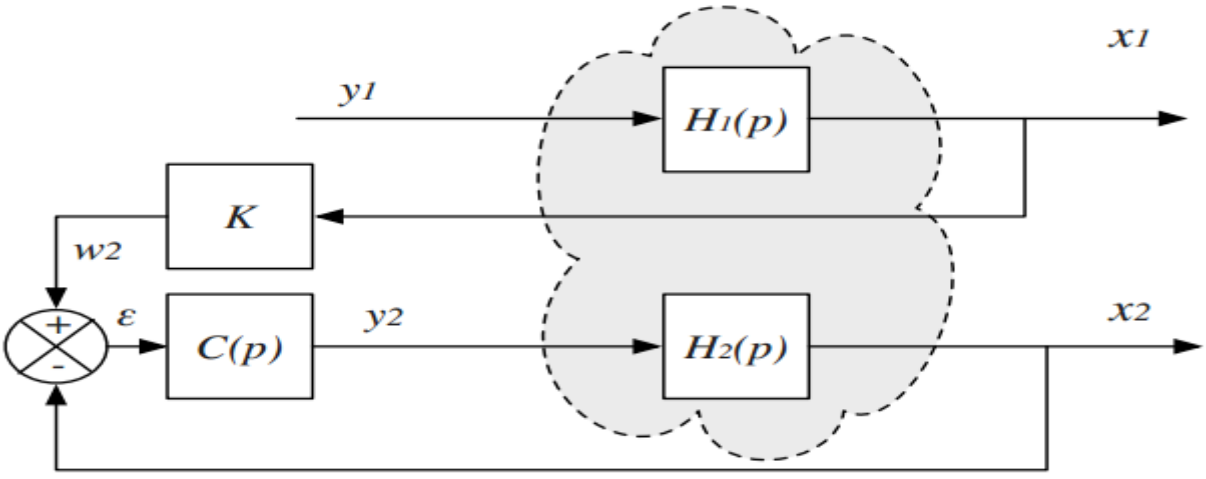


Figure 2.6.exemple de régulation de rapport

2.6 Précision, stabilité et rapidité des systèmes asservis

2.6.1 la précision des systèmes asservis

Un système asservi en boucle fermée est dit précis, si sa sortie S(t) est proche de la consigne (valeur désirée) E(t), on peut représenter l'erreur entre la consigne et la sortie :

$$\varepsilon = E(p) - S(p) \qquad (2.4)$$

cette erreur dépend de l'entrée et du gain de la fonction de transfert, car l'augmentation du gain permet d'avoir une meilleur précision [4].

2.6.2 la stabilité des systèmes asservis

Dans le cas des systèmes linéaires représentés par une fonction de transfert, l'analyse des poles permet de conclure sur la stabilité du système [4].

On définira la stabilité par une des propositions suivantes : un système linéaires est stable :

- Lorsque sa réponse à un échelon prend une valeur finie en régime permanent.
- Lorsque sa réponse à une impulsion tend vers 0.
- Lorsque sa réponse à une sinusoïde est une sinusoïde d'amplitude finie.

2.6.3 la rapidité des systèmes asservis

La rapidité d'un système régulé s'évalue par le temps nécessaire à la mesure pour entrer dans une zone ± 5% de sa valeur final (soit entre 95% et 105%). Le système régulé est d'autant plus rapide que le temps de réponse à 5% est court [4].

2.7 Les caractéristiques du régulateur PID

La commande PID est dite aussi (correcteur, régulateur, contrôleur), se compose de trois termes P,I et D, ou le « P » correspond au terme proportionnel, « I » pour le terme intégral et « D » pour le terme dérivée de la commande.

Un régulateur PID permet d'effectuer une régulation en boucle fermée d'une grandeur physique d'un système industriel.

La commande PID est insérée dans la chaine directe de l'asservissement, en série avec le processus, comme indiqué dans la (figure.2.6). Ce régulateur élabore à partir du signal d'erreur ε une commande U en fonction de trois actions proportionnelle, intégrale et dérivée[3].

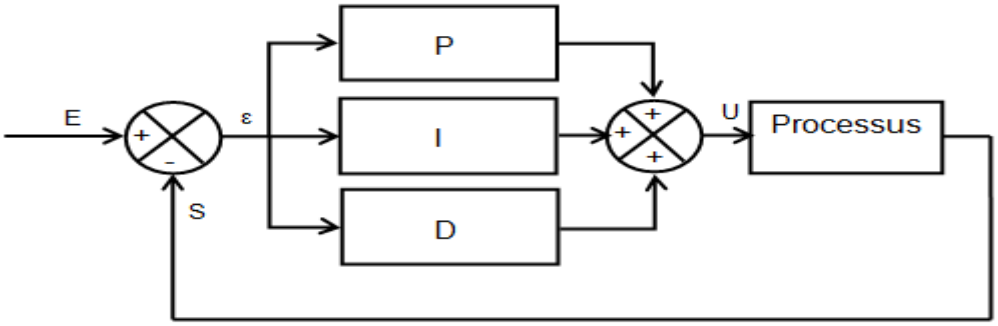


Figure 2.7.schéma bloc d'un système avec correcteur PID

$$U_c(t) = K_p\, e(t) + \frac{1}{Ti}\int_0^t e(t)dt + T_d \frac{de(t)}{dt}$$

$$= K_p\, e(t) + K_i \int_0^t e(t)dt + T_d \frac{de(t)}{dt}$$

$$\overset{T.L}{\Longrightarrow} \quad U_c(p) = K_p\, e(p) + K_i \frac{e(p)}{P} + k_d\, p\, e(p) \qquad (2.5)$$

K_p : gain d'action proportionnelle

K_i : gain d'action intégrale

K_d: gain d'action dérivée

T_i : constante de temps, dite temps d'action intégrale.

T_d :constante de temps, dite temps d'action dérivée.

2.7.1 Action proportionnelle

Ce correcteur élémentaire est le correcteur de base, il agit principalement sut le gain du système asservi et permet donc d'améliorer notablement la précision [2].

$$U_c(t) = K_p\, e(t) \quad \overset{T.L}{\Longrightarrow} \quad U_c(p) = K_p\, e(p) \qquad (2.6)$$

Lorsque l'on augmente le gain « K_p » le système réagit plus vite et l'erreur statique s'en trouve améliore , mais en contrepartie le système perd la stabilité.

- Effets du correcteur :
  - Diminution du temps de montée.
  - Diminution de l'erreur statique.
  - Augmentation du temps de stabilisation.
  - Augmentation du dépassement.

2.7.2 Action proportionnelle-intégrale (PI)

Ce correcteur contient l'action intégrale en plus de l'action proportionnel précédemment discutée. la loi de commande est donnée par :

$$U_c(t)= K_p\ e(t) + \frac{1}{Ti}\int_0^t e(t)dt \quad \overset{T.L}{\Longrightarrow} \quad U_c(p)= K_p\ e(p) + K_i\frac{e(p)}{P} \qquad (2.7)$$

Le terme intégral complète l'action proportionnelle puisqu'il permet de compenser l'erreur statique et d'augmenter la précision en régime permanent. L'intégrale agissant comme un filtre sur le signal intégré, elle permet de diminuer l'impact des perturbations (bruit, parasites), et il en résulte alors un système plus stable.

Malheureusement, un terme intégral trop important peut lui aussi entrainer un dépassement de la consigne et une stabilisation plus lente [3].

- Effets du correcteur :
  - Diminution du temps de montée.
  - Elimination de l'erreur statique.
  - Augmentation du temps de stabilisation.
  - Augmentation du dépassement.

2.7.3 Action proportionnelle-intégrale-dérivée (PID)

L'action conjugée PID permet une régulation optimal en associant les avantages de chaque action : la composante P réagit à l'apparition d'un écart de réglage, la composante D s'oppose aux variations de la grandeur réglée et stabilise la boucle de régulation et la

composante I élimine l'erreur statique. Et c'est pou cela que ce type de correcteur est le plus utilisé en milieu industriel [12].

- Effets du correcteur :
  - Diminution du temps de montée.
  - Elimination de l'erreur statique.
  - Diminution du dépassement.
  - Diminution du dépassement.

Remarque :

Pour construire un correcteur PID, il suffit d'assembler les trois actions (P,I et D), cet assemblage peut être fait de plusieurs façons possibles. On peut par exemple mettre les trois actions en série, ou en parallèle.

2.7.4 Résumé des actions PID [2] :

Action	Rôle et domaine d'utilisation
P	L'action proportionnelle agit de manière instantanée, donc rapide. A fin de diminuer l'écart de réglage et rendre le système plus rapide, on augmente le gain mais, on est limité par la stabilité du système. Le régulateur P est utilisé lorsque on désire régler un paramètre dont la précision n'est pas importante, exemple : régler le niveau dans un bac de stockage.
I	L'action intégral complète l'action proportionnelle. Elle permet d'éliminer l'erreur résiduelle en régime permanent. A fin de rendre le système plus dynamique (diminuer le temps de réponse). L'action intégrale est utilisé lorsque on désire avoir en régime permanent, une précision parfaite.
D	L'action dérivée, en compensant les inerties dues au temps mort, accélère la réponse du système et améliore la stabilité de la boucle, en permettant notamment un amortissement rapide des oscillations dues à l'apparition d'une perturbation ou à une variation subite de la consigne. L'action D est utilisée dans l'industrie pour le réglage des variables lentes telles

	que la température, elle n'est pas recommandée pour le réglage d'une variable bruitée ou trop dynamique (la pression)

2.8 Le PID numérique

Afin d'implémenter le correcteur dans un calculateur numérique, il faut discrétiser les résultats obtenus en continu pour avoir la version numérique de la fonction du transfert.

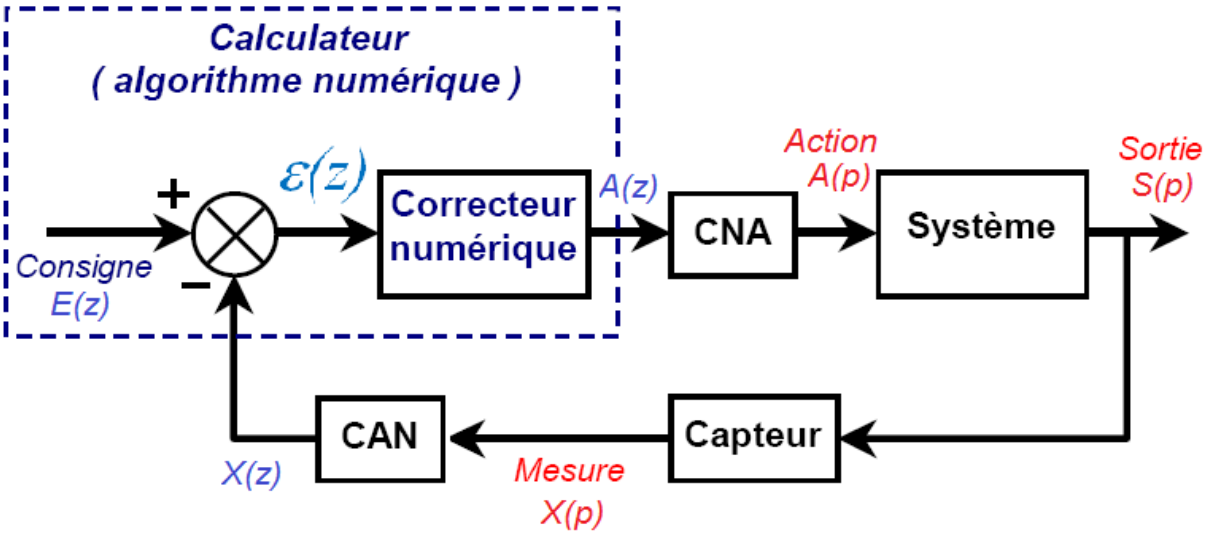
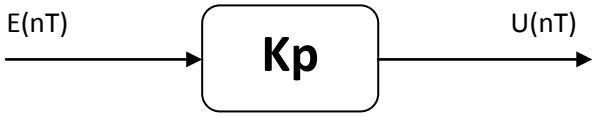


Figure 2.8.asservissement numérique en boucle fermée

2 .8.1 Action proportionnelle (P)

Le contrôleur P proportionnelle à l'erreur entre la consigne et le signal de sortie mesurée. Avec le coefficient d'amplification Kp on obtient la relation entre la commande U(nT) et l'écart de réglage E(nT) par :



$$U(nT) = Kp\ E(nT) \Rightarrow U(z) = Kp\ E(z) \qquad (2.8)$$

Donc la fonction de transfert est : $\frac{U(z)}{E(z)} = Kp$ (2.9)

2.8.2 Action intégrale et correcteur (PI)

L'opération d'intégration doit être remplacée par une opération de sommation, le correcteur I somme l'écart de réglage en fonction du temps dans le cas discret E(nT)

$$U(nT) = Ki \sum_{j=0}^{n} E(jT)$$

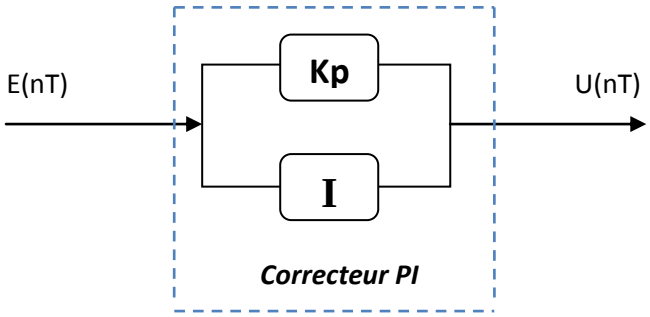
$$= Ki \sum_{j=0}^{n-1} E(jT) + Ki\ E(nT)$$

$$U(nT) = U((n-1)T) + Ki\ E(nT) \qquad (2.10)$$

D'où la fonction de transfert du correcteur I est :

$$R(z) = \frac{U(z)}{E(z)} = Ki \frac{1}{(1-z^{-1})} = Ki \frac{Z}{(Z-1)} \qquad (2.11)$$

Le correcteur PI est une combinaison d'un correcteur P et correcteur I :



$$U(nT) = Kp\ E(nT) + \sum_{j=0}^{n} E(jT)$$

Alors la fonction de transfert en Z est :

$$R(z) = \frac{U(z)}{E(z)} = Kp + Ki \frac{Z}{(Z-1)}$$

D'où

$$R(z) = \frac{(Kp+Ki)Z - Kp}{(Z-1)} \qquad (2.12)$$

2.8.3 Action dérivée et correcteur (PD)

L'action dérivée se traduit par :

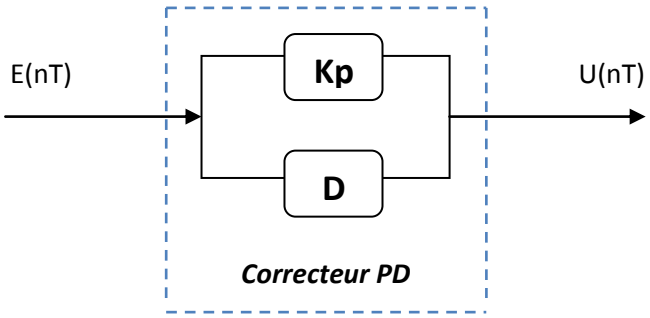
$$U(nT) = Kd\ \frac{E(nT)-E((n-1)T)}{nT-(n-1)T}$$

$$U(nT) = Kd\ (E(nT) - E((n-1)T))$$

D'où la fonction du transfert en Z est :

$$R(z) = \frac{U(z)}{E(z)} = Kd\ (1 - z^{-1}) = Kd\ \frac{(Z-1)}{Z} \qquad (2.12)$$

Le correcteur PD est une combinaison d'un correcteur P et correcteur D :

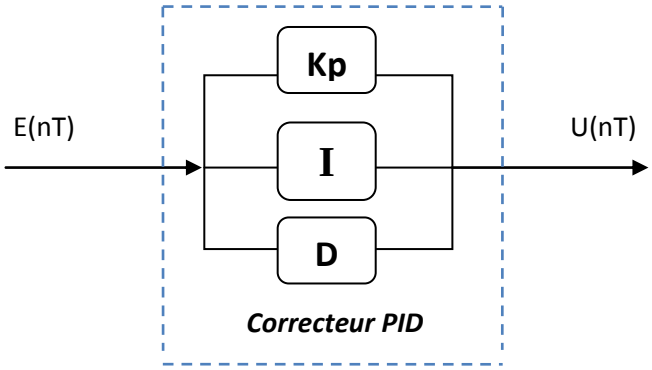


$$R(z) = \frac{U(z)}{E(z)} = Kp + Kd\ \frac{(Z-1)}{Z}$$

D'où : $$R(z) = \frac{(Kp+Kd)Z-Kd}{Z} \qquad (2.13)$$

2.8.4 correcteur PID

Le correcteur PID est une combinaison d'un correcteur P et I et d'un correcteur D c'est le plus connu entre les correcteurs :



La fonction de transfert est : $R(z) = \frac{U(z)}{E(z)} = Kp + Ki\frac{Z}{(Z-1)} + Kd\frac{(Z-1)}{Z}$

D'où $$R(z) = \frac{(Kp+Kd+Ki)Z^2 - (Kp+2Kd)Z + Kd}{Z(Z-1)} \quad (2.14)$$

2.9 Conclusion

Dans ce chapitre nous avons présenté le principe général de la régulation, les éléments principaux d'une boucle de régulation, et ces différents types, ainsi que les caractéristiques de régulateur PID numérique. Pour réaliser ce dernier, nous avons utilisé une carte Arduino comme élément principal, qui sera détaillé dans le prochain chapitre.

3.1 Introduction

Le système qu'on se propose de réaliser a pour objectif ; la régulation ou l'asservissement numérique de la vitesse d'un moteur à courant continue. Dans ce chapitre, on va présenter la partie électronique et la partie software de notre dispositif.

3.2 Synoptique générale

Le schéma synoptique de notre projet est donné par la figure 3.1

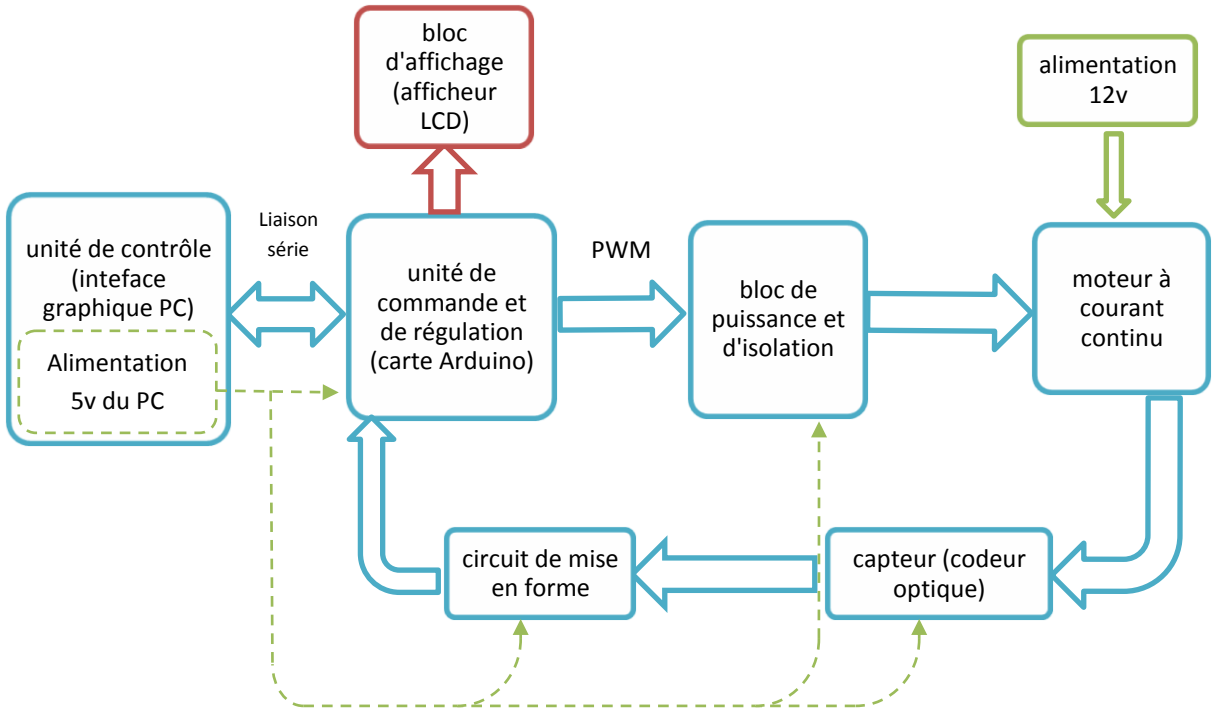


Figure 3.1.Synoptique générale du projet

3.3 Partie hardware

3.3.1 Bloc de commande et d'affichage

Dans notre projet nous avons utilisé une carte Arduino pour commander et contrôler le système de régulation. Cette carte est une plate-forme de prototypage d'objets interactifs à usage créatif constituée d'une simple carte à microcontrôleur, et d'un logiciel, véritable environnement de développement intégré, pour écrire, compiler et transférer le programme vers la carte à microcontrôleur.

Pour afficher les résultats de notre commande nous avons utilisé un afficheur LCD.

3.3.1.1 constitutions de la carte Arduino

Il existe plusieurs types de cartes Arduino. Dans notre cas nous avons utilisé une carte Arduino UNO qui est représentée dans La figure ci-dessous.

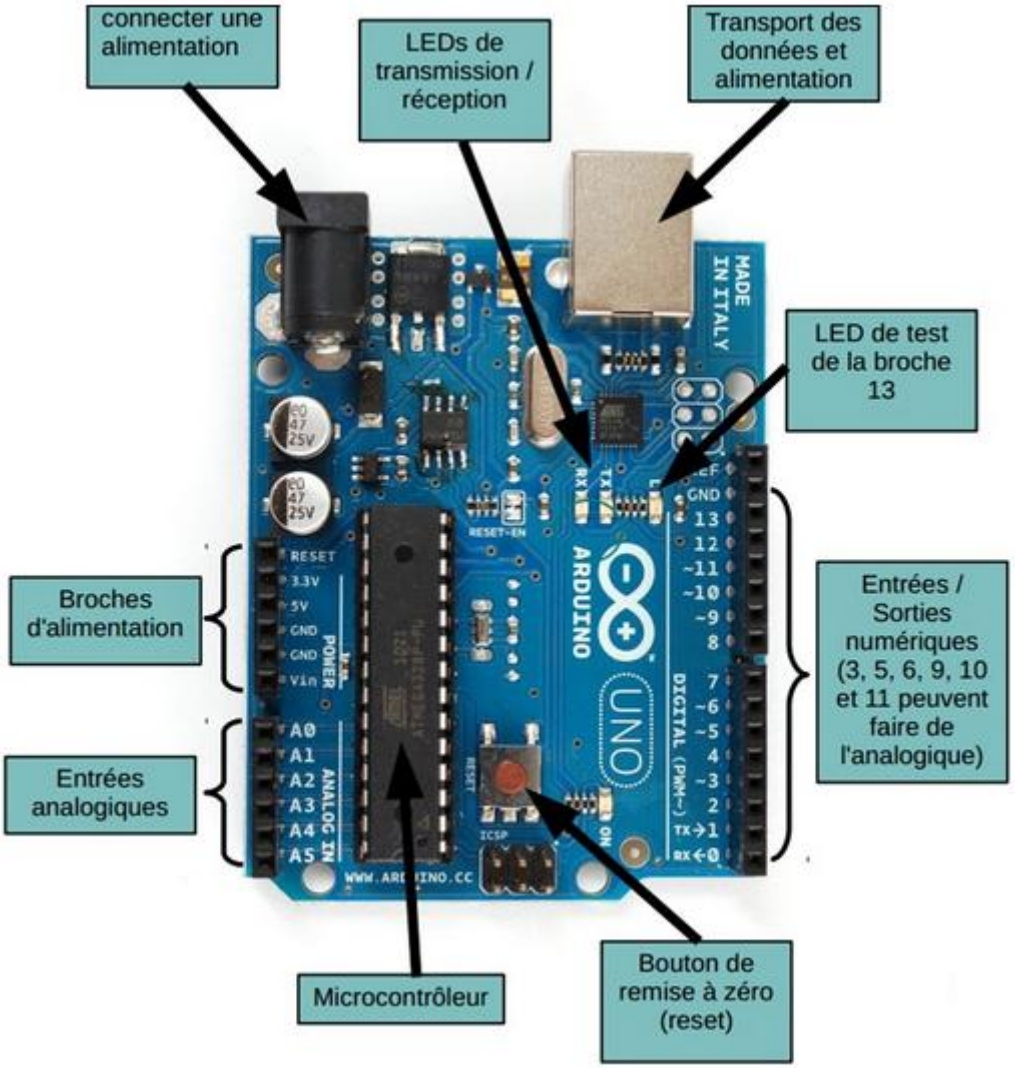


Figure 3.2.Carte Arduino UNO

a) Alimentation de la carte

Pour fonctionner, la carte à besoin d'une alimentation. Le microcontrôleur fonctionnant sous une tension de 5V, la carte peut être alimentée en 5V par le port USB ou bien par une alimentation externe qui est comprise entre 7V et 12V. Cette tension doit être continue et peut par exemple être fournie par une pile de 9V. Un régulateur se charge ensuit de réduire la tension à 5V pour le fonctionnement de la carte.

Pour alimenter la carte par un port USB, il faut assurer la protection du port et pour cela la carte Arduino UNO intègre un fusible réinitialisable qui protège le port USB de votre

ordinateur contre les surcharges en intensité (le port USB est généralement limité à 500mA en intensité). Bien que la plupart des ordinateurs aient leur propre protection interne, le fusible de la carte fournit une couche supplémentaire de protection. Si plus de 500mA sont appliqués au port USB, le fusible de la carte coupera automatiquement la connexion jusqu'à ce que le court-circuit ou la surcharge soit stoppé.

b) connecteurs des sources de tension (broches alimentation)

La première broche d'alimentation correspond à Reset qui a la même fonction de réinitialisation que le bouton de remise à zéro (reset). Pour forcer un redémarrage avec cette broche, il faut la forcer momentanément à 0 V (état bas). La deuxième broche fournie une tension de 3.3 V et la troisième une tension de 5 V, on a aussi deux broches pour la masse.

c) Entrées analogiques

Les broches d'entrée analogique de A0 à A5 permettent de mesurer une tension pour pouvoir en utiliser les valeurs dans les programmes. Précisons que les broches servent à mesurer une tension et non un courant. Le circuit d'entrée auquel chacune des broches donne accès possède une résistance interne (impédance) très importante, ce qui fait que le courant passant par la broche est toujours très faible. Ces entrées sont désignées comme étant de type analogique, et elles le sont par défaut, mais vous pouvez également les exploiter en tant qu'entrées numériques ou même comme sorties numériques.

d) Entrées et soties numériques

Les broches d'entrée et de sortie numérique de 0 à 13, lorsque vous les utilisez en tant que sorties, elles se comportent comme les broches de source de tension, sauf qu'elles fonctionnent toutes en 5V et qu'elles peuvent être activées et désactivées depuis le programme. Si vous activez une de ces broches, la tension présentée sera à 5V. Si vous désactivez la broche, la tension sera à 0V. Certaines de ces broches peuvent être configurées en sortie PWM. Les deux premières broches RX et TX sont réservées à la communication, transmission et la réception.

Toutes les broches numériques peuvent supporter jusqu'à 40 mA à 5V.

e) Les LEDs (visualisation)

La première LED qui se trouve à droite de la carte est connectée à une broche (pin 13) du microcontrôleur et va servir pour tester le matériel.

Les deux autre LEDs servent à visualiser l'activité sur la voie série, une pour l'émission et l'autre pour la réception.

f) Bouton de remise à zéro

Une pression sur cet interrupteur envoie une impulsion logique sur la broche RESET du microcontrôleur, ce qui le forcer à redémarrer en effaçant la mémoire. Sachez que les programmes stockés sur la carte ne sont pas effacés, parce qu'ils sont stockée dans la mémoire flash, c'est-à-dire que le contenu n'est pas perdu en cas de mise hors tension.

g) Microcontrôleur

Le microcontrôleur est l'élément principal de la carte Arduino, il est aujourd'hui implanté dans la plupart des applications grand public ou professionnelles.

Un microcontrôleur est littéralement un petit ordinateur contenu dans une seule puce. Il embarque même plus de fonctions que les premiers ordinateurs personnels, puisqu'il réunit un processeur central, un kilo-octet de mémoire vive (RAM) pour stocker les données, quelques kilo-octets de mémoire en lecture seule effaçable (EEPROM) ou en mémoire flash, dans lesquels vous pouvez stocker les programmes, ainsi que des broches d'entrée et de sortie. Ces broches permettent de relier de microcontrôleur à vous composants électroniques.

Le microcontrôleur utilisé sur la carte Arduino UNO est un microcontrôleur ATMega328. C'est un microcontrôleur ATMEL de la famille AVR.

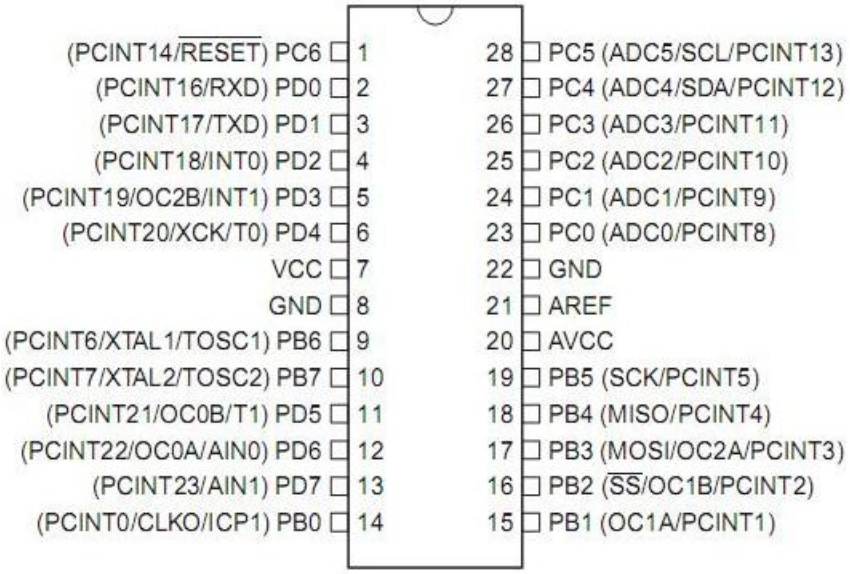


Figure 3.3.boitier de l'ATMega328

- **Caractéristique de l'ATMega328**
  - **Mémoire**

- flash de 32Koctets, c'est celle qui contiendra le programme à exécuter. Cette mémoire est effaçable et réinscriptible.
- RAM de 2 Koctets, servant à stocker les variables créées par le programme
- EEPROM de 1 Koctet, c'est le disque dur du microcontrôleur. Vous pourrez y enregistrer des informations qui ont besoin de survivre dans le temps, même si la carte doit être arrêtée. Cette mémoire ne s'efface pas lorsque l'on éteint le microcontrôleur ou lorsqu'on le reprogramme.
- Registres : c'est un type de mémoire utilisé par le processeur.
- Mémoire cache : c'est une mémoire qui fait la liaison entre les registres et la RAM.

  - **Le processeur**

C'est le composant principal du microcontrôleur. On le nomme souvent le CPU (Central Processing Unit). C'est lui qui va exécuter le programme que nous lui donnerons traité. Il lit une par une les instructions du programme qui est stocké dans la mémoire flash, puis les exécute en séquence. Parfois, cela suppose d'aller lire des données dans la mémoire vive (RAM), de les modifier et de les réécrire.

  - **Digital I/O (entrée sortie tout ou rien)**

3 ports : portB, portC,portD (soit 23 broches en tout I/O)

  - **Timer/counter**

L'ATMega328 comprend 3 Timers, c'est-à-dire trois compteurs qui sont incrémenté par le microcontrôleur tout les N ticks d'horloge. Le Timer0 est utilisé par la bibliothèque Arduino pour évaluer l'écoulement du temps et pour les sorties PWM (6 broches OCxA/OCxB) associés à ce Timer. Le Timer 1 dispose d'un compteur 16bits, alors que les Timers 2 et 0 sont en 8bits.

  - **Port série**

Emission et réception série via les broches TXD(PD1) et TXD(PD0)

  - **Comparateur analogique**

Broches AIN0(PD6) et AIN1(PD7) peut déclencher interruption

  - **Conversion analogique numérique**

6 entrées multiplexées ADC0(PD0) à ADC5(PC5)

- **Gestion bus I2C (TWI Two Wire Interface)**

Les bus est exploité via les broches SDA(PC5)/SCL(PC4)

- **Gestion d'interruptions (24 sources possibles)**

- Interruptions liées aux entrée INT0(PD2) et INT1(PD3)
- Interruptions sur changement d'état des broches PCINT0 à PCINT23
- Interruptions liées aux timers 0, 1 et 2.
- Interruption liée au comparateur analogique.
- Interruption de fin de conversion ADC.
- Interruptions du port série USART.
- Interruption du bus TWI(I2C).

3.3.1.2 Les avantages de la carte Arduino

- Les cartes Arduino sont relativement peu coûteuses comparativement aux autres plateformes.
- Un environnement de programmation claire et simple : il est facile à utilisé.
- Multiplateforme : le logiciel Arduino, écrit en java, tourne sous les systèmes d'exploitation windows, macintosh et linux. La plupart des systèmes à microcontrôleurs sont limités à windows.
- Logiciel open source et extensible : le logiciel Arduino et le langage Arduino sont publiés sous licence open source, disponible pour être complété par des programmateurs expérimentés.
- Matériel open source et extensible : les cartes Arduino sont basée sur les microcontrôleurs Atmel ATMEGA8, ATMEGA168, ATMEGA328, etc.… les schémas des modules sont publiés sous une licence Creative Commons, et les concepteurs de circuits expérimentés peuvent réaliser leur propre version des cartes Arduino, en les complétant et en les améliorant.
- Nombreux conseils, tutoriaux et exemples en ligne.

3.3.1.3 Afficheurs à cristaux liquide (LCD)

Les afficheurs à cristaux liquides sont des modules compacts intelligents et nécessitent peu de composants externes pour un bon fonctionnement. Ces écrans sont partout, on les trouve dans plein d'appareils électroniques : les montres, les tableaux de bord des voitures, les calculatrices, etc. Cette utilisation intensive est due à leur faible consommation et coût.

Les écrans LCD sont aussi sous forme complexes telle que la plupart des écrans d'ordinateur ainsi que les téléviseurs à écrans plat.

Plusieurs afficheurs sont disponibles sur le marché et ne diffèrent les uns des autres, non seulement par leurs dimensions, de 1 à 4 lignes de 6 à 80 caractères. L'afficheur utilisé dans notre projet est afficheur LCD 16*2 (figure3.4).

Figure 3.4.afficheur LCD 2*16

a) Principe des cristaux liquides

L'afficheur est constitué à deux lames de verre, distantes de 20 µm environ, sur lesquelles sont dessinées les mantisses formant les caractères. L'espace entre elles est rempli de cristal liquide normalement réfléchissant (pour les modèles réflectifs). L'application entre les deux faces d'une tension alternative basse fréquence de quelques volts (3 à 5 V) le rend absorbant. Les caractères apparaissent sombres sur fond clair. N'émettant pas de lumière, un afficheur à cristaux liquide réflectif ne peut être utilisé qu'avec un bon éclairage ambiant. Sa lisibilité augmente avec l'éclairage. Les modèles transmissifs fonctionnent différemment : normalement opaque au repos, le cristal liquide devient transparent lorsqu'il est excité ; pour rendre un tel afficheur lisible, il est nécessaire de l'éclairer par l'arrière [3].

b) Brochages

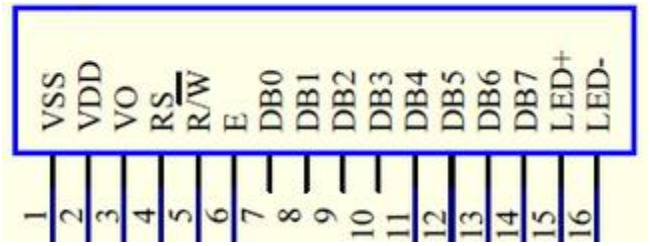


Figure 3.5.brochage du LCD

La broche 1 se nomme VSS et elle est censée se connecter à GND.

La broche 2 se nomme VDD et correspond à la broche d'alimentation de +5V.

La broche 3 est connectée au potentiomètre pour définir le contraste de l'afficheur.

La broche 4 est la broche RS et selon sa position, l'afficheur se prépare à recevoir des instructions ou des données.

La broche 5 est la broche R/W qui contrôle si le LCD est en mode émission ou réception.

La broche 6 est la broche d'activation, lorsqu'elle passe du niveau bas à haut et revient à bas, le LCD lit les broches 4,5 et 7-14.

La broche 7 à 14 sont les lignes de bus de données appelées DB0-DB7, qui sont les bits de données principaux envoyés au LCD et qui contrôlent ou et quoi écrire sur l'afficheur.

c) Commande d'un afficheur LCD

Deux modes de fonctionnement de l'affichage sont disponible, le mode 4 bits et le mode 8 bits :

- **Mode 8 bits :**

Dans ce mode 8 bits, les données sont envoyées à l'afficheur sur les boches D0 à D7. On place la ligne RS à 0 ou à 1 selon que l'on désire transmettre une commande ou une donnée. Il faut aussi placer la ligne R/W à 0 pour indiquer à l'afficheur que l'on désire effectuer une écriture. Il reste à envoyer une impulsion d'au moins 450 ns sur l'entrée E, pour indiquer que des données valides sont présentes sur les broches D0 à D7. L'afficheur lira la donnée sur le front descendant de cette entrée. Si on désire au contraire effectuer une lecture, la procédure est identique, mais on place cette fois la ligne R/W à 1 pour demander une lecture. Les données seront valides sur les lignes D0 à D7 lors de l'état haut de la ligne E. [3]

- **Mode 4 bits :**

Dans ce mode, seul les 4 bits de poids fort (D4 à D7) de l'afficheur sont utilisés pour transmettre les données et les lires. Les 4 bits de poids faible (D0 à D3) sont alors connectés à la masse, on a donc besoin hors alimentation de sept fils pour commander l'afficheur. Dans notre projet nous avons utilisé ce mode d'affichage.

3.3.1.4 bronchement d'un écran LCD avec une carte Arduino

En l'absence de l'ordinateur, nous avons besoin d'un écran, pour afficher les informations transmises par Arduino, et aussi d'un clavier pour contrôler ces informations.

Dans notre projet nous avons branché un écran LCD avec la carte Arduino UNO; Pour pouvoir lire la consigne et la vitesse réelle du moteur. Nous avons aussi ajouté deux boutons poussoir pour varier la vitesse.

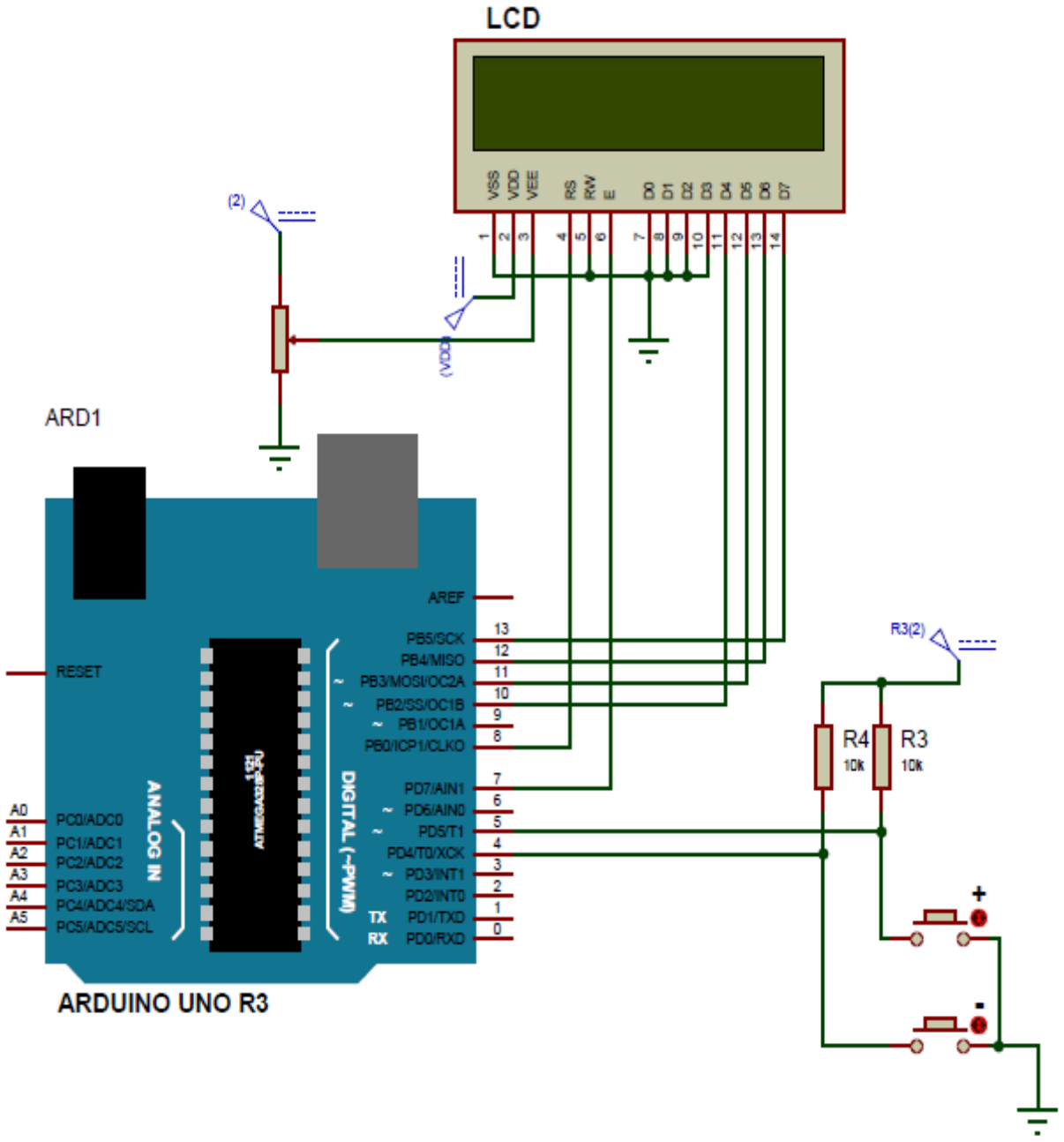


Figure 3.6. Schéma du branchement de LCD avec ARDUINO

L'afficheur LCD utilise 6 broches de données (D4 à D7 + RS et E) et deux d'alimentations +5 et masse. La plupart des écrans possèdent aussi une entrée analogique pour régler le contraste des caractères. Les 6 broches de données peuvent être placées sur n'importe quelles entrées/ sorties numériques de l'Arduino. Dans notre cas nous avons utilisé les broches (7, 8, 10, 11, 12 et 13).

Donc la carte Arduino est le cerveau de notre projet. Elle peut commander et contrôler tous notre système. Par exemple, dans ce cas elle contrôle l'affichage sur l'écran LCD.

3.3.2 Bloc de puissance et d'isolation

3.3.2.1 fonctionnements de la partie puissance

Pour contrôler la partie puissance de notre système, nous avons utilisé un MOSFET. C'est un type de transistor à effet de champ. Comme tous les transistors, le MOSFET module le courant qui le traverse à l'aide d'un signal appliqué sur son électrode centrale. Dans notre montage, on ne peut pas utiliser un transistor à la place du MOSFET car la perte de tension est très grande au borne du transistor.

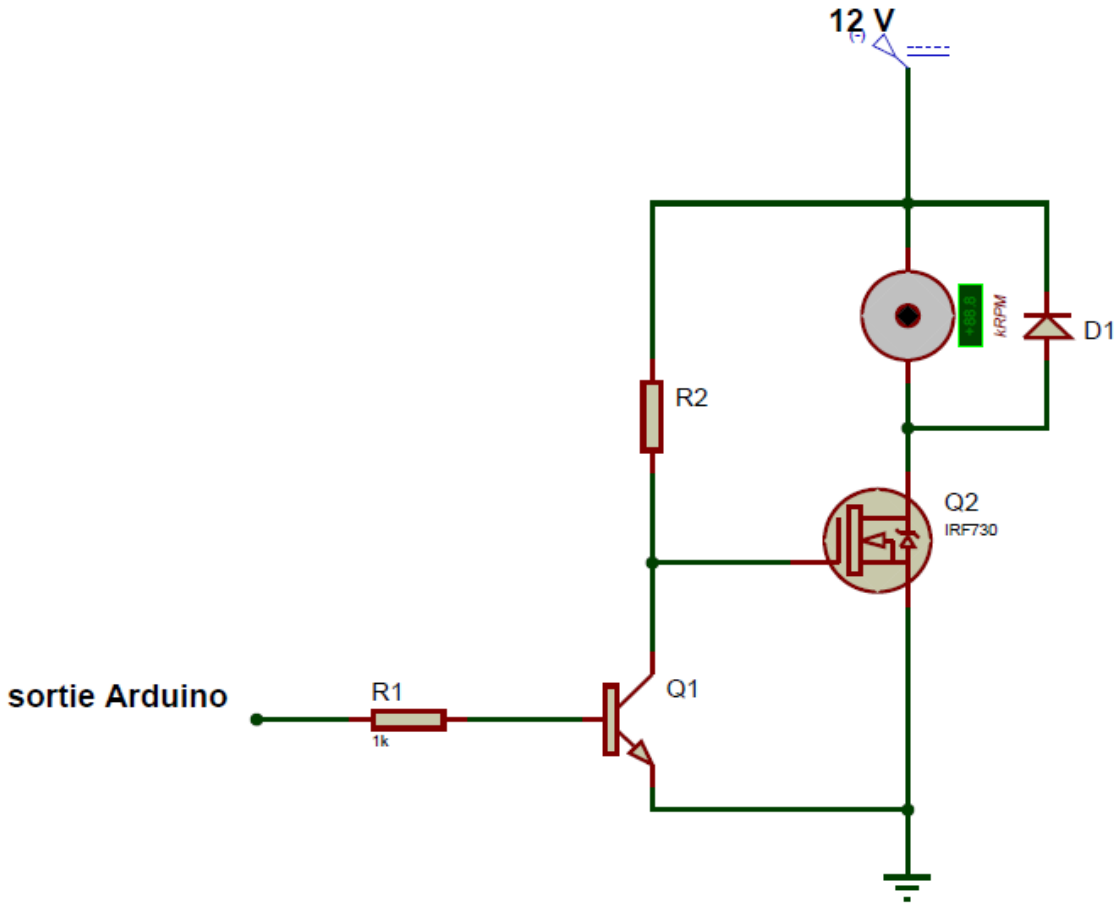


Figure 3.7. Schéma électrique de la carte de puissance

Les MOSFET sont plus faciles à utiliser. Il suffit d'appliquer une différence de potentiel sur la gâchette. Donc, Le transistor Q1 vient driver le MOSFET de puissance Q2. Lorsque la sortie Arduino est à l'état haut, Q2 est bloqué et le moteur ne tourne pas. A l'inverse, lorsque la sortie Arduino est à l'état bas, alors Q2 devient passant et le moteur tourne à plein régime.

La résistance R1 sert à limiter le courant du transistor. La résistance R2 permet de polariser la gâchette du MOSFET quand le transistor est bloqué, et de le faire ainsi conduire. Quand le transistor est passant, la sortie retrouve à la masse et le MOSFET ne conduit plus.

La diode D1 est une diode de roue libre qui est connectée en parallèle du moteur, pour la continuité du courant électrique. Si on ne met pas cette diode, le courant ne peut aller nulle part, va décroitre ainsi très vite et créer une surtension qui détruira le transistor. La roue libre écrête cette surtension en offrant un passage pour le courant. Dans notre cas, la diode de roue libre doit supporter en tension au moins 12 V, puisque nous avons utilisé un moteur de 12V.

Pour le choix du MOSFET, il faut s'intéresser à la tension maximale qu'il doit accepter avant de claquer, et aussi à la résistance drain source à l'état passant. Plus cette valeur est faible et moins l'échauffement du MOSFET est élevé.

Donc, Le moteur se contrôle en tout ou rien, ce qui est bien, car la sortie analogique de l'Arduino est en réalité une sortie PWM de rapport cyclique ajustable.

3.3.2.2 la génération du signal PWM (Pulse Width Modulation)

La modulation de largeur d'impulsion (MLI ou PWM), est une technique pour obtenir des effets d'allure analogique avec des broches numériques. Le contrôle est utilisé pour créer une onde carrée, un signal basculant entre un niveau HAUT et BAS. Cette succession de niveau HAUT/BAS peut simuler des tensions entre le niveau HAUT (5 volts) et le niveau BAS (0 volts) en faisant varier la proportion du temps où le signal est HAUT sut la proportion du temps où le signal est bas. La durée du temps du niveau HAUT est appelé largeur d'impulsion. Pour obtenir une variation analogique, il suffit de changer ou de modifier cette largeur d'impulsion (figure 3.8).

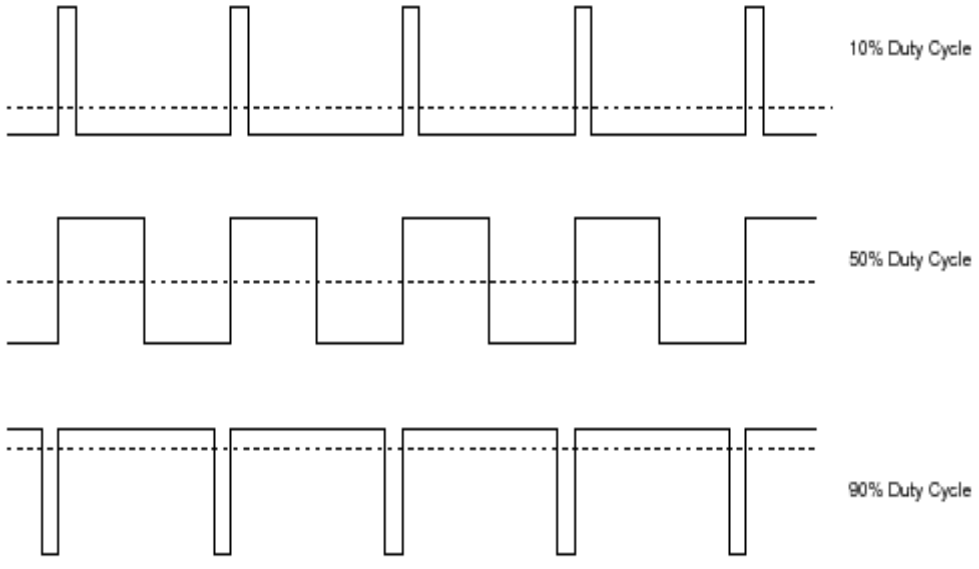


Figure 3.8. Modulation de largeur d'impulsion

Le principe est de générer un signal logique (valant 0 ou 1) à une fréquence fixe mais dont le rapport cyclique est contrôlé numériquement, dans notre cas la fréquence générée par la carte Arduino de l'ordre 500 Hz d'où la période égale à 2 millisecondes.

On peut changer la fréquence générée par Arduino, selon le type du moteur. Une fréquence de commutation plus élevée permet d'avoir un moteur plus silencieux mais fait chauffer un peu plus le MOSFET, d'où le choix et la commande de cet dernier devient plus critique.

3.3.2.3 fonctionnements de la partie d'isolation

Dans la plupart des circuits électroniques, les tensions mises en jeux ne sont pas compatibles de part et d'autre. Par exemple, dans notre circuit d'une part la carte Arduino est alimentée en 5V et de l'autre part le moteur est alimenté en 12V qui peut présenter une tension dangereuse.

Afin d'éviter ce problème et de protéger notre circuit, nous avons isolé la partie commande et la partie puissance. Cette isolation est réalisée par un optocoupleur.

Un optocoupleur est un composant qui permet le transfert d'information entre deux parties électroniques isolées l'une de l'autre d'un point de vue électrique. La première partie est un émetteur, et la seconde partie est un récepteur. Quand on parle d'émission, c'est en général parce que l'on émet quelque chose. Ici, il s'agit d'une émission de lumière. L'émetteur produit donc de la lumière, et le récepteur, qui est sensible à la lumière émise par l'émetteur.

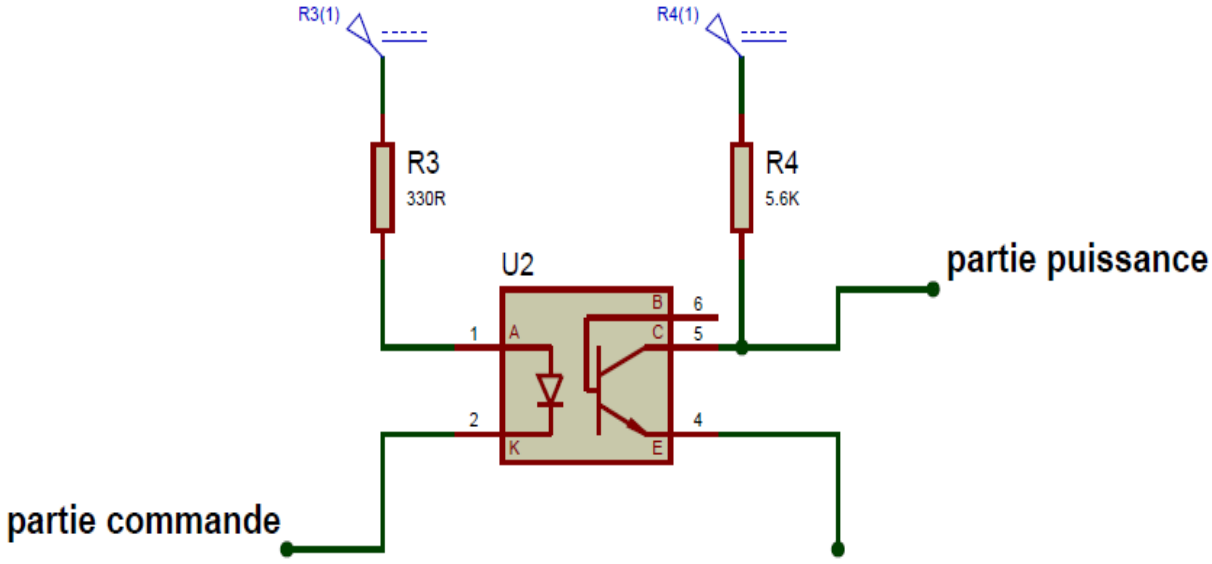


Figure 3.9. Circuit d'isolation par un optocoupleur

L'entrée est composée d'une diode électroluminescente « LED » et la sortie est un photo-détecteur c'est-à-dire une photo diode, phototransistor ou un photo-thyristor.

La LED et le photo-détecteur sont reliés optiquement, mais sont isolés électriquement dans un même boitier.

Le circuit que nous avons utilisé est le 4N28, avec des résistances R3 pour abaisser la tension à une valeur acceptable pour l'émetteur, et R4 pour la polarisation du transistor.

3.3.3 Capteur et circuit de mise en forme

Grace au circuit d'isolation, on peut transférer la commande de vitesse de la carte Arduino vers le moteur à courant continu. Mais l'Arduino à besoin aussi d'une information sur la vitesse réelle du moteur pour faire un asservissement. Donc nous avons utilisé un capteur optique qu'on à bien détaillé au premier chapitre.

Notre codeur optique rotatif fonctionne selon le principe de balayage infrarouge d'un disque gradué (roue) noir et blanc. Lorsque ce disque effectue une rotation, si le capteur est sur la ligne blanche, le signal infrarouge émis par la diode est réfléchie et capté par le phototransistor.

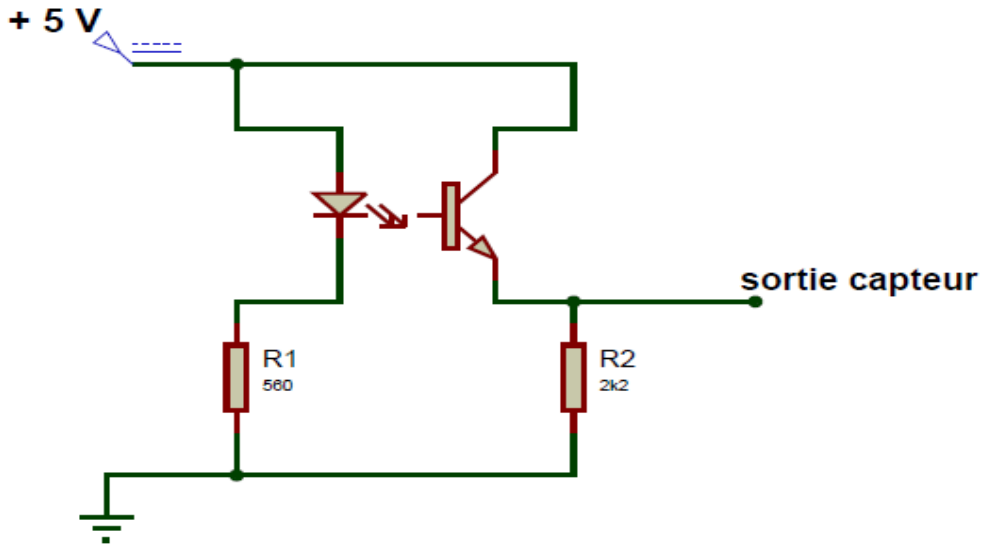


Figure 3.10. Schéma développé du capteur optique

Le signal qui sort du capteur est un signal pulsé, et pour pouvoir le rendre lisible pour la carte Arduino nous avons utilisé un circuit de mise en forme.

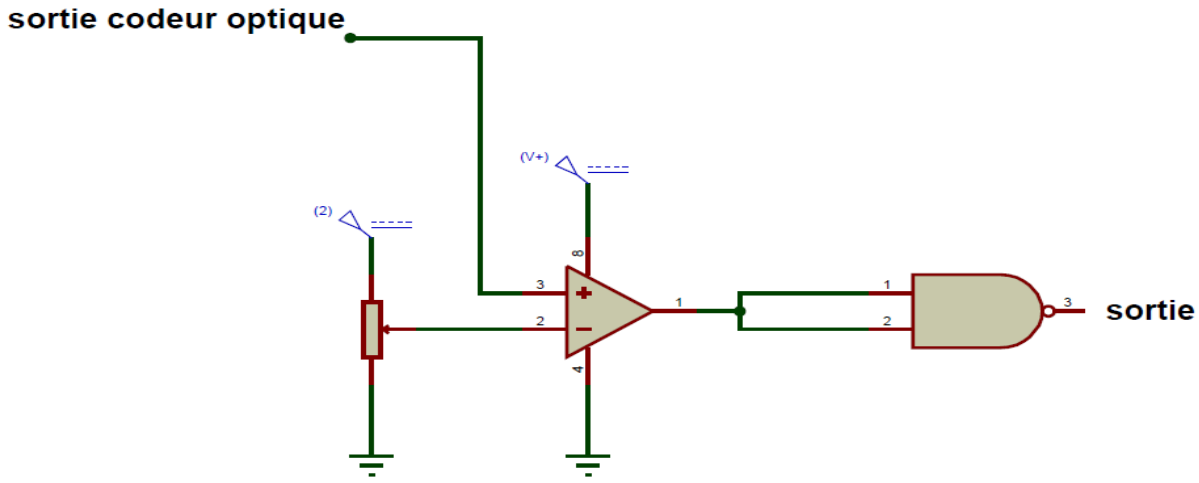


Figure 3.11. Circuit de mise en forme

Nous avons utilisé un amplificateur opérationnel LM358 qui est monté en comparateur, pour comparer la tension reçue du capteur avec une tension de seuil réglable grâce à un potentiomètre. Les tensions extrêmes maximales que peut faire sortir l'amplificateur opérationnel n'atteignent pas les valeurs des tensions d'alimentation. Selon les modèles, une différence de 2V peut être observée. Le signal qui sort de notre comparateur est un signal carré qui varie entre deux valeurs 0V et 3.6V. Donc nous avons branché à la sortie un circuit intégré de type 7400 qui contient 4 portes logique NON-ET (NAND), et qui donne à la sortie un signal carré inversé variant entre 0V et 5V; qui sera reconnu par la carte Arduino.

3.3.4 Alimentation

L'alimentation étant nécessaire pour tout circuit électronique. La fonction principale est de délivrer d'une ou de plusieurs tensions (ou courant) bien précises et souvent stables. Aussi de fournie une énergie électrique avec un minimum de pertes.

Les basses puissances dans notre circuit sont alimentées par l'ordinateur à une tension de 5V à partir du port USB (figure 3.1).

Pour l'alimentation du moteur, nous avons réalisé une alimentation stabilisée de 12V.

La carte réalisée contient :

- **Un transformateur abaisseur**

Qui fournit sur son secondaire une tension alternative très inferieur à celle du secteur.

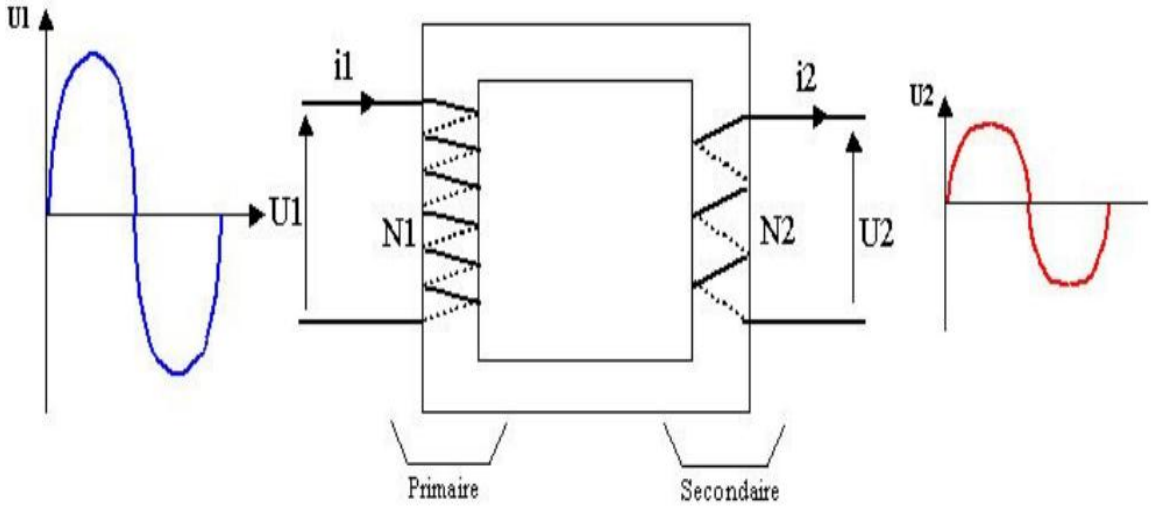


Figure 3.12. Fonctionnement du transformateur

Nous avons choisi un transformateur dont la tension au secondaire est de 12V et un courant de 1A.

- **Un pont redresseur**

Qui fournie en sortie une tension non plus alternative mais redressé.

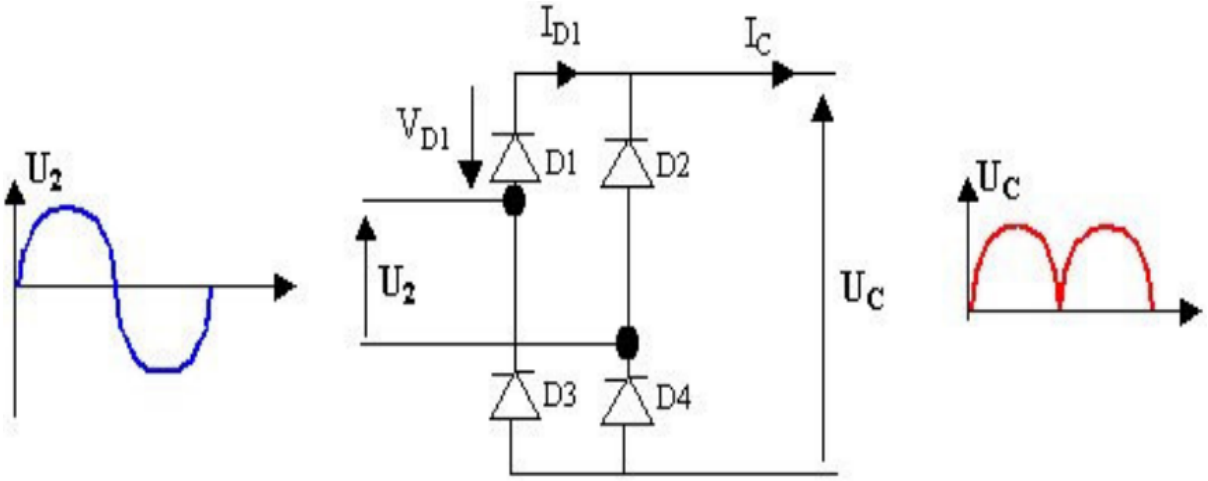


Figure 3.13. Redressement double alternance

Le choix de pont de redressement est basé sur :

- La tension inversée maximale de diode.
- Le courant moyen direct.

- **Condensateur de filtrage**

Après le redressement, la tension de sortie aux bornes du pont redresser est loin d'être continue. Le filtrage à pour but de transformer cette tension redressée en une tension continue légèrement ondulée.

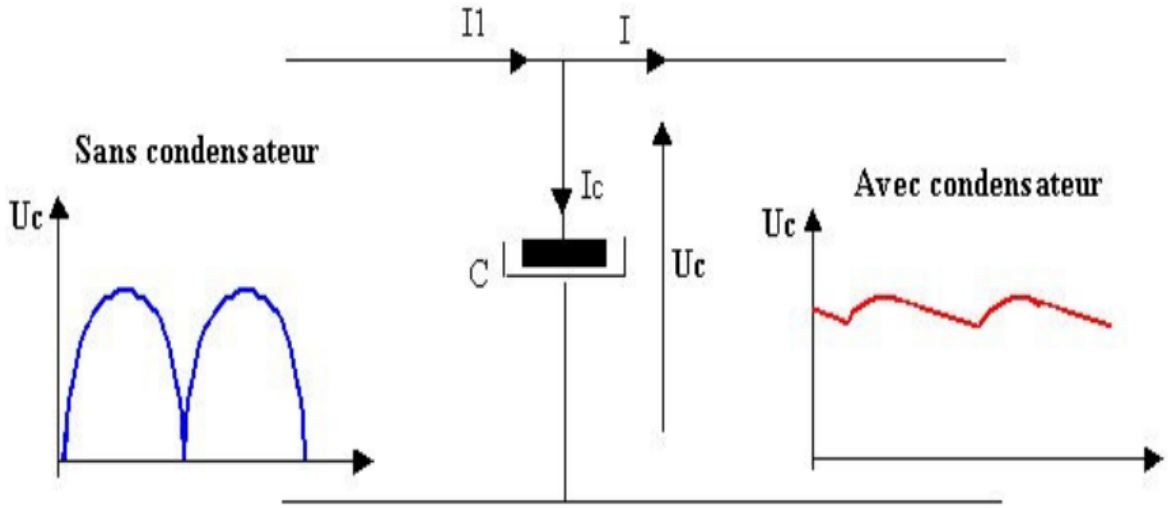


Figure 3.14. Fonctionnement de condensateur

Pour obtenir une tension presque constante, il faut brancher un ou plusieurs condensateurs en parallèle juste après le pont redresseur. Plus la valeur de la capacité est élevée, plus le filtrage sera meilleur. Les deux principaux critères à considérer dans le choix d'un condensateur sont :

- Sa capacité.
- Sa tension de service.

Régulateur de tension

Malgré le filtrage, la tension aux bornes du condensateur n'est pas parfaitement continue, elle présente une légère ondulation. Pour obtenir une tension parfaitement continue, on utilise un régulateur de tension.

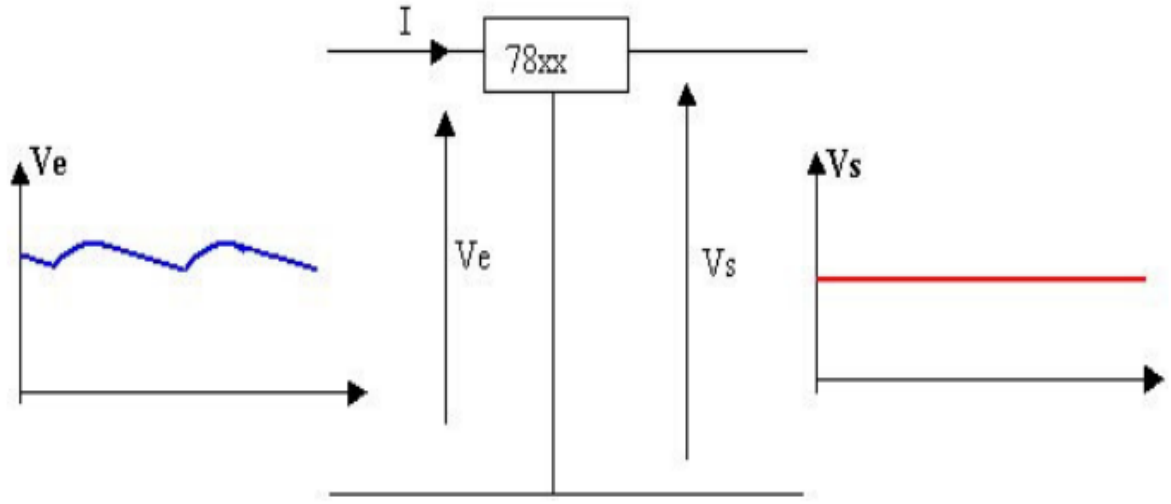


Figure 3.15. Régulation de tension

La tension à régler étant 12V, donc on choisi le régulateur 7812.

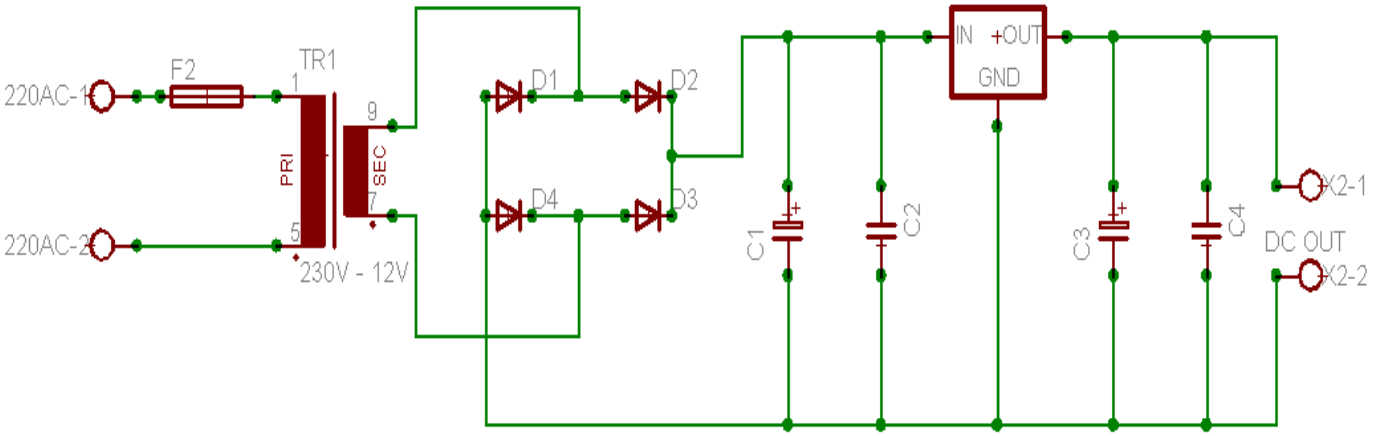


Figure 3.16. Schéma électrique de la carte d'alimentation

3.4 Partie software

3.4.1 Partie software MATLAB

Arduino va utiliser les impulsions issues du capteur optique, pour pouvoir calculer la vitesse du moteur. Ainsi, pour une régulation PID, on utilise les paramètres Ki, Kd et Kp fixés par l'utilisateur pour pouvoir atteindre la vitesse voulue.

Le contrôle de vitesse et des paramètres est fixé par une interface en machine qui est développée à partir du logiciel MATLAB.

3.4.1.1 présentations du logiciel MATLAB

Le logiciel MATLAB permet de faire des calcules matriciels, d'afficher des courbe et des données, de visualiser des résultats en 2D et 3D, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autre langages.

3.4.1.2 Interfaces graphiques

Les IHM (interfaces Homme Machine), sont appelées GUI (Graphical User Interfeces) sous MATLAB. Elles permettent à l'utilisateur, grâce à des objets graphiques (boutons, menus, cases à cocher...) d'interagir avec un programme informatique.

Le développement des interfaces graphiques peut être séparé en deux parties :

- **Gestion de la mise en place et des propriétés des objets (GUIDE)** :

C'est un constructeur d'interface graphique qui regroupe tous les outils dont le programmeur à besoin pour créer une interface graphique de façon intuitive.

- **Programmation des interactions avec les objets :**

Il est possible de programmer une interface graphique entièrement à la main sous MATLAB.

Bien que cette seconde méthode semble beaucoup moins intuitive que celle utilisant le GUIDE.

Du fait du nombre important d'objets et surtout du nombre des propriétés associées, la programmation à la main est généralement déroutante au début. Il est nécessaire de se servir de la documentation MATLAB et de savoir en tirer les bonnes informations.

Dans notre étude nous avons utilisé cette dernière méthode de programmation.

3.4.1.3 Présentation de l'interface

Il est nécessaire de mettre à la disposition de l'utilisateur une interface pratique, permettant de varier la vitesse du moteur, et aussi de régler les paramètres du contrôleur PID. Cette interface est présentée par le schéma ci-dessous :

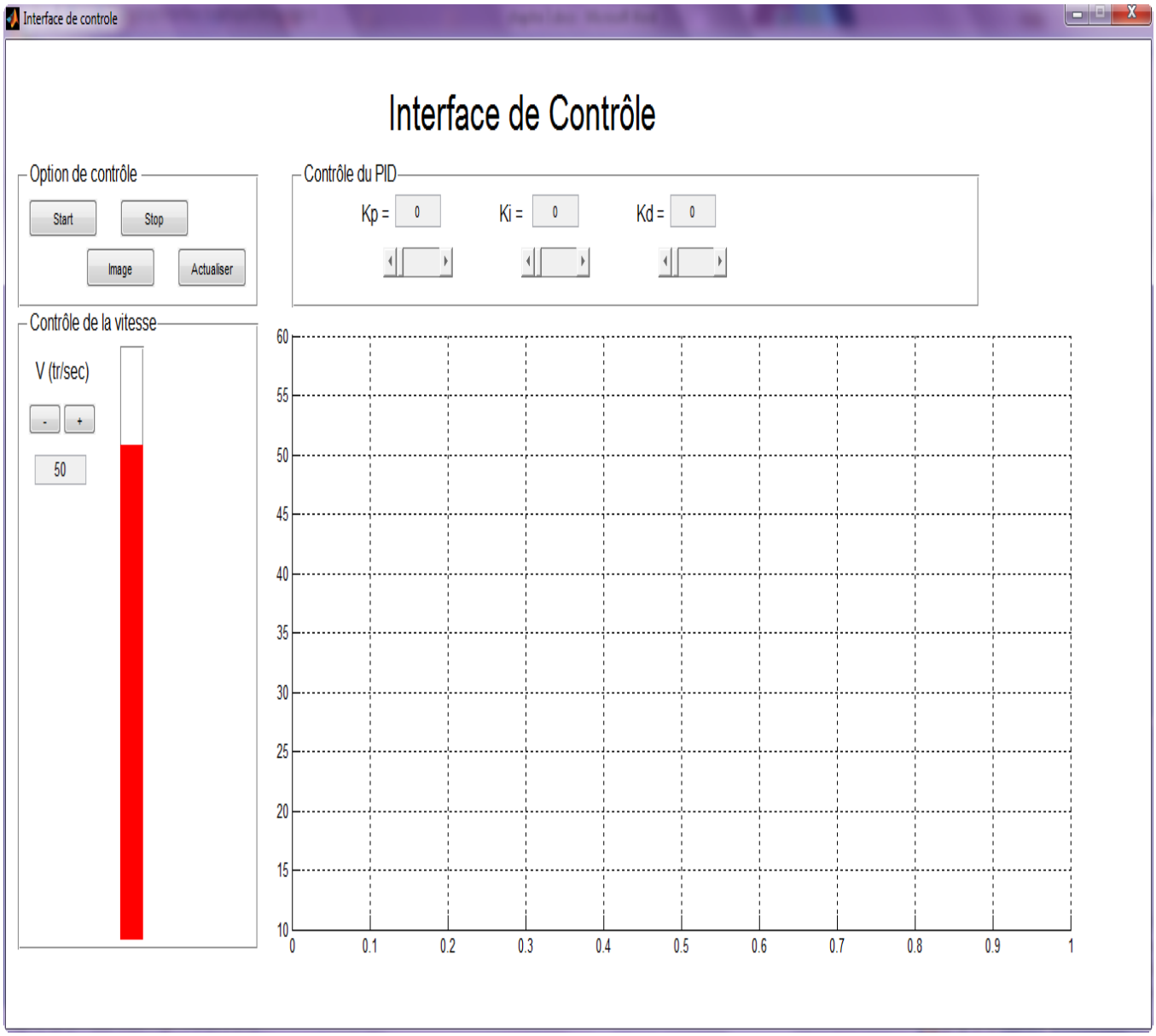


Figure 3.17. Interface de contrôle

L'interface graphique présentée ci-dessus est composée de :

- Contrôle de la vitesse : l'utilisateur peut contrôler la vitesse du moteur par les boutons + et -, ou bien écrire directement la vitesse voulue.
- Contrôle du PID : on peut régler les paramètres Kp, Ki et Kd à partir de cette interface pour pouvoir atteindre la vitesse voulue.
- Les axes : qui sont des zones de traçage de la consigne et de la vitesse réelle du moteur. on peut visualiser les graphes de la vitesse du moteur en temps réel.
- Option de contrôle : est constituée de 4 boutons : 2 boutons pour le démarrage et de l'arrêt de l'exécution, un bouton pour actualiser la fenêtre et le dernier bouton pour prendre une image du graphe.

3.4.2 Partie software Arduino

Nous avons utilisé le logiciel de programmation des modules Arduino. La programmation à largement été simplifiée grâce à ses nombreuses bibliothèques.

Avant d'aborder la programmation du Arduino, Il est nécessaire de définit l'élément qui communique la partie hardware et la partie software.

La carte Arduino UNO dispose de toute une série de facilités pour communiquer avec un ordinateur, une autre carte Arduino, ou avec d'autres microcontrôleurs. L'ATmega328 dispose d'une UART (Universal Asynchrones Receiver Transmitter) pour communication série à 5V et qui est disponible sur les broches 0(RX) et 1(TX). Un circuit intégré ATmega8U2 sur la carte assure la connexion entre cette communication série vers le port USB de l'ordinateur et apparaît comme un port COM virtuel pour les logiciels de l'ordinateur.

Le logiciel Arduino inclut une fenêtre terminal série (ou moniteur série) sur l'ordinateur et qui permet d'envoyer des textes depuis et vers la carte Arduino. Les LEDs RX et TX sur la carte clignote lorsque les données sont transmises via le circuit intégré USB vers série et la connexion USB vers l'ordinateur (mais pas pour les communications série sur la broche 0 et la broche 1).

L'organigramme ci-dessous représente les étapes de la programmation de notre projet sur le logiciel Arduino.

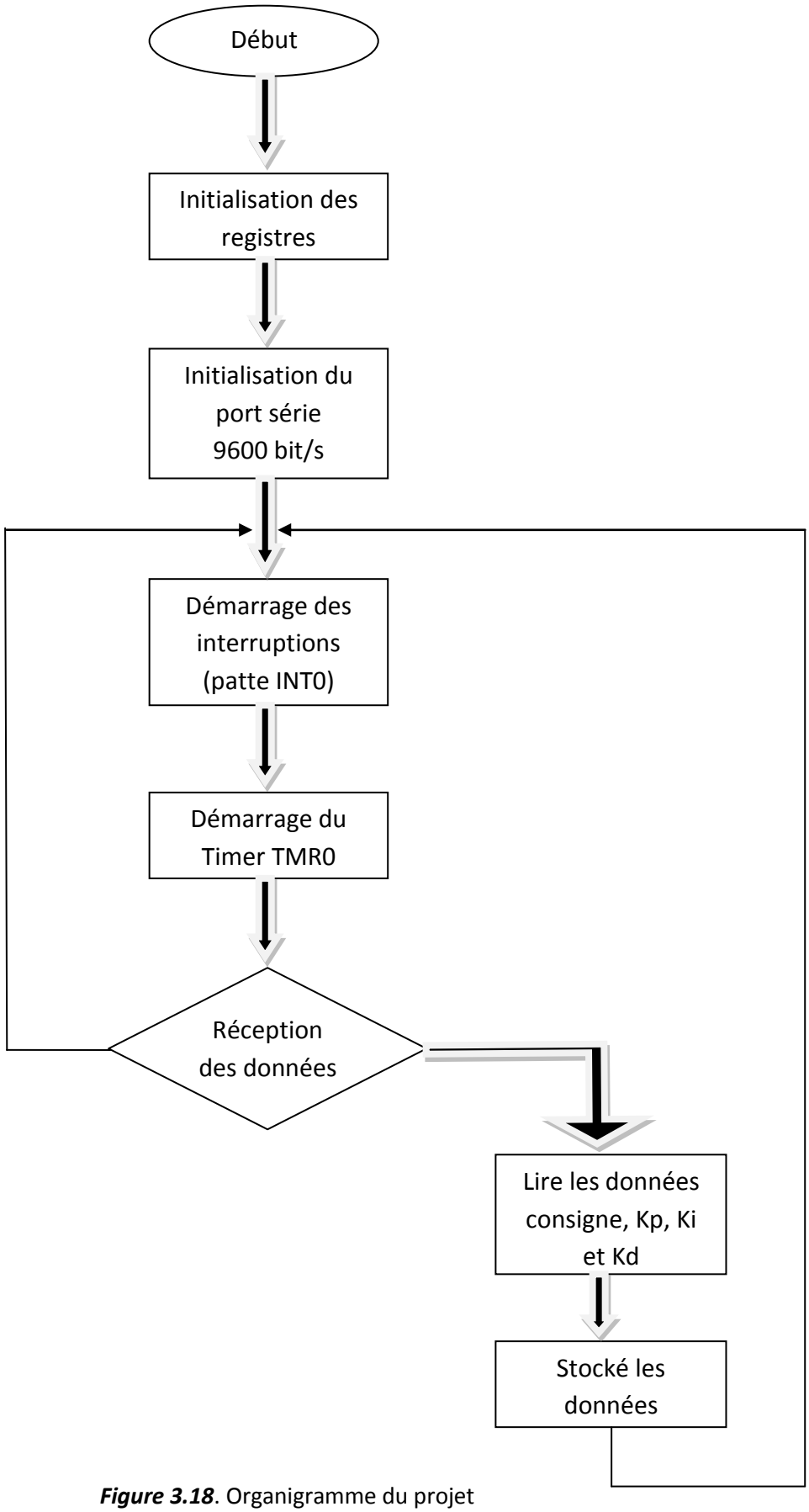


Figure 3.18. Organigramme du projet

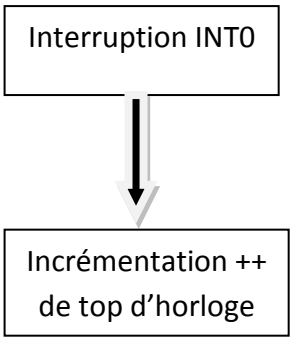


Figure 3.19. Interruption du top d'horloge du capteur

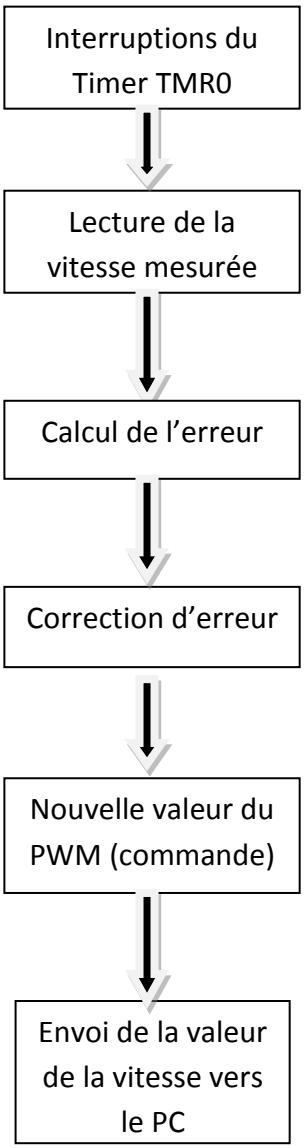


Figure 3.20. Interruptions du Timer

3.5 Schéma global du projet

Après la description de chaque élément de notre projet, Nous allons présenter maintenant Le schéma global de notre régulateur PID.

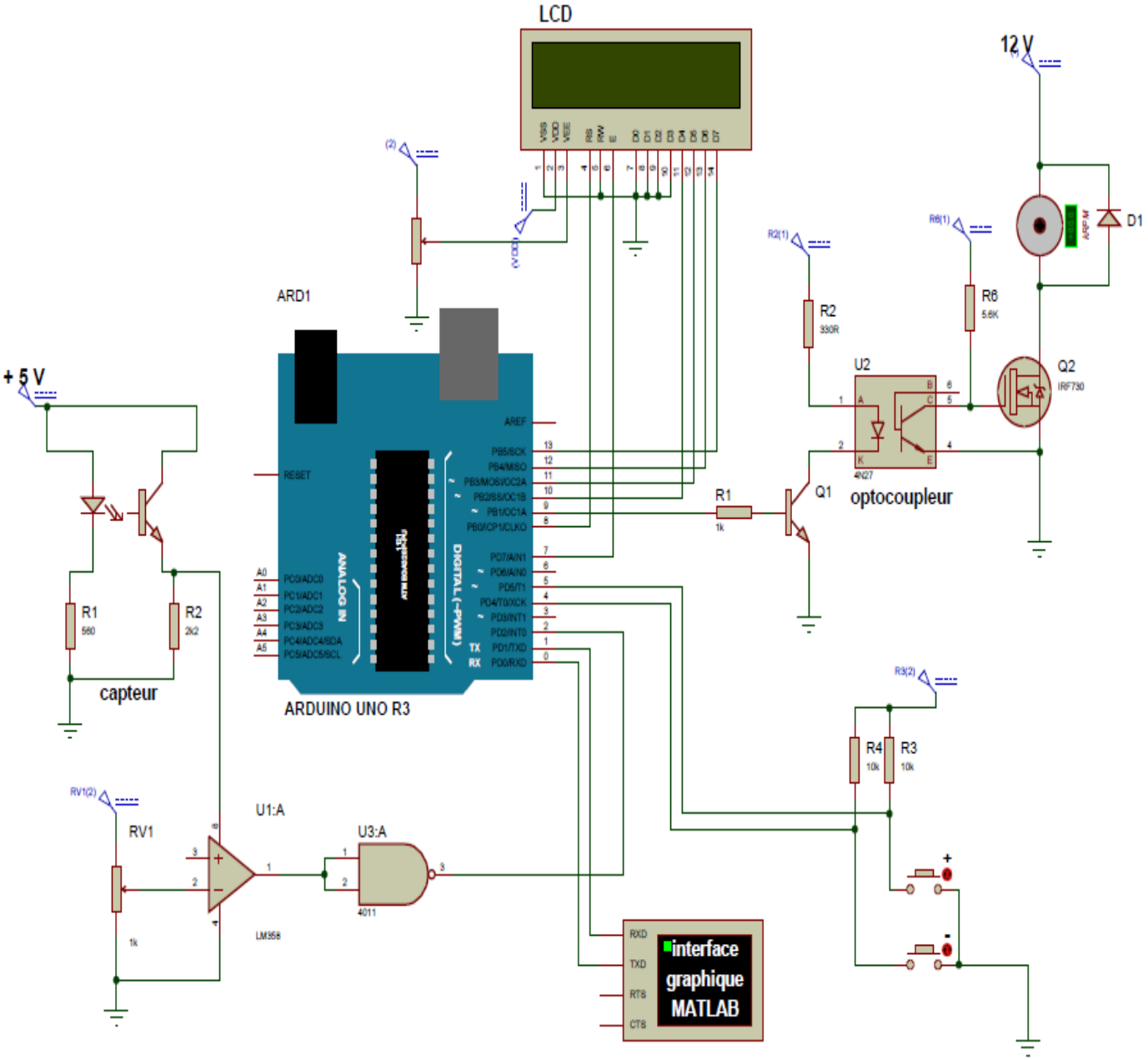


Figure 3.21. Schéma global du projet

3.6 Conclusion

Dans ce chapitre, nous avons bien détaillé notre projet, qui consiste à une étude de conception des différents cartes électroniques, et aussi d'une partie de programmation sur logiciel ARDUINO et MATLAB. Le chapitre suivant sera donc consacré à la réalisation de ces cartes électronique.

4.1 Introduction

Après avoir présenté dans le chapitre précédent la conception de différentes cartes électroniques. Nous allons aborder dans ce chapitre la réalisation pratique de ces cartes. Ainsi que le réglage des paramètres Kp, Ki et Kd du correcteur PID.

4.2 Réalisation du PCB (printed circuit board)

Nous avons réalisé nos circuits imprimés avec le logiciel EAGLE (Easily Applicable Graphical Layout Editor).

C'est un logiciel de conception assistée par ordinateur de circuits imprimés. Il comprend un éditeur de schémas, un logiciel de routage de circuit imprimé avec une fonction d'autoroutage, et un éditeur de bibliothèque. Le logiciel est fourni avec une série de bibliothèques de composants de bases.

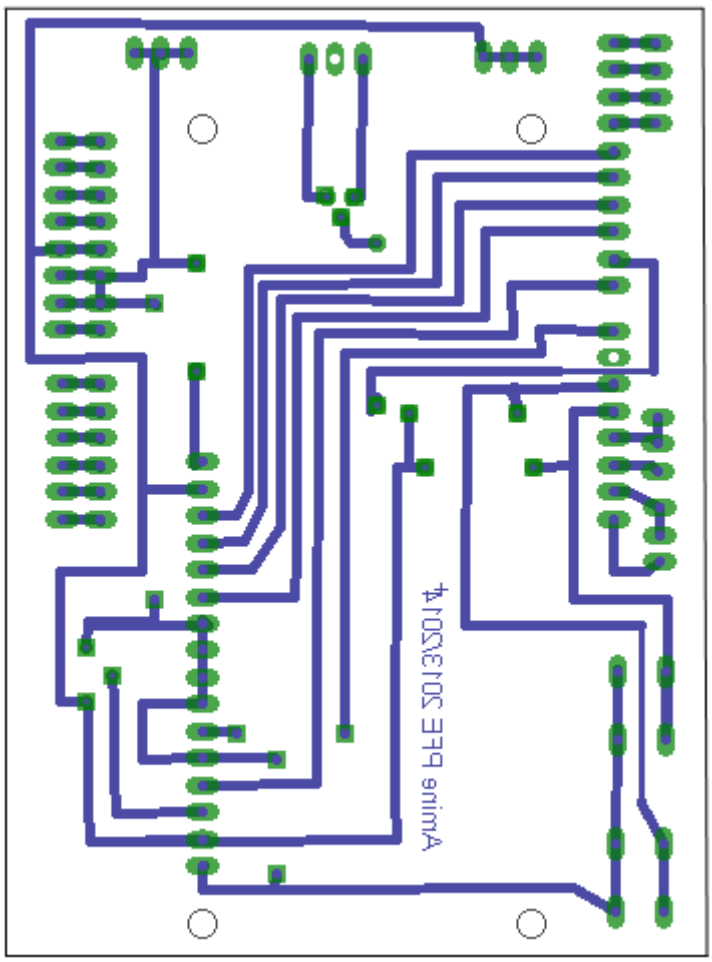


Figure 4.1.Circuit imprimé de la carte additionnelle pour l'interface Arduino

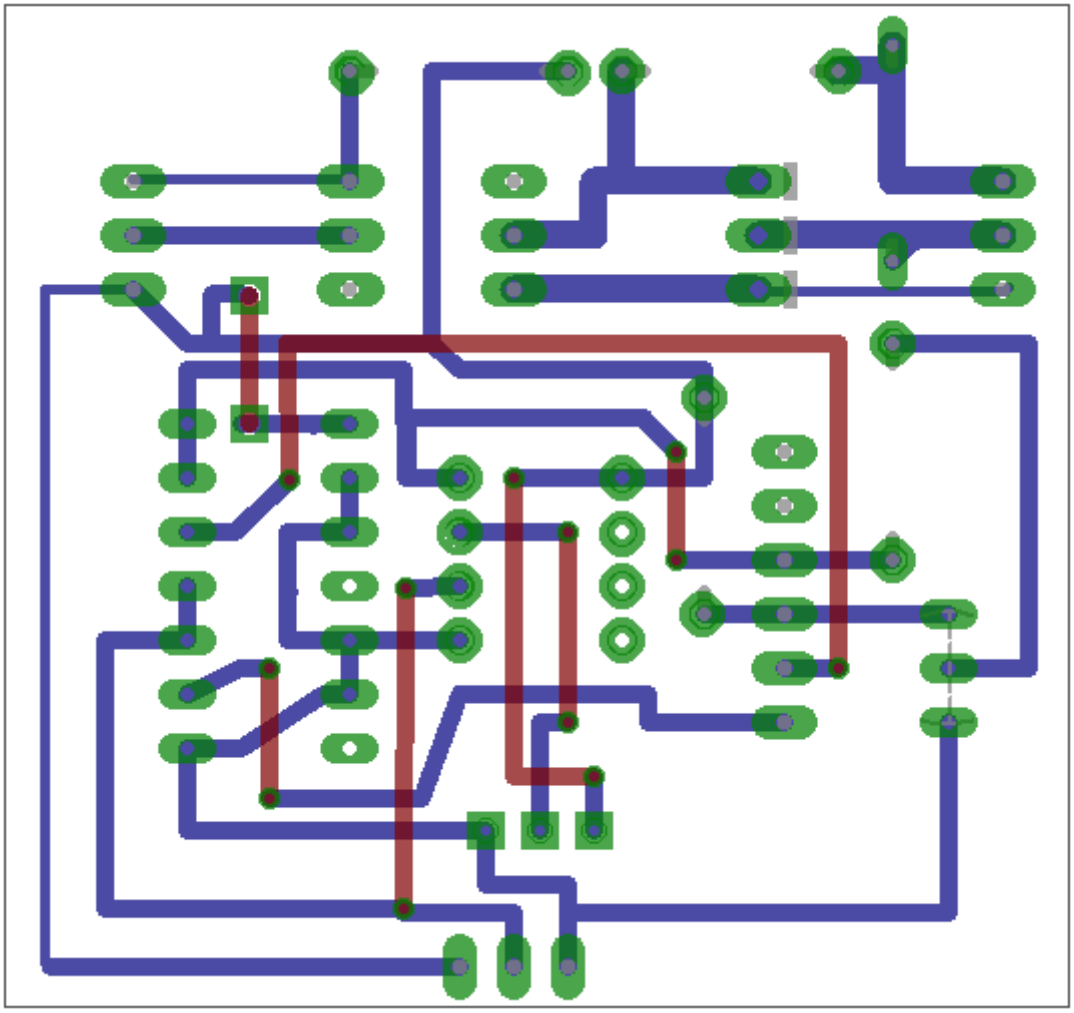


Figure 4.2.Circuit imprimé de puissance et de mise en forme

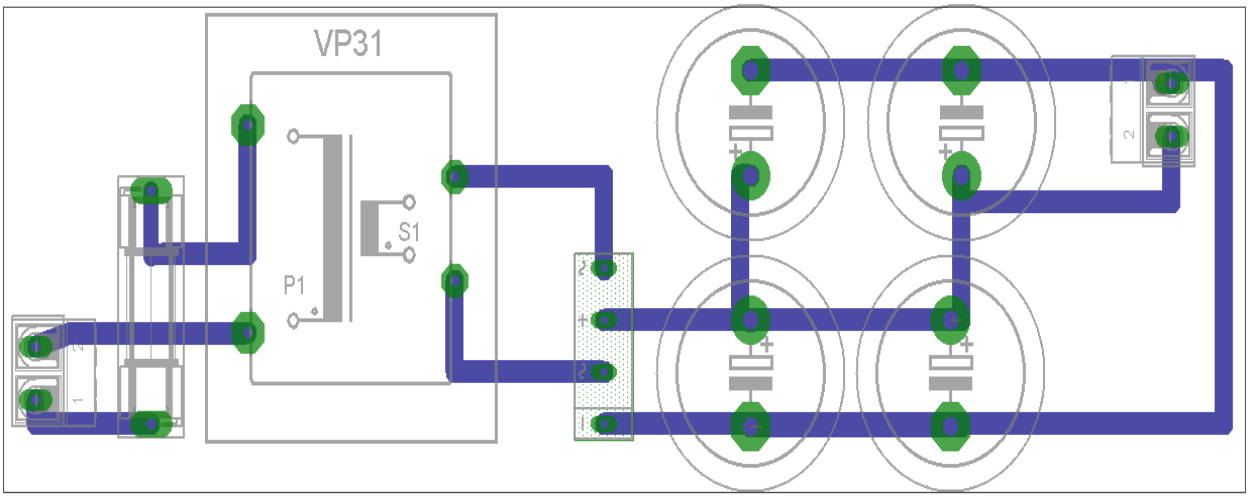


Figure 4.3.Circuit imprimé de l'alimentation stabilisé de 12V

4.3 Conception des cartes

Nous avons implémenté puis soudé les composants sur nos circuits imprimés

4.3.1 Carte additionnelle pour l'interface Arduino

Il est possible de spécialiser la carte Arduino en l'associant avec des circuits additionnels que l'on peut fabriquer soi-même. Lorsqu'ils sont branchés directement sur la carte, ces circuits s'appellent des « shield » ou cartes d'extension. Ces circuits spécialisés apportent au système des fonctionnalités diverses et étendues.

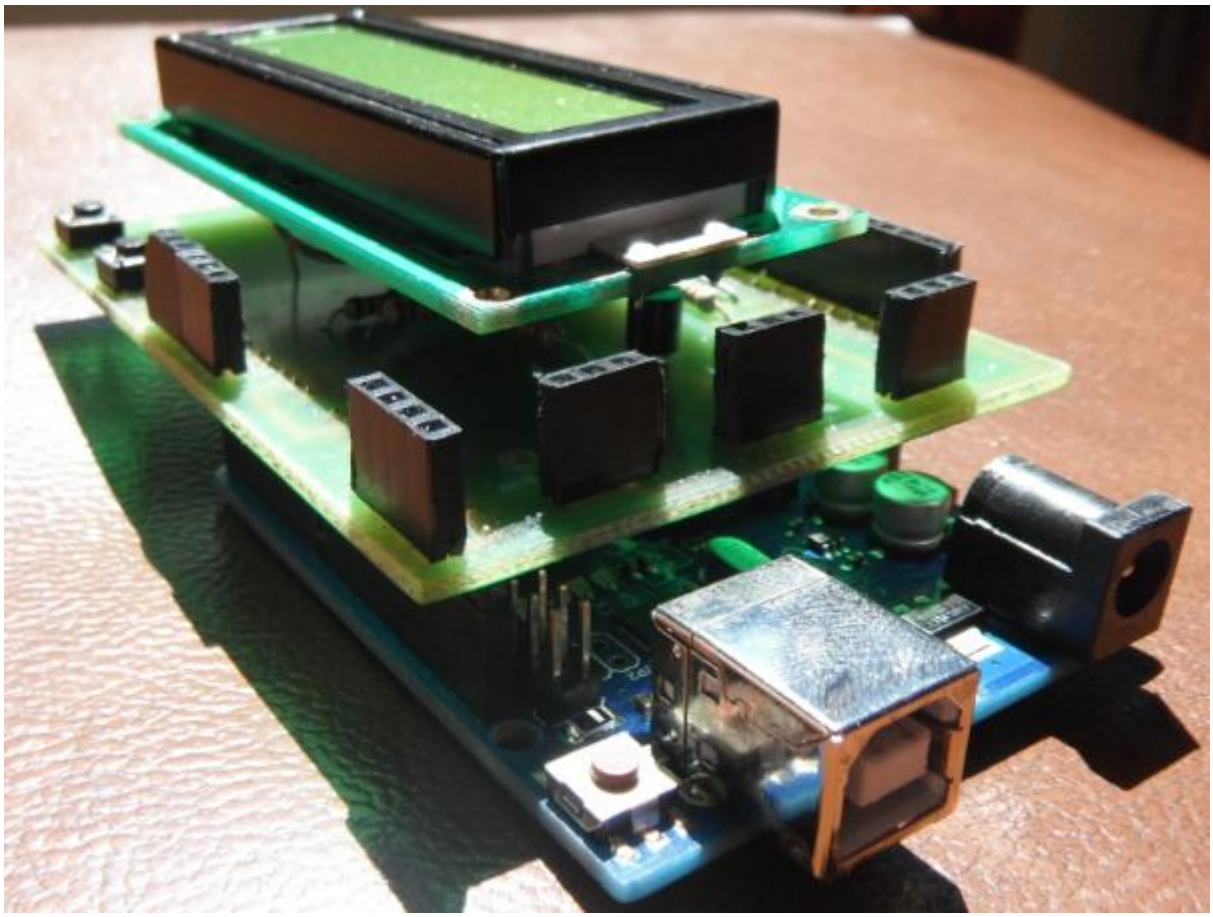


Figure 4.4.carte de commande et d'affichage « shield »

Ce schéma présente une carte additionnelle pour l'interface Arduino UNO. Elle est constituée de :

- Deux boutons poussoirs : l'un pour augmenter la vitesse et l'autre pour diminuer la vitesse. Donc on peut varier la vitesse du moteur a partir de ces boutons.
- Un afficheur LCD : pour afficher la consigne et la vitesse réelle du moteur.

On peut placée cette carte sur n'importe quelle Arduino UNO, et commandé n'importe quelle moteur, en changeant juste la carte de puissance.

4.3.2 Carte de puissance et de mise en forme

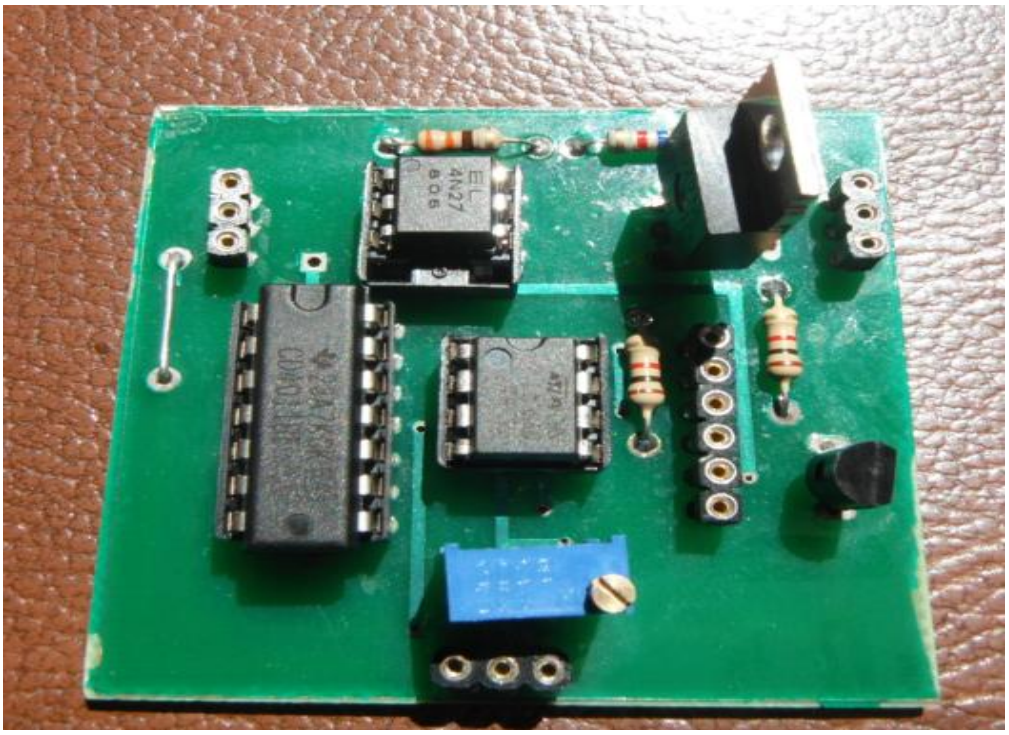


Figure 4.5.carte de puissance et de mise en forme

On a bien détaillé dans le chapitre précédent la conception des circuits de puissance et de mise en forme.

4.3.3 Carte d'alimentation

Figure 4.6.carte d'alimentation de 12V

4.3.4 Moteur à courant continu et codeur optique :

Notre moteur à courant continu est alimenté par une tension continue de 12V, il consomme en fonctionnement nominal un courant de 1 A.

Figure 4.7.notre moteur à courant continu

Notre moteur contient un codeur optique qui comporte 3 fils.

- deux pour l'alimentation.
- Un pour le signal de sortie.

Ce dernier donne des impulsions suivant la rotation d'un disque gradué de 72 points. le passage d'un point à un autre nous donne une impulsion. Le nombre de ces impulsions permet de déduire la vitesse de ce moteur.

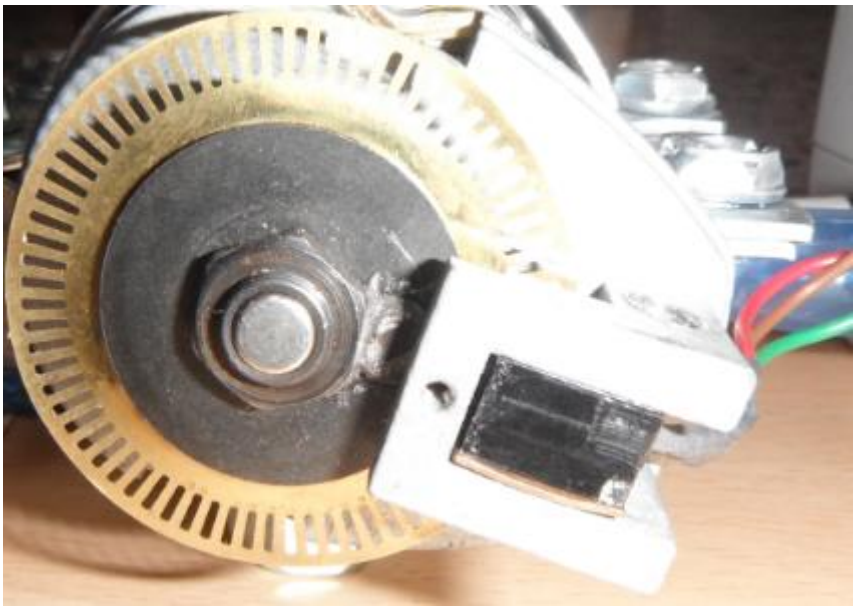


Figure 4.8.codeur optique de notre moteur

Figure 4.9.le montage final

4.4 Tests et discussions

4.4.1 Méthodes de réglages des paramètres du régulateur PID

On rappelle que le rôle d'un régulateur est de maintenir la grandeur régulée à une valeur de la consigne malgré la présence des perturbations dans le fonctionnement en régulateur ou suivre la variation d'une consigne dans le fonctionnement en asservissement. Au moyen du choix des actions et de leurs paramètres. Il est possible d'obtenir un comportement désiré en boucle fermée, caractérisant les performances du système.

De manière **qualitative**, les critères à satisfaire sont les suivants :

- Les effets de perturbations doivent être minimisés ou encore mieux, ils doivent être effacés complètement et ce, le plus rapidement possible.
- Les changements de consigne doivent être suivis rapidement et avec une bonne précision.

De manière **quantitative**, il s'agit de proposer les actions P, I, D su régulateur et de fixer les valeurs à donner aux paramètres Kp, Ki, Kd répondant le mieux possible aux spécifications d'un cahier des charges.

Ce problème est connu par la synthèse des systèmes bouclés. Les méthodes de synthèse sont très nombreuses et une classification rigoureuse n'est pas une tâche facile. Néanmoins, on peut distinguer dans les deux types de méthodes :

- Des méthodes dites empiriques ne nécessitant pas une connaissance parfaite du modèle du procédé à commander. Les paramètres du régulateur seront calculés à partir des observations expérimentales sur le procédé. L'intérêt majeur de ces méthodes réside dans leur simplicité. Elles sont largement utilisées dans le domaine industriel et elles sont dans la plus part des cas suffisantes mais ne permettent pas un réglage fin.
- Des méthodes basées sur la connaissance du modèle du système sous forme de fonction de transfert par exemple. Les actions du régulateur seront calculées de façon à obtenir la fonction de transfert souhaitée en boucle ouverte ou en boucle fermée.

Dans notre cas nous avons utilisé la première méthode qui consiste à régler les paramètres du régulateur à partir des observations expérimentales sur le MATLAB.

4.4.2 Tests et réglages des paramètres du régulateur PID

Pour déterminer les paramètres du correcteur PID et connaitre leurs rôles. Nous allons utiliser une méthode d'approximations successives. Cette dernière consiste à faire des testes sur chacune des actions du régulateur indépendamment des autres actions.

Premièrement, nous devons fixer la consigne pour faire le test. Pour cela nous avons choisi une vitesse de 20 tours/seconde.

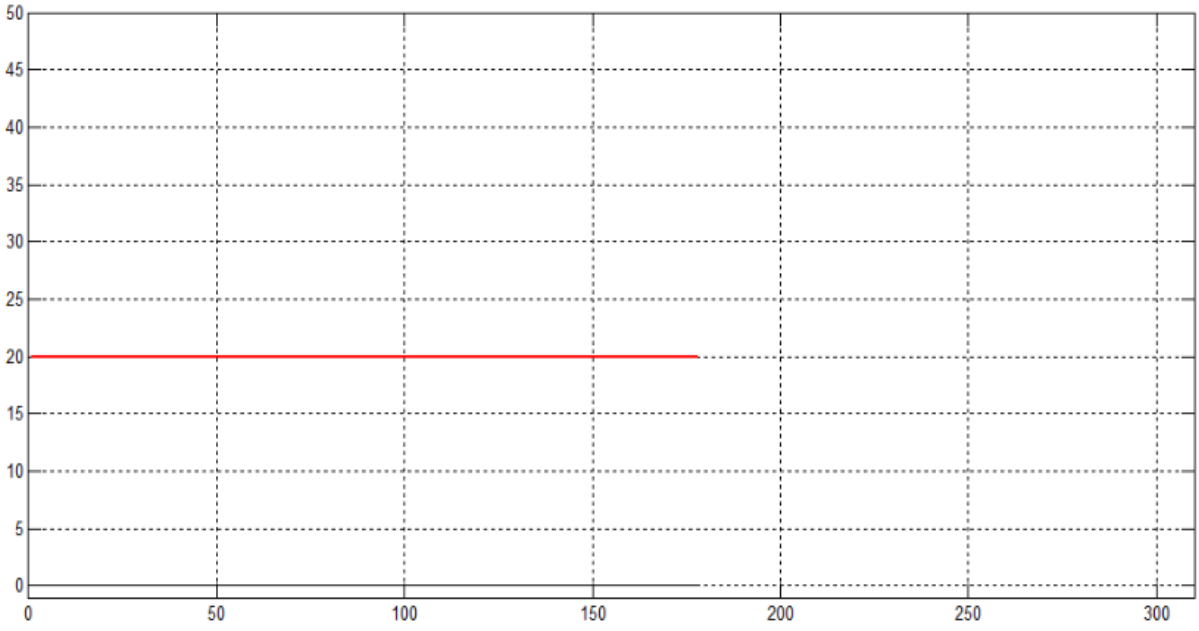


Figure 4.10.réponse du moteur en fonction du temps avec Kp=Ki=Kd=0

- Nous avons mis en place **un régulateur proportionnel** (donc les coefficients Ki et Kd sont nuls).

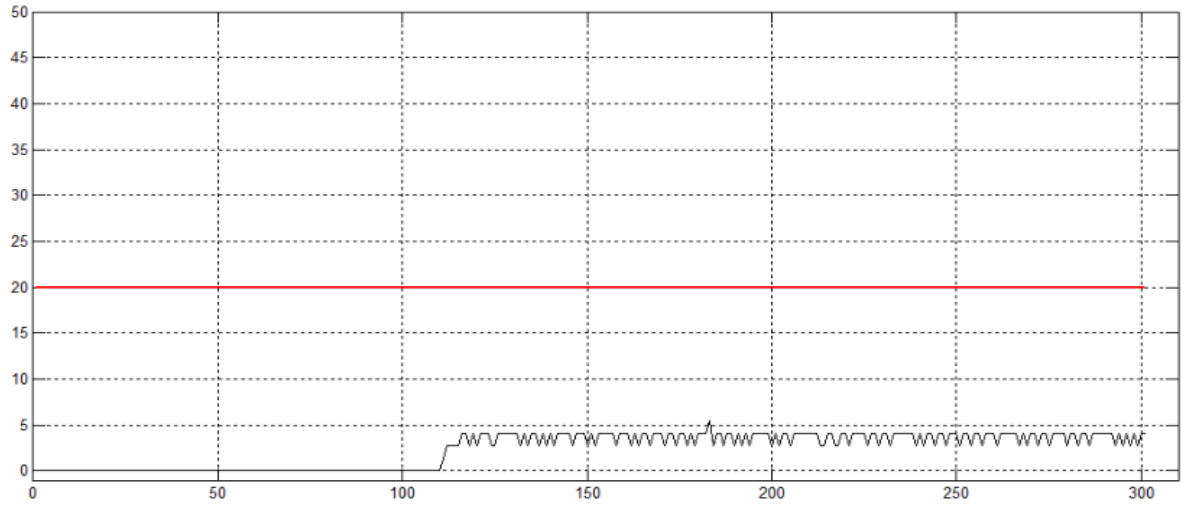


Figure 4.11.réponse du moteur en fonction du temps avec Kp=2 et Ki=Kd=0

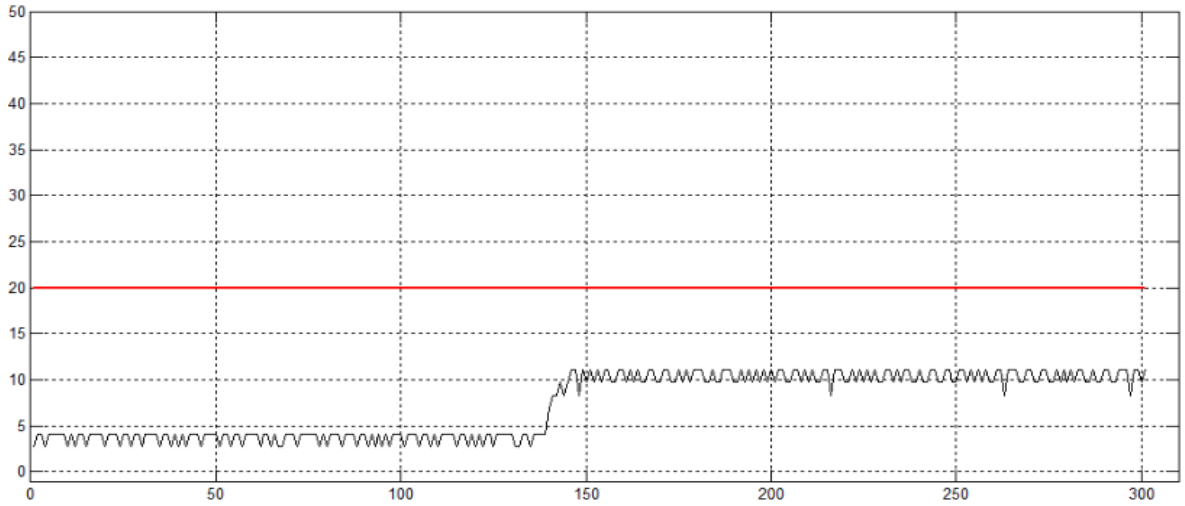


Figure 4.12.réponse du moteur en fonction du temps avec Kp=5 et Ki=Kd=0

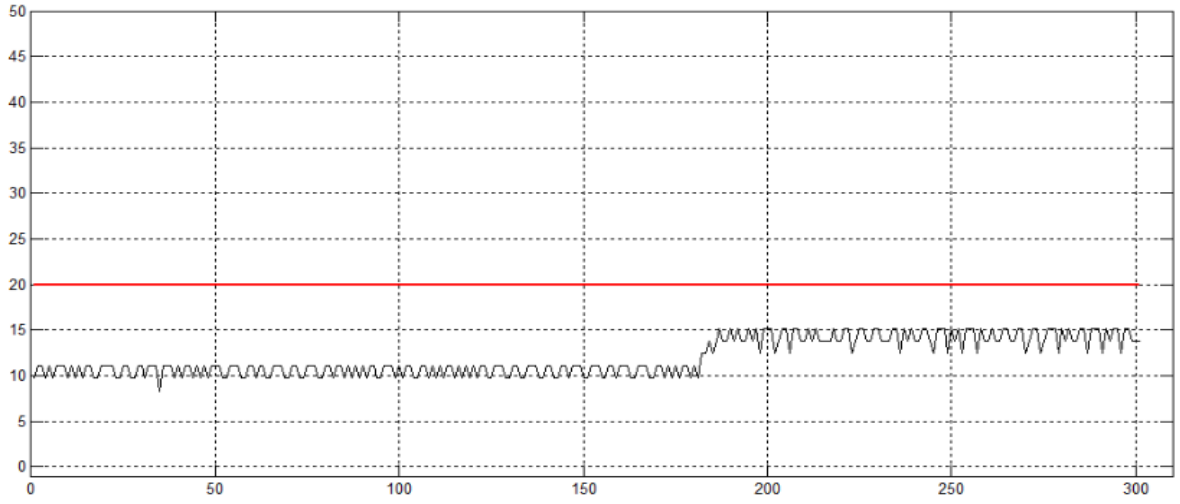


Figure 4.13.réponse du moteur en fonction du temps avec Kp=10 et Ki=Kd=0

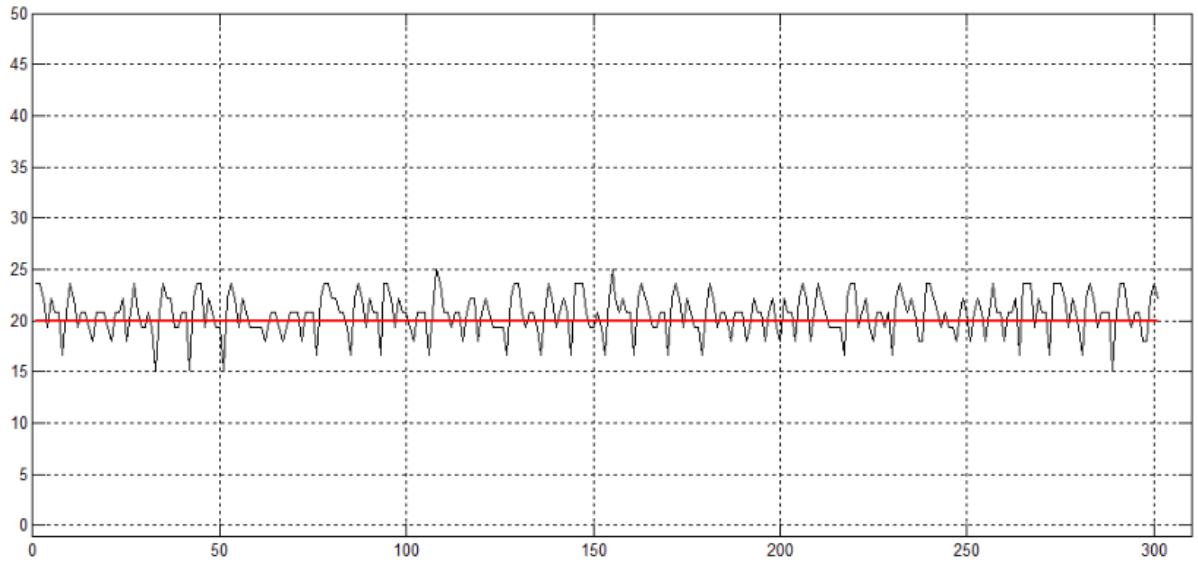


Figure 4.14.réponse du moteur en fonction du temps avec Kp=100 et Ki=Kd=0

On remarque bien que quand Kp augmente, la réponse se rapproche de plus en plus à la consigne voulu. Mais quand Kp est trop grand, la réponse oscille fortement autour de la consigne. C'est-à-dire, il faut choisir un Kp qui permet au système de se rapprocher très vite de la consigne tout en faisant attention de garder la stabilité du système.

- Nous avons ajouté un intégrateur pour obtenir **un régulateur PI** (le coefficient Kd est nul).

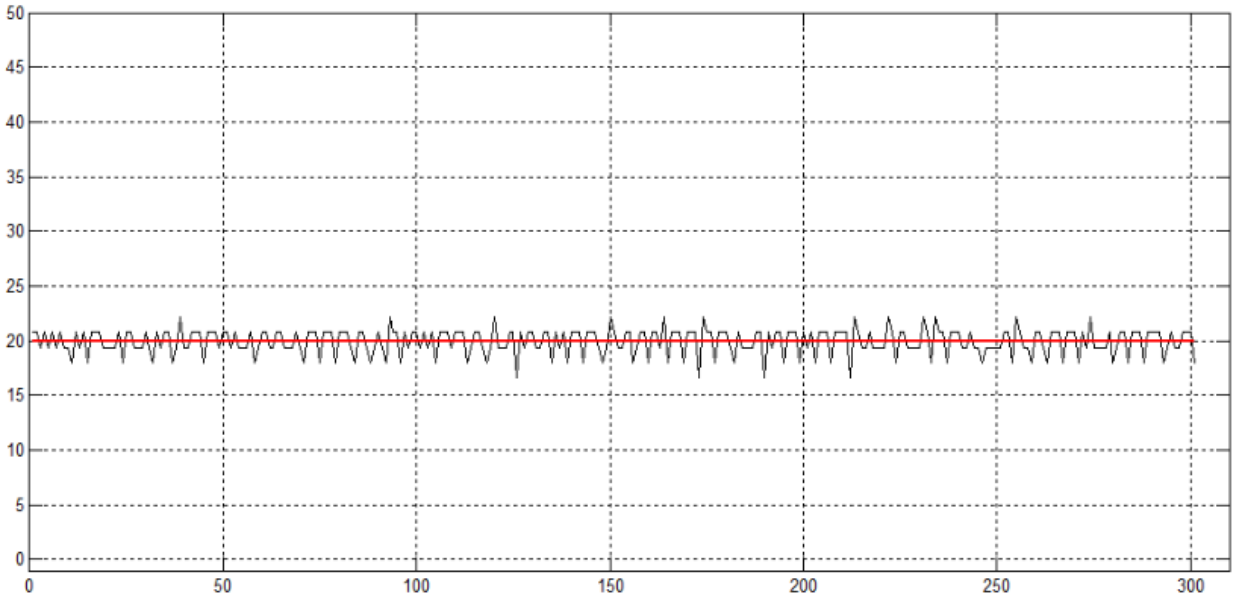


Figure 4.15.réponse du moteur en fonction du temps avec Kp=5, Ki=0.1 et Kd=0

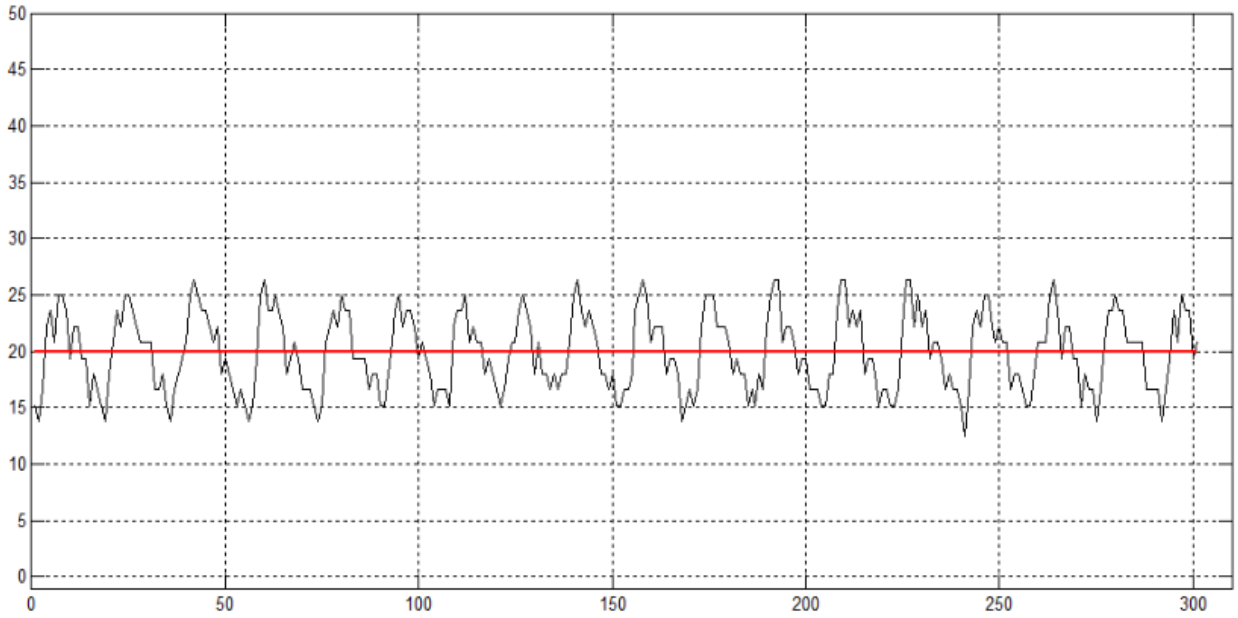


Figure 4.16.réponse du moteur en fonction du temps avec Kp=5, Ki=20 et Kd=0

Plus on augmente le coefficient d'intégration Ki, et plus le système répond vite. Mais le système devient de plus en plus instable. Ce coefficient permet d'annuler l'erreur finale du système.

- Pour mettre en place **un asservissement PID** complet, nous avons rajouté un dérivateur.

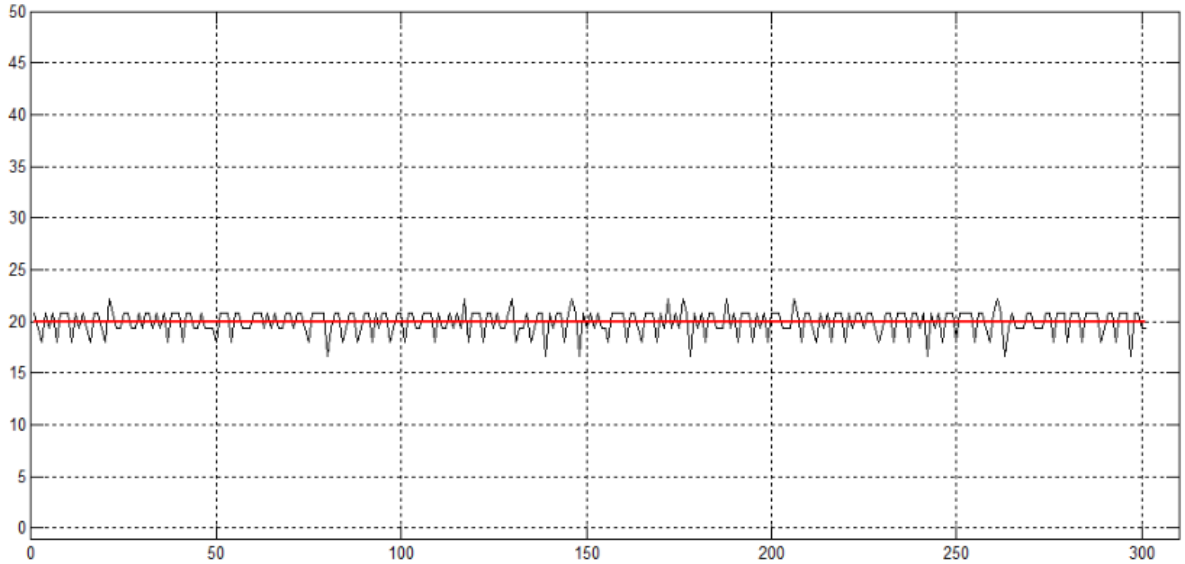


Figure 4.17.réponse du moteur en fonction du temps avec Kp=5, Ki=0.5 et Kd=1

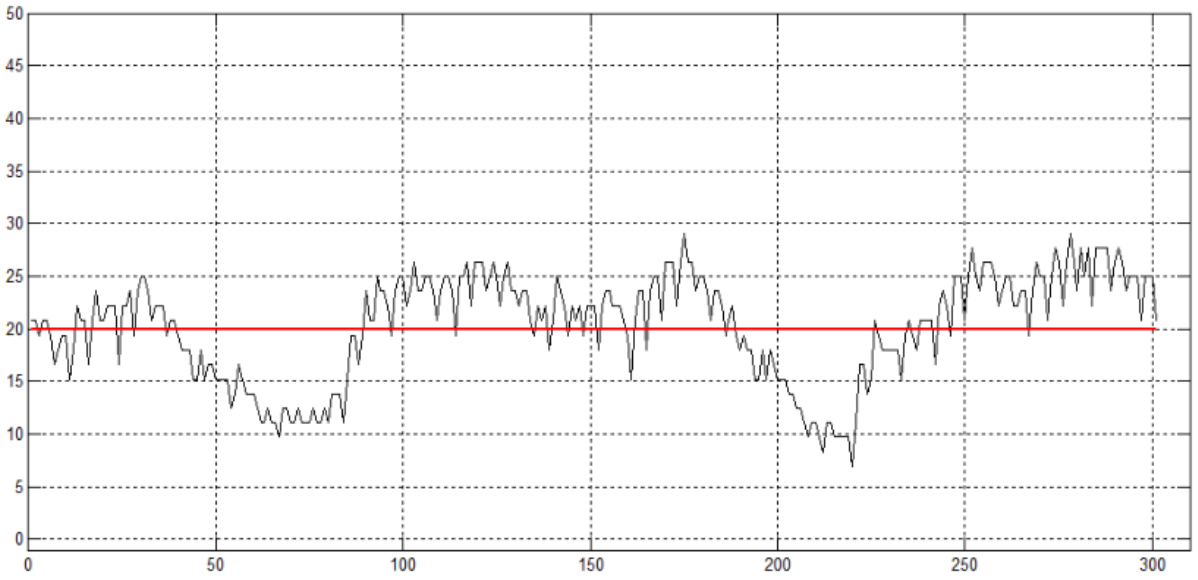


Figure 4.18.réponse du moteur en fonction du temps avec Kp=5, Ki=0.5 et Kd=100

Le dérivateur permet de rendre le système plus stable. Il permet donc de diminuer les oscillations. Mais quand le coefficient Kd est trop grand le temps de réponse augmente.

- Les coefficients que nous avons choisis pour notre système sont : Kp = 5, Ki = 0.5 et Kd=1

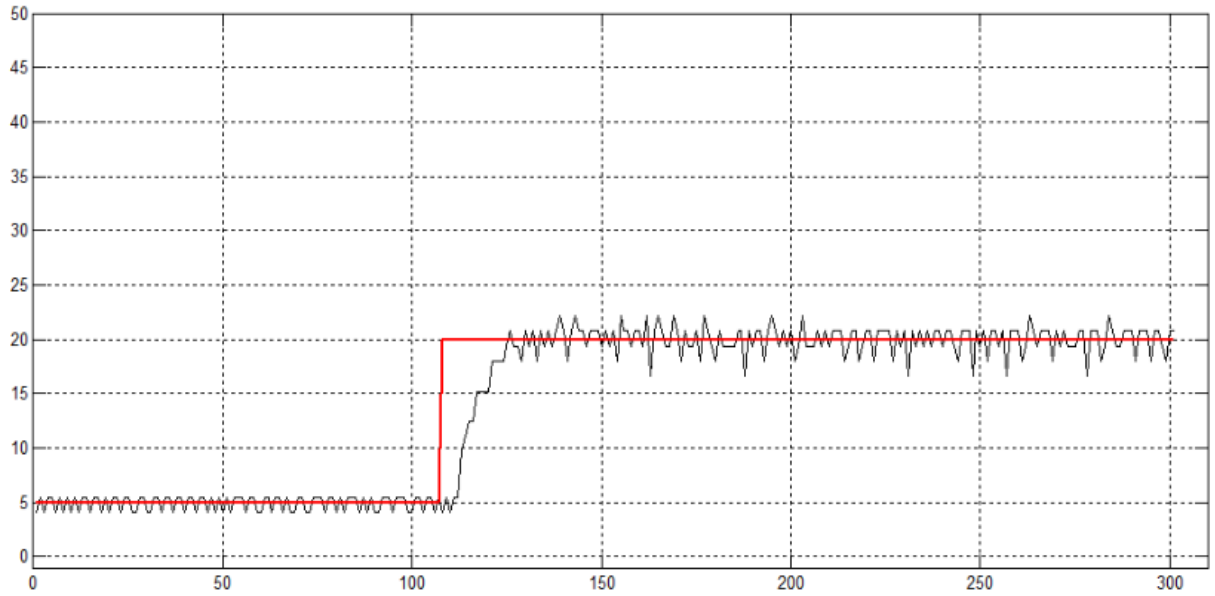


Figure 4.19.réponse du moteur après un changement de consigne de 5tr/s à 20tr/s avec les coefficients Kp=5, Ki=0.5 et Kd=100

4.5 Conclusion

Chacun des coefficients du PID à un rôle à jouer sur la réponse à une consigne. Il faut juste trouver un compromis qui répond au cahier de charge.

La réalisation de notre montage a été faite en totalité. Le montage fonctionne correctement et répond à la consigne désirée.

Le Logiciel Arduino

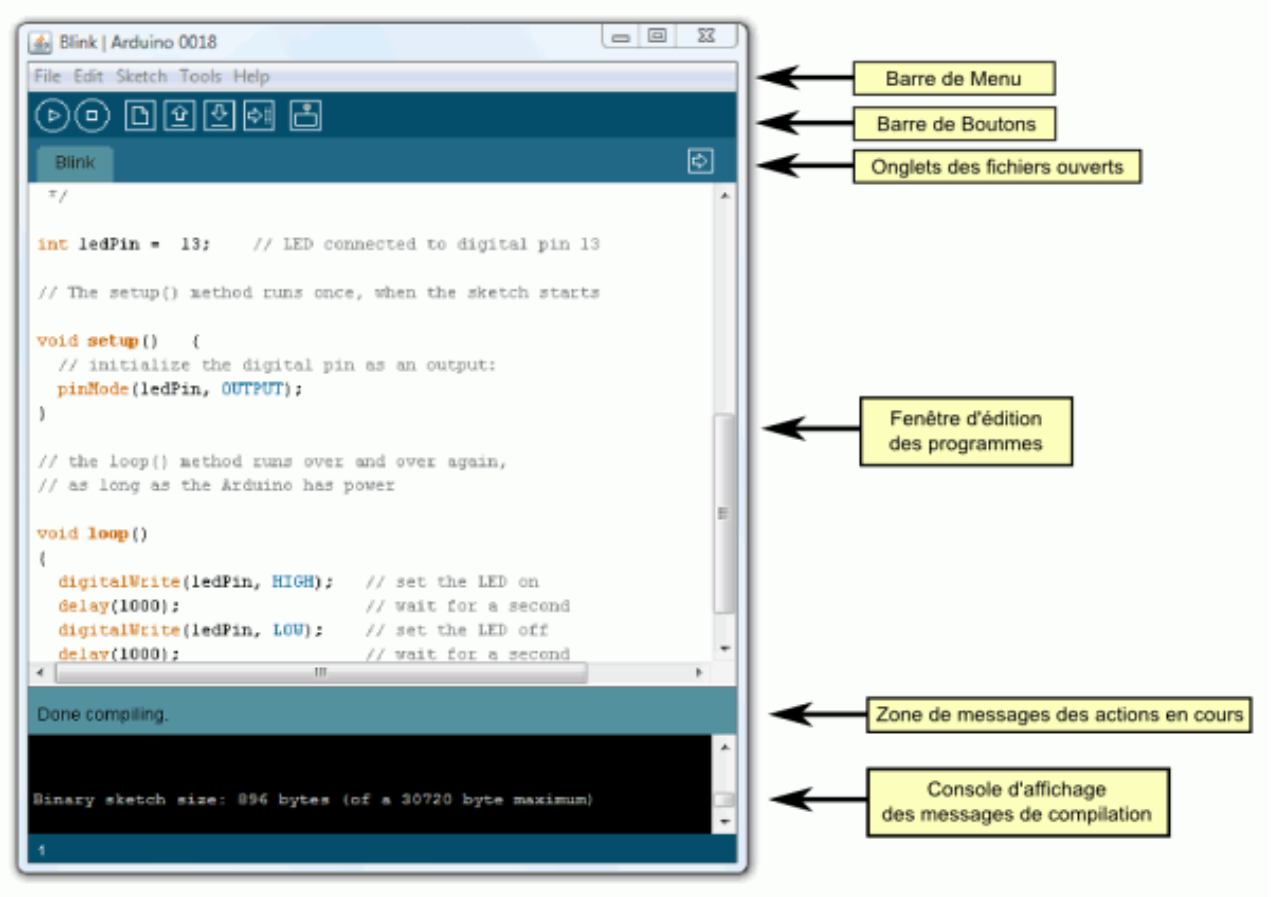
1. Description

Le logiciel Arduino a pour fonctions principales :

- de pouvoir écrire et compiler des programmes pour la carte Arduino
- de se connecter avec la carte Arduino pour y transférer les programmes
- de communiquer avec la carte Arduino

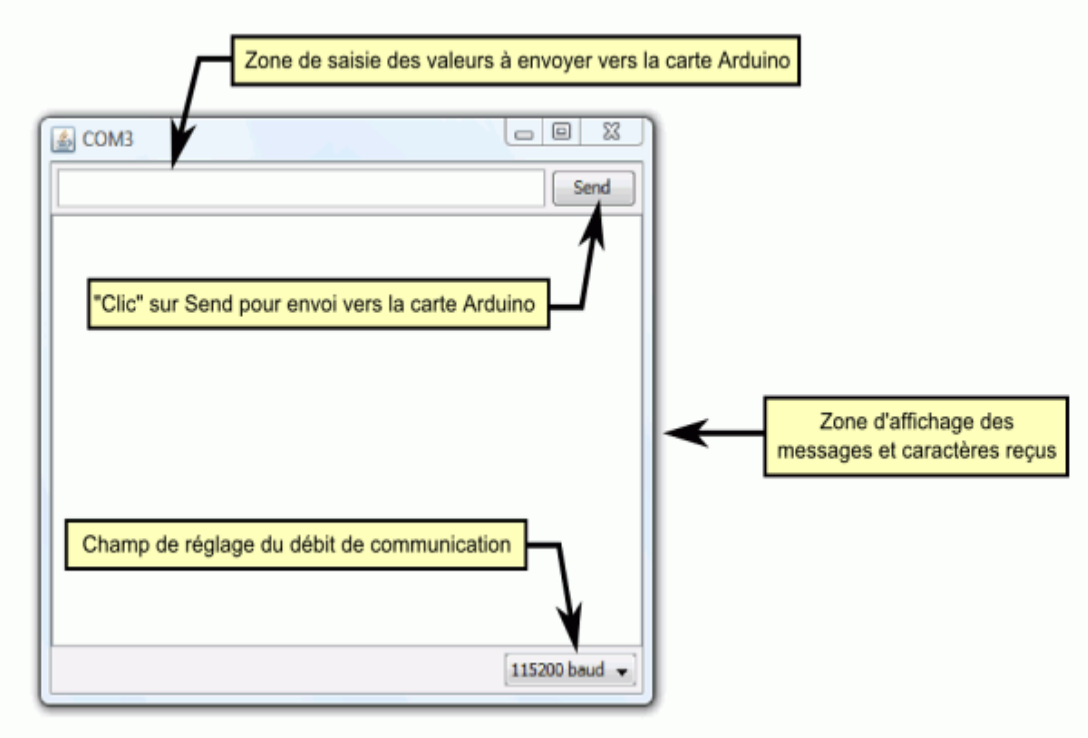
Cet espace de développement intégré (EDI) dédié au langage Arduino et à la programmation des cartes Arduino comporte :

- une **BARRE DE MENUS** comme pour tout logiciel une interface graphique (GUI),
- une **BARRE DE BOUTONS** qui donne un accès direct aux fonctions essentielles du logiciel et fait toute sa simplicité d'utilisation,
- un **EDITEUR** (à coloration syntaxique) pour écrire le code de vos programmes, avec onglets de navigation,
- une **ZONE DE MESSAGES** qui affiche indique l'état des actions en cours,
- une **CONSOLE TEXTE** qui affiche les messages concernant le résultat de la compilation du programme



Le logiciel Arduino intègre également :

- un **TERMINAL SERIE** (fenêtre séparée) qui permet d'afficher des messages textes reçus de la carte Arduino et d'envoyer des caractères vers la carte Arduino. Cette fonctionnalité permet une mise au point facilitée des programmes, permettant d'afficher sur l'ordinateur l'état de variables, de résultats de calculs ou de conversions analogique-numérique : un élément essentiel pour améliorer, tester et corriger ses programmes.



2. Principe général d'utilisation

Le code écrit avec le logiciel Arduino est appelé un programme (ou une séquence - sketch en anglais) :

- Ces programmes sont écrits dans **l'éditeur de texte**. Celui-ci a les fonctionnalités usuelles de copier/coller et de rechercher/remplacer le texte.
- la **zone de messages** donne l'état de l'opération en cours lors des sauvegardes, des exportation et affiche également les erreurs.
- La **console texte** affiche les messages produit par le logiciel Arduino incluant des messages d'erreur détaillés et autres informations utiles.
- la **barre de boutons** vous permet de vérifier la syntaxe et de transférer les programmes, créer, ouvrir et sauver votre code, et ouvrir le moniteur série.
- la barre des menus vous permet d'accèder à toutes les fonctionnalités du logiciel Arduino.

3. Description de la barre des boutons

Vérifier/compiler : Vérifie le code à la recherche d'erreur.

Stop : Stoppe le moniteur série ou les autres boutons activés.

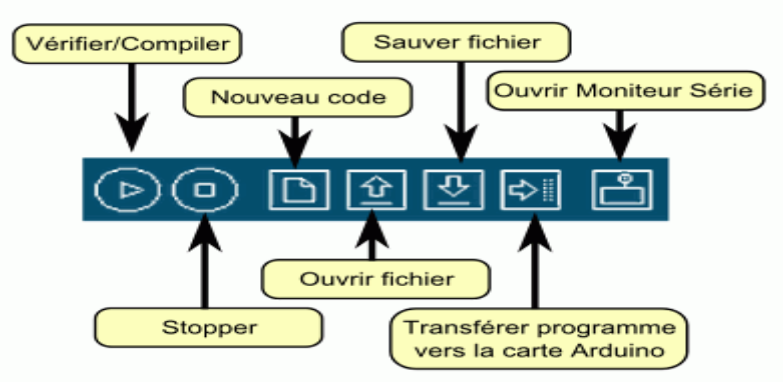
Nouveau : Crée un nouveau code (ouvre une fenêtre d'édition vide).

Ouvrir : Ouvre la liste de tous les programmes dans votre "livre de programmes". Cliquer sur l'un des programmes l'ouvre dans la fenêtre courante.

Sauver : Enregistre votre programme.

Transférer vers la carte : Compile votre code et le transfère vers la carte Arduino. Voir ci-dessous "Transférer les programmes" pour les détails.

Moniteur Série : Ouvre la fenêtre du moniteur (ou terminal) série.

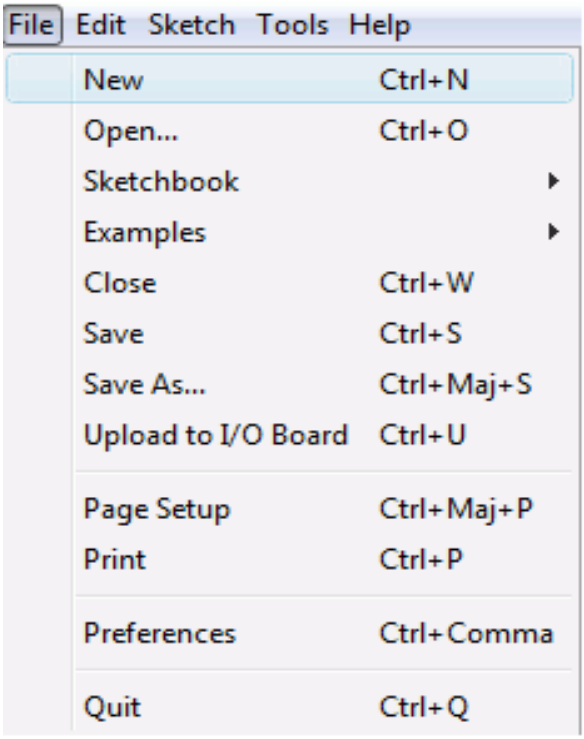


4. Description des menus

Des commandes complémentaires sont disponibles dans cinq menus :

- **File** (Fichier),
- **Edit** (Editer),
- **Sketch** (Programme ou Séquence),
- **Tools** (Outils),
- **Help** (Aide),

Le menu est sensible au contexte ce qui signifie que seulement les items correspondant au travail en cours sont disponibles.



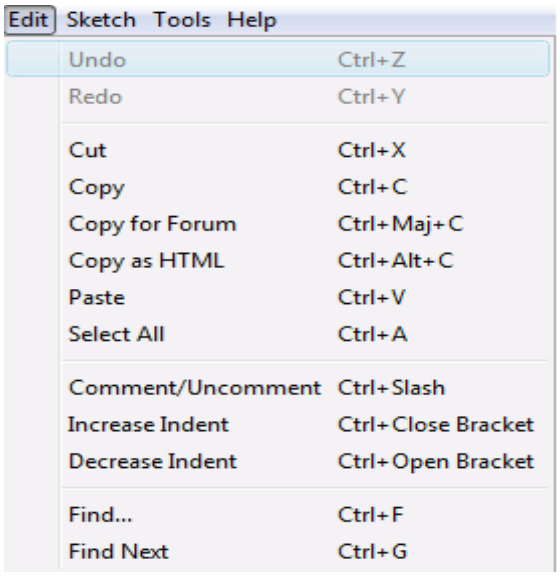
Menu **File** (Fichier) :

- Propose toutes les fonctionnalités usuelles pour gérer les fichiers,
- Sketchbook (Programme) :

  Fonctionnalité vous permettant d'avoir accès directement à tous vos programmes dans votre répertoire de travail. Voir ci-dessous pour les détails.

- Examples (Exemples) :

  Cet item vous propose un menu déroulant vers toute une série de programmes d'exemples disponibles.



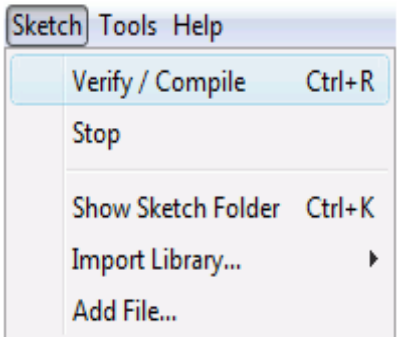
Menu **Edit** (Editer):

- Copy for forum (copier pour le forum) :

Copie le code du programme dans le presse-papier dans un format approprié pour poster sur le forum, avec coloration syntaxique complète.

- Copy as HTML (copier en tant qu'HTML) :

Copie le code de votre programme dans le presse-papier en tant qu'HTML, au format adapté pour être intégré dans des pages web.



Menu **Sketch** (Programme):

- Verify/Compile (Vérifier/compiler) :

Vérifie le code à la recherche d'erreurs

- Import Library (Importer la librairie) :

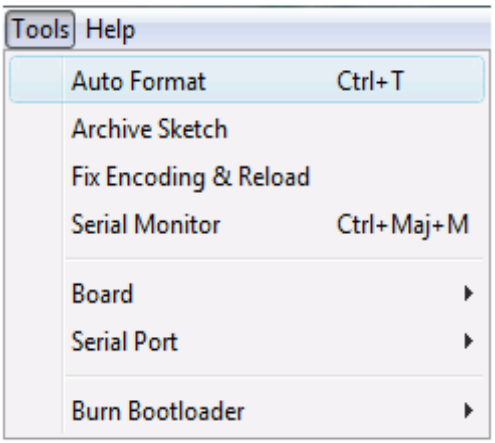
Ajoute une librairie à votre programme en insérant l'instruction #include dans votre code. Pour plus de détails, voir "librairies" ci-dessous.

- Show Sketch Folder (Montrer le répertoire du programme) :

Ouvre le répertoire courant du programme sur votre bureau

- Add File... (Ajouter un fichier) :

Ajoute un fichier source à votre programme (il sera copier à partir de sa localisation courante. Le nouveau fichier apparaît dans un nouvel onglet dans la fenêtre d'édition. Les fichiers peuvent être retirés du programme en utilisant le menu "tab".



Menu **Tools** (Outils):

- Auto Format (Mise en forme Automatique) :

Cette fonction formate votre code joliment : c'est à dire ajuste le code de façon à ce que les accolades soient alignées ey que ce que les instructions entre les accolades soient davantage décalées.

- Board (Carte) :

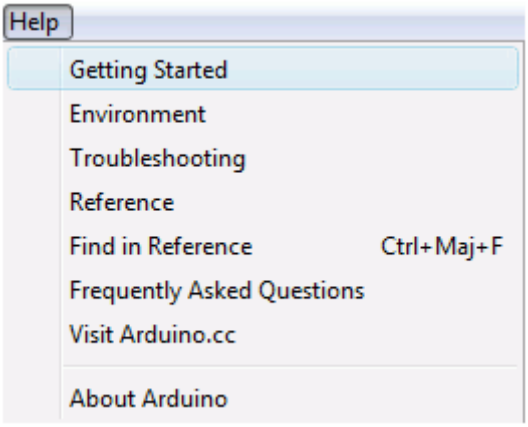
Sélectionne la carte Arduino que vous utilisez. Voir ci-dessous pour la description des différentes cartes.

- Serial Port (Port Série) :

Ce menu contient tous les ports séries (réels ou virtuels) présents sur votre ordinateur. Il est automatiquement mis à jour à chaque fois que vous ouvrez le niveau supérieur du menu outils.

- Burn Bootloader (Graver le bootloader) :

Cette fonctionnalité vous permet de graver le bootloader dans le microcontrôleur sur une carte Arduino. Ceci n'est pas nécessaire pour une utilisation normale de votre carte Arduino (le bootloader est déjà gravé dans votre carte quand vous l'achetez) mais peut être utile si vous achetez un nouvel ATmega (qui sera normalement livré sans bootloader). Assurer vous que vous avez sélectionné la carte correcte dans le menu**Boards** avant de graver le booloader. Si vous utilisez un AVR ISP, vous devez sélectionner l'item correspondant à votre programmeur dans le menu **Serial Port**.



Menu **Help**

- Propose tout ce qui peut être utile pour être aidé lors de l'écriture des programmes, notamment un lien vers la référence du langage en anglais.

5. Sketchbook ("Livre de programmes")

Le logiciel Arduino intègre le concept d'un "sketchbook" (livre de programme) : un endroit réservé pour stocker vos programmes. Les programmes que vous mettez dans votre "sketchbook" pourront être ouvert directement depuis le menu **File > Sketchbook** ou à l'aide du bouton **Open** (Ouvrir) dans la barre d'outils.
La première fois que vous démarrer le logiciel Arduino, un chemin automatique sera créé pour votre "sketchbook". Vous pouvez voir ou modifier cette localisation depuis le menu **File > Preferences**.

6. Onglets, fichiers multiples et compilation

Vous permet de gérer les programmes avec plus d'un fichier (chaque fichier apparaissant dans son propre onglet). Ces fichiers doivent être des fichiers Arduino normaux (no extension), des fichiers C (extension .c), des fichiers C++ (.cpp) ou des fichiers d'entête (.h).

Introduction MATLAB : complément GUI

I - Elements de base de l'interface graphique

Pour créer une interface, il faut disposer d'une fenêtre de base dans laquelle seront insérés les éléments graphiques (objets). A noter que tout dessin graphique ou affichage d'image (résultat de plot, mesh, imshow) peut servir de fenêtre de base.

 Création d'une nouvelle fenêtre pour application:

fig1 = figure

Le paramètre fig1 est le handle de la fenêtre, c'est à dire le numéro de repère de la fenêtre attribué par Matlab à sa création. Il est possible d'appliquer des fonctions sur cette fenêtre (redimensionnement, ajout de menus, boutons, ...) en précisant dans les fonctions le handleauquel elle s'applique. La fenêtre active à un instant donné a pour handle implicite gcf. De façon générale, tout objet graphique se voit attribué un handle; ce handle sert de référence à cet objet dans l'application.

 Propriétés d'une fenêtre graphique (ou d'un objet)

get(fig1)

Les principales propriétés sont : le titre, la position et la dimension dans l'écran, la couleur de fond, la présence et le type de menus, le redimensionnement...

Toute propriété particulière est obtenu par :

valeur_propriété = get(fig1, 'nom_propriété')

Toute propriété (modifiable!) peut être modifiée en définissant une nouvelle valeur pour la propriété considérée (valeur numérique, chaîne, liste de valeur, tableau...)

set(fig1, 'nom_propriété' , valeur_propriété)

Ex : set(fig1 , 'Name' , 'Demo GUI' , 'NumberTitle' , 'off');

La fenêtre de base est l'écran qui a pour handle "0". Par get (0 , 'ScreenSize'), on obtient la taille de l'écran physique de l'écran. Ces valeurs permettent de fixer la taille d'une fenêtre en

rapport avec la dimension physique de l'écran et d'éviter de créer une application qui "déborde" de l'écran!

La taille et la position de la fenêtre (ou d'un objet) se fixent par modification de sa propriété ou contrôle "position", comprenant les coordonnées (Xor,Yor) du coin inférieur gauche et ses dimensions (Xfen,Yfen):

set(fig1 , 'position' , [10 , 10 , 300 , 200])

L'ensemble des propriétés modifiables d'un objet est donné par set(handle_objet) . La liste s'affiche avec les valeurs possibles pour les différentes propriétés; les valeurs par défaut sont signalées par des crochets { } . Exemple :

set(fig1)

Tout objet graphique créé pourra être supprimé par :

delete (handle_objet)

La suppression d'un objet entraîne la suppression des objets qui lui sont liés (objets fils).

Insertion d'un Objet dans la fenêtre

L'insertion d'un objet dans une fenêtre se fait par la fonction "uicontrol", dont le premier paramètre est le handle de la figure de référence. Le deuxième paramètre précise le "style" ou type d'objet à insérer.

Exemple le "texte fixe" est l'objet le plus simple; il permet de placer un texte dans la fenêtre.

text1 = uicontrol(fig1 , 'style' , 'text' , 'position' , [100,150,170,30] ,...

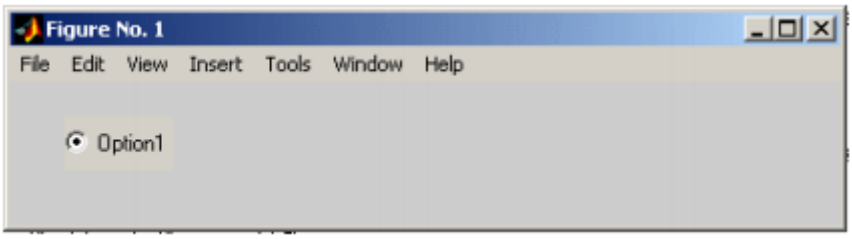
'string' , 'Bonjour' , 'fontsize' , 15)

Toutes les propriétés de cet objet peuvent être modifiées par la commande "set". Par exemple, le texte en lui-même ('string') étant une propriété, il peut être modifié:

set(text1 , 'string' , 'Au revoir');

Autre exemple : insertion d'un bouton-radio :

radio1 = uicontrol(fig1 , 'style' , 'radio' , 'String' , 'Option1' , 'Position' , [30,30,60,30])



A la différence du "texte", on remarque que cet objet est "cliquable" à la souris, avec modification de son aspect (cette fonction est prise en charge sans aucune programmation par l'utilisateur).

II - Principe de l'interaction avec la souris

La presque totalité des objets de l'interface graphique (curseur, case à cocher...) peut interagir avec la souris.

 La fonctionnalité la plus courante est la modification de la valeur associée à l'objet (si elle existe): pour les objets destinés à faire une saisie (case à cocher, curseur, champ de saisie, choix de liste...), Matlab gère automatiquement la valeur associée. Cette valeur est récupérable par toute partie de l'application par la fonction "get" :

valeur = get (handle_objet , 'value');

Cette fonctionnalité permet de saisir la valeur d'une variable par l'interface graphique plutôt que par le clavier.

 La deuxième interaction courante est une action déclenchée par le "clic" souris sur l'objet (appuyé puis relâché): la fonction associée est décrite dans la propriété "callback" de l'objet. Cette fonction peut être une instruction de base Matlab ou une fonction définie par l'utilisateur (stockée en fichier .m)

set(radio1 , 'callback' , 'get(radio1 , ' 'value' ') ');

Remarquer que la fonction est décrite en chaîne de caractères avec des " ' " en début et fin, ce qui oblige à doubler les " ' " pour ceux qui sont inclus dans la chaîne principale.

Cette description en chaîne permet de définir en callback une liste d'instruction, ce qui évite d'écrire une multitude de petites fonctions externes dédiées aux callbacks.

 Certains objets n'ont pas de callback (cas des figures) mais possèdent d'autres actions associées à la souris. Leur emploi est identique au callback classique (description de la fonction en liste d'instructions). Les principales sont :

WindowButtonUpFcn WindowButtonDownFcn WindowButtonMotionFcn

Exemple : récupération des coordonnées en pixels de la souris au clic

fig1 = figure ;

set(fig1 , 'WindowButtonDownFcn' , 'get(fig1 , ' 'CurrentPoint ' ') ');

Si on désire obtenir des coordonnées dans l'espace de mesure des axes d'un graphique, plutôt qu'en pixels de la figure, il faut faire référence aux axes (gca) dans la fonction de clic :plot(20 , 20 , ' r+ ') % tracé d'une croix rouge au centre

set(gcf , 'WindowButtonDownFcn' , 'get(gca , ' 'CurrentPoint' ') ')

 Certains objets possèdent une fonction callback et une fonction associée au clic souris (par exemple : ButtonDownFcn)

III - Principaux Objets Graphiques

 Bouton poussoir

Un bouton poussoir se crée par :

bp1= uicontrol (fig1 , 'style' , 'push' , 'position' , [10 100 60 30] ,...'string' , 'Début' , 'callback' , 'plot(T,X)')

Lorsqu'on clique sur le bouton poussoir, il provoque l'exécution de la fonction indiquée dans le 'callback'. Cette fonction peut être une instruction de base Matlab ou une liste d'instruction, ce qui évite d'écrire une multitude de petites fonctions exécutées pat les callbacks.

Un bouton-poussoir s'inactive par la commande :

set(bp1 , 'enable' , 'off')

Par cette commande, on peut rendre inactif certaines commandes, par exemple lorsqu'il manque des informations pour traiter un problème.

- Menus

Généralement, les menus de la fenêtre d'application ne sont pas les menus standard (voir vue ci-dessus mais des menus spécifiques. Un menu est un titre complété par une liste de sous-menu.

Les actions (callbacks) sont généralement lancés à partir des sous-menus. L'ajout de menus spécifique se fait par :

menu1 = uimenu(fig1 , 'label' , ' Statist.');

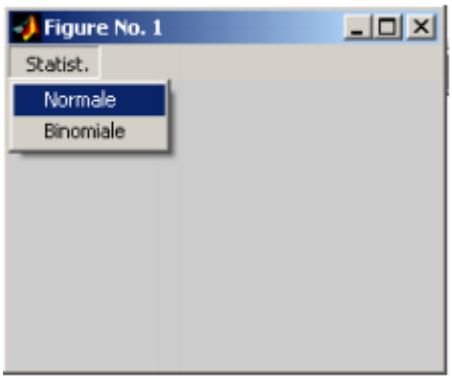
Un sous-menu est un élément du menu principal, donc de l'entité père. Il est donc déclaré car menu du menu principal.

smenu1 = uimenu(menu1 , 'label' , 'Normale' , 'callback' , 'methode_normale')

smenu2 = uimenu(menu1 , 'label' , 'Binomiale' , 'callback' , 'methode_binomiale');

Pour enlever les menus standards de la fenêtre, il faut fixer la propriété "Menubar" à la valeur par défaut menubar :

set(fig1,'menubar',menubar);



- Ascenseur ou slider

L'ascenseur a pour objectif de fixer la valeur d'un paramètre entre deux bornes fixées. La valeur prise par la variable est représentée par la position du curseur.

slid1=uicontrol(fig1,'style','slider','position', [100,50,150,20] , 'Min' , -1 , 'Max' , 1 , ...'callback' , 'val_variable = get(slid1 , ''value'')');

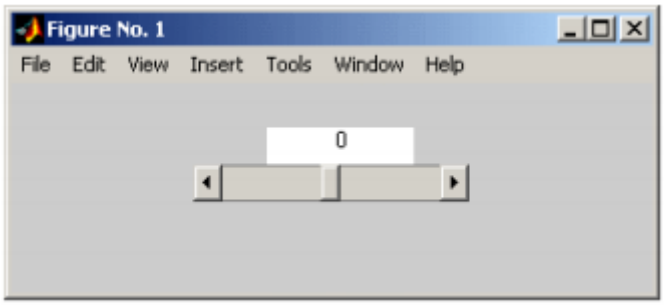
Les textes (variable affectée, valeurs..) ne sont pas définis par le slider. Il faut le compléter par des éléments textes convenablement placés et paramétrés; leur valeur est à modifier par le callback du slider.

Exemple d'ascenseur avec affichage de la valeur:

fig1=figure;

texte1=uicontrol(fig1,'Style','text','String',0,'Position', [140,70,80,20],'BackGroundcolor','w');

slid1=uicontrol(fig1,'style','slider','position', [100,50,150,20] , 'Min' , -1 , 'Max' , 1 , …'callback' , 'set(texte1,''String'', get(slid1 , ''value''))');



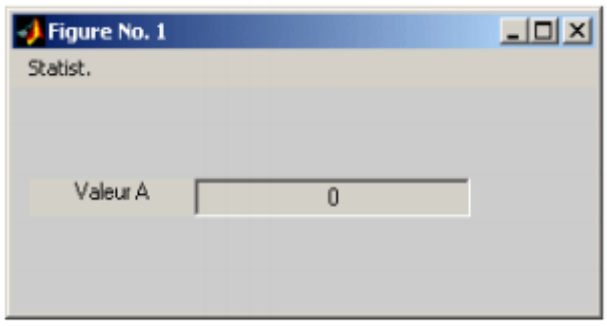
☐ Texte Editable

Permet à l'utilisateur de saisir une valeur. C'est une fonction importante.

Text1 = uicontrol (fig1 , 'style' , ' edit' , 'position', [100,50,150,20] , 'Max' , 1 , 'string' , '0');

Généralement, Il faut associer un texte fixe pour préciser le rôle de la fenêtre de saisie à l'utilisateur. Exemple : le texte est placé à gauche de la fenêtre de saisie

uicontrol (fig1 , 'style' , ' texte' , 'position', [10,50,90,20] , 'string' , 'Valeur A');

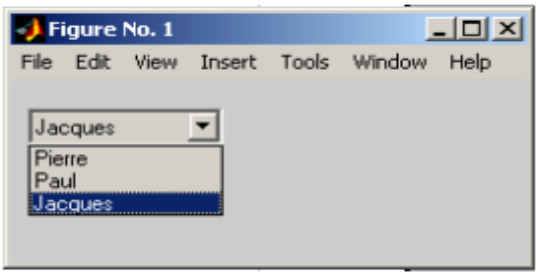


☐ Liste de choix

La liste de choix ou pop-up menu permet de sélectionner une valeur parmi une liste.

Généralement, cette valeur est un texte. La valeur retournée lors du choix (paramètre 'Value') est le numéro de ligne du choix.

choix1 = uicontrol (gcf , 'Style' , 'popup' , 'String' , 'Pierre|Paul|Jacques' , 'Position' , [10 10 100 80]);



La liste de texte est modifiable après la création de la fenêtre de choix, en modifiant la propriété 'String'.

☐ Bouton Radio

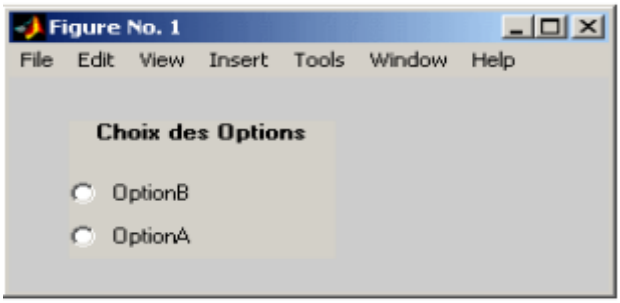
Le bouton Radio permet de fixer un paramètre binaire (0 ou 1), représentant souvent un choix ou une option dans une application.

fig1 = figure ;

radio1 = uicontrol(fig1 , 'style' , 'Radio' , 'Position' , [30 20 130 25] , 'String' , ' OptionA ');

radio2 = uicontrol(fig1 , 'style' , 'Radio' , 'Position' , [30 45 130 25] , 'String' , ' OptionB ');

uicontrol(fig1 , 'style' , 'Text' , 'Position' , [30 70 130 30] , 'String' , ...' Choix des Options ' , 'FontWeight', 'bold');



Remarquer que les choix ne sont pas exclusifs (chaque choix peut être sélectionné). Pour obtenir l'exclusion mutuelle, il faut agir sur les valeurs de choix par les callbacks.

set(bradio1 , ' Value' , 1);

set(radio1 , 'callback' , 'set(radio2 , ' ' Value' ' , 0) ') ;

set(radio2 , 'callback' , 'set(radio1 , ' ' Value' ' , 0) ') ;

- Cadre

Le cadre permet de dessiner un rectangle de présentation (par exemple regroupement de diverses entités graphiques dans un rectangle).

Le cadre se déclare par :

cadre1 = uicontrol (fig1 , 'style' , 'frame' , 'position' , [posX ,posY,tailleX,tailleY])

- Graphiques

Les graphiques se dessinent dans une partie de la fenêtre définie par la fonction 'subplot', dont les paramètres sont différents de l'emploi classique (division de la fenêtre en sous-fenêtres de taille égale)

subplot('Position' , [Xpos Ypos Xtaille Ytaille])

Les paramètres se définissent en % de la fenêtre (redimensionnement automatique des zones graphiques avec le redimensionnement de la fenêtre). Il est possible d'ouvrir plusieurs zones graphiques dans une même fenêtre. Les zones ne doivent pas se chevaucher, sous peine d'effacement de la zone située sous la nouvelle zone.

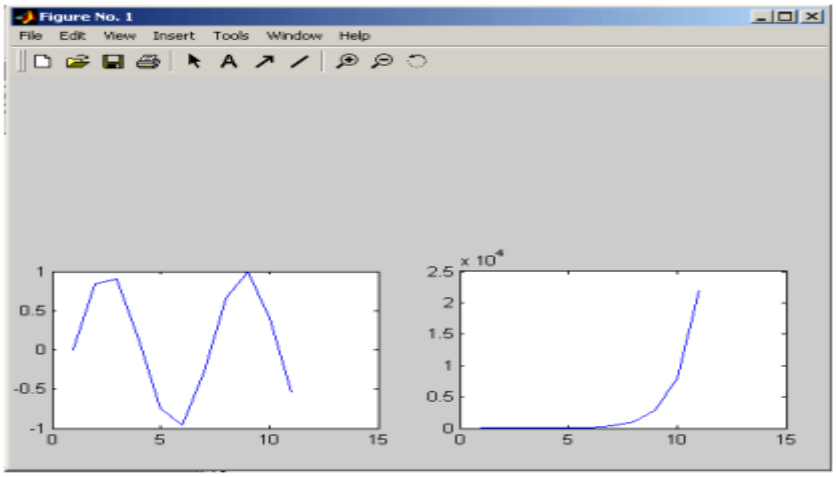
fig1 = figure ;

z1 = subplot ('Position' , [.05 .1 .4 .4]) ;

plot (sin(0: 10))

z2 = subplot ('Position' , [.55 .1 .4 .4]) ;

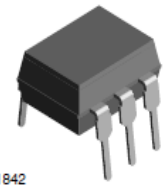
plot (exp(0 : 10))

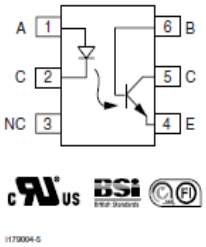


4N25, 4N26, 4N27, 4N28

Vishay Semiconductors

Optocoupler, Phototransistor Output, with Base Connection





FEATURES

- Isolation test voltage 5000 V_{RMS}
- Interfaces with common logic families
- Input-output coupling capacitance < 0.5 pF
- Industry standard dual-in-line 6 pin package
- Compliant to RoHS directive 2002/95/EC and in accordance to WEEE 2002/96/EC

Pb-free e3 RoHS COMPLIANT

APPLICATIONS

- AC mains detection
- Reed relay driving
- Switch mode power supply feedback
- Telephone ring detection
- Logic ground isolation
- Logic coupling with high frequency noise rejection

AGENCY APPROVALS

- UL1577, file no. E52744
- BSI: EN 60065:2002, EN 60950:2000
- FIMKO: EN 60950, EN 60065, EN 60335

DESCRIPTION

The 4N25 family is an industry standard single channel phototransistor coupler. This family includes the 4N25, 4N26, 4N27, 4N28. Each optocoupler consists of gallium arsenide infrared LED and a silicon NPN phototransistor.

ORDER INFORMATION	
PART	**REMARKS**
4N25	CTR > 20 %, DIP-6
4N26	CTR > 20 %, DIP-6
4N27	CTR > 10 %, DIP-6
4N28	CTR > 10 %, DIP-6

ABSOLUTE MAXIMUM RATINGS (1)				
PARAMETER	**TEST CONDITION**	**SYMBOL**	**VALUE**	**UNIT**
INPUT				
Reverse voltage		V_R	5	V
Forward current		I_F	60	mA
Surge current	t ≤ 10 µs	I_{FSM}	3	A
Power dissipation		P_{diss}	100	mW
OUTPUT				
Collector emitter breakdown voltage		V_{CEO}	70	V
Emitter base breakdown voltage		V_{EBO}	7	V
Collector current		I_C	50	mA
	t ≤ 1 ms	I_C	100	mA
Power dissipation		P_{diss}	150	mW

4N25, 4N26, 4N27, 4N28

Vishay Semiconductors Optocoupler, Phototransistor Output, with Base Connection

CURRENT TRANSFER RATIO (1)							
PARAMETER	TEST CONDITION	PART	SYMBOL	MIN.	TYP.	MAX.	UNIT
DC current transfer ratio	V_{CE} = 10 V, I_F = 10 mA	4N25	CTR_{DC}	20	50		%
		4N26	CTR_{DC}	20	50		%
		4N27	CTR_{DC}	10	30		%
		4N28	CTR_{DC}	10	30		%

Note

(1) Indicates JEDEC registered values.

SWITCHING CHARACTERISTICS						
PARAMETER	TEST CONDITION	SYMBOL	MIN.	TYP.	MAX.	UNIT
Rise and fall times	V_{CE} = 10 V, I_F = 10 mA, R_L = 100 Ω	t_r, t_f		2		µs

TYPICAL CHARACTERISTICS

T_{amb} = 25 °C, unless otherwise specified

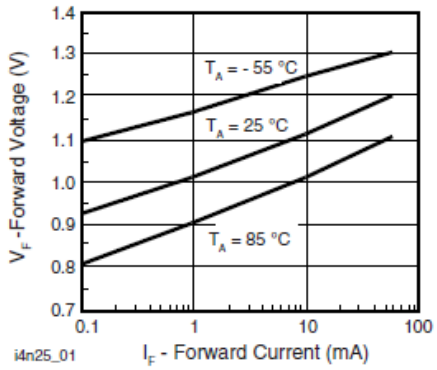


Fig. 1 - Forward Voltage vs. Forward Current

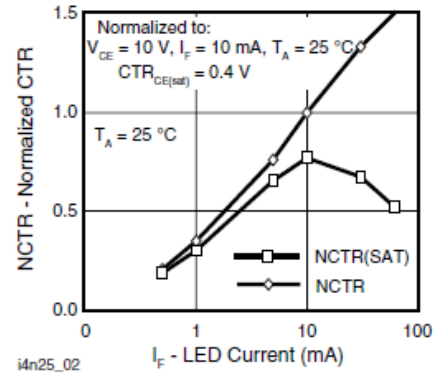


Fig. 2 - Normalized Non-Saturated and Saturated CTR vs. LED Current

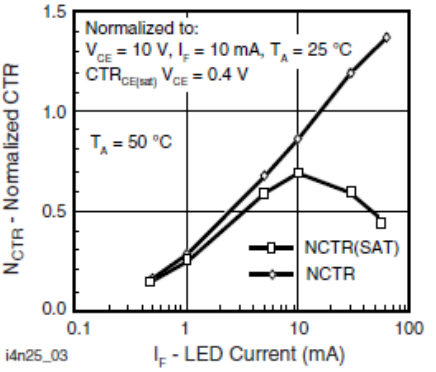


Fig. 3 - Normalized Non-Saturated and Saturated CTR vs. LED Current

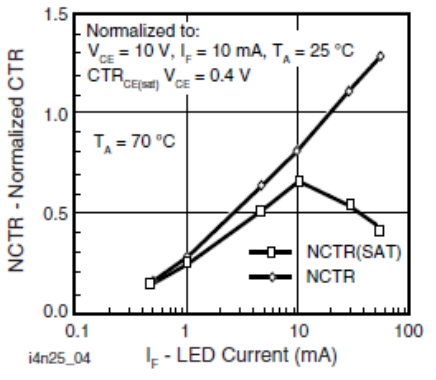


Fig. 4 - Normalized Non-Saturated and Saturated CTR vs. LED Current

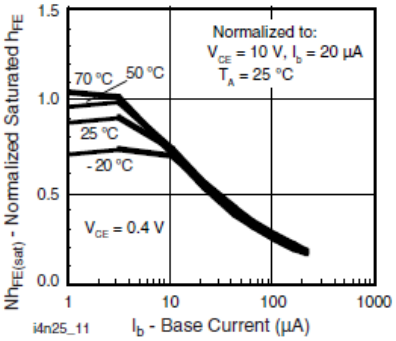


Fig. 11 - Normalized h_{FE} vs. Base Current and Temperature

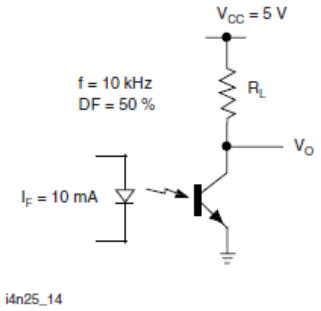


Fig. 14 - Switching Schematic

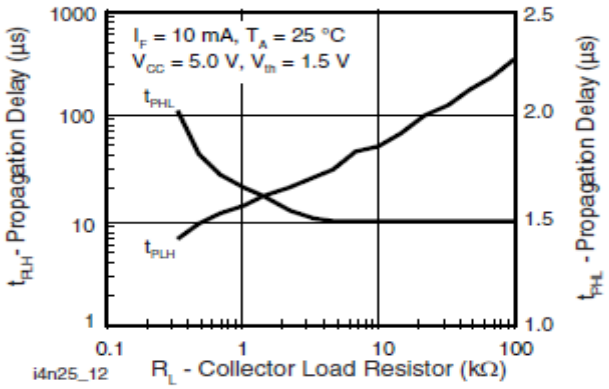


Fig. 12 - Propagation Delay vs. Collector Load Resistor

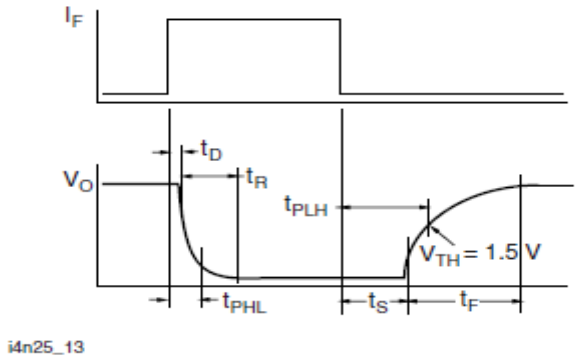
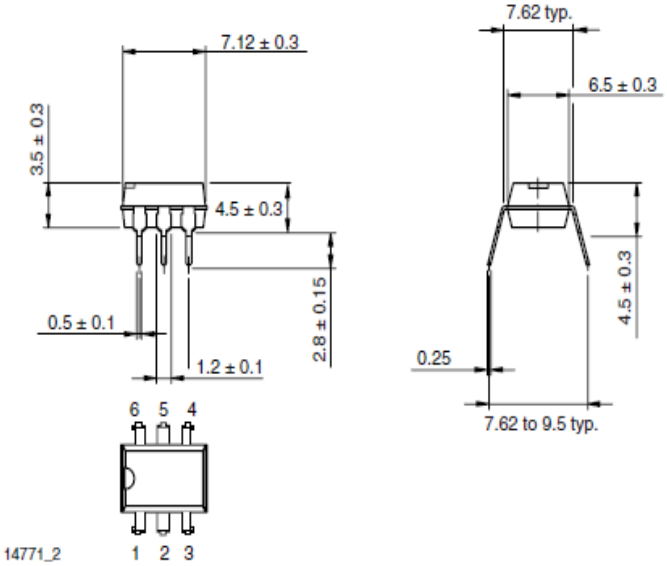
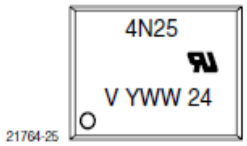


Fig. 13 - Switching Timing

PACKAGE DIMENSIONS in millimeters



PACKAGE MARKING

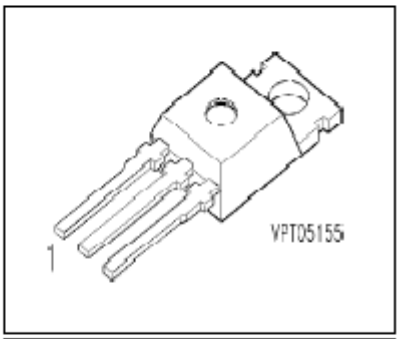


SIEMENS

BUZ 70

SIPMOS ® Power Transistor

- N channel
- Enhancement mode
- Avalanche-rated



Pin 1	Pin 2	Pin 3
G	D	S

Type	V_{DS}	I_D	$R_{DS(on)}$	Package	Ordering Code
BUZ 70	60 V	12 A	0.15 Ω	TO-220 AB	C67078-S1334-A2

Maximum Ratings

Parameter	Symbol	Values	Unit
Continuous drain current T_C = 33 °C	I_D	12	A
Pulsed drain current T_C = 25 °C	I_{Dpuls}	48	
Avalanche current,limited by T_{jmax}	I_{AR}	12	
Avalanche energy,periodic limited by T_{jmax}	E_{AR}	1	mJ
Avalanche energy, single pulse I_D = 12 A, V_{DD} = 25 V, R_{GS} = 25 Ω L = 48.6 µH, T_j = 25 °C	E_{AS}	6	
Gate source voltage	V_{GS}	± 20	V
Power dissipation T_C = 25 °C	P_{tot}	40	W
Operating temperature	T_j	-55 ... + 150	°C
Storage temperature	T_{stg}	-55 ... + 150	
Thermal resistance, chip case	R_{thJC}	≤ 3.1	K/W
Thermal resistance, chip to ambient	R_{thJA}	75	
DIN humidity category, DIN 40 040		E	
IEC climatic category, DIN IEC 68-1		55 / 150 / 56	

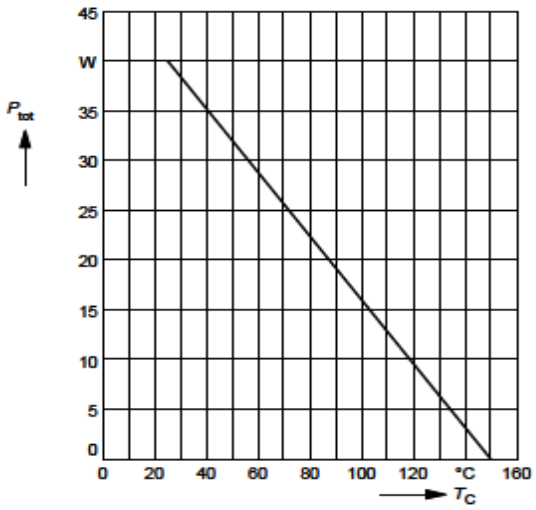
SIEMENS BUZ 70

Electrical Characteristics, at T_j = 25°C, unless otherwise specified

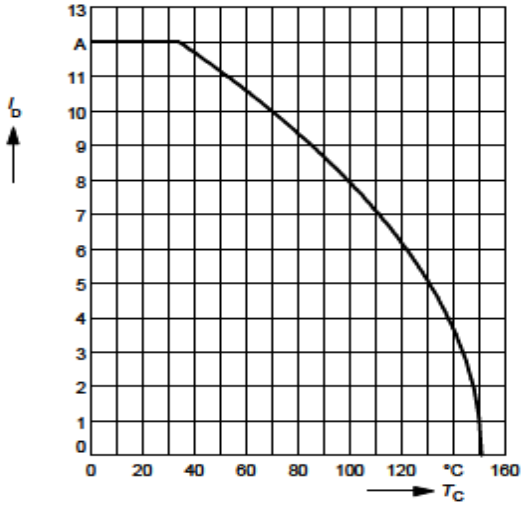
Parameter	Symbol	Values			Unit
		min.	typ.	max.	
Static Characteristics					
Drain- source breakdown voltage V_{GS} = 0 V, I_D = 0.25 mA, T_j = 25 °C	$V_{(BR)DSS}$	60	-	-	V
Gate threshold voltage V_{GS}=V_{DS}, I_D = 1 mA	$V_{GS(th)}$	2.1	3	4	
Zero gate voltage drain current V_{DS} = 60 V, V_{GS} = 0 V, T_j = 25 °C V_{DS} = 60 V, V_{GS} = 0 V, T_j = 125 °C	I_{DSS}	- -	0.1 10	1 100	µA
Gate-source leakage current V_{GS} = 20 V, V_{DS} = 0 V	I_{GSS}	-	10	100	nA
Drain-Source on-resistance V_{GS} = 10 V, I_D = 7.5 A	$R_{DS(on)}$	-	0.12	0.15	Ω

SIEMENS BUZ 70

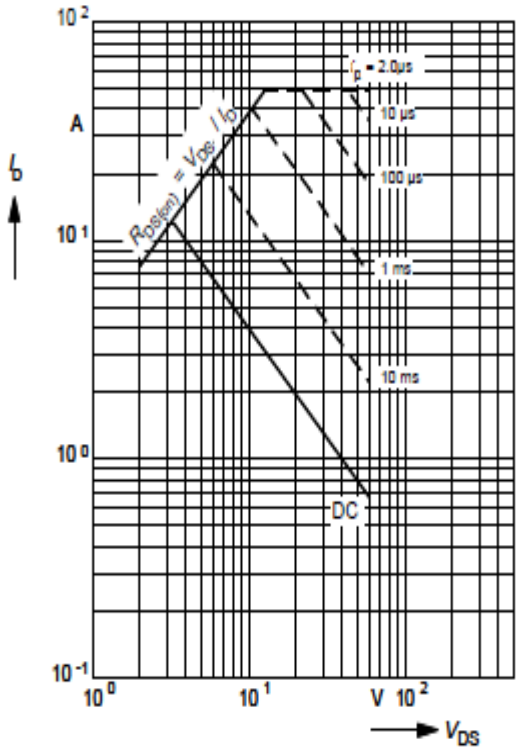
Power dissipation
$P_{tot} = f(T_C)$



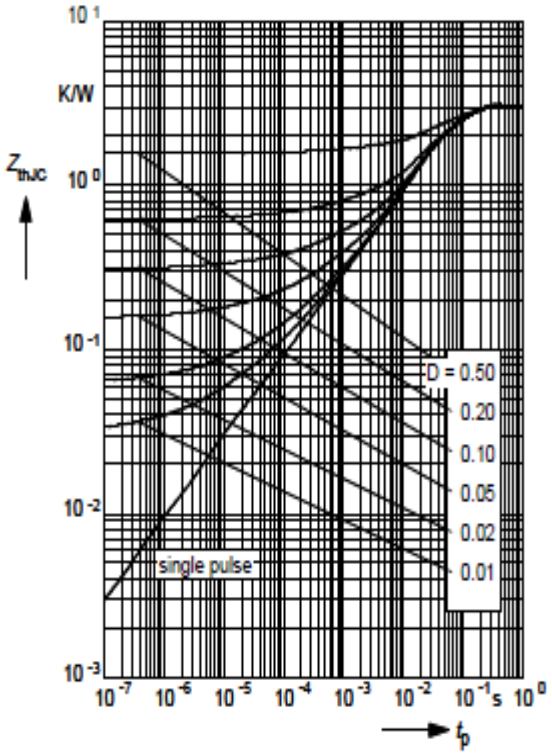
Drain current
$I_D = f(T_C)$
parameter: $V_{GS} \geq 10$ V



Safe operating area
$I_D = f(V_{DS})$
parameter: $D = 0.01$, $T_C = 25$°C



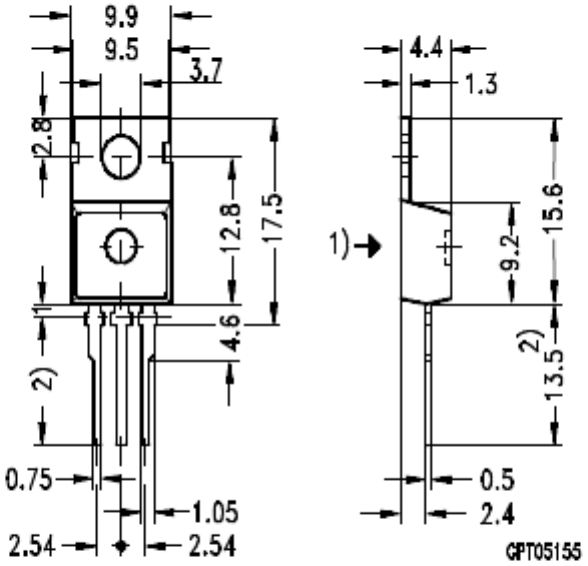
Transient thermal impedance
$Z_{th\,JC} = f(t_p)$
parameter: $D = t_p / T$



Package Outlines

TO-220 AB

Dimension in mm



1) punch direction, burr max. 0.04

2) dip tinning

3) max. 14.5 by dip tinning press burr max. 0.05