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 Master

Filière Électrotechnique
Spécialité machines électriques

présenté par

Hachim manel

&

Abadou tahani

Réalisation d'une régulation numérique par PID pour la commande en position d'un moteur à courant continu

Proposé par : Promoteur : Benselama Zoubir

Co-promoteur : Bacha Amine

Année Universitaire 2019-2020

Remerciement

Tout d'abord, on tient à remercier le bon Dieu le tout Puissant de nous avoir donné la force, la patience et la volonté pour réaliser ce travail.

Nos remerciements s'adressent ensuite à Notre encadreur

Mr Benslama Zoubir

Pour le choix de ce sujet et aussi pour encadrement et ses précieux conseils.

C'est un plaisir de remercier les membres du jury qui nous font l'honneur de participer à notre soutenance.

On remercie sincèrement Mr: Bacha amine pour son aide dans la pratique.

On remercie également nos parents pour leur soutien moral et financier durant nos études

Nos remerciements les plus vifs reviennent aux étudiants du département d'électronique sans exception.

Nous profitons aussi de ce mémoire pour exprimer nos plus vifs remerciements envers tous les enseignants qui nous ont apportés du soutien durant nos études et

Envers tous nos amis qui ont été toujours prés de nous avec leurs encouragements, critiques et conseils

Enfin, Nous voudrions associer à nos remerciements toutes les personnes qui ont contribué de près ou de loin à l'aboutissement de ce travail.

Hachim manel & Abadou tahani

Dédicace

A nos chers parents

Pour leur soutien, leur patience, leur sacrifice

Et leur amour, vous méritez tout éloge,

Vous qui avez fait de nous ce que nous sommes maintenant.

Nous espérons être l'image que vous avez fait de

Nous, que dieu vous garde et vous bénisse.

Nous dédions aussi ce travail à nos chers frères et sœurs,

Pour leur affection et leur encouragement qui ont toujours

Été pour nous des plus précieux.

Que ce travail soit pour vous le gage de notre profond amour

A tout nos amis

A rabi3, kachkouch et macha

A tous ceux qui nous ont aidés.

A tous ceux que nous aimons, nous dédions ce travail

Hachim manel & Abadou tahani

ملخص: الهدف من مشروع تخرجنا هو تنفيذ ضبط موضع محرك ذات التيار المستمر ، هذا الأخير سيتم تطويره بواسطة المنظم الرقمي بي اي دي المصمم بواسطة بطاقة اردوينو. التي ستنفذ التحكم لكي يتمكن النظام للوصول إلى نقطة محددة و الحفاظ عليها مع تحقيق معاييردفتر شروط (استقرار، سرعة، دقة).
التعليمات ،خصائص بي اي دي و الرسم البياني لتطور الموضع في الزمن الحقيقي سترسل و تعرض بمساعدة واجهة بيانية مطورة على الدلفي.
كلمات المفاتيح: ضبط ، موضع، محرك ذات التيار المستمر ، المنظم، بي اي دي، اردوينو، استقرار،السرعة، الدقة، الدلفي.

Résumé : L'objectif de notre projet de fin d'étude est de Réaliser la Régulation de position d'un moteur à courant continu, cette dernière sera mise au point par un régulateur PID dont sa réalisation s'articulera autour de la carte de prototypage Arduino. Qui élaborera une commande qui permettra au système d'atteindre et de maintenir une certaine consigne tout en satisfaisant les critères d'un cahier de charges (Stabilité, Rapidité, Précision). Les consignes, les paramètres du PID ainsi que le graphe d'évolution de la position en temps réel seront transmis et affiché à l'aide d'une interface graphique développé sur Delphi.
Mots clés : Commande; Position; Moteur à courant continu; Régulateur ;PID ;Arduino Stabilité ; Rapidité ; Précision ; Delphi.

Abstract: The objective of our end-of-study project is to realize a regulation of position of a direct current motor which will be developed by a PID which revolve around an Arduino prototyping board that will elaborate a command in order to the system is able to reach and maintain a certain set point while satisfying the criteria of a specification (stability, speed, precision). The instructions, PID parameters and the position evolution graph in real time will be transmitted, display with the help of graphical interface developed in Delphi.

Key Words : Command; Position; DC motor; Regulator; PID; Arduino ;Stability; Speed; Precision; Delphi.

Liste des définitions

e(p) ,e(t) : Consigne (grandeur d'entrée).

C(P) **:** Fonction de transfert du correcteur.

H(p) : Fonction de transfert la boucle de retour (capteur).

G(s) **:** Fonction de transfert de l'ensemble actionneur + système.

ε (p),ε(t) : L'écart.

Y(p),Y(t) : La mesure.

S(p),S(t) : La sortie.

U(p) ,U(t): Signal de commande.

D(p) **:** Fonction de transfert d'une perturbation.

$\mathbf{D_1}$: Premier dépassement.

$\mathbf{T_1}$: Instant du premier dépassement.

$\mathbf{Y(\infty)}$: Valeur asymptotique de la sortie en régime permanent.

$\mathbf{T_i}$: Constante d'intégrations.

$\mathbf{T_d}$: Constante de dérivation.

$\mathbf{K_d}$: Gain dérivé.

$\mathbf{K_p}$: Gain proportionnelle.

$\mathbf{K_i}$: Gain d'intégral.

F(p) **:** Fonction de transfert de procédé.

$\mathbf{K_{cr}}$ **:** Gain proportionnelle critique.

$\mathbf{T_{cr}}$: Période d'oscillation critique.

$\mathbf{T_u}$: Point d'intersection entre l'abscisse et la tangente.

$\mathbf{T_e}$: Période d'échantillonnage.

$\mathbf{U_{pwm}}$: Rapport cyclique.

$\mathbf{U_c}$: Sortie apres filtrage.

e(k) : La consigne discrete.

ε (k) : L'écart discret.

U(k) : Commande discrète.

Y(k) : La mesure discrete.

Liste des abréviations

PID : **P**roportionnel **I**ntégral **D**érivée.

P : **P**roportionnel.

PI : **P**roportionnel **I**ntégral.

BO : **B**oucle ouvert.

BF : **B**oucle fermé.

CN : **C**ommande **N**umérique.

DIY : **D**o **I**t **Y**ourself.

SRAM : **S**tatic **R**andom **A**ccess **M**emory.

EEPROM: **E**lectrical **E**rasable **P**rogrammable **R**ead **O**nly **M**emory.

FTDI : Future **T**echnology Devices **I**nternational.

USB : **U**niversal **S**erial **B**us.

PWM : **P**ulse **W**idth **M**odulation.

ICSP : **I**n-**C**ircuit **S**erial **P**rogramming.

IC : **I**ntegrated **C**ircuit.

RAM : **R**andom **A**ccess **M**emory.

CPU : **C**entral **P**rocessing **U**nit.

AC : **A**lternating **C**urrent.

DC : **D**irect **C**urrent.

GND : **G**rou**ND**.

LED : **L**ight-**E**mitting Diode.

SS : **S**lave **S**elect.

MOSI : **M**aster **O**ut **S**lave **I**n.

MISO : **M**aster **I**n **S**lave **O**ut.

SCK : **S**erial **C**lock.

SPI : **S**erial **P**eripheral **I**nterface.

TWI : **T**wo **W**ire **I**nterface.

SDA : **D**ata **L**ine.

SCL : **C**lock **L**ine.

AREF : **A**nalog **REF**ence.

Rx : **R**eceiver.

Tx : **T**ransmitter.

UART : **U**niversal **A**synchronous **R**eceiver **T**ransmitter.

RS-232 : **R**ecommended **S**tandard for serial communication.

ASCII : **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.

TTL : **T**ransistor- **T**ransistor **L**ogic.

IDE : **I**ntegrated **D**evelopment **E**nvironment.

PC : **P**ersonnel **C**omputer.

NPN : **N**égative **P**ositive **N**égative.

PNP : **P**ositive **N**égative **P**ositive.

PPR : **P**oints Par **R**otations.

Rad : **R**apid **A**pplication **D**éveloppent.

UAL : **E**lectronics **D**ata **I**nterchange.

UAL : **U**nité **A**rithmétique **L**ogique.

MLI : **M**odulation a **L**argeur d **I**mpulsion.

I2C : **I**ntegret- **I**ntegret **C**ircuit

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Depuis des années, l'humanité connaît une évolution exponentielle qui ne fait que s'accroître avec l'essor des nouvelles technologies. Elles sont le signe de notre modernité et seraient désormais représentatives des Hommes. Ces technologies sont significatives de progrès, permettant ainsi une évolution de la société vers un futur désirable pour but, d'améliorer le quotidien et les conditions de vie, parmi ces technologies **''** l'automatique**''** [1].

L'automatique est généralement définie comme la science qui traite des ensembles qui se suffissent à eux-mêmes et où l'intervention humaine est limitée à l'alimentation en énergie et en matière première.

L'objectif de l'automatique est de remplacer l'homme dans la plupart des tâches (tâches répétitives, pénibles, dangereuses, trop précises, trop rapides) qu'il réalise dans tous les domaines par des systèmes automatiques [2].

Ces système automatiques peuvent opérer en boucle ouverte à partir d'un seul signal de commande, n'ayant aucune information sur la sortie la correction est impossible, c'est uniquement avec une boucle fermée (contre réaction) qu'on peut stabiliser, améliorer les performances et de compenser l'effet des perturbations, la commande de ces système est faite par des régulateurs.

Un régulateur compare la consigne à une mesure qui est effectué par des capteurs puis élabore une commande qui sera transmise aux actionneurs (vannes, moteurs, etc.), afin de corriger les erreurs et conduire la sortie du système vers la consigne.

Le régulateur PID (proportionnel intégral dérivé) est bien adapté à la plupart des processus industriel d'une part grâce a la simplicité de sa structure , le nombre restreint de paramètres a régler et d'une autre part car il permet d'obtenir une régulation optimale et bien satisfaire les cahiers des charges si les paramètres sont bien choisie (rapidité , précision , stabilité et robustesse).

Dans la réalité industrielle, la complexité des systèmes, ainsi que celle des traitements à réaliser, nécessite souvent le recours à des outils numériques de traitement appelé calculateurs numériques. L'utilisation d'un calculateur à la place d'un correcteur analogique est remarquable, parmi les avantages de la commande numérique :

- mise au point souple.
- meilleurs résultats en termes de performances.
- capacité de mémoire élevée.

L'objectif de notre étude est de réaliser un système capable de réguler la position d'un moteur à courant continu par un correcteur PID numérique à l'aide d'une carte de prototypage Arduino.

Notre projet est par conséquent organisé de la manière suivante :

- Le premier chapitre s'étalera sur des généralités sur la régulation analogique et numérique en ciblant le régulateur PID.
- Le deuxième chapitre traitera la carte de prototypage Arduino qui servira de calculateur numérique pour élaborer la commande de notre système.
- le dernier chapitre sera consacré à la mise en œuvre de notre système ainsi que les résultats obtenus et leurs interprétations.
- Le mémoire sera clôturé par une conclusion générale synthétisant le travail réalisé en donnant aperçu à des perspectives pour une continuation éventuelle du présent travail.

Chapitre 1 Régulation analogique et numérique

1.1 Introduction

En automatique lorsque l'on souhaite atteindre une certaine vitesse, température, position, angle..., il est très souvent nécessaire d'avoir recours à un asservissement ou bien une régulation, c'est à dire un système capable d'atteindre et de maintenir une consigne en utilisant une mesure. Il s'agit donc d'un système bouclé, dont il reste à déterminer la fonction permettant de corriger la commande en fonction de la consigne initiale et de l'erreur mesurée [3].

Ce chapitre va présenter des généralités sur la régulation ou l'asservissement analogique et numérique, afin de cibler le régulateur PID (Proportionnel Intégral Dérivé) qui est le plus utilisé dans l'industrie. Même les systèmes les plus complexes peuvent comporter un réseau dont le principal élément Principal de contrôle est un PID.

1.2 Définition de la régulation et de l'asservissement

Le contrôle d'un système ne peut être optimal que si l'on gère correctement son asservissement ou sa régulation [4].

1.2.1 Asservissement

La consigne, traduisant l'objectif désiré du procédé, n'est pas constante.

Exemples:

- Asservissement de vitesse d'une broche d'un tour à commande numérique.
- Asservissement en position d'une parabole d'un radar de contrôle aérien.

1.2.3 Régulation

La consigne, traduisant l'objectif désiré du procédé, est constante.

Exemples:

- Régulation de température dans un local subissant les variations climatiques.
- Régulation de PH de rejets d'eau destinés à être déversés dans une rivière.

1.3 Objectif de la régulation automatique :

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). Les aspects de sécurité du personnel et des installations sont à prendre en compte comme ceux concernant l'énergie et le respect de l'environnement. 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 [5].

1.4 Principe général de la régulation

Toute chaine de régulation comprend trois maillons indispensables : l'organe de mesure, l'organe de réglage 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 comparer 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 agit sur le processus. Les paramètres qui régissent le processus sont ainsi stabilisés en permanence à des niveaux souhaités [6].

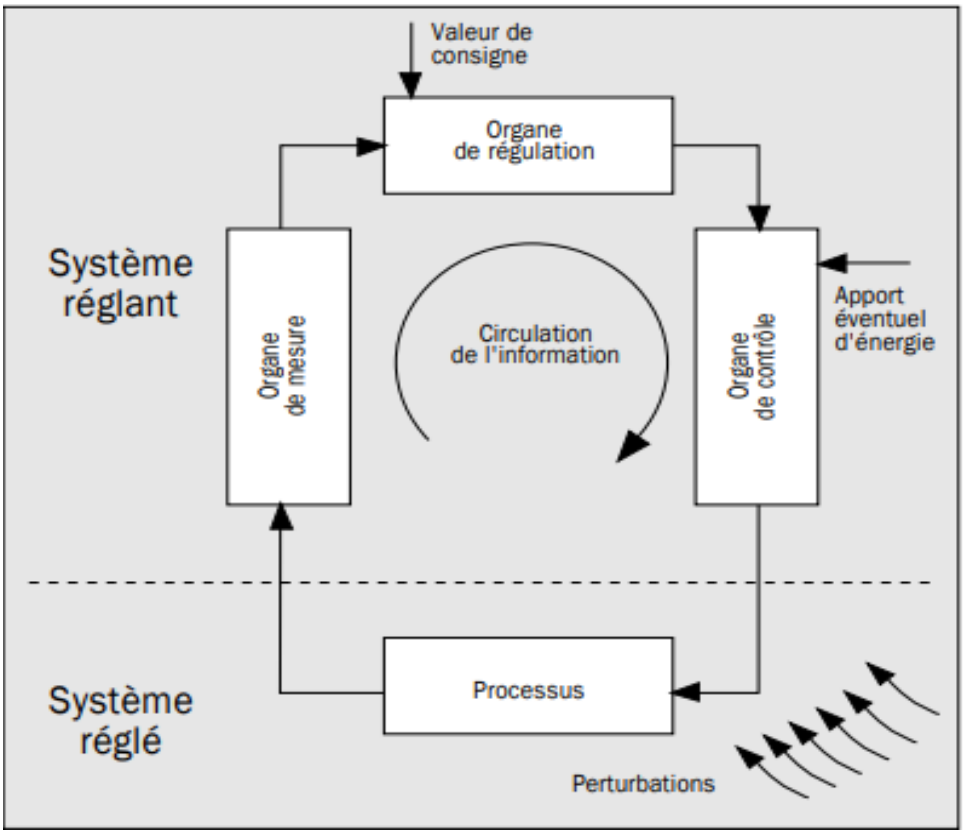


Figure 1.1. Schéma de principe d'une chaine de régulation [6].

1.5 Régulation analogique

C'est un système, qui fonctionne à partir d'un signal continu spécifique sous forme de tension, nécessaires à la réalisation des régulateurs analogique. Les valeurs des grandeurs physiques constituant les signaux analogiques doivent être représentés par des nombres [7].

La grandeur réglante (la commande) exerce une influence sur la grandeur réglée (sortie) pour la maintenir dans des limites définies malgré les perturbations.

Lorsqu'on compare la mesure de la sortie du système fournit par un capteur à une grandeur de consigne (ou référence), on sera capable d'agir sur la grandeur d'entrée du système, pour en corriger le fonctionnement [8] [9].

On peut représenter un système de régulation par une boucle fermée comme suit :

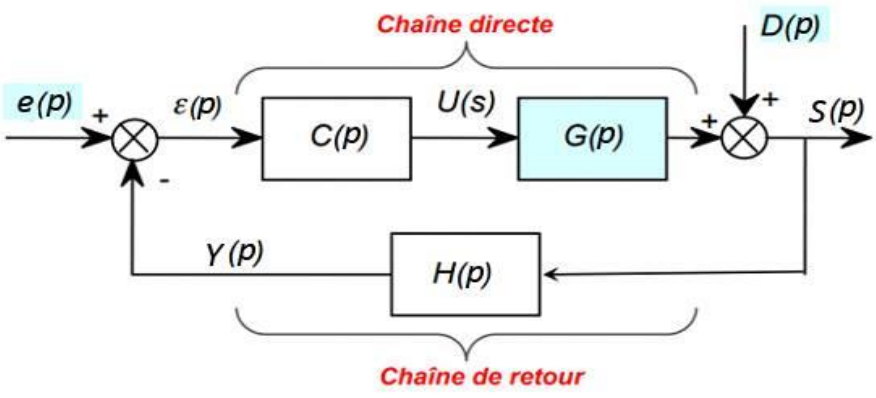


Figure1.2. Schéma bloc d'un système en BF [10].

e(p) : consigne (grandeur d'entrée).

G(s) : fonction de transfert de l'ensemble actionneur + système.

U(p) : signal de commande.

D(p) : fonction de transfert d'une perturbation.

C(p) : fonction de transfert du correcteur.

H(p) : Fonction de transfert la boucle de retour (capteur).

ε (p) : L'écart.

Y (p) : La mesure.

S(p) : Signal de sortie (grandeur de sortie)

1.5.1 Performances attendue d'une régulation

Il s'agit d'analyser la réponse d'un système à un signal, que ce soit lors d'une expérimentation ou d'une simulation. Les critères permettant de qualifier et quantifier les performances du système sont : la stabilité, la précision et la rapidité [4] [11].

a- La stabilité

La qualité essentielle pour un système régulé est la stabilité, un système instable se caractérise soit par des oscillations d'amplitude de plus en plus grande de la grandeur observée (courbe 1 figure 1.3), soit par une croissance irréversible négative ou positive de la grandeur observée (courbe 2 figure1.3). Dans les deux cas, l'objectif de la régulation n'est pas atteint, mais surtout il y'a risque de détérioration physique du procédé.

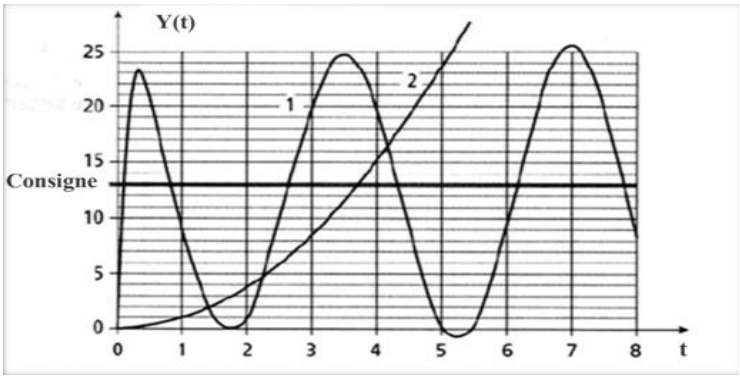


Figure 1.3. Evolution de deux systèmes régulés instables : inacceptable [4].

Dans une approche simplifiée, un système est considéré comme stable si, pour une grandeur à maîtriser se stabilise à une valeur finie. Plus le régime transitoire d'un système soumis à une telle variation est amorti plus il est stable .Le degré de stabilité est alors caractérisé par l'amortissement de ce régime transitoire [4].

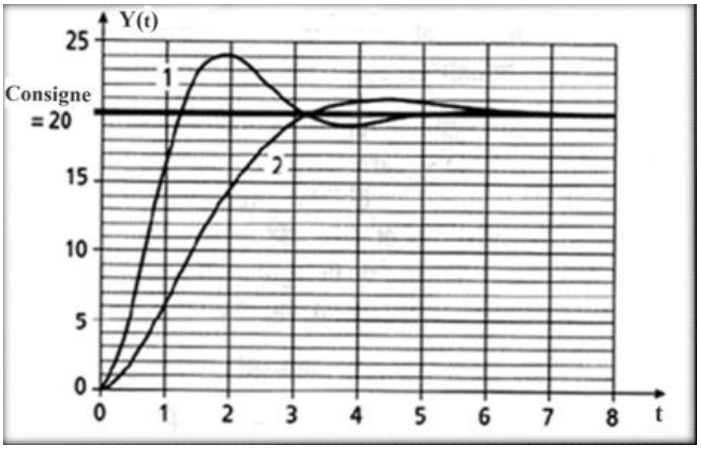


Figure 1.4. Evolution de deux systèmes régulés stables [4].

b- La Précision

Lors du traitement d'une régulation on s'intéresse à la précision statique et la précision dynamique.

– **La précision statique**

La précision statique d'un système régulé se mesure donc à l'écart entre la consigne demandée et la mesure en régime permanant, plus l'écart est petit plus le système est précis. L'évaluation de la précision statique s'effectue en réalisant une variation rapide de consigne en amplitude et en mesurant la variation d'amplitude finalement obtenue de la mesure.

Cet écart est caractérisé par :

$$\boldsymbol{\varepsilon(t) = e(t) - y(t) \quad (Quand\ t \to \infty)} \qquad (1)$$

Selon le théorème de la valeur finale l'écart est définit par:

$$\boldsymbol{\lim_{t\to\infty} \varepsilon(t) = \lim_{p\to 0} p.(e(p) - y(p)) = \lim_{p\to 0} p.\varepsilon(p)} \qquad (2)$$

Il existe différents types d'écarts, en fonction du signal d'entrée :

- Ecart statique :

La réponse indicielle permet la mise en évidence de l'écart statique.

Cet écart peut ne pas être nul, et que des corrections (augmentation du gain, du nombre d'intégrations...) peuvent réduire ou annuler cette erreur.

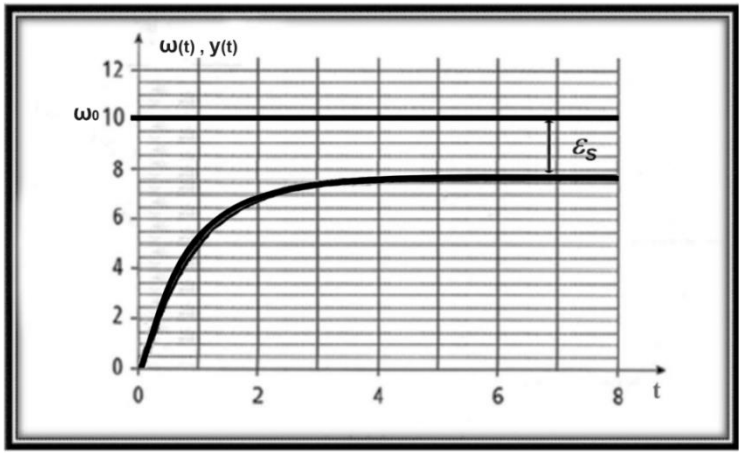


Figure 1.5. Ecart statique relative à la réponse indicielle [4].

- Ecart de traînage (ou de poursuite) :

La réponse à une rampe permet la mise en évidence de l'écart en poursuite d'un système suiveur. Cet écart participe aussi à la précision d'un système, que l'on peut améliorer par des correcteurs.

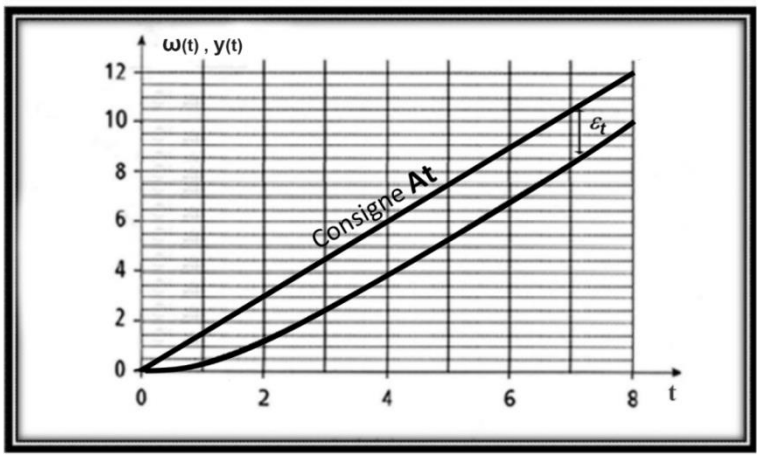


Figure 1.6. Ecart de trainage relatif à la réponse à une rampe [4].

- **La précision dynamique**

La précision dynamique est donc à prendre en compte lors des réglages des régulateurs. Elle s'évaluera généralement par le dépassement maximal D_1 que peut prendre la mesure par rapport à la consigne.

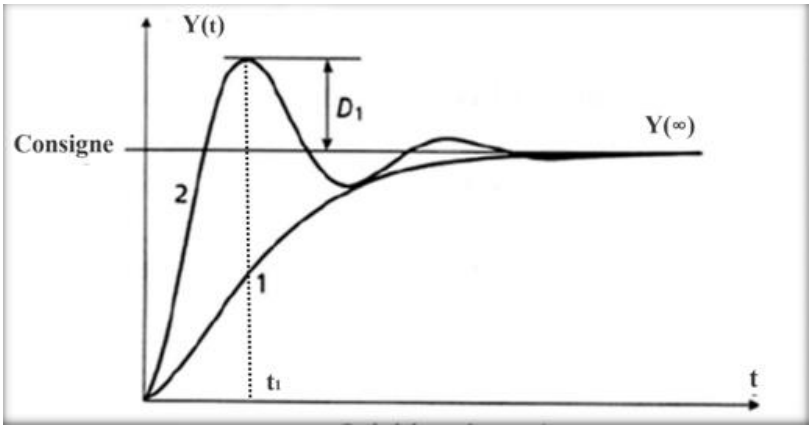


***Figure* 1.7**. Précision dynamique [4].

Le premier dépassement exprimé en % est définit par :

$$\mathbf{D_1}\,\% = \frac{\mathbf{y(t_1)-y(\infty)}}{\mathbf{y(\infty)}} \times \mathbf{100} \qquad (3)$$

D_1 : Premier dépassement.

t_1 : Instant du premier dépassement.

$y(\infty)$: Valeur asymptotique de la sortie en régime permanent.

c- La Rapidité

La rapidité d'un système régulé s'évalue par le temps de réponse à 5% qui est le temps nécessaire à la mesure pour entrer dans une zone ± 5% de sa valeur finale (soit entre 95% et 105%) et aussi par le temps de montée qui est définit par le temps mis par la mesure pour passer de 10% de sa valeur finale `a 90% de cette dernière . Le système régulé est d'autant plus rapide que le temps de réponse à 5% et le temps de montée sont courts [4].

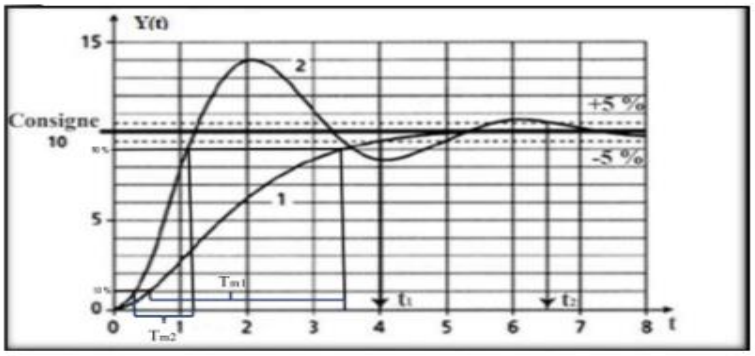


Figure 1.8. Evaluation de la rapidité par le temps de réponse à 5% et le temps de montée [4].

1.5.2 Action élémentaire d'un régulateur PID

Pour optimiser un P.I.D il faut améliorer les actions P.I.D.

a- L'action proportionnelle

La commande de type 'Proportionnelle' est la plus simple qui soit. Il s'agit d'appliquer une correction proportionnelle à l'erreur corrigeant de manière instantanée tout écart de la grandeur a réglé.

Son rôle est d'amplifier l'erreur pour que le système réagisse plus vivement, comme si l'erreur était plus grande qu'elle ne l'est en réalité [12].

Elle agit donc principalement sur le gain du système asservi et permet :

- D'entraîner une augmentation du gain, d'où une diminution de l'erreur statique (amélioration de la précision).
- D'améliorer la rapidité du système.
- D'augmenter l'instabilité du système (donne lieu à des oscillations) [13].

L'équation de commande de l'action proportionnel est donnée par :

$$\mathbf{U(t) = K_p . \varepsilon(t) = K_p . (e(t) - y(t)) = \frac{1}{BP} . \varepsilon(t)} \qquad (4)$$

Ce qui en Laplace donne :

$$\mathbf{U(p) = K_p . \varepsilon(p)} \qquad (5)$$

K_p: Gain proportionnelle.

BP : Bande proportionnelle (étendu de la variation de la mesure exprimé en pourcentage).

b- L'action intégrale

L'action intégrale agit proportionnellement à la surface de l'écart entre la consigne et la mesure, et elle poursuit son action tant que cet écart n'est pas nul [14].

L'intérêt principal de ce correcteur est d'ajouter dans la chaîne de commande une intégration qui augmente la classe du système et annule, selon le type d'entrée, l'erreur statique du système.

L'action intégrale pure permet:

- D'améliorer la précision en annulant l'erreur statique.
- D'introduire un déphasage de -90° qui risque de déstabiliser le système (diminution de la marge de phase).

Le régulateur à action exclusivement intégrale n'est pratiquement jamais utilisé, en raison de sa lenteur et de son effet déstabilisant. Il est en général, associé au correcteur Proportionnel [13].

L'équation de commande de l'action intégrale est donnée par :

$$\mathbf{U(t)=K_i \int_0^t \varepsilon(t).dt = \frac{1}{T_i} \int_0^t \varepsilon(t).dt} \quad (6)$$

Ce qui en Laplace donne:

$$\mathbf{U(p)= K_i . \frac{\varepsilon(p)}{p} = \frac{\varepsilon(p)}{p.K_i}} \quad (7)$$

K_i = Gain d'integral.

T_i : Constante d'intégration.

c- L'action dérivée

C'est une action qui tient compte de la vitesse de variation de l'écart entre la consigne et la mesure, elle joue aussi un rôle stabilisateur [14].

Cette action permet :

- D'améliore la stabilité du système par l'introduction d'un déphasage supplémentaire de + 90° (augmentation de la marge de phase).
- De faire diminuer la précision du système, et amplifie les bruits de hautes fréquences.
- D'augment la rapidité du système (diminution des temps de réponses) [13].

L'équation de commande de l'action dérivé est donnée par **:**

$$\mathbf{U(t)= T_d \frac{d\,\varepsilon(t)}{dt}} \tag{8}$$

Ce qui en Laplace donne :

$$\mathbf{U(p) = K_d . p . \varepsilon(p)} \tag{9}$$

T_d : Constante de dérivation.

K_d : Gain dérivé.

1.5.3 Régulateur PID

Le régulateur standard le plus utilisé en milieu industriel est le régulateur PID (proportionnel intégral dérivé), il permet d'obtenir une régulation optimale en associant les avantages de chaque action. ; La composante P réagit à l'apparition d'un écart de réglage et donne un système plus précis, plus rapide, la composante I élimine l'erreur statique et la composante D s'oppose aux variations de la grandeur réglée et stabilise la boucle de régulation en accélérant la correction [11].

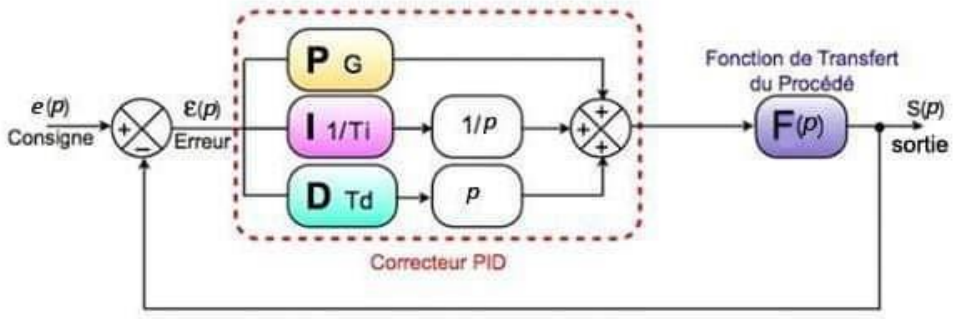


Figure1.9. Schéma bloc d'un système avec un régulateur PID [15].

L'équation de commande de ce correcteur est comme suit :

$$\mathbf{U(t) = K_p.\,\varepsilon(t) + \frac{1}{T_i}\int_0^t \varepsilon(t).\,dt + T_d.\frac{d\,\varepsilon(t)}{dt}} \quad (10)$$

Ce qui en Laplace donne :

$$\mathbf{U(p) = K_p.\,\varepsilon(p) + K_i.\frac{\varepsilon(p)}{p} + K_d.\,p.\,\varepsilon(p)} \quad (11)$$

Les effets de chaque action du régulateur PID peuvent être résumés dans le tableau suivant :

	Temps de montée	Dépassement	Temps d'établissement	Erreur statique
Si K_p croît	Diminue	Augmente	Augmente	Diminue
Si K_i croît	Diminue	Augmente	Augmente	Élimine
Si K_d croît	Peu de changement	Diminue	Diminue	Peu de changement

Tableau 1.1.Effets des correcteurs P, I et D sur les régimes statique et dynamique d'un système en boucle fermé [3].

1.5.4 Les Différentes structures du régulateur PID

Dans un régulateur PID, il existe plusieurs façons d'associer les paramètres P, I et D, en effet, le correcteur PID peut avoir une structure série, parallèle ou mixte [14].

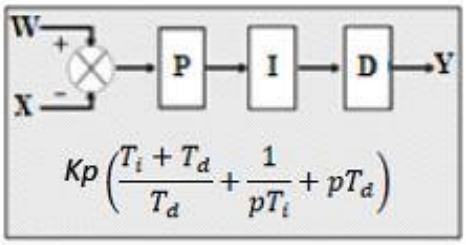
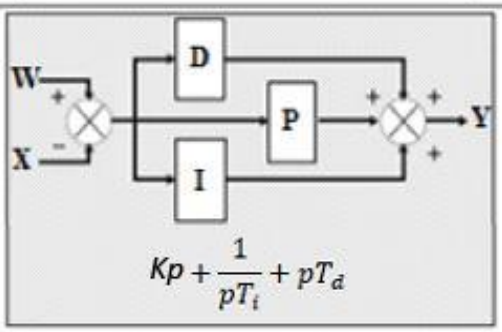
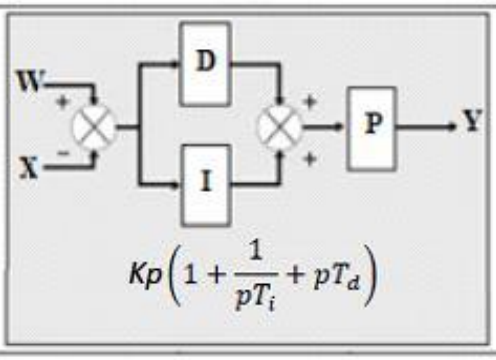
Structure du régulateur PID	Schéma et fonction de transfert
Série	 $Kp\left(\frac{T_i + T_d}{T_d} + \frac{1}{pT_i} + pT_d\right)$
Parallèle	 $Kp + \frac{1}{pT_i} + pT_d$
Mixte	 $Kp\left(1 + \frac{1}{pT_i} + pT_d\right)$

Figure1.10. Différentes structures du régulateur PID [14].

1.5.5 Réglage des coefficients d'un PID

Le réglage d'un PID consiste à trouver les meilleurs coefficients K_p, K_i et K_d dans le but d'obtenir une réponse adéquate du procédé et de la régulation. L'objectif est d'être robuste, rapide, précis et stable, il existe deux façons de procéder, l'une par la modélisation et l'autre par l'expérimentation, parmi les approches expérimentales les méthodes de Ziegler et Nichols et la méthode d'approximations successives [3].

a- Méthodes de Ziegler et Nichols

En 1942, Ziegler et Nichols ont proposé deux approches heuristiques basées sur leurs expériences et quelques simulations pour ajuster rapidement les paramètres des régulateurs P, PI et PID. La première méthode nécessite l'enregistrement de la réponse indicielle en boucle ouverte, alors que la deuxième demande d'amener le système bouclé à sa limite de stabilité [16].

- **Méthode de la réponse indicielle (boucle ouverte)**

Pour obtenir les paramètres du régulateur PID, il suffit d'enregistrer la réponse indicielle du processus seul (c'est-à-dire sans le régulateur), puis de tracer la tangente au point d'inflexion de la courbe. On mesure ensuite les deux grandeurs T_u correspondant au point d'intersection entre l'abscisse et la tangente ainsi que le temps et T_a comme indiqué par la figure (1.12) [16].

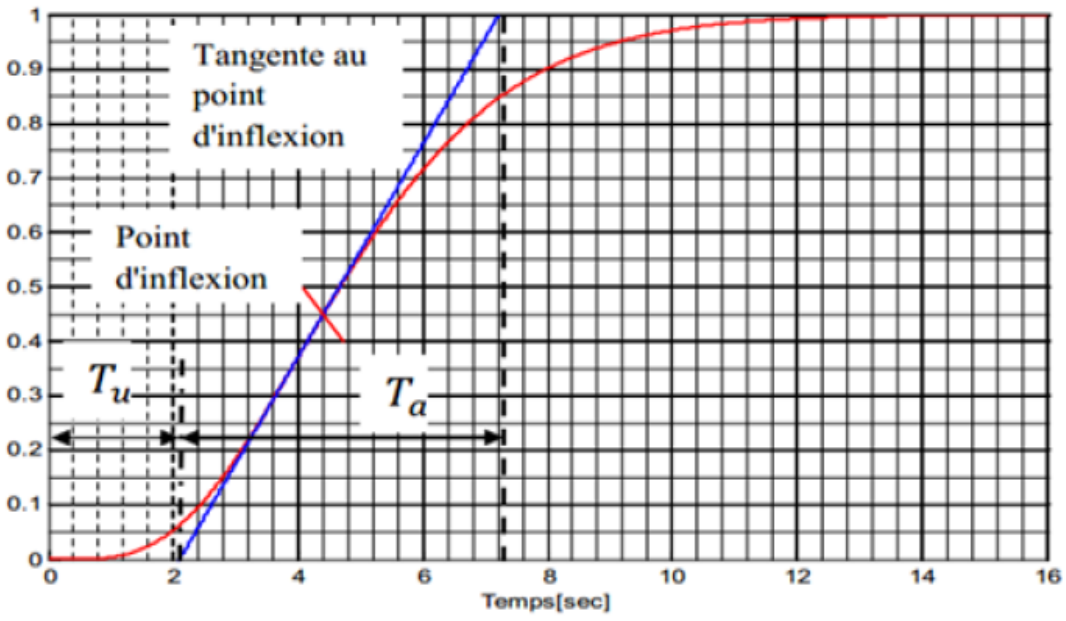


Figure 1.11 . Réponse en boucle ouverte[16] .

Pour les différentes structures du PID, le calcule des paramètres du régulateur choisi est donnée a l'aide de la figure (1.11) et du tableau (1.2), pour les régulateurs P et PI la référence [16] expose le réglage de leurs paramètres :

Type	PID série	PID parallèle	PID mixte
K_p	$0.6\frac{T_a}{T_u}$	$1.2\frac{T_a}{T_u}$	$1.2\ \frac{T_a}{T_u}$
K_i	$\frac{1}{T_u}$	$\frac{T_a}{1.67\ T_u^{\ 2}}$	$\frac{1}{2T_u}$
K_d	T_u	$0.6\ T_a$	$\frac{T_u}{2}$

Tableau 1.2 .Réglages de Ziegler et Nichols en boucle ouverte [16].

- **Méthode du point critique**

Le processus est bouclé sur un simple régulateur proportionnel dont on augmente le gain jusqu'à amener le système à osciller de manière permanente, on se trouve ainsi à la limite de stabilité (figure 1. 12) et Après on relève le gain critique K_{cr} du régulateur et la période d'oscillation T_{cr} de la réponse [16].

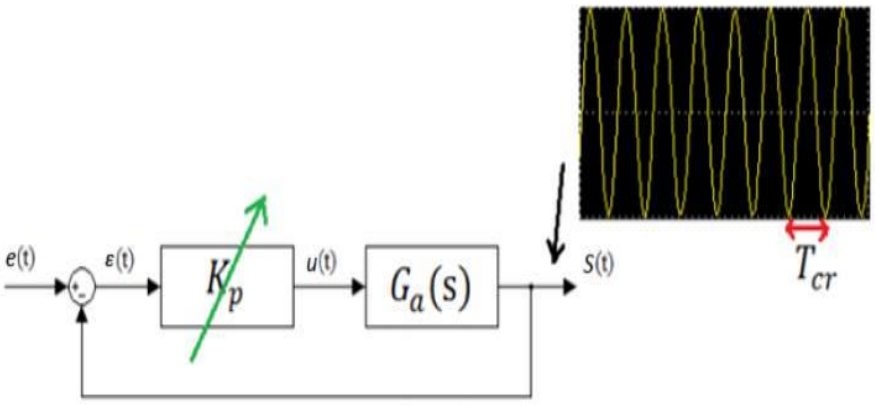


***Figure 1.12*.** Méthode du gain critique [16]**.**

Pour les différentes structures on peut calculer les paramètres du régulateur choisi à l'aide du tableau (1.3). Pour les régulateurs P et PI la référence [16] expose le réglage de leurs paramètres.

Type	PID série	PID parallèle	PID mixte
K_p	0.3 K_{cr}	0.6 K_{cr}	0.6 K_{cr}
K_i	$\frac{4}{T_{cr}}$	$\frac{1.2\ K_{cr}}{T_{cr}}$	$\frac{2}{T_{cr}}$
K_d	$\frac{T_{cr}}{4}$	$\frac{T_{cr}\ K_{cr}}{13.3}$	$\frac{T_{cr}}{8}$

***Tableau 1.3*.** Réglage de Ziegler et Nichols par méthode du gain critique [16].

b- Méthode d'approximations successives

Le réglage des coefficients K_p , K_i et K_d d'un PID est fait expérimentalement par essais/erreurs, il s'agit de régler un paramètre à la fois en suivant quelques règles d'ajustement de ces correcteurs :

- Mettre en place un correcteur P pour améliorer la rapidité du système : modifier K_p pour obtenir le temps de montée voulu tout en faisant attention à la stabilité du système.
- Rajouter un correcteur I pour éliminer l'erreur statique : modifier K_i pour améliorer les performances en régime statique du système.
- Rajouter un correcteur D pour réduire les dépassements et améliorer le temps d'établissement : modifierK_d pour améliorer les caractéristiques en régime transitoire.
- Ajuster K_p, K_i et K_d jusqu'à obtenir les performances voulues [17].

1.6 Régulation numérique

Afin de mettre en œuvre les régulations en milieu industriel, l'usage d'outils informatiques comme organes de contrôle des processus asservis est essentiel. C'est le cas par exemples des ordinateurs ou des microcontrôleurs qui peuvent, entre autre, assumer des fonctions de calculateurs numériques. Mais de tels instruments sont à base de composants électroniques (microprocesseurs, mémoires, ...) et fonctionnent avec des signaux binaires, porteurs d'informations numériques (signaux numériques),

à savoir qu'un outil numérique ne peut s'accommoder de signaux analogiques, pourtant quasi exclusifs dans la majorité des systèmes physiques. En effet, le mode de traitement des informations imposé par un calculateur est de nature numérique et cadencé dans le temps de façon périodique grâce à une horloge [18].

- **Intérêts de la commande par calculateurs :**

- La souplesse d'utilisation du calculateur (machine programmable) à la place d'un correcteur analogique (machine câblée) est remarquable.

- La flexibilité de la programmation permet de réaliser des correcteurs finis, facilement ajustables et auto-ajustables.

- Fourni une grande précision, résous de problème de complexité (grand nombre de paramètres), augmente les rendements, améliore les performances, etc.… [19].

1.6.1 Structure générale d'une régulation numérique d'un processus analogique

L'utilisation des calculateurs numériques utilisés en temps réel pour commander, piloter, guider…des procédés ou systèmes physiques qui par essence sont le plus souvent continus a donné naissance aux systèmes commandés échantillonnés (discrets/numériques). La commande par calculateur d'un procédé nécessite la mise en œuvre d'un certain nombre d'éléments qui sont représenté dans la figure suivante [19] :

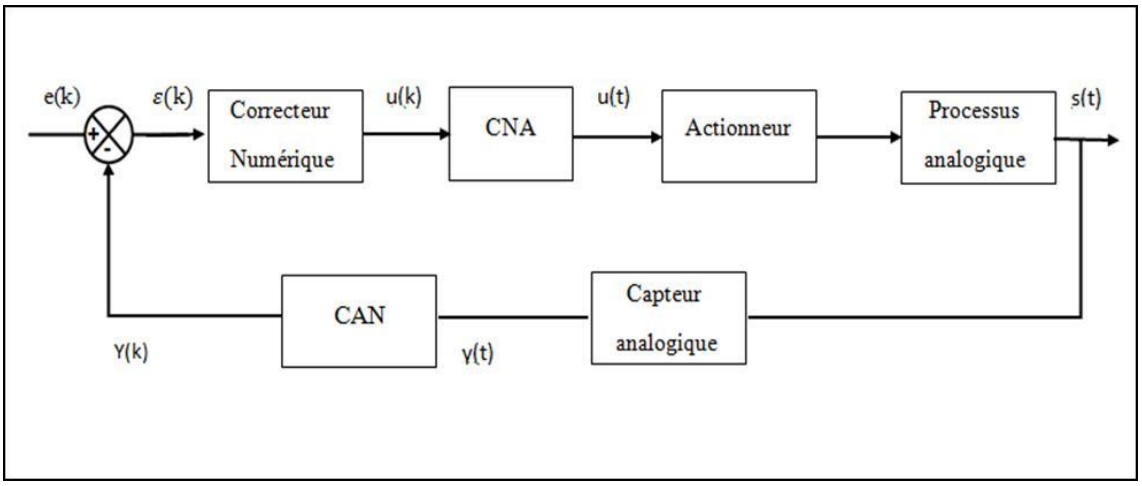


Figure 1.13. Structure typique d'une régulation numérique [19].

1.6.2 Les problèmes à résoudre pour le contrôle des processus analogique par une régulation numérique

Lors de l'implémentation d'un régulateur numériques différentes opérations sont tenues en compte [18].

a) L'échantillonnage d'un signal continu

Cette opération consiste à relever les informations prises par un signal continu à intervalle de temps régulier, appelé période d'échantillonnage. Cela signifie que le calculateur ne tiendra compte que des valeurs prises par le signal aux instants d'échantillonnage.

b) La conversion d'un signal analogique en un signal numérique

Cette opération est faite par un CAN Il s'agit de convertir la valeur prise par un signal analogique à l'instant d'échantillonnage en une valeur numérique afin qu'elle soit traitée par le calculateur, Un tel signal peut, par exemple, provenir d'un capteur.
Le fonctionnement du CAN et décomposé en deux étapes :

- **L'échantillonnage**

L'échantillonneur crée un signal peigné. Il est schématisé par un Interrupteur dont l'ouverture et la fermeture sont cadencées à la période d'échantillonnage par l'horloge du calculateur. En pratique, la fréquence d'horloge est bien plus élevée que la fréquence d'échantillonnage, ceci afin de permettre au calculateur de délivrer le résultat des opérations nécessaires avant l'acquisition de l'échantillon suivant.

- **La numérisation**

Un bloqueur d'ordre zéro maintient chaque échantillon de signal pendant la durée d'échantillonnage. Cela permet au quantificateur d'avoir le temps d'associer à l'échantillon une valeur dans un intervalle de nombres entiers sous forme binaire. Le nombre entier issu du quantificateur est alors associé par le codeur à un autre nombre binaire mais dont la valeur a un sens au regard des coefficients programmés dans le calculateur, c'est-à-dire tenant compte de son signe ou de la virgule flottante.

c) La conversion d'un signal numérique en un signal analogique

Cette opération est faite par un CNA, elle consiste à transformer le signal numérique issu du calculateur à l'instant d'échantillonnage (signal numérique de commande), en signal analogique de commande existant sur toute la période d'échantillonnage, l'objectif étant de commander le système physique. Son fonctionnement est identique au fonctionnement inverse du CAN.

d) La synthèse d'un algorithme de calcul

Il s'agit d'établir une loi d'évolution du signal de commande numérique en fonction des signaux de mesure et de référence, également numériques, afin de permettre au système asservi de satisfaire un cahier des charges. Cette fonction est appelée correcteur numérique. Elle a pour objectif de déterminer la valeur du signal numérique de commande à un instant d'échantillonnage, à partir des valeurs antérieures des signaux numériques de commande, de mesure et de référence. Concrètement, la loi de commande numérique s'exprime comme une relation de récurrence qui permet aisément son implémentation dans un calculateur numérique.

1.6.3 Régulateur PID numérique

Afin de pouvoir importer un régulateur PID analogique dans un calculateur numérique, il faut le transposé (numérisé) en utilisant les différentes techniques de discrétisation. Les régulateurs discrets élaborent une grandeur de commande discrète u(k) en fonction de l'écart de réglage discret $\varepsilon(k)$ du système à commander et du temps d'échantillonnage te. Selon la complexité du régulateur, la grandeur de commande à l'instant k est formée en fonction de la valeur de l'écart à cet instant, mais aussi instants précédents (k-1), etc., les actions du régulateur PID numérique sont comme suit [20] [21] :

a- L'action proportionnelle

Dans le cas discret, la sortie du régulateur est déduite à partir de la relation(4) en passant par la transformée en z on obtient l'équation de récurrence suivante :

$$\mathbf{U(k) = K_p . \varepsilon(k)} \qquad (12)$$

b- L'action intégrale

Pour le cas discret, à partir de la formule (6) et en passant par la transformée en z, on a :

$$\mathbf{U(k) = \frac{T_e}{T_i} \sum_{k=0}^{n} \varepsilon(k) = u(k-1) + \frac{T_e}{T_i} . \varepsilon(k)} \quad (13)$$

c- L'action dérivée

À partir de la formule (8) et en passant par la transformée en z on obtient :

$$\mathbf{U(k) = \frac{T_d}{T_e} (\varepsilon(k) - \varepsilon(k-1))} \quad (14)$$

Le PID numérique est obtenu par l associations des trois actions P, I et D et donc

L'équation de commande générale du PID numérique est la suivante :

$$\mathbf{U(k) = k_p . \varepsilon(k) + \frac{T_e}{T_i} \sum_{k=0}^{n} \varepsilon(k) + \frac{T_d}{T_e} . (\varepsilon(k) - \varepsilon(k-1))} \quad (15)$$

1.7 Conclusion

Dans ce chapitre nous avons présenté les notions de régulations ou d'asservissement en ciblant le régulateur PID, tout en précisant le rôle de chaque paramètre. Nous avons aussi détaillés les deux méthodes empiriques de réglages de ces dernières, en le complétant par la présentation de sa version numérique qui peut être réalisé par un calculateur numérique qui pour notre réalisation sera la carte de prototypage Arduino qui sera développé dans le chapitre suivant.

Chapitre 2 Carte de prototypage Arduino Uno

2.1 Introduction

Apparue il y a seulement quelques dizaines d'année, la commande numérique impose actuellement sa technologie dans le monde, conçue pour piloter le fonctionnement d'une machine à partir des instructions d'un programme sans intervention direct de l'operateur pendant son exécution. Aujourd'hui, de plus en plus étroitement associé aux progrès de la microélectronique et de l'informatique, la CN voit ses performances augmenter régulièrement tandis que, son prix et son encombrement ne cessent de diminuer. Elle s'appelle commande numérique par calculateur [22].

L'Arduino est l'un de ses calculateurs qui englobe l'univers de l'électronique et de la programmation pour offrir un monde créatif à la mode DIY. C'est une carte électronique qui est un croisement entre un ordinateur très simplifié et un automate programmable. En clair, une carte prête à l'emploi et qui peut être programmée pour piloter tout ce que l'on souhaite [23].

Ce chapitre va être consacré au développement du principal élément de commande de notre réalisation, la carte de prototypage Arduino plus précisément la carte Arduino Uno tout en présentant sa partie matérielle ainsi que sa partie logiciel, cette carte va permettre de relier un système physique a un système de commande numérique.

2.2 Historique

Au début des années 2000, Hernando Barragan avait créé un langage de programmation simplifié et l'avait baptisé Wiring. En 2005, Massimo Banzi et David Cuartielles ont conçu un appareil programmable facile à utiliser pour la conception et la réalisation de toutes sortes de projets artistiques interactifs. Le fruit de ce travail, réalisé à l'institut de design interactif d'Ivrea, en Italie, a été nommé Arduino. David Mellis a écrit le logiciel de l'atelier Arduino, directement inspiré de Wiring. Peu de

temps après, Gianluca Martino et Tom Igoe sont venus rejoindre leurs collègues, portant ainsi l'équipe fondatrice à cinq personnes. Le produit devait être simple d'emploi, facile à connecter à toutes sortes de capteurs et d'actionneurs (des relais, des moteurs) et surtout facile à programmer, tout en restant bon marché [24].

2.3 Description de la carte Arduino

La carte de prototypage Arduino est une plateforme open-source d'électronique programmable qui est basée autour d'un microcontrôleur (de la famille AVR), des composants complémentaires et un logiciel, c'est un véritable environnement de développement intégré, pour écrire, compiler et transférer les programmes vers la carte à microcontrôleur. L'Arduino peut être utilisé pour développer des objets interactifs, pouvant recevoir des entrées d'une grande variété d'interrupteurs ou de capteurs, et pouvant contrôler une grande variété de lumières, moteurs ou toutes autres sorties matérielles. Les projets Arduino peuvent être autonomes, ou bien ils peuvent communiquer avec des logiciels tournant sur votre ordinateur (tels que Flash, Processing ou MaxMSP). Ces cartes électroniques peuvent être fabriquées manuellement ou bien être achetées pré-assemblées; le logiciel de développement open-source peut être téléchargé gratuitement [25].

On trouve l'utilisation d'Arduino dans plusieurs applications tels que[23][26] :

- Contrôler des appareils domestiques.
- Donner une "intelligence" à un robot.
- Réaliser des jeux de lumières.
- Permettre à un ordinateur de communiquer avec une carte électronique et Différents capteurs.
- Télécommander un appareil mobile (modélisme).
- Gérer des caméras.
- Commander des moteurs.
- Connaître la température des pièces.
- Allumer ou éteindre une lampe suivant une présence.
- Faire sa propre alarme.

2.4 Les versions des cartes arduino

- Les dites « officielles », qui sont fabriquées en Italie par le fabricant officiel: Smart Projects.
- Les dites « compatibles », qui n'est pas fabriqués par Smart Projects, mais qui sont totalement compatibles avec les Arduino officielles.
- Les « autres », fabriquées par diverses entreprises et commercialisées sous un nom différent (Freeduino, Seeduino, Femtoduino, …) [27].

2.5 Les différentes cartes Arduino

Actuellement il existe plus de 20 types du module Arduino sous différentes tailles et formes, les plus connues et fréquemment utilisé sont Arduino Uno , Arduino leonardo , arduino micro ,Arduino nano , Arduino mega .

Le tableau (2.1) décrit brièvement certaines caractéristiques importantes des cartes susmentionnées :

Arduino	Microcontrô-leur	SRam Ko	EEPROM ko	Flash Ko	Broches D E/S numeriques	Broches d entrée analogiqu e	Frequence du procecesse ur en MHz
Uno	Atmega328	2	1	32	14	6	16 MHz
Nano	Atmega328	2	1	32	14	6	16 MHz
Due	Atmel SAM3X8E	96	0	512	54	12	84 MHz

Mega	Atmega1280	8	4	128	54	16	16 MHz
Leonardo	Atmega32u4	2.5	1	32	20	6	16 MHz
YUN	Atmega32u4	2.5	1	32	20	12	16 MHz
Mega	Atmega2560	8	4	256	54	16	16 MHz
Micro	A tmega32u4	2.5	8	32	20	12	16 MHz
Zero pro	ATSAMD21G 18	32	16	256	14	6	48 MHz

Tableau 2.1. Cartes Arduino et leurs caractéristiques.

La carte Arduino Uno est le choix le plus adéquat pour notre réalisation suite à ses performances qui sont largement suffisantes et son prix abordable.

2.6 La carte Arduino Uno

La carte Arduino Uno comporte tous les composants électroniques nécessaires pour faire fonctionner un microcontrôleur (ATmega328), elle diffère de toutes les cartes car elle n'utilise pas le circuit intégré FTDI USB-vers-série. A la place, elle utilise un Atmega8U2 programmé en convertisseur USB-vers-série [25].

2.6 .1 partie matérielle

Généralement la structure de tout module électronique avec une interface de programmation est toujours basée sur un ou plusieurs circuits programmables, La carte Arduino UNO dispose [28] :

1. D'un microcontrôleur ATmega328.

2. D'une connexion USB (alimentation et le transport des données).

3. D'un connecteur d'alimentation jack.

4. LED de visualisation.

5. a) De 14 broches numériques d'entrées/sorties.

5. b) De 6 entrées analogiques.

6. D'un quartz 16Mhz.

7. D'un connecteur ICSP (programmation "in-circuit").

8. Un bouton de réinitialisation (reset).

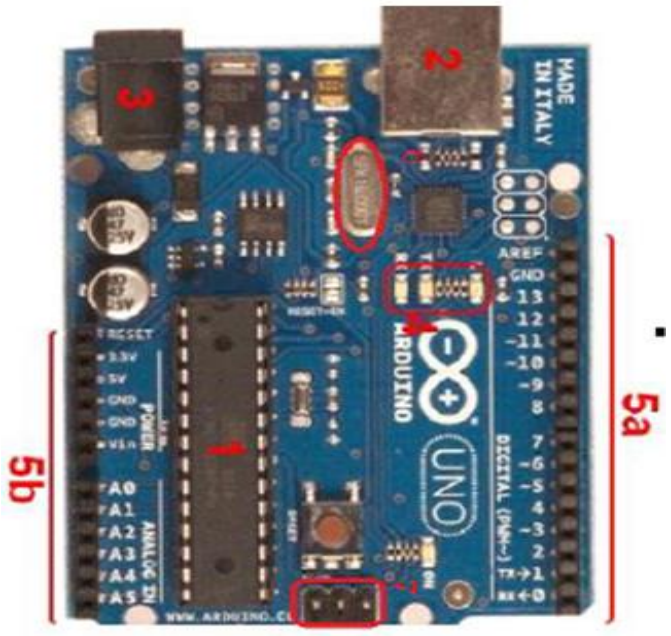


Figure2.1. Brochage de la Carte Arduino UNO [25].

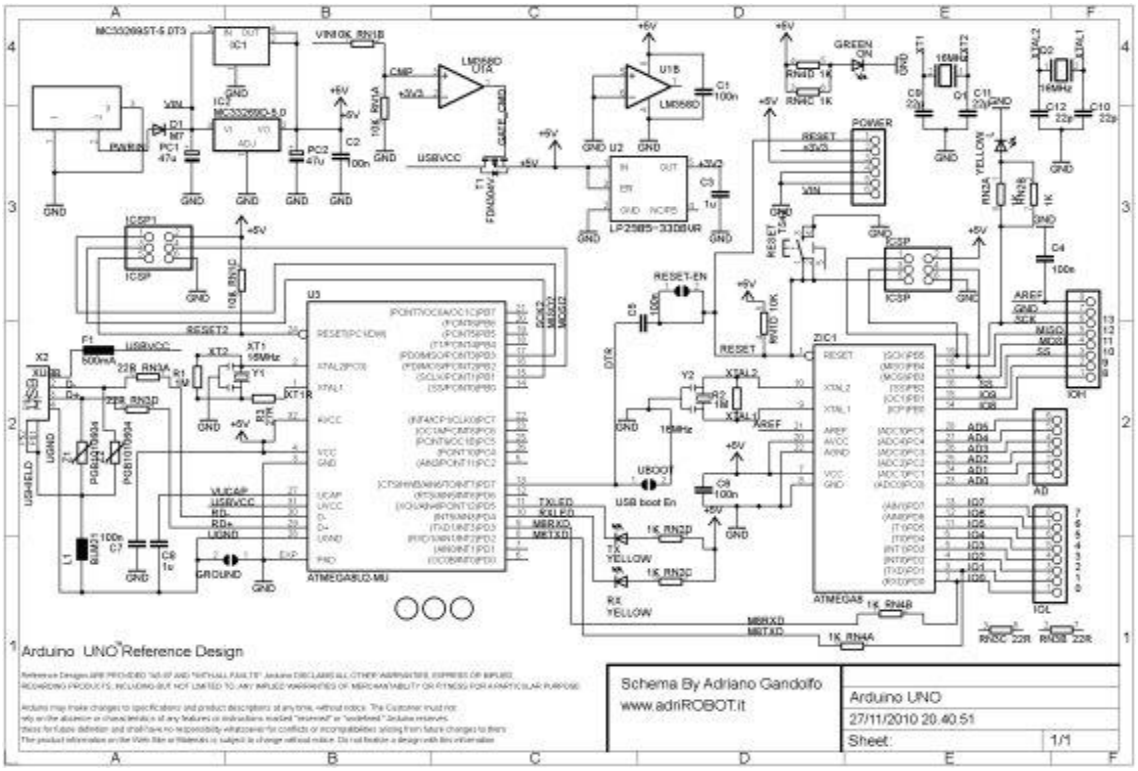


Figure2.2. Le schéma électronique de la carte Arduino Uno[29].

a. Le microcontrôleurAtmega328

C'est le cerveau de la carte. Il va recevoir le programme crée et va le stocker dans sa mémoire avant de l'exécuter.

Un microcontrôleur (Atmega328) est un circuit intégré(ou IC, Integrated circuit), qui rassemble sur une puce plusieurs éléments complexes dans un espace réduit. Au temps des pionniers un grand nombre de composants encombrant était soudé, tels que les transistors, les résistances ou les condensateurs sur des cartes plus ou moins grande. Aujourd'hui tout peut loger dans un boitier en plastique noire muni d'un certain nombre de broches, ces dernières sont les connexions du circuit intégré au moyen des quelles s'effectue la communication [28] [30].

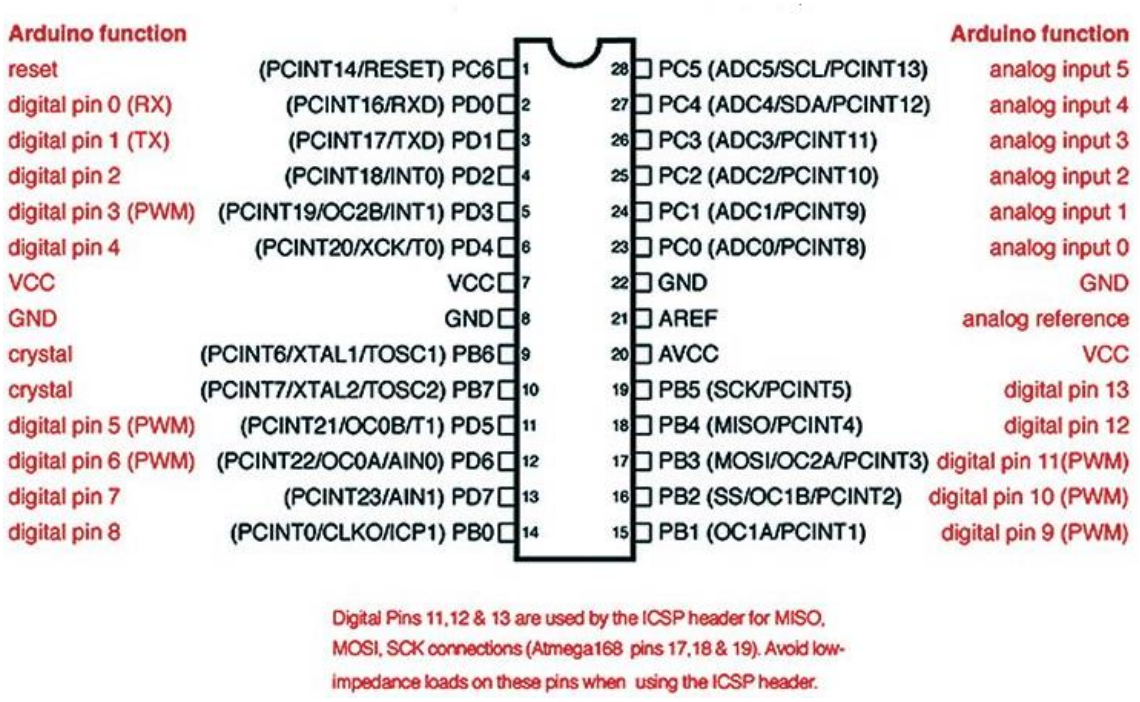


Figure 2.3 .Brochage du microcontrôleur Atmega328 [31].

Le microcontrôleur ATMega328 est constitué par un ensemble d'éléments qui ont chacun une fonction bien déterminée. Il est en fait constitué des mêmes éléments que sur la carte mère d'un ordinateur. L'architecture interne de ce circuit programmable se compose essentiellement sur :

- **La Mémoire**

La mémoire Flash: C'est celle qui contiendra le programme à exécuter. Cette mémoire est effaçable et réinscriptible (32 Ko).

RAM : C'est la mémoire dite "vive", elle va contenir les variables de votre programme. Elle est dite "volatile" car elle s'efface si on coupe l'alimentation du microcontrôleur (2 Ko).

EEPROM : C'est le disque dur du microcontrôleur. Les informations enregistrées sont celles 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 (1 Ko).

Les registres : C'est le type de mémoire utilisé par le processeur.

- **Le processeur**

C'est le composant principal du microcontrôleur. C'est lui qui va exécuter le programme traité. Il est nommé souvent par le CPU et il est composé d'une unité arithmétique et logique(UAL), d'un bus de données, d'adresse et de commande [32].

- **Les Timers:**
  - Timer0 (TCCR0B) : génère le signal PWM sur les pins 5 et 6.
  - Timer1 (TCCR1B) : génère le signal PWM sur les pins 9 et 10.
  - Timer2 (TCCR2B) : génère le signal PWM sur les pins 11 et 3 [33].
  - **Les ports entrés sortie.**
  - **Les conversions A/N.**

b. L'alimentation

La carte Arduino UNO peut être alimentée par l'USB (5v jusqu'à 500 ma) ou par une alimentation externe. La source est sélectionnée automatiquement. La tension d'alimentation extérieure (hors USB) peut venir soit d'un adaptateur AC-DC ou de piles. L'adaptateur peut être connecté grâce à un 'jack' de 2.1mm positif au centre. Le raccordement vers un bloc de piles peut utiliser les bornes GND et Vin du connecteur d'alimentation (POWER). La carte peut fonctionner à l'aide d'une tension extérieure de 7 à 12 volts. Les broches (pins) d'alimentation sont les suivantes :

- **VIN** : La tension d'entrée positive lorsque la carte Arduino est utilisée avec une source de tension externe (à distinguer du 5V de la connexion USB ou autre source 5V régulée). La carte peut être alimenté à l'aide de cette broche, ou, si l'alimentation est fournie par le jack d'alimentation, accéder à la tension d'alimentation sur cette broche.
- **5V** : La tension régulée utilisée pour faire fonctionner le microcontrôleur et les autres composants de la carte (les circuits électroniques numériques nécessitent une tension d'alimentation parfaitement stable dite « tension régulée » obtenue à l'aide d'un composant appelé un régulateur et qui est intégré à la carte Arduino). Le 5V régulé fourni par cette broche peut donc provenir soit de la tension d'alimentation VIN via le régulateur de la carte, ou bien de la connexion USB (qui fournit du 5V régulé) ou de tout autre source d'alimentation régulée.

- **3.3V** fournie par le circuit intégré FTDI (circuit intégré faisant l'adaptation du signal entre le port USB de l'ordinateur et le port série de l'Atmega) de la carte est, ceci est intéressant pour certains circuits externes nécessitant cette tension au lieu du 5V). L'intensité maximale disponible sur cette broche est de 50mA .
- **GND** Broche de masse (ou 0V)[34].

c. Entrées et sorties numériques

Chacune des 14 broches digitales de la Uno (de 0 a 13) peut être utilisée en input ou output, en utilisant les fonctions pinMode() , digitalWrite() , et digitalRead().

Elles fonctionnent en logique 0V-5V ; chacune pouvant fournir (source) ou recevoir un courant maximal de 40 mA et dispose si besoin d'une résistance interne de 'pull-up'.

En outre, certaines broches ont des fonctions spécialisées:

• **Interruptions externes 2 et 3**. Ces broches peuvent être configurées pour déclencher une interruption sur une valeur LOW, sur un front montant ou descendant, ou encore sur le changement de valeur. (voir la fonction attachInterrupt() pour des détails).

• **PWM (pulse width modulation)**: les broches 3, 5, 6, 9, 10, et 11 peuvent fonctionner en mode PWM avec la fonction analogWrite().

• **SPI:** 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Ces broches fournissent le support de communication SPI en utilisant la 'library' spécialisée.

•**I2C**: 4 (SDA) and 5 (SCL). Permettent le support de communication I2C (TWI) en utilisant le 'library' Wire.

• **LED:** 13. Il y a une LED connectée à la broche digitale 13 [33].

d. Entrées analogiques

La carte UNO dispose de 6 entrées analogiques (A0 à A5), chacune pouvant fournir une mesure d'une résolution de 10 bits (de 0 à 1023) a l'aide de la fonction analogRead ( ) du langage Arduino.

Par défaut, ces broches mesurent une tension comprise entre le 0V (valeur 0)et le 5 V (valeur 1023). Mais il est possible de modifier le niveau supérieur en utilisant la broche AREF et l'instruction analog Reference ( ) du langage Arduino [26].

Il ya d'autres broches disponibles sur la carte:

- **AREF** : tension de référence.
- **Reset** : Permet au niveau bas (LOW°) de faire un reset du contrôleur. Elle est utilisée typiquement pour monter un bouton 'reset' aux cartes additionnelles ('shields') bloquant celui de la carte principale [33].

e. Les ports de communication

L'Arduino Uno possède de nombreuses possibilité de communication avec l'extérieur, parmi eux la communication série matérielle (Rx : broches numériques N°0 et Tx : broche numérique N°1) qui permet d'établir la communication avec l'ordinateur via le port USB ou d'autres cartes. C'est cette liaison qui permet d'émettre ou de recevoir des messages depuis le moniteur série, les broches Rx et Tx sont raccordées à leurs homologues sur le chip Atmega8U2 spécialisé dans la conversion USB to TTL série.

Les liaisons séries, en général, sont des moyens de transport d'informations entre divers systèmes numériques, les différents bits d'une donnée ne sont pas envoyés en même temps mais les uns après les autres, ce qui limite le nombre de fils de transmission.

Seuls deux composants peuvent communiquer avec un câble croisé. La sortie Tx de l'un doit être reliée à l'entrée Rx de l'autre. Bien entendu, les deux éléments doivent avoir également une masse commune.

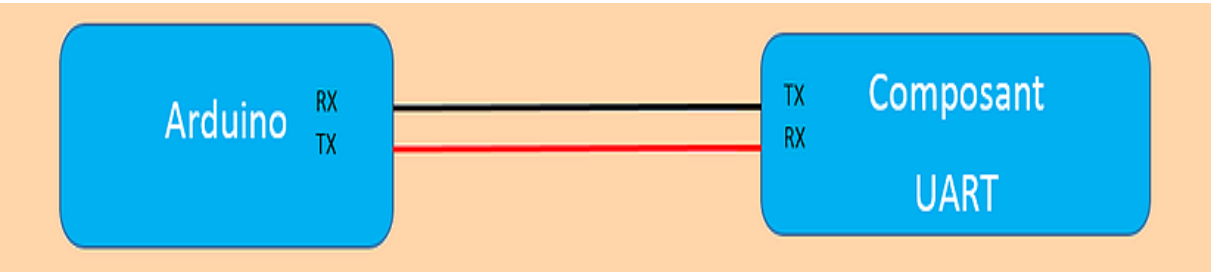


Figure2.4. La liaison série [35].

Le protocole d'échange asynchrone est défini par l'envoi grâce à un UART (Universal Asynchronous Receiver Transmitter), Dans la technologie Arduino, tout est simplifié au maximum et la trame est constituée de :

- Un bit de Start.
- Un bit de parité
- Les 8 bit de données.
- Un bit de Stop.

Lorsque des informations sont transmises par une liaison série, elles ne sont transmises que par des 0 et des 1, soit un nombre. Dans le cas de ce type de liaison, la transmission des informations est faite par paquet de 8 bits, Les caractères envoyés sont codés en ASCII [35].

2.6.2 Partie logicielle

Le logiciel de programmation des modules Arduino est une application Java, libre et multiplateformes, servant d'éditeur de code et de compilateur, et qui peut transférer le firmware et le programme au travers de la liaison série (RS232, Bluetooth ou USB selon le module). Il est également possible de se passer de l'interface Arduino , et de compiler les programmes en ligne de commande. Le langage de programmation utilisé est le C++, compilé avec avr-g++, et lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties. La mise en place de ce langage standard rend aisé le développement de programmes sur les plates-formes Arduino, à toute personne maitrisant le C ou le C++[36].

a. Structure générale de l'IDE Arduino

L'interface de l'IDE Arduino est plutôt simple, il offre une interface minimale et épurée pour développer un programme sur les cartes Arduino. Il est doté d'un éditeur de code avec coloration syntaxique(1) et d'une barre d'outils rapide(2). Ce sont les deux éléments les plus importants de l'interface, c'est ceux que l'on utilise le plus souvent. On retrouve aussi une barre de menus (3) plus classique qui est utilisé pour accéder aux fonctions avancées de l'IDE. Enfin, une console(4) affichant les résultats de la compilation du code source, des opérations sur la carte, etc... [37].

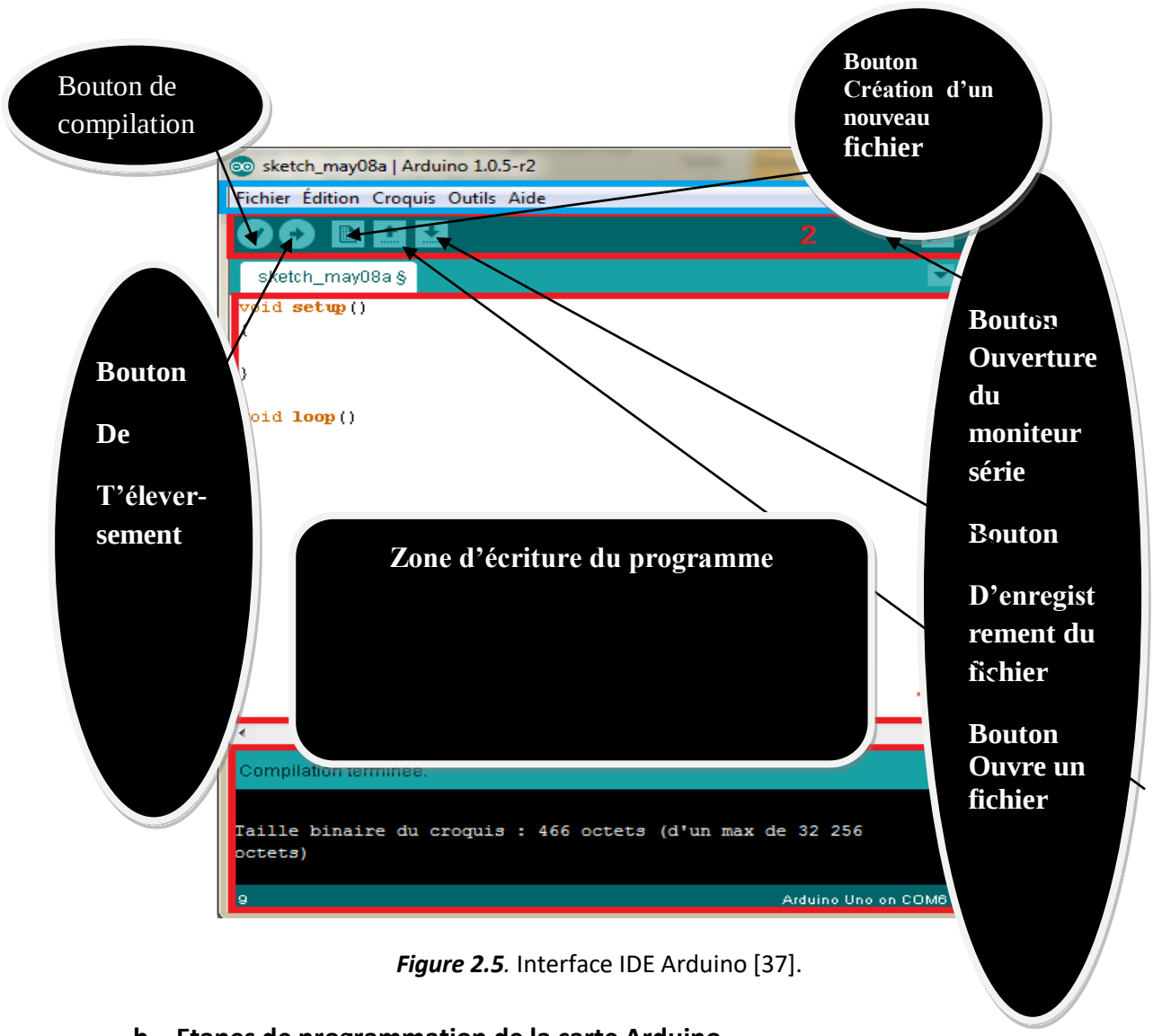


Figure 2.5. Interface IDE Arduino [37].

b. Etapes de programmation de la carte Arduino

- **Configuration de la carte**

Pour établir le dialogue entre la carte et l'atelier, il faut définir deux paramètres : le type de carte et le nom du port de communication USB.

Tout d'abord la carte doit être branché via un cable USB, ensuite il faut ouvrir le menu Outils puis le sous-menu Type carte, il est possible que le modèle Arduino/ Genuino Uno soit déjà sélectionné. Si ce n'est pas le cas, il faut sélectionner ce modèle de carte [38].

La sélection du numéro du port USB (COM) est faite depuis le menu outils →port le nom du port diffère selon le type système (Windows, linux …).

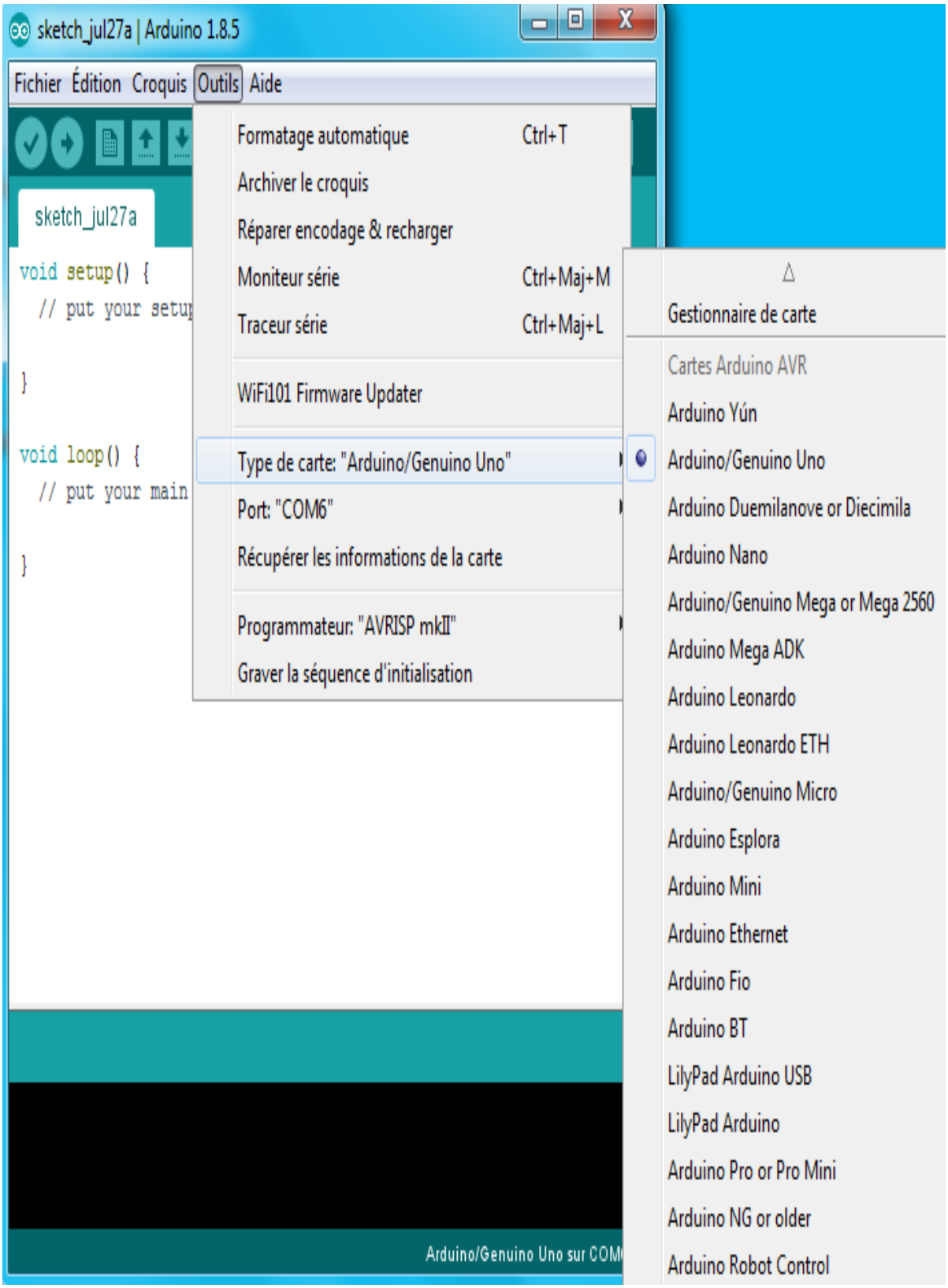


Figure 2.6 : Choix du type de carte.

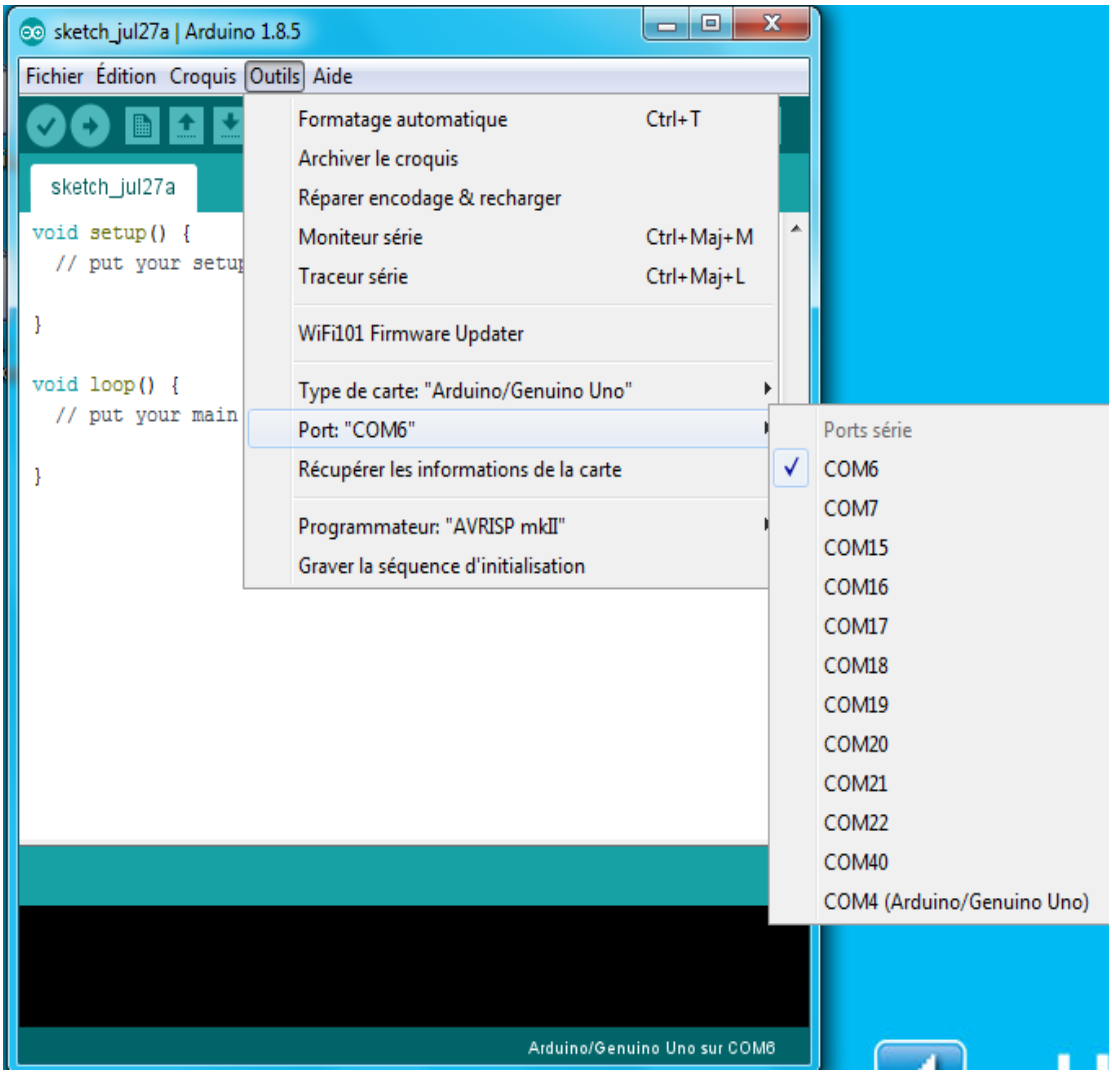


Figure 2.7. Choix du port série.

- **Ecriture du programme**

Un programme est une suite d'instructions, sa structure se décompose en 3 parties principales :

- La partie déclaration des variables et des constantes.
- La partie void setup ( ) : c'est la partie du programme qui ne va s'exécuter qu'une seule fois, à la mise en route (ou après un reset, ou quand on se connecte au port COM). On y écrit généralement les fonctions d'initialisation (fonction principale)
- La partie void loop( ) : c'est une boucle infinie dans laquelle le programme écrit s'exécute tant que la carte est alimentée. Le programmeur peut également écrire ses propres fonctions annexes (fonction boucle) [39].

Un programme complet peut rassembler plusieurs boucles et fonctions, ces dernières peuvent contenir des instructions qui appellent d'autres fonctions.

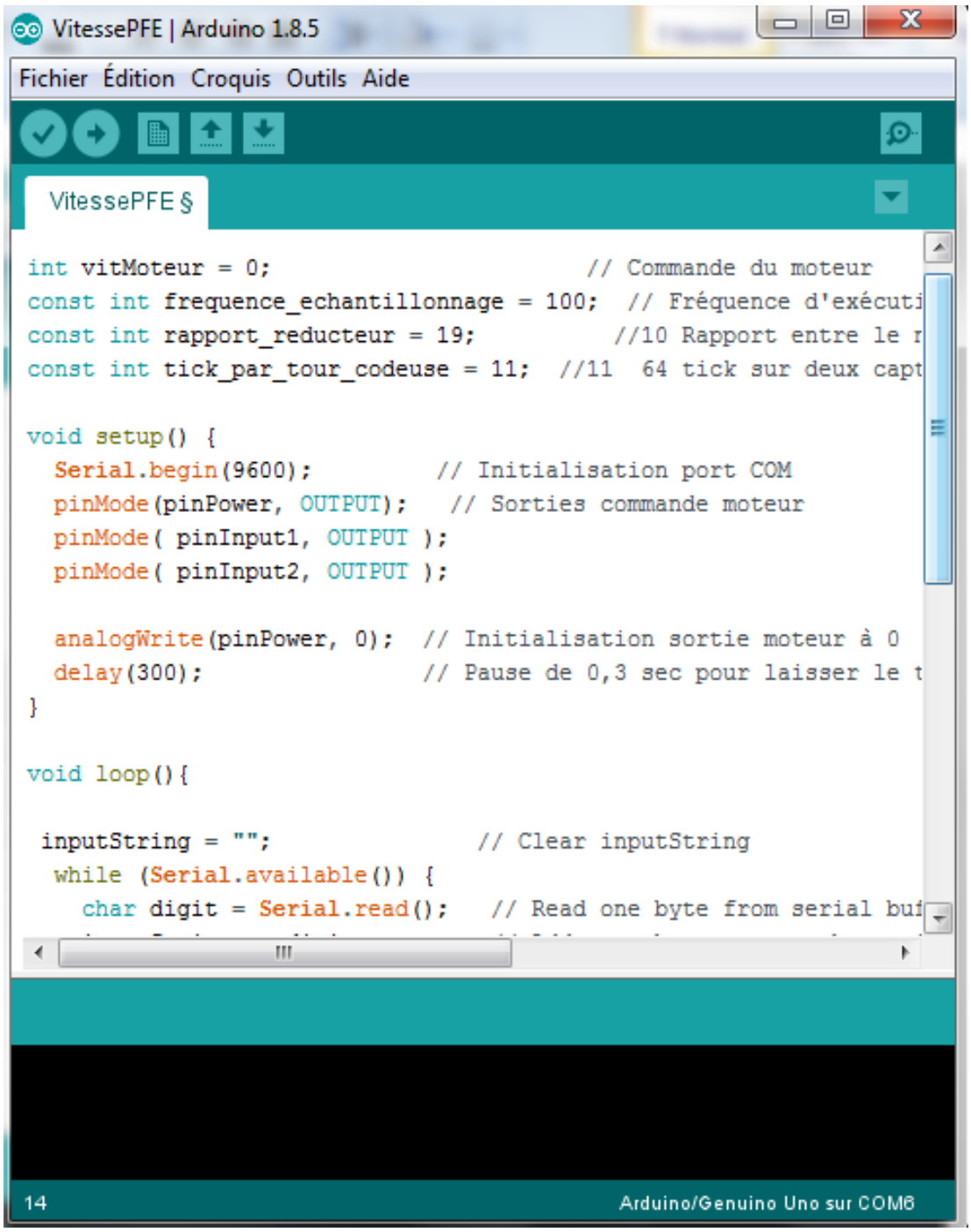


Figure 2.8.Exemple d'un programme.

-Compilation du programme

La compilation est la vérification du programme après l'avoir écrit, si le programme est correct un message s'affichera sur la console ' compilation terminée' ainsi que la taille du programme, s'il contient des erreurs un message d'erreur s'affichera en indiquant le type d'erreur et la ligne correspondante. Cette compilation sert aussi à la conversion du langage de programmation utilisé en langage machine (0 et 1) compréhensible par l'Arduino.

- Téléversement du programme

C'est le transfert du programme compilé vers la carte Arduino (mémoire du microcontrôleur), tant que l'Arduino est alimenté le programme est stocké en mémoire et sera exécuté, les leds Tx et Rx clignoterons en témoignage de la réception des informations.

2.7 Les avantages et les inconvénients de la carte Arduino

Au cours des dernières années, l'utilisation d'Arduino a augmenté de façon exponentielle en raison de sa lisibilité et de sa facilité. L'utilisation d'Arduino présente certains avantages et inconvénients [40].

2.7.1 Les avantages

- Pas cher : les cartes Arduino sont relativement peu coûteuses comparativement aux autres plateformes.
- 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.
- Un environnement de programmation clair et simple: L'environnement de programmation Arduino est facile à utiliser, tout en étant assez flexible pour que les utilisateurs avancés puisse en tirer profit également.
- Matériel Open source et extensible : Les cartes Arduino sont basé sur les microcontrôleurs Atmel ATMEGA8, ATMEGA168, ATMEGA 328, etc... Les schémas des modules sont publiés sous une licence Créative Commons, et les

concepteurs de circuits expérimentés peuvent réaliser leur propre version des cartes Arduino, en les complétant et en les améliorant.

- 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 [41].
- Le système Arduino s'avère ingénieux et propice à la créativité, car il permet véritablement de créer un projet unique et plus ou moins abouti en fonction du temps consacré. Quoi qu'il en soit, la carte Arduino Uno permet de programmer une multitude de choses et de s'adonner à des projets divers et unique comme des projets électriques, microbiologique, domestique [42].

2.7.1 Les inconvénients

- Il y a plusieurs composants sur les cartes qui sont inutiles pour certains projets, Donc il peut encore y avoir optimisation.
- Le langage Arduino est unique, donc le code ne sera portable que sur des cartes de type Arduino.[43]
- L'absence d'une option de connectivité intégrée ce qui limite les utilisations possibles de l'Internet des objets.
- Une mémoire embarquée limitée qui rend les programmes complexes difficiles à réaliser [44].

2.8 Conclusion

Dans ce chapitre, nous avons détaillé et explicite les principaux fonctions du calculateur que nous avons choisi, à savoir la carte de prototypage Arduino ainsi que sa partie logiciel, ce module est un outil Complet qui dispose de toute les fonctionnalités essentielles à un traitement ou la communication avec des interfaces comme le PC.

Apres avoir développé dans le premier chapitre les fondements théorique nécessaires sur la régulation et présenté dans le deuxième chapitre le dispositif permettant d'élaborer une commande le chapitre suivant sera consacré a la réalisation d'une régulation numérique par l'Arduino.

Chapitre 3 Réalisation et Interprétations

3.1 Introduction

Ce chapitre présentera de manière explicite une vue d'ensemble du dispositif expérimental pour élaborer une régulation en position d'un moteur à courant continu en commençant par le synoptique, puis en détaillant chaque élément de ce dernier pour ensuite entamé la programmation sous Arduino et l'interface de contrôle et finir par l'obtention des résultats tout en les interprétant.

3.2 Synoptique général

Le schéma synoptique général de notre dispositif est donné par la figure (3.1).

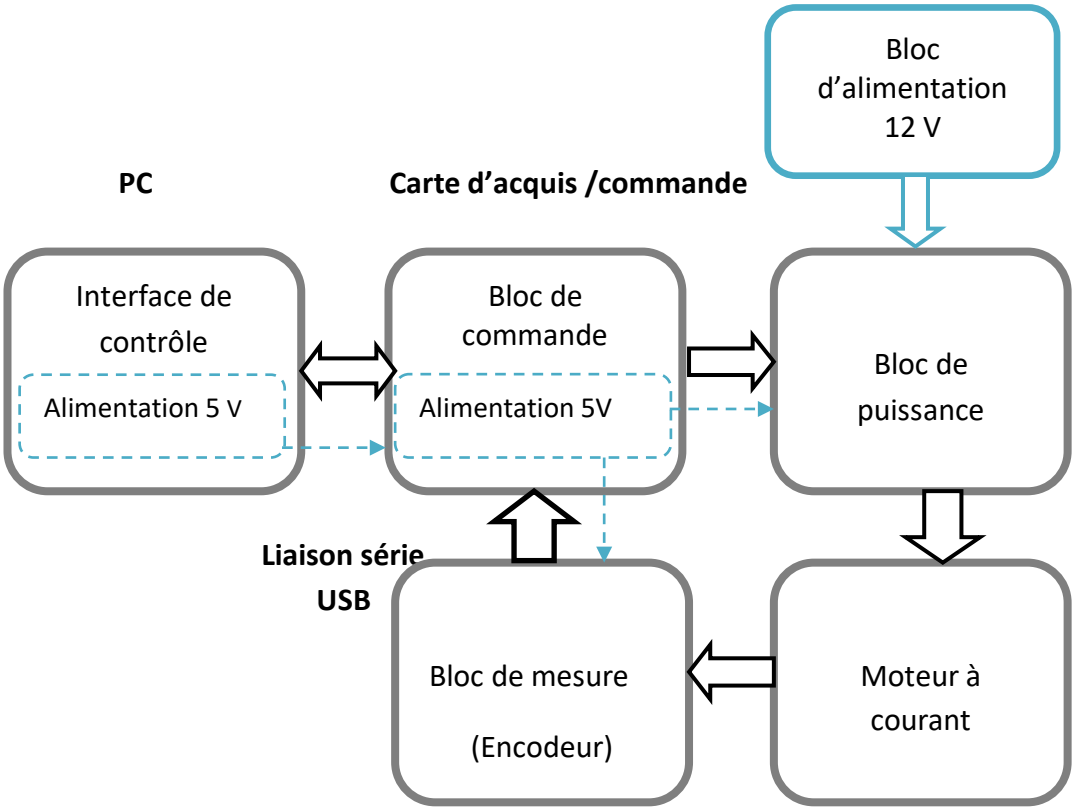


Figure3.1. Schéma synoptique général du projet.

3.2.1 Bloc de commande

Le bloc d'acquisition et de commande, représente le cœur de notre travail il s'articulera autour de la carte de prototypage Arduino Uno, ce dernier a été détaillé précédemment dans le chapitre 2, la communication avec l'ordinateur est assuré par la liaison série et la commande est assuré par une sortie PWM.

Pour la régulation en position, on a implémenté un régulateur PID sur ce calculateur numérique, ce régulateur permet de maintenir la grandeur à réguler à la valeur prédéfinie de la consigne malgré la présence des perturbations et ceci en délivrant un signal de commande en forme de tension analogique (vu par le moteur) qui s'annule quand la consigne donnée est atteinte tout en respectant des critères qualitatifs.

La commande varie en fonction des erreurs entre la consigne et la mesure du capteur de position ainsi que les paramètres K_p, K_i et K_d qui assure la correction.

Pour notre travail nous avons réalisé une interface graphique, (qui sera détaillé par la suite) nous permettant de transmettre la consigne ainsi que les paramètres de l'interface de control via la liaison série vers notre calculateur, où la mesure de la position se fera grâce à l'encodeur dont le moteur en est muni.

Cette commande est à la fois proportionnelle à l'erreur, à la somme des erreurs et à la variation de l'erreur, elle est transmise au circuit de puissance à travers une Pin PWM du calculateur et son équation peut être modélisé par la formule ci-dessous :

Commande Moteur = K_p* erreur + K_i* somme des erreurs + K_d*variation de l'erreur (16)

Erreur = consigne-mesure.

Somme des erreurs = $\sum$ erreur.

Variation de l'erreur=erreur –erreur précédente.

- **Technique PWM (pulse width modulation)**

La modulation à largeur d'impulsion (MLI ou PWM) est une technique pour obtenir des effets de tensions analogiques avec des broches numériques à la sortie de notre

carte de commande. Elle consiste a généré un signal carré, basculant entre un niveau HAUT (5V) et BAS (0) avec un rapport cyclique modulable.

Le signal PWM est codé sur 8 bits il dispose de 256 valeurs possibles de 0 à255 (2^8= 256) , d'une fréquence fixe (500 Hz) et un rapport cyclique qui varie.

Le rapport cyclique est le pourcentage du temps de la période (2ms) durant laquelle le signal est au niveau logique 1 (5v), il est possible de déterminer pour chaque valeur PWM la valeur du rapport cyclique qui représente la tension moyenne qui attaque le moteur, cette tension variera proportionnellement au rapport cyclique de notre PWM.

PMW=0 ↔ rapport cyclique = 0% ↔ tension =0 V.

PWM=64 ↔ rapport cyclique = 25% ↔ tension =1,25 V.

PWM=127↔ rapport cyclique = 50% ↔ tension =2,5 V.

PWM=191 ↔ rapport cyclique = 75% ↔ tension =3,75V.

PWM=255 ↔ rapport cyclique = 100% ↔ tension =5 V.

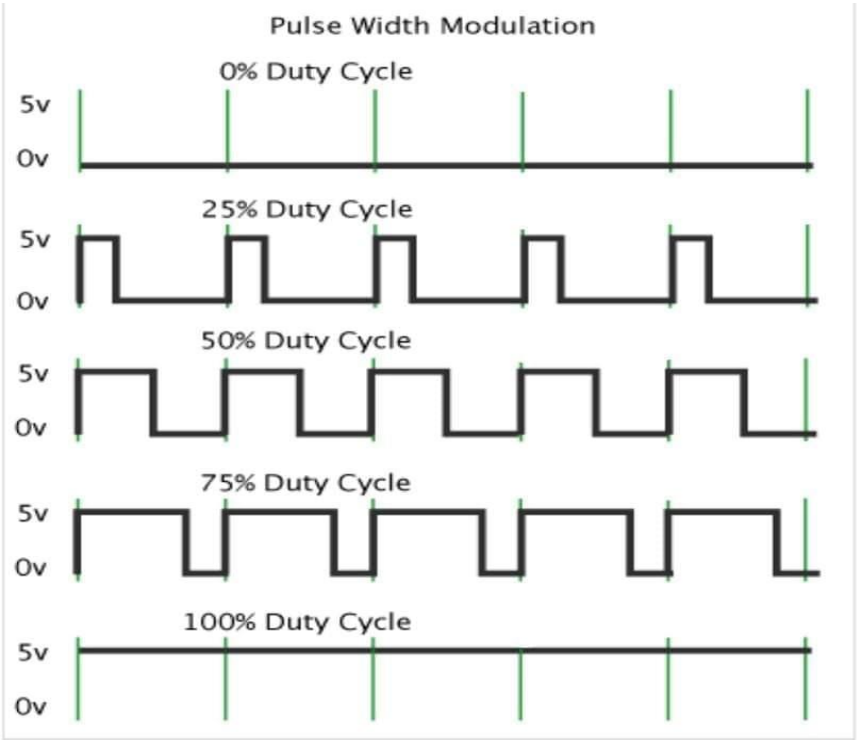


Figure 3 .2. Signaux aux rapports cycliques differents [45].

3.2.2 Bloc de puissance

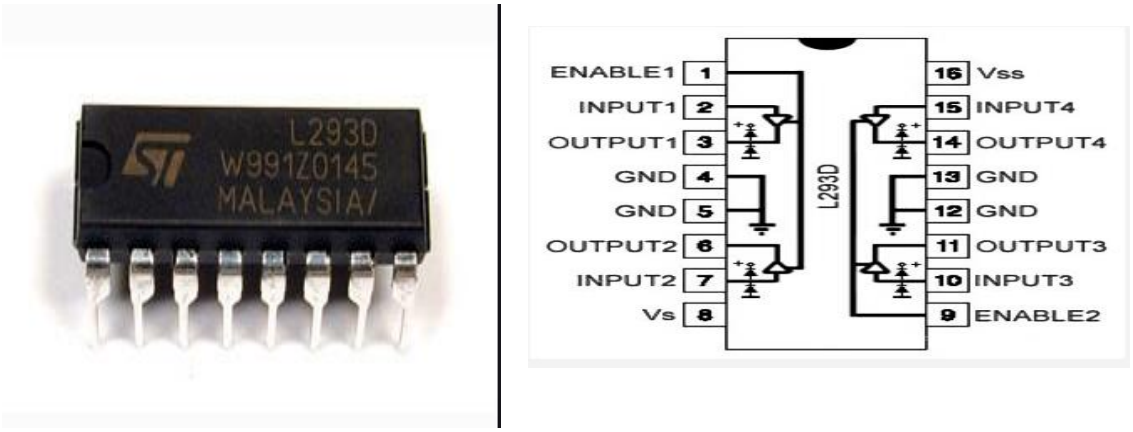
Sachant que les tensions et les courant gérer par le calculateur sont de faible puissance nous avons introduit entre la carte Arduino et notre actionneur (Moteur CC) un bloc de puissance qui permettra d'alimenter la charge avec une puissance adéquate.

Le bloc se base sur un circuit intégré '' L293D'', qui est équipé d'un double pont en H (un pont en H de chaque côté) qui permet de piloter 4 moteurs en un sens unique ou 2 moteur dans les deux sens de rotation et permet aussi de faire varier la vitesse de rotation suivant l'activation de la pin Enable par une attaque de la PWM.

Un pont en H possède 4 interrupteurs commandés (transistors NPN et PNP) qui fonctionnent deux par deux et est équipé de diodes a roues libres (le D dans L293D signifie diode) qui servent à la protection des transistors contre le retour de courants néfastes du moteur en particulier lorsqu'il s'arrête, ces transistors permettent d'aiguiller la polarité de la source d'alimentation des moteurs.

Ce circuit permet la transmission de l'énergie nécessaire pour faire fonctionner des moteurs, il supporte des tensions allant de 4 V à 36 V (courant maximal 0,6 A), dédié à alimenter les moteurs.

Le brochage du L293D et représenté dans ce qui suit :



(a) Boitier du L293D. (b) Structure interne de L293D ***Figure 3.3.***Représentation du L293D [46].

Le tableau (3.1) présente les différentes broches du L293D ainsi que leurs configurations :

Broches	Configurations
Enable 1 Enable 2	contrôlent l'activation /désactivation des deux ponts en H . (Enable 1 pour le premier pont, Enable 2 pour le deuxième) Permettent la réception des informations envoyées depuis pin PWM de l'Arduino Et la variation de l'alimentation du premier et deuxième moteur.
Input 1 Input 2	Commandent le premier pont en H elles sont branché sur l'Arduino pour commander la polarisation de l'attaque du moteur et donc le sens de rotation du premier moteur.
Output 1 Output 2	Elles sont raccordées aux branches d'alimentation du premier moteur.
GND	Doit être raccordé à la branche de retour de la source d'alimentation de puissance vs et a la masse de la source d'alimentation de la logique vss (GND Arduino).
Vs	Alimentation de puissance des moteurs.
Vss	Alimentation du circuit intégré, à raccorder à la borne 5v d'Arduino
Input 3 Input 4	Commandent le deuxième pont en H elles sont branché sur l'Arduino pour commander la polarisation de l'attaque du moteur et donc le sens de rotation du deuxième moteur.
Output 3 Output 4	Elles sont raccordées aux les branche d'alimentation du deuxième moteur.

Tableau3.1.Configurations des broches du L293D.

Pour notre cas nous allons utiliser la partie gauche du L293D (premier pont en H) pour commander notre moteur, le comportement du moteur en fonction de l'état des broches se résume par le tableau suivant :

Enable 1	Input 1	Input 2	Comportement du moteur
Haut	Haut	Bas	Tourne dans un sens
Haut	Bas	Haut	Tourne dans le sens inverse
Haut	Bas/Haut	Bas/Haut	Arrêt

Tableau 3.2. Le comportement du moteur.

3.2.3 Moteur a courant continu

Un moteur à courant continu est une machine électrique. Il s'agit d'un convertisseur électromécanique d'énergie permettant la conversion de l'énergie électrique absorbée en énergie mécanique il est très utilisé dans les applications à vitesse variable. IL est constitué principalement par:

- Une partie fixe appelée stator ou bien inducteur constitué de la carcasse du moteur et du circuit magnétique qui crée le champ inducteur soit par des enroulements statoriques (bobinages) parcourus par un courant soit par des aimants permanants.
- Une partie mobile appelée rotor ou bien induit constitué d'encoches dans lesquelles est enroulé un bobinage relié à un collecteur rotatif, ce bobinage est parcouru par un courant qui engendre la création d'un champ magnétique rotorique
- Un collecteur qui est un ensemble de lames de cuivre isolés ou sont reliées les extrémités du bobinage de l'induit. des balais qui sont situés au stator en frottant sur le collecteur. ce dispositif assure la liaison électrique entre la partie fixe et mobile en faisant un simple contacte par toucher.

a. Principe de fonctionnement

Lorsque le bobinage d'un inducteur de moteur est alimenté par un courant continu, sur le même principe qu'un moteur à aimant permanent .il crée un champ magnétique (flux d'excitation) de direction Nord-Sud il apparaît un couple de forces. Ce couple de forces crée un couple de rotation qui fait dévier 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.

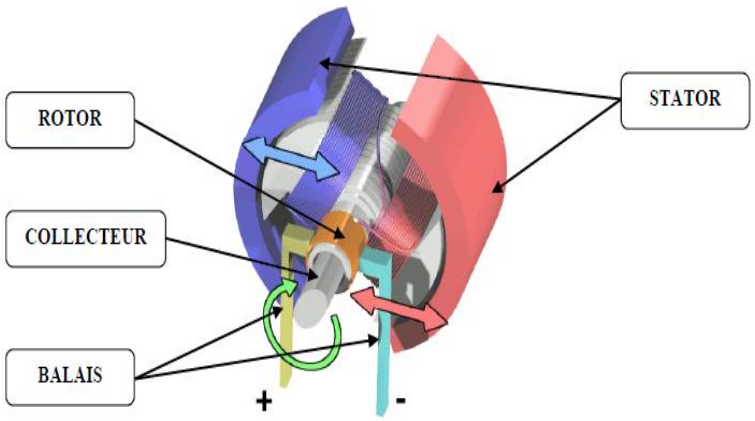


Figure 3.4. Principe de fonctionnement d'un moteur a courant continu [47].

b. Moteur a courant continu GM25-370

Le moteur utilisé pour notre réalisation est un moteur à courant continu disposant d'un encodeur magnétique et d'un réducteur à engrenages.

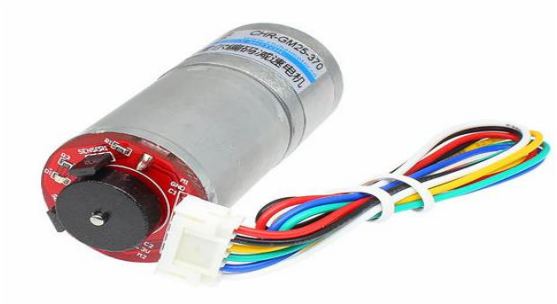


Figure 3.5.Moteur à courant continu GM25-370.

- **Encodeur magnétique**

Pour la mesure de la position d'un moteur rien de mieux qu'un dispositif électromécanique surnommé encodeur, il en existe plusieurs types les plus utilisés sont les encodeurs optiques, mécaniques et magnétiques ce dernier est l'encodeur présent sur notre moteur

Figure 3.6.Encodeur magnétique.

L'encodeur magnétique est un encodeur en quadrature ,également connu sous le nom d'encodeur rotatif incrémental , il possède une roue codeuse et deux capteurs à effet hall cette roue codeuse comporte 11 PPR par tour (points par rotations) qui augmenterons à 224,4 PPR par tour à la sortie du réducteur (11x20,4 définit par le constructeur) et donc grâce à ce dernier la résolution est améliorer cependant l'augmentation du nombre de points magnétiques par tour augmente le nombre de ticks par tour et donne des mesures plus précises.

Les deux capteurs à effet hall détectent tour à tour le passage des points magnétiques plus précisément le champ magnétique crée par ces points, à chaque interaction les capteurs génèrent une tension sur les deux sorties canal A et canal B sous formes de trains d'impulsions qui sont hors phase à 90 degrés ce qui veut dire qu'il ya un décalage de 90 degrés entre les 2 signaux, les sorties sont présenté comme sur la figure (3.7) :

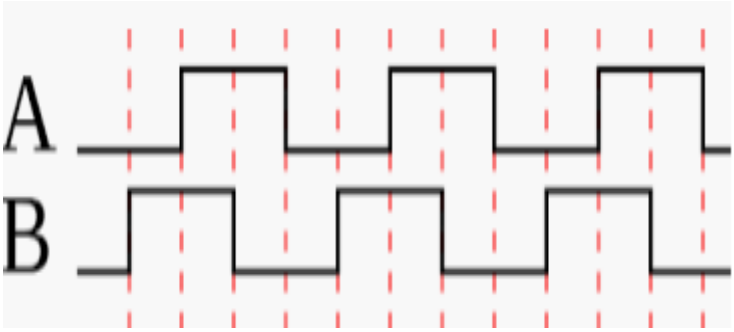


Figure 3.7. Signaux délivrés par les capteurs.

La position pour notre travail sera lié directement au nombre d'impulsion récupéré de la sortie de l'encodeur il suffit, de se fixer une position qui déterminera par la suite le nombre d'impulsion à atteindre après comptage. Apres plusieurs essais nous avons remarqué que cette méthode ne présente pas une bonne précision.

Pour remédier à cette problématique c'est-à-dire avoir une bonne résolution lors du traitement des impulsions nous avons introduit la notion de tick ce dernier sera pris suivant l'état des deux sorties de l'encodeur. Il sera incrémenté à chaque changement d'état de l'une des deux sorties de l'encodeur par rapport à l'autre dans le sens horaire et il sera décrémenté dans le sens inverse, ce qui reviendra à dire que pour un tour après réduction on aura 897.6 ticks.

Sens horaire			Sens anti horaire		
Etat A	Etat B	Nbr de tick	Etat A	Etat B	Nbr d tick
0	0	0	0	0	0
1	0	1	0	1	-1
1	1	2	1	1	-2
0	1	3	1	0	-3

Tableau 3.3. Détermination du sens de rotation et le nombre de tick à partir des impulsions issu des capteurs.

- **Réducteur à engrenages**

Un réducteur à engrenages est un dispositif de transmission mécanique comportant deux ou plusieurs roues dentées.

La vitesse de rotation a la sortie du moteur est très élevée, pour les applications qui ont besoin d'une vitesse pas très grande un réducteur est le dispositif idéal, il permet a la fois de réduire la vitesse de rotation tout en augmentant le couple de sortie et donc faire gagner au moteur de la robustesse et permet d'accroitre la précision comme mentionné précédemment.

(a) Engrenages (b) Moteur à engrenages

Figure 3.8.Représentation d'un moteur à engrenages.

3.2.4 Alimentation

L'alimentation du moteur est effectuée par un adaptateur AC/DC, le modèle utilisé est le HJ-120100E.

Figure 3.9. Adaptateur HJ-120100E.

Cet adaptateur peut être décrit comme un chargeur, il convertie un courant alternatif en un courant continu à faible tension il contient :

- **Un transformateur abaisseur** : converti la tension alternatif du secteur 220 v en une tension alternatif inferieur de 12 v et un courant de 1 A.
- **Un redresseur double alternance :** aussi connue sous le nom de pont de graetz il fournie en sortie une tension alternative redressée.
- **Un condensateur de filtrage : l**e filtrage transforme la tension redressé en une tension aussi constante que possible, le composant chargé du filtrage est un condensateur ou plusieurs en parallèles, plus la capacité du condensateur est élevée plus le filtrage est mieux.
- **Un circuit stabilisateur ou régulateur :** malgré utilisation d'un filtrage, la tension obtenu n'est pas pratiquement continue c'est pour cela qu'il est nécessaire d'ajouter un circuit intégré nommé régulateur de tension qui permet d obtenir une tension continu.

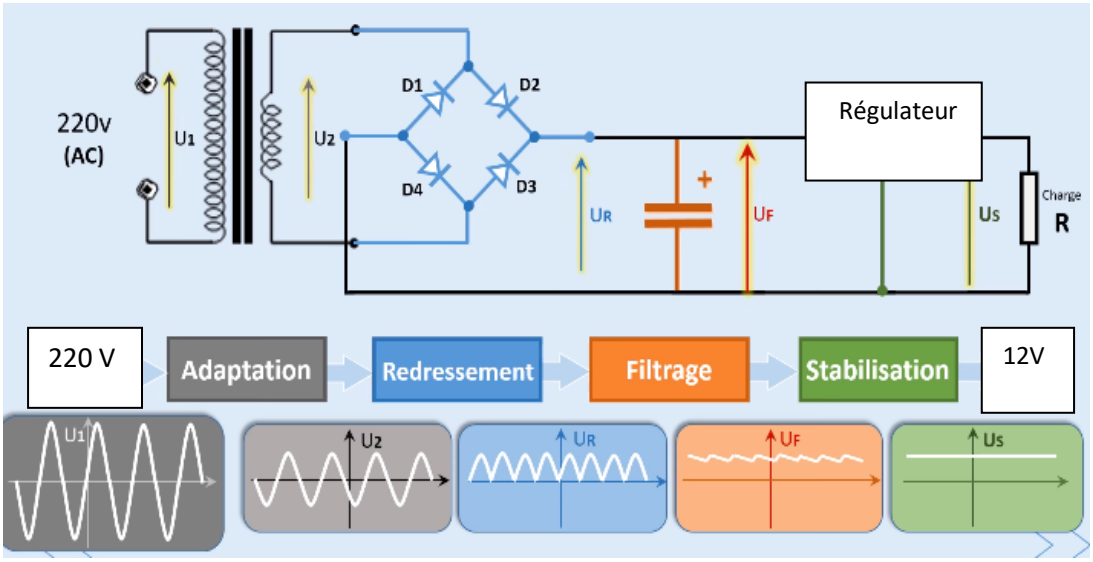


Figure 3.10.Schéma d'alimentation stabilisé [48].

3.2.5 Interface de contrôle

Pour rendre notre travail convivial et pratique nous avons conçu une interface graphique à base du logiciel Delphi cette dernière permettra de faciliter l'envoie de la consigne ainsi que des paramètres du régulateur PID et la fréquence vers la carte Arduino uno et la réception et la visualisation de l'évolution de la mesure du capteur c'est à dire la position. Ces données sont transmises et reçu avec la carte Arduino uno grâce à la liaison série.

a) Présentations du logiciel Delphi

Delphi est un environnement de développement de type RAD (Rapid Application Développent) basé sur le langage Pascal. Il permet de réaliser rapidement et simplement des applications Windows. Cette rapidité et cette simplicité de développement sont dues à une conception visuelle de l'application.

Delphi propose un ensemble très complet de composants visuels prêts à l'emploi incluant la quasi-totalité des composants Windows (boutons, boîtes de dialogue, menus, barres d'outils…) ainsi que des experts permettant de créer facilement divers types d'applications et de librairies [49].

b) Présentations de l EDI Delphi

La figure (3.8) présente l'EDI Delphi, il est composé :

1- D'une barre de menus.

2- D'une barre d'outils.

3- D'une palette de composants.

4- D'un concepteur de fiche.

5- D'un éditeur de code.

6- D'un inspecteur d'objets.

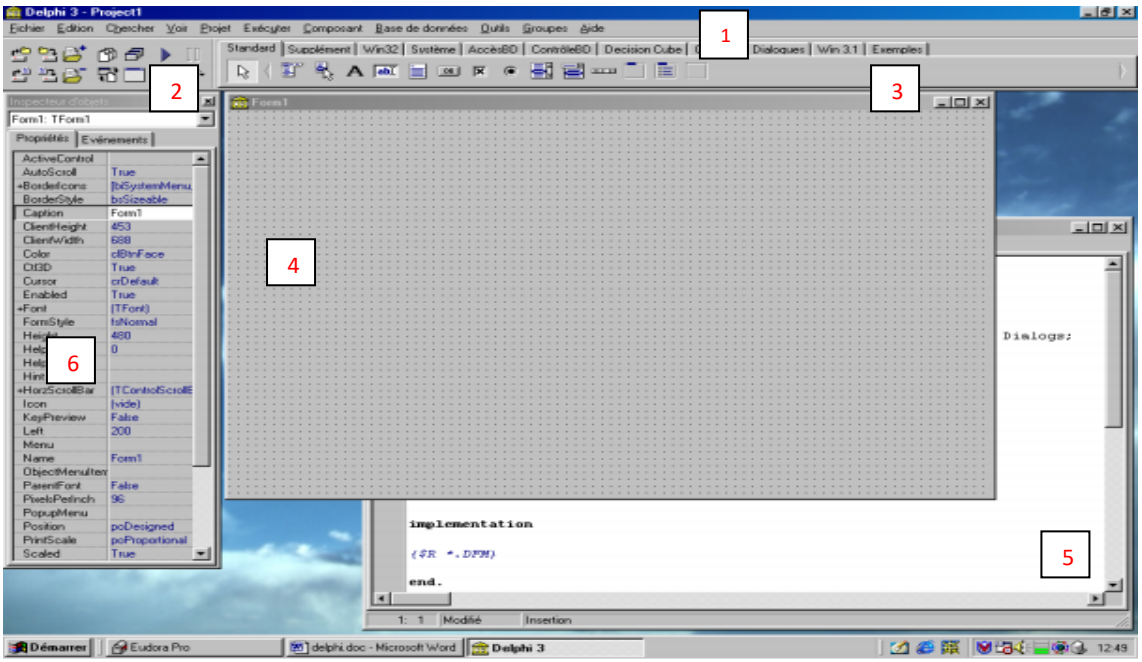


Figure 3.11. EDI Delphi.

c) Présentations de l'interface Delphi

L'interface est crée par l ajout d'objets ou bien composants, ces composants sont des éléments a partir desquels les programmes Delphi sont crée. Cette l'interface est présentée par la figure ci-dessous :

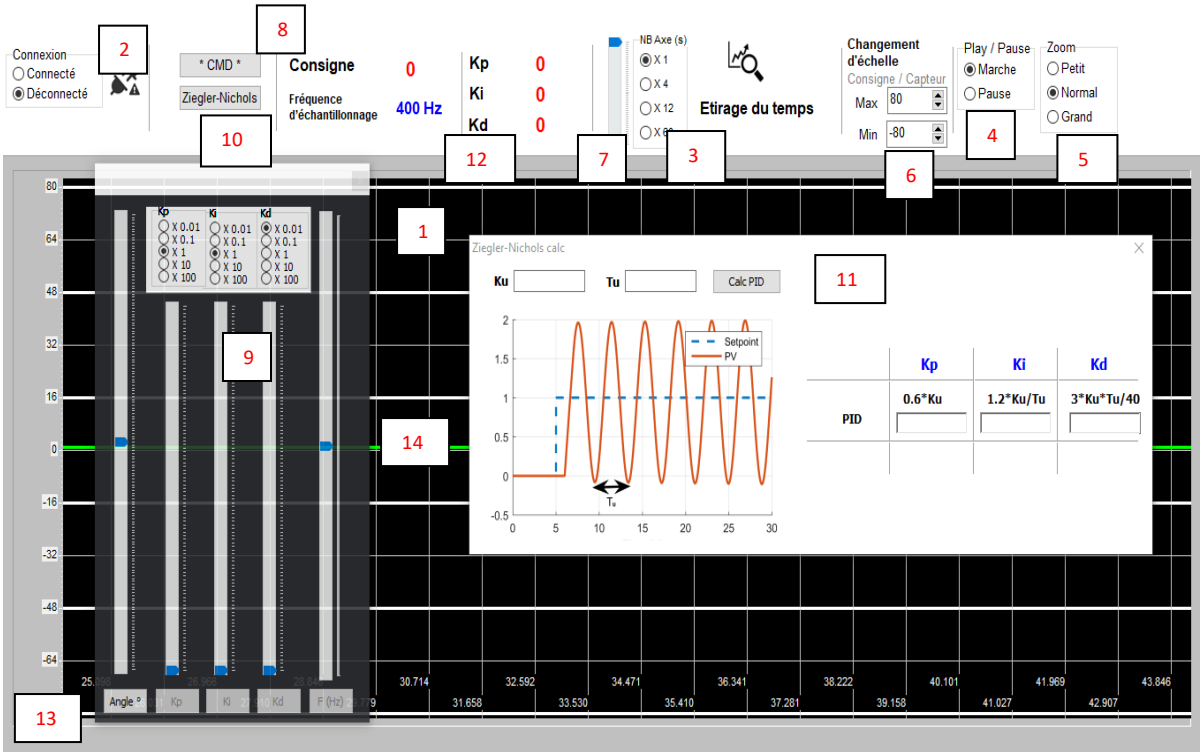


Figure3.12.Interface Delphi.

Les objets contiennent des propriétés et des évènements qui déterminent leurs aspects et leurs comportements, les objets constituons notre interface sont :

1- Form1 : fond de Visualisation avec grillage.
2- Radiogroup1 : permet d'effectuer ou pas la connexion avec la carte de commande.
3- Radiogroup2 : permet d'ajouter des axes de temps pour une meilleure précision.
4- Radiogroup3 : permet de démarrer ou stopper le défilement en temps réel.
5- Radiogroup4 : permet de faire un agrandissement ou une réduction de la taille de la visualisation du comportement du système.
6- Ed min et Ed max : permet le changement d'échelle de la consigne et du capteur.

7- Trackbarscaltemp : permet d'élargir et rétrécir l'écoulement du temps.

8- Button: permetd'afficher form2.

9- Form2 : contient 4 trackbars pour le contrôle de la position (consigne) et le réglage des paramètres K_p,K_i,K_d ainsi que de la fréquence .

10- Button : permet d'afficher forme.

11- Form : fiche pour calculer les paramètresK_p, $K_i$et $K_d$par la méthode de Ziegler et Nichols .

12- Labvaltarget, labfre, labkp, lapki, labkd : zone d'affichage de la consigne, la fréquence et les paramètres du PID issu de form2.

13- Axes : C'est les coordonnées qui servent à fixer la position d'un point d'un plan, c'est les zones de traçage de la consigne ainsi que de la mesure du capteur en temps réel.

14- Curves : les courbes de la consigne ainsi que de la mesure du capteur

Il ya aussi 2 objets non visible sur l interface graphique qui sont le comport1 qui permet d'établir la communication interface/ Arduino , et un Timer pour l'affichage du graphe.

3.3 Schéma global de la réalisation

Apres avoir décrit chaque élément le schéma global du projet est présenté sur la figure (3.13) :

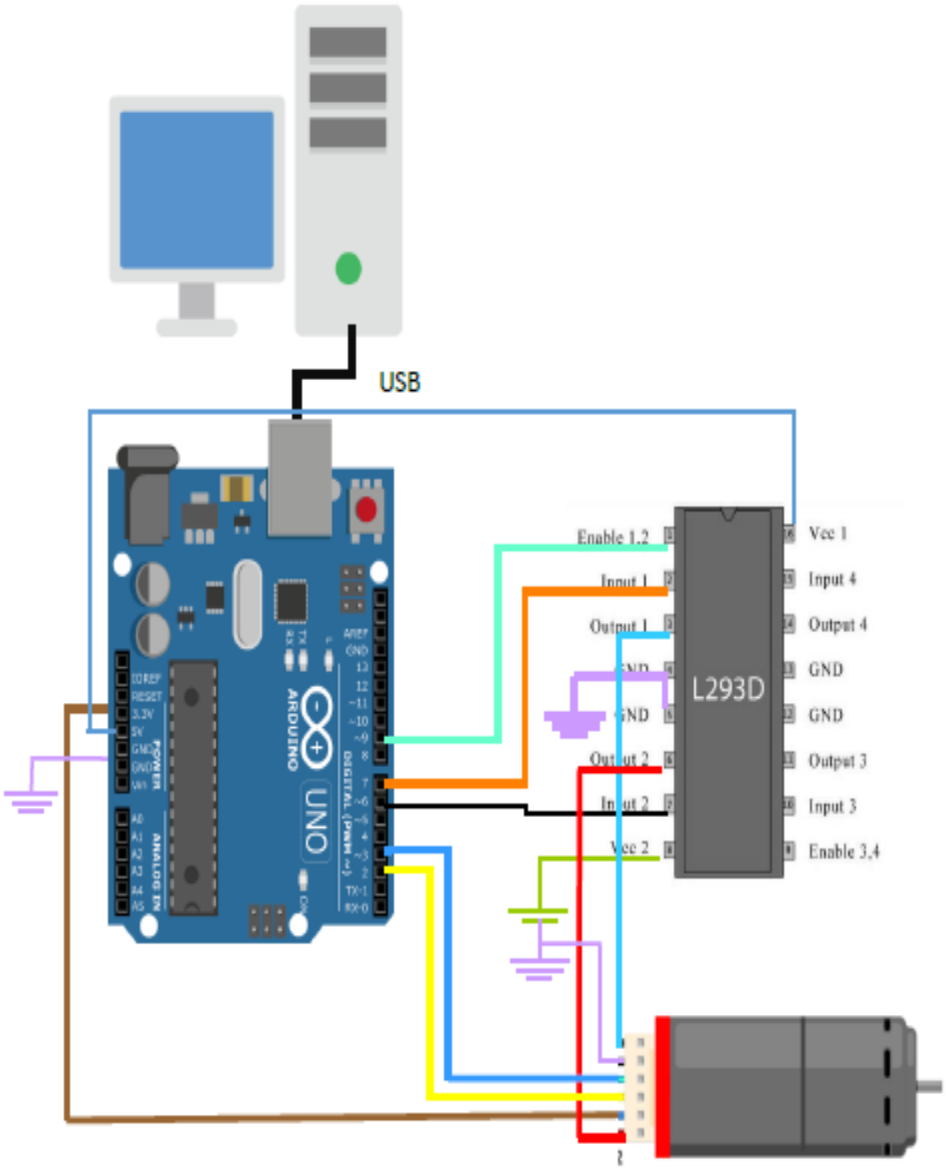
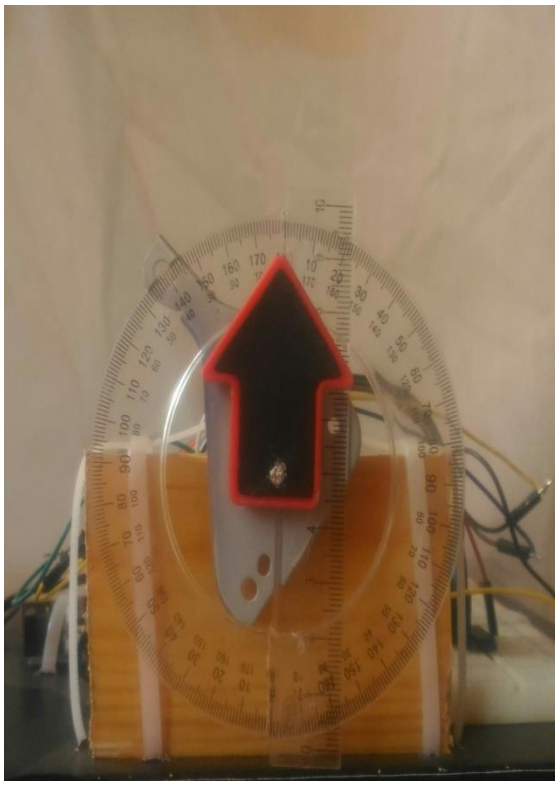
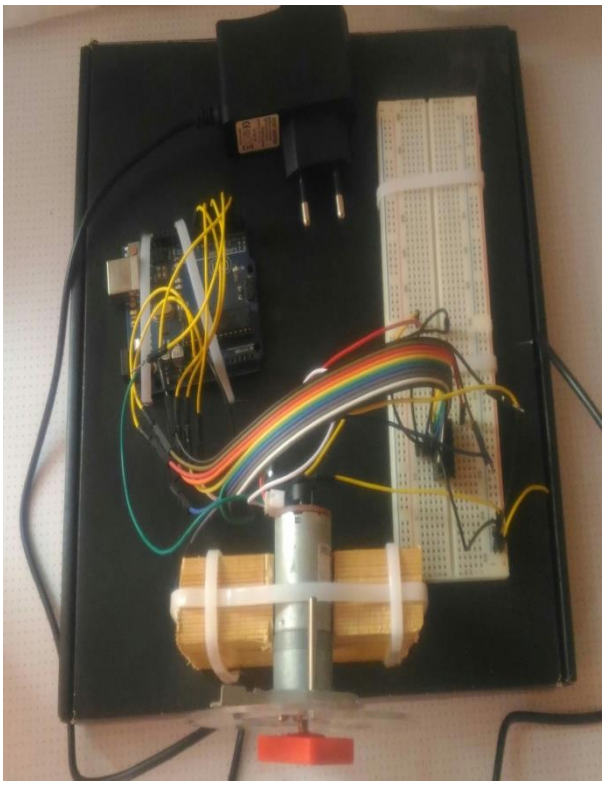
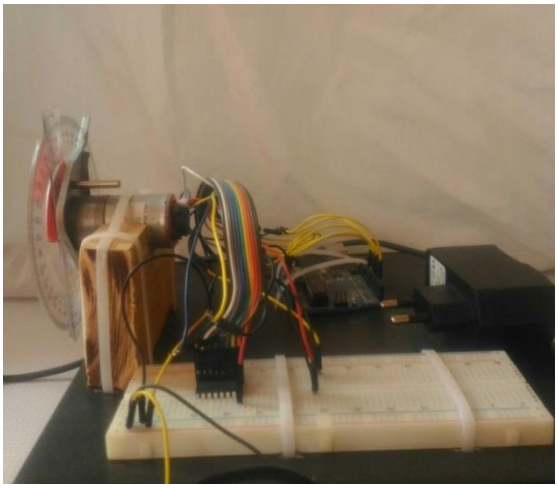


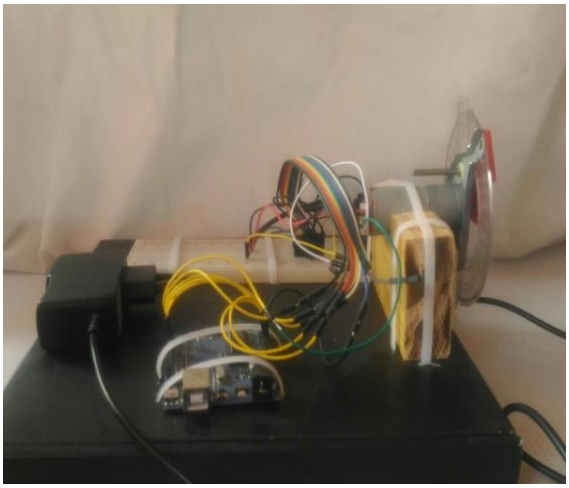
Figure3.13.Schéma global du projet.





(a) Vue de face.
(b) Vue de haut.





(c) vue de droite.

(d) vue de gauche.

***Figure3.14*.** Vue d'ensemble du dispositif expérimentale.

3.4 Programmations

Après avoir vu l'ensemble des éléments constituent notre système nous allons passer à la programmation sous Arduino et sous Delphi.

3.4 .1 Programme sous Arduino

Le programme sous Arduino permet la réception de la consigne et des paramètres du PID, le traitement de la régulation et l'envoie de la position du moteur au pc.
Pour cela le programme se compose de plusieurs parties (communication, réception, envoi d'informations, traitement (régulation), collecte d'information du capteur, action sur le moteur).

a. Communication série

La communication série avec le PC doit être configuré, en configurant le bon port (COM) ainsi que la vitesse de transmission qui est de 9600 bauds.

```
void setup() {
 Serial.begin (9600);
 }
```

- **Réception de la consigne, des coefficients $\mathbf{K_p}$, $\mathbf{K_i}$ et $\mathbf{K_d}$ de la fréquence**

Les commandes reçues par le PC sont envoyées sous la forme d'un message codé Comme suite :

Commande + Valeur + Fin message.

```
If (Serial.available() > 0)
{
inputString=Serial.readStringUntil('*');
}
if (inputString !="") {
if (inputString.indexOf("CO") >=0) {     //Consigne a suivre
inputString =inputString.substring(2);
target_deg = inputString.toFloat();
}else
If (inputString.indexOf("KP") >=0) {
inputString =inputString.substring(2);
kp = inputString.toFloat();
}else
If (inputString.indexOf("KD") >=0) {
inputString =inputString.substring(2);
kd = inputString.toFloat();
}else
If (inputString.indexOf("FE") >=0) {
inputString =inputString.substring(2);
frequence_echantillonnage  =inputString.toFloat();
}
inputString = "";
}
}
```

- **Envoi de la mesure du capteur**

L'envoi de la valeur du capteur au PC est envoyé sous forme numérique contenu dans lavariable 'PosiMoteur' à l'aide de l'instruction 'Serial.println (PosiMoteur)'.

```
Serial.println (PosiMoteur);
```

b. Récupération des informations de l encodeur

La collecte d'information du capteur se fait à l'aide de la bibliothèque 'Encoder.h' Qui permet de donnée la valeur la position du moteur à chaque changement du signal du capteur la valeur obtenue est un nombre entier soit positif soit négatif.

La lecture dans le programme se fait d'abord par une configuration des PINS du capteur utiliser pour notre cas 2 et 3 qui permet les interruptions pour détecter le changement d'état des deux signaux des deux capteurs.

La routine de lecture ce fait par l'appel de la fonction 'myEnc.read()'; et de sauvegarder la valeur obtenu dans une variable .

```
EncodermyEnc(2, 3);

longoldPosition= 0;
voidloop() {
longnewPosition=myEnc.read();
if (newPosition!=oldPosition)
{
oldPosition=newPosition;
PosiMoteur= newPosition;
}
```

c. Régulation

A l'étape actuelle nous Avons obtenu la consigne venue du PC et la position du moteur acquise des capteurs ce qui va permettre d'appliquer la formule du PID à notre régulation.

```
void regulation()
{
dt=1/(frequence_echantillonnage*1.0);
// (01) Conversion consigne en nombre deTiks
target_ticks=target_deg*NB_ticks_Tour/360;
// (02) Formule PID
// (Consigne)
erreur=target_ticks-PosiMoteur;
erreur_d= (erreur- erreur_old)/dt;
erreur_i=erreur_i+dt*1.0* erreur;
intvitMot=kp*erreur+ki*erreur_i+kd*erreur_d;// Calcul de la
vitesse courante du moteur
erreur_old=erreur; // Ecrase l'erreur précédente par la nouvelle
erreur
//Fin formule PID
  // (03) Limitation de vitesse du moteur
If (vitMot>100)vitMot=100;
else if (vitMot< -100)vitMot= -100;
}
// (04) Actionner le moteur
Tourner(vitMot);

// (05) Envoi de la position du moteur au PC
Serial.println (PosiMoteur);
}
```

Tous les étapes précédentes contenu dans la fonction void regulation () sont exécuté dans un Timer formant une boucle a l'infini qui s'exécute a chaque intervalle de temps du Timer.

Le Timer est utilisé pour ne pas bloqué le fonctionnement de tout le programme et donnée plus de souplesse au déroulement du programme dans Arduino ce qui va échantillonner le traitement.

```
//Configuration
void setup(){
timer.setInterval(1000/frequence_echantillonnage,
regulation);
}
//Execution du timer
void loop (){
timer.run();
}
```

d. Commande moteur

La commande du moteur doit faire l'objet de configuration des pins pour le sens de rotation et le pin de la vitesse du moteur.

La fonction 'void Tourner (int vit Mot)' permet de faire tourner le moteur dans les deux sens avec une vitesse variable

```
intpinInput1= 6; // Commande de sens moteur, Input 1
intpinInput2= 7; // Commande de sens moteur, Input 2
intpinPower= 9;  // Commande de vitesse moteur, Output Enabled1
pinMode(pinPower, OUTPUT);  // Configuration en sortie
pinMode(pinInput1, OUTPUT ); // Configuration en sortie
pinMode(pinInput2, OUTPUT ); // Configuration en sortie
```

```
//Fonction appelée pour contrôler le moteur
void Tourner(int vitMot){
if (vitMot> 0) {
digitalWrite(pinInput1, HIGH );
digitalWrite(pinInput2, LOW );
}
else{
digitalWrite(pinInput1, LOW );
digitalWrite(pinInput2, HIGH );
vitMot= -vitMot;
}
analogWrite(pinPower, vitMot );
```

3.4 .2 Programme sous Delphi

Le Programme est composé de plusieurs partie chaque partie à un fonctionnement spécifique :

- Une partie pour l'envoi de commande et paramètres.
- Une partie pour la réception d'information du capteur.
- Une partie pour l'affichage du graphe.

a. Communication série

Deux paramètres essentielle vont être défini , le port de communication 'COM4'ainsi que la vitesse de transmission qui est assuré par l'objet 'ComPort1'.

Apres avoir configuré ces deux paramètres il faut ensuite ouvrir le port par l'élément ' Radiogroup1'.

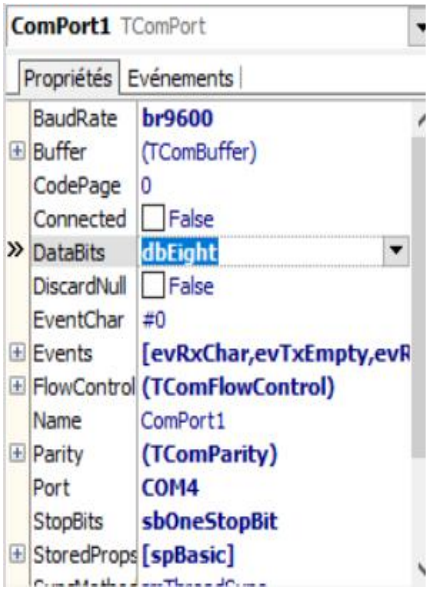


Figure3.15. Configuration du port série ainsi que la vitesse de transmission.

- **Envoie de la consigne, des paramètres et de la fréquence**

Les commandes sont envoyées sous forme de message texte avec le format suivant qui à été déjà décrite précédemment commande+ valeur+fin message.

```
// envoi consigne
procedure TForm2.BTConsigClick(Sender: TObject);
begin
COEnvoi:='CO'+inttostr(Trunc(-1*TrackBarConsign.Position))+'*';
COEnvoi:=StringReplace(COEnvoi,',','.',[rfReplaceAll,rfIgnoreCase]);
if Form1.ComPort1.Connected then
Form1.ComPort1.WriteStr(COEnvoi);
end;
// envoi fréquenced'échantillonage
procedure TForm2.BTFeClick(Sender: TObject);
begin
```

```
// envoi kp,ki,kd
procedure TForm2.BTKpClick(Sender: TObject);
procedure TForm2.BTKiClick(Sender: TObject);
procedure TForm2.BTKdClick(Sender: TObject);
procedure TForm2.BTFeClick(Sender: TObject);
begin
caseRGKp.ItemIndex of
caseRGKi.ItemIndex of
caseRGKd.ItemIndex of
caseRGFe.ItemIndex of
0:Coef:=0.01;
1:Coef:=0.1;
2:Coef:=1;
3:Coef:=10;
4:Coef:=100;
end
//envoi kp
KpEnvoi:='KP'+FloatToStr((100-Coef*TrackBarKp.Position))+'*';
KpEnvoi:=StringReplace(KpEnvoi,',','.',[rfReplaceAll,rfIgnoreCase]);
if Form1.ComPort1.Connected then
Form1.ComPort1.WriteStr(KpEnvoi);
BTKp.Enabled :=False;
end;
```

```
// envoi ki
KiEnvoi:='KI'+FloatToStr(Coef*(100-TrackBarKi.Position))+'*';
KiEnvoi:=StringReplace(KiEnvoi,',','.',[rfReplaceAll,rfIgnoreCase]);
if Form1.ComPort1.Connected then
Form1.ComPort1.WriteStr(KiEnvoi);
BTKi.Enabled :=False;
end;
// envoi kd
dEnvoi:='KD'+FloatToStr((100-Coef*TrackBarKD.Position))+'*';
KdEnvoi:=StringReplace(KdEnvoi,',','.',[rfReplaceAll,rfIgnoreCase]);
if Form1.ComPort1.Connected then
Form1.ComPort1.WriteStr(KdEnvoi);
BTKd.Enabled :=False;

end;
```

- **Réception de la mesure**

Dans notre objet de communication série '**ComPort1'**dans l'évènement '**OnRxChar'**on établi le programme suivant pour la lecture de informations reçus

```
procedure TForm1.ComPort1RxChar(Sender: TObject; Count:
Integer);
var
Str: String;
i:integer;
begin
//Réception de l'intégralité du message
ComPort1.ReadStr(Str, Count);
```

```
//Recherche de la fin du message
for I := 1 to Count do
begin
STemp:=STemp+Str[i];
ifStr[i]=#13 then beginSRcu:=STemp;  STemp:='';  end;
end;
//Recherche et remplacement par une virgule valide par Windows
SRcu:=StringReplace(SRcu,',',DecimalSeparator,[rfReplaceAll,rfIgnoreCase]);
SRcu:=StringReplace(SRcu,'.',DecimalSeparator,[rfReplaceAll,rfIgnoreCase]);

TThread.Synchronize(nil,
Procedure
begin
ExctractNumb(SRcu);
end);
end;
```

Pour que notre message reçu soit exploitable il nous reste un dernier traitement c'est l'élimination des caractères non valide, car parfois la communication série subit des perturbations ce qui va générer des caractères non valides généralement juste après la mise en marche de la communication série.

Alors pour avoir une valeur numérique valide ont va procédé par l'élimination de tous les caractères sauf les numéros de **0** à **9** ou le signe **'-'** ou la **virgule.**

```
for I := 1 to length(Val) do
if(Val[i] IN ['0'..'9']) OR (Val[i]=DecimalSeparator) OR (Val[i]='-')
then
StrFilt:=StrFilt+Val[i];

PosiMoteur:=StrToFloat(StrFilt) ;
```

Une fois le message reçu bien filtré il peut être converti de chaine de caractère vers un réel dans notre variable **PosiMoteur** puis converti en **degré.**

```
PosiMoteur:=StrToFloat(StrFilt) ;

angle_deg := ((PosiMoteur/(NB_ticks_Tour))*360.0);//
```

b. Afichage du graphe

L'affichage du graphe passe par plusieurs étapes à commencer par le changement d'échelle de la valeur du capteur à la valeur sur l'objet qui permet le dessin.

La valeur du capteur ou de la consigne est converti à une plage de 0 à 100 a l'aide de la fonction suivant :

```
Function ChangeEchel(ValeurAConvertir,MesurMax,MesurMin,SortiMax,SortiMin:Real):Real;
Begin
Result:=(((SortiMax-SortiMin) /(MesurMax-MesurMin)) *(ValeurAConvertir-MesurMin)) +SortiMin ;
End;
```

Une fois notre nouvelle valeur du capteur compris entre 0 et 100, elle peut être afficher dans sa totalité dans l'objet qui permet le dessin sans avoir un dépassement au-delà des limites de l'objet.

```
//Changement d'échelle
//-----------------------------------------------
NewValConsigne:=Trunc(ChangeEchel(ValConsigne,MesurMaxi,MesurMini,100,0));
NewValCapteur:=Trunc(ChangeEchel(ValCapteur,MesurMaxi,MesurMini,100,0));
```

Dans les instructions qui suive, on fait une adaptation des valeurs du capteur et la consigne aux dimensions de l'objet a dessinée le PaintBox (Pb2) pour la consigne la variable (h) et le capteur la variable (h2)

```
//----Consigne
h:=Trunc(100*100/Pb2.height*U)-Trunc(NewValConsigne*100/Pb2.height*U)+2 ;
//----Capteur
 h2:=Trunc(100*100/Pb2.height*U)-Trunc(NewValCapteur*100/Pb2.height*U)+2 ;
```

L'affichage du graphe est assuré par les instructions **moveto** et **lineto** qui dessine des droites entre **moveto** et **lineto.**

Ou **moveto** représente l'Anciene valeur du capteur et **lineto** la nouvelle valeur du capteur le programme dessine une droite de l'anciene valeur vers la nouvelle valeur.

```
//Graphe Capteur
Pen.color :=clLime; //couleur du graphe
Pen.width := 3;
moveto(pb1.width - pas , oldh2 * U);
lineto(pb1.width      ,  h2 * U);
```

Une fois le message reçu bien filtré il peut être converti de chaine de caractère vers réel dans notre variable **Posimoteur** puis converti en **degré.**

Une fois notre graphe dessiné il nous reste a faire le décalage de droite à gauche pour visualiser le comportement en temps réel du moteur avec la fonction suivant :

```
Procedure decale(n : integer);
var
  r1, r2 :Trect;  { rectangles }
begin
WITH form1 DO
begin
  r1 := rect(0,0,pb1.width, pb1.height);     // fenètre à décaler
scrolldc(bmp1.canvas.handle, -n, 0, r1, r1, 0, nil); // décalage
  r2 := rect(pb1.width-n, 0, pb1.width, pb1.height);   // initialise à noir
bmp1.canvas.brush.color := clblack;
bmp1.canvas.Fillrect(r2);
lignesh(pb1.width-n);      // dessineligneshoriziontales
end;
end;
```

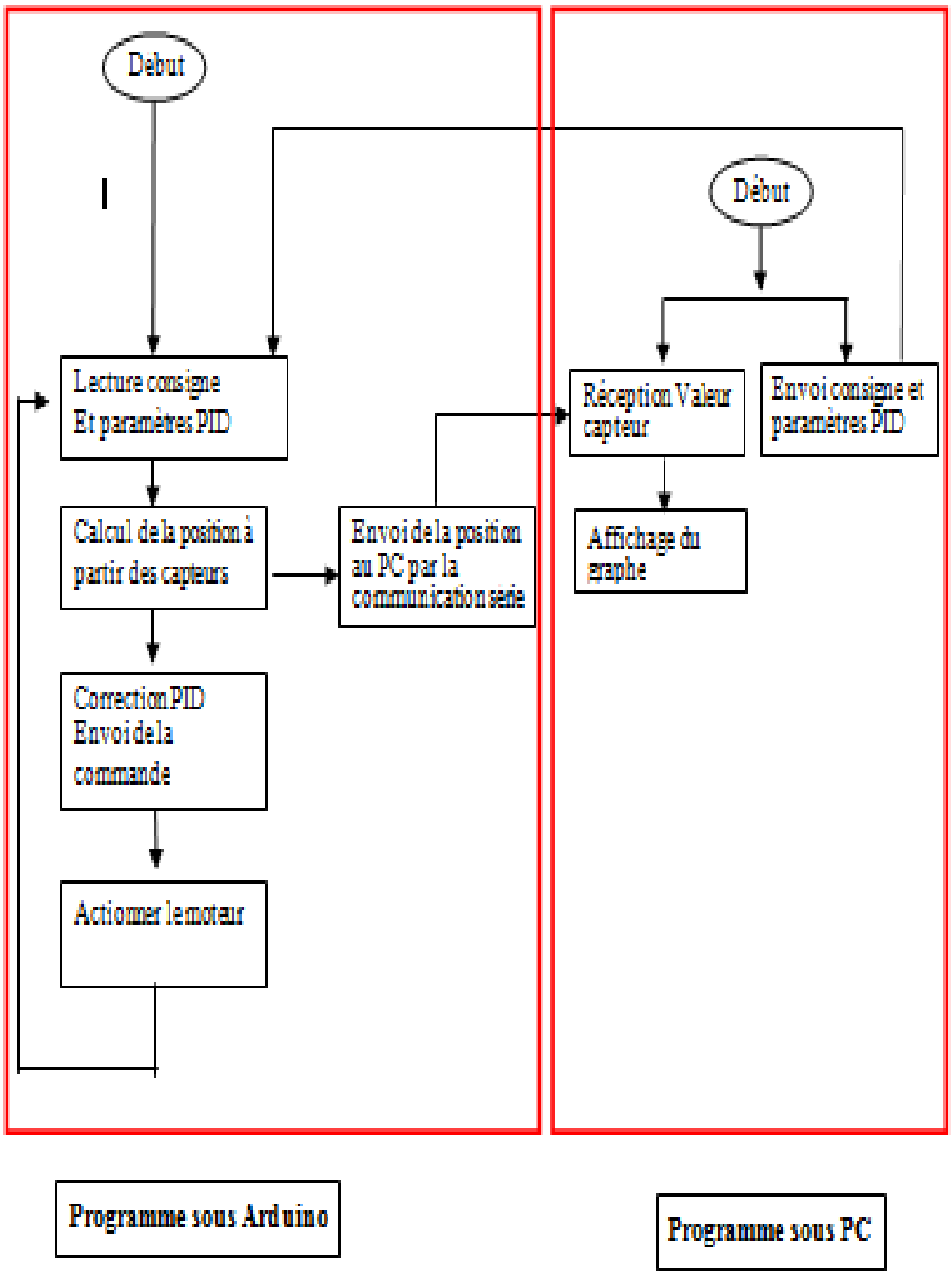


Figure3.16.Organigramme du projet.

3.5 Tests et discussions

Apres avoir détaillé chaque élément du synoptique ainsi que les outils de programmation nous passerons dans ce qui suit au test c'est-à-dire comment répond notre système par rapport au consignes en commençant par la détermination des paramètres PID.

3.5.1 Méthodes de réglages des paramètres du régulateur PID

Le réglage d'un PID consiste à trouver les meilleurs coefficientsK_p, $K_i$et $K_d$pour obtenir un système robuste, rapide et précis tout en limitant les dépassements.

Pour cela nous avons utilisé deux méthodes pour le réglage des paramètres:

La première méthode appelé méthode d'approximations successives et la deuxième méthode de Ziegler et Nichols appelé méthode du point critique, c'est des méthodes dites empiriques et donc qui s'appuient sur l'expérience, l'observation et non sur la théorie. 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 suffisants mais ne permettent pas un réglage très fin.

a. Réglage par la méthode d'approximations successives et tests

Nous allons régler les actions du PID l'une après l'autre en suivant l'ordre suivant : P, D, I en faisant des tests tout en examinant la réponse et trouver le bon compromis entre les trois actions pour cela nous avons fixé une consigne de 45 degré.

- **Régulation proportionnelle**

On fait tout d'abord apparaitre le gain proportionnel en commençant par une petite valeur K_p=0,3 et on remarque que la mesure se rapproche de la consigne sans toutefois l'atteindre d'où le temps de montée =0,099 ms et le temps d'établissement = 0,9821 ms.

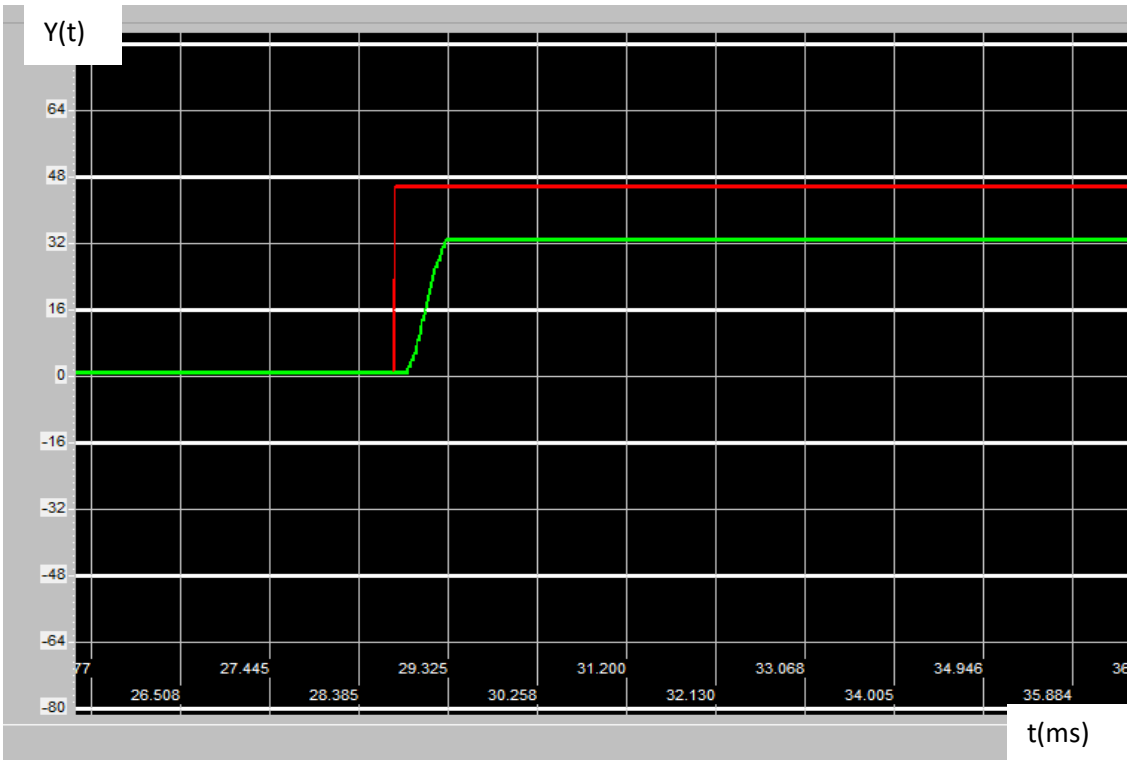


Figure 3.17.Réponse du moteur en fonction du temps avec K_p= 0,3 ; K_i=0 ; K_d=0.

Lorsque l'on augmente K_p le système réagit plus vite d'où le temps de montée diminue et l'erreur statique se trouve éliminé, mais en contrepartie le système perd en stabilité. Le dépassement se fait de plus en plus grand, et le système peut même diverger dans le cas d'un $K_p$démesuré (très grand) après plusieurs essais on a choisie K_p= 7,5, le temps de montée =0,0642 ms. (le temps d'établissement =15,003ms) et l'erreur statique est annulée.

Figure3.18. Réponse du moteur en fonction du temps avec K_p= 7,5 ; K_i=0 ; K_d=0.

- **Régulation proportionnelle – derive**

L'action proportionnel a fait apparaitre des oscillations pour remédier à ce problème on a rajouté un terme dérivateurK_d=0,23 afin d'obtenir un régulateur PD.

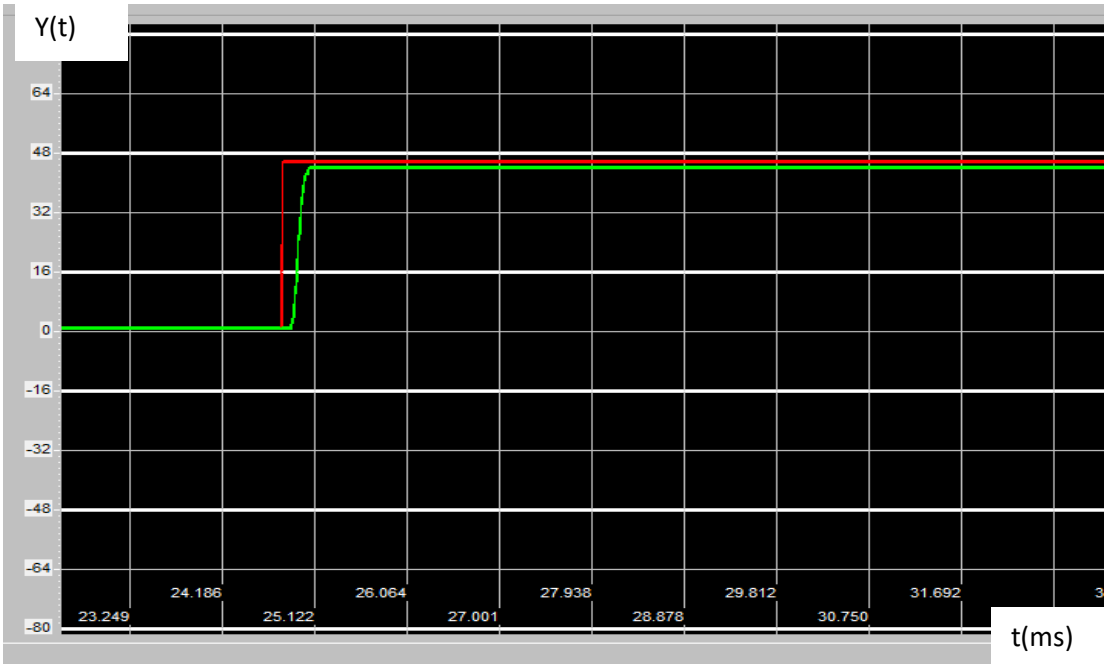


Figure 3.19.Réponse du moteur en fonction du temps avec K_p=7,5; K_i=0 ; K_d=0,23

On remarque que le système est devenu stable et ne présente plus de dépassement et la rapidité est amélioré du fait que le temps d'établissement =0,753 ms (temps de montée= 0,639ms) mais en contrepartie la précision à diminuer c'est pour ca qu'il est nécessaire de faire apparaitre un autre terme pour annuler l'erreur statique.

- **Régulation proportionnelle –dérivée – integral**

Figure 3.20.Réponse du moteur en fonction du temps avec K_p= 7,5 ; K_i=3,1 ; K_d=0,23.

Afin de mettre en place un régulateur PID complet, on a rajouté le terme intégral qui a permis d'annuler l'erreur statique .Après avoir trouvé les bons coefficients, l'assemblage des 3 actions du PID permet une régulation optimale, grâce a l'action P le système est devenu rapide, en rajoutant l'action D le système est devenu stable et a encore gagné en rapidité en rajoutant l'action I le système devient précis.

b. Réglage par la méthode du point critique et tests

Nous allons fixer les coefficients K_p, $K_i$et K_dà 0 et la consigne a 45 degré et augmenter le gain du correcteur proportionnel K_p jusqu'à obtenir des oscillations périodiques stables et ensuite déduire le gain critique K_{cr} et la période des oscillationsT_{cr}.

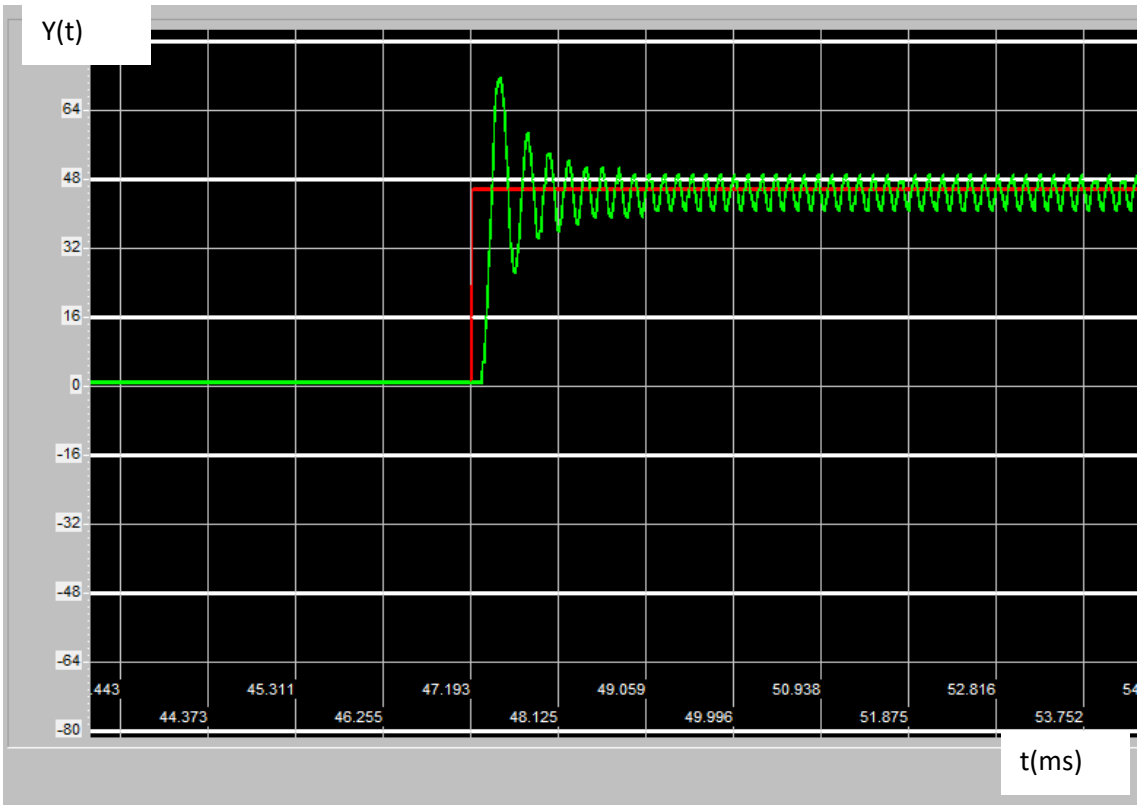


Figure3.21.Réponse du moteur en fonction du temps avec K_p= 13 ; K_i=0 ; K_d=0.

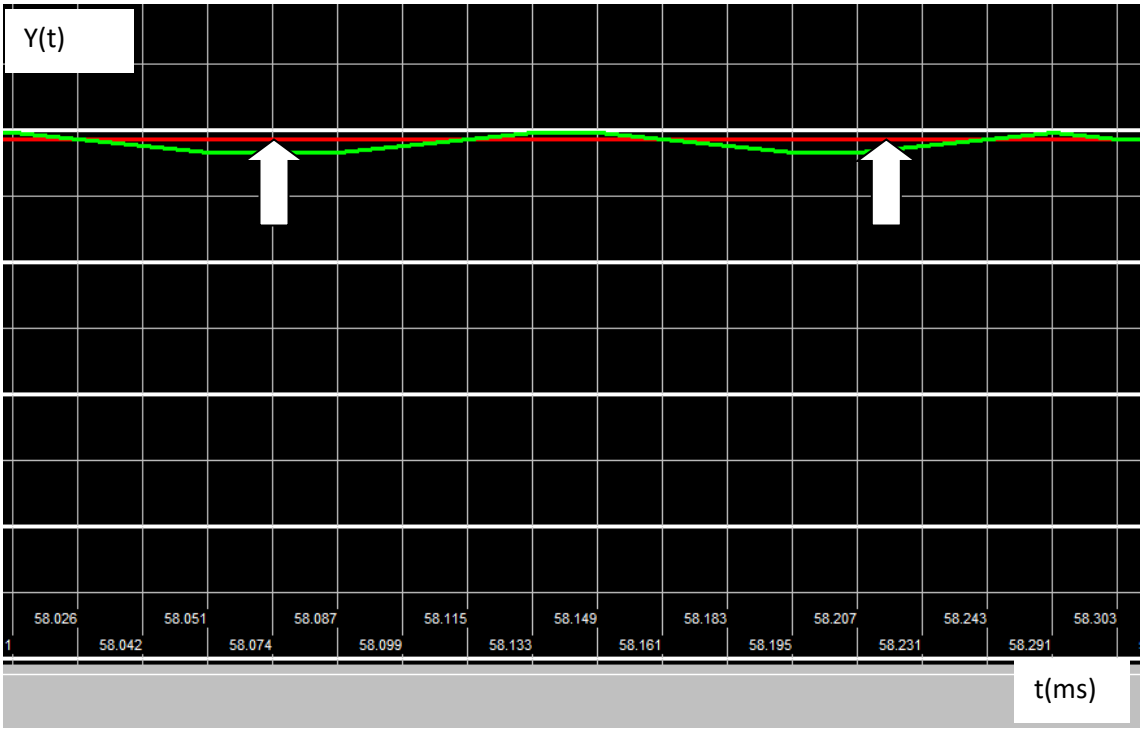


Figure3.22.Détermination de la période T_{cr} pourK_p =13.

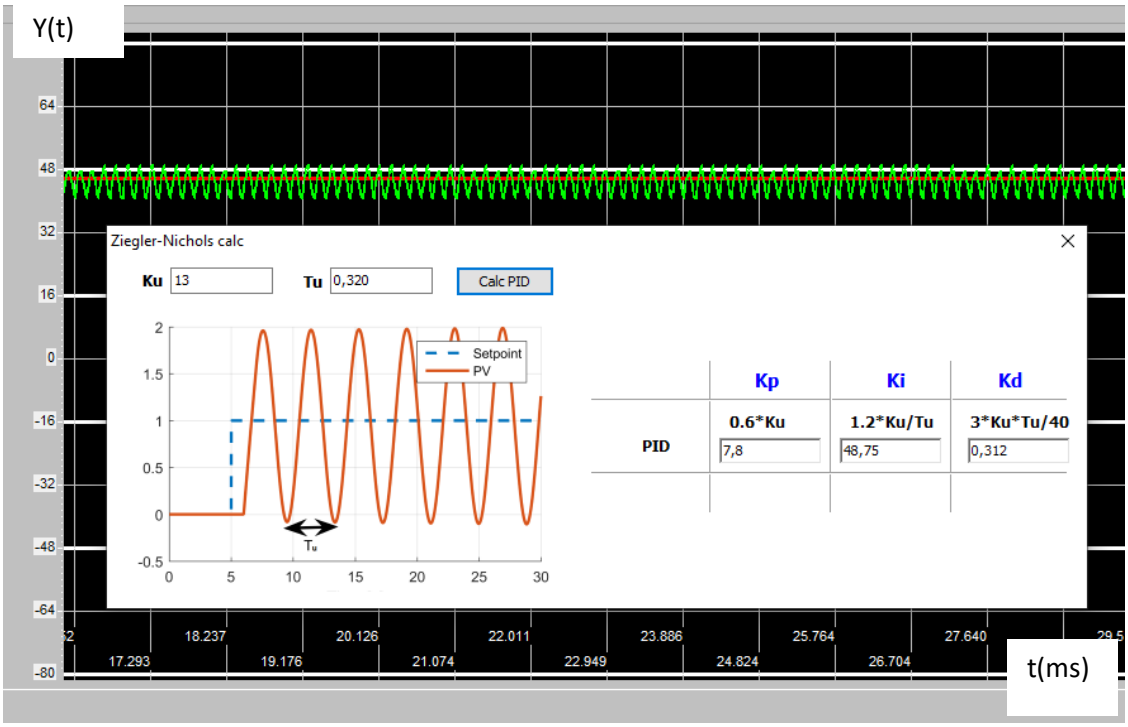


Figure3.23.Détermination des paramètres par la méthode de Zeigler Nichols.

Apres avoir essayé plusieurs valeurs deK_p, nous avons relevé le gain d'oscillation $K_{cr} = 13$, et la période d'oscillation,$T_{cr} = 0{,}320$ en appliquant les valeurs données dans le tableau (1.3) de Ziegler-Nichols, on a déduit les valeurs des paramètres : K_p= 7,8 ; K_i=48,75 ; K_d= 0,312 pour la même consigne de 45 degré on a obtenues les résultats suivants :

- **Régulation proportionnelle**

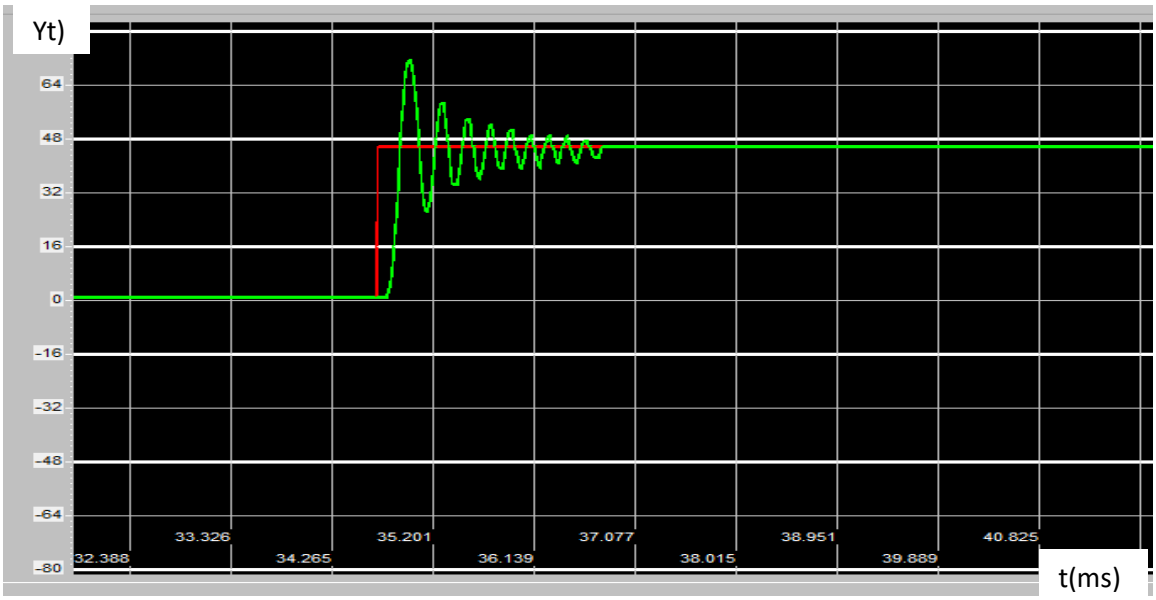


Figure3.24.réponse du moteur en fonction du temps avec K_p= 7,8 ; K_i=0 ; K_d=0.

On remarque que l'action proportionnel dont le gain K_p= 7,8 permet d'accéléré le comportement globale de la boucle fermé d'où le temps de montée= 0,639 ms et de (temps d'établissement= 14,896 ms), et permet aussi d'améliorer la précision en éliminant l'erreur statique mais engendre des oscillations et donc augmente l'instabilité du système

- **Régulation proportionnelle – derivé**

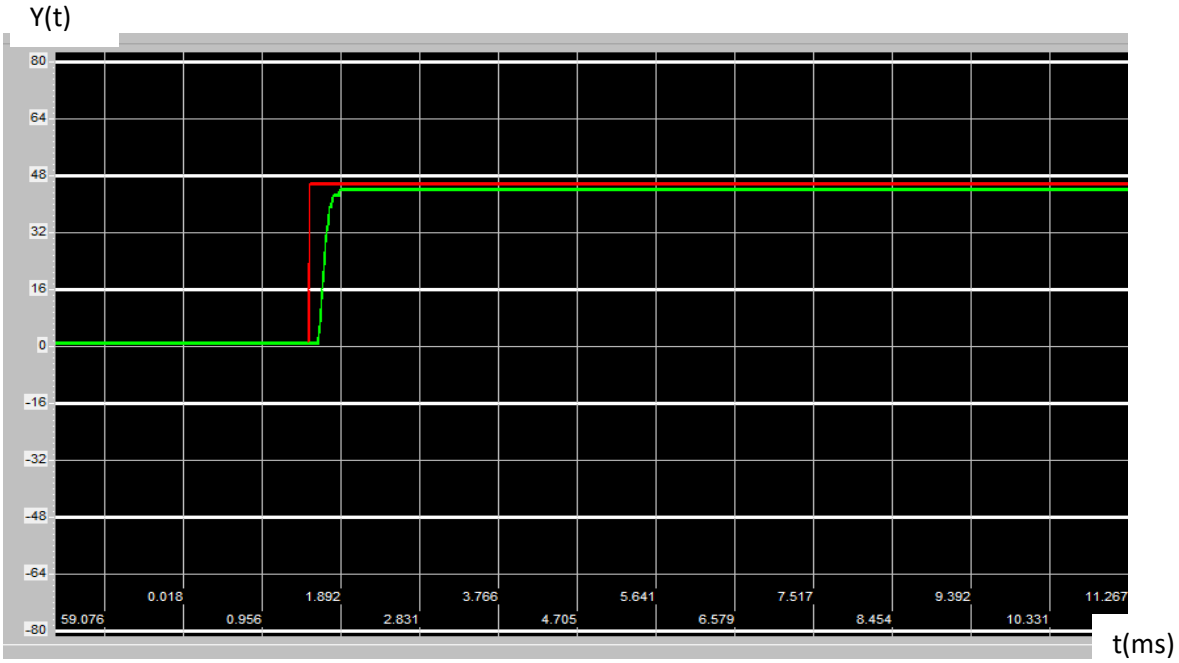


Figure3.25.Réponse du moteur en fonction du temps avec K_p= 7,8 ; K_i=0 ; K_d=0,312.

La combinaison des deux actions P et D dont les coefficients sont K_p = 7,8 etK_d = 0,312, permet d'associer l'avantage du régulateur P c'est à dire la rapidité (temps de montée =0,637 ms) a l'avantage du régulateur D qui est l'augmentation de la stabilité en atténuant les oscillations tout en d'améliorant la rapidité (temps d'établissement= 0,751 ms) mais, cette dernière engendre une diminution de la précision par l'apparition d'un écart statique.

- **Régulation proportionnelle –dérivée – integral**

Le coefficient K_i déterminer précédemment par la méthode de Ziegler-nicols est K_i=48,75. Ce dernier nous a donné de mauvais résultats. Pour obtenir une meilleur précision nous avons choisi ce paramètre par tâtonnement, nous l'avons modifié et nous avons pris : K_i= 2,77.

La correction proportionnelle- dérivé n'élimine pas l'erreur statique pour y remédier on introduit le terme intégrale qui permet d'éliminer l'erreur statique et d'augmenter la précision en régime permanant tout en gardant les bénéfices des actions P et I d'où la stabilité et la rapidité.

Figure3.26.Réponse du moteur en fonction du temps avec K_p= 7,8 ; K_i=2,77 ; K_d=0,312.

- **Réponse du système lors du changement de consigne**

Pour voir comment réagi notre système lorsqu'on change de consignes, nous avons introduit quatre position 15°, 60°, -18° ;-49 a atteindre en utilisant la méthode d'approximations successives pour l'identification des paramètres et trois autres position 16°,-19° ,50°, en utilisant la méthode du point critique de Ziegler et Nichols les réponses du système ont était comme suit:

- **Réglage par Méthode d'approximations successive**

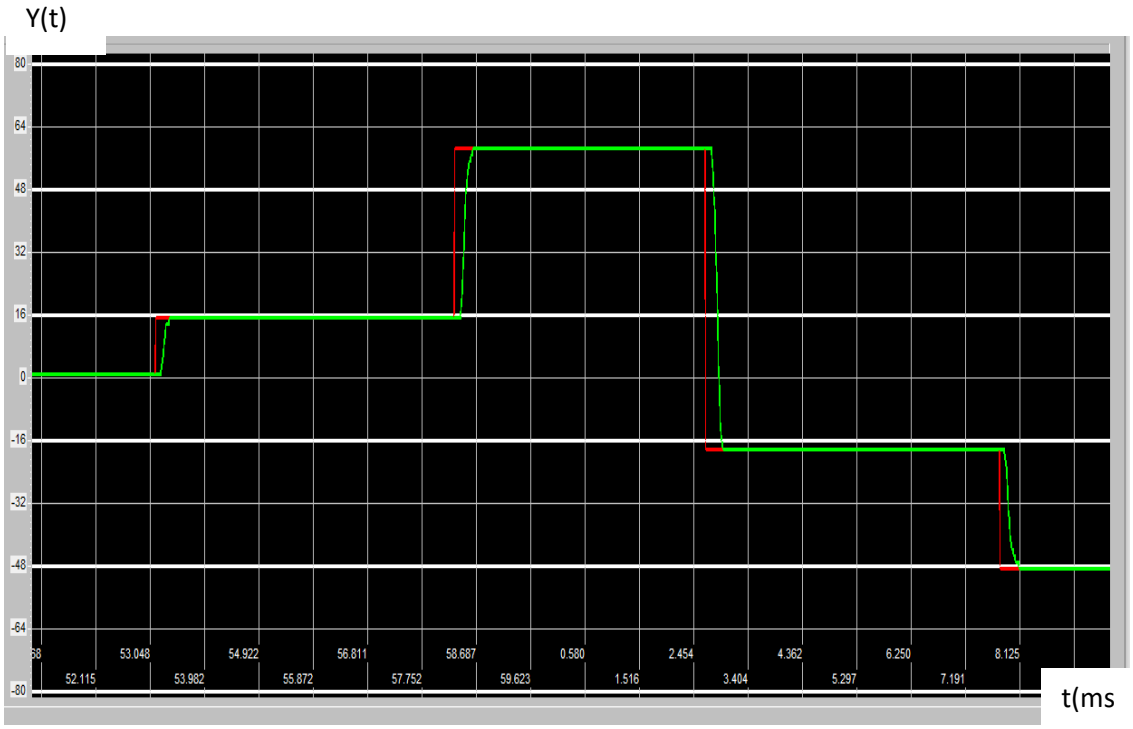


Figure3.27. Réponse du moteur après un changement de consigne 15°à 60° à-18°à-49° avec les paramètres K_p= 7,8 ; K_i=2.8 ; K_d=0,25.

- **Réglage par la méthode du point critique**

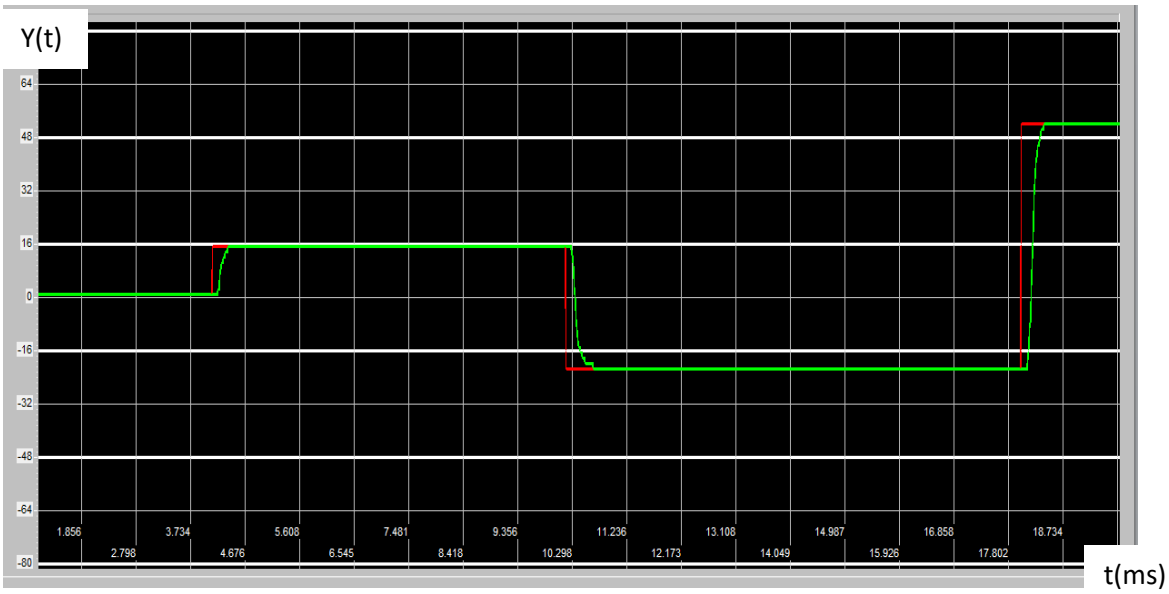


Figure3.28.Réponse du moteur après un changement de consigne 16°à-19° à50°, avec les paramètresK_p= 7,8 ; K_i=2,77 ; K_d=0,312.

On remarque que le système répond correctement lors de changements de consignes avec les paramètres réglés par les deux méthodes.

3.5.2 Comparaison des deux méthodes

Apres avoir régler les coefficients du PID avec la méthode de Ziegler- Nichols et la méthode d'approximations successives on constate d'après les résultats obtenus que ces deux méthodes nous donnes presque les mêmes résultats et elles permettent d'avoir de bonnes performances. Sauf que la méthode de Ziegler Nichols a été pour nous plus facile, surtout du point de vu temps alloué à la détermination des paramètres PIDK_p, K_i et K_d relativement à la méthode d'approximations successives qui nous a pris beaucoup plus de temps et d'essais pour avoir un bon compromis .

3.6 Conclusion

Dans ce chapitre nous avons commencé par présenter le synoptique général de notre réalisation, par la suite nous avons détaillé chaque élément du système. Nous avons aussi détaillé les outils qui nous ont permis d'analyser les réponses de notre système à savoir l'interface graphique sur Delphi. Nous avons remarqué que notre système répond correctement, à chaqu'une des technique de détermination des paramètres du PID, du point de vu temps de monté, temps d'établissement, dépassement et l'erreur statique

Conclusion générale

Le travail qui nous a été proposé est la réalisation d'une régulation PID numérique pour la commande en position d'un moteur à courant continu.

Pour y parvenir nous avons commencé par donné quelques notions et principes sur la régulation analogique et numérique tout en insistant sur le régulateur PID, ce dernier ne pouvant être réalisé d'une manière simple et pratique qu'en l'implémentant sur un calculateur numérique, par la suite nous nous sommes intéressés aux types de calculateurs qui existent et notre choix c'est dirigé vers la carte de prototypage Arduino Uno que nous avons étudié ces caractéristiques et son fonctionnement, cette étude nous a permis de voir de près sa flexibilité et sa facilite pour l'acquisition, la transmission de données et la commande.

Pour rendre notre réalisation convivial nous avons conçu une interface graphique sur Delphi nous permettant de transmettre les consignes ainsi que les paramètres du PID vers la carte d'acquisition Arduino et de visualiser l'évolution des mesures afin de les représentée graphiquement en temps réel pour les comparer avec les consignes données.

Les résultats obtenus ont été assez prometteur pour les deux méthodes adoptées pour le réglage des paramètres sauf que la méthode de Ziegler-Nichols reste la mieux placé au niveau du temps nécessaire pour la détermination des paramètres K_p, K_i et K_d permettant de satisfaire aux mieux un cahier de charge (stabilité , rapidité et précision).

La régulation PID est aujourd'hui l'une des méthodes les plus utilisés car elle est simple à mettre en place pour la plupart des systèmes réels. De plus, le calcul des coefficients Laisse le choix entre plusieurs méthodes, comme les méthodes empiriques qui permettent l'obtenir rapidement des coefficients corrects sans connaissance approfondie du système en donnant des performances acceptables.

Et à la fin, nous pouvons dire que ce mémoire de fin d'étude est une expérience qui nous a été très bénéfique, nous avons consolidé nos acquis théoriques dans le

domaine de la régulation et nous avons touché une application qui nous a permis de voir de près tout un process de l'étude théorique à la réalisation pratique.

En perspective à ce travail nous proposons de voir et de développe autre technique de régulation tels que : la correction RST, la Logique floue, les réseaux des neurones artificiels…etc. et en faire une comparaison.

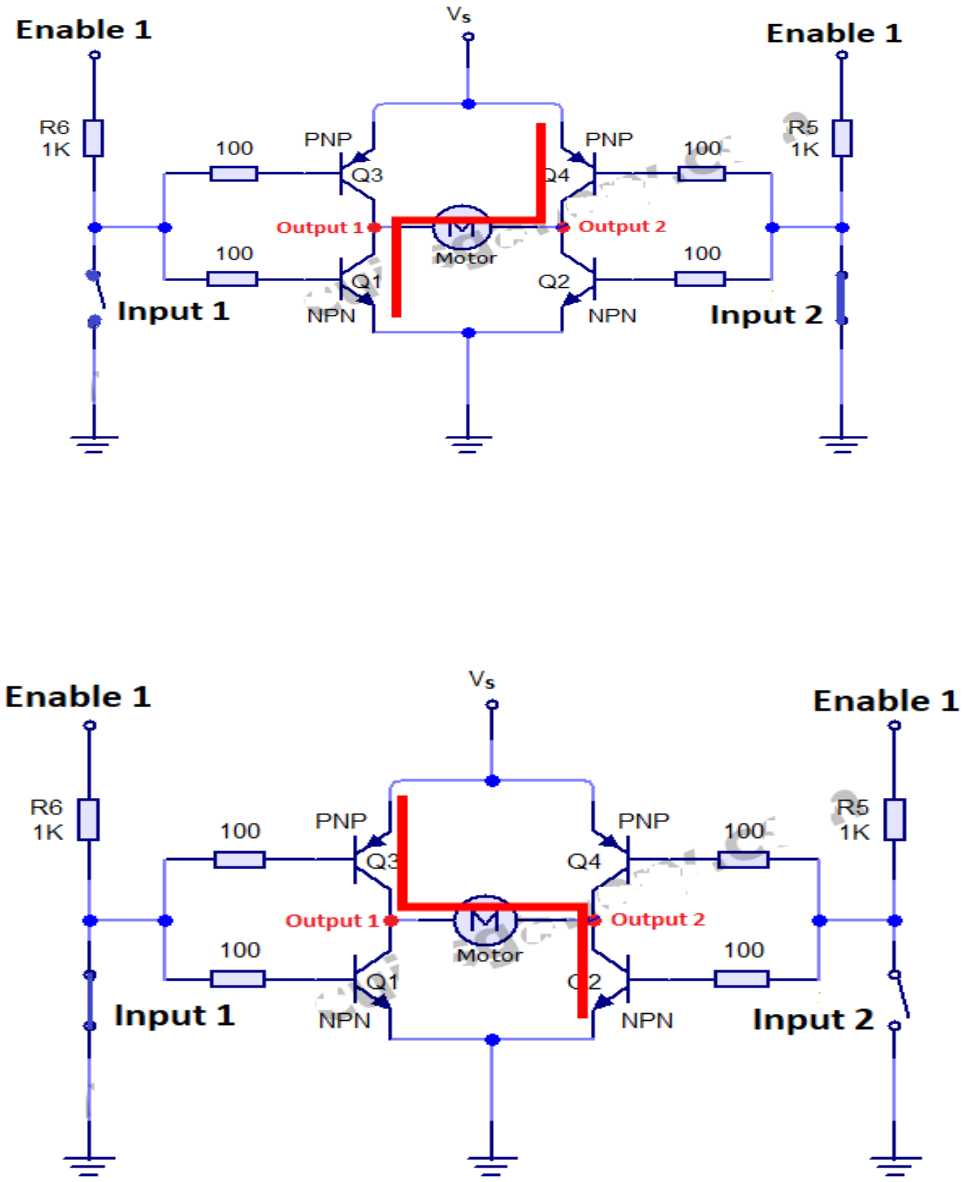
Annexe1

驰海电机 CHIHAI MOTOR	CHR-GM25-370-12K-20 Specification	
Voltage	DC 6.0 V	DC 12.0 V
No-load speed	300 rpm	600 rpm
No-load current	0.12 A MAX	0.2 A MAX
Rated load	1.8 kg.cm	3.5 kg.cm
Rated speed	160 rpm	350 rpm
Rated current	0. 8 A MAX	1.5 A MAX
Rated power	2.1 W	6.6 W
Stall torque	≥ 3.0 Kg.cm	≥ 6 Kg.cm
Stall current	≤ 2.7 A	≤ 5.5 A
Output pulse number	Basics 11PPR×i20.4=224.4PPR	
RATIO	1 : 20.4	

Product photo

Figure(1) : Spécification GM-370

Annexe2



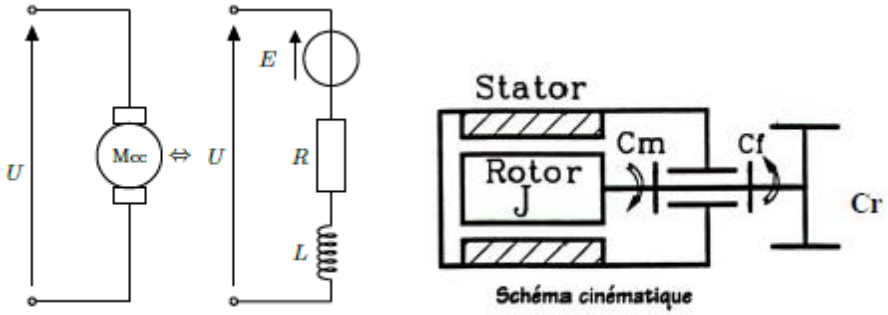
Figure(2) : Circuit équivalent L293D

Annexe3

1. Modélisation d'un moteur à courant continu.

Un moteur à courant continu commandé par l'induit est utilisé pour commander en vitesse un axe de robot.

Le schéma fonctionnel décrivant le fonctionnement du moteur est le suivant:



Figure(3) : Circuit équivalent de moteur a courant continue.

1.1Les équations différentielles régissant le comportement du moteur sont:

$U(t) = e(t) + R.i(t) + L.\frac{di}{dt}$ | $Cm(t) = K_c.\, i(t)$

$J.\frac{d\omega}{dt} = Cm(t) - Cf(t) - Cr(t)$ | $Cf(t) = a.\,\omega\,(t)$

$E(t) = K_e\,.\,\omega\,(t)$

Avec les notations suivantes:

U : tension aux bornes de l'induit en v

I : Courant dans l'induit en A

R : Resistance aux bornes de l'induit R=0.1 Ω

L : Inductance aux bornes de l'induit L=0,5 mH

E : Force électro-motrice en V

J : Moment d'inertie J=0.01 kg.m²

Cf : Couple de frottement en m.N

a : Coefficient de frottement visqueux en m.N.rad^{-1}.s

Cm : couple moteur en m.N.

Cr : couple résistant en m.N.

Ke : constante de f.e.m. Ke =0,1 V/rd/s.

Kc : constante de couple Kc =0,1 Nm

ω : pulsation de rotation du moteur en rad.s^{-1}

Bibliographie

[1] Antoine Masset, le meilleur des mondes les technologies : un progrès pour les hommes,2020 ;http ://lecube.com/revue/tous-createurs/les-technologies-un-progres-pour-les-hommes.

[2] Electromécanique pro, l'automatique les automatismes séquentiels les asservissements génie électrique, 2018, https ://www.electromecaniqueprof.com/2017/04/lautomatiquelesautomatismes.html,22.03.2020.

[3] Christophe Le Lann, Le PID utilisé en régulation de position et/ou de vitesse de moteurs électriques, 2007. https ://www.academia.edu/7377889/Cours_PID_Moteur?auto=download

[4] PROUVOST PATRICK, « Automatique contrôle et régulation », EditionDunod ,2010.

[5] Soumya Sekhsokh, Kawtaroukili, ETUDE D'UNE BOUCLE DE REGULATION DE NIVEAU : - IMPLEMENTATION DU REGULATEUR ET REGLAGE DU PROCÉDÉ, rapport de projet de fin d'étude en génie des procède, école supérieure de technologie –FES-, Maroc, 2011.

[6] S.GOURARI, LA REGULATION INDUSTRIELLE DES ECHANGEURS DE CHALEUR CALCUL ET DIMENSIONNEMENT, Courrier du Savoir – N°1 pp.19-24,2013 https ://docplayer.fr/48388763-La-regulation-industrielle-des-echangeurs-de-chaleur-calcul-et-dimensionnement.html.

[7] TOUCHERIFT Houcine, TAOUNIT Kocéila, Etude de la régulation P PI PID de débit sur le didactique de type 38-001 département de génie électrique et informatique département d'électroniques, université de Mouloud Mammeri Tizi-Ouzou ,2017 .

[8] Ayoub CHRIFI EL, IDRISSI Mohammed BOUJIDA, commande par pid (supervision et maquette), rapport de projet, université de LORRAINE ,France, 2017,https ://fr.slideshare.net/MohammedBoujida2/rapport-de-projet-commande-par-pid.

[9] Dr. Leila BOUCERREDJ, Pr. Abdelkrim MOUSSAOUI, brochure pédagogique TP régulation industrielle, Université 8 Mai 1945 – Guelma, 2016. http ://dspace.univ-guelma.dz:8080/xmlui/bitstream/handle/123456789/609/Brochure_régulation.pdf?sequence=1&isAllowed=y .

[10] Hugues GARNIER, Automatique continue Systèmes bouclés : commande, stabilité & performances université de lorraine, 2019, http://w3.cran.univ-lorraine.fr/perso/hugues.garnier/Enseignement/Auto/E-Auto-Systemes_boucles.pdf.

[11] Mr si brahim madjid, Mr djadane mustapha, étude et modélisation sous matlab-simulink d'une commande d'un système hydraulique et validation sur une maquette expérimentale, master académique en électrotechnique université mouloud Mammeri de Tizi-ouzou, 2010.

[12] BOUICHE Hachemi, BRAHAMI Mohamed , mini projet command pidd un moteur a courant continu , Université Abderrahmane Mira – Bejaia,2010 https://fr.scribd.com/doc/33033914/Commande-PID-d-un-moteur-a-courant-continu.

[13] Dr,BEKAKRA Youcef , cours de technique de commande département de génie électrique, Université EchahidHamma Lakhdar – El Oued https://www.academia.edu/24751009/Cours_de_Niveau_2_ième_Année_Master_Commande_Electrique_Techniques_de_Commande.

[14] SaidiaBdelhamide,larégulation,2012,https://fr.scribd.com/document/93882487/regulateurPID?fbclid=IwAR22zxWtJ3UVQqLSNAe65fCpAFYGpLq6gJuqpnf94IdzO6EGsvRZHaunznE ,04.06.2020.

[15] Hichem ZAYANI, Support de cours : Régulation et contrôle des systèmes de climatisation, Institut Supérieur des Etudes Technologiques de Sfax , 2015 https://fr.scribd.com/document/397072195/Modele-de-Broida-pdf.

[16] F. Mudry Ajustage des Paramètres d'un Régulateur PID, , école d ingénieurs de Canton de Vaud DEPARTEMENT D'ELECTRICITE ET INFORMATIQUE,2006 http://freddy.mudry.org/public/NotesApplications/NAPidAj_06.pdf.

[17] Mohammed-Karim FELLAH, Automatique 1 et 2 (Asservissements Linéaires Continus), Université DjillaliLiabès – Sidi Bel-Abbès Faculté de Technologie Département d'Electrotechnique,2013, https://www.univ sba.dz/fsi/downloads/Cours_Complet_Automatique.pdf

[18] S. TLIBA, M. JUNGERS, Y. CHITOUR COMMANDE DES PROCESSUS,ASSERVISSEMENTS NUMÉRIQUES Notes de cours Université Paris-Sud XI - ENS de Cachan https://www.coursehero.com/file/13982864/good-polymaster-final1/.

[19] A.Meghebbar,Module Commande Numérique. , Université AboubekrBelkaid, Tlemcen https://ft.univ-tlemcen.dz/assets/uploads/pdf/departement/gee/Support-Cours-Commande-Numerique-Master-Automatique10.pdf.

[20] Dr. KHARROUBI Larbi, Eléments de Régulation Numérique Cours et Exercices, Université des Sciences et de la Technologie d'Oran, 2019 https://www.univ-usto.dz/images/coursenligne/ern_kl.pdf.

[21] MrAous.Walid ,MrSeddaoui. Ghiles Conception et réalisation d'un régulateur PID numérique à base d'un microcontrôleur PIC 16f877a Université Mouloud Mammeri De Tizi-Ouzou ,2014.

[22] Mr MELLAH Rabah, Mr DJIOUA Smail , Conception et réalisation d'une machine CNC, UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU,2015.

[23] GaetanCottrez, Pourquoi vous devriez utiliser des cartes arduino dans votre projets domotique? ,https://www.maison-et-domotique.com/72194-devriez-utiliser-cartes-arduino-vos-projets-domotique/,28.02.2012.

[24] J.M. Hughes, Arduino : le guide complet, 2018 Éditions First.

[25] Khouidmi Houari, Polycopié Pédagogique NIVEAU: M1 Automatique & Informatique Industrielle TP : Systèmes Embarqués & Systèmes Temps Réel, 2017.

[26] Fabien Le Bris, Cours Arduino - apprendre à réaliser un prototype à base d'Arduino. 2013, https://fleb.developpez.com/tutoriels/arduino/univers_arduino/part2,08.07.2020.

[27] Simon Landrault (Eskimon), Hippolyte Weisslinger (olyte), Arduino : Premiers pas en informatique embarquée, Édition 2014.

[28] Frédéric Genevey& Jean-Pierre, Arduino à l'école , Edition,VERSION 5.2,2018 http://arduino.education/wpcontent/uploads/2018/08/Arduino_cours_sept2018.pdf

[29] Meftah Ahmed "Arduino – présentation arduino | [تاسايزينوتتايدتنم] " https://www.tunisia-sat.com/forums/threads/3300077/ ,08.07.2015.

[30] Erik Bartman, Le grand livre d'Arduino ,2eme édition ,2013.

[31] MOHAMMED AZHER THERIB, double gate security system based on RFID technology , ,Almustaqbel university college ,Babylon ,Iraq , 2015.

[32] Astalaseven,Eskimon et olyte , arduino pour bien commencer en electronique et en programmation , 2012,https://wiki.mdl29.net/lib/exe/fetch.php?media=elec:arduino-pour-bien-commencer-en-electronique-et-en-programmation.pdf,30.07.2020.

[33] MICHEL FAYAT, électronique, domotique et robotique avec Arduino,Arduino UNO 2011 http://diy-duino.blogspot.com/2011/10/arduino-uno.html, 24.05.2020.

[34] YOUNSI A. ROBOT SUIVEUR DE LIGNE, La carte ARDUINO UNO, 2016, https://docplayer.fr/13362914-La-carte-arduino-uno.html.

[35] François Pecquery, Arduino-passion, 2016, https://pecquery.wixsite.com/arduino-passion/la-liaison-serie,15.07.2020.

[36] Jarod.p, l'arduino, 2018, https://ismvsectioninfo.wordpress.com/2018/09/24/larduino/, 25.07.2020.

[37] LECHALUPÉ Julien, Cours d'initiation à Arduino ASTUPS – CampusFab, ,Université Paul Sabatier ,2014. http://philippelopes.free.fr/ArduinoUnoCoursInitiations

LechalupeJulien2014.pdf

[38] Olivier Engler, Programmer avec Arduino en s'amusant, Éditions First, 2017.

[39] CHRISTOPHE ULTRÉ,Une carte pour vos projets, 2013, https://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/techniques/5187/5187-186-p84.pdf,20.6.2020.

[40] Ismail sarwar, Advantages and Disadvantages of Using Arduino,October 1 ,2016, http://engineerexperiences.com/advantages-and-disadvatages.html, 26/7/2020.

[41] Mami Wadï, Télé-Information Compteur Electrique/Gaz/Eau, 2012, https://www.academia.edu/40609461/TéléInformation_Compteur_Electrique_Gaz_Eau

[42] EVELYN B. WISE Réaliser ses premiers projets électroniques avec une carte ArduinoUno, 2019, http://www.artisans-services.net/realiser-ses-premiers-projets-electroniques-avec-une-carte-arduino-uno/,16.04.2020.

[43] Nathanaël Cherrier, Arduino ou circuit homemade?, arduino, c, robotic, electronic, 2012,https://blog.nathanaelcherrier.com/fr/arduino-ou-circuit-homemade/ ,14.06.2020

[44] Donna M. Matthews, Arduino Nano : Avantages Et Inconvénients, 2019, http://www.jca-informatique.com/arduino-nano-avantages-et inconvenients/

,12.07.2020.

[45] David Gilbert, AnalogWrite() DESCRIPTION , 2012,,http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.AnalogWrite ,23.07.2020 .

[46] Texas Instruments, L293x Quadruple Half-H Drivers, 2016,https://electrotoile.eu/arduino-moteur-DC-shield.php.26.08.2020.

[47] Vincent Alzieu, Alexandra Bellamy, les nouveaux aspirateurs Dyson utilisent des moteurs numériques, 2013,https://www.lesnumeriques.com/aspirateur/nouveaux-aspirateurs-dyson-utilisent-moteurs-numeriques-n31294.html ,19.07.2020.

[48] Sami Soudani, Redressement & Filtrage d'une tension, 2016 http://www.soudanisami.com/index.php/autres-ressources?id=19,18.08.2020

[49] Jérôme Darmont Programmation sous Delphi Faculté de Sciences Économiques et de Gestion université lumière, lyon, 2000, http://eric.univ lyon2.fr/~ricco/cours/cours/delphi_first_approche.pdf,27.08.2020.