الجمهـــوريـــــة الجزائـــريـــــة الديمقـــراطيـــــة الشـعبيـــة
République Algérienne Démocratique et Populaire
وزارة التـــعـلـيـــــم العـــــالـــــي والبـحـــــث العـلمـــي
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

جامعة وهران 2 محمد بن أ حمد
Université d'Oran 2 Mohamed Ben Ahmed

معهد الصيانة و الأمن الصناعي
Institut de Maintenance et de Sécurité Industrielle

Département de Maintenance en Instrumentation

MÉMOIRE

Pour l'obtention du diplôme de Master

Filière : Génie industriel.

Spécialité : Ingénierie de Maintenance en Instrumentation.

Thème

La régulation PID d'un système réel identifié par l'interface Arduino Matlab.

Présenté et soutenu publiquement par :

FEKHAR Said Lotfi et ABBAD Badreddine

Devant le jury composé de :

Nom et Prénom	Grade	Etablissement	Qualité
MEKKI Ibrahim El Khalil	MCA	IMSI-Univ d'oran 2	**Président**
ADDA NEGGAZ Samir	MAA	IMSI-Univ d'oran 2	**Encadreur**
TITAH Mawloud	MAA	IMSI-Univ d'oran 2	**Examinateur**

Année 2019/2020

Remerciements

Nos remerciements vont tout premièrement à Dieu « Allah » Le Tout Puissant pour la volonté, la santé et la patience qu'il nous a donné pour terminer notre travail.

Tout d'abord, nous tenons à exprimer notre profonde gratitude et notre vifs remerciements, respectivement à notre encadreur Mr. ADDA NEGGAZ Samir , enseignant à l'institut de Maintenance et de Sécurité Industrielle (IMSI) de l'université d'Oran 2, pour avoir accepté de diriger ce mémoire et de sa patience durant la période de l'encadrement et pour la confiance qu'il nous a témoigné sans oublier ses conseils avisés, ses encouragements et son soutien indéfectible, tout le long de cette étude.

Nous exprimons également nos reconnaissances et nos sincères remerciements à Mr. MEKKI Ibrahim El Khalil, chef département et enseignant à l'institut de Maintenance et de Sécurité Industrielle (IMSI) de l'université d'Oran 2, qui nous a fait l'immense honneur d'accepter de présider le jury.

Nous tenons beaucoup à témoigner l'expression de notre gratitude et notre reconnaissance à Mr. Titah Mawloud enseignant à l'institut de Maintenance et de Sécurité Industrielle (IMSI) de l'université d'Oran 2, pour l'honneur qu'il nous a fait de prendre part à ce jury en qualité d'examinateur.

Nous tenons également à témoigner notre gratitude et notre reconnaissance à tous les enseignants qui nous ont éduqué et formé pour faire face à la vie, Surtout Mr. HASSINI Abdellatif, professeur et créateur de notre spécialité.

Enfin, nous ne saurons oublier d'exprimer nos sentiments les plus profonds et nos vifs remerciements à nos famille et notre entourage, ainsi qu'à tous nos amis et collègues qui n'ont cessé de nous encourager et de nous soutenir.

Dédicaces

Je dédie le fruit de ce travail tout d'abord à L'âme de notre vénéré Prophète MOHAMED, Que la prière et le salut soit sur lui. Ensuite à Ma mère bien aimée, qui m'a comblé de son amour et de sa tendresse, et qui a éclairé mon chemin tout au long de ma vie. A Mon père, qui a veillé à mon éducation et fait de moi un homme droit et sage ; Mon cher frère et mes chères sœurs bien aimées qui me chérissent ; Mes grands-parents que j'affectionne tendrement, et qui ne cessent de prier pour moi ; Mon cher oncle Mustafa que j'adore particulièrement; Mes très chères tantes Houda et Soumia que j'aime, et qui m'ont toujours soutenu tout au long de ma vie ; Mon cousin Dr. Bennihi Aymen Salah qui est le meilleur homme que j'ai rencontré dans ma vie ; Mes amis qui me connaissent chacun son nom, particulièrement : Mon meilleur ami Kazouz Mohammed Lamine et mon Binôme Fekhar Saïd Lotfi. A Ma moitié Hayet qui m'a toujours aidé et encouragé dans tous les moments.

Tous les collègues du l'Institut de l'IMSI de l'université d'Oran et tous ceux qui m'estiment et pensent à moi.

ABBAD Badreddine

Dédicaces

Je dédie ce modeste travail tout d'abord à L'âme de notre vénéré Prophète MOHAMED, Que la prière et le salut soit sur lui. Ensuite à mes très chers parents, en témoignage et en gratitude de leurs dévouements et leurs soutiens permanents durant toutes mes années d'études, leurs sacrifices illimités, leur réconfort moral et tous les efforts qu'ils ont consentis pour mon éducation et mon instruction pour me voir réussir un jour. Que Dieu les gardes.

À mes chers frères et ma chère sœur, mes oncles surtout Ibrahim et Hamza, mes tantes surtout Khoukha et son mari Youcef, et à toute ma famille.

À mon binôme Abbad Badreddine.

À tous mes amis et mes collègues de L'Institut de Maintenance et de Sécurité Industrielle.

Fekhar Saïd Lotfi

Table des matières

Table des matières 4
Liste des figures 6
Liste des tableaux 8
Liste des symboles et abréviations 9
Introduction générale 10
I . Chapitre 1 : Le régulateur PID 11
I.1 Introduction 12
I.2 Le régulateur PID 12
I.2.1 Définition du régulateur PID 12
I.2.2 Régulateur à action proportionnel (P) 13
I.2.3 Régulateur à action proportionnelle intégrale (PI) 14
I.2.4 Régulateur à action proportionnelle intégrale dérivée (PID) 16
I.3 Les structures du régulateur PID 18
I.3.1 Structure série 18
I.3.2 Structure parallèle 19
I.3.3 Structure mixte 19
I.4 Réglage des paramètres 20
I.4.1 Méthodes de Ziegler-Nichols 20
I.4.2 Résumé sur l'action des paramètres (les coefficients) 23
I.5 Conclusion 23
II . Chapitre 2 : La carte Arduino 25
II.1 Introduction 26
II.2 La carte Arduino 27
II.2.1 Définition de la carte Arduino 27
II.2.2 Types de la carte Arduino 27
II.2.3 Les différentes cartes 27
II.2.4 Le but de l'utilité 30
II.2.5 Applications 30
II.3 La carte Arduino Uno 30
II.3.1 Les avantages de la carte Arduino UNO 30
II.3.2 La partie matérielle 31
II.3.3 Le signal PWM 36
II.4 Conclusion 39
III . Chapitre 3 : L'interface Arduino Matlab 40

III.1 Introduction 41
III.2 Matlab Simulink 41
III.2.1 Définition de Matlab 41
III.2.2 Définition de Simulink 42
III.3 L'interface entre Simulink et Arduino 42
III.3.1 Obtenir le support Arduino sur Logiciel Matlab 43
III.3.2 Configuration des paramètres 45
III.3.3 Faire des modifications pour la compatibilité des paramètres 48
III.4 La bibliothèque Arduino sur Simulink 48
III.5 Les avantages et les enjeux de l'interface 50
III.6 Conclusion 50
IV . Chapitre 4 : L'acquisition des données et l'identification du système 51
IV.1 Introduction 52
IV.2 Les matériaux utilisés et le branchement du système 52
IV.2.1 Le ventilateur du processeur à 4 sorties 52
IV.2.2 La carte L298N 54
IV.2.3 Le transformateur de tension 55
IV.2.4 Le branchement du système 55
IV.3 L'acquisition des données 56
IV.4 L'identification du système 60
IV.5 La fonction de transfert du système 62
IV.6 Conclusion 63
V . Chapitre 5 : Implémentation de la commande PID sur Simulink 64
V.1 Introduction 65
V.2 La régulation PID numérique 65
V.2.1 L'action PI dans un régulateur numérique 65
V.2.2 L'action PID dans un régulateur numérique 66
V.3 La régulation PID pratique 67
V.4 Implémentation du régulateur PID sous Simulink 70
V.4.1 La régulation PI 70
V.4.2 La régulation PID 71
V.4.3 Interprétation des résultats 73
V.5 Conclusion 73
Conclusion générale 75
Références Bibliographiques 76
Résumé 78

Liste des figures :

Figure	Titre	Page
Figure I.1 :	Asservissement par un régulateur PID.	13
Figure I.2 :	Schéma d'un régulateur électronique P.	14
Figure I.3 :	Schéma d'un régulateur électronique PI.	15
Figure I.4 :	Exemple d'un schéma du régulateur électronique PID (structure parallèle)	17
Figure I.5 :	Régulateur PID montage en série.	18
Figure I.6 :	Régulateur PID montage parallèle.	19
Figure I.7 :	Régulateur PID montage mixte.	19
Figure I.8 :	Réponse indicielle du processus.	21
Figure II.1 :	Les modèles d'Arduino.	29
Figure II.2 :	Schéma d'une platine Arduino Uno.	31
Figure II.3 :	Un microcontrôleur de la carte Arduino.	33
Figure II.4 :	Processus de génération PWM.	36
Figure II.5 :	Modulation de largeur d'impulsion.	38
Figure III.1 :	Fenêtre Add-Ons.	43
Figure III.2 :	La barre de recherche du Support Package.	44
Figure III.3 :	Support Package compatible.	44
Figure III.4 :	Fenêtre Add Support Package.	44
Figure III.5 :	Icône de la bibliothèque.	45
Figure III.6 :	Fenêtre configuration des paramètres de Simulink.	45
Figure III.7 :	Fenêtre Hardware board.	46
Figure III.8 :	La carte Arduino connectée avec COM6.	46
Figure III.9 :	Fenêtre Host-board connection.	47
Figure III.10 :	Fenêtre Serial port properties.	47
Figure III.11 :	Les modes de connexion entre Simulink et Arduino.	48
Figure III.12 :	Les blocs d'Arduino les plus utilisés.	49

Figure IV.1 :	Modèle du ventilateur à 4 sorties.	53
Figure IV.2 :	Les entrées et sorties de la carte L298N.	54
Figure IV.3 :	L'alimentation du système (transformateur 220V-12V)	55
Figure IV.4 :	Le montage du système.	56
Figure IV.5 :	Schéma blocs de l'acquisition de l'entrée du système.	57
Figure IV.6 :	Le sous-système du compteur des impulsions.	57
Figure IV.7 :	Le sous-système du filtre passe-bas triplé.	58
Figure IV.8 :	Le graphe de l'entrée du système.	59
Figure IV.9 :	Le graphe de la sortie du système.	59
Figure IV.10 :	Les valeurs de l'entrée et la sortie du système.	60
Figure IV.11 :	L'icône de l'identification du système.	60
Figure IV.12 :	Les options de l'identification du système.	61
Figure IV.13 :	La fenêtre de déclaration de l'entrée et la sortie.	61
Figure IV.14 :	nombre des pôles et le type de système.	62
Figure IV.15 :	La réponse indicielle de la fonction G (Z)	63
Figure V.1 :	Asservissement continu corrigé numériquement.	65
Figure V.2 :	Modèle à temps continu de l'asservissement.	66
Figure V.3 :	L'icône de PID Tuner	67
Figure V.4 :	L'icône « importer la fonction de transfert »	67
Figure V.5 :	Le graphe de la réponse du système dans l'action PI.	68
Figure V.6 :	Le graphe de la réponse du système dans l'action PID	69
Figure V.7 :	Schéma synoptique de l'asservissement à implémenter.	70
Figure V.8 :	Modèle Simulink d'asservissement de vitesse du moteur par un régulateur PI.	70
Figure V.9 :	La sortie du système après l'application de l'action PI.	71
Figure V.10 :	Modèle Simulink d'asservissement de vitesse du moteur par un régulateur PID.	71
Figure V.11 :	La sortie du système après l'application de l'action PID.	72
Figure V.12 :	La sortie du système durant la perturbation avec le régulateur PID.	73

Liste des tableaux :

Tableau	Titre	Page
Tableau I.1 :	Ajustage de gains des régulateurs P, PI et PID selon la première méthode de Ziegler-Nichols.	21
Tableau I.2 :	Réglage du contrôleur par la méthode d'oscillation de Zeigler-Nichols.	22
Tableau I.3 :	Récapitulatif des paramètres PID.	23
Tableau III.1 :	La valeur de tension moyenne de la sortie pour chaque entier (PWM)	39
Tableau IV.1 :	Tableau du fonctionnement des sorties du ventilateur.	53

Liste des symboles et abréviations :

P :	Proportionnel
PI :	Proportionnel Intégral
PID :	Proportionnel Intégral Dérivé
Kp :	Gain proportionnel
Ki :	Gain intégral
Kd :	Gain dérivé
M :	La valeur mesurée
C :	La consigne
E :	Signal d'erreur ou signal d'écart
U :	La valeur d'entrée
Ti :	la constante de temps d'action intégrale **[min]**
Td :	la constante de temps d'action dérivée **[min]**
R :	Résistance **[Ω]**
C :	Capacité **[F]**
p :	Opérateur de Laplace
Tu :	Temps de retard du système **[min]**
Ta :	Temps de montée de la tangente **[min]**
Ku :	Le gain critique
IDE :	Integrated Development Environment
E/S	Entrées soties
SMD :	Surface Mounted Device
USB :	Universal Serial Bus
PWM :	Pulse Width Modulation
MLI :	Modulation de Largeur d'Impulsion
Led	Light-emitting diode
DC :	Direct current
IO :	Input/output
CPU :	Central Processing Unit
GND :	Ground
RPM :	Tour par minute **[tr/min]**
Z^{-1} :	Opérateur retard
G (Z) :	La fonction de transfert discrète
Ts, Te :	Temps d'échantillonnage **[min]**
Tf :	Temps de filtrage **[min]**
Tr :	Temps de réponse **[min]**
D % :	Le dépassement

Introduction générale

Le développement extraordinaire des processeurs numériques (microprocesseurs, microcontrôleurs) et leurs larges utilisations dans tous les systèmes de contrôle (domestique et industriel) ont mené aux changements importants dans la conception des systèmes de contrôle. Leurs applications sont à faible coût et elles les rendent appropriés pour l'usage dans des systèmes de contrôle. Dans certaines branches d'industries, le problème de contrôle de la vitesse d'un système est souvent une problématique. La nature du système et le frottement du mécanisme de contrôle et d'autres facteurs rendent le système non linéaire. Dans nos jours, le contrôleur le plus connu de processus industriel est le contrôleur PID en raison de sa simplicité, robustesse, la fiabilité élevée et peut être facilement mis en application sur n'importe quel processeur.

La pertinence de cette problématique s'est d'ailleurs confirmée au cours des travaux préparatoires de la présente étude : « la conception et la réalisation du système », et le plus important point du cette étude qui permet de modéliser le système et appliquer la régulation.
On obtient des résultats pour déterminer quel est le bon régulateur qui assure la meilleure réponse de notre système qui est sous forme d'un ventilateur de processeur de PC, alors notre thème est intitulé « La régulation PID d'un système réel identifié par l'interface Arduino/Matlab ». Ce mémoire tend ainsi à démontrer les points suivants :

La première partie est consacrée à la régulation PID, nous allons définir c'est quoi le régulateur PID et identifier ses paramètres : proportionnel, dérivé et intégral.

Le chapitre deux est consacré à la carte Arduino, nous allons présenter la carte Arduino qui va jouer le rôle d'une carte acquisition et du contrôleur.

Le chapitre trois décrit comment réaliser l'interface entre la carte Arduino et le logiciel Matlab pour faciliter le travail avec Simulink.

Dans le quatrième chapitre, nous allons parler avec tous les détails sur l'identification du système et comment le modéliser et obtenir sa fonction de transfert.

Le cinquième chapitre explique la conception du contrôleur PI, ensuite le contrôleur PID, l'implémentation sur la carte Arduino et les résultats expérimentaux.

I. Chapitre 1 : Le régulateur PID

I.1 Introduction :

Le régulateur standard le plus utilisé dans l'industrie est le régulateur PID (proportionnel intégral dérivé). Ce régulateur linéaire est basé sur une structure très simple dont le fonctionnement ne dépend que de trois coefficients, qui sont les gains appliqués sur les signaux : proportionnel (Kp), intégral (Ki) et dérivé (Kd)

Le terme de régulation est employé lorsqu'on cherche à combattre des perturbations afin de garder une valeur constante par exemple : une vitesse, position, température, pression… etc.

La régulation mesure en permanence par les capteurs du système à régler puis transmet ces informations au régulateur, celui-ci compare cette mesure à la valeur désirée (la consigne), puis suivant ses coefficients, le régulateur va transmettre ses ordres aux actionneurs (moteurs, contacteur, vannes, thermostat… etc), afin de corriger les erreurs et conduire la sortie du système vers la consigne.

Dans ce chapitre, on présente les régulateurs PID et les méthodes de synthèse de Ziegler et Nichols (1942). Dans chacune de ces méthodes, les trois gains sont fixés en suivant une procédure de réglage qui garantit un fonctionnement optimal selon un ou plusieurs critères, dans tous les cas, la fonction de transfert du régulateur PID reste linéaire.

À ce stade, il est important d'insister sur le fait que les méthodes présentées ci-après ne sont applicables qu'à des processus non oscillants.

I.2 Le régulateur PID :

I.2.1 Définition du régulateur PID :

Le régulateur PID, appelé aussi correcteur PID (proportionnel, intégrateur, dérivateur) est un système de contrôle qui permet d'effectuer une régulation en boucle fermée d'un procédé industriel.

Le régulateur compare une valeur mesurée M sur le procédé avec une valeur de consigne C, la différence entre ces deux valeurs (le signal d'erreur ou signal d'écart E) est alors utilisée pour calculer une nouvelle valeur d'entrée du processus U tendant à réduire au maximum l'écart entre la mesure et la consigne (signal d'erreur le plus faible possible)

Le contrôle par PID peut ajuster les sorties du procédé, en fonction de l'amplitude du signal d'erreur, et en fonction du temps. Il donne des résultats plus précis et un contrôle plus stable. [1-1]

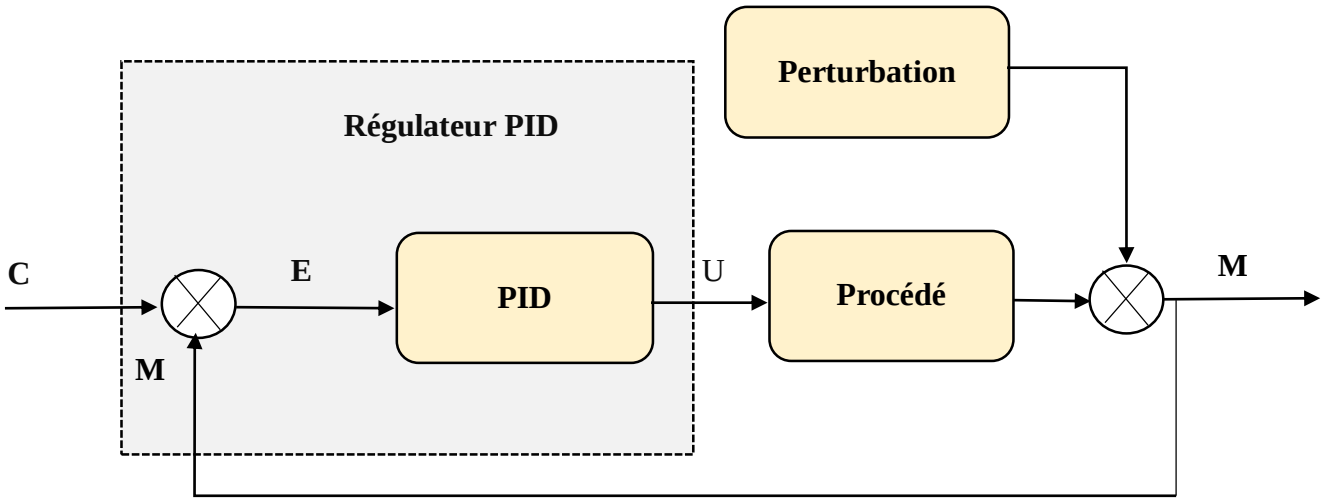


Figure I.1 : Asservissement par un régulateur PID.

I.2.2 Régulateur à action proportionnel (P) :

Dans le cas de la commande proportionnelle pure, l'action proportionnelle consiste à générer une action qui varie de façon proportionnelle au signal d'erreur, la loi de commande se réduit à :

$$\boldsymbol{U(p) = Kp.E(P)} \qquad \textbf{(I.1)}$$

I.2.2.a Effets du régulateur à action proportionnel :

- Diminue l'erreur statique.
- Diminution du temps de montée.
- Augmente la rapidité et le dépassement.
- Augmente le temps de stabilisation.

I.2.2.b Le schéma d'un régulateur électronique P :

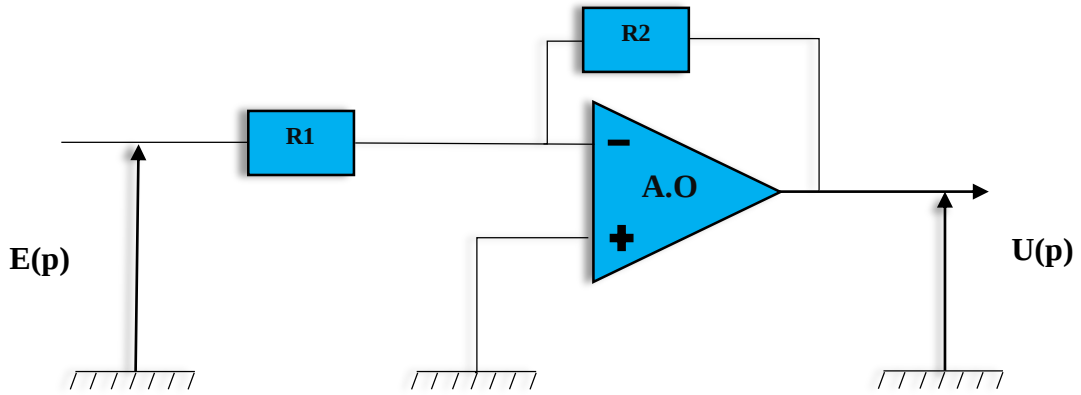


Figure I.2 : Schéma d'un régulateur électronique P.

I.2.2.c Fonction de transfert d'un régulateur proportionnel :

La fonction de transfert du régulateur proportionnel est comme suit :

$$\boldsymbol{G(p) = \frac{U(p)}{E(p)} = -\frac{R2}{R1} = Kp} \qquad \textbf{(I.2)}$$

I.2.3 Régulateur à action proportionnel intégral (PI) :

Le correcteur du type PI est une régulation du type P avec un terme intégral, l'intérêt d'ajouter le terme intégral est d'annuler l'erreur statique et d'augmenter la précision en régime permanent. L'idée est d'intégrer l'erreur depuis le début et d'ajouter cette erreur à la consigne, lorsque la mesure rapproche de la valeur de la consigne, l'erreur devient de plus en plus faible, la loi de commande se réduit à :

$$\boldsymbol{U(p) = Kp.E(p) + Ki.\frac{E(p)}{p} = E(p).(Kp + \frac{Ki}{p})} \qquad \textbf{(I.3)}$$

- Avec : $Ki = \frac{1}{Ti}\ en\ (min^{-1})$
- Tel que : Ti la constante de temps d'action intégrale en (min)

I.2.3.a Effets du régulateur à action proportionnel-intégral :

- L'action intégrale prend en compte (intègre) le passé.
- Le correcteur PI peut éliminer l'erreur statique.
- Diminution du temps de montée.
- Augmentation du temps de stabilisation.
- Augmentation du dépassement.

I.2.3.b Le schéma d'un régulateur électronique PI :

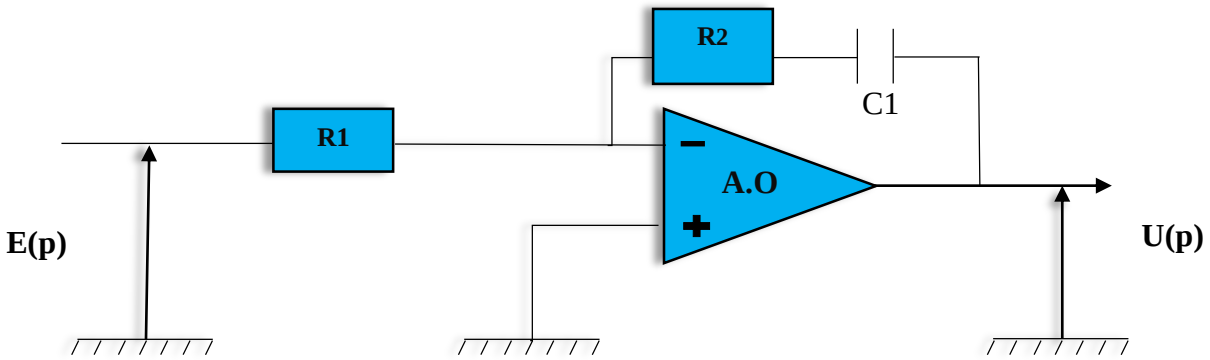


Figure I.3 : Schéma d'un régulateur électronique PI.

I.2.3.c Fonction de transfert d'un régulateur proportionnel intégral :

La fonction de transfert du régulateur proportionnel intégral est comme suit :

$$\mathbf{G(p)} = \frac{\mathbf{U(p)}}{\mathbf{E(p)}} = -\frac{\mathbf{R2}+\frac{\mathbf{1}}{\mathbf{C1.p}}}{\mathbf{R1}} = -\frac{\mathbf{R2}}{\mathbf{R1}}\left(\mathbf{1}+\frac{\mathbf{1}}{\mathbf{R2.C1.p}}\right) \qquad \textbf{(I.4)}$$

I.2.4 Régulateur à action Proportionnel Intégral Dérivée (PID) : [1-2]

Les termes proportionnel et intégral peuvent amener un dépassement de la consigne et des oscillations. Cela implique pour le moteur des inversions de polarité, ce qui est loin d'être idéal. Pour limiter ce phénomène indésirable, on introduit un troisième élément : le terme dérivé. Son action va dépendre du signe et de la vitesse de variation de l'erreur, et sera opposée à l'action proportionnelle. Elle devient prépondérante aux abords de la valeur demandée lorsque l'erreur devient faible, que l'action du terme proportionnel faiblit et que l'intégrale varie peu : elle freine alors le système, limitant le dépassement et diminuant le temps de stabilisation.

La loi de commande se réduit à :

$$\boldsymbol{U(p) = Kp.E(p) + Ki.\frac{E(p)}{p} + Kd.pE(p) = E(p).(Kp + \frac{Ki}{p} + Kd.p)} \qquad \textbf{(I.5)}$$

I.2.4.a Effets du régulateur à action proportionnelle intégrale dérivée

- Diminue le dépassement.
- Éliminer l'erreur statique.
- Diminue le temps de montée.
- Diminue le temps de stabilisation.

I.2.4.b Le schéma d'un régulateur électronique PID :

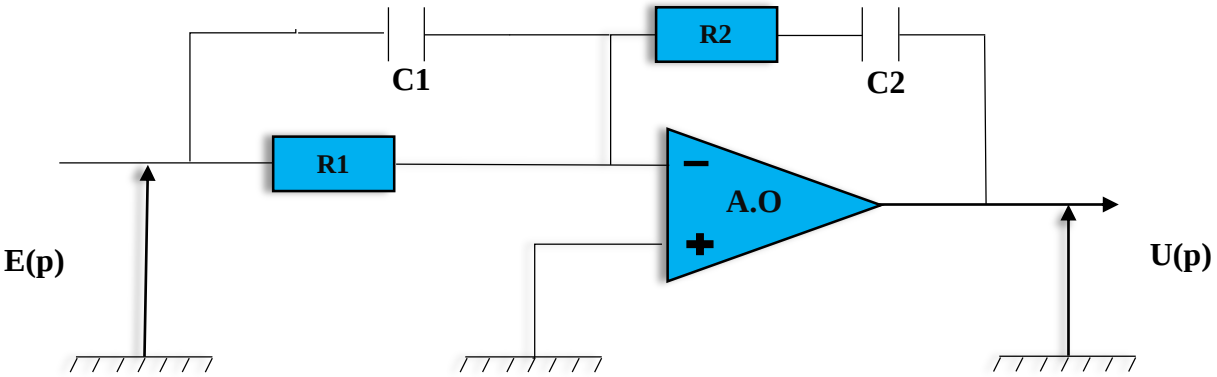


Figure I.4 : Exemple d'un schéma du régulateur électronique PID (structure parallèle)

I.2.4.c Fonction de transfert d'un régulateur proportionnel intégral dérivé : [1-3]

$$G(p) = \frac{U(p)}{E(p)} = -\frac{R1.C1+R2.C2}{R1.C2} + \frac{1}{R2.C2.p} + R2.C1.p \qquad \textbf{(I.6)}$$

- Avec :
  - $Kp = -\frac{R1.C1+R2.C2}{R1.C2}$
  - $Ki = \frac{1}{R2.C2}$
  - $Kd = R2.C1$

I.3 Les structures du régulateur PID : [1-4]

I.3.1 Structure série :

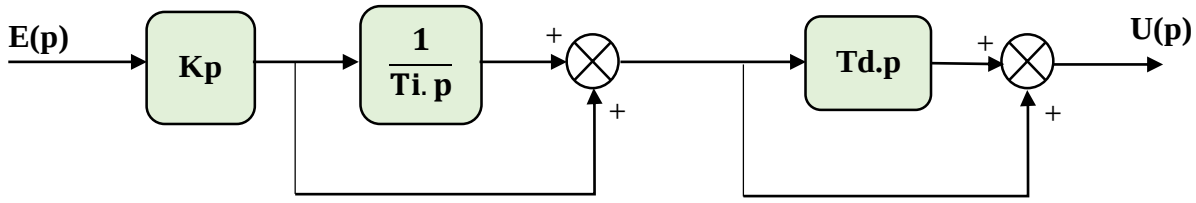


Figure I.5 : Régulateur PID montage en série.

La commande est de la forme :

$$\mathrm{U(p)} = \mathrm{Kp}\left(1 + \frac{1}{\mathrm{Ti.p}} + \mathrm{Td.\,p} + \frac{\mathrm{Td}}{\mathrm{Ti}}\right)\mathrm{E(p)} \qquad \textbf{(I.7)}$$

On pose :

$$K(p) = \mathrm{Kp}\left(1 + \frac{1}{\mathrm{Ti.p}} + \mathrm{Td.\,p} + \frac{\mathrm{Td}}{\mathrm{Ti}}\right) = Kp\frac{Ti+Td}{Ti}\left(1 + \frac{1}{(Ti+Td)p} + \frac{Ti.Td}{Ti+Td}p\right) \qquad \textbf{(I.8)}$$

On pose :

- $K'p = Kp\frac{Ti+Td}{Ti}$
- $T'i = Ti + Td$
- $T'd = \frac{Ti.Td}{Ti+Td}$

Donc la relation **(I.8)** devient :

$$\mathrm{K(p)} = \mathrm{K'p}\left(1 + \frac{1}{\mathrm{T'i.p}} + \mathrm{T'd.\,p}\right) \qquad \textbf{(I.9)}$$

I.3.2 Structure parallèle :

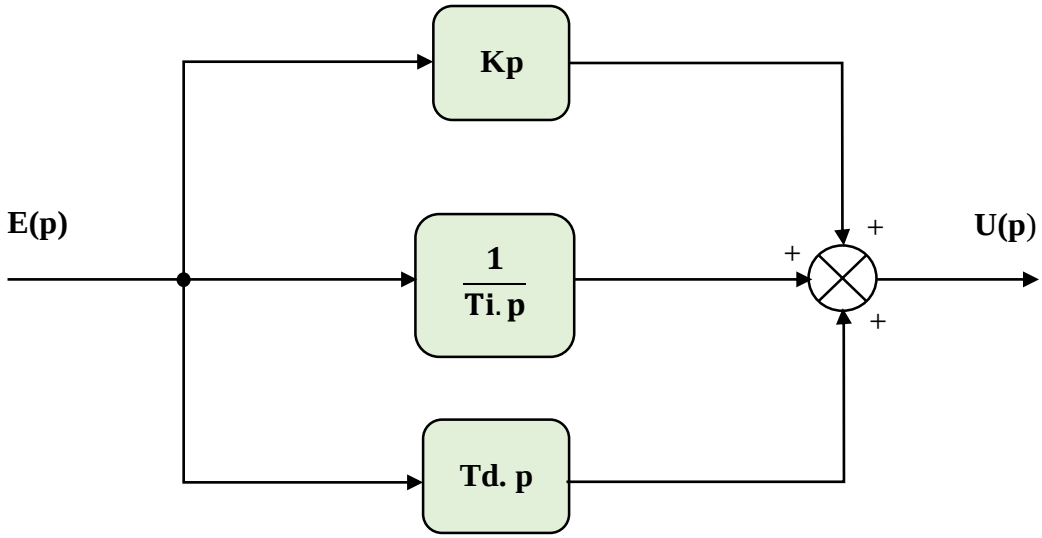


Figure I.6 : Régulateur PID montage parallèle.

La commande est de la forme :

$$U(p) = \left(Kp + \frac{Ki}{p} + Kd.p\right) E(p) \qquad \textbf{(I.10)}$$

I.3.3 Structure mixte :

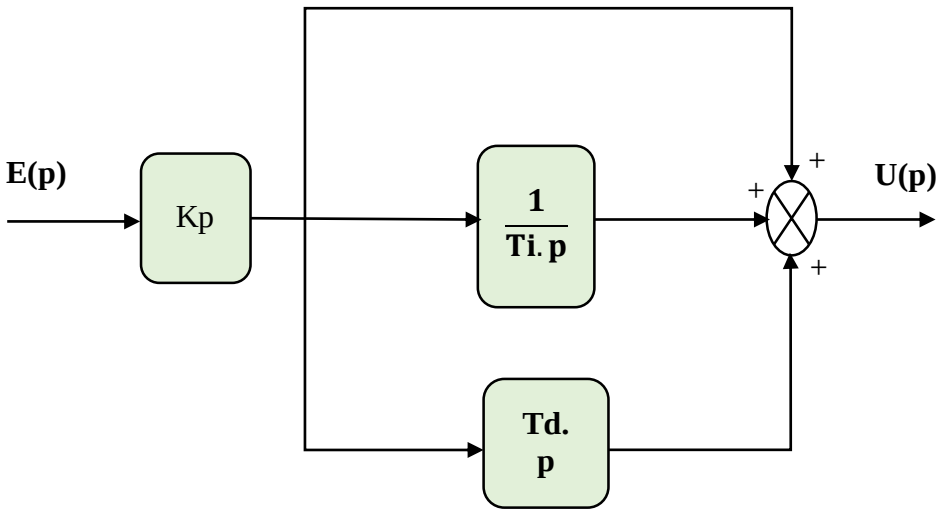


Figure I.7 : Régulateur PID montage mixte.

La commande est de la forme :

$$U(p) = Kp\left(1 + \frac{1}{Tip} + Td.p\right)E(p) \qquad \textbf{(I.11)}$$

Avec :

- $Ki = \frac{Kp}{Ti}$
- $Kd = Kp.Td$

I.4 Réglage des paramètres :

I.4.1 Méthodes de Ziegler-Nichols :

Deux méthodes classiques expérimentales ont été présentées par Ziegler et Nichols en 1942. Ces méthodes destinées à ajuster rapidement les paramètres des régulateurs P, PI et PID. La première nécessite l'enregistrement de la réponse indicielle du système à régler seul, alors que la deuxième demande d'amener le système en boucle fermée à sa limite de stabilité. Les méthodes sont basées sur la détermination de certaines caractéristiques de la dynamique des processus. Les paramètres du régulateur sont alors exprimés en termes de fonctionnalités par des formules simples. Il est important de souligner que ces méthodes ne s'appliquent en général qu'à des systèmes sans comportement oscillant et dont le déphasage en hautes fréquences dépasse -180 degrés. Ces systèmes possèdent souvent un retard pur et/ou plusieurs constantes de temps. [1-5]

I.4.1.a Méthode de la réponse indicielle : [1-6]

Cette méthode est basée sur la modélisation de l'information indicielle du processus en boucle ouverte, d'où seulement les processus simples sont utilisés, le principe est d'enregistrer la courbe de réponse du système non régulé à un échelon puis en déduire la valeur des coefficients par analyse de la réponse (i.e. "lecture graphique"), ainsi mettre le système hors ligne. C'est pour cette raison que cette méthode n'est pas très utilisée dans l'industrie. [1-11]

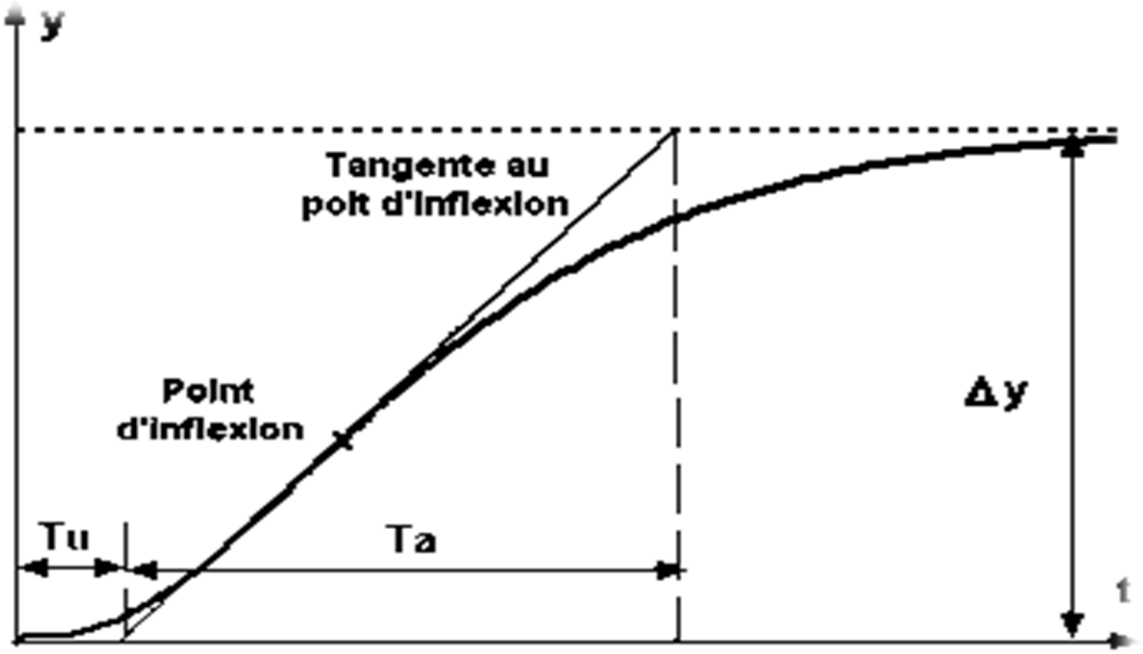


Figure I.8 : Réponse indicielle du processus. [1-6]

Ziegler et Nichols proposent de calculer les paramètres du régulateur P, PI ou PID à l'aide des recommandations suivantes :

Tableau I.1 : Ajustage de gains des régulateurs P, PI et PID selon la première méthode de Ziegler-Nichols.

Régulateur	Kp	Ti	Td
P	$\frac{T_a}{T_u}$	-	-
PI	$0.9\frac{T_a}{T_u}$	$3.3\ T_u$	-
PID	$1.2\frac{T_a}{T_u}$	$2\ T_u$	$0.5\ T_u$

Généralement les gains proportionnels (Kp) proposés par Ziegler-Nichols sont trop élevés et conduisent à un dépassement supérieur à 20 %. Il ne faut donc pas craindre de réduire ces gains d'un facteur 2 pour obtenir une réponse satisfaisante. [1-7]

I.4.1.b Méthode d'oscillation :

Cette méthode empirique de Ziegler-Nichols est très répandue dans l'industrie ou chez les automaticiens pour régler les correcteurs de ce type de chaîne. Elle a l'avantage de ne pas nécessiter de modélisation précise du système asservi, mais se contente d'essais expérimentaux, ce qui rend cette méthode très simple. [1-11]

La conception est basée sur la connaissance d'un point dans la courbe de Nyquist de la fonction de transfert du processus, à savoir le point où la courbe de Nyquist coupe l'axe réel négatif. Ce point peut être caractérisé par deux paramètres de la fréquence 180ω et le gain à cette fréquence 180K. Pour des raisons historiques le point a été appelé le point ultime et caractérisé par les paramètres $Ku=1/180\omega$ que l'on appelle le gain ultime et $Tu=2\pi/(180\omega)$ la période ultime.

NB : avec la fonction de Matlab **« Margin »**, on peut déterminer la fréquence **180 ω** du système. [1-10]

Ces paramètres peuvent être déterminés de la façon suivante :

On connecte le régulateur au moteur électrique, fixer $Ki=0$ et $Kd=0$, on augmente doucement le gain du correcteur proportionnel pur Kp jusqu'à obtenir des oscillations persistantes, pour des moteurs, il s'agira plutôt d'une vibration autour de la position demandée. Le gain dans ce cas est Ku, et la période d'oscillations est Tu, nous avons $Ku=1/(180\ \omega)$ et $Tu=2\pi/(180\ \omega)$. [1-12]

Tableau I.2 : Réglage du contrôleur par la méthode d'oscillation de Zeigler-Nichols.

Régulateur	Kp	Ki=1/Ti	Kd=Td
P	Ku/2	-	-
PI	Ku/2.2	1.2/Tu	-
PID	Ku/1.7	2/Tu	Tu/8

I.4.2 Résumé sur l'action des paramètres (les coefficients) : [1-5]

Après avoir expliqué précédemment le rôle de chaque action, proportionnelle, intégrale et dérivée, on représente un résumé sur l'action des paramètres (coefficients) dans le tableau suivant :

Tableau I.3 : Récapitulatif des paramètres PID. [1-5][1-8]

Coefficient	Temps de montée	Temps de stabilisation	Dépassement	Erreur statique
Kp	Diminue	Augmente	Augmente	Diminue
Ki	Diminue	Augmente	Augmente	Élimine
Kd	Changement faible	Diminue	Diminue	Changement faible

I.5 Conclusion :

Le PID représente les abréviations des trois actions qu'il utilise pour effectuer ses corrections, ce sont des ajouts d'un signal à un autre. Tous agissent sur la quantité régulée. Les actions aboutissent finalement à des soustractions de l'erreur de mesure, parce que le signal proportionnel est habituellement négatif.

Les trois actions du régulateur PID permettent de commander le moteur électrique à courant continu, tout en garantissant une annulation de l'erreur permanente de la sortie régulée, vis-à-vis d'échelons en entrée de consigne. Cette propriété de précision est due à la présence d'une action intégrale. L'ajout d'une action dérivée permet d'augmenter la stabilité du système, et donc de diminuer le dépassement de la réponse indicielle. L'ajout de cette dérivée n'est pas systématique et dépend des propriétés du système, du cahier de charge imposé (en termes de dépassement), dans certains cas il suffit l'utilisation d'un régulateur PI uniquement, comme l'asservissement en vitesse du système. La réalisation de l'action dérivée est préférentiellement réalisée sur la mesure, afin d'éviter la saturation de la commande.

Un des intérêts du régulateur PID, qui explique sa popularité dans le milieu industriel, est sans conteste la possibilité de le régler sans connaissance approfondie du système. En effet, on

dispose de méthodes empiriques, fondées uniquement sur la réponse temporelle du système, selon une procédure expérimentale, comme la méthode d'oscillation de Ziegler-Nichols, permettant dans la majorité des cas d'aboutir à des performances acceptables. [1-9] [1-5]

II. Chapitre 2 : La carte Arduino

II.1 Introduction :

Le projet Arduino (Arduin en français) était réalisé par une équipe d'enseignants et d'étudiants (David Mellis, Tom Igoe, Gianluca Martino, David Cuartielles, Massimo Banzi ainsi que Nicholas Zambetti), de l'école de Design interaction avec Ivrea (Italie). Avant cette période (de 2003 à 2004), ils ont rencontré un problème majeur : les outils nécessaires à la création de projets interactifs étaient complexes et coûteux (entre 80 et 100 euros). Donc ces coûts sont généralement trop élevés, il est donc difficile de nombreux projets de développement des étudiants, qui ont ralenti la mise en œuvre leur apprentissage. [2-2]

Auparavant, les outils prototypes étaient principalement utilisés en ingénierie, en robotique et en technologie. Ils sont puissants, mais le processus de développement est très long, et il est difficile pour les artistes, les concepteurs d'interaction et un plus large éventail de débutant d'apprendre et d'utiliser, donc leur objectif est de fabriquer moins de matériel. Cher et facile à utiliser. Ils voulaient créer un environnement proche du traitement, ce langage de programmation a été développé par Casey Reas et Ben Fry en 2001. Ce sont deux anciens élèves du MIT John Maeda, initiateur du projet DBN. En 2003, la thèse finale d'Hernando Barragan était de développer une carte électronique dénommée Wiring, accompagnée d'un environnement de programmation libre et ouverte. Pour ce travail, Hernando Barragan a réutilisé les ressources du projet Processing. La carte de connexion est basée sur un langage de programmation facile d'accès et adaptée au développement de projets de concepteurs, elle a donc inspiré le projet Arduino (2005). En ce qui concerne le câblage, leur objectif est de fournir un appareil facile à utiliser à faible coût, le code et les plans sont « gratuits » (c'est-à-dire que ses ressources sont ouvertes et peuvent être modifiées, améliorées et distribuées par les utilisateurs), et enfin, « multi-plates-formes » (indépendant du système d'exploitation utilisé.)

La plate-forme Arduino n'est pas seulement conçue pour être utilisée avec des cartes ou des microcontrôleurs, c'est également une solution matérielle et logicielle complète qui est plus facile à utiliser que les autres microcontrôleurs. Utiliser le logiciel officiel Arduino IDE (téléchargement gratuit), puis écrire le code à charger sur la carte dans un langage proche de C++. On peut également utiliser le logiciel Matlab (Simulink) pour programmer la carte Arduino, qui convertir le schéma à un programme en langage C++. [2-1]

II.2 La carte Arduino :

II.2.1 Définition de la carte Arduino :

Arduino est une plate-forme « open source » utilisée pour la construction de projets électroniques. Arduino se compose à la fois d'une carte de circuit imprimé physique (souvent appelée microcontrôleur) et d'un logiciel, ou IDE (Integrated Development Environment) qui s'exécute sur l'ordinateur, utilisé pour écrire et télécharger du code informatique sur la carte Arduino. [3-3]

II.2.2 Types de la carte Arduino : [3-5]

Il y a trois types de cartes :

II.2.2.a Officielles :

Qui sont fabriquées en Italie par le fabricant officiel : Smart Projects.

II.2.2.b Compatibles :

Qui ne sont pas fabriqués par Smart Projects, mais qui sont totalement compatibles avec les Arduino officielles.

II.2.2.c Les autres :

Fabriquées par diverse entreprise et commercialisées sous un nom différent (Freeduino, Seeduino, Femtoduino, …)

II.2.3 Les différentes cartes :

La carte de développement Arduino doit répondre à diverses exigences, et changeable. Certains utilisateurs peuvent vouloir prototyper et testez de nouveaux assemblages ou idées, la carte de développement Arduino Uno a suffisamment d'entrée et de sortie pour les grands projets raisonnables. En raison de sa taille, la carte devrait également pouvoir être utilisée à l'avenir dans des projets plus ambitieux. D'autres auront un grand nombre de ports sont nécessaires pour se connecter de nombreux capteurs ou actionneurs. Le troisième type

d'utilisateurs ne cherchera qu'à attirer l'attention et à utiliser des diodes clignotantes pour transmettre des signaux à leurs compagnons. Ces exemples ne sont qu'une partie des attentes du fabricant depuis leur carte de développement Arduino. La forme, la taille ou les possibilités de connexion jouent un rôle décisif dans le choix de la bonne carte. C'est pourquoi les développeurs Arduino ont développé une large sélection de cartes microcontrôleurs afin que chacun puisse trouver un modèle qui réponde à ses besoins. [3-6]

Arduino Uno

Arduino Leonardo

Arduino Mega ADK

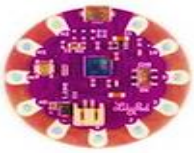
Arduino Due

Arduino Yún

Arduino Mega 2560

Arduino Tre

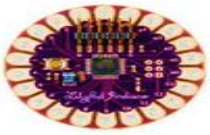
Arduino Micro



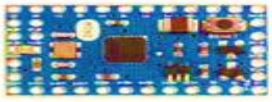
LilyPad Arduino USB

Arduino Ethernet

LilyPad Arduino SimpleSnap

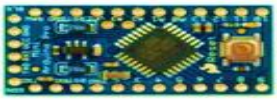


LilyPad Arduino



Arduino Mini

Arduino Nano



Arduino Pro Mini

LilyPad Arduino Simple

Arduino Pro

Arduino Fio

Figure II.1 : Les modèles d'Arduino.

II.2.4 Le but de l'utilité :

Le système Arduino nous permet de combiner les performances de programmation avec les performances électroniques. Plus précisément, nous programmerons le système électronique. Le plus grand avantage de la programmation d'équipements électroniques est qu'elle simplifie considérablement le schéma électronique, simplifiant ainsi les coûts de production et simplifiant la charge de travail de la conception de cartes électroniques. Il ne fait aucun doute que l'utilitaire n'était pas idéal lorsque' on a commencé, mais une fois qu'on est rentré dans le monde d'Arduino, on sera fasciné par les fonctions et applications incroyablement puissantes. [3-7]

II.2.5 Applications :

Le système Arduino nous permet de réaliser beaucoup de choses, qui ont des applications dans divers domaines. Le champ d'utilisation d'Arduino est énorme. On peut mentionner et donner quelques exemples qui sont : [3-7]

- Contrôler la vitesse d'un moteur.
- Contrôler les appareils domestiques.
- Fabriquer votre propre robot.
- Faire un jeu de lumière.
- Communiquer avec l'ordinateur.
- Télécommander un appareil mobile (modélisme)… etc.

II.3 La carte Arduino Uno :

II.3.1 Les avantages de la carte Arduino UNO :

Les avantages de la carte Arduino sont nombreux, on mentionne :

- Réalisation des programmes complexes.
- Connexion facile entre la carte et les composants externes.

- Aucun court-circuit possible.
- Stocker un programme et le faire fonctionner.
- Peut recevoir des informations analogiques et numériques.
- Des détrompeurs intégrés aux connecteurs.
- Équipée d'une carte motorshield (qui permet de gérer plus finement les moteurs)
- Utiliser le logiciel Scratch pour simuler le comportement d'un robot sur une version virtuelle.

II.3.2 La partie matérielle :

II.3.2.a Présentation de la carte Arduino UNO :

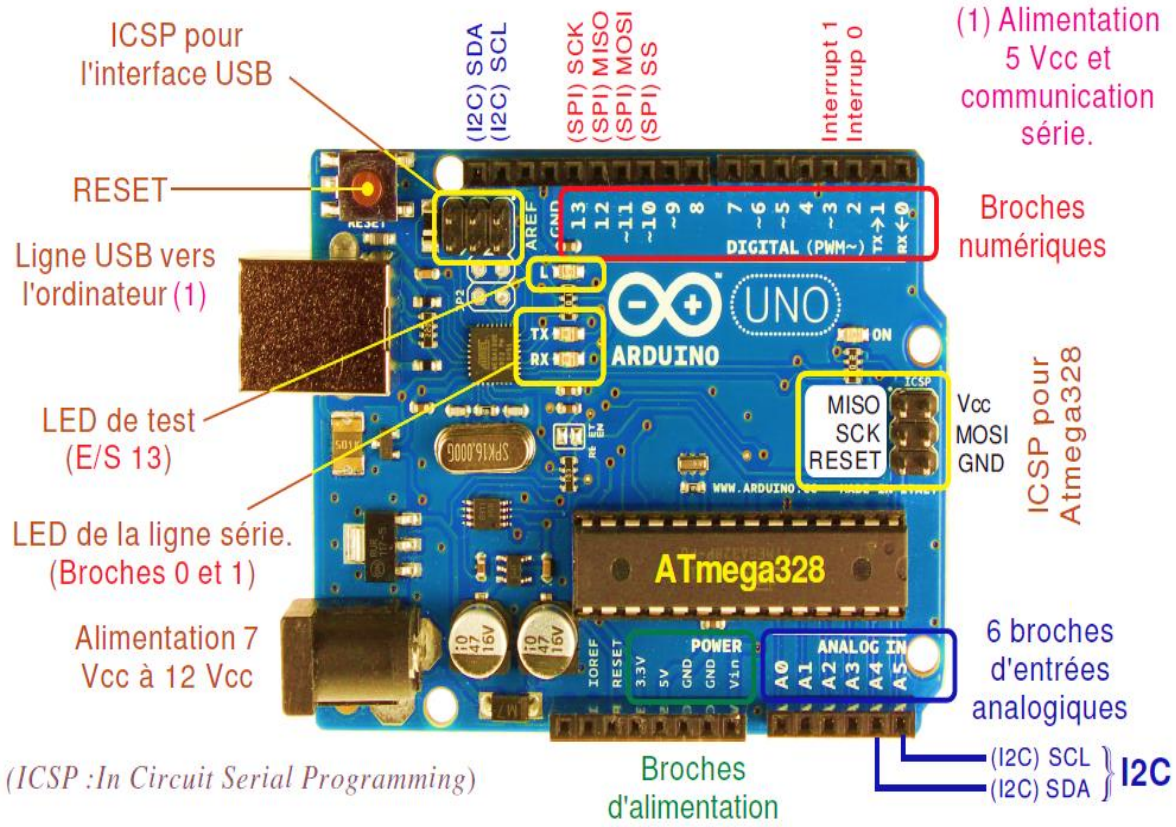


Figure II.2 : Schéma d'une carte Arduino Uno.

II.3.2.b Caractéristiques techniques de la carte Arduino UNO :

- Microcontrôleur : ATmega328.
- Fréquence horloge : 16 MHz.
- Tension d'alimentation interne : 5Vcc.
- Tension d'alimentation externe recommandée : 7Vcc à 12Vcc. (Limites : 6Vcc à 20Vcc)
- Courant max sur la sortie 3,3 V généré par le régulateur interne : 50 mA.
- Entrées/sorties binaires : 14 broches.
- Courant MAX par broches en sortie : 40 mA. (85 mA en court-circuit)
- Courant MAX cumulé par les broches en sorties : 200 mA. (Soit 14 mA en moyenne)
- Les E/S binaires 0 et 1 sont mobilisées par le dialogue sur la ligne série.
- S0 pour RX et S1 pour TX. Chaque broche est reliée à une LED via une résistance de 1 kw.
- Les E/S binaires 3, 5, 6, 9, 10, et 11 sont dédiées au mode PWM.
- L'E/S 13 est reliée sur la carte à la LED de test via une résistance de 1 kW.
- Entrées analogiques : 6. Le niveau logique maximal doit être de +5Vcc.
- Mémoire Flash 32 KB dont 0.5 KB utilisée par le Bootloader.
- Mémoire SRAM 2 KB. Mémoire EEPROM 1 KB.
- La carte s'interface au PC par l'intermédiaire de sa prise USB.
- La carte s'alimente par le jack d'alimentation. (Utilisation autonome)

II.3.2.c Le microcontrôleur :

C'est lui qui va recevoir le programme que nous aurons créé et qui va le stocker dans sa mémoire puis l'exécuter. Grâce à ce programme, il va savoir faire des choses, qui peuvent être : faire clignoter une LED, afficher des caractères sur un écran, envoyer des données à un ordinateur, mettre en route ou arrêter un moteur. [3-5]

Il existe deux modèles d'Arduino Uno : l'un avec un microcontrôleur de grande taille, et un autre avec un microcontrôleur dit SMD (SMD : Surface Mounted Device) soit composants montés en surface, en opposition aux composants qui traversent la carte électronique et qui sont

soudés du côté opposé. D'un point de vue de l'utilisation, il n'y a pas de différence entre les deux types de microcontrôleurs. [3-8]

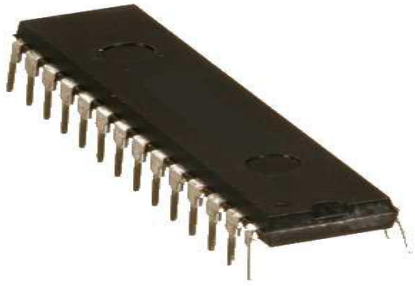


Figure II.3 : un microcontrôleur de la carte Arduino.

II.3.2.d L'Alimentation :

La carte Arduino Uno peut-être alimentée soit via la connexion USB (qui fournit 5V jusqu'à 50 mA) ou à l'aide d'une alimentation externe. La source d'alimentation est sélectionnée automatiquement par la carte.

La source d'alimentation externe (non-USB) peut-être un adaptateur secteur (fournissant généralement une tension de 3V à 12V à 50 mA) ou une batterie (ou batterie rechargeable). L'adaptateur secteur peut être connecté en insérant la prise positive de 2,1 mm au centre dans le connecteur Jack de la carte. Vous pouvez insérer les fils de la batterie ou de la batterie rechargeable dans les connecteurs à broches de la carte. Ces connecteurs à broches sont appelés Gnd du connecteur d'alimentation (masse ou 0 V) et Vin (tension d'entrée positive)

La carte peut fonctionner avec une alimentation externe de 6 à 20 volts. Cependant, si la tension d'alimentation de la carte est inférieure à 7 V, la tension d'alimentation de la broche 5 V peut-être inférieure à 5 V et la carte peut être instable. Si la tension utilisée dépasse 12 V, le régulateur de tension de la carte peut chauffer et endommager la carte. De plus, la plage idéale pour alimenter la carte Uno est de 7V à 12 V. [3-9]

II.3.2.e Les broches d'alimentation : [3-10] [3-9]

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.) Vous pouvez alimenter la carte 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 : L'alimentation régulée utilisée pour alimenter le microcontrôleur et les autres composants de la carte. Cela peut provenir soit du VIN via un régulateur embarqué, soit être fourni par USB ou une autre alimentation 5V régulée.

3V3 : Le circuit intégré FTDI de la carte (un circuit intégré qui adapte le signal entre le port USB de l'ordinateur et le port série de l'ATmega) peut fournir une alimentation de 3,3 V, cela est nécessaire pour certaines tensions au lieu de 5V.) Le courant maximum sur cette broche est de 50 mA.

GND : Broche de masse (0V)

II.3.2.f Entrées et sorties numériques : [3-10] [3-9]

Chacune des 14 broches numériques de la carte UNO (numérotées des 0 à 13) peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique. Ces broches fonctionnent en 5 V. Chaque broche peut fournir ou recevoir un courant maximal de 40 mA d'intensité et dispose d'une résistance interne de « rappel au plus » (pull-up) (déconnectée par défaut) de 20-50 KOhms.

De plus, certaines broches ont des fonctions spécialisées :

- **Communication série :** Broches 0 (RX) et 1 (TX). Utilisées pour recevoir (RX) et transmettre (TX) les données sérient de niveau TTL. Ces broches sont connectées aux broches correspondantes du circuit intégré ATmega8U2 programmé en convertisseur USB-vers-série de la carte, composant qui assure l'interface entre les niveaux TTL et le port USB de l'ordinateur.
- **Interruptions externes :** Broches 2 et 3. Ces broches peuvent être configurées pour déclencher une interruption sur une valeur basse, sur un front montant ou descendant, ou sur un changement de valeur.

- **Impulsion PWM (largeur d'impulsion modulée) :** Broches 3, 5, 6, 9, 10, et 11. Fournissent une impulsion PWM 8-bits.
- **SPI (interface série périphérique) :** Broches 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Ces broches supportent la communication SPI (Interface Série périphérique) disponible avec la librairie pour communication SPI. Les broches SPI sont également connectées sur le connecteur ICSP qui est mécaniquement compatible avec les cartes Méga.
- **I2C :** Broches 4 (SDA) et 5 (SCL). Supportent les communications de protocole I2C (ou interface TWI (Two Wire Interface - Interface « 2 fils »).
- **LED :** Broche 13. Il y a une LED incluse dans la carte connectée à la broche 13. Lorsque la broche est au niveau HAUT, la LED est allumée, lorsque la broche est au niveau BAS, la LED est éteinte.

II.3.2.g Broches analogiques :

La carte Arduino Uno possède 6 entrées analogiques connectées à un convertisseur analogique/numérique, qui renvoie un code numérique sur 10 bits avec une valeur comprise entre 0 et 1023. La pleine échelle est de 5 V, c'est-à-dire que la valeur numérique 0 correspond à 0 V et la valeur 1023 correspond à 5 V. Par conséquent, la taille du pas de quantification est de 5 V/1024, soit environ 5 mV.

Les entrées sur la carte sont marquées par les broches A0 à A5.

II.3.2.h Autres broches :

Il y a deux autres broches disponibles sur la carte :

AREF : La tension de référence de l'entrée analogique (si différente de 5V)

Reset : mettre cette broche au niveau BAS entraîne la réinitialisation (= le redémarrage) du microcontrôleur. Typiquement, cette broche est utilisée pour ajouter un bouton de réinitialisation sur le circuit qui bloque celui présent sur la carte.

II.3.3 Le signal PWM :

II.3.3.a Définition du signal PWM : [3-11]

Tout d'abord, PWM (Pulse Width Modulation), qui est traduit en français par MLI (Modulation de Largeur d'Impulsion), est un signal dans lequel le rapport cyclique change. De tels signaux sont généralement utilisés dans des applications où la valeur moyenne est variable (par exemple : commande moteur, alimentations réglables, etc.)

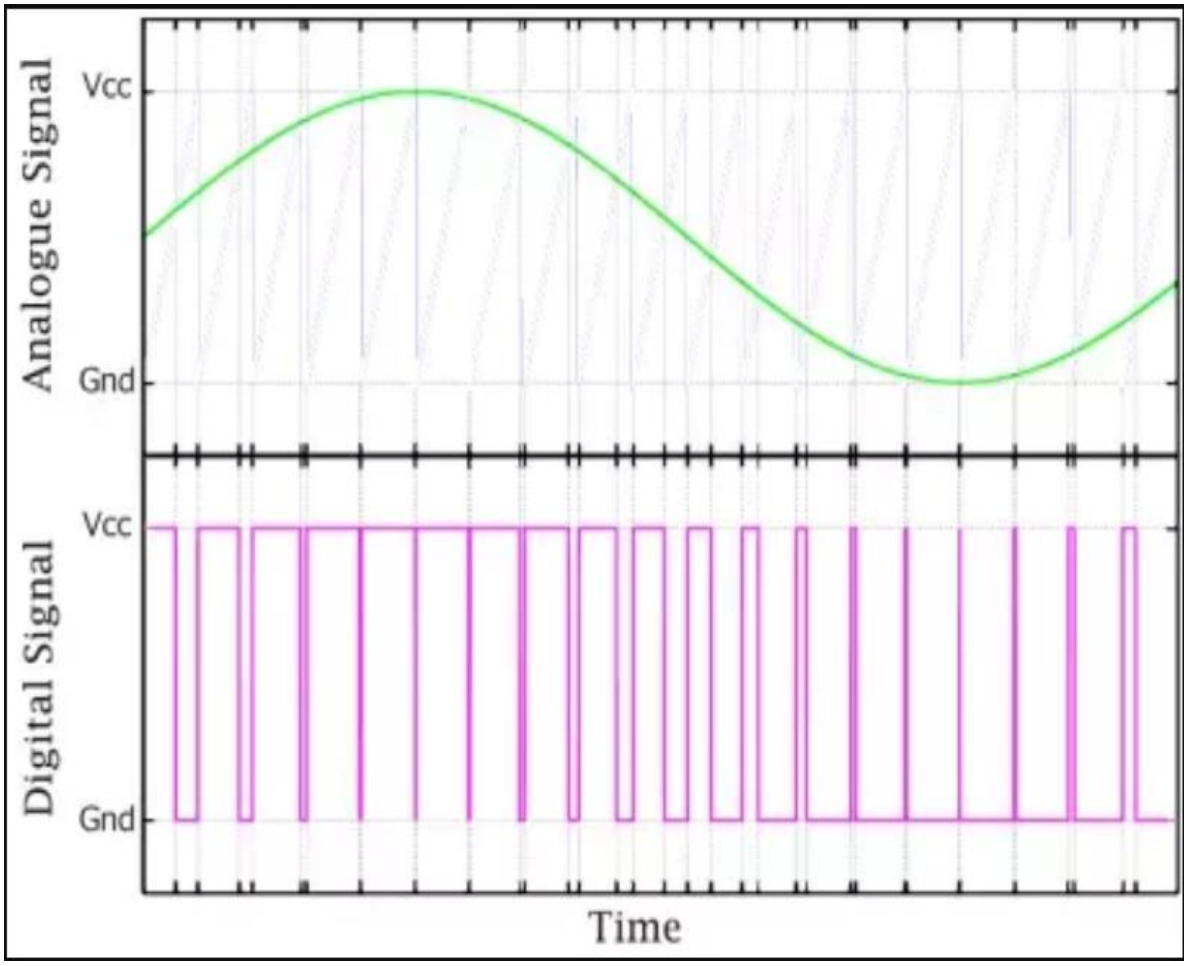


Figure II.4 : Processus de génération PWM.

Lorsque vous souhaitez modifier la luminosité de la diode (LED), la couleur de la LED RGB ou modifier la vitesse du moteur DC, il est nécessaire de générer un signal analogique.

Cependant, Arduino est équipé de broches analogiques comme entrées, mais ne peut pas être utilisé comme sorties.

Afin de modifier artificiellement la tension de sortie de l'Arduino, nous devrons utiliser certaines broches numériques sur la carte. Mais par définition, les broches numériques ne transmettent que des signaux binaires (0 ou 1), c'est-à-dire que le signal électrique de sortie ne peut avoir que deux valeurs : HIGH (niveau haut = 5V) ou LOW (niveau bas = 0V.)

Les broches digitales compatibles PWM vont permettre d'émettre un signal numérique disposant de 256 valeurs différentes (résolution de 8 bits soit $2^8 = 256$) et non plus seulement 2 valeurs (tout ou rien TOR ou binaire.)

II.3.3.b Comment utiliser le signal PWM pour changer de tension ?

Afin de générer une tension variable ou pseudo-analogique à la sortie des broches numériques de l'Arduino, il est nécessaire de changer l'état de sortie très rapidement. En fait, le fait qu'il passe très rapidement d'un état bas à un état haut entraînera une modification de la tension moyenne.

Or, c'est ce changement de la valeur moyenne du signal électrique qui va changer la luminosité de la LED ou changer la vitesse du moteur.

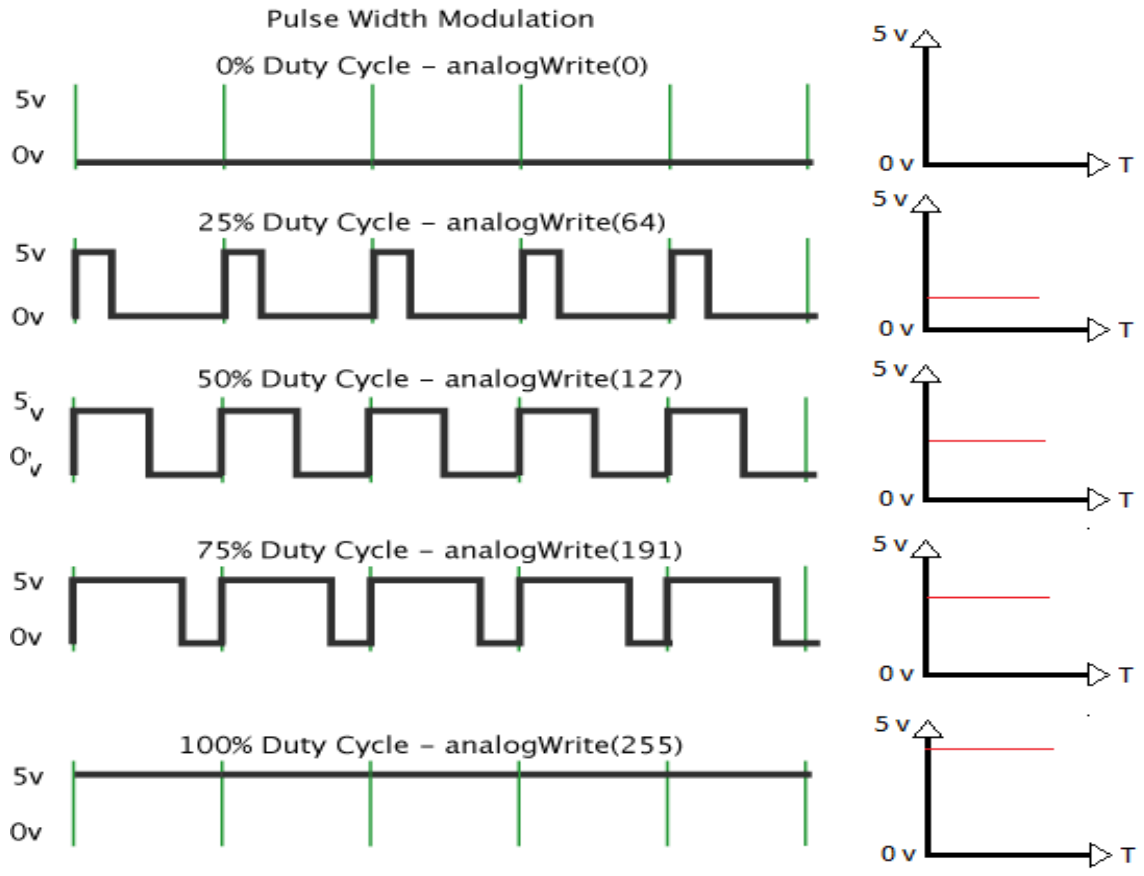


Figure II.5 : Modulation de largeur d'impulsion.

Le récepteur ne détectera plus simplement l'état BAS (0V) ou l'état HAUT (5V), mais une variété de tensions intermédiaires, qui variera de 0V à 5V, avec une résolution de 256 valeurs possibles. Ce signal pseudo-analogique est possible, car la fréquence de commutation entre les niveaux haut et bas est très élevée : 500 Hz ou 500 fois par seconde. L'état change très rapidement, de sorte que le récepteur n'y est pas sensible, comme notre œil avec le clignotement des tubes fluorescents avec le signal alternatif sinusoïdal 50 Hz.

Afin de déterminer la tension moyenne à appliquer au récepteur, le rapport cyclique doit être modifié, c'est-à-dire la durée de maintien du niveau haut du signal de sortie. Ainsi, le rapport cyclique pourra varier de 0 % (tension nulle 0V) jusqu'à 100 % (Tension égale à 5V).

Par contre, au niveau Arduino, il n'est pas nécessaire de fixer une valeur entre 0 et 100, mais une valeur numérique entre 0 et 255 (8 bits). En prenant la règle de trois ou un produit croix, le rapport cyclique ou la valeur de tension moyenne de la sortie Arduino peut être déterminé pour chaque entier (PWM) entre 0 et 255 :

Tableau III.1 : La valeur de tension moyenne de la sortie pour chaque entier (PWM)

PWM	Rapport cyclique	Tension
0	0 %	0V
33	10 %	0.5V
127	50 %	2.5V
230	90 %	4.5V
255	100 %	5V

II.4 Conclusion :

Dans ce chapitre, nous donnons un aperçu général de la carte Arduino, puis nous nous concentrons sur la carte Arduino UNO, et démontrons ses avantages pour prouver la rationalité de ce choix. De plus, nous avons également introduit l'une de ces deux parties essentielles (la partie matérielle), et enfin, nous avons présenté comment utiliser le signal PWM pour changer la tension de sortie entre 1 et 5 voltes.

III . Chapitre 3: L’interface Arduino Matlab

III.1 Introduction :

L'interface de Matlab Simulink est un moyen très connu et très utilisable dans le domaine du prototypage informatique grâce à sa simplicité et sa réalisation moins coûteuse.

Dans ce chapitre, on va parler sur les étapes de cette interface et leurs détails, commençant par montrer les réglages de la carte Arduino et son équipage avec le Matlab Simulink, ainsi que les enjeux qui nous permettent de travailler confortablement avec ses outils.

III.2 Matlab Simulink :

III.2.1 Définition de Matlab : [4 -1]

MATLAB est une plate-forme de programmation conçue spécifiquement pour les ingénieurs et les scientifiques. Le cœur de MATLAB est le langage MATLAB, un langage matriciel permettant l'expression la plus naturelle des mathématiques computationnelles. En utilisant MATLAB, nous pouvons :

- Analyser les données
- Développer des algorithmes
- Créer des modèles et des applications

Le langage, les applications et les fonctions mathématiques intégrées vous permettent d'explorer rapidement plusieurs approches pour arriver à une solution. MATLAB nous permet de faire passer vos idées de la recherche à la production en les déployant vers des applications d'entreprise et des appareils embarqués, ainsi qu'en les intégrant à Simulink® et à la conception basée sur des modèles.

Des millions d'ingénieurs et de scientifiques de l'industrie et du monde universitaire utilisent MATLAB. Nous pouvons utiliser MATLAB pour une gamme d'applications, y compris l'apprentissage en profondeur et l'apprentissage automatique, le traitement du signal et les communications, le traitement d'images et vidéo, les systèmes de contrôle, les tests et mesures, la finance informatique et la biologie informatique.

III.2.2 Définition de Simulink : [4-2]

Simulink est un outil de simulation des systèmes dynamiques avec une interface spécialement développée à cet effet. Au sein de l'environnement de MATLAB, Simulink est une boîte à outils MATLAB qui diffère d'autres boîtes à outils, à la fois dans cette interface spéciale et dans la technique de programmation spéciale qui lui est associée.

Les systèmes dynamiques peuvent être simulés à l'aide de Simulink. Dans la grande majorité des cas, cela signifie des processus linéaires ou non linéaires dépendant du temps qui peuvent être décrits à l'aide d'équations différentielles ou (dans le cas de temps discrets) d'équations de différence. Une autre façon courante de décrire les systèmes dynamiques consiste à utiliser des schémas fonctionnels.

Il s'agit d'une tentative de comprendre le comportement du système au moyen d'une représentation graphique, qui consiste essentiellement en des représentations de composants individuels du système avec le flux de signal entre ces composants.

Nous notons qu'une utilisation bien fondée de Simulink nécessite une certaine connaissance de la technologie de contrôle et de la théorie des systèmes qui dépasse le cadre de cette définition.

III.3 L'interface entre Simulink et Arduino :

L'interfaçage d'Arduino avec Simulink permet de construire des projets Arduino en utilisant une programmation de haut niveau et des diagrammes de blocs. Le module de prise en charge MATLAB pour Arduino nous permet de communiquer via USB avec notre carte Uno et les périphériques connectés. Étant donné que MATLAB est un langage interprété de haut niveau, on peut voir immédiatement les résultats des instructions d'entrées sorties sans les compiler et aussi il inclut des fonctions mathématiques, d'ingénierie et de traçage intégrés que l'on va utiliser pour analyser et visualiser les données de notre système.

Il existe plusieurs possibilités d'interfacer la carte Arduino avec Matlab/Simulink, telles que :

- Programmation de la carte Arduino Uno comme une carte d'interface.
- Utilisation du package Arduino.

- Utilisation du package Arduino Target.

Le langage amélioré et la capacité de tracer facilement les données des capteurs nous attirent tant que des instrumentistes à l'utilisation du package Arduino sur Simulink. Le module de support Simulink pour Arduino nous permet de développer des algorithmes autonomes sur notre carte.

Après la création d'un modèle sous Simulink on pourra bien le simuler, ainsi régler les paramètres de l'algorithme et le télécharger pour une exécution indépendante sur le périphérique.

On va faire l'interface selon les étapes suivantes :

III.3.1 Obtenir le support Arduino sur Logiciel Matlab :

Pour obtenir le support Arduino, on doit télécharger les packages de la carte concernée dans la bibliothèque de Matlab sur les étapes suivantes :

- Cliquer sur « Add-ons » et choisir « Get Hardware Support Packages ».

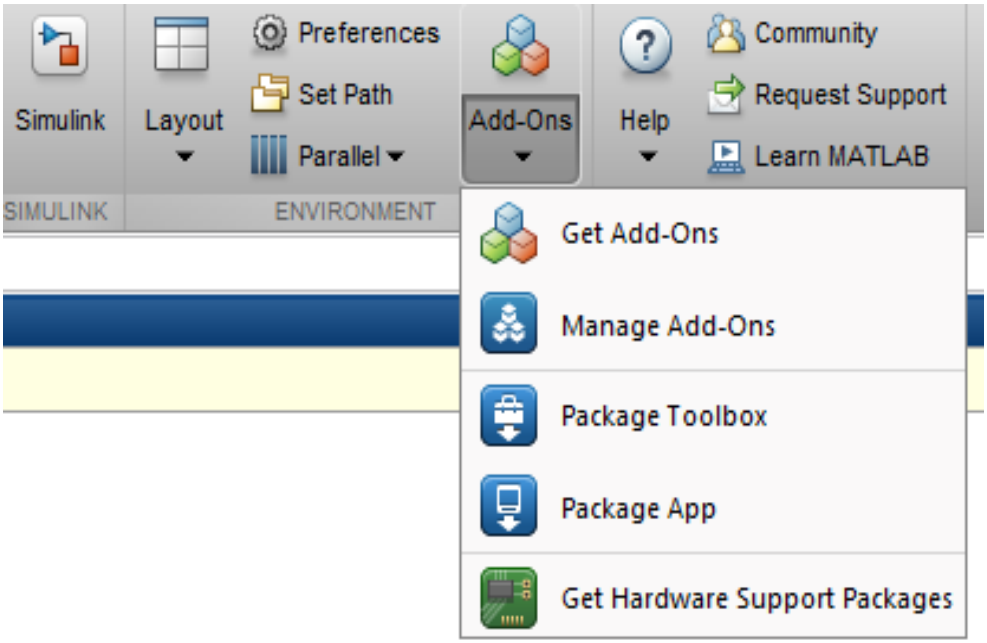


Figure III.1 : Fenêtre Add-Ons.

- Rechercher « Simulink Support Package for Arduino Hardware » sur la barre de recherche.

Figure III.2 : La barre de recherche du Support Package.

- Choisir le support compatible avec la version MATLAB, pour notre version MATLAB on a choisi ce support package :

Figure III.3 : Support Package compatible.

- Cliquer sur Add, ensuit cliquer sur Add to MATLAB :

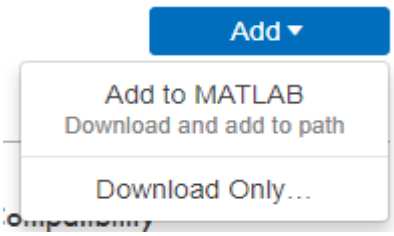


Figure III.4 : Fenêtre Add Support Package.

- Passer sur la bibliothèque de Simulink pour voir si l'installation est réussie.

Figure III.5 : Icône de la bibliothèque.

III.3.2 Configuration des paramètres : [4-3]

Cette étape consiste de faire des modifications pour que l'Arduino fonctionne correctement :

- Aller sur « Configuration des paramètres » sur Simulink.
- Choisir le type « fixed step » dans « solver options »
- Régler la taille de fixed step sur 0.1 dans « additional options » ou bien une autre valeur si on veut travailler avec un autre temps d'échantillonnage.

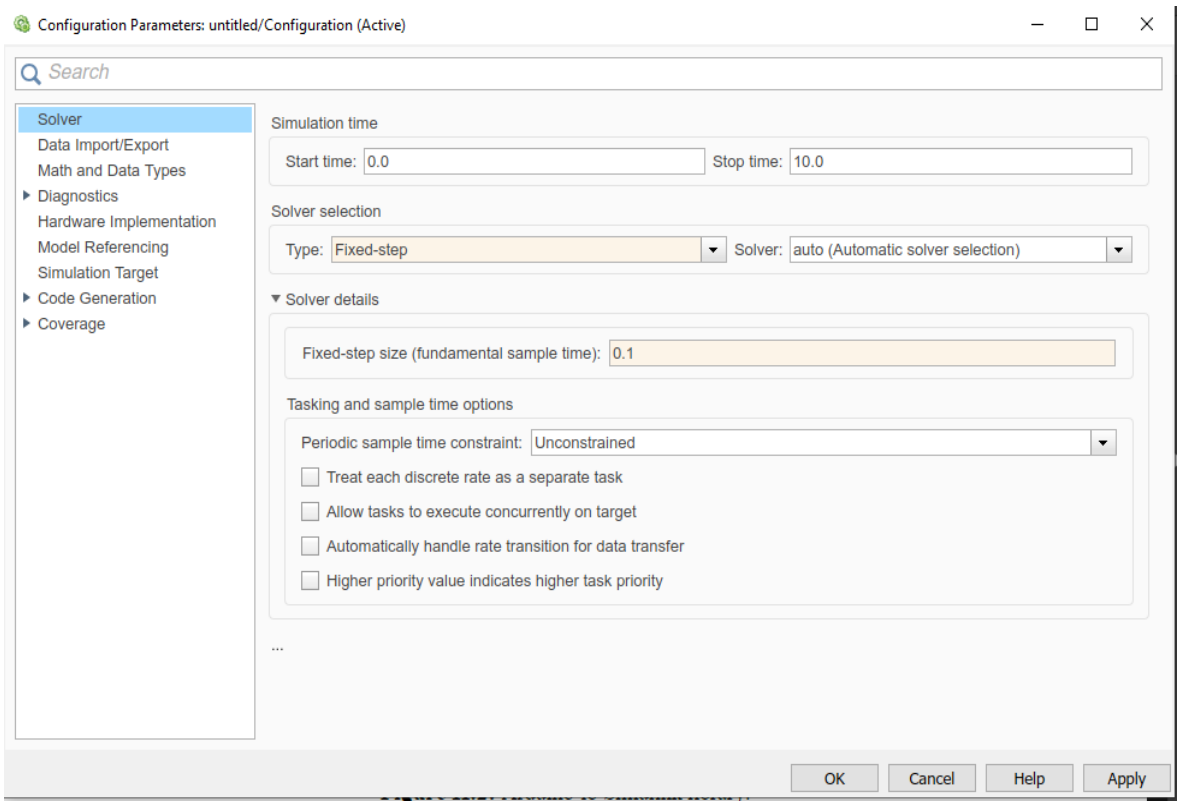


Figure III.6 : Fenêtre configuration des paramètres de Simulink.

- Aller sur Hardware Implementation et choisir le type de la carte Arduino.

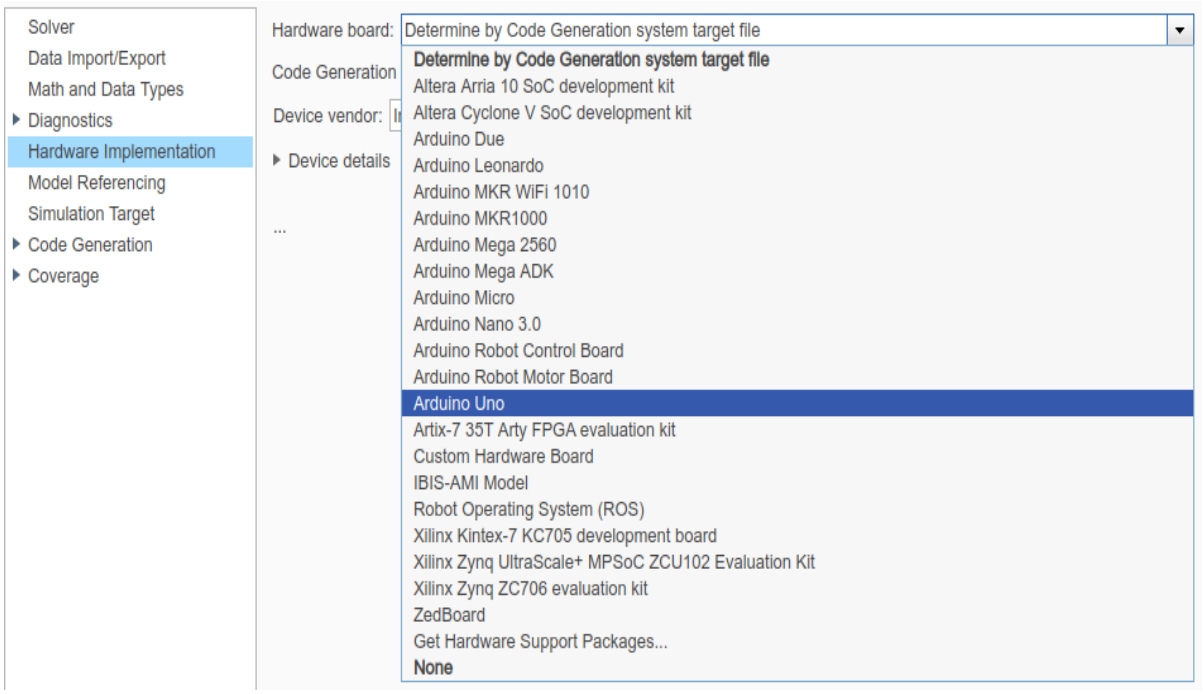


Figure III.7: Fenêtre Hardware board.

- Sélectionner Host-board connection et mettre « host COM port » sur manuel.
- Aller sur Configuration des périphériques de Windows et voir à quel port la carte Arduino est connectée avec le Windows (par exemple : COM 6.)

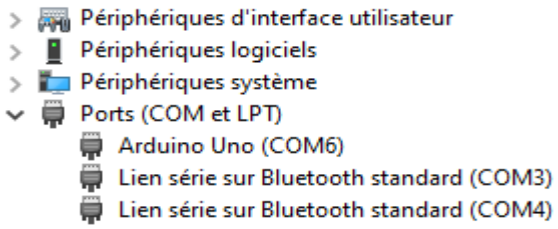


Figure III.8 : La carte Arduino connectée avec COM6.

- Retourner à la fenêtre précédente et choisir le port connecté avec Windows.

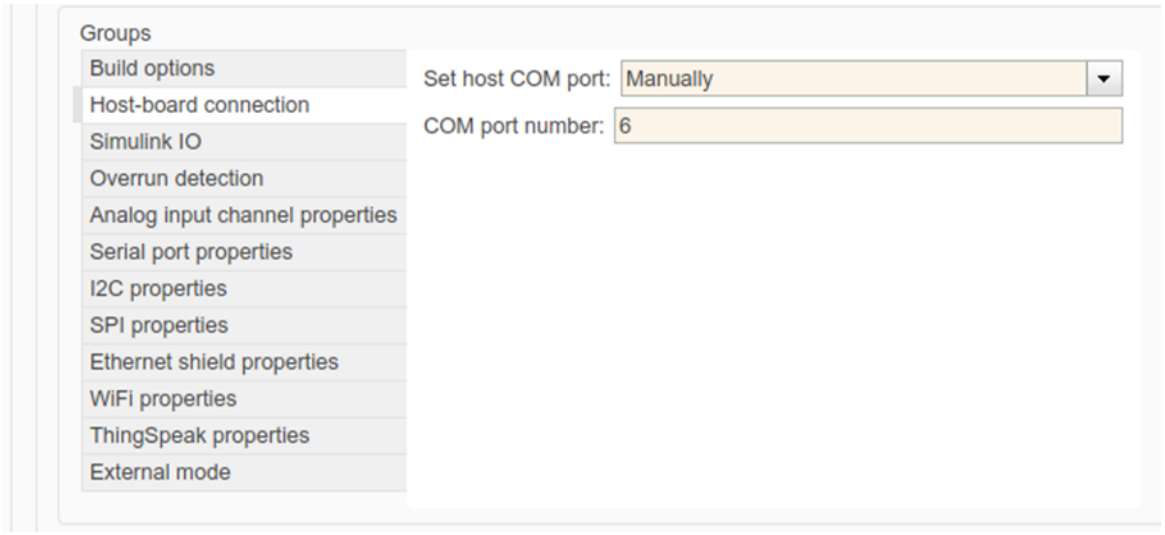


Figure III.9: Fenêtre Host-board connection.

- Régler la vitesse « sérial 0 baud rate » dans « serial port properties » sur la valeur 9600 et valider les modifications.

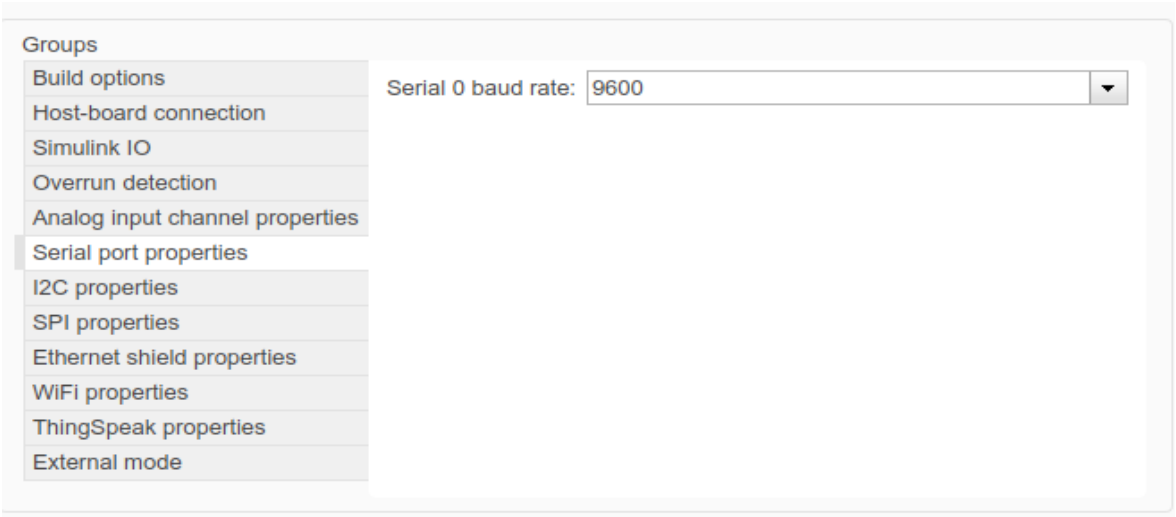


Figure III.10 : Fenêtre Serial port properties.

III.3.3 Faire des modifications pour la compatibilité des paramètres : [4-3]

Cette étape est très importante pour mettre à jour le temps et la vitesse de connexion entre le Software et le Hardware :

- Sélectionner le mode de Simulink sur « External » et son temps sur « inf ».

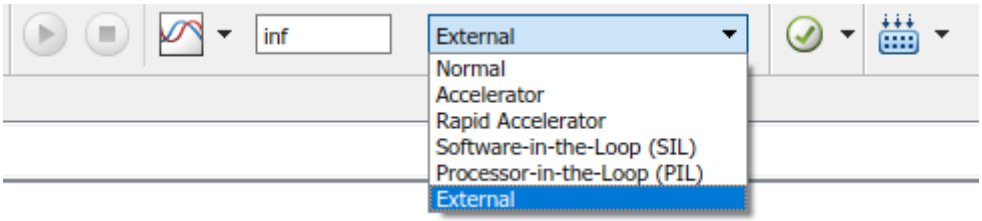


Figure III.11 : Les modes de connexion entre Simulink et Arduino.

- Faire sur « Command Window » la commande suivante :

  ''Codertarget.arduinobase.registry.setBaudRate (gcs, 9600).''

III.4 La bibliothèque Arduino sur Simulink :

Après l'installation de toutes les données et finir tous les réglages pour obtenir une compatibilité entre la carte Arduino et Simulink, nous constaterons que dans les bibliothèques MATLAB Simulink, l'une des bibliothèques sera la bibliothèque Arduino.

Cette bibliothèque contient 14 blocs dans le support « commun » que nous pourrons faire connecter avec Simulink et Arduino comme elle permet d'envoyer les informations du MATLAB à la carte Arduino connectée ainsi que de recevoir depuis la carte, c'est ce qu'on appelle (serial communication.)

Les blocs opérations sont illustrés dans la figure suivante :

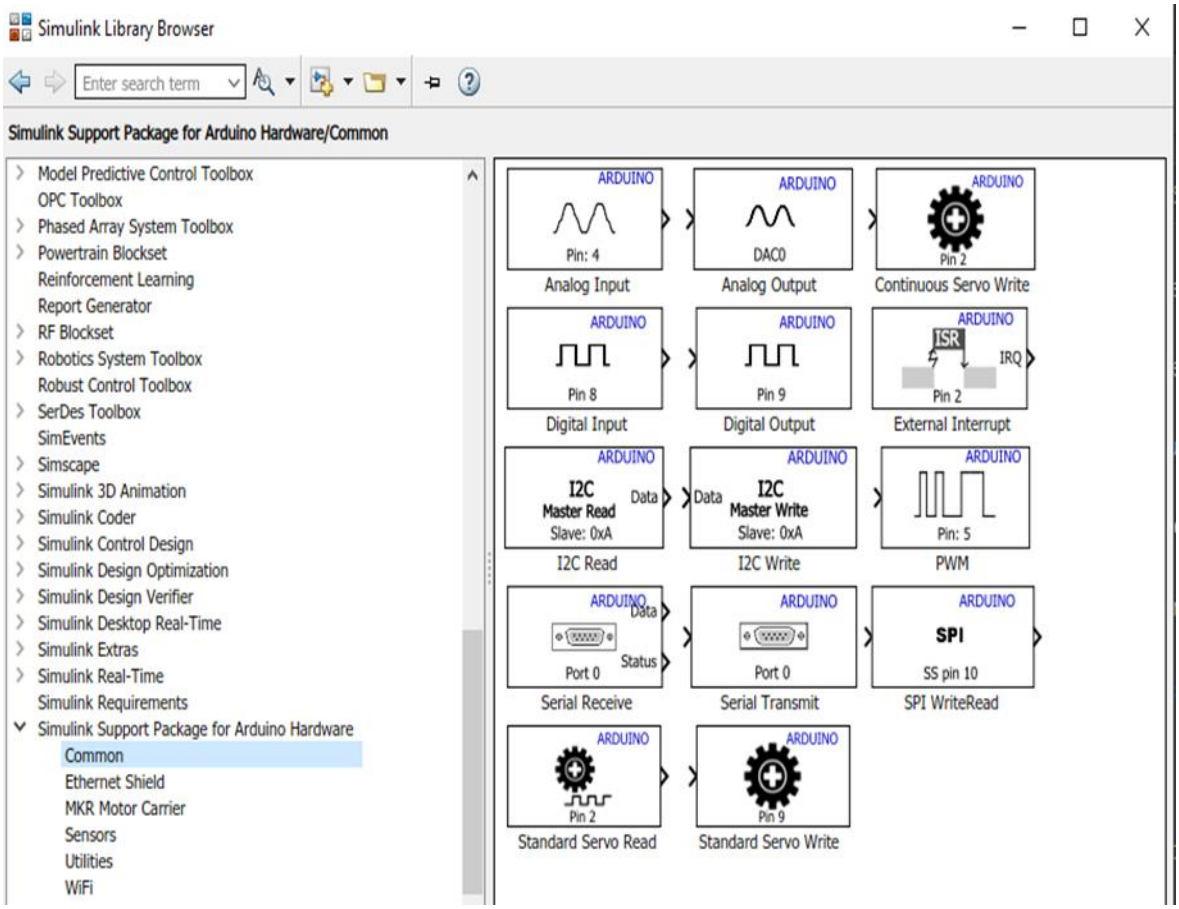


Figure III.12 : Les blocs d'Arduino les plus utilisés.

Depuis les blocs opérations ci-dessus, on va parler juste sur les blocs qu'on va utiliser dans notre projet de régulation et de commande, ces blocs sont les suivants : [4-4]

a) Arduino Analog Read :

Pour configurer à partir de quel pin [0, 1, 2, 3, 4, 5] on va acquérir les données du capteur.

b) Arduino Analog Write :

Pour configurer à partir de quel pin [3, 5, 6, 9, 10, 11] on va envoyer la commande en PWM vers l'actionneur.

c) Arduino Digital Read : pour recevoir les données de capteur du système sous forme des impulsions.

III.5 Les avantages et les enjeux de l'interface :

Un des avantages majeurs à tirer de l'utilisation du module Simulink est la possibilité de régler les paramètres en direct de notre ordinateur pendant que l'algorithme fonctionne sur le matériel. Un autre avantage très important c'est que l'interface nous permet de voir notre travail en simulation avant de faire la réalisation ; c'est-à-dire minimiser la possibilité de commettre des erreurs.

Enfin, l'interface nous donne l'avantage de voir les schémas et les graphes de notre projet ainsi que les entrées et les sorties du système afin de connaître tous les détails à propos de notre travail.

III.6 Conclusion :

Dans ce chapitre, nous apprenons que l'interface de la carte Arduino avec le Matlab Simulink nous permet d'avoir beaucoup des choix pour manipuler notre projet, mais nous devons choisir correctement les paramètres pour avoir la compatibilité entre le Software et le Hardware et compléter notre projet.

IV. Chapitre 4 : L’acquisition des données et l’identification du système

IV.1 Introduction :

L'acquisition des données et l'identification du système sont des moyens pratiques pour modéliser le système et obtenir sa réponse indicielle et sa fonction de transfert.

Pour atteindre ces objectifs, on doit d'abord savoir comment manipuler ces étapes et comment agir à notre système en mettant les configurations appropriées à fin d'avoir un meilleur résultat.

Dans ce chapitre, on va parler sur toutes les étapes qu'on doit faire pour la collecte des données et identifier notre système, et enfin pouvoir le modéliser et obtenir la fonction de transfert.

IV.2 Les matériaux utilisés et le branchement du système :

Avant d'entamer dans la partie pratique, on doit d'abord définir notre système et détailler ses composants qui sont les suivants :

– Une carte Arduino Uno.

– Une carte électronique L298N (Pont en H)

– Un ventilateur du processeur (CPU fan) avec 4 Pins.

– Des câbles.

– Une source de tension compatible pour notre système 12V-1A.

IV.2.1 Le ventilateur du processeur à 4 sorties :

Le ventilateur du processeur de l'ordinateur occupe une fonction essentielle :

Le refroidissement du CPU pour éviter que des pièces délicates de la fonte ou de surchauffe ou d'endommager pendant l'utilisation. [4-1]

L'importance des processeurs dans les ordinateurs modernes qui produisent plus de chaleur que leurs prédécesseurs. C'est pour cette raison que les ordinateurs sont désormais de puissants ventilateurs conçus exclusivement pour le refroidissement du CPU.

Le ventilateur de notre système (MT-CF1155) consiste de 4 sorties d'alimentation, contrôle et capteur, la figure.1 suivante illustre ce modèle :

Figure IV.1 : Modèle du ventilateur à 4 sorties.

Le ventilateur contient un pin d'alimentation (12V), un pin de masse (GND), un pin du contrôleur PWM et un pin de la sortie du capteur de vitesse qui génère des pulsations à chaque tour de ventilateur.

Le code couleur des sorties est expliqué dans le tableau suivant :

Tableau IV.1 : Tableau du fonctionnement des sorties du ventilateur.

La sortie	La couleur du câble	Le fonctionnement
1	Noire	La masse (GND 0V)
2	Rouge	L'alimentation 12V
3	Jaune	Capteur de vitesse (tachymètre)
4	Bleue	Contrôle PWM

Quand le ventilateur fonctionnant à vitesse maximale consomme un courant I= 250 mA et une tension de 12V, ça nous crée un problème de branchement avec la carte Arduino qui ne supporte ni sa tension ni son courant.

Alors, nous devons trouver un intermédiaire entre le ventilateur et la carte Arduino, cet intermédiaire est appelé la carte électronique L298N.

IV.2.2 La carte L298N :

L'interface de puissance L298N est un module comportant un double pont en H capable de piloter deux moteurs à courant continu ou un moteur pas à pas sous 36 V et consommant 2A au maximum, des diodes de roue libre pour protéger le L298N, un régulateur de tension 5V, des condensateurs de filtrage, des borniers à vis pour y brancher les moteurs et des broches pour y relier la carte Arduino. L298N possède trois entrées de commande par pont en H pour piloter le moteur.

La figure ci-dessous illustre les différents inputs-outputs de la carte L298N :

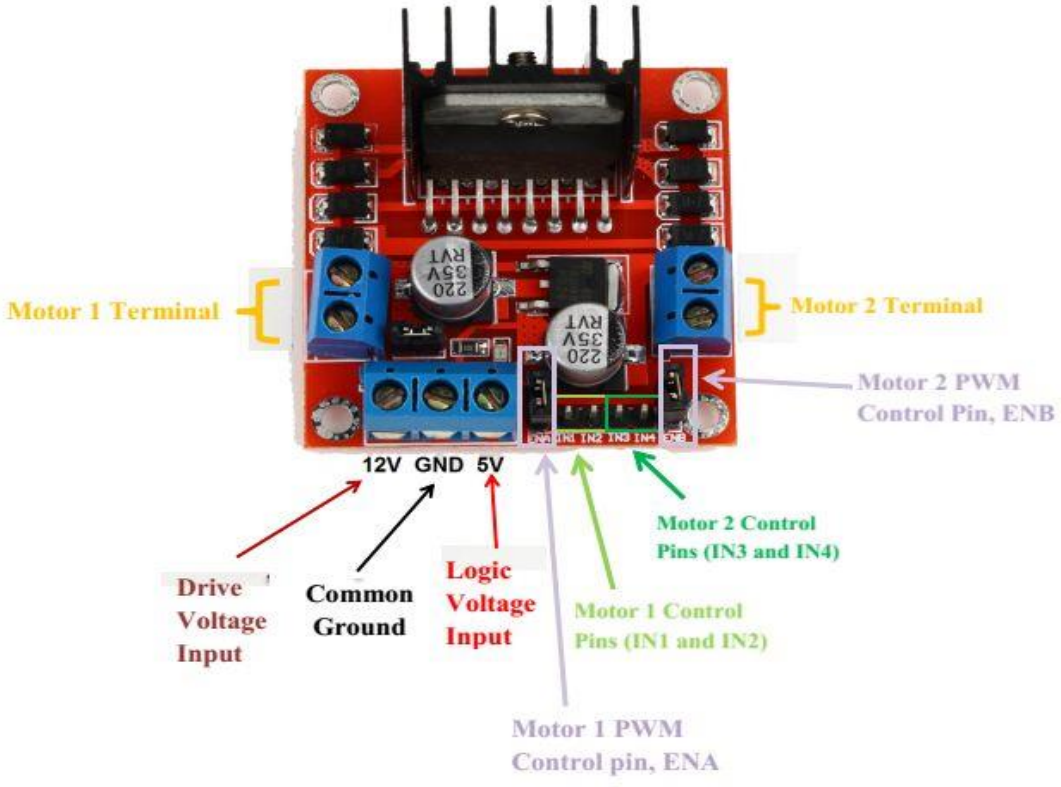


Figure IV.2 : Les entrées et sorties de la carte L298N.

IV.2.3 Le transformateur de tension :

Pour alimenter notre système, nous avons besoin d'une source de la tension à courant continu.

C'est pour ça qu'on a choisi de travailler avec un transformateur 220V-12V qui est disponible partout, car il alimente plusieurs des appareils électroniques comme les modems de l'internet, les démos de TV… etc.

Voici en figure ci-dessous le modèle du transformateur utilisé dans notre système :

Figure IV.3 : L'alimentation du système (transformateur 220V-12V.)

IV.2.4 Le branchement du système :

Maintenant, on fait le branchement de notre système sous les étapes suivantes :

1- Brancher l'alimentation du ventilateur (pin 12v et pin GND) avec la sortie 1 de la carte L298N (output1.)
2- Connecter la broche ENA1 du L298N avec une des sorties PWM de la carte Arduino (3, 5, 6, 9, 10, 11), dans notre branchement on choisit le pin 5.
3- Brancher le capteur de vitesse du ventilateur (tachymètre) avec une des entrées numériques de la carte Arduino, on choisit le pin 2.
4- Brancher la masse de la carte Arduino avec celle de la carte L298N.

5- Alimenter la carte L298N en 12V utilisant la source de tension. (Le + avec 12V et le – avec GND.)
6- Connecter les broches IN1 et IN2 dans 5V et GND de la carte Arduino (on peut inverser le branchement pour changer la direction de rotation.)
7- Brancher le câble USB de la carte Arduino avec le PC pour commencer la commande du système via Simulink.

Après ces étapes, on assure le bon branchement des câbles entre eux, et on obtient le montage montré dans la figure suivante :

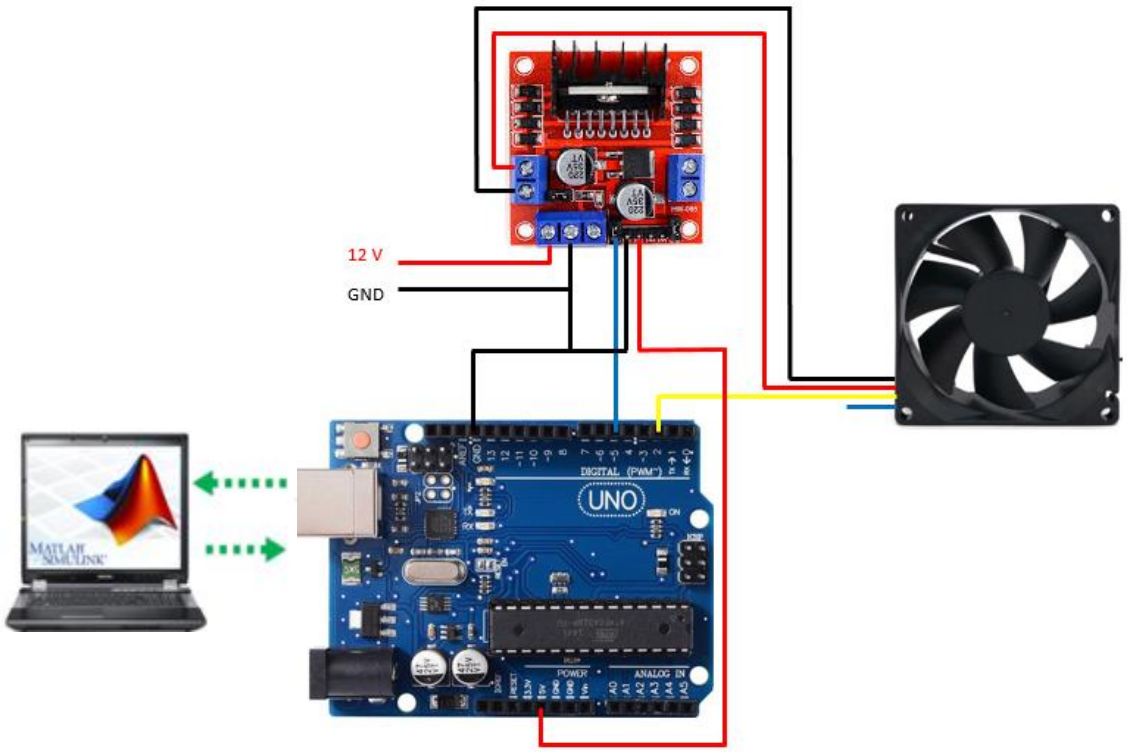


Figure IV.4 : Le montage du système.

Après le branchement des équipements, on commence la simulation avec Matlab/Simulink.

IV.3 L'acquisition des données :

Plusieurs méthodes sont utilisées pour la modélisation d'un système comme la détermination des équations physiques du système, l'étude de la réponse d'un système à une

entrée...etc. Dans notre cas on va identifier notre système en étudiant la réponse de notre système à échelon de tension comme entrée.

Le modèle Simulink permettant de réaliser l'acquisition de la réponse du système à un échelon de tension est le suivant :

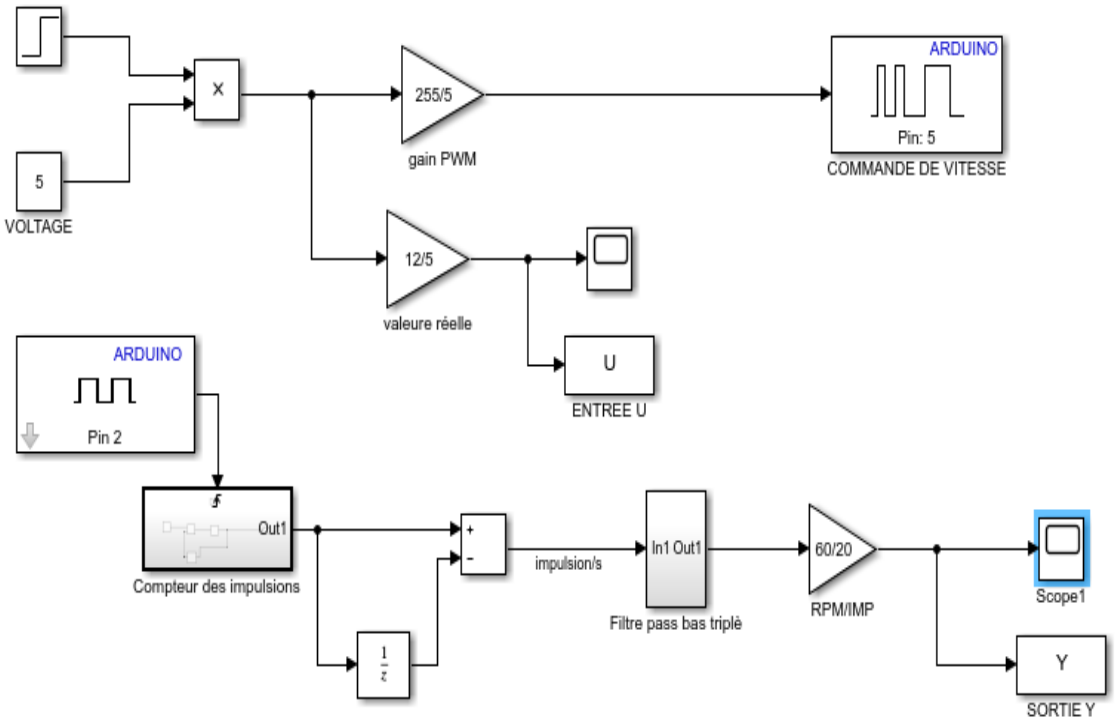


Figure IV.5 : Schéma blocs de l'acquisition de l'entrée du système.

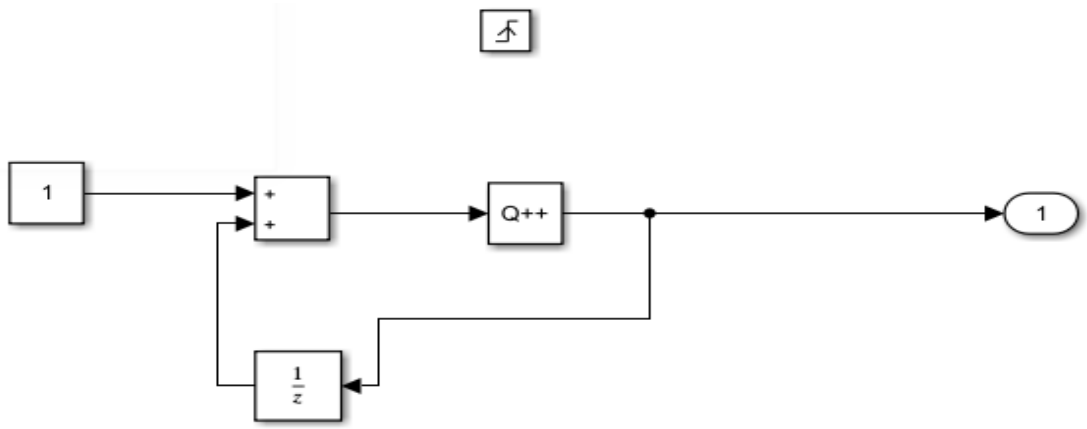


Figure IV.6 : Le sous-système du compteur des impulsions.

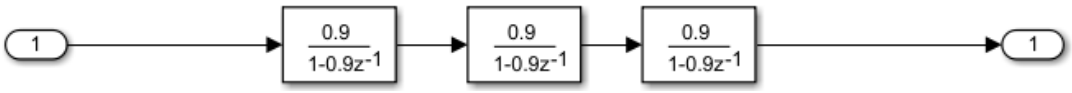


Figure IV.7 : Le sous-système du filtre passe-bas triplé.

Ce schéma consiste d'un signal du pin 5 PWM de la carte Arduino vers le système comme une entrée qui excite le ventilateur pour marcher selon le voltage donné, la tension du pin PWM diffère du 0 v à 5 v qui commande la vitesse de rotation du ventilateur du 0 V à 12 V.

Le schéma contient la sortie du système qui est fournie sur le pin 2 de la carte Arduino, ce pin nous donne des impulsions qui représentent le nombre des rotations du ventilateur (20 impulsions ↔ 1 rotation). Alors on doit convertir ces impulsions pour connaître combien nous avons des tours par minute (RPM) suivant chaque voltage.

Dans notre essai nous avons choisi le voltage maximal pour obtenir la valeur maximale des rotations par minute (2000 RPM +/- 10%.)

Nous avons remarqué que la sortie était très perturbée à cause des bruits de mesure et les bruits de l'environnement, même les erreurs du capteur ont une influence sur cette sortie. Alors on a implémenté un filtre passe-bas pour lisser le système et obtenir la meilleure réponse.

Les figures suivantes montrent l'entrée et la sortie du système obtenues durant la simulation :

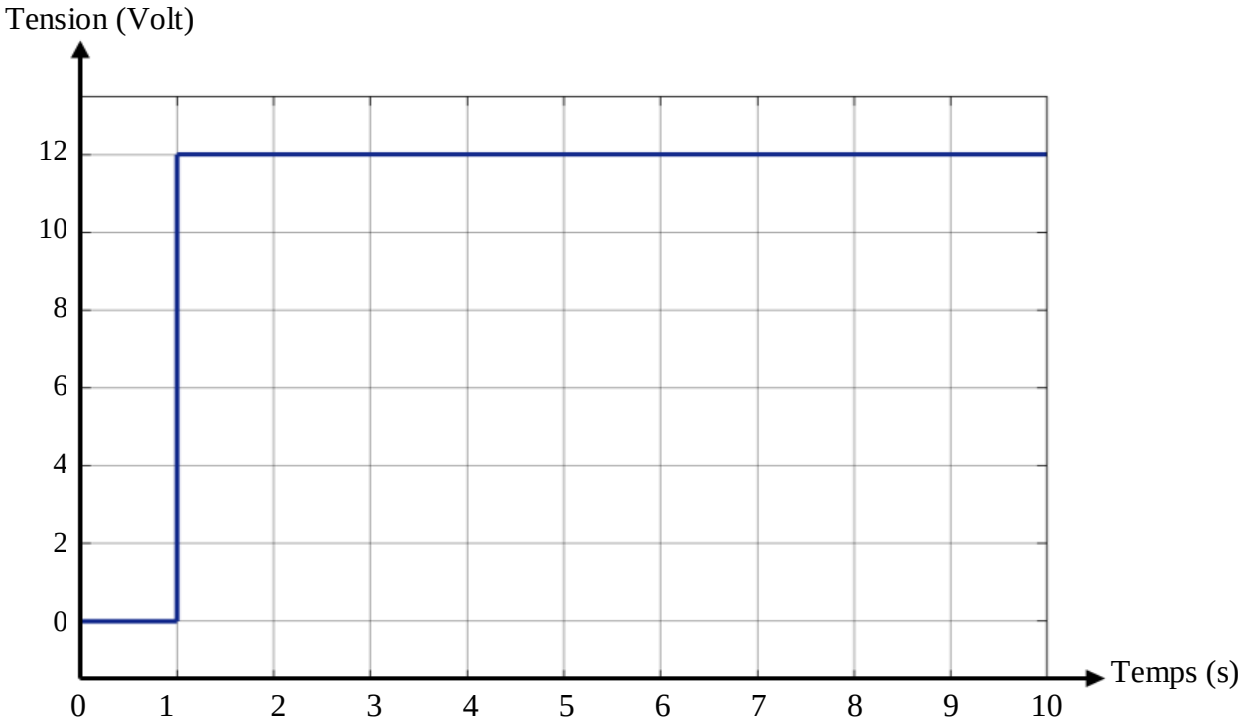


Figure IV.8 : Le graphe de l'entrée du système.

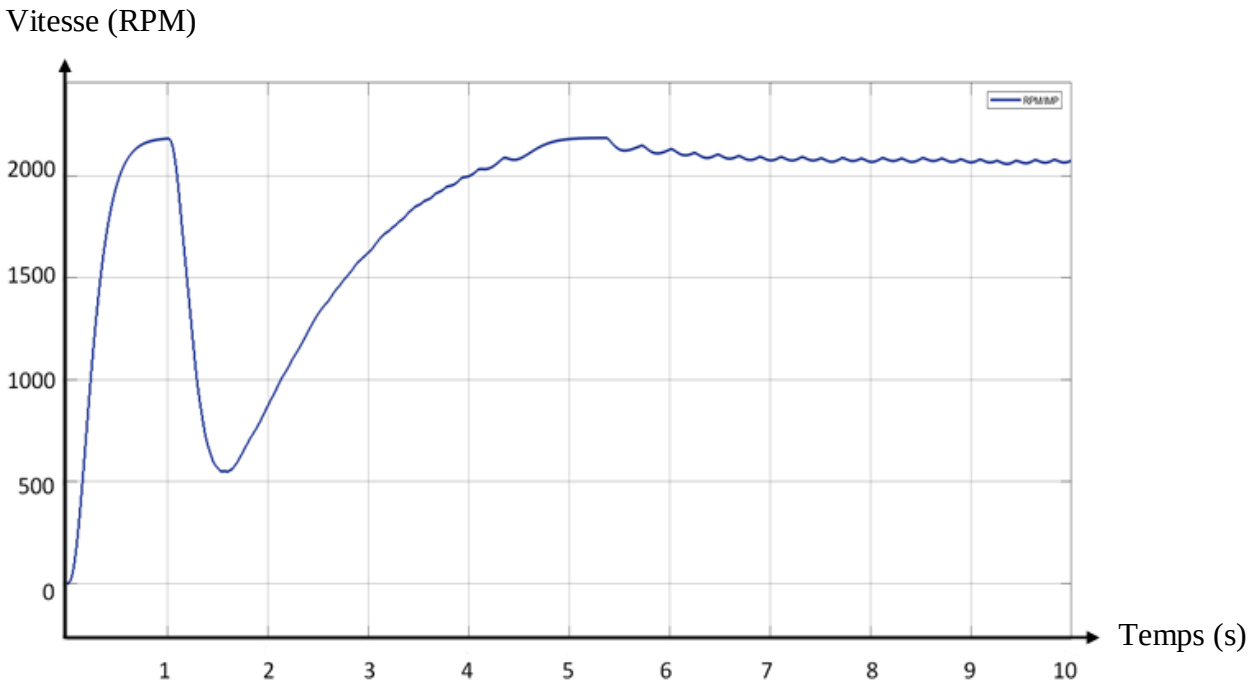


Figure IV.9 : Le graphe de la sortie du système.

Après la simulation, on obtient deux valeurs dans la fenêtre de Workspace de Matlab ; l'entrée du système ou bien la commande du voltage PWM converti en valeur réelle fournie au système qui est appelé U, et la sortie du système Y qui est sous forme de la vitesse de rotation (tours par minute.)

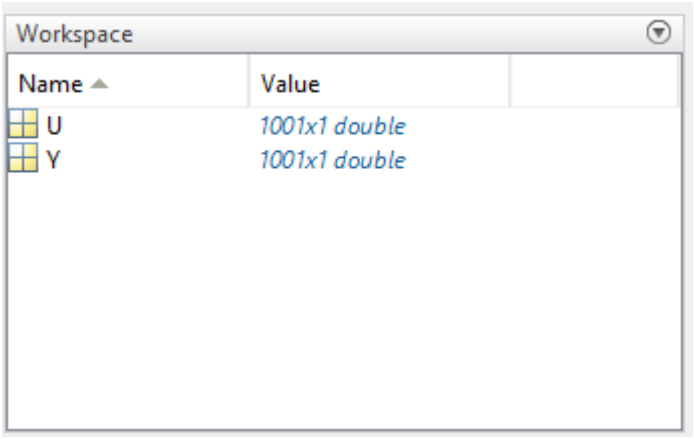


Figure IV.10 : Les valeurs de l'entrée et la sortie du système dans Workspace.

Ces valeurs nous permettent de faire l'identification du système en utilisant l'outil « System identification » qui est disponible sur Matlab.

IV.4 L'identification du système :

Le but de cette partie est de déterminer la fonction de transfert échantillonnée de notre système en boucle ouverte notée G(z). L'entrée du système est la tension U(z) en volts et la sortie est la vitesse V(z) en (tours/minute.)

Pour faire l'identification, nous suivons ces étapes :

1. Cliquer sur l'icône de « System identification » qui se trouve sur Matlab.

Figure IV.11 : L'icône de l'identification du système.

2. Une fenêtre s'apparaît pour choisir les options de l'identification, sélectionner « Time domain data » dans la partie de l'importation des données.

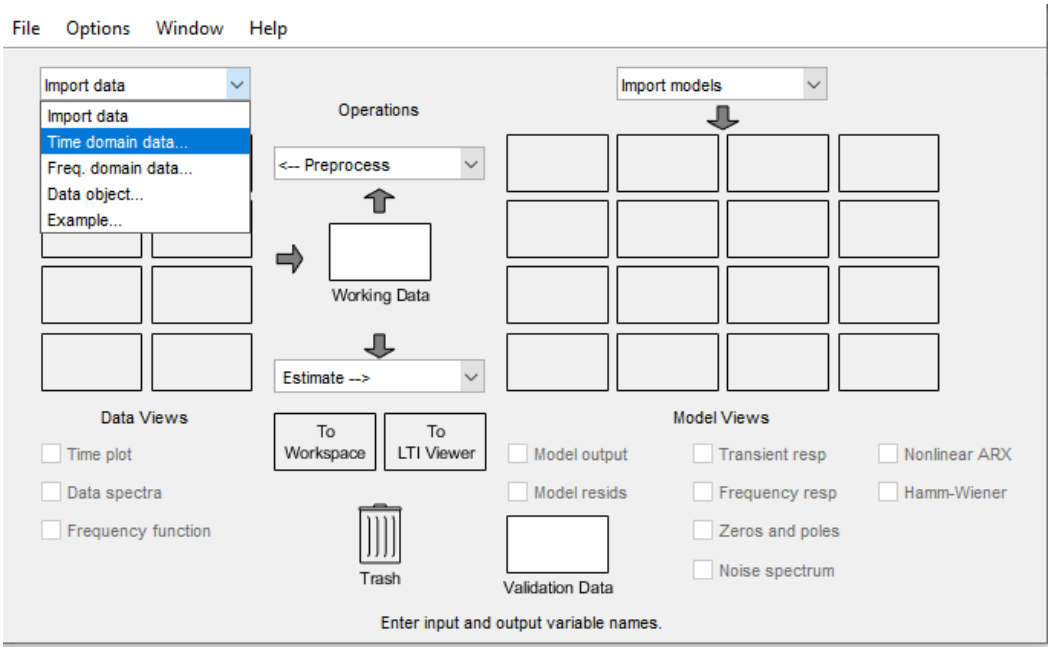


Figure IV.12 : Les options de l'identification du système.

3. Dans la fenêtre suivante, définir les noms de l'entrée et la sortie qui sont définis déjà dans le Workspace et donner la valeur d'échantillonnage utilisé dans la simulation, et cliquer sur « Import ».

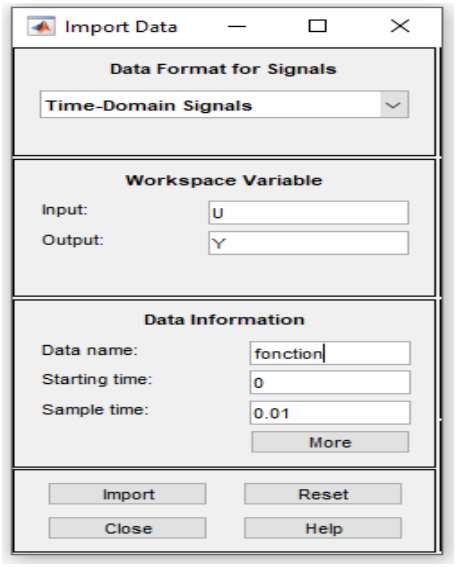


Figure IV.13 : La fenêtre de déclaration de l'entrée et la sortie.

4. Après l'importation, choisir le degré des pôles et le type de système et on clique sur « Estimate », dans notre cas c'est un système échantillonné de 2ème degré avec un temps d'échantillonnage Ts=0.01s.

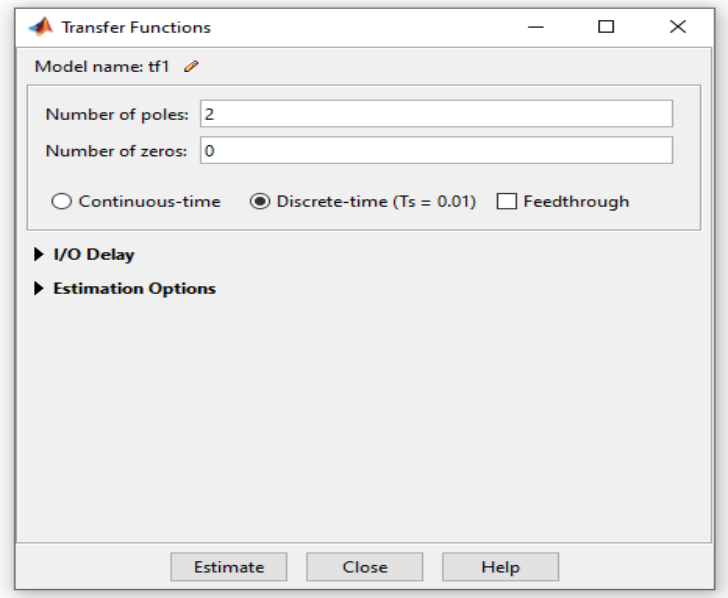


Figure IV.14 : nombre des pôles et le type de système.

IV.5 La fonction de transfert du système :

L'identification du système nous a donné le pouvoir de savoir la fonction de transfert en utilisant la méthode ARX. La fonction de transfert est :

$$G(Z) = \frac{0.02416}{0.9803\,Z^{-2} - 1.98\,Z^{-1} + 1} \qquad \textbf{(IV.1)}$$

La fonction G(Z) est une fonction de transfert discrète d'ordre 2 avec un temps d'échantillonnage Ts = 0,01 s.

La figure suivante montre la réponse de la fonction G(Z) :

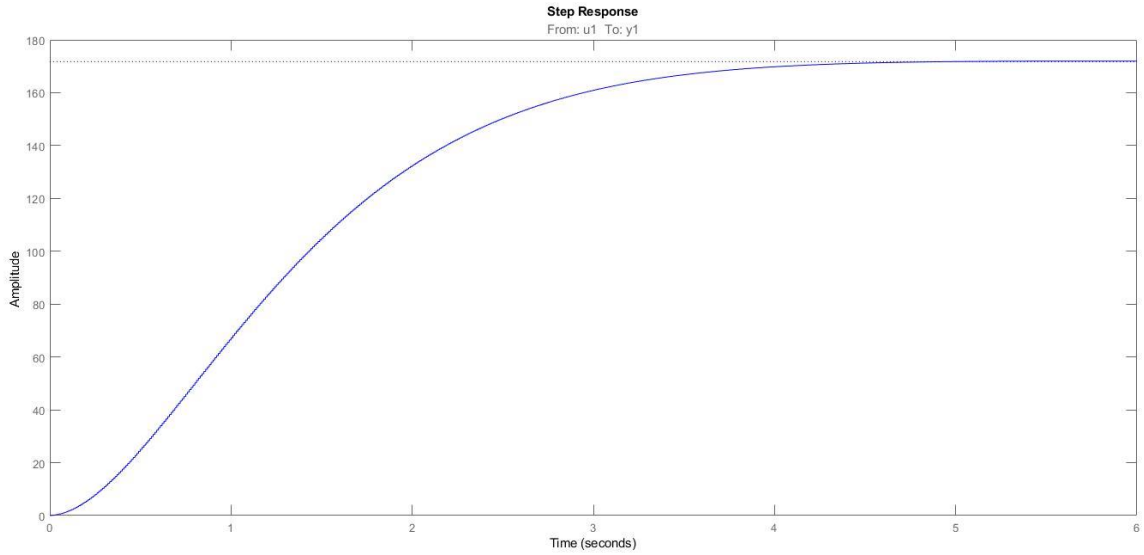


Figure IV.15 : La réponse indicielle de la fonction G(Z).

IV.6 Conclusion :

Dans ce chapitre, nous avons présenté, notre banc d'essai et ses différentes composantes, et nous avons montré comment se conduit une opération d'identification d'un procédé, et comment nous avons réalisé à l'aide de deux outils « Arduino et Matlab », l'identification de notre système. Cette méthode présente une solution idéale pour identifier n'importe quel procédé réel surtout pour des applications pédagogiques.

Dans le chapitre suivant de ce travail et après avoir obtenu le modèle de notre système, on cherche de le réguler et le commander par des méthodes de régulation, pour qu'il nous donne de bonnes performances.

V . Chapitre 5 : Implémentation de la commande PID sur Simulink

V.1 Introduction :

Après l'identification du système, nous allons entamer la régulation PID de ce système. Dans le 1er chapitre, nous avons parlé sur les régulateurs PID pour les fonctions de transferts continues mais on a trouvé que notre système est défini par une fonction de transfert discrète, alors on doit utiliser un régulateur PID numérique où les paramètres du régulateur diffèrent.

Dans ce chapitre nous allons parler sur ces paramètres, extraire les valeurs de ces paramètres et construire un schéma bloc pour compléter la régulation PID de ce système.

V.2 La régulation PID numérique :

La régulation PID d'un système asservi discret est implémentée par un régulateur numérique PID(Z) qui est constitué par ses paramètres Kp (gain proportionnel numérique), Ki (gain intégrateur numérique) et Kd (gain dérivateur numérique.)

Nous allons définir brièvement ces paramètres et donner leurs équations en parlant sur les actions P, PI et PID.

V.2.1 L'action PI dans un régulateur numérique :

La version de base du régulateur PI numérique résulte de la discrétisation du régulateur PI continu. Parmi les méthodes qui s'adaptent particulièrement bien aux problèmes de synthèse d'une correction numérique d'un asservissement continu. Nous supposerons donc que nous cherchons à asservir un système de fonction de transfert G(p) au moyen d'un correcteur C(z) Pour simplifier, nous supposerons que la boucle est à retour unitaire.

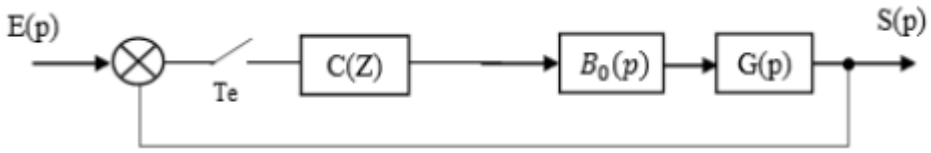


Figure V.1 : Asservissement continu corrigé numériquement.

La technique consiste à étudier cet asservissement en temps continu (Figure **V.2**) puis à rechercher le modèle numérique équivalent au correcteur continu C(p)

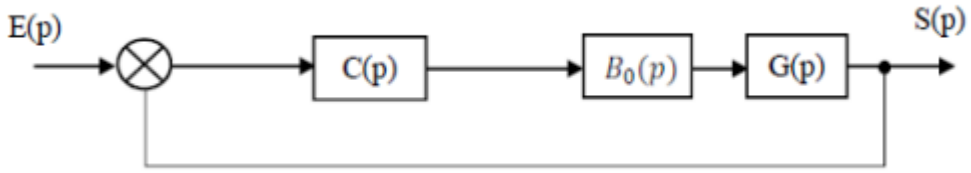


Figure V.2 : Modèle à temps continu de l'asservissement.

En théorie, il faut tenir compte de la présence du bloqueur dans l'étude en temps continu. Toutefois, une fréquence d'échantillonnage suffisamment grande peut nous permettre de le négliger.

Pour avoir la formule de régulateur PI numérique, il est nécessaire de passer par l'étape de discrétisation, l'équation **(V.1)** donne la formule de transformation du régulateur intégral de domaine de Laplace vers le domaine discret, par contre l'équation **(V.2)** donne la formule finale du régulateur discret (numérique) :

$$\frac{1}{T_i S} \rightarrow \frac{T_e}{T_i} \times \frac{1}{1-Z^{-1}} \qquad \textbf{(V.1)}$$

$$C(Z^{-1}) = K[1 + \frac{T_e}{T_i} \times \frac{1}{1-Z^{-1}}] \qquad \textbf{(V.2)}$$

D'où :

- K : gain proportionnel
- Ti=1/Ki : action intégrale
- Te : le temps d'échantillonnage

V.2.2 L'action PID dans un régulateur numérique :

Dans l'action PID, on doit ajouter un autre paramètre dérivatif pour assurer la rapidité du système et diminuer le dépassement. Tout comme le gain proportionnel et intégrateur, il faut discrétiser l'équation du dérivateur du domaine Laplace vers le domaine discret pour avoir le régulateur final PID numérique :

$$\frac{S}{T_D} \rightarrow \frac{T_e}{T_D} \times (1 - Z^{-1}) \qquad \textbf{(V.3)}$$

Alors, on obtient la formule finale du régulateur PID numérique suivante :

$$C(Z^{-1}) = K[1 + \frac{T_e}{T_i} \times \frac{1}{1-Z^{-1}} + \frac{T_e}{T_D} \times (1 - Z^{-1})] \qquad \textbf{(V.4)}$$

Vu la complexité de notre système, nous ne pouvons pas utiliser les méthodes théoriques de calcul des coefficients Kp, Ki et Kd. Alors nous devons trouver ces paramètres en utilisant une méthode pratique via l'application « PID tuner » qui se trouve dans Matlab.

V.3 La régulation PID pratique :

Le but de cette option est de connaître les paramètres du système qui sont : les gains du régulateur choisi, le dépassement du système, le temps de montée, le temps de réponse…etc. Pour atteindre ça, nous devons suivre les étapes suivantes :

1. Ouvrir l'application PID tuner.

Figure V.3 : L'icône de PID Tuner.

2. Importer la fonction de transfert obtenue par l'identification qui se trouve dans le Workspace.

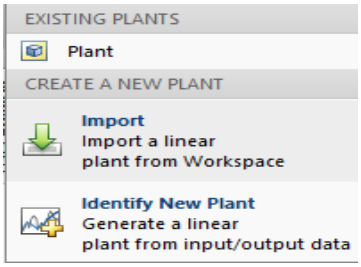


Figure V.4 : L'icône « importer la fonction de transfert ».

3. Choisir le type de régulateur soi PI ou PID et afficher les paramètres qui se trouve pour connaître les gains Kp, Ki et Kd.

Dans la partie suivante, nous montrons les graphes obtenus et nous expliquons les résultats trouvés :

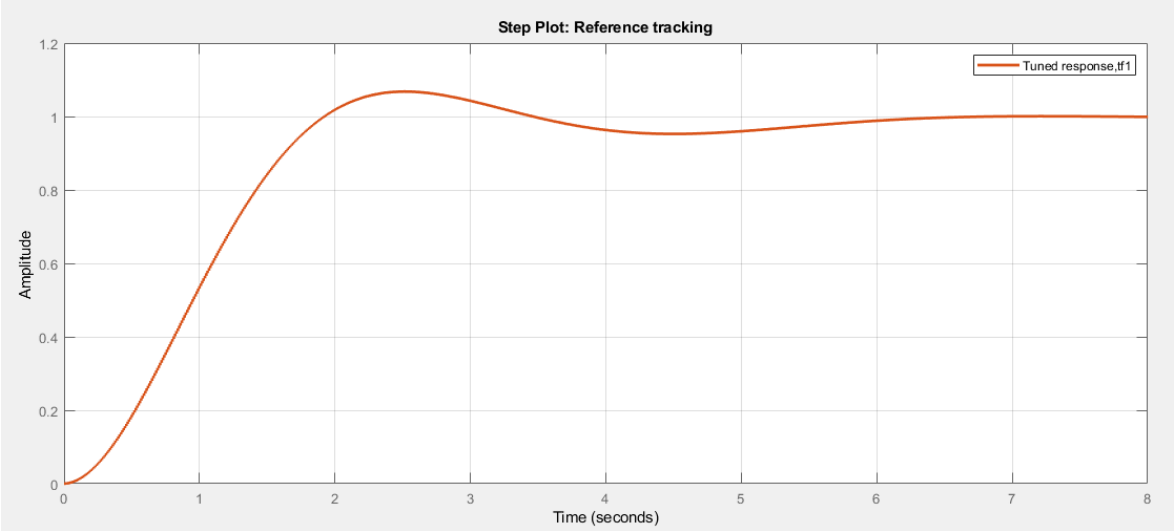


Figure V.5 : Le graphe de la réponse du système dans l'action PI.

Dans l'action PI, on obtient les résultats suivants :

Kp = 0.0075423

Ki = 0.005742

Tr = 1.805 s

D% = 7 %

Ts = 5,68 s (Temps de stabilisation du système)

Nous voyons que le dépassement est un peu élevé par rapport à ce que nous voulons atteindre (dépassement optimal 5 %), alors on va essayer d'ajouter une action D pour minimiser le dépassement et le temps de réponse, et enfin avoir un temps de stabilisation du système très bas.

Remarque : Il y a des cas où l'action D crée des perturbations et des nuisances dans le système, alors elle doit être filtrée, à cause de la fréquence d'échantillonnage élevée de notre système, nous devons le filtrer alors nous allons choisir le régulateur PIDF.

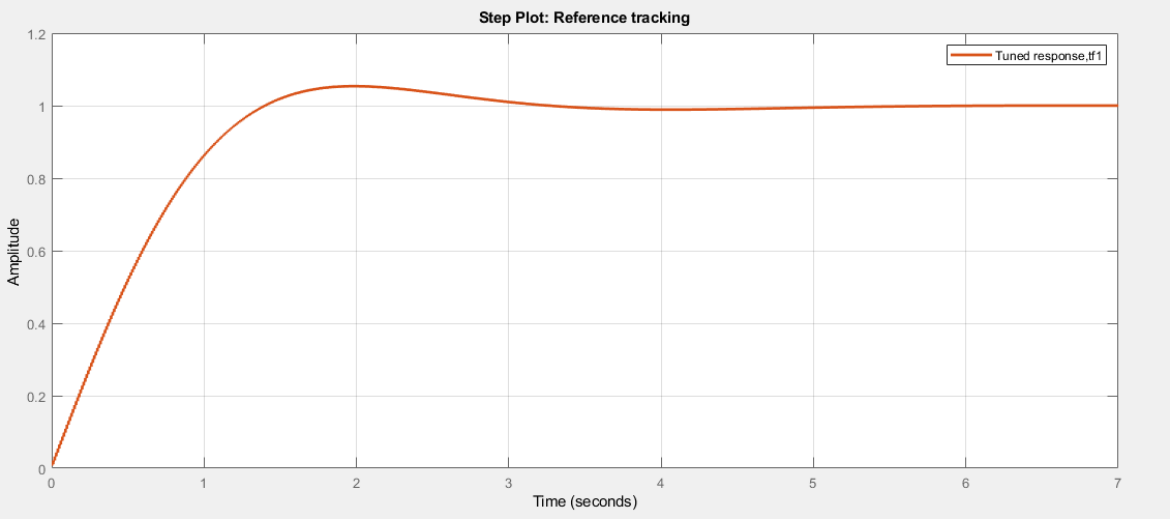


Figure V.6 : Le graphe de la réponse du système dans l'action PID.

Dans l'action PID filtrée, on obtient les résultats suivants :

Kp = 0.01443

Ki = 0.011577

Kd = 0.0044197

Tf = 0.010324s (temps de filtrage)

Tr = 1,217s

D% = 5%

Ts = 2.8s

Nous voyons que le dépassement est 5 % et le temps de réponse diminue, mais le plus important c'est la diminution de la valeur du temps d'établissement du système où nous voyons qu'elle est égale à 2,8 s.

Alors la régulation PID nous a fourni la stabilité du système et la vitesse de réponse ainsi que le dépassement optimal.

Maintenant, nous allons implémenter les paramètres du régulateur dans notre modèle Simulink pour finir la régulation PID de notre système.

V.4 Implémentation du régulateur PID sous Simulink :

Le but de cette partie est d'appliquer la régulation sur le système réel en utilisant la carte Arduino et le Matlab/Simulink.

La figure suivante montre le schéma synoptique de l'implémentation de régulateur :

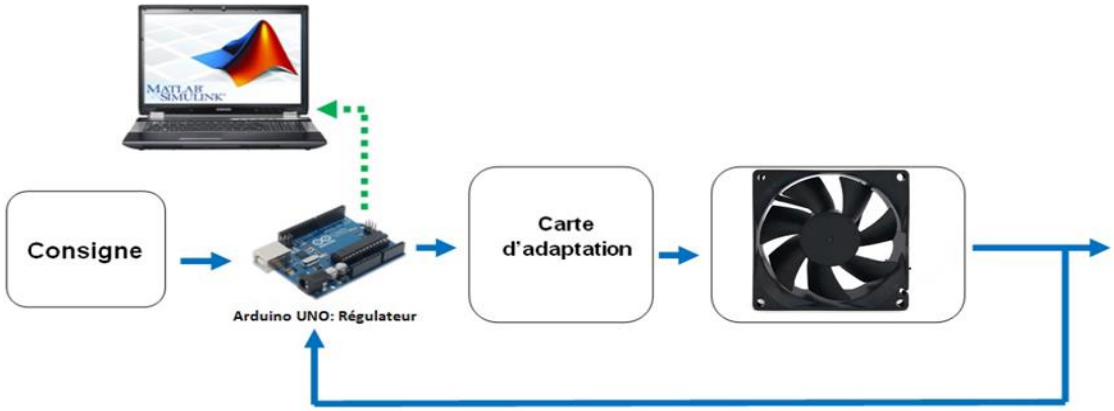


Figure V.7 : Schéma synoptique de l'asservissement à implémenter.

V.4.1 La régulation PI :

La boucle d'asservissement pour l'action PI à implémenter sur Simulink se traduit par le schéma suivant :

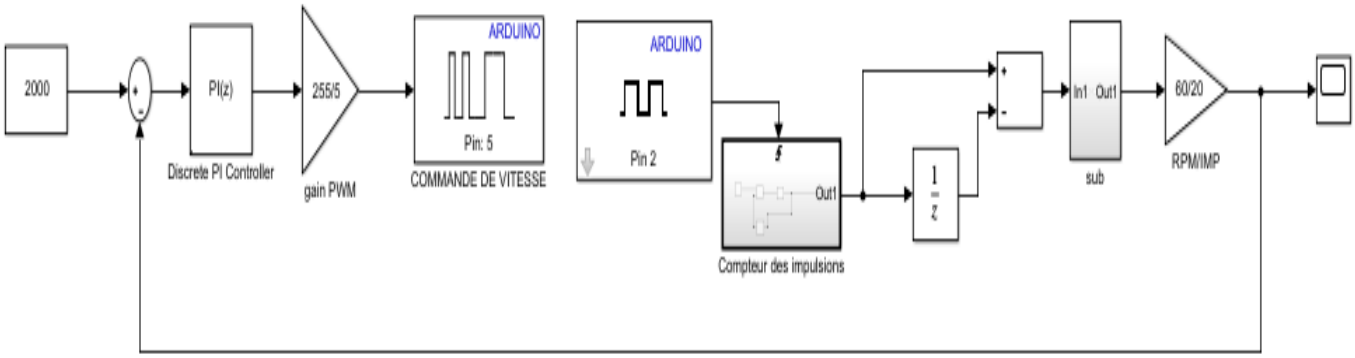


Figure V.8 : Modèle Simulink d'asservissement de vitesse du moteur par un régulateur PI.

L'asservissement de notre procédé est assuré par le schéma Simulink ci-dessus qui regroupe la consigne, le comparateur, le correcteur PI(z), l'acquisition de vitesse et l'envoie de la commande PWM.

Nous modifions les valeurs des gains proportionnel et intégrateur et nous commençons la simulation, la sortie de système est illustrée dans la figure suivante :

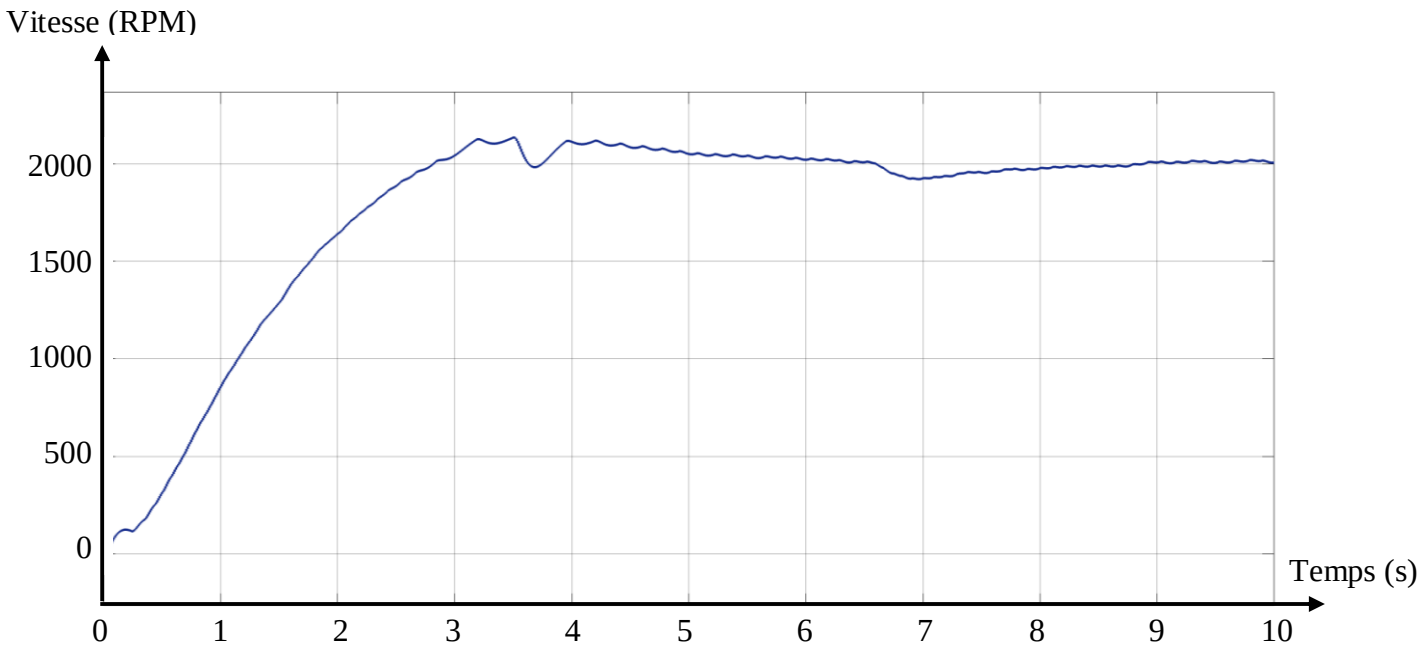


Figure V.9 : La sortie du système après l'application de l'action PI.

V.4.2 La régulation PID :

La boucle d'asservissement pour l'action PID à implémenter sur Simulink se traduit par le schéma suivant :

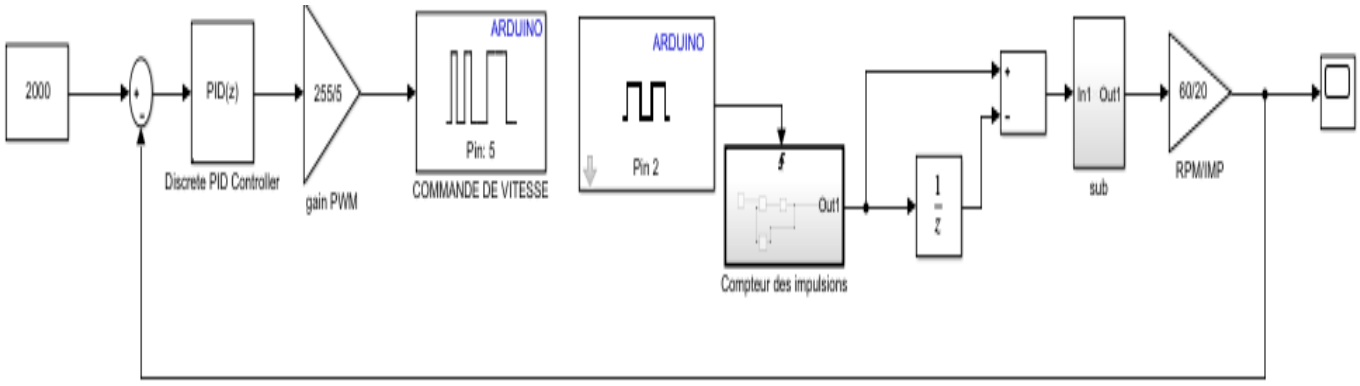


Figure V.10 : Modèle Simulink d'asservissement de vitesse du moteur par un régulateur PID.

L'asservissement de notre procédé est assuré par le schéma Simulink ci-dessous qui regroupe la consigne, le comparateur, le correcteur PID(z), l'acquisition de vitesse et l'envoie de la commande PWM.

Nous modifions les valeurs des gains proportionnel, intégrateur et dérivateur et nous commençons la simulation, la sortie de système est illustré dans la figure suivante :

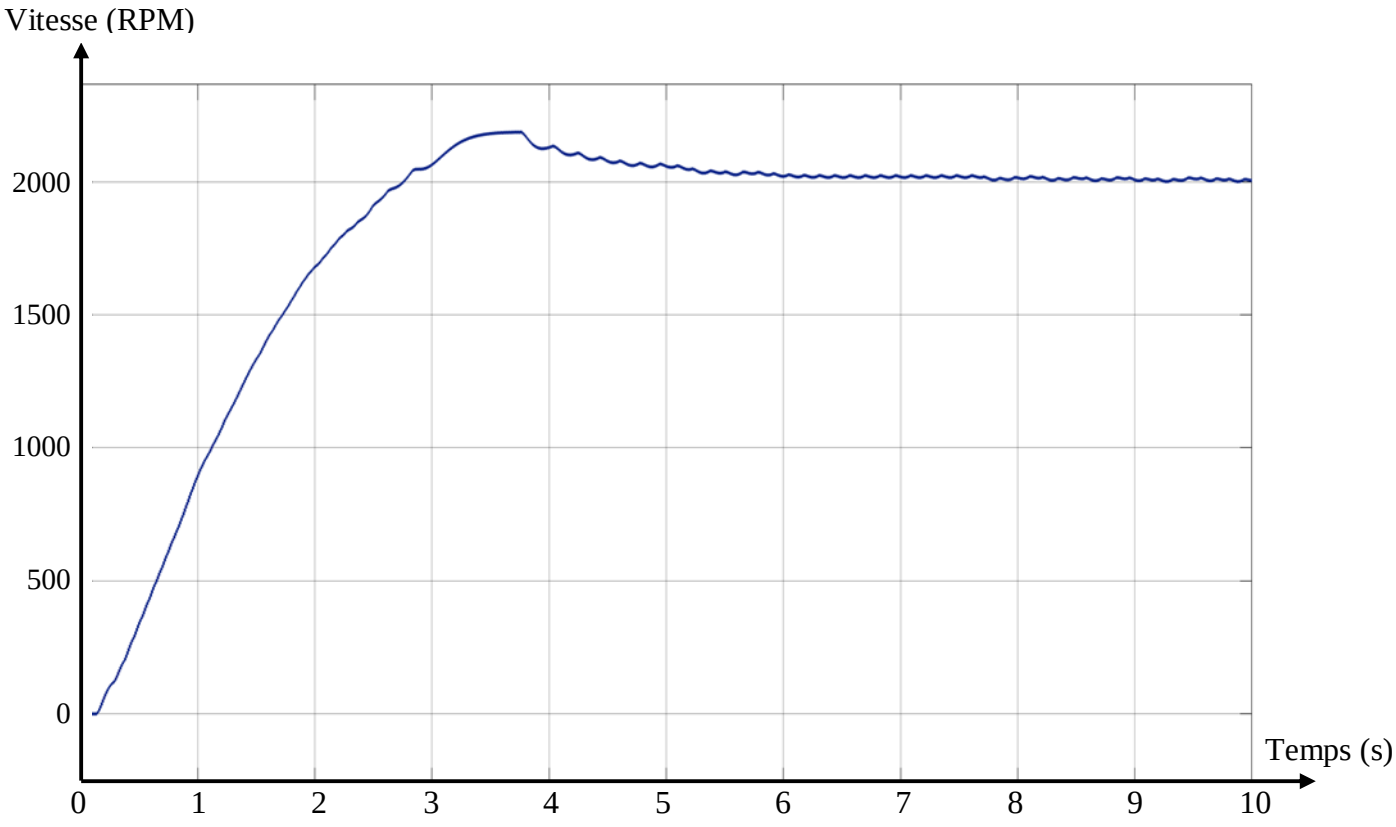


Figure V.11 : La sortie du système après l'application de l'action PID.

Nous appliquons une perturbation dans le système et nous voyons si la vitesse de réponse va revenir à la vitesse de consigne.

La figure ci-dessous montre la sortie du système avant et après la perturbation :

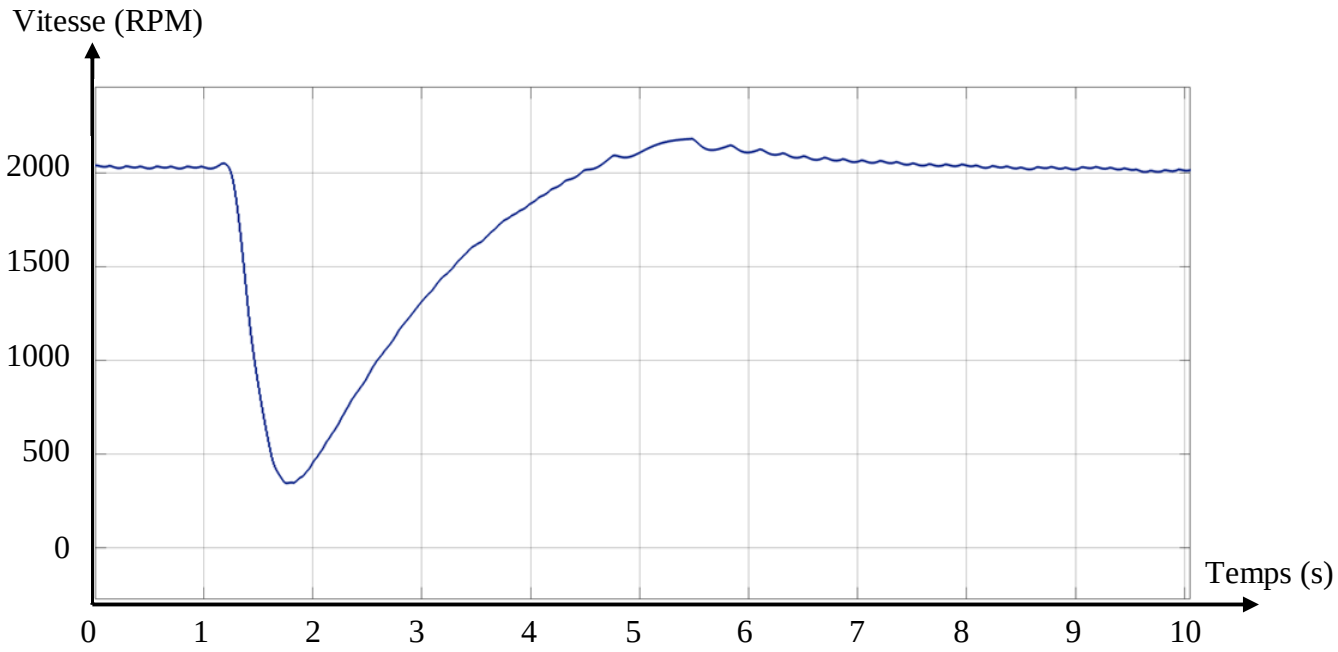


Figure V.12 : La sortie du système durant la perturbation avec le régulateur PID.

V.4.3 Interprétation des résultats :

Les résultats obtenus dans les simulations précédentes nous assurent que la régulation PID nous donne un bon contrôle sur notre système. Nous voyons que la régulation PI était bonne, mais pas optimale à cause de son dépassement et son temps de réponse et stabilité, mais le contrôleur PID fournit un dépassement de 5% qui est le meilleur dépassement pour les régulations de ce genre, aussi une réponse rapide du système et un temps de stabilité très petit. Le seul inconvénient de l'action PID sur ce système particulier est : l'instabilité de la valeur de vitesse de rotation après le changement de consigne, ce défaut n'est pas causé par le régulateur, c'est à cause de la fréquence de la sortie du ventilateur PWM qui est estimée entre 21Khz et 28Khz. Cette valeur n'est pas compatible avec la fréquence standard du régulateur PID (Z) de Simulink qui se varie entre 480 Hz et 960 Hz.

V.5 Conclusion :

Dans ce chapitre, nous avons présenté, comment obtenir les paramètres des régulateurs PI et PID et nous avons transformé la carte Arduino Uno d'une carte d'acquisition à un régulateur réel ce qui permit de réaliser un régulateur numérique avec un prix bas.

Conclusion générale

Conclusion générale :

De nos jours, avec l'évolution des microcontrôleurs, et par l'invention de la carte de développement Arduino, on peut réaliser et étudie des systèmes asservis en temps réel.

L'objectif de ce projet de fin d'études était la conception et la réalisation d'une commande numérique pour la régulation de la vitesse d'un ventilateur à l'aide d'une carte Arduino UNO.

On a utilisé cette carte pour l'acquisition des données entre le système et le PC pour l'identification, puis comme un contrôleur pour la régulation de la vitesse de rotation du ventilateur.

Ce travail nous a permis de toucher à plusieurs domaines et notamment dans le domaine de la régulation numérique, la réalisation, la mise en pratique du régulateur numérique, la conception des circuits électroniques, la programmation d'Arduino et les outils « System Identification », « PID tuner », sous logiciel Matlab qui fait la plus grande partie de notre travail (l'identification, calcule des paramètres du régulateur… etc.)

Durant notre parcours de réalisation de ce travail, nous avons rencontré plusieurs difficultés à savoir : le choix du système réel qu'on doit réguler, à cause du manque des équipements, et le problème de la défaillance du capteur de vitesse « encodeur magnétique » de notre premier choix du système qui était un moteur CC et que nous avons l'acheté déjà défaillant. Ça nous a obligé de commander un autre système « un ventilateur de PC » qui donne des résultats perturbés et bruités en cas de diminution de sa vitesse. Malgré toutes les difficultés, nous avons pu réaliser notre projet, et obtenir sa fonction de transfert à l'aide de la carte Arduino UNO et logiciel Matlab Simulink.

Ainsi nous avons appliqué deux types de régulateurs numériques (PI et PID) à notre système, et on a fait une comparaison entre les deux correcteurs. Elle résulte que le contrôleur PID est plus rapide avec un dépassement bas par rapport au régulateur PI dans la régulation de vitesse.

Enfin comme perspective de notre travail : nous avons le pouvoir de connaître comment modéliser un système quelconque juste en connaissant son entrée et sa sortie. Après nous pouvons le réguler avec notre choix de régulateur.

Références Bibliographiques

[1-1]: ssi science de l'ingénieur, fiche de cours, distribuer l'énergie électrique vers un Mcc : changement de sens & variation de vitesse.

[1-2]: CHRISTOPHE, Le Lann2. (Le PID utilisé en régulation de position et/ou de vitesse de moteurs électriques1). 2006-2007, 24p.

[1-3]: JEAN, Mbihi. (Automatique analogique et techniques de commande et régulation numérique) London.2017. Disponible sur : <https://bit.ly/2ZByNIC>

[1-4]: ROLAND, Longchamp. (Commande numérique de systèmes dynamiques cours d'automatique) lausanne France.2006. Disponible sur : <https://bit.ly/3glRPtu>

[1-5]: BOUICHE,Hachemi ; BRAHAMI,Mohamed. (Commande PID d'un Moteur à Courant Continu). Bejaia : Université Abderrahmane Mira – Faculté de la Technologie Département d'Electronique. 2009-2010, 49p.

[1-6]: BENNIS, Najib. Méthodes empiriques de synthèse des régulateurs PID. Disponible sur: < http://www.specialautom.net/synthese-empirique.htm>

[1-7]: <http://php.iai.heig-vd.ch/~mee/cours/cours_ra/Chap_04/html/node7.htm>

[1-8]: B.Hachemi, B.Mohamed, « Commande Pid D'un Moteur A Courant Continu», Mini Projet,Université Abderrahmane Mira – Bejaia, 2010.

[1-9]: ZAOUIA Abdessamed. (Réglage Optimal des Paramètres du Régulateur PID). Guelma : Faculté des sciences et de la Technologie-Département d'Electroniquee et Télécommunications. 2014, 56p.

[1-10]: B. C. Kuo & F. Golnaraghi «Automatic Control Systems», John wiley and sons, New-York, 2003.

[1-11]: K. J. Astrom & T. Hagglund «Advanced PID Control», ISA, New York, 2006.

[1-12]: C. le Lann «Le PID utilisé en régulation de position et/ou de vitesse de moteurs électriques», Projet de fin d'étude, 2007.

[2-1]: SCHAWARTZ, Marc-Olivier. Arduino pour la domotique. Paris : Dunod, 2015, 254p.

[2-2]: QUETIN, Jean-Christophe. Arduino Apprivoisez l'électronique et le codage. 2018.420p

[2-3]: ARDUINO. What is an Arduino? Disponible sur : <https://learn.sparkfun.com/tutorials/what-is-anarduino/ >

[2-4]: MONK, Simon. Mouvement lumière et son avec ARDUINO et RASPBERRY PI. Paris : bld saint germain, 2016, 330p.

[2-5]: LANDRAULT, Simon. WEISSLINGER, Hippolyte. Arduino : Premiers pas en informatique embarquée. 2014, 454p.
[2-6]: BARTMANN, Erik. LE GRAND LIVRE D'ARDUINO. Paris: bld saint germain, 2014, 585p.
[2-7]: ESKIMON. OLYTE. Arduino pour bien commencer en électronique et en programmation. 2012, 326p.
[2-8]: GENEVEY, Frédéric. DULEX, Jean-Pierre. Arduino à l'école Cours pour l'apprentissage des bases de l'électronique et de la programmation sur Arduino. 2018, 102p.
[2-9]: GILBERT, David. Référence Arduino français Main/Materiel Uno. **[En ligne]**. (Modifié le 01 juillet 2013) Disponible sur :< http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.MaterielUno>. (Consulté le 11/07/2020)
[2-10]: ARDUINO.Arduino-Board. Disponible sur: <https://www.arduino.cc/en/reference/board> (Consulté le 14/07/2020)
[2-11]: Electrotoile. C'est quoi le signal PWM. **[En ligne]**. Disponible sur : <https://electrotoile.eu/c-est-quoi-la-PWM-avec-arduino.php#002>. (Consulté le 18/07/2020)
[3-1]: What is Matlab? < https://www.mathworks.com/discovery/what-is-matlab.html>.
[3-2]: O. BEUCHER, M. WEEKS. INTRODUCTION TO MATLAB & SIMULINK. United States of America, 2008, 404p.
[3-3]: Tutorial «comment faire le réglage optimale pour faciliter l'interface Arduino-Matlab»
[3-4]: https://www.researchgate.net/figure/Les-Blocs-dArduinoIO-necessaires-pour-la-commande-Real-Time-Pacer-Ce-bloc-permet_fig6_314674254
[4-1]: http://www.ordinateur.cc/Matériel/CPUs/6349.html
[5-1]: http://www.mathworks.com
[5-2]: http://www.arduino.cc/
[5-3]: PID Control: A brief introduction and guide, using Arduino.
[5-4]: https://en.wikipedia.org/wiki/PID_controller.

Résumé :

Ce travail a permis l'implémentation du régulateur PID numérique dans une carte Arduino pour la régulation de la vitesse d'un ventilateur de PC.

Nous avons commencé, tout d'abord, par une identification du système en estimant sa fonction de transfert pour obtenir un modèle mathématique décrivant son comportement à l'aide de l'outil de Matlab « System identification ».

Puis nous avons utilisé ce modèle afin de trouver les paramètres du régulateur PI et PID pour pouvoir contrôler notre système à l'aide de l'outil de Matlab « PID Tuner » et après on a implémenté ce régulateur au système à l'aide du 'Matlab Simulink'. Ensuite, on a obtenu des résultats de simulation.

Finalement, on a fait une comparaison entre les résultats des deux régulateurs. On a conclu que le régulateur PID donne une meilleure performance que le régulateur PI.

Mots-clés :

Contrôleur numérique, carte Arduino, régulation, identification, régulateur PID, 'Matlab Simulink', vitesse de rotation, acquisition des données.

Abstract:

This work has allowed the implementation of PID digital regulator in an Arduino Uno board for the purpose of the speed control of a CPU fan of the personal computer.

At first, we started with identifying our system by estimating its function transfer to obtain a mathematical model that describes its behavior with the help of the "System identification" toolbox.

Next, we used this model to find the PI and PID regulator parameters using the "PID tuner" Matlab toolbox for the purpose of being able to control our system. After that, we implemented this regulator to the system using "Matlab Simulink". Then, we got the results after finishing the simulation.

Finally, we made a comparison between the PI and PID controller and we concluded that the PID controller gives us better performance and better results than the PI controller.

Keywords:

Digital controller, Arduino board, controlling, Identification, PID controller, "Matlab Simulink", rotation speed, data acquisition.

ملخص :

لقد تمكنا بفضل هذا العمل من تطبيق المنظم المتناسب المكامل المشتق على لوحة "الأردوينو أونو" من أجل التحكم في سرعة الدوران لمروحة الحاسوب.

في البداية، استهلّينا المشروع عن طريق تعريف نظامنا من خلال تحديد علاقة تحوله من أجل الحصول على علاقة رياضية تعبر عن تغير حركته وهذا بواسطة أداة الماطلاب "تعريف النظام".

بعد ذلك، استعملنا هذه العلاقة الرياضية لايجاد معاملات المنظم المتناسب المكامل والمنظم المتناسب المكامل والمشتق باستعمال أداة الماطلاب " توفيق معاملات المنظم" وهذا من أجل القدرة على التحكم في سرعة المروحة، ثم أدخلنا هذه المعاملات في السيميولينك وتمكنا من الحصول على النتائج.

في النهاية، قمنا بعمل مقارنة بين المنظمين واستنتجنا أن المنظم المتناسب المكامل المشتق يعطينا أداء أحسن ونتائج أفضل من المنظم المتناسب المكامل.

كلمات مفتاحية :

المنظم الرقمي، لوحة الأردوينو، التحكم، التعريف، المنظم المتناسب المشتق المكامل، "الماطلاب سيميولينك"،التحكم في سرعة الدوران، الحصول على البيانات.