الجمهـوريـــة الجزائـريـــة الديمقـراطيـــة الشـعبيـــة
République Algérienne Démocratique et Populaire
وزارة التـــعلـيـــم العـــالـــي والبـحـــث 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 : Génie Industriel

Thème

Réalisation d'une commande numérique à l'aide d'une carte Arduino pour la régulation d'un niveau d'eau dans un réservoir et la supervision sous l'environnement Matlab

Présenté et soutenu publiquement par :

Nom AIRECHE **Prénom** Sofyane

Nom BENYETTOU **Prénom** Abdelghafour

Devant le jury composé de :

Nom et Prénom	Grade	Etablissement	Qualité
Mr. HACHEMI Khalid	MCA	IMSI-Univ d'oran 2	**Président**
Mr. ADDA NEGGAZ Samir	MAA	IMSI-Univ d'oran 2	**Encadreur**
Mme. ZEBIRATE Soraya	Pr	IMSI-Univ d'oran 2	**Examinatrice**

Juin 2017

السلام عليكم ورحمة الله وبركاته

سُبْحَانك اللّهم لَا عِلْم لَنَا إِلاَّ مَا علّمْتَنَا إِنّكَ أنْتَ العَلِيمُ الحَكِيم

اللّهُم عَلّمنا مَايَنْفَعُنا وَ انْفَعْنَا بِمَا عَلّمْتَنا وَزِدْنا عِلْماً

اللّهُم أُرْزُقْنَا عِلْماً نَافِعاً، والحَمْدُ لِلّهِ رَبِّ العَالَمِين

اللّهُم صَلّ و سَلّم وَ بَارِك علَى نَبِيّنَا مُحَمَد (صلّى اللُّه عَلَيْهِ وَ سَلّم)

صَلَاةً وَ سَلَاماً مُتَلَازِمَيْنِ إِلَى يَوْمِ الدِّين

آمين

Remerciements

Mes remerciements vont tout premièrement à Dieu « Allah » le tout puissant pour la volonté, la santé et la patience qu'il m'a donnée pour terminer mon travail.

Tout d'abord, je tiens à exprimer ma profonde gratitude et mes vifs remerciements, respectivement à mon 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 patiente durant la période de l'encadrement et Pour la confiance qu'ils m'ont témoigné, sans oublier leurs conseils avisés, leurs encouragements et leur soutien indéfectible, tout le long de cette étude.

J'exprime également ma reconnaissance et mes sincères remerciements à **Mr. HACHEMI Khalid**, enseignant à l'institut de Maintenance et de Sécurité Industrielle (IMSI) de l'université d'Oran 2, qui m'a fait l'immense honneur d'accepter de présider le jury.

Je tiens beaucoup à témoigner l'expression de ma gratitude et ma reconnaissance à **Mme. ZEBIRATE Soraya** enseignante à l'institut de Maintenance et de Sécurité Industrielle (IMSI) de l'université d'Oran 2, pour l'honneur qu'elle m'a fait de prendre part à ce jury en qualité d'examinatrice

Je tiens également à témoigner ma gratitude et ma reconnaissance à tous les enseignants qui m'ont éduqué et formé pour faire face à la vie.

Enfin, Je ne saurais oublier d'exprimer mes sentiments les plus profonds et mes vifs remerciements à ma famille et mon entourage, ainsi qu'à tous mes amis et collègues qui n'ont cessé de m'encourager et de me soutenir ; et je m'excuse à tous les personnes qui j'oublier ses noms et qui ne sont pas été citées.

Dédicaces

Je dédie le fruit de ce travail à :

_L'âme de notre vénéré **Prophète MOHAMED** : Que la prière et le salut soit sur lui

_ 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 ;

_ Mon père, qui a veillé à mon éducation et fait de moi un homme droit et sage ;

_ Mes chères sœurs bien aimées qui me chérissent ;

_ Mes grand-mères que j'affectionne tendrement, et qui ne cessent de prier pour moi ;

_ Mes chers oncles, que j'aime tous, et qui m'ont toujours soutenu, tout au long de ma vie ;

_ Mes très chères tantes que j'adore particulièrement ;

_ Tous mes cousins et cousines, chacun par son nom ;

_ Ma grande famille : **AIRECHE, GHANIM;**

_ Mes amis qui me connaissent chacun son nom, particulièrement : mon binôme **BENYETTOU Abdelghafour, BOUDALIA Abdelkarim** ;

_ Tous les collègues du l'Institut de l'IMSI de l'université d'Oran

_ Et tous ceux qui m'estiment et pensent à moi.

AIRECHE Sofyane

Dédicaces

Avant toute chose je loue Allah de m'avoir donné la force et le courage d'aboutir à ce travail.

A celle que le miséricordieux a béni et a mis sous ces pieds le paradis.

A la femme la plus courageuse, sensible, généreuse, à celle qui a su me donne amour et joie de vivre, à celle qui a toujours montrée affection et compréhension à mon égard, ma mère que j'aime.

A l'homme de courage et de force à celui qui a toujours été présent, qui m'a appris les vraies valeurs de la vie à celui qui m'a soutenu en toutes circonstances, mon père que j'aime.

A tous ma famille mon frère et ma sœurs mes tantes et mes oncles.

A tous les étudiants de I 'institut de Maintenance et de Sécurité Industriel et spécialement les étudient master 2 génie industriel.

A tous mes amis, pour leurs présence et encouragements. Durant les moments difficiles.

BENYETTOU Abdelghafou

Table de matière

Listes des figures

Listes des tableaux

Liste des acronymes et symboles

AVR	Alf and Vegard's RISC processor.
ADC	Analog to Discret Converter.
CAN	Convertisseur Analogique Numérique.
USB	Universal Serial Bus.
Vin	Voltage in.
GND	Ground.
CA	Courant Alternatif.
CC	Courant Continu.
PWM	Pulse Width Modulation.
ICSP	In Circuit Serial Programming.
IOREF	IO voltage reference.
EEPROM	Electrically Erasable Programmable Read-Only Memory.
LCD	Lquid Crystal Display.
SD	Storage Device.
SPI	Serial Peripheral Interface.
LED	light emitting diode.
GPS	global positioning system.
DC	Direct Courant.
FCEM	Force Contre-ElectroMotrice.
MOSFET	Metal-Oxyde-Semiconductor Field Effect Transistor.
ARX	Auto Régressive à variable eXogène.
ARMAX	Auto régressive à moyenne ajustée et variable exogène.
OE	Output Error.
PID	proportional-Integral-Derivative regulateur.
RST	Regulation-Sensitivity-Tracking regulateur.
RLF	Régulateur à Logique Floue.

Les autres acronymes et les symboles utilisés sont définis dans le texte.

Introduction générale

Introduction Générale

Le développement extraordinaire des processeurs numériques (microprocesseurs, microcontrôleurs) et leur large utilisation dans tous les systèmes de contrôle (domestique et industriel) a mené aux changements importantes dans la conception des systèmes de contrôle. Leurs applications sont à faible coût, les rendent appropriés pour l'usage dans des systèmes de contrôle.

Dans certaines branches d'industries, le problème de contrôle du niveau du liquide est souvent une problématique. La nature du liquide et le frottement du mécanisme de contrôle et d'autres facteurs rendent le système non linéaire. Dans de 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, mais l'utilisation d'un contrôleur PID n'est pas entièrement commode quand il s'agit de contrôler les systèmes non linéaires. Mais ces systèmes peuvent être avec succès commandés en utilisant des contrôleurs à base de logique floue, en raison de leur indépendance du modèle mathématique du système.

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 modélise le système, appliquer à la régulation. On obtient des résultats pour déterminer quelle est le bon régulateur qui assure la meilleure réponse de notre système intitulé "Réalisation d'une commande numérique avec Arduino pour la régulation d'un niveau d'eau dans une cuve et la supervision sous l'environnement Matlab", ce mémoire tend ainsi à démontrer que :

La première partie est consacrée à la carte Arduino, nous allons présenter la carte Arduino qui va jouer le rôle d'une carte acquisition et du contrôleur.

Le chapitre deux introduit les différentes parties du projet et porte sur la description détaillée du système et comment obtenir son modèle mathématique.

Le chapitre trois décrit la conception du contrôleur PI, l'implémentation sur la carte Arduino, et les résultats expérimentaux.

Dans le quatrième chapitre une description générale de la logique floue et la conception du contrôleur floue. Son implémentation sur Arduino et ses résultats expérimentaux. La conclusion est donnée dans la dernière section.

I. Chapitre I

Description de la carte Arduino

I.1 Introduction

Le développement extraordinaire des calculateurs numériques (microprocesseurs, Microcontrôleurs) et leur large utilisation dans les systèmes de commande dans tous les domaines d'applications a entraîné des changements importants dans la conception des systèmes de commande.

L'objectif du ce chapitre est de donner un aperçu sur la carte Arduino, nous allons parler d'une carte Arduino de faible coût.

I.2 Historique de l'Arduino

Projet Arduino est issu d'une équipe d'enseignants et d'étudiants de l'école de Design d'Interaction d'Ivrea (école Interaction Design Institute Ivrea (IDII) est aujourd'hui située à Copenhagen sous le nom de « Copenhagen Institute of Interaction Design » Italie). Ils rencontraient un problème majeur à cette période (avant 2003 - 2004) : les outils nécessaires à la création de projets d'interactivité étaient complexes et onéreux (entre 80 et 100 euros). Ces coûts souvent trop élevés rendaient difficiles le développement par les étudiants de nombreux projets et ceci ralentissait la mise en œuvre concrète de leur apprentissage.

Jusqu'alors, les outils de prototypage étaient principalement dédiés à l'ingénierie, la robotique et aux domaines techniques. Ils sont puissants mais leurs processus de développement sont longs et ils sont difficiles à apprendre et à utiliser pour les artistes, les designers d'interactions et, plus généralement, pour les débutants.

Leur préoccupation se concentra alors sur la réalisation d'un matériel moins cher et plus facile à utiliser. Ils souhaitaient créer un environnement proche de Processing, ce langage de programmation développé dès 2001 par Casey Reas (Casey Reas était lui-même enseignant à IVREA, pour Processing, à cette époque) et Ben Fry, deux anciens étudiants de John Maeda au M.I.T., lui-même initiateur du projet DBN (Design By Numbers, un langage de programmation spécialement dédié aux étudiants en arts visuels)

En 2003, Hernando Barragan, pour sa thèse de fin d'études, avait entrepris le développement d'une carte électronique dénommée Wiring, accompagnée d'un environnement de programmation libre et ouvert. Pour ce travail, Hernando Barragan réutilisait les sources du projet Processing. Basée sur un langage de programmation facile d'accès et adaptée aux développements de projets de designers, la carte Wiring a donc inspiré le projet Arduino (2005).

Comme pour Wiring, l'objectif était d'arriver à un dispositif simple à utiliser, dont les coûts seraient peu élevés, les codes et les plans « libres » (c'est-à-dire dont les sources sont ouvertes et peuvent être modifiées, améliorées, distribuées par les utilisateurs eux-mêmes) et, enfin, « multi-plates-formes » (indépendant du système d'exploitation utilisé). Conçu par une équipe de professeurs et d'étudiants (David Mellis, Tom Igoe, Gianluca Martino, David

Cuartielles, Massimo Banzi ainsi que Nicholas Zambetti), l'environnement Arduino est particulièrement adapté à la production artistique ainsi qu'au développement de conceptions qui peuvent trouver leurs réalisations dans la production industrielle.

Le nom Arduino trouve son origine dans le nom du bar dans lequel l'équipe avait l'habitude de se retrouver. Arduino est aussi le nom d'un roi italien, personnage historique de la ville « Arduin d'Ivrée », ou encore un prénom italien masculin qui signifie « l'ami fort » **(TLILI, 2017)**.

I.3 Présentation de l'Arduino

Arduino est un ensemble d'outils matériel et logiciel pour le prototypage électronique et l'apprentissage de la programmation des microcontrôleurs. Les schémas électroniques des cartes Arduino et les codes sources pour la partie développement des programmes, sont distribués librement et téléchargeables gratuitement. L'environnement Arduino a été conçu pour être utilisé facilement par les débutants sans expérience de la programmation ou de connaissances en électronique. La **Figure I.1** illustre un type des cartes Arduino.

Figure I.1. *La carte Arduino UNO.*

Il existe plusieurs types de la carte Arduino, Arduino UNO, Arduino MEGA, Arduino NANO, Arduino DEU ... etc. La seule différence entre eux c'est les performances et les caractéristiques de chacune.

I.3.1 Description de la carte Arduino UNO :

C'est un circuit imprimé comportant tous les composants électroniques nécessaires pour faire fonctionner un microcontrôleur (Atmega 328) associé à une interface USB lui permettant de communiquer avec un ordinateur (CHELLY & CHARED, 2014).

I.3.2 Fiche technique de la carte ARDUINO UNO:

- Microcontrôleur : ATmega328
- Tension d'alimentation interne = 5V
- Tension d'alimentation (recommandée)= 7 à 12V, limites =6 à 20 V
- Entrées/sorties numériques : 14 dont 6 sorties PWM
- Entrées analogiques 10 bits = 6
- Courant max par broches E/S = 40 mA
- Courant max sur sortie 3,3V = 50mA
- Mémoire Flash 32 KB dont 0.5 KB utilisée par le boot-loader
- Mémoire SRAM 2 KB
- mémoire EEPROM 1 KB
- Fréquence horloge = 16 MHz
- Dimensions = 68.6mm x 53.3mm x 15 mm

La **Figure I.2** montre les différents constituants de la carte Arduino (UNO).

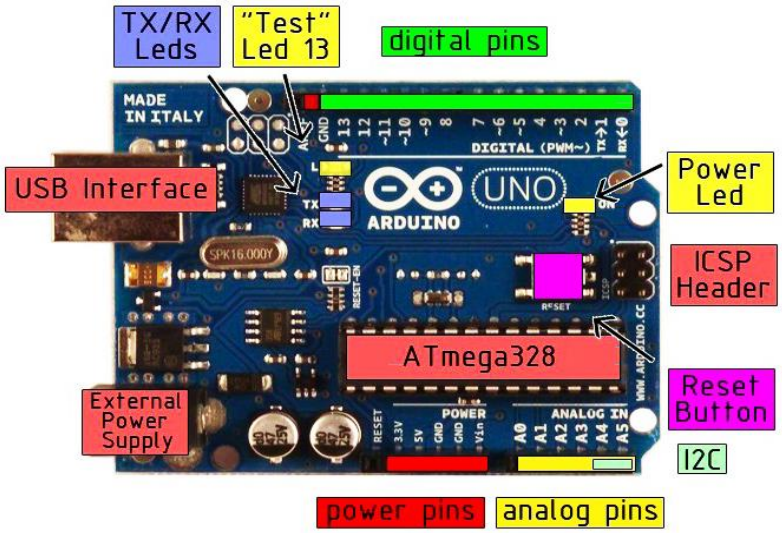

Figure I.2. *Les différents constituants de la carte Arduino (UNO).*

I.3.3 Matériel

Un module Arduino est généralement construit d'un microcontrôleur Atmel AVR (ATmega328 ou ATmega2560 pour les versions récentes, ATmega168 ou ATmega8 pour les plus anciennes) et des composants complémentaires qui facilitent la programmation et

l'interfaçage avec d'autres circuits. Chaque module possède au moins un régulateur linéaire 5 V et un oscillateur à quartz de 16 MHz. Le microcontrôleur est préprogrammé avec un (boot-loader) de façon à ce qu'un programmateur dédié ne soit pas nécessaire (OULD SIDI Mohamed & SERSOUB, 2015).

I.3.3.1 ADC (Analog to Discret Converter)

La carte Arduino UNO dispose de 6 entrées analogiques notées A0, A1…A5 mais d'un seul convertisseur analogique/numérique, la durée d'une conversion est de l'ordre de 100µs.Il a une résolution de 10 bits. La donnée numérique qu'il fournit après conversion est donc comprise entre 0 et 1024 (CHELLY & CHARED, 2014).

La **Figure I.3** montre les signaux d'entrés et de sortis dans et depuis le convertisseur *ADC*.

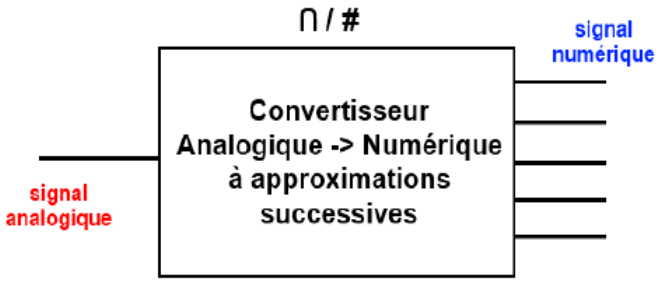

Figure I.3. *Type du CAN de la carte Arduino UNO.*

I.3.3.2 Alimentation

L'Arduino UNO peut être alimenté par la connexion USB ou par une alimentation électrique externe. L'alimentation électrique est automatiquement sélectionnée. L'alimentation externe (non-USB) peut provenir d'un adaptateur CA/CC (mural) ou d'une batterie. L'adaptateur peut être connecté en branchant une prise de centre positif de 2,1 mm dans la prise de courant de la carte représentée sur la **Figure I.4**. Les broches de raccordement d'une batterie peuvent être insérées dans les embases mâles *Gnd* et *Vin* (tension en entrée) du connecteur POWER (alimentation).

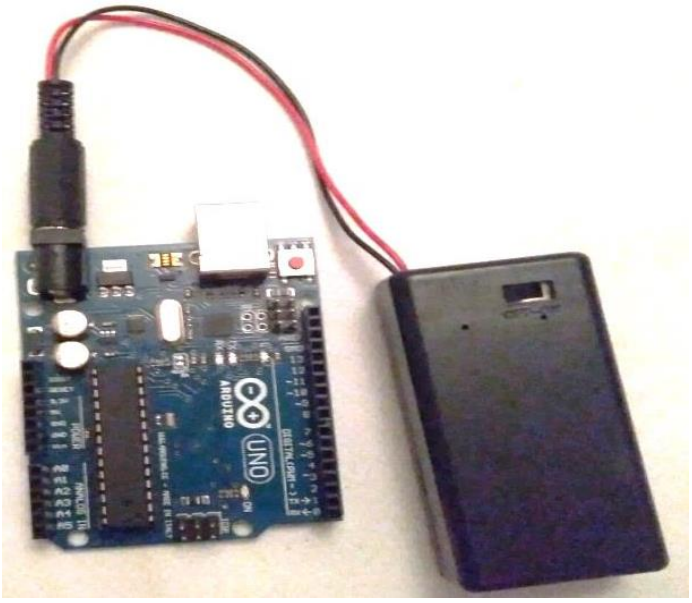

Figure I.4. *Carte Arduino Avec une boite de pile 9V.*

C'est une solution très pratique et sûre pour éviter tout problème avec le port USB de son ordinateur. La carte peut fonctionner avec une alimentation externe de 6 à 20 volts. Si cependant elle est alimentée à moins de 7 V, la broche du 5 V peut fournir moins de cinq volts et la carte peut devenir instable. Si elle reçoit plus de 12 V, le régulateur de tension peut surchauffer et endommager la carte. La plage recommandée et de 7 à 12 volts (Kouadri bou djelthia & Kouidri zourgui, 2013).

I.3.3.3 Les entrées/sorties

Le langage Arduino vient avec un nombre important de fonction de base permettant d'interagir avec son environnement. Les fonctions les plus utilisée sont les fonctions d'entrée/sorties. Ce sont elles qui permettent d'envoyer ou de mesurer une tension sur une des broches de la carte (LECHALUPÉ , 2014).

Dans un premier temps, avant d'effectuer une mesure ou d'envoyer une commande, Il est nécessaire de définir la direction des broches utilisées. Pour cela on fait appel à la fonction pin Mode en lui donnant d'une part, la broche concernée, et d'autre part, la direction, (LECHALUPÉ , 2014) :

```
void setup () {

pin Mode (1, OUTPUT) ; // Broche 1 en sortie

pin Mode (2, INPUT) ; // Broche 2 en entrée
```

}

Une fois cette configuration faite, on peut procéder à l'utilisation des broches. Toutes les broches sont capables d'écrire et de lire des données numériques dire des 0 (0V) ou des 1 (5V). Mais, certaines disposent de fonctionnalité supplémentaire.

Tout d'abord, toutes les cartes Arduino possèdent des entrées analogiques. Ce sont les broches A0-A1-A2 etc. Elles permettent de lire des tensions analogiques (comprise entre 0 et 5V) et de le convertir en entier (compris entre 0 et 1023) proportionnellement à la tension mesurée. Certaines cartes Arduino possède des sorties analogiques faisant l'opération inverse (met une tension sur la broche proportionnellement à l'entier donné), mais ce n'est pas le cas pour l'Arduino UNO.

Pour pouvoir tout de même contrôler des composants autrement qu'en « tout ou rien » il est possible d'utiliser des broches *PWM*.

Toutes ces fonctionnalités sur les broches d'entrées sorties sont utilisables par le biais de quatre fonctions (LECHALUPÉ , 2014) **:**

- **digitalRead(pin)**

Mesure une donnée numérique sur une des broches, la broche en question doit être réglée en entrée.

- **DigitalWrite (pin, value)**

Écrit une donnée numérique sur une des broches, la broche Concernée doit être réglée en sortie. Le paramètre valu doit être égal à HIGH (état 1 soit 5V) ou LOW (état 0 soit 0V).

- **AnalogRead (pin)**

Mesure une donnée analogique sur une des broches (compatible seulement), la broche doit être réglée sur entrée.

- **analogWrite (pin, value)**

Écrit une donnée sous forme de *PWM* sur une des broches (Compatible uniquement), la broche doit être réglée en sortie. Le paramètre valu doit être compris dans l'intervalle [0;255].

I.3.3.4 PWM

Ce signale signifie en anglais « Pulse Width modulation » ; en français l'on parle de MLI : « Modulation de largeur d'impulsion ». Il s'agit d'un artifice permettant de produire une tension variable à partir d'une tension fixe.

Ce sont les broches annotées par un tilde ~ sur la carte. Les *PWM* (Pulse Width Modulation) sont utilisées pour synthétiser des signaux analogiques en modulant le temps passé à l'état 1 (5V). Le signal obtenu est représenté figure 5. En utilisant une fréquence relativement élevée, les *PWM* permettent de commander certains composants comme si il recevait une tension analogique. Cela provient du fait que les composants utilisés dans l'électronique analogique, ne changes pas d'états instantanément. Par exemple, une ampoule à incandescence reste chaude et éclaire un court instant après avoir été éteinte. Ce phénomène est généralement invisible à l'œil nu. Grâce à elles, on pourra par exemple faire varier l'intensité d'une LED. La plupart des cartes Arduino utilisent des *PWM* cadencées à 490 Hz environ. (LECHALUPÉ , 2014).

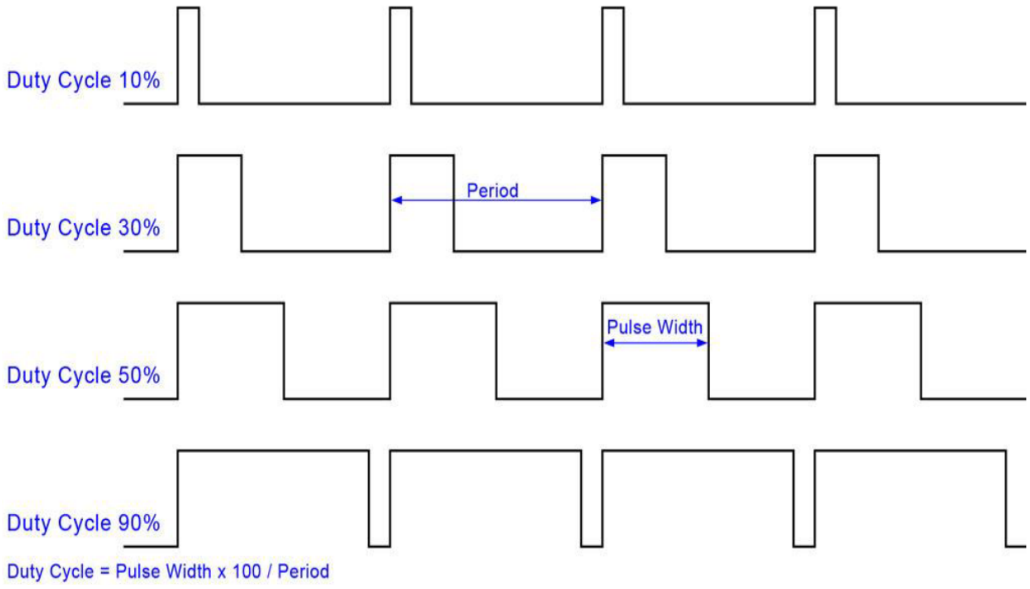

Figure I.5. *Signal PWM.*

I.3.3.5 VIN

La tension en entrée vers la carte Arduino quand on utilise une source d'alimentation externe (par opposition aux 5 V provenant de la connexion USB ou d'une autre source d'alimentation régulée). Vous pouvez fournir une tension au travers de cette broche, ou, si l'alimentation passe par la prise électrique, y accéder au travers de cette broche (LECHALUPÉ , 2014).

I.3.3.6 ICSP

ICSP signifie « In-circuit Serial Programming ». Dans le cas où le microcontrôleur d'une carte Arduino viendrait à ne plus fonctionner, il est possible de remplacer la puce défectueuse. Ainsi le port ICSP vous permet de configurer une nouvelle puce pour la rendre compatible avec l'IDE Arduino. En effet le microcontrôleur de la carte Arduino possède un

boot-loader (ou bios) qui lui permet de dialoguer avec l'IDE. Il faut pour cela acheter un programmateur ISP ou il est aussi possible d'utiliser une autre carte Arduino comme programmateur *ISP* (MARGOlIS, 2011)**.**

I.3.3.7 IOREF

Cette broche de la carte Arduino fournit la tension de référence à laquelle le microcontrôleur fonctionne. Un blindage correctement configuré peut lire la tension de la broche IOREF et sélectionner la source d'alimentation appropriée ou activer les convertisseurs de tension sur les sorties pour travailler à 5 ou 3,3 V (Kouadri bou djelthia & Kouidri zourgui, 2013).

I.4 Logiciel

Le logiciel de programmation des modules Arduino est une application Java, libre (open source) et, servant d'éditeur de code et de compilateur, et qui peut transférer le programme au travers de la liaison série (RS-232, Bluetooth ou USB selon le module). Il est également possible de se passer de l'interface Arduino, et de compiler et uploader les programmes via l'interface en ligne de commande ou par un autre logiciel de programmation comme « éclipse ».

Le langage de programmation utilisé est le C++, compilé avec *AVR*-g++, et lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties. La mise en place de ce langage standard rend aisé le développement de programmes sur les plates-formes Arduino, à toute personne maîtrisant le C ou le C++ (LECHALUPÉ , 2014).

I.4.1 Programmation avec langage " Arduino C "

I.4.2 Structure d'un programme

Chaque programme peut être décomposé en trois parties la **Figure I.6** ci-dessous montre ça :

- La partie déclaration des variables (optionnelle).
- La partie initialisation et configuration des entrées/sorties: la fonction setup ().
- La partie principale qui s'exécute en boucle: la fonction loop ().

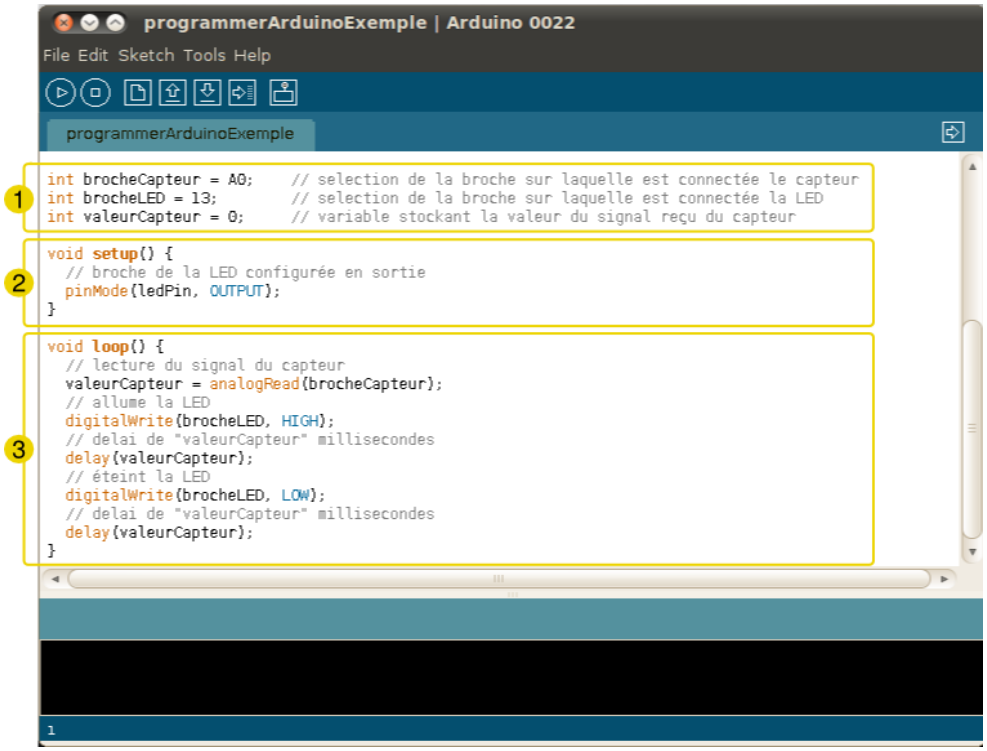

Figure I.6. *Composition minimum d'un programme en Arduino C*

I.4.3 Coloration syntaxique

Lorsque le code est écrit dans l'interface de programmation, certains mots apparaissent en différentes couleurs qui clarifient le statut des différents éléments :

- En orange, apparaissent les mots-clés reconnus par le langage Arduino comme des fonctions existantes. Lorsqu'on sélectionne un mot coloré en orange et qu'on effectue un clic avec le bouton droit de la souris, on aura la possibilité de choisir « Find in reference » : cette commande ouvre directement la documentation de la fonction sélectionnée.
- En bleu, apparaissent les mots-clés reconnus par le langage Arduino comme des constantes.
- En gris, apparaissent les commentaires qui ne seront pas exécutés dans le programme. Il est utile de bien commenter son code pour s'y retrouver facilement ou pour le transmettre à d'autres personnes. L'on peut déclarer un commentaire de deux manières différentes :
- Dans une ligne de code, tout ce qui se trouve après « // » sera un commentaire.

– On peut encadrer des commentaires sur plusieurs lignes entre « /* » et « */».

I.4.4 La syntaxe du langage

I.4.4.1 Ponctuation

Le code est structuré par une ponctuation stricte (ASTALASEVEN & ESKIMONET, 2012):

– Toute ligne de code se termine par un point-virgule « ; »
– Le contenu d'une fonction est délimité par des accolades « { » et «} »
– Les paramètres d'une fonction sont contenus pas des parenthèses « (» et «) ».

I.4.4.2 Les variables

Une variable est un espace réservé dans la mémoire de l'ordinateur. C'est comme un compartiment dont la taille n'est adéquate que pour un seul type d'information. Elle est caractérisée par un nom qui permet d'y accéder facilement (ASTALASEVEN & ESKIMONET, 2012).

Il existe différents types de variables identifiés par un mot-clé dont les principaux sont (ASTALASEVEN & ESKIMONET, 2012) :

– Nombres entiers (*int*).
– Nombres à virgule flottante (*float*).
– Texte (*String*).
– Valeurs vrai/faux (*booléen*).

Un nombre à décimales, par exemple est: *int Val=10*; Int est le type de variable, 10 est sa valeur.

I.4.4.3 Les fonctions

Une fonction (également désignée sous le nom de procédure ou de sous-routine) est un bloc d'instructions que l'on peut appeler à tout endroit du programme. Le langage Arduino est constitué d'un certain nombre de fonctions, par exemple (ASTALASEVEN & ESKIMONET, 2012) :

analogRead(), digitalWrite() ou delay().

Comme est possible de déclarer nos propres fonctions par exemple :

```
void clignote()

{

digitalWrite (brocheLED, HIGH) ;

delay (1000) ;

digitalWrite (brocheLED, LOW) ;

delay (1000) ;

}
```

I.4.4.4 Les structure de contrôle

If else *:* est utilisé pour dire sinon si on veut tester une entrée digitale et faire n'action si elle vaut HIGH.

If : vérifie si une certaine condition est atteinte et exécute des instructions contenues dans la structure.

else : il faut garder à l'esprit qu'une seule partie des instructions sera exécutée, selon les conditions vérifiées.

***While*( )** : continuera infiniment tant que l'expression entre parenthèse est vraie. Cela peut être dans le code, comme l'incrémentation d'une variable, ou une condition extérieure

I.5 Bibliothèque externes

Une bibliothèque est un ensemble de fonctions utilitaires, regroupées et mises à disposition des utilisateurs de l'environnement Arduino afin de ne pas avoir besoin de réécrire des programmes parfois complexes. Les fonctions sont regroupées en fonction de leur appartenance à un même domaine conceptuel (mathématique, graphique, etc). Arduino comporte par défaut plusieurs bibliothèques externes.

Lors d'utilisation d'une bibliothèque, l'instruction suivante doit être ajoutée au début de programme.

```
#include <la_bibliothèque.h>
```

Cette commande inclut au code source tout le contenu de la bibliothèque. Les fonctions qu'elle contient peuvent alors être appelées au même titre que les fonctions de base.

Les bibliothèques fournies par défaut dans le logiciel Arduino sont :

- EEPROM : lecture et écriture de données dans la mémoire permanente.
- Ethernet : pour se connecter à Internet en utilisant le Shield Ethernet.
- LiquidCrystal : pour contrôler les afficheurs à cristaux liquides (LCD).
- SD : pour la lecture et l'écriture de données sur des cartes SD.
- Servo : pour contrôler les servomoteurs.
- SPI : pour communiquer avec les appareils qui utilisent le protocole de communication SPI (Serial Peripheral Interface).
- Stepper : pour commander des moteurs « pas à pas ».
- Wire : pour interfacer plusieurs modules électroniques sur un bus de données utilisant le protocole de communication *TWI* (Two Wire Interface) / I2C.

I.6 Circuits additionnels

Il est possible de spécialiser la carte Arduino en l'associant avec des circuits additionnels que l'on peut fabriquer soi-même ou acheter déjà montés. Lorsqu'ils se branchent directement sur la carte, ces circuits s'appellent des « shields » ou cartes d'extension (**Figure I.7**).

Ces circuits spécialisés apportent au système des fonctionnalités diverses et étendues dont voici quelques exemples :

- Ethernet: communication réseau.
- Bluetooth ou Zigbee : communication sans fil.
- Pilotage de moteurs (pas à pas ou à courant continu).

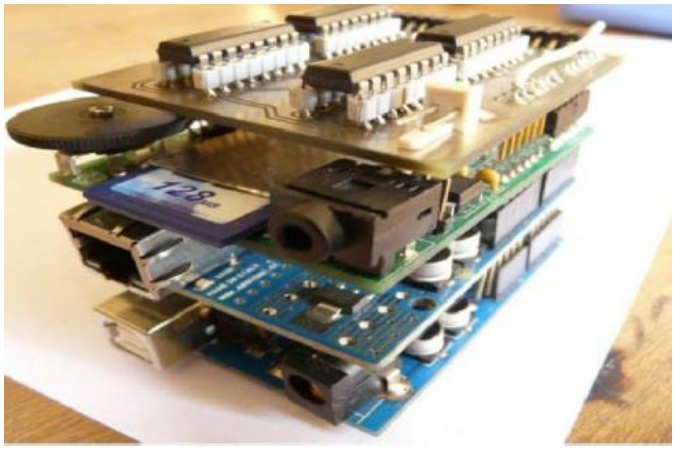

Figure I.7. *Une carte Arduino équipée de plusieurs cartes d'extension.*

- Pilotage de matrices de LED : pour piloter de nombreuses LED avec peu de sorties
- Ecran LCD : pour afficher des informations
- Lecteur de carte mémoire : lire ou stocker des données
- Lecteur de MP3
- GPS : pour avoir une information de position géographique
- Joystick

I.7 Conclusion

Les microcontrôleurs sont largement utilisés dernièrement, parmi eux l'Arduino, des dizaines de milliers de designers, d'ingénieurs, de chercheurs, d'enseignants et même d'entreprises l'utilisent pour réaliser des projets dans de multiples domaines :

- Prototypage rapide de projets innovants utilisant l'électronique.
- Production artisanale d'objets numériques et de machines-outils à faible cout.
- Captation et analyse des données scientifiques.
- Projets pédagogiques.

Dans ce chapitre on a parlé de la carte Arduino, et on a cité la plupart de ses caractéristiques, ainsi ses performances, dans les chapitres suivants on va expliquer le rôle de cette carte dans notre projet.

Chapitre II

Description de matériel et identification de système

II.1 Introduction

Un système industriel c'est un ensemble des équipements (étuve, compresseur… etc.), et des matériels (actionneurs, capteur, détecteur … etc.), intègrent entre eux. Pendant la phase de maturité de cycle de vie de système, il donne un bon rendement, grâce à l'efficacité de sa commande et sa robustesse, mais après certains temps de fonctionnement le système devient instable, perd sa fiabilité, et son rendement va s'abaisser, et il commence à donner des mesures erronées pour son asservissement, donc des changements de comportement va se crée dans le system, par conséquent sa commande n'est plus valable dans ce cas. Pour augmenter le rendement il faut implémenter une nouvelle commande convenable à ces changements sur le système, et ça nous oblige de déterminer son modèle (sa fonction de transfert), pour cela il existe une méthode très utile en pratique c'est *l'identification des systèmes*. Même les nouveaux systèmes qui n'ont pas des modèles (en cas de perte de cahier de charge), l'identification est une solution pour les déterminer.

Le domaine de l'identification est actuellement très actif. En effet ces modèles sont nécessaires dans le cadre de la commande prédictive.

Dans ce chapitre, nous allons présenter notre système étudié, matériel utilisé, et la méthode d'identification et les étapes à suivre pour obtenir le modèle de notre système, et ses résultats, afin que nous puissions d'appliquer la commande de la régulation appropriée à notre système (voir les chapitres suivants), et aussi afin de connaitre bien le comportement intérieur du système.

II.2 Description du système

Pour attendre notre but, dans la régulation de notre système, il est impérativement de le construire en utilisant différent équipements et matériel, le matériel utilisé est :

- Cuve d'eau.
- Capteur de niveau du liquide.
- Pompe à eau à courant continue.
- Ordinateur.
- Circuit électronique (Driver moteur DC + Arduino).

La **Figure 0.1** montre un schéma explicatif d'une chaine d'acquisition de notre système pour le banc d'essai d'une cuve d'eau.

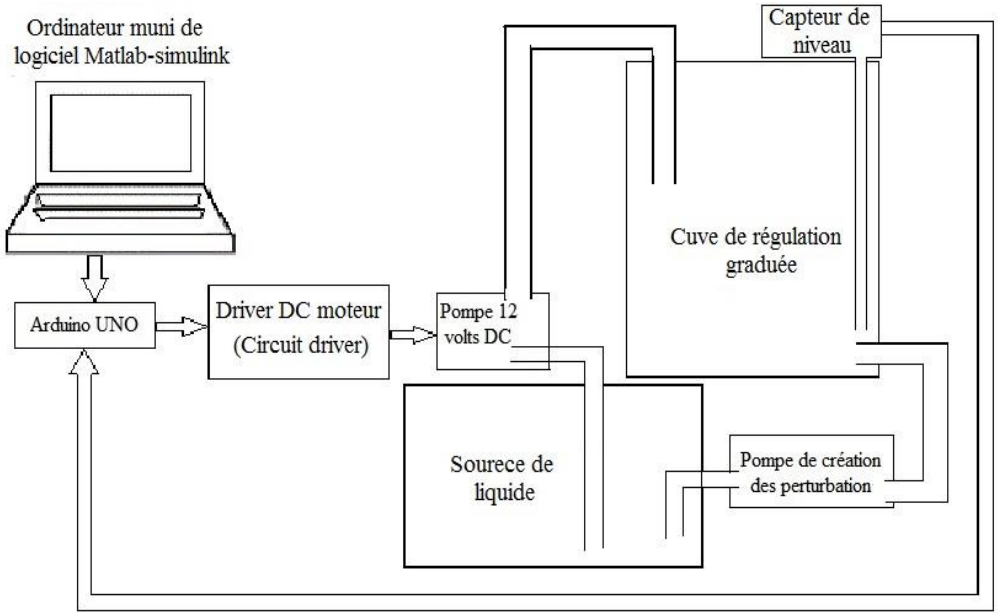

Figure 0.1. *Schéma synoptique de banc d'essai.*

II.2.1 Cuve d'eau

Le réservoir a une configuration géométrique rectangulaire, de hauteur graduée d'environ 18 cm (**Figure 0.2**), Il est alimenté par une arrivée d'eau. Le débit livré par cette arrivée est contrôlé par une pompe, Le réservoir dispose d'une pompe permettant de réguler le débit de sortie. On peut varier le débit de sortie pour simuler les fuites (perturbations).

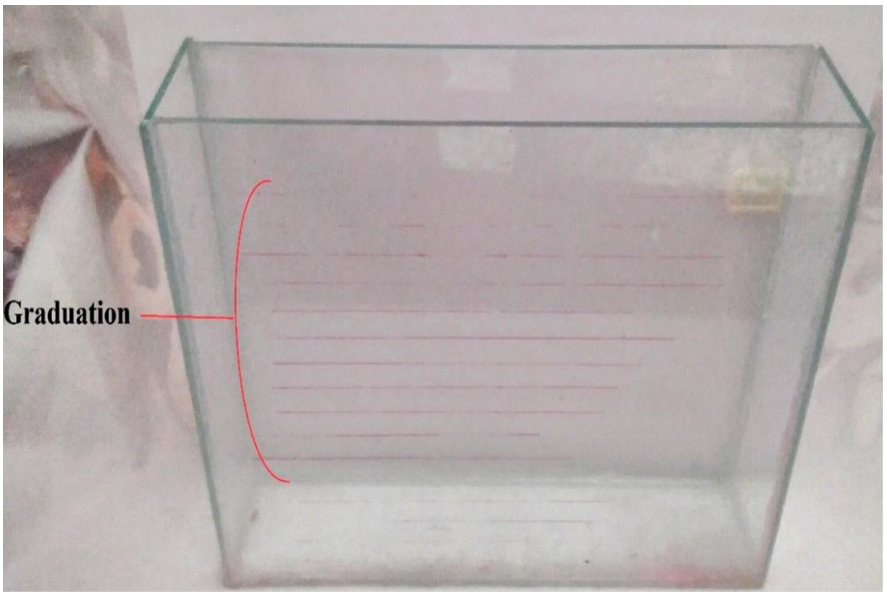

Figure 0.2. *La cuve de banc d'essai.*

II.2.2 Capteur de niveau du liquide

C'est un potentiomètre fixée dans le coin supérieur latéral du cuve muni d'un flotteur qui donne un mouvement mécanique si le niveau d'eau augmente, et ça va se crée par un mouvement rotoïde au niveau de bras de potentiomètre. La carte Arduino reçoit le signal générer par le potentiomètre, ce signal est exploité pour connaitre le niveau d'eau dans la cuve (**Figure 0.3**).

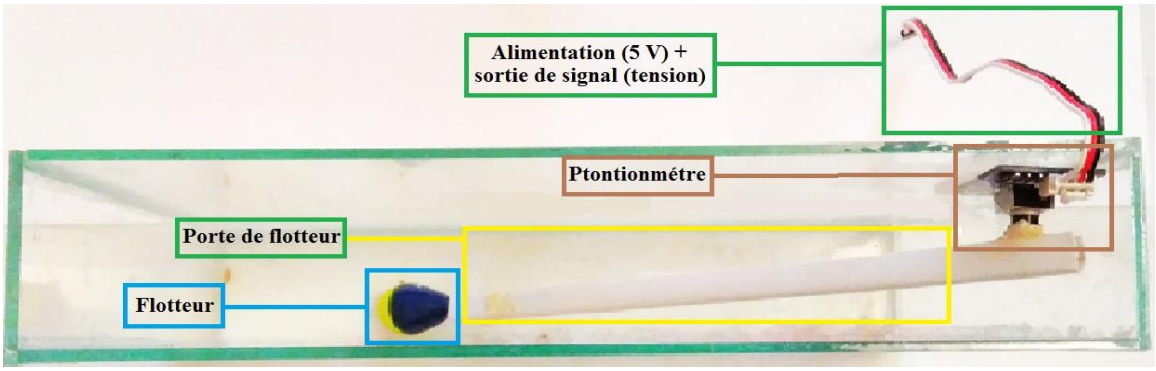

Figure 0.3. *Capteur de niveau.*

La plupart des capteurs ont le même principe de fonctionnement, ils converissent la grandeur mesurée en un signal exploitable, en présence d'une source d'énergie exterieur (**Figure 0.4**.).

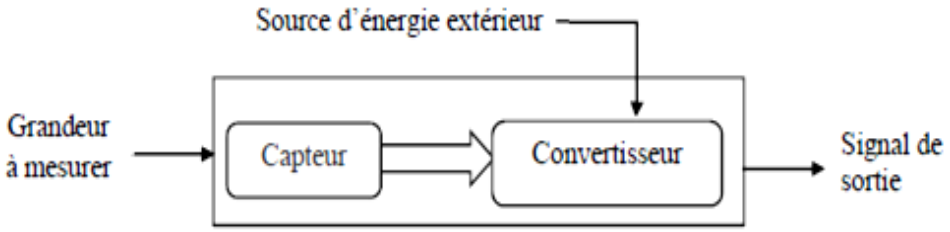

Figure 0.4. *Schéma de principe de fonctionnement de capteur.*

II.2.3 Pompe à eau à base d'un moteur à CC

La pompe est construite à base d'un moteur à *courant continu* d'une tension nominale 12 volts (**Figure 0.5**), le flux de liquide fourni par la pompe est proportionnel à la vitesse du moteur.

Figure 0.5. *Pompe à eau et la cuve de source d'eau.*

II.2.3.1 Constitution d'un moteur à cc et principes physiques

Une machine électrique à courant continu est constituée (**Figure 0.6**) (Moteur à courant continu, 2015) :

- D'un stator qui est à l'origine de la circulation d'un flux magnétique fixe créé soit par des enroulements *statoriques* (bobinage) soit par des aimants permanents. Il est aussi appelé « inducteur » en référence au fonctionnement en génératrice de cette machine.

- D'un rotor bobiné. Les enroulements *rotoriques* sont aussi appelés enroulements d'induits, ou communément « induit » en référence au fonctionnement en génératrice de cette machine.

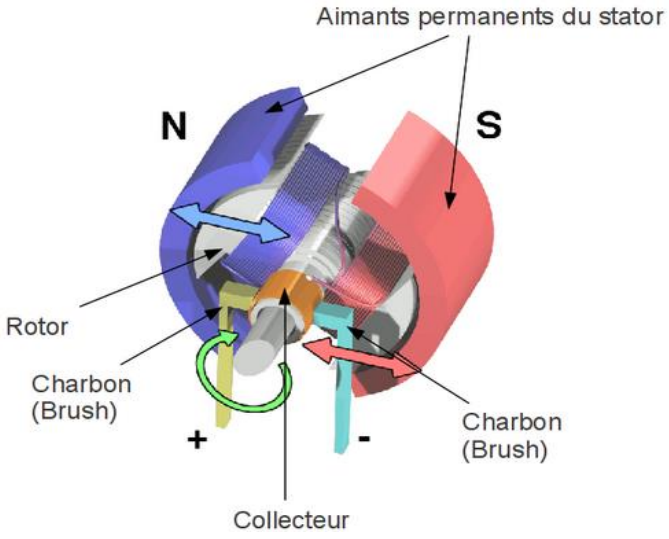

Figure 0.6. *Schéma montre la constitution d'un moteur à CC.*

Le courant *I*, injecté via les balais au collecteur, traverse le bobinage du rotor et change de sens (commutation) pendant la rotation grâce au système balais/collecteur. Ceci permet de maintenir la magnétisation du rotor perpendiculaire à celle du stator pendant la rotation.

L'existence du couple s'explique par l'interaction magnétique entre stator et rotor et est proportionnelle à *I*.

II.2.3.2 Force contre-électromotrice

Dans le cas d'un fonctionnement en génératrice, le bobinage du rotor, traversé par le courant *I* se déplace dans le champ *statorique*. Il est donc le siège d'un courant induit (loi de Lenz) proportionnelle à l'intensité du champ *statorique* et à sa vitesse de déplacement, donc à la fréquence de rotation. L'ensemble de ces forces a pour conséquence l'apparition d'une force électromotrice (FEM) globale aux bornes de l'enroulement *rotorique* qui est proportionnelle à l'intensité du champ *statorique* et à la vitesse de rotation du moteur et qui permet la production de courant (Moteur à courant continu, 2015).

Dans le cas d'un fonctionnement en moteur, cette FEM est produit également mais s'oppose au courant d'alimentation du moteur. Elle est alors appelée "force contre-électromotrice" (FCEM) (Moteur à courant continu, 2015).

II.2.3.3 Modélisation de la partie électrique

La partie électrique du moteur peut être simplifiée au schéma dans la **Figure 0.7** (Moteur à courant continu, 2015)

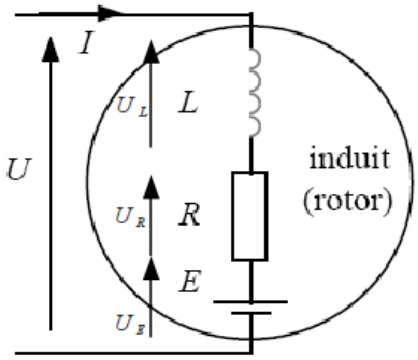

Figure 0.7. *Schématisation de moteur à courant continu***.**

Où :

- I est le courant traversant l'induit (A).
- U est la tension aux bornes de l'induit (V).
- L est l'inductance des enroulements du moteur (mH).
- R est la résistance électrique interne du moteur (Ohm).
- E est la force contre-électromotrice (symbolisé par un générateur).
- E_L , U_R et U_E sont les tensions aux bornes de l'inductance, de la résistance et de la FCEM.

Les équations électriques sont (Moteur à courant continu, 2015):

$$U(t) = U_L\,(t) + U_R(t) + U_E(t) = L \cdot (dI(t)/dt)\; + R \cdot I(t) + U_E(t) \qquad \text{(II.1)}$$

$$U_E(t) = K_E \cdot \omega(t) \qquad \text{(II.2)}$$

Ou K_E est la constante de force électromotrice qui relié cette FCEM à la vitesse de rotation de l'arbre moteur.

II.2.3.4 Modélisation de la partie mécanique

Le modèle mécanique simplifié consiste à représenter le rotor par un volant d'inertie J soumis à (Moteur à courant continu, 2015) :

- un couple moteur C_m provenant du champ magnétique tel que $C_m = K_c \cdot I(t)$ où K_c est la constante de couple

- un couple de frottement C_f proportionnel à la vitesse de rotation du rotor tel que :

$$C_f = f \cdot \omega(t) \quad \text{(II.3)}$$

où f est le coefficient de frottement visqueux.

Le principe fondamental de la dynamique (seconde loi de Newton) appliqué à un solide en rotation permet d'écrire (Moteur à courant continu, 2015) :

$$C_m - C_f = J.(d\omega(t)/dt) \quad \text{(II.4)}$$

Donc :

$$K_c \cdot I(t) - f \cdot \omega(t) = J \cdot (d\omega(t)/dt) \quad \text{(II.5)}$$

II.2.3.5 Fonction de transfert de système

$U(t)$: est l'entrée du système et $\omega(t)$ est la sortie. On considère que les conditions initiales sont nulles.

Transformation dans le domaine de Laplace

1) Equations électriques :

$$U(p) = L \cdot p \cdot I(p) + R \cdot I(p) + U_E(p) \quad \text{(II.6)}$$

$$U_E(p) = K_E \cdot \Omega(p) \quad \text{(II.7)}$$

➔ $$U(p) = L \cdot p \cdot I(p) + R \cdot I(p) + K_E \cdot \Omega(p) \quad \text{(II.8)}$$

2) Equations mécanique :

$$K_c \cdot I(p) - f \cdot \Omega(p) = J \cdot p.\Omega(p) \quad \text{(II.9)}$$

La fonction de transfert donne :

$$\textbf{(II.8)} \rightarrow U(p) - K_E \cdot \Omega(p) = I(p)(L \cdot p + R) \quad \text{(II.10)}$$

$$\rightarrow \quad I(p) = (U(p) - K_E \cdot \Omega(p))/((L \cdot p + R)\,) \quad \text{(II.11)}$$

En remplaçant dans l'équation **(II.9)**

$$K_c \cdot \big((U(p) - K_E \cdot \Omega(p)\big)/(L \cdot p + R)\big) - f \cdot \Omega(p) = J \cdot p.\Omega(p) \quad \text{(II.12)}$$

$$\rightarrow K_c \cdot (U(p)/(L \cdot p + R)) = \big(J \cdot p + f + ((K_c \cdot K_E)/(L \cdot p + R)) \cdot \Omega(p)\big) \quad \text{(II.13)}$$

$$\rightarrow \Omega(p)/U(p) = K_c/\big((J \cdot p + f)(L \cdot p) + K_c \cdot K_E\big) \quad \text{(II.14)}$$

La fonction de transfert du moteur à cc est :

$$H(p) = \Omega(p)/U(p) = K_c/(J \cdot L \cdot p^2 + (J \cdot R + L \cdot f) \cdot p + R \cdot f + K_c K_E) \quad \text{(II.15)}$$

II.2.4 Circuit électronique

La plupart des régulations des systèmes contient des composants électroniques, de même notre projet nécessite un circuit électronique qui se constitue des composants pour le pilotage de signal de commande, et le signal d'alimentation. Le circuit électronique est très important pour le fonctionnement de notre système, et aussi pour sa régulation. Pour cela on est besoin de deux circuits :

- Circuit de puissance
- Circuit de commande

II.2.4.1 Circuit de puissance

On a fait la conception de ce circuit sous logiciel *Proteus ISIS* afin de l'imprimer, la **Figure 0.8** ci-dessous donne notre circuit sous logiciel de conception (PCB Layout).

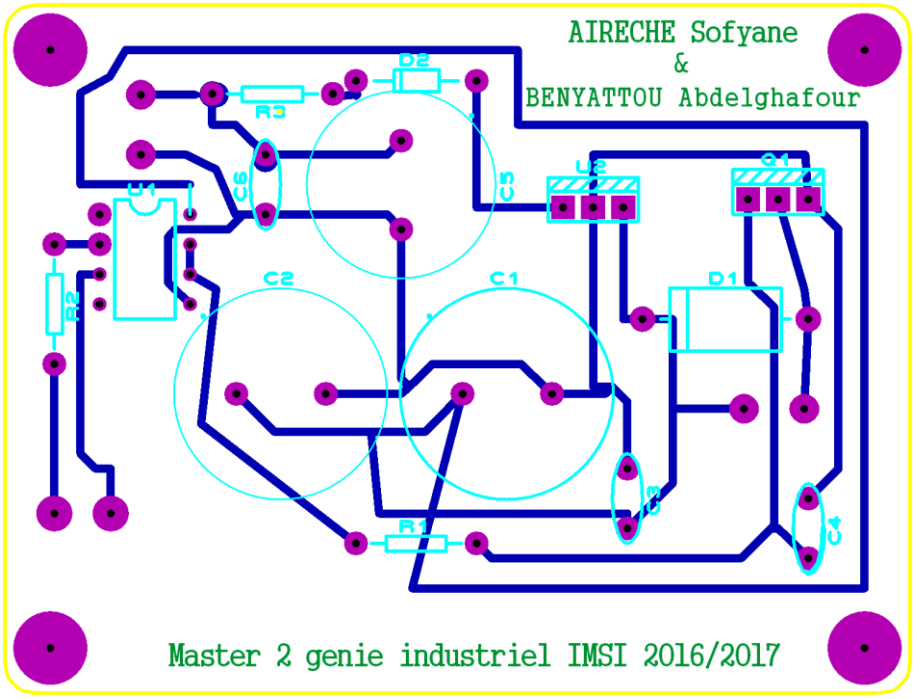

Figure 0.8. *Circuit de puissance sous logiciel de conception Proteus (PCB Layout).*

La **Figure 0.9** illustre le circuit imprimé de l'unité de puissance de notre système.

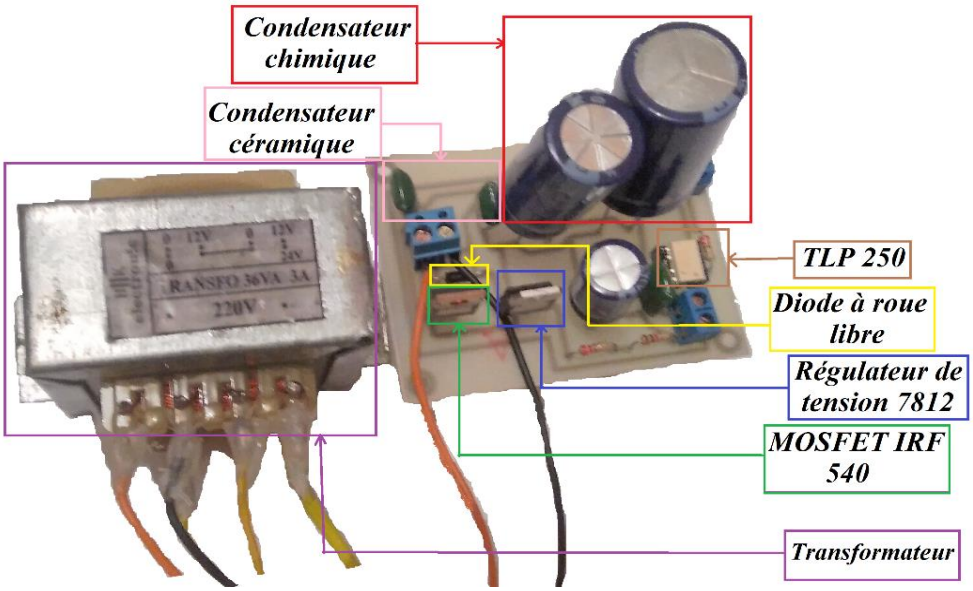

Figure 0.9. *Unité de puissance utilisée pour la commande de système.*

II.2.4.1.1 MOSFET IRF 540N

Joue le rôle d'un interrupteur, il est commandé par une tension entre 0 et 5 V.

II.2.4.1.2 Diode à roue libre

Cette diode protège la pompe contre le routeur de courant.

II.2.4.1.3 Capacité chimique et céramique

L'utilisation de la tension sortie de transformateur nécessite d'un condensateur de type chimique pour le filtrage de tension reçu du transformateur (2200 µF, 25 V), son davantage c'est leur plage de capacité est très large. Le signal de commande est aussi besoin des capacités céramiques (100 nF). L'avantage de ce type de condensateur est ne possède pas une polarité, son rôle est aussi le filtrage des impulsions soudaine.

II.2.4.1.4 Optocoupleur

On a utilisé TLP250 Toshiba, il isole entre le circuit de commande et le circuit de puissance, a un rôle de protection de la partie de commande (Arduino).

II.2.4.1.5 Transformateur

L'utilisation de transformateur nous permettre de transformer la source de tension 220 Volt alternatif en une tension continue favorable nécessaire (12 V) pour le fonctionnement de la pompe, son rôle essentiel est le redressement de signal. Nous avons utilisé dans notre projet un transformateur abaisseur de caractéristique 12V et 3 ampères.

II.2.4.1.6 Pont de diode

A pour objectif le redressement de la tension d'entrée (convertis la tension alternative en une tension continue).

II.2.4.1.7 Régulateur de tension 7812

Pour bien fixée la tension au niveau de 12 volts, c'est à dire ce régulateur à un rôle de stabilisation de tension (les tensions supérieures à 12 volts va se dissipées sous forme de chaleur)

II.2.4.2 Circuit de commande

Englobe tous les composant et les équipements qui nous permettre à commander, contrôler ou bien à réguler notre le système automatiquement à l'aide d'un algorithme ou programme bien définis (**Figure 0.10**).

II.2.4.2.1 La carte Arduino

Nous avons bien détaillé et présenté cette carte dans le chapitre I. C'est l'élément capitale dans toutes les étapes de réalisation de notre projet (l'étalonnage de capteur, identification de système, implémentation du régulateur, … etc.), elle est considérée comme une carte d'acquisition des données, et un microcontrôleur qui comporte le programme de régulation

reçu via le logiciel Matlab, donc c'est elle qui joue le rôle d'un régulateur dans notre application.

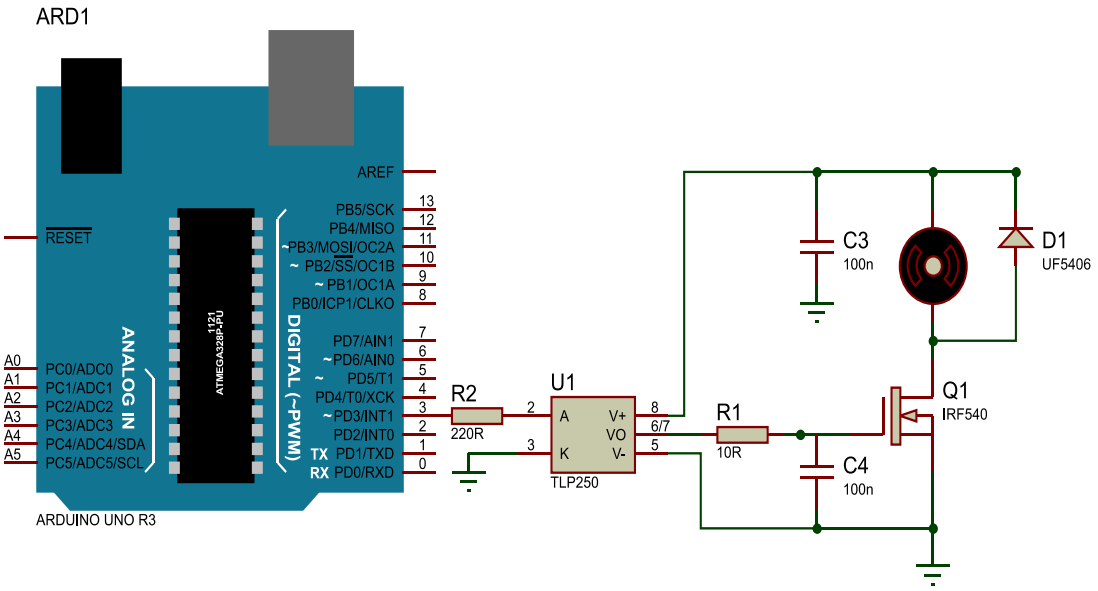

Figure 0.10. *Schéma d'unité de commande de la pompe sous logiciel Proteus.*

II.2.4.2.2 Ordinateur

Un ordinateur muni d'un Logiciel Matlab-Simulink, on peut entrer des consignes au système, et relever des résultats en temps réel comme des graphes, les mesures … etc, grâce à son interface graphique.

II.3 Identification de système (partie théorique)

L'identification c'est l'opération de détermination des caractéristiques dynamiques d'un procédé (système) dont la connaissance est nécessaire pour la conception et la mise en œuvre d'un système performant de régulation (LANDAU, 1993).

Autrement dit, l'identification est une approche expérimentale pour la détermination du modèle dynamique d'un système.

II.3.1 Procédure d'identification d'un système

L'approche de l'identification peut être décomposée en cinq étapes. La **Figure II.11** montre ces cinq étapes.

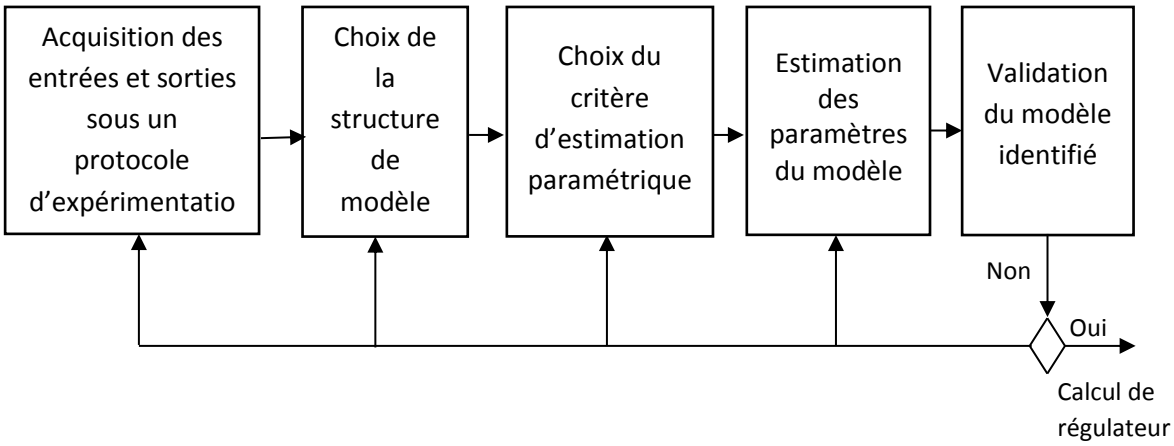

Figure 0.11. *Procédure d'identification d'un modèle de système.*

II.3.1.1 Acquisition des entrées sorties sous un protocole d'expérimentation

Durant cette phase, des mesures sont effectuées sur les variables sensées caractériser le système, ces variables peuvent être des variables externes qui agissent sur le système, des variables internes qui introduisent l'état du système, ou la réponse du système. Il existe souvent des perturbations non mesurables qui agissent sur le système rendant plus difficile sa modélisation (LANDAU, 1993).

II.3.1.2 Choix de la structure de modèle

Il s'agit de définir d'une façon formelle la relation expliquant le fonctionnement du système. Cette relation correspond à une famille de fonctions mathématiques dont une seule correspond au modèle recherché (LANDAU, 1993).

II.3.1.3 Choix du critère d'estimation paramétrique

C'est le choix de la fonction objectif (fonction coût) dont l'optimisation (minimisation) permet de déterminer la structure du modèle de façon unique. Ce critère est en fonction de l'écart entre la sortie du système et celle du modèle (LANDAU, 1993).

II.3.1.4 Estimation des paramètres du modèle

Elle s'agit alors de trouver la valeur des paramètres permettant la satisfaction d'un critère de performance donné (optimisation de la fonction objectif) (LANDAU, 1993).

II.3.1.5 Validation du modèle

C'est une procédure qui permet d'évaluer l'exactitude (ou fidélité) du modèle. Pendant cette phase, Le modèle est testé avec des données non utilisées pendant la phase d'identification (LANDAU, 1993).

Une opération d'identification complète doit nécessairement comporte les cinq phases indiquées plus haut. Les méthodes spécifiques utilisées dans chaque phase dépendent du type de modèle recherché (paramétrique ou non paramétrique, continu ou échantillonné).

L'identification doit comprendre l'actionneur, le procédé et le capteur. En effet, il est impossible d'exciter le procédé sans son actionneur et il est également impossible de mesurer sa sortie sans un capteur. De plus, il n'est pas nécessaire de connaitre la fonction de transfert de chacun des éléments pris individuellement pour faire la commande de l'ensemble sauf pour remettre en cause la conception de l'ensemble**.**

II.3.2 Choix de signal d'excitation

Afin de pouvoir faire l'identification d'un procédé, le signal d'entrée doit être très riche en fréquences. Les signaux les plus couramment utilisés sont (POMERLEAU , 1997) :

- L'échelon unité,
- Les essais sinusoïdaux.
- La séquence pseudo-aléatoire.

Notre système utilise un signal d'excitation à un échelon, car il est le plus convenable pour un système dans la régulation d'un niveau.

II.3.3 Choix du modèle d'identification utilisé

L'objectif de l'identification est de trouver les paramètres du système d'un modèle entrée-sortie. Mais, en pratique, la sortie mesurée est généralement bruitée. Cela est dû soit à l'effet des perturbations aléatoires agissant à différents endroits du procédé, soit à des bruits de mesure.

Ces perturbations sont à caractères aléatoire sont modélisées souvent par le model *ARX* et le modèle *ARMAX*. En temps discret La forme générale de la sortie est donnée par (**Figure 0.12)** :

$$y(k) = G(z)u(k) + H(z)e(k) \qquad \text{(II.16)}$$

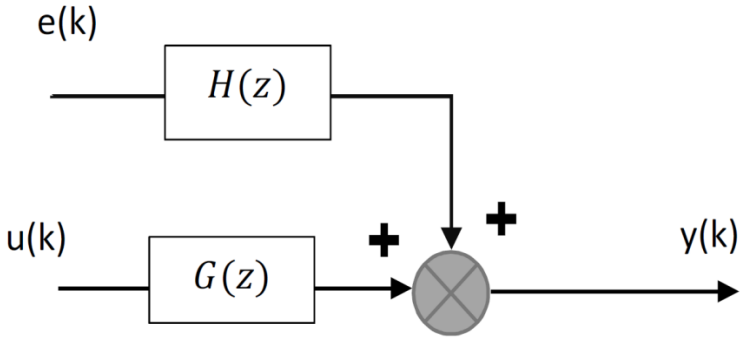

Figure 0.12. *Forme générale « procédé+ bruit ».*

Avec :

- y(k) : les séquences sorties
- u(k) : les séquences entrées
- e(k) : les séquences bruits

Ces deux modèles sont largement employés en automatique et très connus. On note qu'il n'y a pas pratiquement, une structure unique concernant le modèle général « procédé + perturbation ».

II.3.3.1 Modèle ARX (Auto Régressive à variable eXogène)

Le modèle ARX décrit par (LJUNG, 2008):

$$y(t) + a_1 y(t-1) + \cdots + a_{na} y(t-n_a) = b_1 u(t-1) + \cdots + b_{nb} u(t-n_b) + e(t) \quad \text{(II.17)}$$

Avec :

- y(t) sortie du system au différents instants t
- u(t) entrée de système au différents instants t
- e(t) une séquences de nombres aléatoires indépendantes.(bruit blanc)

Ce modèle est généralement représenté sous la forme compacte :

$$A(z^{-1})Y(z) = B(z^{-1})U(z) + E(z) \quad \text{(II.18)}$$

Avec :

$$A(Z^1) = 1 + a_1 Z^{-1} + \cdots + a_{na} Z^{-na} \quad \text{(II.19)}$$

$$B(Z^1) = 1 + b_1 Z^{-1} + \cdots + b_{nb} Z^{-nb} \quad \text{(II.20)}$$

La **Figure 0.13** présente le modèle ARX

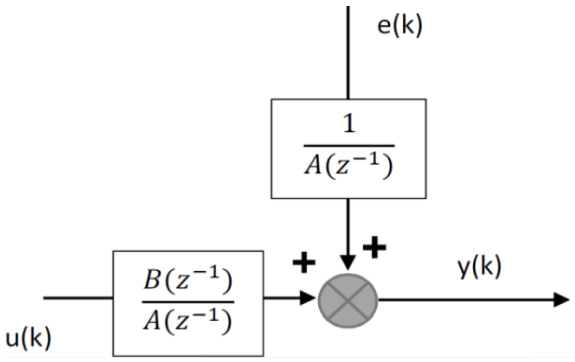

Figure 0.13. *Modèle de structure ARX.*

Ce Modèle est le plus simple, il donne souvent de bons résultats. Son seul inconvénient est le traitement du bruit qui est soumis à la même dynamique que l'entrée.

II.3.3.2 Modèle ARMAX (Auto régressive à moyenne ajustée et variable exogène)

Le Modèle ARMAX est un Modèle entrée-sortie de la forme générale (LJUNG, 2008):

$$\begin{aligned} y(t) + a_1 y(t-1) + \cdots + a_{na} y(t-n_a) \\ = b_1 u(t-1) + \cdots + b_{nb} u(t-n_b) + e(t) + c_1 e(t-1) \\ + \cdots + c_{nc} e(t-n_c) \end{aligned} \quad \text{(II.21)}$$

Sous la forme compacte :

$$A(z^{-1})Y(z) = B(z^{-1})U(z) + C(z^{-1})E(z) \quad \text{(II.22)}$$

Avec :

$$A(Z^{-1}) = 1 + a_1 Z^{-1} + \cdots + a_{na} Z^{-na} \quad \text{(II.23)}$$

$$\mathrm{B}(\mathrm{Z}^{-1}) = 1 + \mathrm{b}_1 \mathrm{Z}^{-1} + \cdots + \mathrm{b}_{\mathrm{nb}} \mathrm{Z}^{-\mathrm{nb}} \quad \text{(II.24)}$$

$$C(Z^{-1}) = 1 + C_1 Z^{-1} + \cdots + C_{nc} Z^{-nc} \quad \text{(II.25)}$$

La **Figure 0.14** présente le modèle ARMAX

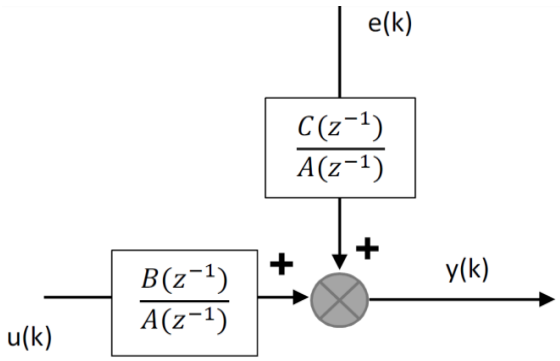

Figure 0.14. *Modèle de structure ARMAX.*

II.3.3.3 Identification basée sur l'erreur de sortie

Il existe autre modèle d'identification basé sur l'erreur de sortie. Le principe de cette méthode d'identification est illustré en **Figure 0.15**. Le modèle est une fonction de n paramètres θ_i, i variant de 1 à n. Il s'agit alors de déterminer les paramètres θ_i tels que le critère soit minimum. Le critère est en général choisi de la forme (GONZALO, 2009):

$$J = \sum \varepsilon^2 \tag{II.26}$$

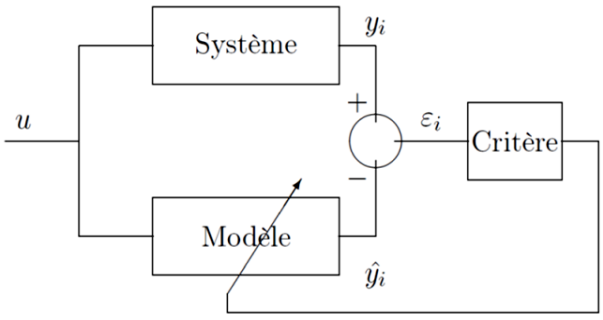

Figure 0.15. *Principe d'identification fondé sur l'erreur de sortie.*

Avantages de cette méthode :

- Pas d'hypothèse sur la forme du modèle : non linéaire
- Adapté à la recherche de paramètres physiques si modèle de connaissances (modèle continu)

– L'inconvénient majeur est qu'aucun de ces algorithmes n'est capable de garantir que le résultat est réellement l'optimum.

II.4 Partie pratique

II.4.1 Présentation de l'étape d'identification avec Matlab

Cette étape est constituée de deux parties, la première est assuré par l'environnement *Simulink* et le package *ArduinoIO* pour l'envoie et l'acquisition des données. La deuxième partie est assurée par l'outil 'System identification' sous Matlab (**Figure 0.16**).

Pour plus de détails sur l'identification voir l'annexe A.

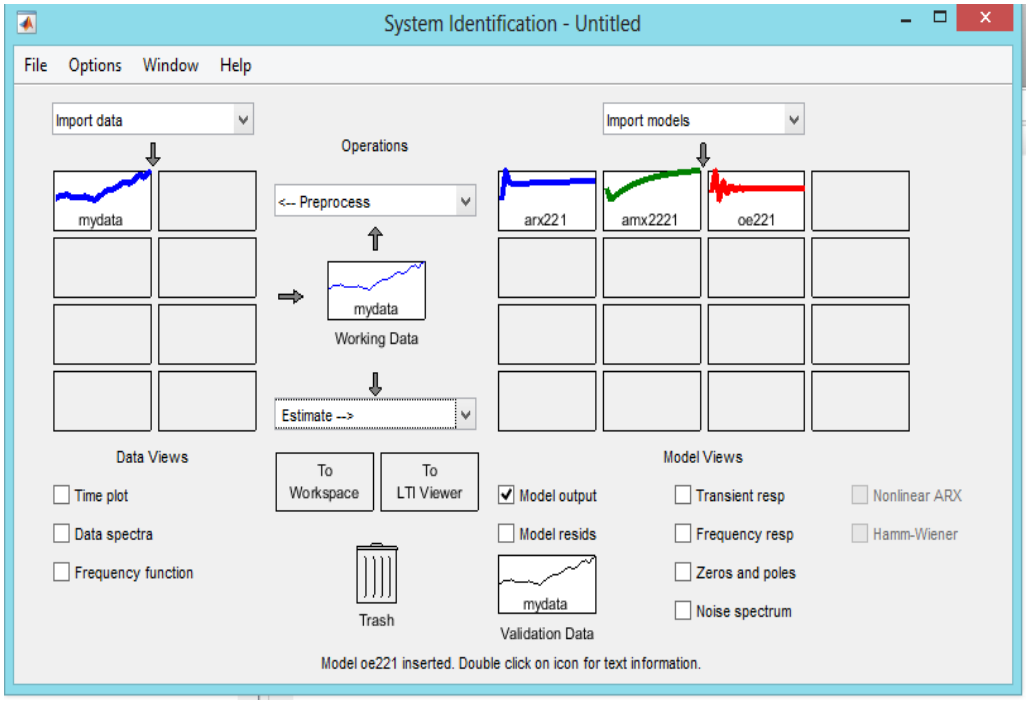

Figure 0.16. *Interface de l'outil d'identification sous Matlab « Matlab identification toolbox ».*

II.4.2 Acquisition de la réponse indicielle du système

Le modèle Simulink permettant de réaliser l'acquisition de la réponse du système à un échelon de tension, la **Figure 0.17** donne les différents blocs de Simulink utilisé pour le modèle d'identification.

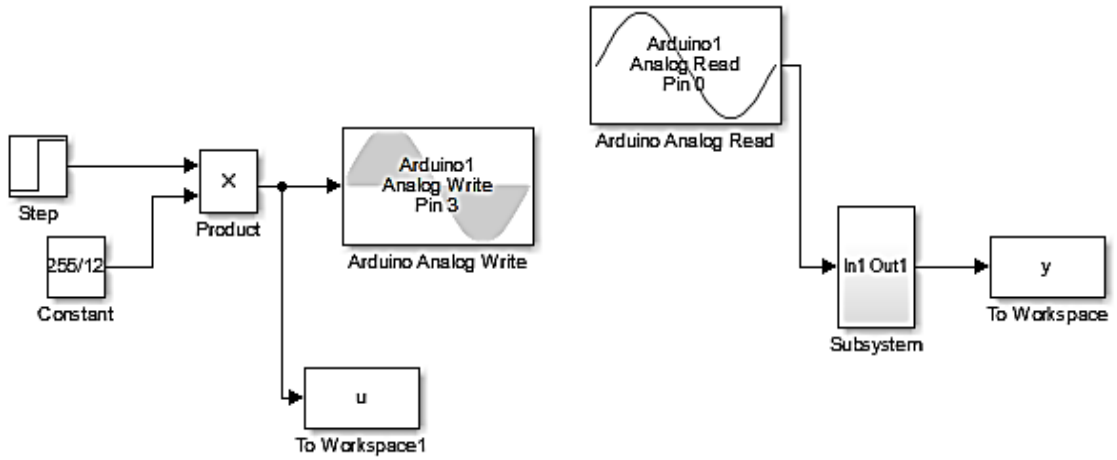

Figure 0.17. *Modèle Simulink pour l'identification.*

Après cette opération on obtient des réponses d'entrée et de sortie du système en temps réel, qui nous a permet de trouver la fonction de transfert de notre système.

II.4.3 Les résultats obtenus de l'identification en temps réel

La **Figure 0.18** suivante illustre le signal de tension à un échelon de 130/255 PWM.

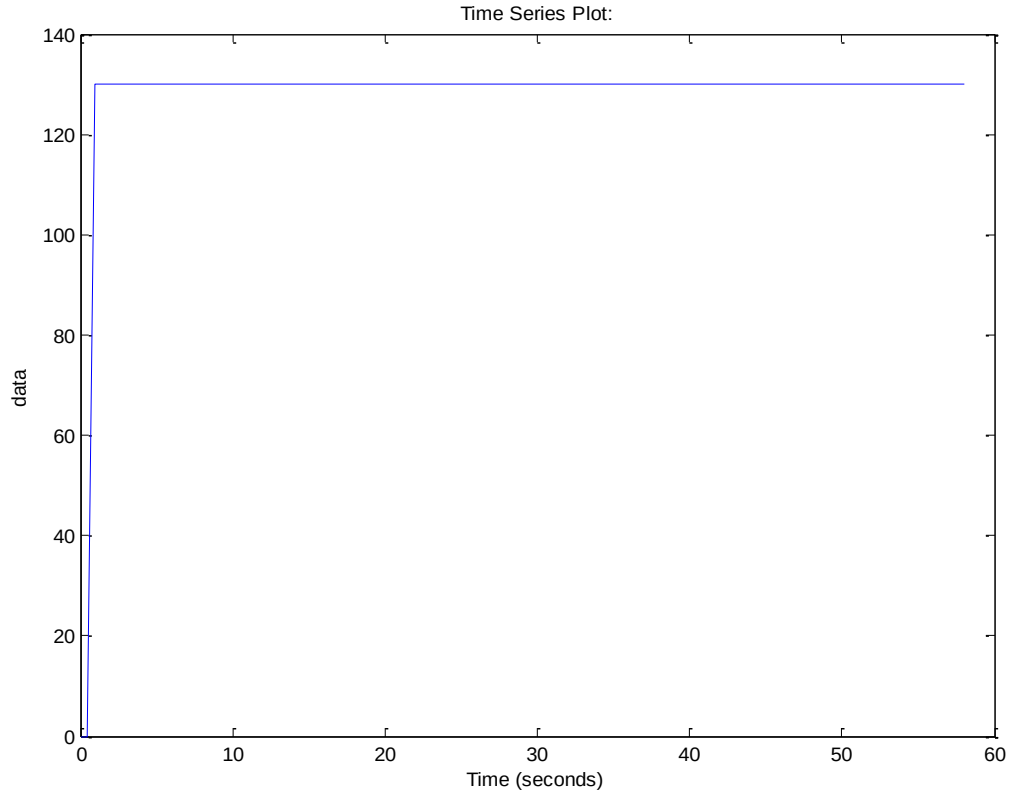

Figure 0.18. *L'échelon de de tension.*

Et la **Figure 0.19** ci-dessous montre la sortie d'identification de notre système, et il est clair qu'elle est perturbée à cause des bruits de mesure et les bruits de l'environnement, même les erreurs de capteur a une influence sur cette sortie :

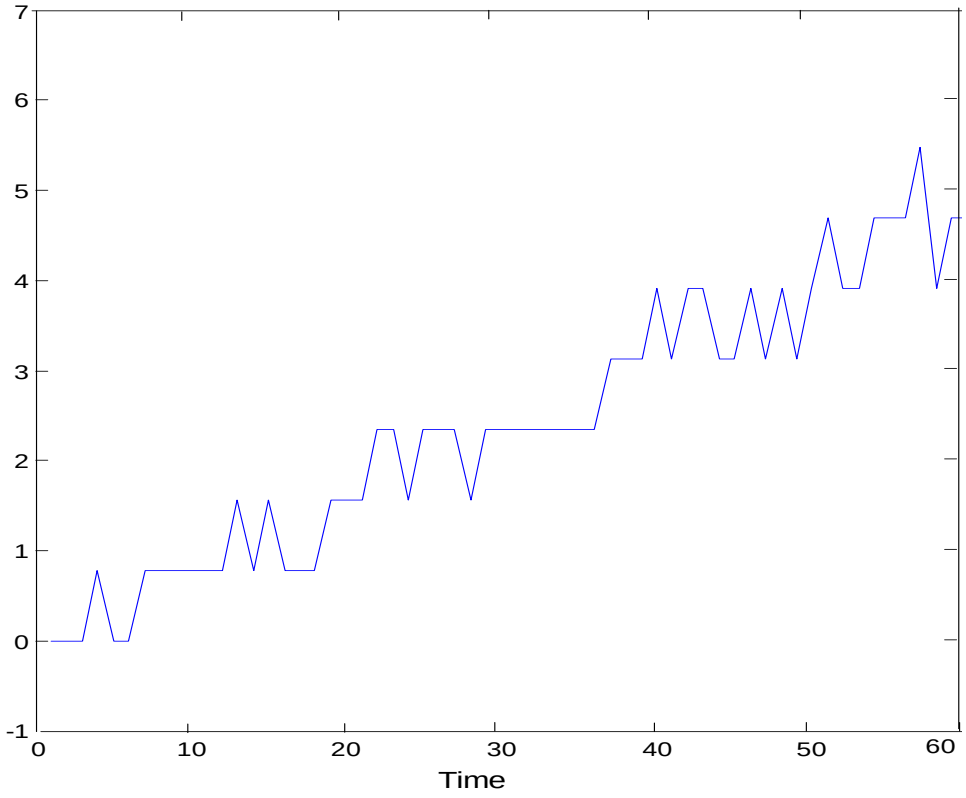

Figure 0.19. *La repense de la sortie de système à un échelon avant le filtrage.*

L'autre (**Figure 0.20)** donne la réponse de système en temps réel avec le signal d'excitation à un échelon après le filtrage avec un filtre passe bas numérique de premier ordre sous Simulink, **l'équation II.27** montre la formule de filtre *f(z)* utilisée dans notre système :

$$f(z) = 0.9/\left((1 - 0.9) \cdot z^{-1}\right) \qquad \text{(II.27)}$$

Cette réponse est en augmentation progressive depuis de 10 secondes, et ça nos indique que notre système est un système de type intégrateur.

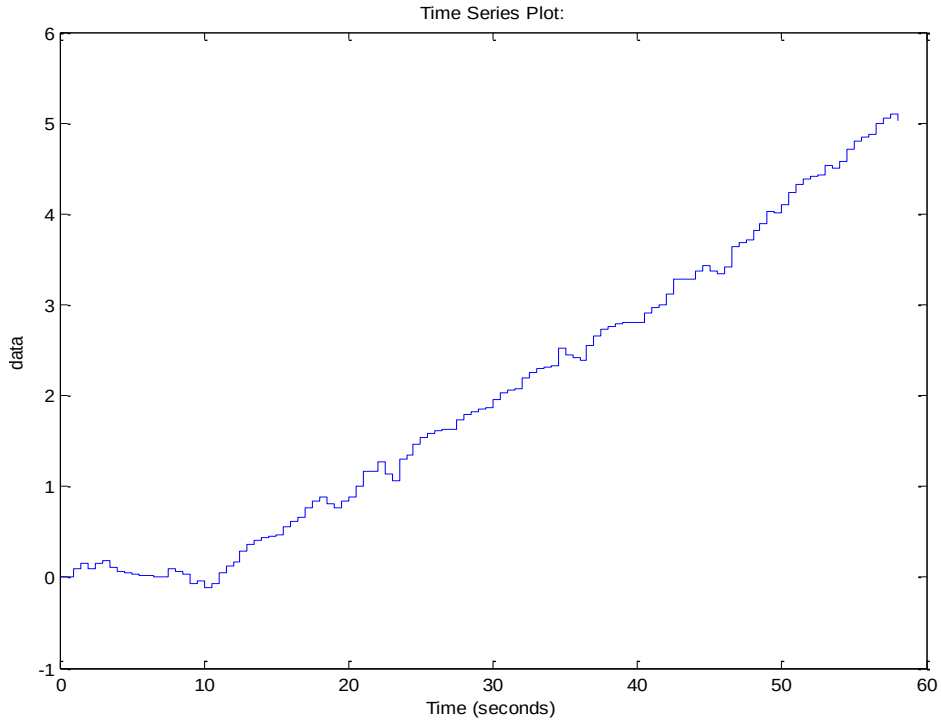

Figure 0.20. *La repense de la sortie de système à un échelon après le filtrage.*

II.4.4 Détermination de la fonction de transfert G(z)

Après avoir déterminé la réponse du système, on passe à la détermination de la fonction de transfert G(z) à l'aide de l'outil 'System identification Toolbox'.

Pour cela on applique les modèles présentés ci-dessus dans le sous-**titre II.3.3** sous Matlab, et on va comparer les modèles obtenu afin de choisis le meilleur modèle et le plus proche de la réponse de notre système (**Figure 0.20**).

II.4.4.1 Résultats d'identification des procédés

On a identifié trois fois selon les trois modèles décris au plus haut dans le sous-**titre II.3.3**, donc on a trouvé trois résultats d'identification, qui sont :

II.4.4.1.1 Identification en ARX

Tout d'abord, on a choisi le modèle ARX, Ce qui a donné les polynômes suivants :

$$\begin{cases} A(z) = A(z) = 1 - 1.087\, z^{-1} + 0.07946\, z^{-2} \\ B(z) = 0.0003729\, z^{-1} - 0.0001865\, z^{-2} \end{cases} \quad \text{(II.28)}$$

II.4.4.1.2 Identification en ARMAX

Le type du modèle en sélectionnant le type ARMAX, donne les polynômes suivants :

$$\begin{cases} A(z) = 1 - 1.884\, z^{-1} + 0.8831\, z^{-2} \\ B(z) = -0.0001893\, z^{-1} + 0.0002239\, z^{-2} \\ C(z) = 1 - 0.8285\, z^{-1} - 0.1715\, z^{-2} \end{cases} \quad \text{(II.29)}$$

II.4.4.1.3 Identification en erreur de sortie (OE)

Puis on a changé le type du modèle en sélectionnant le type *OE*, on a obtenu les polynômes suivants :

$$\begin{cases} B(z) = 0.001316\, z^{-1} - 0.0008937\, z^{-2} \\ F(z) = 1 - 0.3546\, z^{-1} - 0.6454\, z^{-2} \end{cases} \quad \text{(II.30)}$$

Les résultats graphiques des trois modèles sont donnés par la **Figure 0.21** ci-dessous :

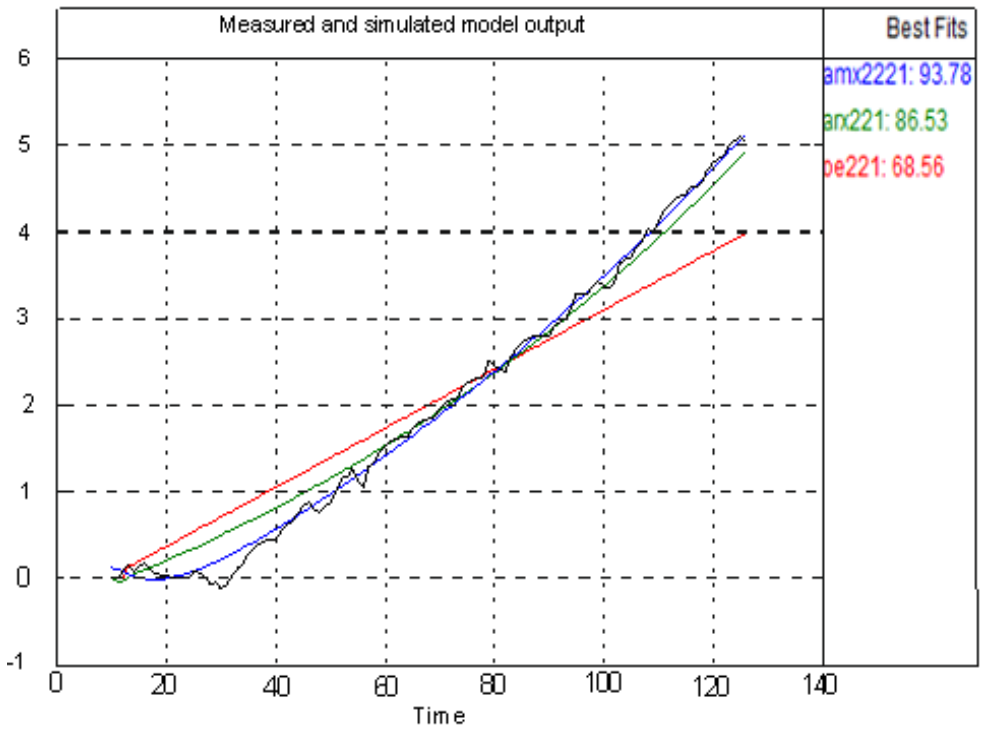

Figure 0.21. *Comparaison entre les trois modèles et la sortie de système.*

À partir du résultat qui nous avons obtenus par l'outil du 'System identification' de Matlab "*model output*" on a les pourcentages suivent :

- Le modèle ARMX suit la repense du system 93.78 %,

- Le modèle ARX suit la repense du system 86.53 %,
- Le model OE suit la repense du system 68.56%,

D'après les pourcentages donnés par l'outil d'identification, il est mieux d'adopter le model ARMAX parce qu'il a donné des résultats satisfaisants. Alors la fonction de transfert du système (en z) est :

$$G(z) = \frac{-0.00018932\, z^{-1}(1 - 1.182z^{-1})}{(1 - 1.006z^{-1})\,(1 - 0.8775z^{-1})} \qquad \text{(II.31)}$$

II.5 Conclusions

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 les chapitres suivants 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 des bonnes performances.

Chapitre III

Régulateur Classique PID

II.6 Introduction

Dans la plupart des processus industriels, il est indispensable de maîtriser certains paramètres physiques. En automatique lorsque l'on souhaite atteindre une certaine vitesse, température, position, angle..., il est donc très souvent nécessaire d'avoir recours à un asservissement, c'est à dire un système capable d'atteindre et de maintenir une consigne en utilisant une mesure. Il s'agit donc d'un système bouclé, dont il reste à déterminer la fonction permettant de corriger la commande en fonction de la consigne initiale et de l'erreur mesurée. Parmi ces méthodes le régulateur PID, le régulateur floue, RST … etc.

II.7 Les systèmes asservis linéaires continus

Les systèmes asservis linéaires découlent d'un modèle physique continu défini par des équations intégrales et différentielles, à coefficients constants, en fonction du temps, qui régissent les grandeurs présentes dans le modèle (OULD SIDI Mohamed & SERSOUB, 2015).

II.8 Les systèmes asservis linéaires discrets :

La généralisation de l'utilisation des calculateurs a permis leur exploitation comme structures de commande numérique pour les systèmes dynamiques asservis et ceci par le biais de l'implantation d'algorithmes gérés par le calculateur remplaçant ainsi avantageusement les régulateurs analogiques de commande. L'exploitation des calculateurs pour la commande des processus est plus flexible et permet d'obtenir des performances meilleures que celles obtenues par des commandes analogiques (OULD SIDI Mohamed & SERSOUB, 2015).

II.9 Définition de la régulation

La régulation est l'action de contrôler automatiquement une grandeur physique (niveau, température, pression, débit.), en temps réel, de telle sorte que celle-ci garde constamment sa valeur ou reste proche de la valeur désirée malgré les perturbations de procédé (MIMOUNI & BENGOUFA, 2005).

L'objectif global de la régulation peut se résumer par ces trois mots clefs :

- Mesurer.
- Comparer.
- Corriger.

II.10 Régulateur classique PID (théorie)

Le régulateur *PID*, appelé aussi correcteur PID (Proportionnel, Intégrateur, Dérivateur ou proportionnel, intégral, dérivé) est un système de contrôle, permettant d'effectuer un asservissement en boucle fermée d'un système industriel ou « procédé ». C'est le

régulateur le plus utilisé dans l'industrie, et il permet de régler un grand nombre de grandeurs physiques.

II.10.1 Principe générale

Un correcteur est un algorithme de calcul qui délivre un signal de commande à partir de la différence entre la consigne et la mesure.

Le correcteur PID agit de trois manières

1) Action **proportionnelle** : l'erreur est multipliée par un gain G ;
2) Action **intégrale** : l'erreur est intégrée et divisée par un gain T_i ;
3) Action **dérivée** : l'erreur est dérivée et multipliée par un gain T_d;

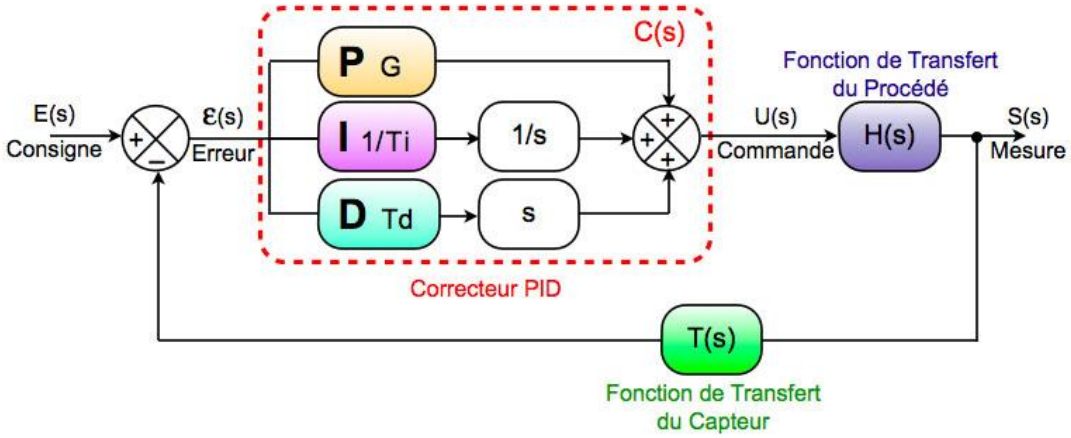

Figure 0.1. *La structure d'un régulateur PID parallèle dans une boucle de régulation.*

La fonction de transfert exprimée dans le domaine de Laplace **:**

$$C(s) = G + (1/s) \cdot (1/s) + T_d \cdot s \qquad \text{(III.32)}$$

II.10.2 Aspects fonctionnels

Le PID parfait n'existe pas, tout est une question de compromis. Certaines applications autoriseront un dépassement afin d'améliorer le temps de stabilisation, alors que d'autres ne l'autoriseront pas (exemple, contrôler un stylo pour écrire sur une feuille par une CNC machine (Computer Numerical Control). S'il y a dépassement dans le PID, le stylo traversera la feuille). Tout dépend donc du cahier des charges. Chacun des coefficients à un rôle à jouer sur la réponse à une consigne :

1) L'erreur statique, c'est l'erreur finale une fois que le système est stable. Cette erreur doit être nulle. Pour diminuer l'erreur statique, il faut augmenter K_p et K_i.
2) Le dépassement, c'est le rapport entre le premier pic et la consigne. Ce dépassement diminue si K_p ou K_i diminuent ou si K_d augmente.
3) Le temps de montée correspond au temps qu'il faut pour arriver ou dépasser à la consigne. Le temps de montée diminue si K_p ou K_i augmentent ou si K_d diminue.
4) Le temps de stabilisation, c'est le temps qu'il faut pour que le signal commette une erreur inférieure à 5% de la consigne. Ce temps de stabilisation diminue quand K_p et K_i augmentent.

	stabilité	**rapidité**	**précision**	**Dépassement**
Si **P** augmente	Diminue	Augmente	Augmente	Augmente
Si **I** augmente	Augmente	Diminue	Pas d'influence	Augmente
Si **D** augmente	Diminue	Augmente	Pas d'influence	Diminue

Tableau 0.1. : *Influence des paramètres P, I et D.*

Les coefficients K_i et K_d dépendent de la fréquence d'échantillonnage du système. En effet, l'intégrateur fera la somme des erreurs au cours du temps. Si on échantillonne deux fois plus vite, on sommera deux fois plus d'échantillons. Du coup, le coefficient K_i devra être divisé par 2. A l'inverse, pour le dérivateur, si on double la fréquence d'échantillonnage, il faudra doubler le coefficient K_d afin de garder les mêmes performances du PID. Plus la fréquence d'échantillonnage est élevée et plus le *PID* sera performant. (En effet, plus on échantillonne souvent et plus l'intégration et la dérivée seront précises), donc il faut bien choisir le temps d'échantillonnage.

Où :

$$K_i = 1/T_i \tag{III.33}$$

$$K_d = 1/T_d \tag{III.34}$$

II.10.3 Le choix de régulateur (correcteur)

L'action dérivée est utilisée dans le cas de variables non bruitées, car la dérivation est très sensible au bruit additive au signal, variable soumise à de nombreuses perturbations, à cause de ce dernier on n'utilisera pas ce type de la correction dans notre système qui ai une erreur de mesure remarquable. Pour un régulateur intégral pur, le régime dynamique est relativement long. D'un autre côté, le régulateur proportionnel réagit immédiatement aux écarts de réglage mais il n'est pas en mesure de supprimer totalement l'erreur statique. La combinaison des actions proportionnelle et intégrale permet d'associer l'avantage du régulateur P, C'est-à-dire la réaction rapide à un écart de réglage, à l'avantage du régulateur qui est la compensation exacte de la grandeur pilote, donc on va utiliser un correcteur *PI* (CHRISTOPHE, 2007).

II.10.3.1 Régulation analogique avec l'action PI

Le contrôleur PI a élaboré une commande qui peut être donnée par la relation suivante en temps continu :

$$u(t) = K_P \cdot \left[\varepsilon(t) + K_i \cdot \int_0^t \varepsilon(t) \cdot dt\right] \quad \text{(III.35)}$$

La fonction de transfert du correcteur :

$$C(s) = (K_P \cdot s + K_i)/s \quad \text{(III.36)}$$

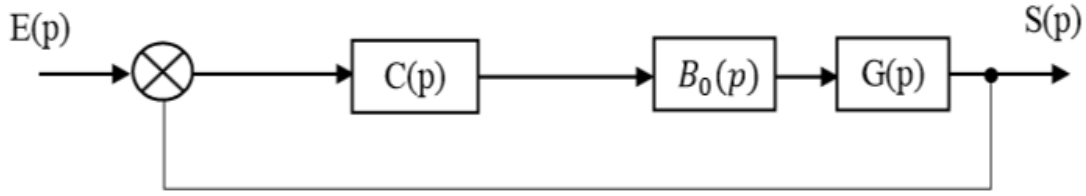

Figure 0.2. *Modèle à temps continu de l'asservissement.*

II.10.3.2 Régulation numérique avec l'action PI

La version de base du régulateur PI numérique résulte de la discrétisation du régulateur PI continu. Parmi Les méthodes qui s'adapte 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.

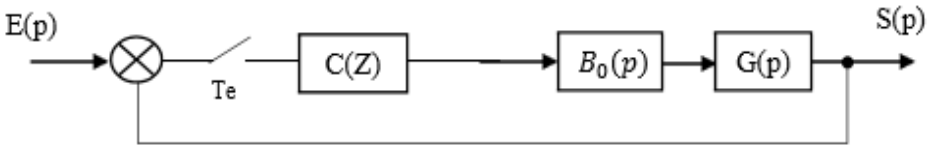

Figure 0.3. *Asservissement continu corrigé numériquement.*

La technique consiste à étudier cet asservissement en temps continu (**Figure 0.4**) puis à rechercher le modèle numérique équivalent au correcteur continu C(p) (BEN ABDENNOUR, BORNE, KSOURI, & M'SAHLI, 2001).

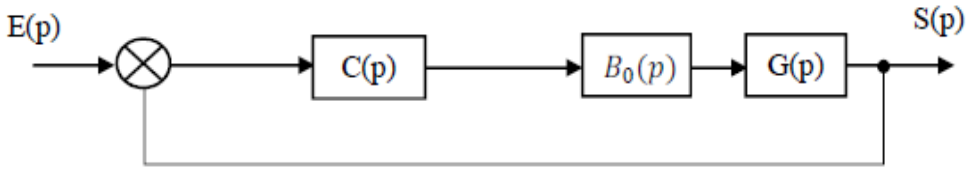

Figure 0.4. *Modèle à temps continu de l'asservissement.*

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 (BEN ABDENNOUR, BORNE, KSOURI, & M'SAHLI, 2001).

II.10.4 La formule du régulateur PI continu après la discrétisation

Pour avoir la formule de régulateur PI numérique, il est nécessaire de passer par l'étape de discrétisation, **l'équation (III.37)** donne la formule de transformation du régulateur *Intégral* de domaine de Laplace vers le domaine discret, par contre **l'équation (III.38)** donne la formule finale du régulateur discret (numérique) :

$$\frac{1}{T_i \cdot s} \rightarrow \frac{T_e}{T_i} \cdot \frac{1}{1 - z^{-1}} \qquad \text{(III.37)}$$

$$C(z^{-1}) = K\left[1 + \frac{T_e}{T_i} \cdot \frac{1}{1 - z^{-1}}\right] \qquad \text{(III.38)}$$

D'où :

- K gain proportionnelle;
- Ti=1/K_i : action intégral;
- Te: le temps d'échantillonnage.

II.10.5 Calcul des paramètres du régulateur

À cause de la complexité de notre système, il est un peu difficile de calculer les paramètres de régulateur PI, par conséquent on a utilisé le logiciel Matlab qui est muni d'un outil de calcul de ces paramètres c'est *'PID TUNER'*, cette application permette de calculer les paramètres de n'importe quel action régulateur (P, PI, PD ou PID).

Il existe une méthode graphique pour déterminer les paramètres de notre régulateur, mais elle est imprécise par rapport à la méthode utilisé par le 'PID Tuner' parce qu'il utilise des algorithmes efficaces pour calculer ces paramètres (**Voir l'annexe B**).

L'interface de l'outil *'PID TUNER'* est donnée par la **Figure 0.5** ci-dessous :

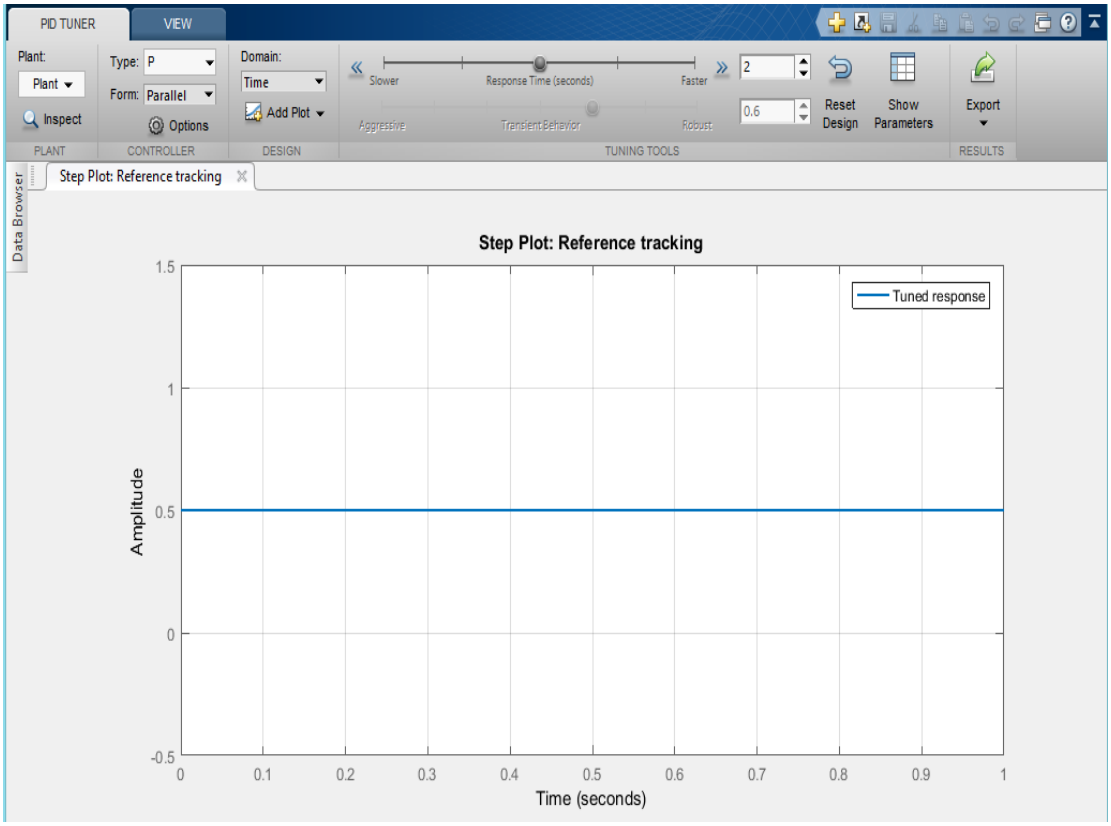

Figure 0.5. *Interface de l'application 'PID Tuner' sous Matlab.*

II.11 Partie pratique

II.11.1 Détermination des paramètres du correcteur

Pour déterminer les paramètres de régulateur on import la fonction de Transfer de notre système obtenu par l'outil d'identification à l'aide l'application '*PID Tuner*' pour trouver les paramètres du correcteur PI.

Les résultats obtenus sont :

- $K_P = 4.315$;
- K_i=0.045;

La **Figure 0.6** montre la réponse de notre système avec les coefficients de régulateur obtenus ci-dessous

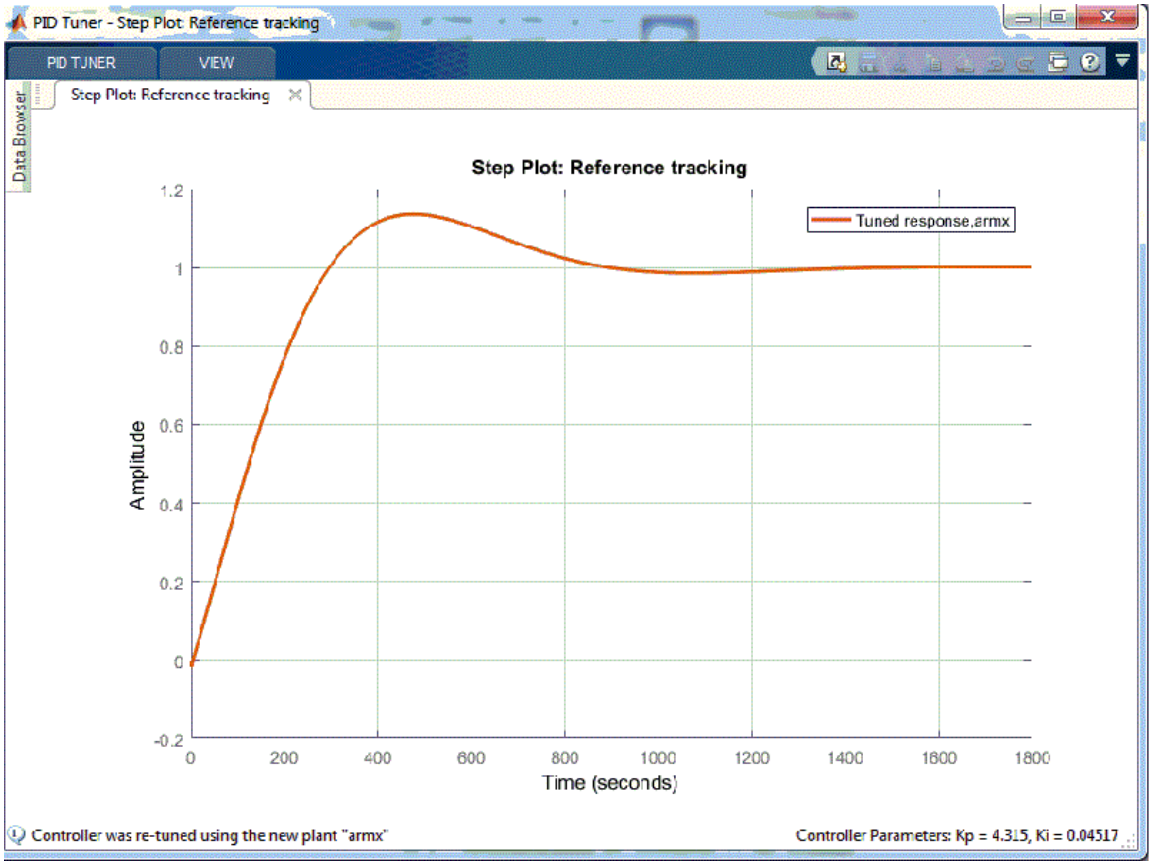

Figure 0.6. *La réponse de système avec le correcteur.*

II.11.2 Implémentation du régulateur PI sur Arduino

On a implémenté un régulateur PI dans la carte Arduino à partir d'un modèle Simulink grâce à la bibliothèque « *Simulink Support Package for Arduino Hardware* ».

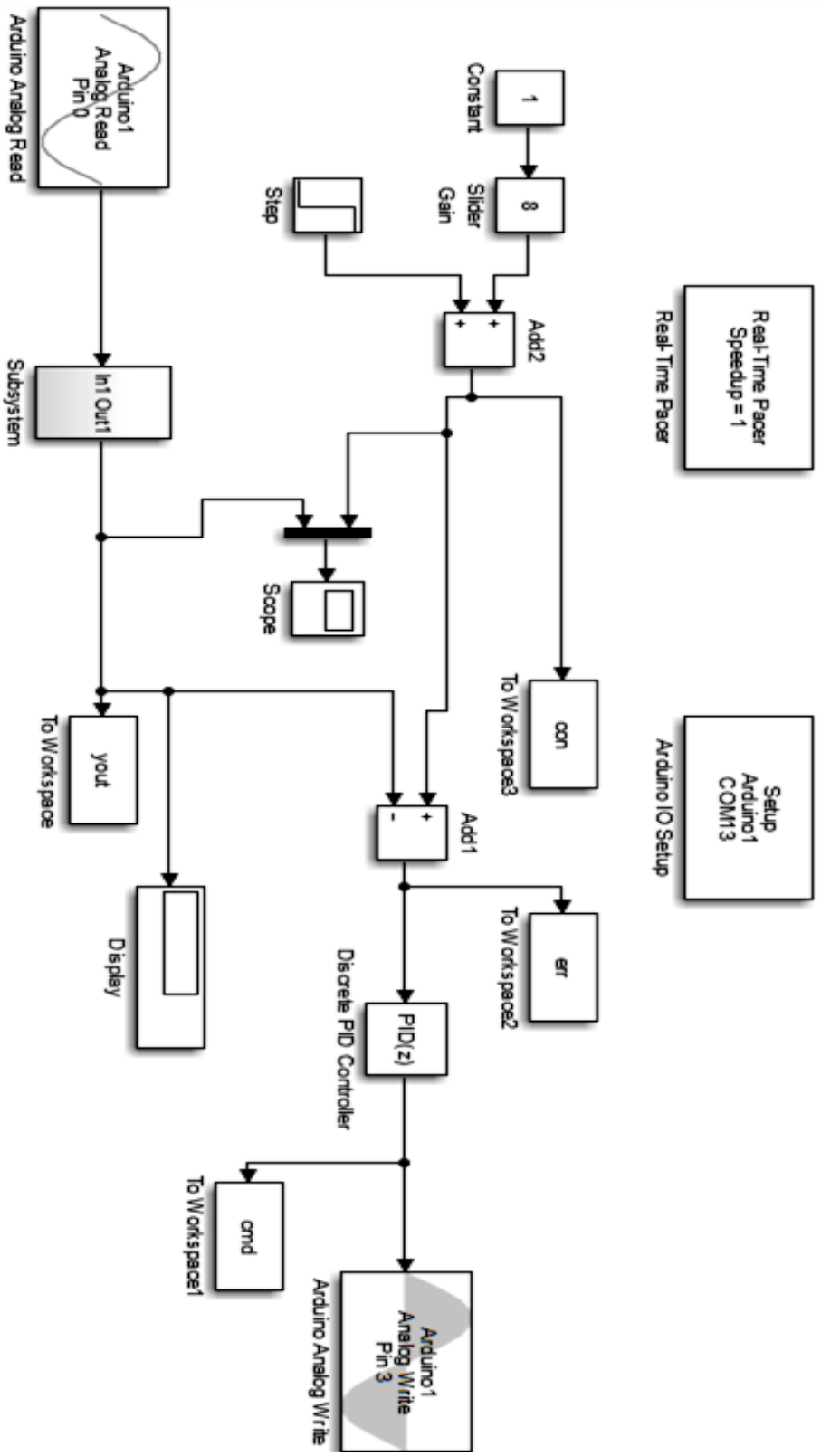

Figure 0.7. *Schémas bloc du correcteur PI numérique sous Simulink adapté à Arduino.*

II.11.3 L'implémentation du régulateur PI sur la carte Arduino

L'implémentation du régulateur PID sur la carte Arduino se fera de la manière suivante :

PI:

Error = Setpoint − Actual

Integral = Integral + (Error∗ dt)

Drive = (Error∗ kP) + (Integral∗ ki)

Previous_error = Error

wait (dt)

GOTO PID

II.11.3.1 Le programme en langage Arduino C

```
Float_delta_erreur=0;
 float somme_erreur = 0; // Somme des erreurs pour l'intgrateur
float kp = 4.315 ; // Coefficient proportionnel
float ki = 0.045 ; // Coefficient intgrateur
float  level, u, e, integral, cmd, ep , derive ;
int consigne ;
 void setup() {
 Serial .begin (9600)  }
void loop()   {
 level=analogRead(0) ;
level=level∗ 0.48828125;
 e=9−level;
somme_erreur=somme_erreur+e∗0.5 ;
 cmd = kp∗ e + Ki∗ somme_erreur;
```

```
analogWrite(6,cmd*(255/12)) ;

ep = e ;

delay(500) ; // periode d'échantillonage

Serial. write(analogRead(0)) ; //envoie de la donnée sur le port série

    }
```

II.11.4 Les résultats expérimentaux de la correction de système avec l'action PI numérique

La figure ci-dessous montre le comportement du système en temps réel avec correction *PI* à une consigne variable.

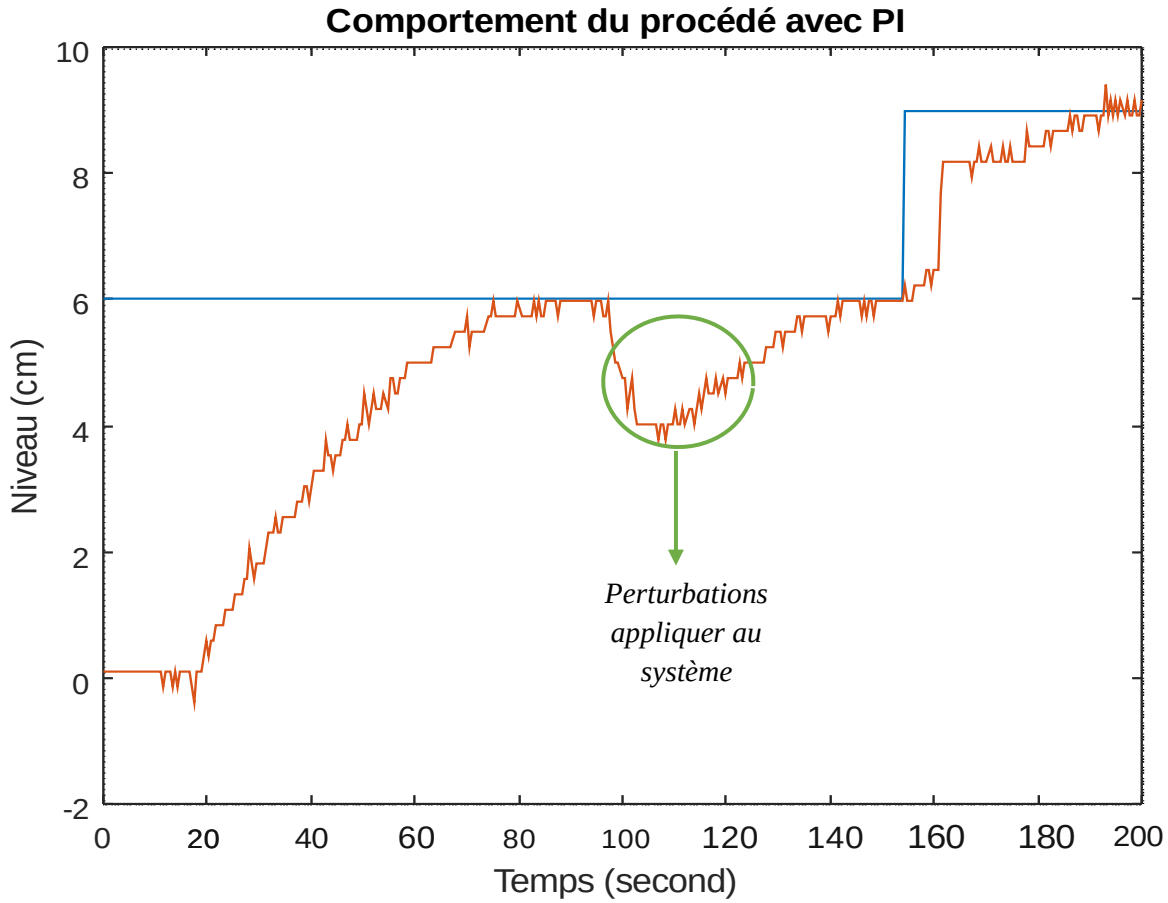

Figure 0.8. *Comportement du procédé avec un régulateur PI (rejet de perturbation et poursuite de consigne).*

II.11.4.1 Interprétation du résultat

Nous avons appliqué deux consignes au système afin de visualiser son comportement de système selon le tableau suivant :

T(s)	0	155
E(cm)	6	9

Tableau 0.2. *Les consignes imposées.*

Nous remarquons que la sortie du système suit l'entrée avec un temps de réponse de 75.5s (un intervalle entre 0 et 20s le capteur n'été pas encours dans son point de fonctionnement). A l'instant 98s on a appliqué une perturbation qui a entrainé une descente de 2cm, le système a pu rejeter cette perturbation grâce à l'action Intégrale du correcteur, mais le temps qu'il a fallu pour le rejet est de 44s.

II.11.4.2 Discussion de résultat

Le test précédent (**Figure 0.8**) a pour objet l'étude du comportement de contrôleur considéré. Dans des conditions de fonctionnement nominal, on peut remarquer que la consigne de niveau est suivie par la sortie du système (mauvaise précision). Le rejet des perturbations se fait d'une manière lente. Nous remarquons aussi que l'influence de perturbation est assez importante, une petite perturbation a couté une baisse supérieure à 10%, cette valeur peut être acceptable pour certaines applications tel que les barrages d'eau ou autre mais elle pourrait être dangereuse pour autre application comme la régulation de niveau des produits chimiques. De plus, la réaction du système pour le rejet de cette perturbation est peu lente, ce qui est non acceptable dans certains domaines. Pour surmonter ce problème, il faut penser à une autre technique de commande permettant d'améliorer le comportement de notre système, c'est ce qu'on va voir dans le chapitre suivant.

II.12 Conclusion

Dans ce chapitre, nous avons présenté, comment obtenir les paramètres régulateur PI et nous avons transformé Arduino d'une carte d'acquisition a un régulateur réel ce qui permis de réaliser un régulateur numérique avec un prix bas. Le régulateur PI a donné des résultats satisfaisants, mais la compensation des pertes se fait un peu d'une manière lente. Alors il faut trouver un régulateur qui peut compenser les pertes rapidement dans le temps. Dans le chapitre suivant on va proposer un correcteur qui peut diminuer le temps de compensation, et qui va donner des performances mieux qu'un PI.

Chapitre IV

Régulateur floue

II.13 Introduction

la logique floue suscite acculement un intérêt général de la part des chercheur des ingénieur et des industriels, mes plus génialement de la part de tous ceux éprouvent le besoins de formaliser les méthodes empirique, de généraliser des raisonnement naturel, d'automatiser la prise de discision de leur domaine de construire des système artificiel effectuaient les tache habituellement prises en charge par les humain , même en absence du modèle mathématique il permet de donner des résultats satisfaisants et robuste que ce soit pour la poursuite ou le rejet de perturbation.

II.14 Historique

Les prémisses de la logique floue sont apparues avant les années 1940, avec les premières approches, par des chercheurs américains, du concept d'incertitude. Il a fallu attendre 1965, pour que le concept de sous ensemble floue soit proposé par L. A. Zadeh, automaticien de réputation internationale, professeur à l'université de Berkeley en Californie, qui a contribué à la modélisation de phénomène sous forme floue, en vue de pallier les limitations dues aux incertitudes des modèles classiques à équation différentielle. En 1974, M. Mamdani expérimentait la théorie énoncée par Zadeh sur une chaudière à vapeur, matériel dont on connait la complexité, introduisant ainsi la commande floue dans la régulation d'un processus industriel. Plusieurs applications ont alors vu le jour, pour des systèmes parfois très complexes, telle que la régulation des fours de cimenterie.

II.15 Théorie des ensembles flous

II.15.1 Notion d'appartenance partielle

Dans la théorie des ensembles conventionnels, un élément appartient ou n'appartient pas à un ensemble. Cette notion essentielle ne permet cependant pas de rendre compte de situations pourtant simples et rencontrées fréquemment. Parmi des fruits, il est facile de définir l'ensemble des pommes. Par contre, il sera plus difficile de définir l'ensemble des pommes mûres. On conçoit bien que la pomme mûrit progressivement... la notion de pomme mûre est donc graduelle (BENYOUCEF).

C'est pour prendre en compte de telles situations qu'a été créée la notion d'ensemble flou. La théorie des ensembles flous repose sur la notion d'appartenance partielle : chaque élément appartient partiellement ou graduellement aux ensembles flous qui ont été définis. Les contours de chaque ensemble floue (**Figure 0.1**) ne sont pas « nets », mais « flous » ou « graduels »

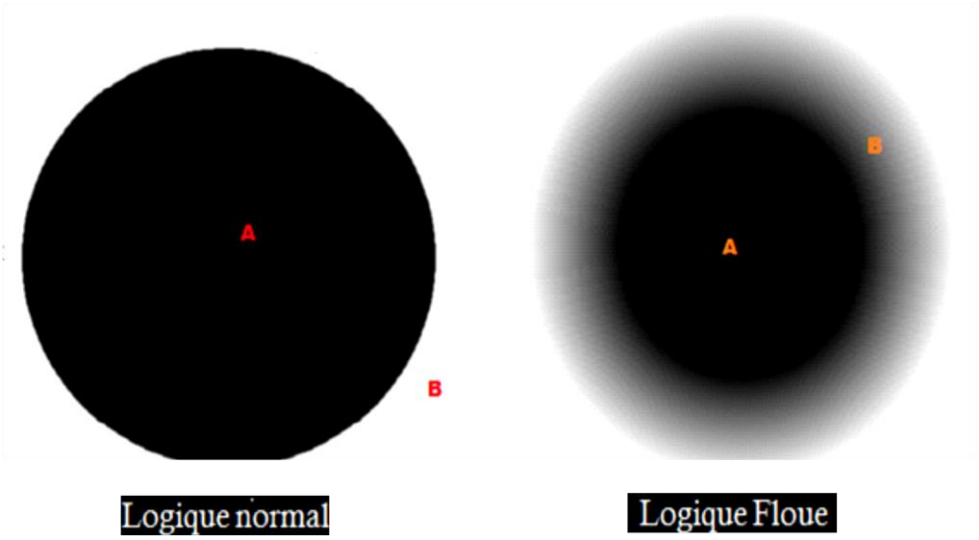

Figure 0.1. *Schémas explicatif de la différence entre la logique normal et la logique floue.*

II.15.2 Fonctions d'appartenance

Chaque sous-ensemble flou peut être représenté par sa fonction d'appartenance. En général la forme de fonctions d'appartenance dépend de l'application, et peut être triangulaire, trapézoïdale ou en forme de cloche comme le montre la **Figure 0.2** (TKOUTI, 2004).

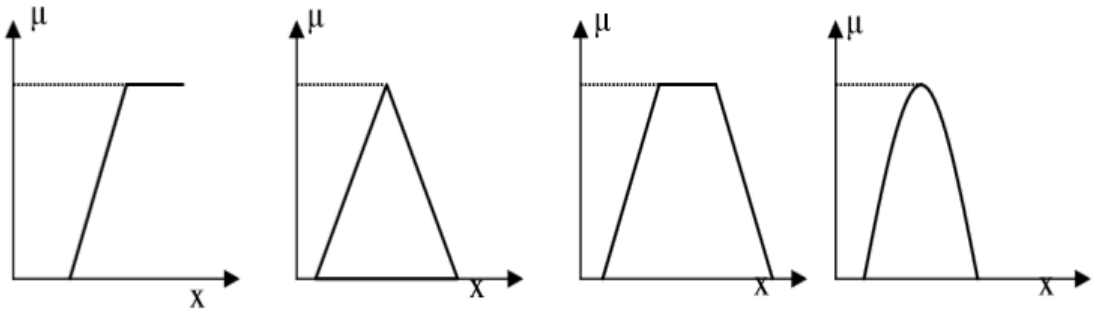

Figure 0.2. *Les différents fonctions d'appartenance de l'application floue.*

Plusieurs ensembles flous peuvent être définis sur la même variable, par exemple les ensembles « taille petite », « taille moyenne » et « taille grande », notions explicitées chacune par une fonction d'appartenance (**Figure 0.3**).

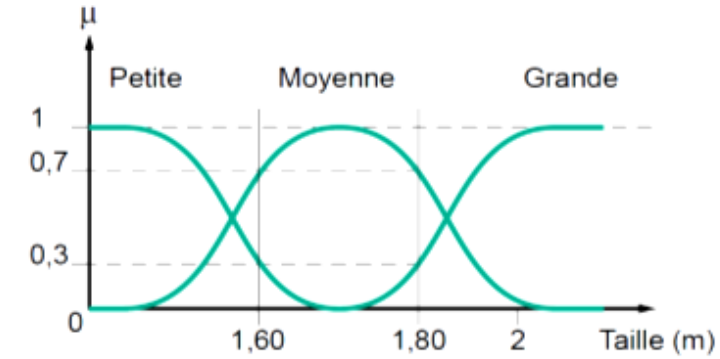

Figure 0.3. *Des ensembles contient plusieurs fonctions d'appartenance.*

II.16 L'utilisation de la logique floue

La logique floue est une technique de résolution de problèmes très puissants avec une large applicabilité dans le control et la prise de décision. Elle est très utile lorsque le modèle mathématique du problème à traiter n'existe pas ou existe mais difficile à implémenter, ou il est trop complexe pour être évaluer assez rapidement pour des opérations en temps réel, Ou bien lorsque des experts humains sont disponibles pour fournir des descriptions subjectives du comportement du système avec des termes en langage naturel. La logique floue est aussi supposée de travailler dans les situations où il y a de large incertitude et des variations inconnues dans les paramètres et la structure du système (TKOUTI, 2004).

II.17 Variables linguistiques

L'ensemble de référence d'un mot du langage naturel s'appelle l'univers du discours. L'univers du discours d'un mot est un ensemble de termes qui évoquent le même concept mais à degrés différents. Il peut être fini ou non. Une variable linguistique représente un état dans le système à régler. Chaque variable linguistique est caractérisée par un ensemble tel que : {X, T(x), U, G, M} (TKOUTI, 2004).

Où :

X : est le nom de la variable,
T(x) : est l'ensemble des valeurs linguistiques que peut prendre x,
U : est l'univers du discours associé avec la valeur de base,
G : est la règle syntaxique pour générer les valeurs linguistiques de x,
M : est la règle sémantique pour associer un sens à chaque valeur linguistique.

Exemple

La variable linguistique x = température ambiante, peut être définie avec un ensemble des termes :

T(x) = {extrêmement froide, très froide, froide, chaude, très chaude, extrêmement chaude}, qui forment son univers du discours U = [-20 C^0, 40 C^0]. La variable de base est la température. Le terme froid représente une valeur linguistique. On peut l'interpréter, par exemple comme « les températures plus petites que 15 C^0 »

II.18 Classification floue

La classification comprend en général deux étapes (OUMAYA & LIMAM, 2012):

- Préparatoire : détermination des classes à considérer.
- En ligne : affectation des éléments aux ensembles ces derniers sont mis aux points à partir:
    - D'une expérience.
    - D'exemples utilisés pour un apprentissage (par exemple dans le cas de classifier à réseaux de neurones).
    - D'une connaissance mathématique ou physique du problème.

Les méthodes de classification graduelle (ou floue) permettent, notamment, de mettre au point des boucles de régulation (OUMAYA & LIMAM, 2012).

II.19 Opérateurs en logiques floue

Ces opérateurs permettent d'écrire des combinaisons logiques entre notions floues, c'est-à-dire de faire des calculs sur des degrés de vérité. Comme pour la logique classique, on peut définir des opérateurs ET, OU et négation (HARROUCHE):

II.19.1 L'opération d'intersection

L'opérateur logique correspondant à l'intersection d'ensembles est le *ET*. Le degré de vérité de la proposition « A ET B » est le minimum des degrés de vérité de A et de B :

$$\mu_{A\cap B}(x) = \mu_A(x) \cap \mu_B(x) = \min\big(\mu_A(x), \mu_B(x)\big), \qquad \forall x \in U \qquad \text{(IV.39)}$$

II.19.2 L'opération d'union

L'opérateur logique correspondant à l'union d'ensembles est le *OU*. Le degré de vérité de la proposition « A OU B » est le maximum des degrés de vérité de A et de B :

$$\mu_{A\cup B}(x) = \mu_A(x) \cup \mu_B(x) = \max\big(\mu_A(x), \mu_B(x)\big), \qquad \forall x \in U \qquad \text{(IV.40)}$$

II.19.3 L'opération NON (complément)

Il est défini mathématiquement par :

$$\bar{A} = \{X/X \in A\} \quad \text{(IV.41)}$$

Et il est représenté par la fonction :

$$non(\mu_A(x)) = \overline{\mu_A(x)} = 1 - \mu_A(x) \quad \text{(IV.42)}$$

II.20 Structure d'un système flou

Un système flou peut être interprété selon deux points de vue : mathématique ou logique. D'un point de vue mathématique. Du point de vue logique, un système flou est un système à structure particulière, composé de quatre modules principaux : à savoir, la base de connaissance, la fuzzification, le moteur d'inférence et la défuzzification (BENBOUALI, 2008).

La commande floue est l'application la plus utilisée dans la logique floue. En effet, cette méthode permet d'obtenir un réglage souvent très efficace sans devoir faire des modélisations approfondies.

Par opposition à un régulateur standard, le régulateur par logique floue ne traite pas une relation mathématique bien définie, mais utilise des inférences (déductions) avec plusieurs règles, se basant sur des variables linguistiques

La **Figure 0.4** montre la configuration de base d'un *RLF*, pris comme exemple d'un système flou et qui sera l'objet de notre étude.

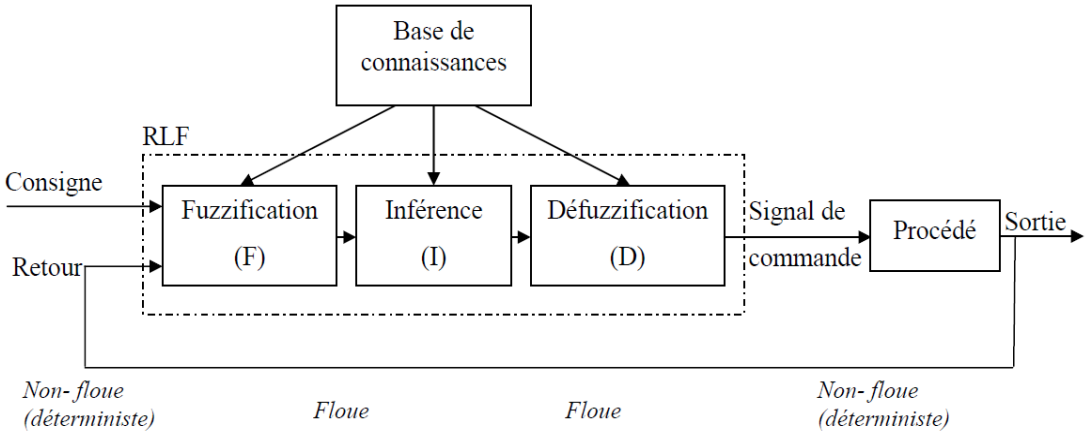

Figure 0.4. *Configuration de base d'un régulateur par logique floue (RLF).*

II.20.1 Interface Fuzzification

Dans les problèmes de commande, les données observées sont habituellement physiques (réelles). Or, le traitement de ces données est basé ici sur la théorie des ensembles flous, ceci nécessite donc une procédure de fuzzification (BENBOUALI, 2008).

La fuzzification, proprement dite, consiste à définir les fonctions d'appartenance pour les différentes variables d'entrées. On réalise ainsi le passage des grandeurs physiques (réelles) en variables linguistiques (variables floues) qui peuvent être alors traitées par les inférences. Dans la littérature de la commande floue, deux approches de fuzzification sont généralement utilisées : la fuzzification singleton et la fuzzification non-singleton (BENBOUALI, 2008).

II.20.2 Base de connaissances (Base de règles)

Une base de règles floues R est une collection de règles floues, décrivant le comportement du système. Elle est le cœur du système entier, dans le sens où tous les autres composants sont utilisés pour interpréter et combiner ces règles pour former le système final. Ces règles permettent de déterminer le signal de sortie du contrôleur en fonction des signaux d'entrées. Elles peuvent être fournies par un expert ou peuvent être extraites de données numériques.

Dans les deux cas, les règles prennent la forme « Si prémisse Alors conclusion ». D'une manière générale, la base de règles d'un système flou doit respecter les conditions de complétude et de consistance afin d'assurer le bon fonctionnement de ce dernier. Une base de règles d'un système flou est dite complète si, pour chaque vecteur d'entrée, il existe au moins une règle floue activée. Afin d'assurer cette propriété, les fonctions d'appartenance doivent couvrir tout l'espace des variables d'entrée. Une base de règles d'un système flou est dite inconsistante, s'il existe deux règles floues ayant la même prémisse mais des conclusions différentes. La propriété de consistance permet d'éviter les contradictions dans une base de règles (BENBOUALI, 2008).

II.20.3 Inférence floue

Le moteur d'inférence floue, ou la logique de prise de décision est le noyau (cerveau) du contrôleur flou. Elle est capable de simuler la prise de décision de l'être humain, en se basant sur le résonnement flou et l'ensemble des règles floues qui forment la base de règle.

Dans les règles floues interviennent les opérateurs « *ET* » et « *OU* ». L'opérateur « ET » s'applique aux variables à l'intérieur d'une règle, tandis que l'opérateur « *OU* » lie les différentes règles. Il existe plusieurs possibilités pour interpréter ces opérateurs (BENBOUALI, 2008).

$$\text{Si } x = A_1 \textbf{ \textit{et}} \ y = B_1 \text{ alors } z = C_1$$

$$\text{Si } x = A_2 \textbf{ \textit{et}} \ y = B_2 \text{ alors } z = C_2$$

…

Si x = A_n ***et*** y = B_n alors z = C_n

Où x, y et z sont des variables linguistiques qui représentent les variables d'état de processus et variables de contrôle; A_i, B_i et C_i (i=1, n) sont les sous-ensembles flous définis dans les ensembles de référence pour x, y et z respectivement. En toute généralité, n'importe quelle combinaison des opérateurs ***ou***, ***et*** et ***non*** peut apparaître dans la condition d'une règle, suivant les conditions imposées par le système à régler.

II.20.4 Interface de défuzzification

Comme nous l'avons vu, les méthodes d'inférence fournissent une fonction d'appartenance résultante $\mu_{res}(z)$ .Mais, l'organe de commande nécessite un signal de commande précis (réel) à son entrée ; donc il faut prévoir une transformation de cet ensemble flou en une grandeur précise. Cette transformation est appelée : défuzzification (BENBOUALI, 2008).

Il existe plusieurs stratégies de défuzzification, les plus utilisées sont (BENBOUALI, 2008) :

- La méthode du centre de gravité ;
- La méthode des hauteurs pondérées ;
- La méthode de la moyenne des maximums ;

La plus souvent rencontrée étant le calcul du « centre de gravité » de l'ensemble flou (**Figure 0.5**).

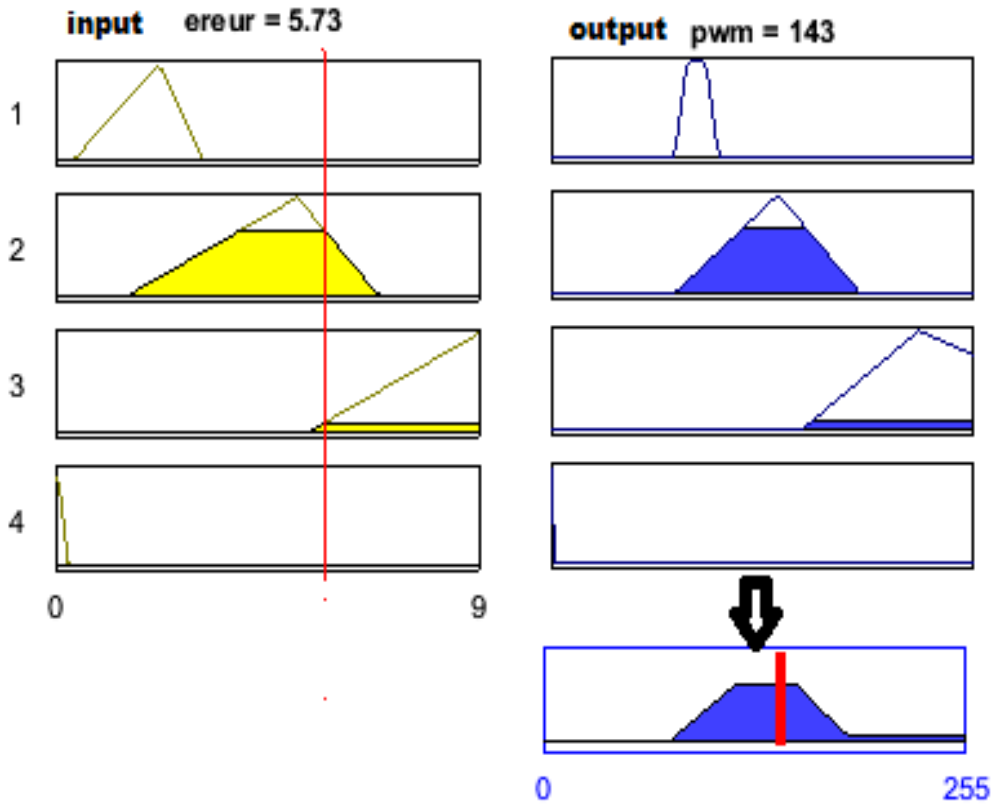

Figure 0.5. *Défuzzification par centre de gravité.*

II.21 Conception du régulateur floue (Pratique)

A l'aide de boite à outil de Matlab « Fuzzy Logic Toolbox » on a construit un régulateur floue à deux entrées (l'erreur et la dérivée d'erreur) et une sortie.

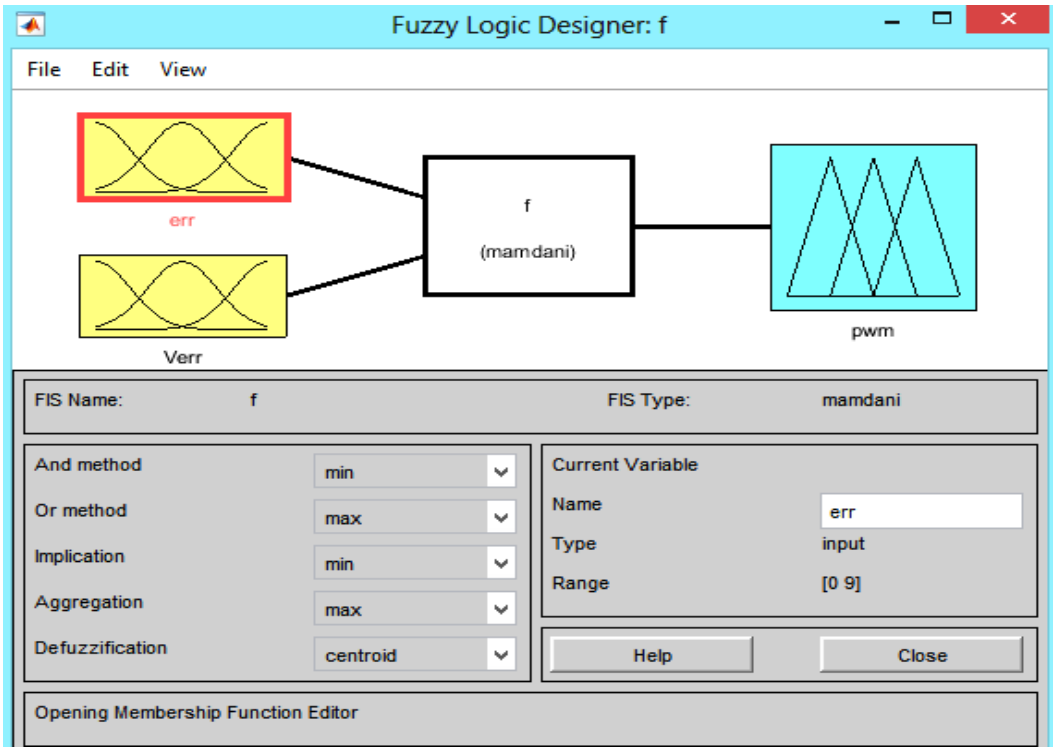

Figure 0.6. *L'interface de 'Fuzzy logic toolbox'.*

II.21.1 Fuzzification

On a retenu les variables suivantes :

T	Très bas
M	moyenne
H	haut
N	négative
Z	Environ zéro
P	Positive

Tableau 0.1. *Les Variables linguistiques utilisées dans RLF.*

a) La première entrée (erreur)

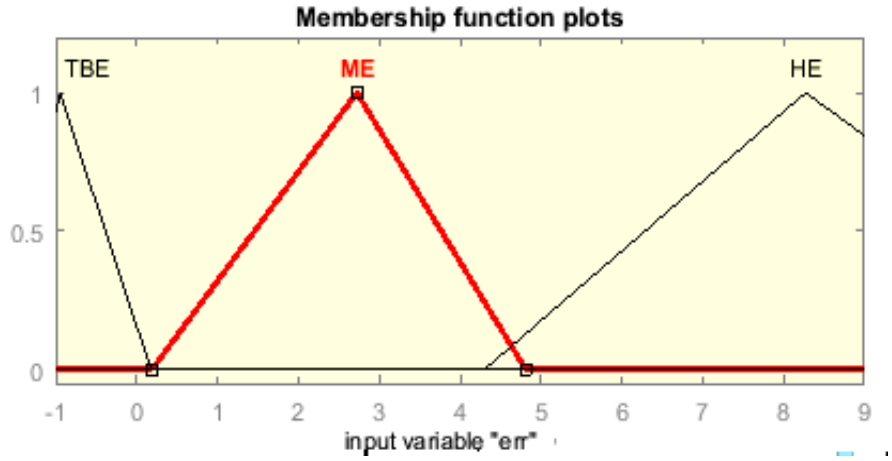

Figure 0.7. *Fonctions d'appartenance d'erreur.*

b) La deuxième entrée (la dérivé d'erreur)

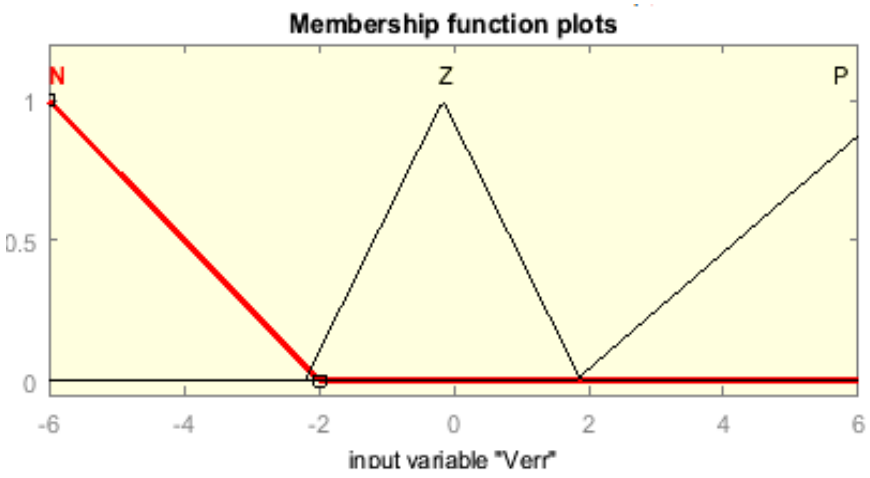

Figure 0.8. *Fonctions d'appartenance du dérivé d'erreur.*

c) La sortie du régulateur

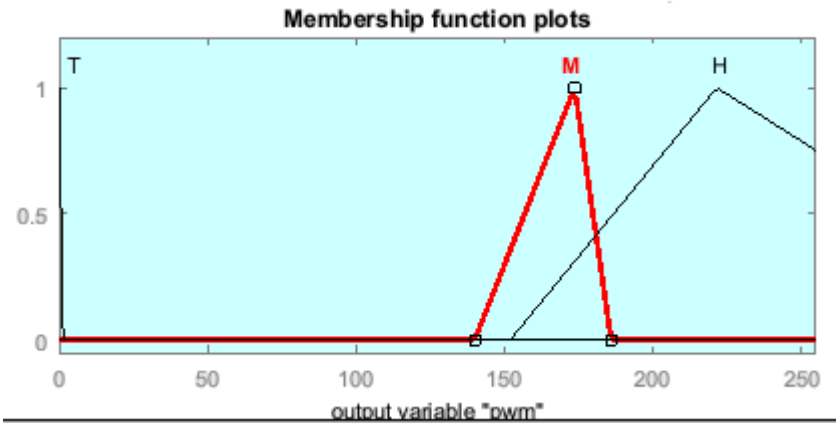

Figure 0.9. *Fonctions d'appartenance de sortie.*

Les règles floues caractérisent ce régulateur est représenté par le tableau :

D-erreur / Erreur	N	Z	P
T	T	T	T
M	M	M	H
H	M	H	H

Tableau 0.2. *Matrice d'inférence de RLF.*

II.21.2 Inférence floue

La méthode d'inférence utilisée dans notre cas est celle de Mamdani (Max- Min), où l'opérateur « ET » est réalisé par la fonction Min, l'opérateur « ALORS » de chaque règle par la fonction Min et la liaison entre toutes les règles (l'opérateur OU) par la fonction Max.

II.21.3 Défuzzification

L'étape de la défuzzification est la dernière étape pour la conception d'un régulateur flou. Elle consiste à transformer une valeur floue (fonction d'appartenance résultante) en une valeur physique (déterministe). Dans notre travail, notre choix c'est porté sur la méthode la plus utilisée grâce à sa simplicité. Il s'agit de la défuzzification par centre de gravité

II.22 Implémentation du régulateur floue

On a implémenté un régulateur PI dans la carte *Arduino* à partir d'un model Simulink grâce à la bibliothèque « Arduino support for Simulink ».

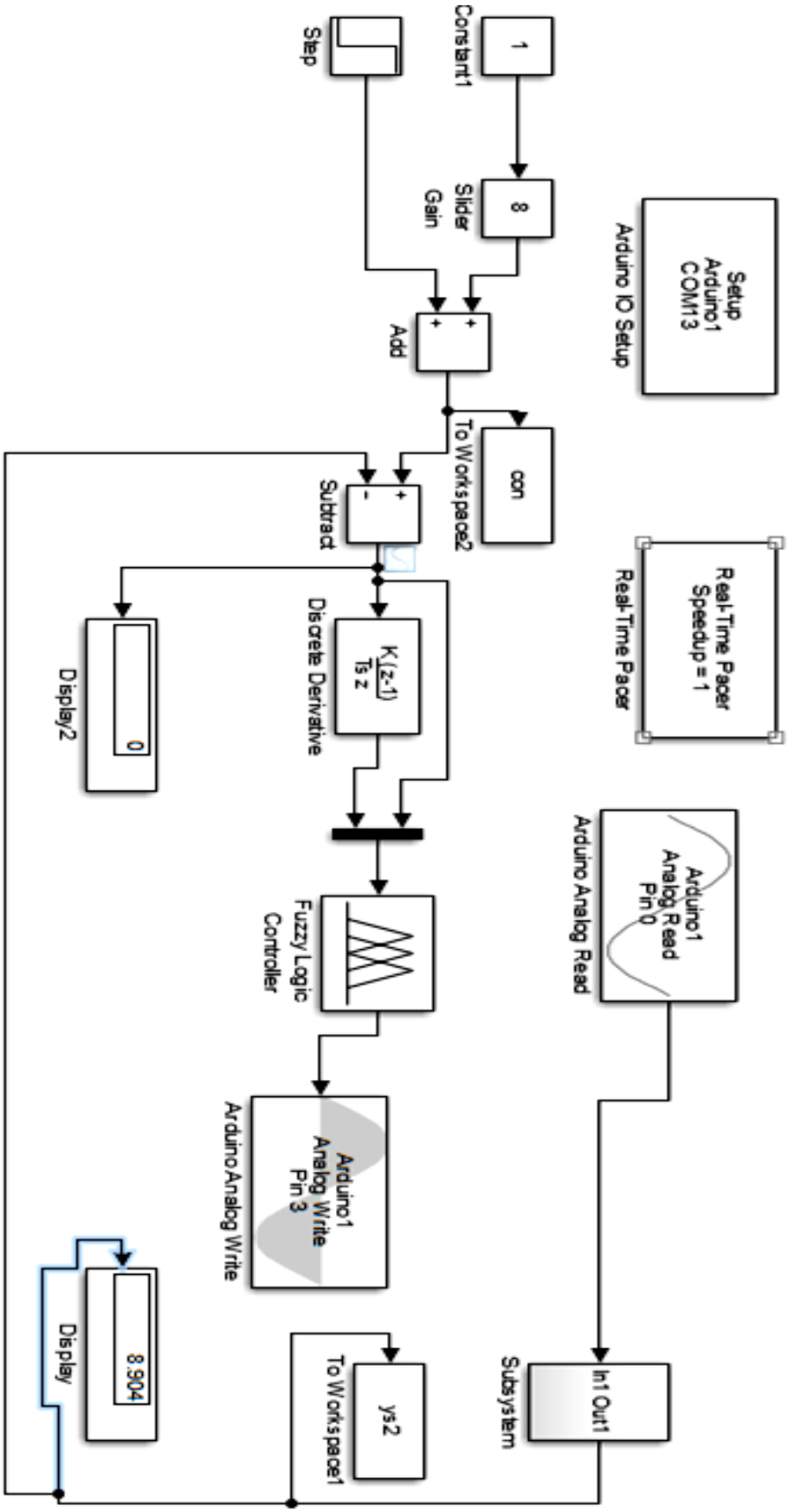

Figure 0.10. *Représentation d'un correcteur flue sous Simulink adapté à Arduino.*

II.23 Résultats expérimentaux et interprétations

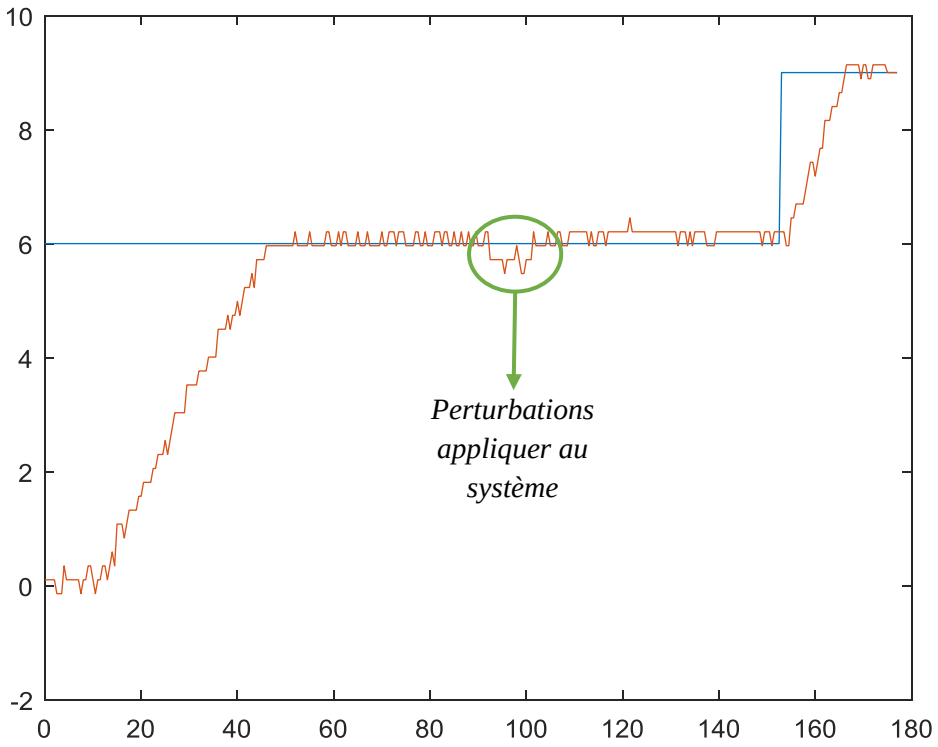

Figure 0.11. *Poursuite de consigne et rejet de perturbation d'un régulateur floue.*

II.23.1 Interprétation du résultat

Nous avons appliqué deux consignes au système afin de valoriser son comportement selon le tableau suivant :

T(s)	0	155
E(cm)	6	9

Tableau 0.3. *Les consignes imposées.*

Nous remarquons que la sortie du système suit l'entrée avec un temps de réponse de 45.5s ([0,20]s le capteur n'été pas encours dans sa point fonctionnement). A l'instant 95s on a appliqué une perturbation qui a causé une descente de 0.4cm le système a pu rejeter cette perturbation grâce à l'action Intégrale du correcteur, mais le temps qu'il a fallu pour ce rejet est 6s

II.23.2 Discussion des résultats

Le test précédent (Figure 0.11) et a pour objet l'étude du comportement de contrôleur considéré. Dans des conditions de fonctionnement nominal on peut remarquer que la consigne de niveau est suivie par la sortie du système (très bonne précision). Le rejet des perturbations se fait d'une manière rapide. Nous remarquons l'influence de perturbation n'est pas importante (négligeable). Si on compare ces résultats avec les résultats obtenus expérimentalement par le régulateur classique PI (**Figure 0.8**), nous voyons des améliorations surtout ce qui concerne le rejet de perturbation et la précision.

II.24 Conclusion

Dans ce chapitre, nous avons présenté la logique floue à travers le régulateur de type Mamdani et on a démontré que cette technique est classée parmi les techniques de l'intelligence artificielle, ensuite nous avons conçu un régulateur à notre système, nous l'avons implémenté à l'aide de *Matlab,* (*Simulink*). L'ensemble des résultats obtenus ont montré la supériorité de la régulation floue à la régulation PID classique.

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 asservir en temps real.

L'objectif de ce projet de fin d'études, était la conception et la réalisation d'une commande numérique pour la régulation de niveau d'eau à l'aide d'une carte Arduino UNO

On a utilisé cette carte pour l'acquisition des donnes entre le system et le pc pour l'identification puis comme un contrôleur pour la régulation de niveau d'eau dans la cuve.

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', 'Fuzzy Logic', 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 circuit de commande de la pompe qui jeux le rôle d'actionneur et le problème de l'indisponibilité du capteur de niveau robuste et compatible avec la carte Arduino, nous oblige de construire un capteur base sure 'un flotteur + potentiomètre' qui donne des résultats perturber et bruité, malgré tous les difficulté nous avons pu réaliser notre maquette, et applique deux types de régulateur numérique (PI et floue) à notre système. Ainsi, on a fait une comparaison entre les deux correcteurs. Il résulte que le contrôleur floue est robuste et efficace par rapport au régulateur PI dans la régulation de niveau et en général.

Enfin comme perspective de notre travail : l'utilisation de l'outil supervision qui nous permettre de visualiser et de changer les paramètres des consignes d'entrée et changer le choix du contrôleur. L'utilisation de la sonde de pression grâce à sa précision et sa robustesse par rapport au capteur résistif flotteur. La réalisation d'un système à deux cuves.

Annexe A

L'outil d'identification et de logique floue sous Matlab

A. 1 Introduction

System Identification Toolbox™ développe des modèles mathématiques de systèmes dynamiques à partir de données d'entrée-sortie mesurées. Il offre des fonctions, une application d'identification de système ainsi que des blocs Simulink® pour créer et utiliser des modèles de systèmes dynamiques difficiles à modéliser à partir de premiers principes ou de premières spécifications. Vous pouvez utiliser des données d'entrée-sortie des domaines temporel et fréquentiel afin d'identifier des fonctions de transfert à temps continu et à temps discret, des modèles de processus et des modèles de représentations d'état.

La boîte à outils fournit des techniques d'estimation incluant la vraisemblance maximale, les schémas de PEM (Prédiction-Error Minimization, minimisation d'erreur de prédiction), des méthodes de sous-espace et d'autres techniques d'identification. Concernant la dynamique d'un système non linéaire, vous pouvez estimer les modèles ARX, ARMAX....ect. à l'aide d'un réseau à ondelettes, du partitionnement d'arbre et des non linéarités d'un réseau sigmoïde. La boîte à outils exécute une identification du système de la boîte grise afin d'estimer les paramètres d'un modèle défini par l'utilisateur. Vous pouvez utiliser le modèle identifié pour prévoir une réponse du système et effectuer une simulation dans Simulink. La boîte à outils vous permet également de modéliser des données de séries chronologiques et de réaliser leur prédiction **(LJUNG, 2008)**.

A. 2 Démarrage de l'outil d'identification

Tapez « Ident » à la ligne de commande de Matlab pour démarrer l'interface graphique.

A. 2.1 Interface

La boîte à outil « System Identification » possède une interface graphique qui est présentée à la **Figure .1**

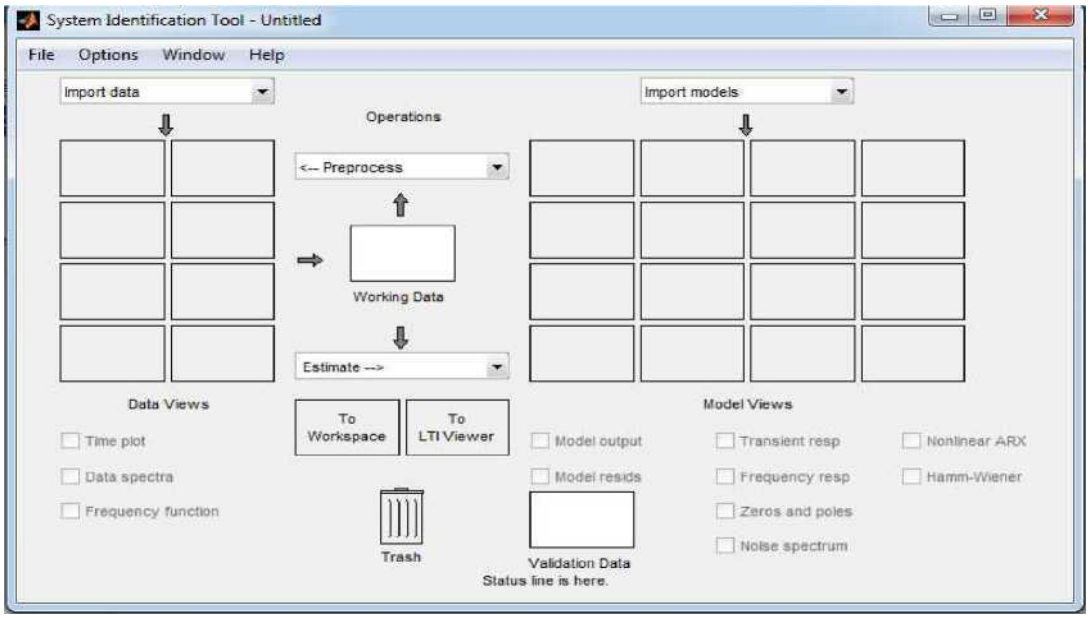

Figure A.1. *Interface graphique de l'outil Identification toolkit.*

Celle-ci représente chaque jeu de données ou modèle estimé par une icône dans les zones rectangulaires. Les jeux de données se retrouvent à gauche et les modèles à droite. Il est possible de déplacer ces icônes en les cliquant et en les déplaçant. Vous pouvez les effacer en déplaçant leur icône sur la poubelle.

L'interface contient également quatre menus déroulants (LJUNG, 2008):

- « Import data » pour les commandes liées à l'importation de données.
- « Import models » pour les commandes liées à l'importation de modèles.
- « Preprocess » pour les commandes liées à la création de nouveaux jeux de données basés sur les données en cours d'utilisation (« Working Data »).
- « Estimate » pour les commandes liées à la création de modèles basés sur les données en cours d'utilisation.

A. 2.3 Importation des données

Pour débuter, vos données d'entrée-sortie du test d'identification doivent se trouver dans l'espace de travail de Matlab. Sélectionnez ensuite « Time domain data... » sous le menu déroulant « Import data » (**Figure A.**0.2). Entrez dans le champ « Input » le vecteur correspondant à vos données d'entrée et sous « Output » celui correspondant aux données de sortie. Vous pouvez spécifier un nom pour votre jeu de données sous « Data name ». Fixez le « Sampling interval » à la période d'échantillonnage en secondes et le « Starting time » à 0. Cliquez sur « import ».

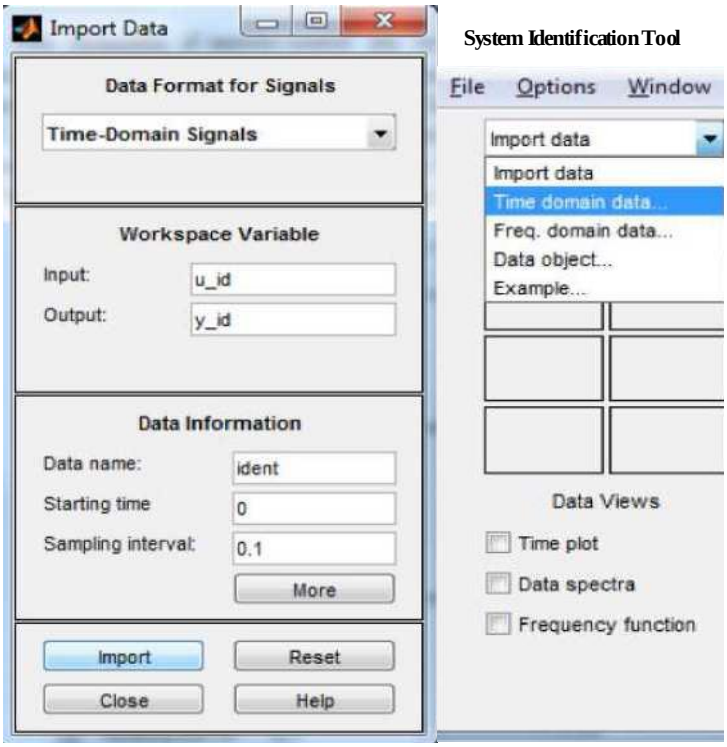

Figure A.0.2. *Fenêtre d'importation de données.*

Il est préférable d'utiliser deux jeux de données pour l'identification : un premier jeu pour estimer les paramètres du modèle et un deuxième pour valider le modèle obtenu. Répétez l'opération d'importation pour les données de validation. Cliquez et glissez le premier jeu de données sur la case « Working Data » et le deuxième sur la case « Validation Data ». Votre interface devrait ressembler à celui de la **Figure .**0.3.

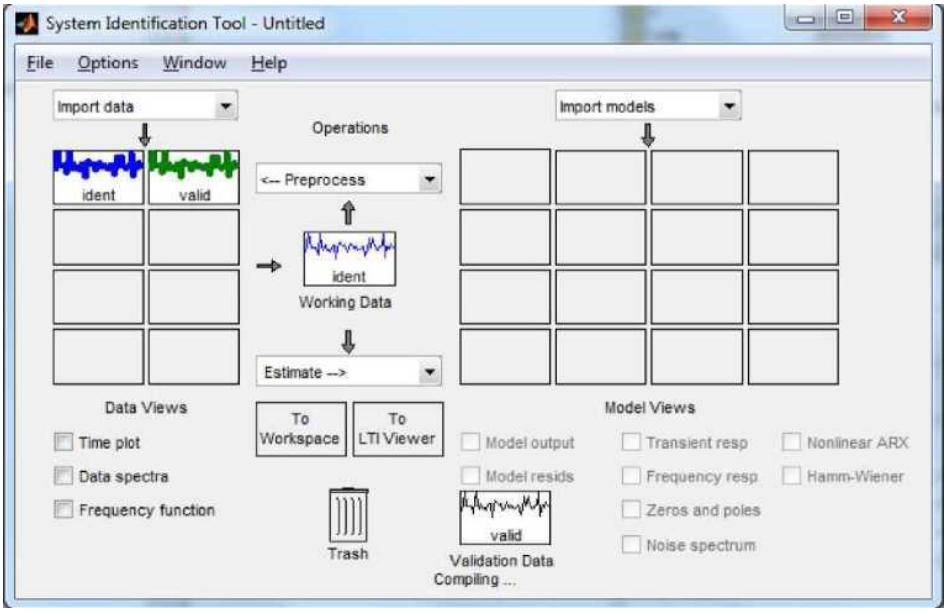

Figure A.0.3. *Jeu de données « ident » dans « Working Data » et « valid » dans « Validation Data ».*

Trois modes sont disponibles pour visualiser les données. Cliquez sur la case correspondant à votre choix dans la section « Data Views » située sous la zone réservée aux données. Les données sont actives pour visualisation si elles sont représentées par une ligne plus épaisse. I faut cliquer sur le jeu de données pour le faire basculer entre l'état actif et inactif dans le graphique. Dans la **Figure A.**0.4, le jeu de données « ident » est actif tandis que « valid » ne l'est pas.

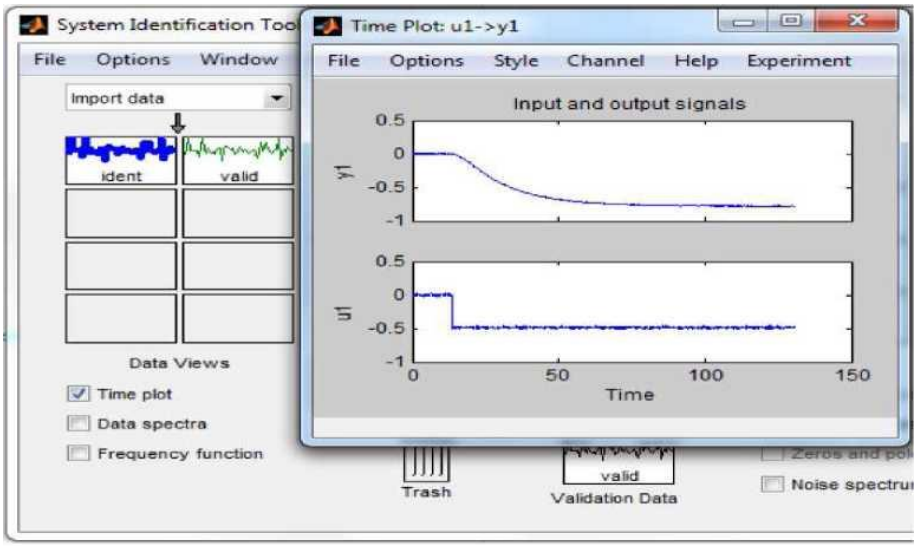

Figure A.0.4. *Visualisation du jeu de données « ident » dans le domaine du temps (« Time plot »).*

A. 2.3 Estimation des paramètres du modèle

Il existe plusieurs méthodes disponibles pour l'estimation de modèles basés sur le jeu de donné en cours d'utilisation :

Linear parametric models;

- Process models.
- Nonlinear models.
- Spectral models.
- Correlation models.

Voyons un peu plus en détail le cas du modèle dans le domaine continu qu'est le « Process model ». Ce modèle est caractérisé par un gain statique, des constantes de temps et un retard. La fenêtre d'estimation des paramètres est présentée à la **Figure** 0.5.

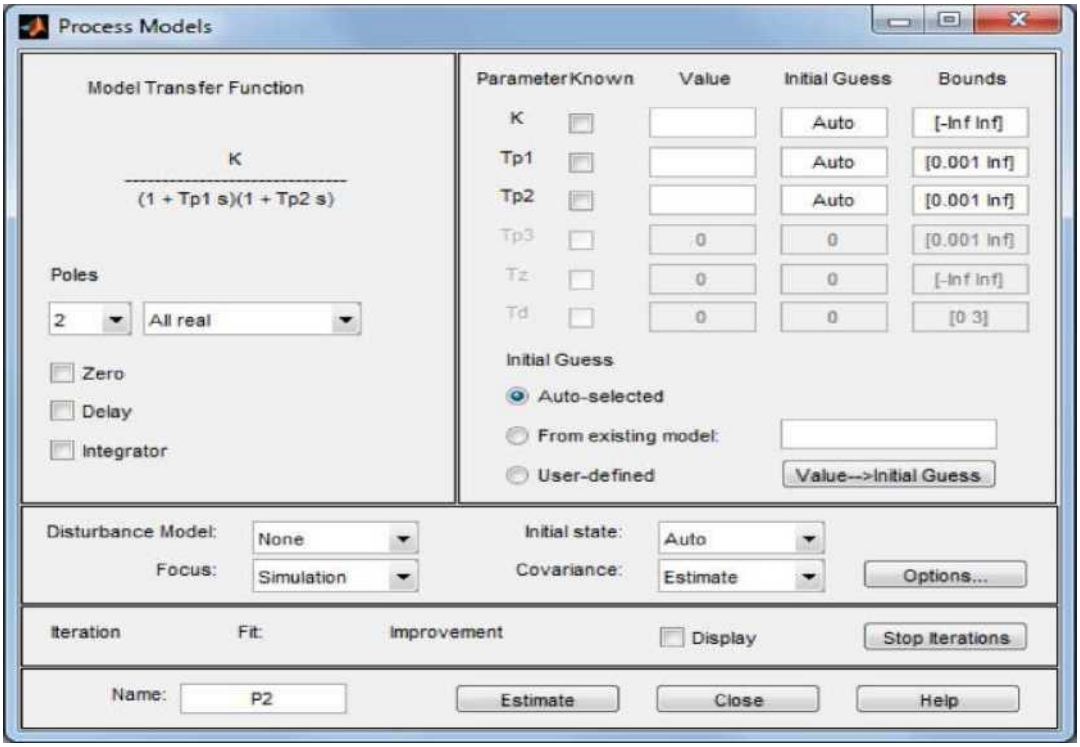

Figure A.0.5. *Fenêtre d'estimation des paramètres du « Process model ».*

La partie supérieure gauche de la fenêtre permet de forcer le nombre de pôles du modèle et la présence ou non d'un zéro, d'un intégrateur ou d'un retard. On peut également spécifier si ces pôles doivent être réels ou complexes.

Si certains paramètres sont connus, on peut entrer leur valeur et cocher la case « Known » à la droite de la fenêtre. L'algorithme utilisera ces valeurs pour le modèle. Il est aussi possible d'entrer une valeur approximative du paramètre pour débuter l'estimation mais il est préférable de laisser l'algorithme chercher par lui-même que d'entrer une approximation erronée. Finalement, si les bornes d'un paramètre sont connues, il est possible de les définir.

Les derniers paramètres sont relatifs au modèle et il est possible d'y spécifier si l'on doit modéliser le bruit, de définir comment sont gérées les conditions initiales du modèle, s'il y a une bande passante dans laquelle le modèle doit être optimal et l'écart type des paramètres estimés.

A. 2.4 Validation du modèle

À l'étape de validation, votre modèle se retrouve dans la section de droite. Il existe 6 options pour les modèles linéaires et 2 pour les non-linéaires. Ces options se retrouvent sous « Model Views ». L'option de visualisation « Model output » permet de d'apprécier le

pourcentage d'ajustement de votre modèle par rapport aux données de validation (**Figure A.**0.6).

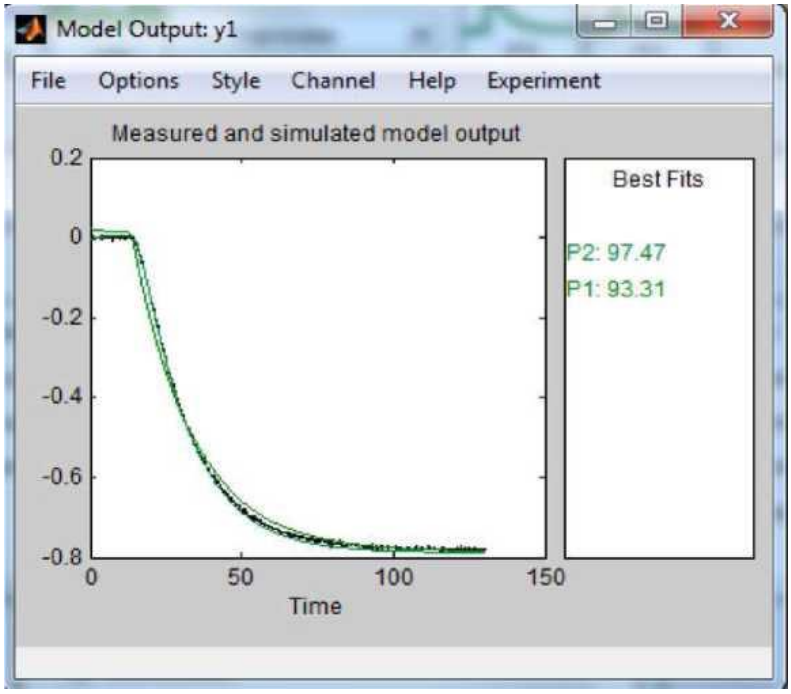

Figure A.0.6. *Visualisation des sorties des modèles P1 et P2 et des données de validation avec l'option « Model output ».*

Un clic droit sur l'icône du modèle permet d'en afficher les informations. Le bouton « Present » inscrit ces informations dans l'espace de travail de Matlab et affiche l'écart-type des paramètres si l'option « Estimate » du menu déroulant « Covariance » a été choisie dans le cas d'un « Process Model ».

A. 3 Logique floue

A. 3.1 Présentation du Toolbox fuzzy logic

La *Toolboxfuzzy Logic* de Matlab possédé un éditeur qui permet de créer des systèmes d'identification floues : des FIS (pour fuzzy interférence système). Cette boite à outils permet de générer des fichier «. Fis », qui correspondent à des systèmes d'inférences floues et dont font partie les RLF.

Cette boite à outils possédé 3 éditeur (de fis, de règle et de fonctions d'appartenance) qui permettent de saisir l'ensemble des données du fis ainsi que 2 interfaces graphiques qui permettent de visualiser les inférences directement sur la base de règles, ainsi qui des surfaces de contrôle.

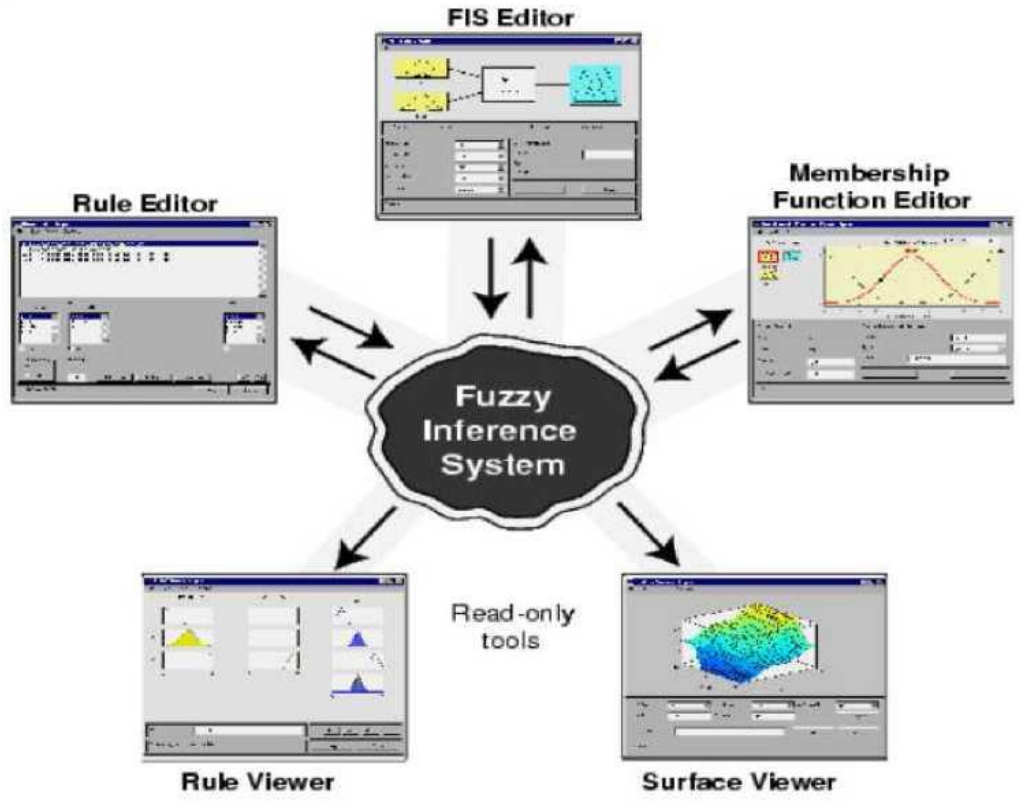

Figure A.0.7. *Présentation du Toolbox fuzzy logic.*

On ajoute une entrée, on accède à l'éditeur de fonctions d'appartenance « *Membership function Editeur* » dans lequel on applique les choix de A. On introduit alors les règles dans le « *Rule Editor* » selon le choix fait en A. Il reste à spécifier la méthode d'inférence.

A. 3.2 Editeur du système d'inférence Flou (FIS Editor)

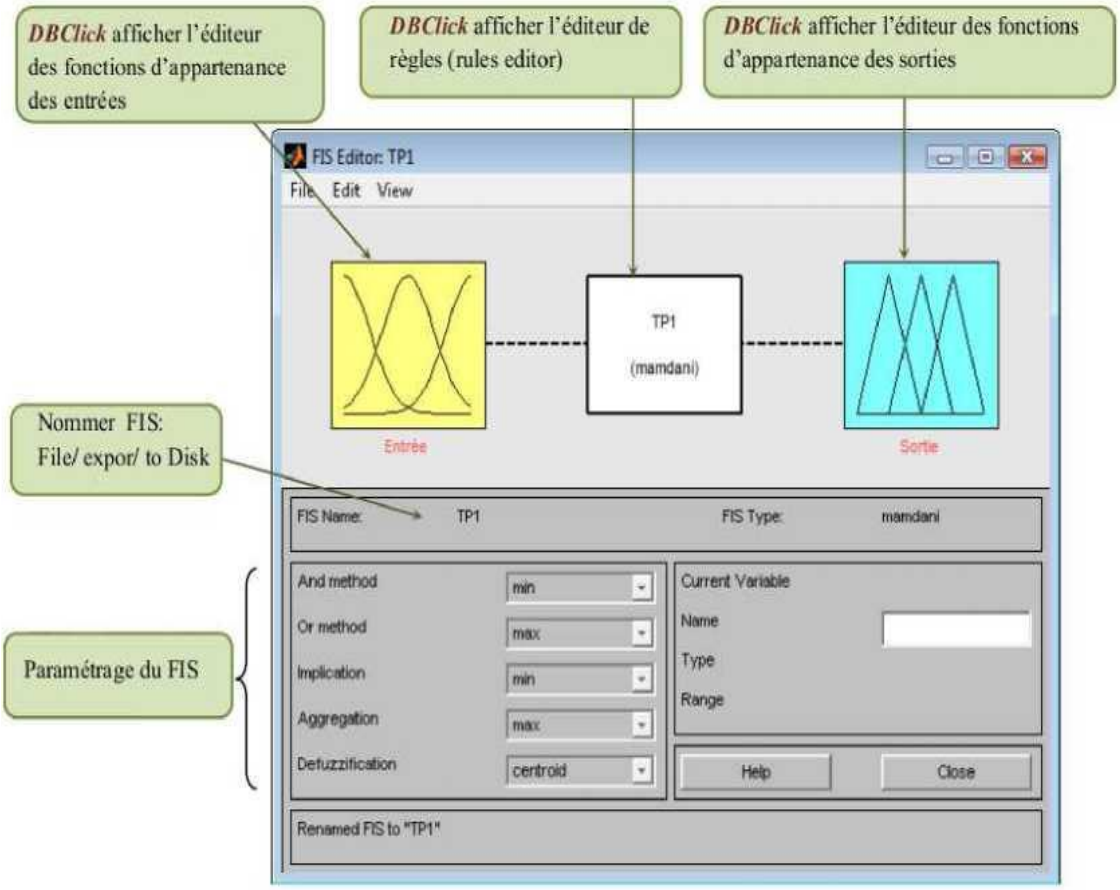

Figure A.0.8. *Editeur du système d'inférence Flou.*

A. 3.3 Editeur du Fonction d'Appartenance (MF Membership Fonction Editor)

Cette interface permet la saisie des fonctions d'appartenances aussi bien des entrées que des sorties, on y effectue plusieurs paramétrages concernant :

A. 3.3.1 Paramétrage du (Univers de discours) Domaine de variation

Chaque domaine de variation (entrée/sortie) est caractérisé par :

- Name : le nom du domaine
- Type entrée /sortie
- Range : Intervalle de variation de la variable linguistique
- Display Range : intervalle de variation de la variable linguistique visible

A. 3.3.1.1 Paramétrages des Termes linguistique (variable linguistique) :

- Edit/Add MFs : Ajoute une fonction d'appartenance

Les paramètres d'une fonction d'apartenances sont :

– Name : Le nom du sous ensemble (terme linguistique).
– Type : la forme de la courbe.
– Param : Vecteur définissant la fonction d'appartenance.

- Edit/ Remove Slected MEs : Ajoute une fonction d'appartenance.
  - Edit/ Remove

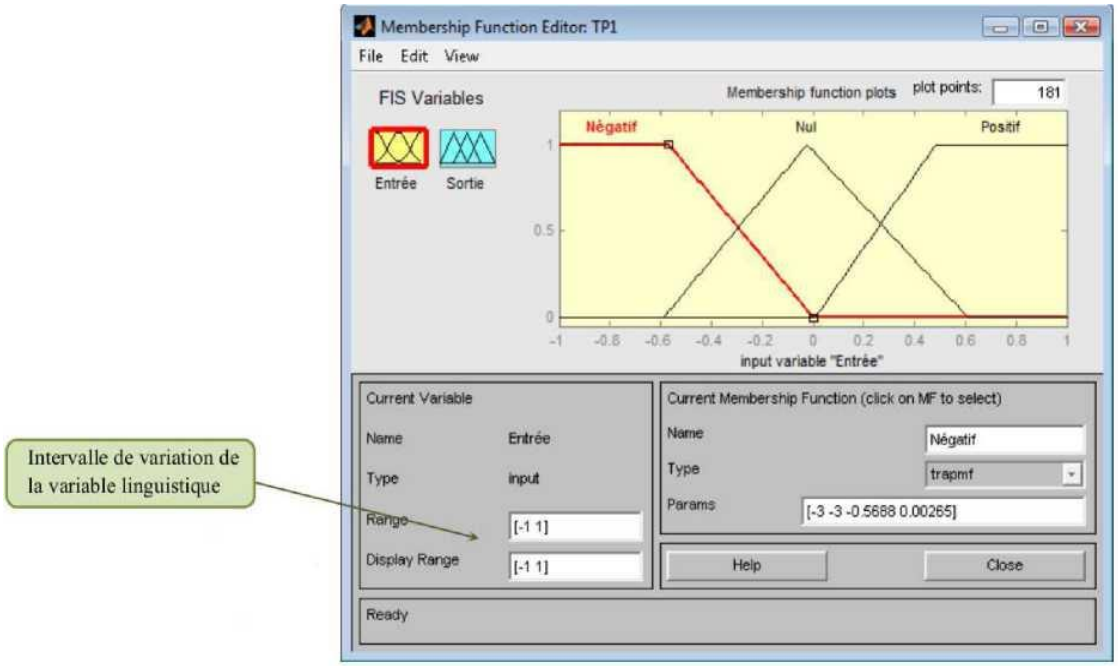

Figure 0A.0.9. *Editeur de fonction d'appartenance.*

B. Annexe B

L'outil de calcul des paramètres de PID 'PID TUNER' sous Logiciel Matlab

B. 1 'PID tuner' toolbox

'PID tuner' toolbox est le processus de trouver les valeurs des gains proportionnels, intégraux et dérivés d'un contrôleur PID pour réaliser la performance (prestation) désirée et respecter des exigences de design(conception). « PID tuner' toolbox » apparaît facile, mais la découverte de l'ensemble des gains qui assure que la meilleure performance(prestation) de votre système de commande est une tâche complexe. Traditionnellement, PID des contrôleurs sont accordé manuellement ou l'utilisation de méthodes à base de règle. Le Manuel accordant des méthodes est itératif et consommateur de temps et si utilisé sur le matériel (la quincaillerie), ils peuvent endommager. Des méthodes à base de règle ont aussi des limitations sérieuses : ils ne supportent(soutiennent) pas les certains types de modèles d'usine (de plante), comme des usines (plantes) instables, des usines (plantes) d'ordre haut (de commande haute), ou des usines(plantes) avec peu ou pas de retard de temps. Vous pouvez automatiquement accorder des contrôleurs PID pour réaliser le design (la conception) de système optimal et respecter des exigences de design(conception).

B. 2 L'utilisation de 'PID Tuner' toolbox

Pour avoir déterminer les paramètres de régulateur PID d'un système a asservis il faut suivre les étapes suivantes :

1) Ouvrir l'outil '*PID Tuner'* : en cliquant sur l'icône de « *PID Tuner* »

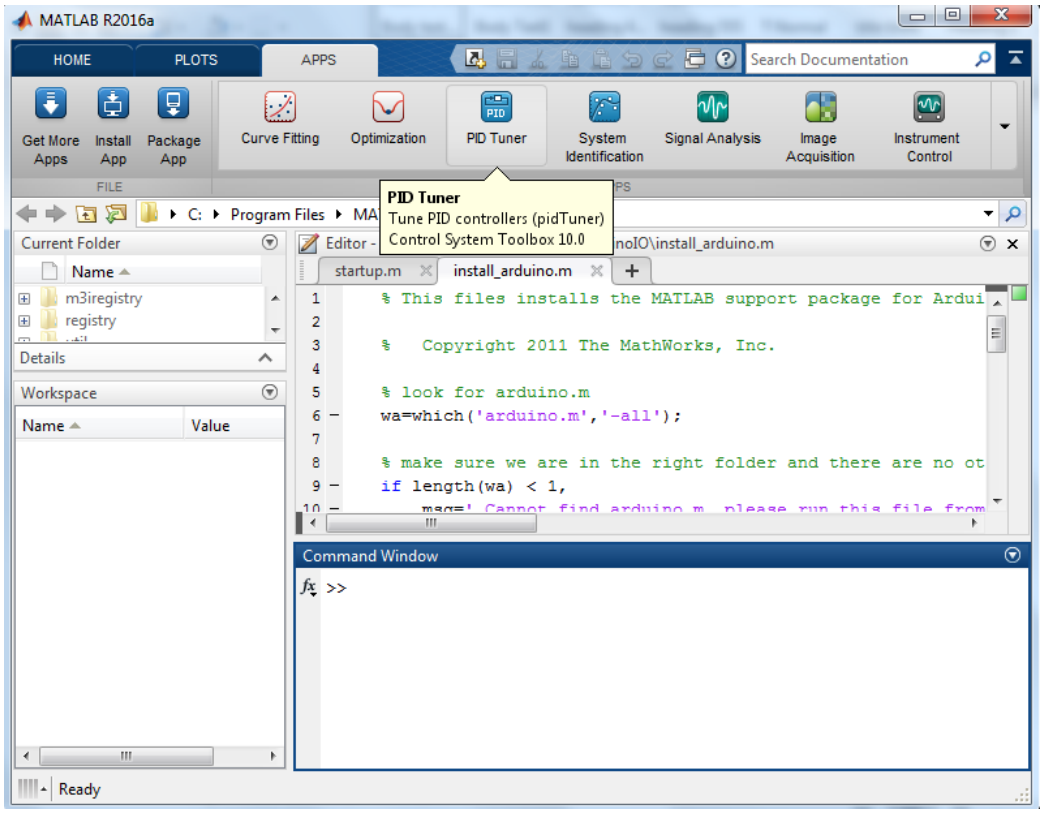

Figure B.1. *Interface de logiciel Matlab indiquant l'emplacement de l'outil "PID tuning".*

2) Cliquer sur *Import new plant*, une nouvelle fenêtre apparait. : comme il montre la figure **ci-dessus**

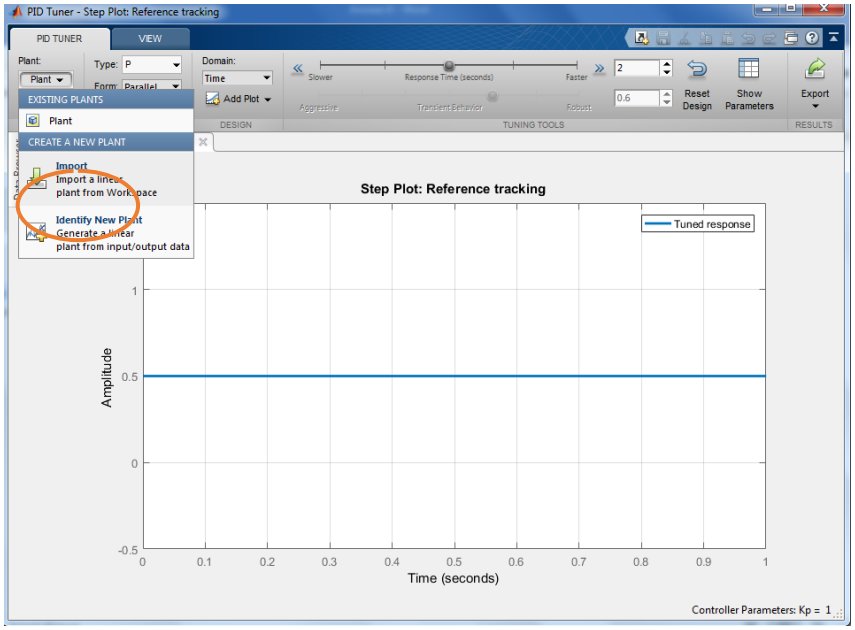

Figure B.2. *Importation du modèle estimé.*

3) Une nouvelle fenêtre apparait dans laquelle vous allez sélectionner tf2 ensuite cliquer sur import puis close.

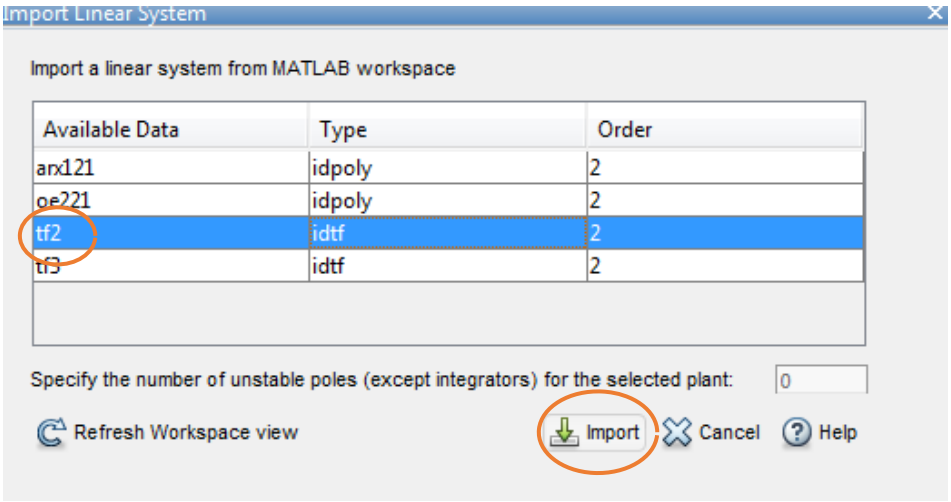

Figure B.3. *L'interface qui apparaitre lors l'importation du modèle.*

4) Revenir à la fenêtre PID Tuner, vous pouvez choisir le type de régulateur à implémenter (P, PI, PID, …etc) et les objectifs de la commande en boucle fermé et voir la réponse de la sortie du système.

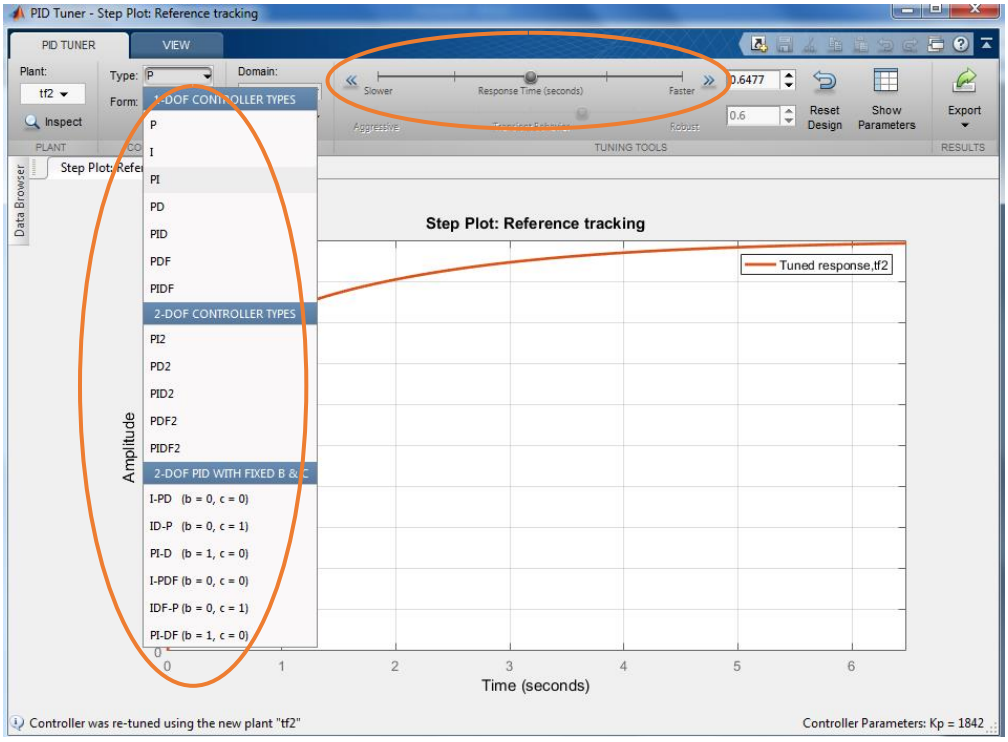

Figure B.4. *Choix du régulateur à implémenter.*

5) Cliquer sur la flèche de *show parameter* pour voir les paramétrés utilisés de votre régulateur ainsi que les performances du système en boucle fermé.

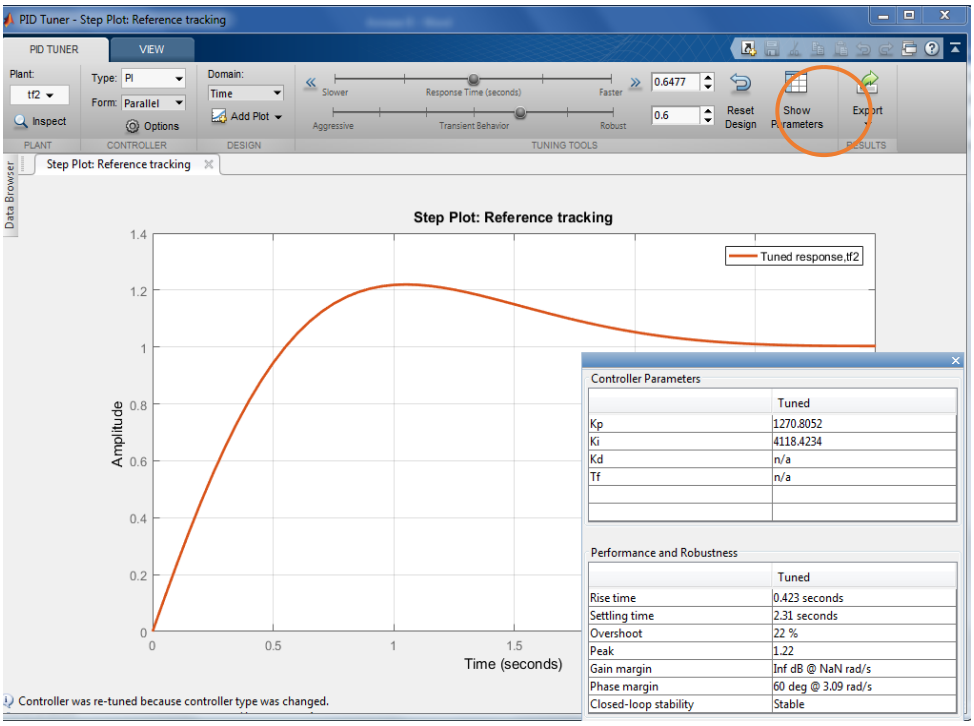

Figure B.5. *Récupération des paramétrés du régulateur.*

B. 3 La méthode graphique pour calculer les paramètres de correcteur

B. 3.1 Méthodes utilisables en boucle ouverte pour des processus instables

L'envoi d'un échelon sur un système en boucle ouverte qui contient une intégration dans sa fonction de transfert en boucle ouverte, on a un signal de sortie qui évolue linéairement donc instable. **La Figure B.6** donne l'allure de cette réponse **(VILLAIN, 2007)**.

Avant de procéder à l'identification il faut isoler le retard naturel. *Strejc* a proposé le modèle suivant

$$T(P) = \frac{K_e{}^{-TP}}{P(1+\tau \cdot P)^n} \tag{B.43}$$

La méthode est alors la suivante **(VILLAIN, 2007)**:

- • on trace la droite du régime permanent, qui coupe la valeur de sortie du point de repos en A,

• on trace la parallèle à cette droite et qui passe par le point O', soit la fin du retard naturel,
• on trace la perpendiculaire en A à l'axe des temps, elle coupe s(t) en B et la parallèle en C.

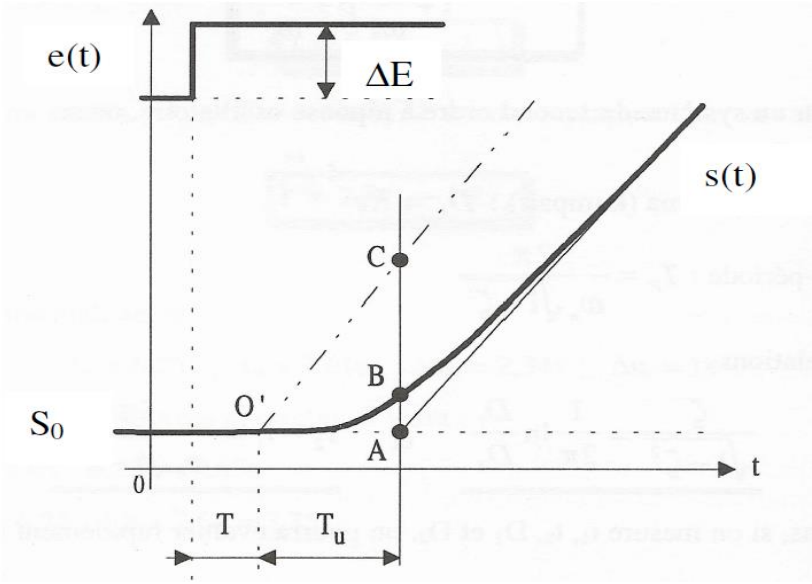

Figure B.6. *Réponse indicielle d'un système instable en boucle ouverte.*

Le gain $K = \frac{a}{\Delta E}$ avec a, la pente de s(t) en régime permanent et DE, la variation de l'échelon, Ensuite, *Strejc* a calculé les réponses indicielles des fonctions de transfert en boucle ouverte

$$T(P) = \frac{1}{P(1+\tau P)^n} \tag{B.44}$$

valeurs de n et a constitué le tableau suivant en fonction du rapport $\frac{AB}{AC}$

n	1	2	3	4	5
$\frac{AB}{AC}$	0.37	0.27	0.255	0.195	0.175

Tableau B.1. *Détermination de n pour une ftbo instable par la méthode de Strejc.*

Une fois n déterminé, on obtient la constante de temps τ par la relation

$$\tau = \frac{T_u}{n} \tag{B.45}$$

Si le transitoire est très plat ou s'il existe un retard important, on peut utiliser le modèle Suivant :

$$T(P) = \frac{K_e{}^{T_u P}}{P} \tag{B.46}$$

Calcul des actions P, I, et D pour les systèmes instables

Mode des actions	P	PI série	PID série	P parallèle	PI parallèle	PID mixte
K_p	$\frac{0.8}{K\cdot\tau}$	$\frac{0.8}{K\cdot\tau}$	$\frac{0.8}{K\cdot\tau}$	$\frac{0.8}{K\cdot\tau}$	$\frac{0.9}{K\cdot\tau}$	$\frac{0.9}{K\cdot\tau}$
K_i	Maxi	5τ	$\frac{(K\cdot\tau\cdot\tau)}{0.15}$	4.8τ	$\frac{(K\cdot\tau\cdot\tau)}{0.15}$	5.2τ
K_d	0	0	0	0	$\frac{0.35T}{k}$	0.4τ

Tableau B.2. *Calcul des actions P, I, et D pour les systèmes instables.*

Choix du mode de réglage dans le cas d'un système instable :

1) 0.05<Kτ<0.1 alors P
2) 0.1< Kτ<0.2 alors PI
3) 0.2< Kτ<0.5 alors PID
4) 0.05< Kτ alors Tout ou rien
5) Kτ>0.5 multi-boucles

B. 3.2 Méthodes utilisables en boucle ouverte pour les systèmes intégrateurs

Identifié sous la forme :

$$H(P) = \frac{K_u}{P} \times \frac{1}{(1+\tau \cdot P)^n} \qquad \text{(B.47)}$$

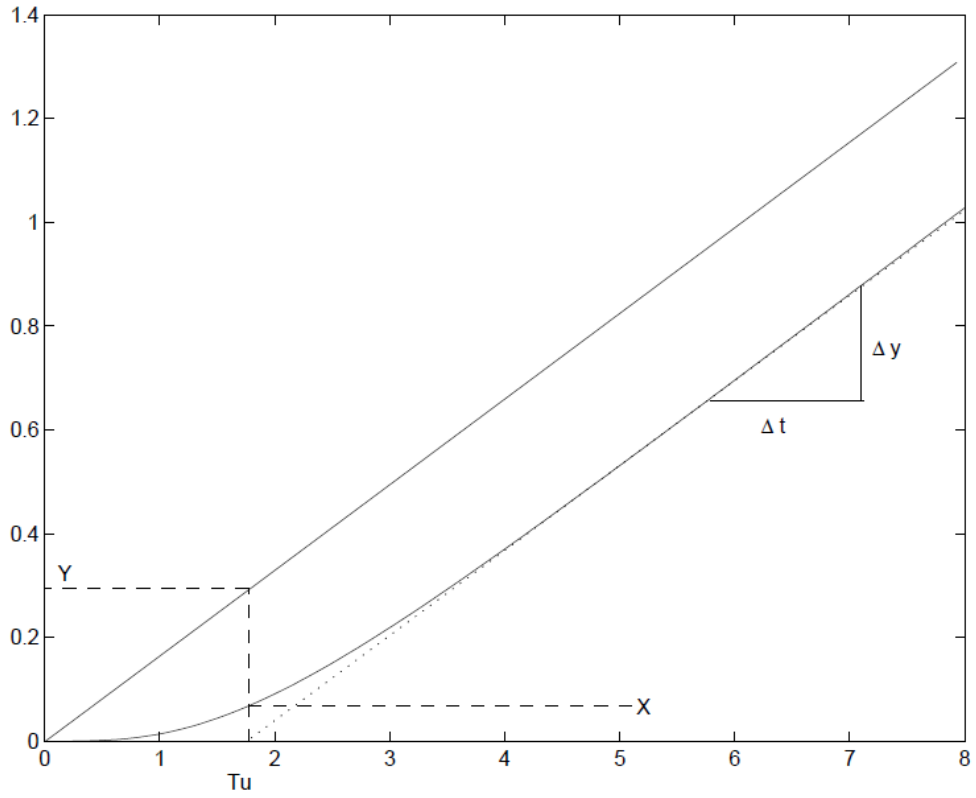

Tableau B.3. *Réponse indicielle d'un système intégrateur en boucle ouverte.*

n	$\frac{x}{y}$
1	0.368
2	0.271
3	0.224
4	0.195
5	0.175

Tableau B.4. *Détermination de l'ordre de système par la méthode de Strejc (système avec intégrateur).*

1) Avec x/y : déterminer n

2) Calcul de τ avec $\tau = T_u/n$

3) Calcul de K_u avec $K_u = (\Delta y/\Delta u) \times (1/\Delta t)$

En pratique il est assez rare d'identifier un système intégrateur en boucle ouverte car le système dérive rapidement en dehors de son domaine de linéarité (saturation, ...). En pratique, on procède à une identification en boucle fermée (GONZALO, 2009).

REFERENCES BIBLIOGRAPHIQUES

Références bibliographiques

ASTALASEVEN, & ESKIMONET, o. (2012). *Arduino pour biencommencer enélectronique et enprogrammation.* Le site de zéro. Consulté le Juin 16, 2017, sur https://www.scribd.com/doc/156564134/Arduino-Pour-Bien-Commencer-en-Electronique-Et-en-Programmation

BEN ABDENNOUR, R., BORNE, P., KSOURI, M., & M'SAHLI, F. (2001). *Identification et commande numérique des procédés industriels.* TECHNIP.

BENBOUALI, A. (2008). *commande logique floue adaptative d'une machine asynchrone avec une estimation de la constante de temps rotorique.* Thése de Doctorat, Université Hassiba Benbouali, Département d'Electrotechnique, Chlef (Algerie). Consulté le Juin 18, 2017, sur http://dspace.univ-chlef.dz:8080/jspui/handle/123456789/754

BENYOUCEF, D. (s.d.). *Cours Identification pour Master 2.* University Hassiba Ben Bouali, D'épartement d'éléctronique, Chlef (Algérie).

BERTRAND, C. (s.d.). *Laboratoire Angevin de recherche en ingénierie des systèmes.* Consulté le Juin 15, 2017, sur univ-angers: http://perso-laris.univ-angers.fr/~cottenceau/ArduinoCottenceau1112.pdf

CHABNI, F., & BOUTHIBA, M. A. (2014). *Réalisation d'une commande floue à base d'Arduino.* Mémoire de Master, Université Hassiba Benbouali de Chlef (U.H.B.C), Département d'Electrotechnique, Chlef (Algérie).

CHELLY, N., & CHARED, A. (2014). *Commande d'un système thermique à l'aide de la carte Arduino UNO.* Hammamet. Consulté le Juin 15, 2017, sur https://chellynizarblog.files.wordpress.com/2015/10/report1.pdf

CHRISTOPHE, L. (2007). *Le PID utilisé en régulation de position et/ou de vitesse de moteurs.* Consulté le Juin 17, 2017, sur https://www.abcelectronique.com/annuaire/montages/cache/3074/rapport.pdf

GONZALO, C. (2009). *Identification des systèmes.* Cours d'identification, École Nationale Supérieure de Mécanique et des Microtechniques, (France).

HARROUCHE, F. (s.d.). *Application de la logique floue sur les machines tournantes.* Thèse de Magister, Université de Farhat Abbas (Institut d'optique et mécanique de précision), Setif (Algerie). Consulté le Juin 18, 2017, sur http://www.univ-setif.dz/MMAGISTER/images/facultes/IOMP/2011/harrouche%20Fateh.pdf

Kouadri bou djelthia, s., & Kouidri zourgui, b. (2013). *Contrôle Arduino par MATLAB GUI Application système supervision de température.* Mémoire de Licence, Université Hassiba Benbouali de Chlef (U.H.B.C), Département d'Electrotechnique, Chlef.

LANDAU, I. D. (1993). *Identification et commande des systémes* (éd. 2 nd revue et augmenter). France: ERREUR PERIMES Hermès Science Publications.

LECHALUPÉ , J. (2014). *Cours d'initiation à Arduino.* University Touulouse 3. Toulouse: UPS – ASTUPS - CampusFab. Consulté le Juin 15, 2017, sur https://fablab.univ-tlse3.fr/wiki/images/9/92/Cours_arduino_v0.2.pdf

LJUNG, L. (2008). System Identification Toolbox 7: Getting Started Guide. *The MathWorks*, 204. Consulté le Juin 23, 2017, sur https://pdfs.semanticscholar.org/61b8/c0471962bcee5a49dee264a9c57d3831c5f3.pdf

MARGOlIS, M. (2011). *Arduino Cookbook.* (1. e. édition) O'Reilly Media,.

MIMOUNI, K., & BENGOUFA, A. (2005). *Conception et réalisation d'un régulateur numérique PID sur réseau MOBDUS.* Mémoire d'ingénieur, University USTO Oran, Déoartement d'automatique, Oran.

OULD SIDI Mohamed, m. l., & SERSOUB, S. (2015). *Etude et réalisation d'un PI numérique en utilisant Arduino dans un réseau local Application machine à courant continu 1 kilo watt.* Mémoire de Master, Université Hassiba Ben Bouali -CHLEF-Faculté de technologie, Département d'Electrotechnique, Chlef.

OUMAYA, M., & LIMAM, M. L. (2012). *Commande par réseaux d'ondelette-floue.* Mémoire de Master en Automatique, Université Kasdi Merbah, Département de génie électrique, Ouargla (Algerie). Consulté le Juin 18, 2017, sur https://bu.univ-ouargla.dz/master/pdf/Oumaya-Limam.pdf?idmemoire=156

POMERLEAU , A. (1997). *La commande de procédés industriels* (éd. Éditions Lavoisier). Hermés.

Technologie d'automatique. (2015, 06 Décembre). Consulté le Juin 16, 2017, sur wikimeca: http://wikimeca.org/index.php?title=Moteur_%C3%A0_courant_continu

TKOUTI, N. (2004). *Optimisation des système Photovoltaique connectés au réseau par la logique floue.* Mémoire d'ingénieur, Université Mohamed Khider, Département d'électrotechnique, Biskra (Algérie). Consulté le Juin 18, 2017, sur http://thesis.univ-biskra.dz/1151/

TLILI, A. (2017, 04 19). *arduino :histoire et plate-forme*. Consulté le Juin 15, 2017, sur High technology: https://arduisolution.blogspot.com/?view=classic

VILLAIN, M. (2007). *Système automatique linéaires* (Vol. tome 2). (ellipses, Éd.) Consulté le Juin 19, 2017.

Résumé:

Ce travail a permis l'implémentation de deux régulateurs (PI numérique, flou) embarqué dans une carte Arduino pour la régulation de niveau d'eau dans une cuve.

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. Puis nous avons utilisé ce modèle afin de trouver les paramètres du régulateur PI pour pouvoir contrôler notre système et après on a implémenté ce régulateur au système à l'aide du 'Matlab Simulink'; ensuite on fait la conception du régulateur floue à partir des essais. On a obtenu des résultats qui nous ont permis l'implémentation du contrôleur floue (à l'aide de l'outil "fuzzy logic toolbox" sous Matlab). Après l'implémentation de ce contrôleur, enfin on a fait une comparaison entre les résultats des deux régulateurs. On a conclu que le régulateur floue donne une meilleure performance que le régulateur classique PI.

Mots clés:

PI numérique, carte Arduino, régulation, identification, régulateur PI, 'Matlab Simulink', contrôleur flou, "fuzzy logic toolbox".

Abstract:

This work allowed the implementation of two regulators (digital PI, fuzzy) embedded in an Arduino board for the regulation of water level in a tank.

We began, first, with an identification of the system by estimating its transfer function to obtain a mathematical model describing its behavior. Then we used this model to find the parameters of the PI regulator to be able to control our system and afterwards we implemented this controller to the system using the 'Matlab Simulink'; Then the design of the fuzzy controller is made from the tests. We obtained results which enabled us to implement the fuzzy controller (using the fuzzy logic toolbox tool under Matlab). After the implementation of this controller, we finally made a comparison between the results of the two controllers. We concluded that the fuzzy controller gives better performance than the conventional PI regulator.

Keywords:

Digital PI, Arduino board, controlling, identification, PI controller, 'Matlab Simulink', fuzzy logic controller, "fuzzy logic toolbox".

ملخص

يسمح هذا العمل بتنفيذ اثنين من المنظمين (PI الرقمي، المضبب) الذي يعتبر جزءا لا يتجزأ من بطاقة اردوينو لتنظيم مستوى المياه في خزان.

بدأنا، أولا وقبل كل شيء، مع تحديد النظام من خلال تقدير وظيفة نقلها للحصول على نموذج رياضي يصف سلوكها. ثم استخدمنا هذا النموذج للعثور على المعلمات من منظم PI¨ لتكون قادرة على السيطرة على نظامنا وبعد ذلك قمنا بتنفيذ هذه وحدة تحكم إلى النظام باستخدام "ماتلاب سيموليننك". ثم يتم تصميم وحدة التحكم المضبب من الاختبارات التجريبية. حصلنا على النتائج التي مكنتنا من تنفيذ وحدة تحكم غامض (باستخدام أداة أداة المنطق المضبب في ماتلاب). بعد تنفيذ وحدة التحكم هذه، قمنا أخيرا بإجراء مقارنة بين نتائج المتحكمين. خلصنا إلى أن وحدة التحكم المضبب يعطي أفضل أداء من منظم PI التقليدي.

الكلمات المفتاحية :

المنظم الرقمي PI, بطاقة الأردوينو, مراقبة, المطابقة, المنظم PI, ماطلاب سيميولينك, المنطق المضبب, اداة المنطق المضبب.