الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

جامعة وهران 2 محمد بن احمد
Université d'Oran 2 Mohamed Ben Ahmed
معهد الصيانة والامن الصناعي
Institut de Maintenance et de Sécurité Industrielle
Département de Maintenance en Instrumentation

MÉMOIRE

De projet de fin d'études en vue de l'obtention du diplôme de Master

Filière : Génie Industriel

Spécialité : Génie Industriel

Thème

Commande d'un système thermique à l'aide de la carte Arduino et Matlab

Présenté et soutenu publiquement par :

KHALFI Abdelghafour

TLEMCANI Bachir

Devant le jury composé de :

Nom et Prénom	Grade	Etablissement	Qualité
RAHIEL Rachida	**MCB**	Université d'Oran 2	**Président**
DJEBLI Yamina	**MAA**	Université d'Oran 2	**Examinateur**
BENMANSOUR Souhila	**MCB**	Université d'Oran 2	**Encadreur**

Juillet 2019

Résumé

Ce projet consiste à réaliser un système de régulation de température à base de Matlab et la plate-forme Arduino. La réalisation de ce système de régulation nécessite l'utilisation de la carte Arduino comme une interface d'entrées/sorties afin de communiquer avec l'environnement Matlab ou seront implantées les boucles de régulation.

Mots-clés : ArduinoIO, PID tuning, System identification, Fonction de transfert, Capteur de température.

Abstract

This project is to provide a temperature control system based on Matlab and Arduino platform. The implementation of this control system requires the use of the Arduino board as an interface I / O to communicate with Matlab or will be implanted control loops.

Key words **:** ArduinoIO, PID tuning, System identification, Transfer function, Temperature sensor

Remerciement

On remercie dieu le tout puissant de nous avoir donné la santé et la volonté d'entamer et de terminer ce mémoire.

Tout d'abord, ce travail ne serait pas aussi riche et n'aurait pas pu voir le jour sans l'aide et l'encadrement de Mme ***BENMANSOUR Souhila****, on la remercie pour la qualité de son encadrement exceptionnel, pour sa patience, sa rigueur et sa disponibilité durant notre préparation de ce mémoire.*

Nous sommes conscients de l'honneur que nous a fait Mme ***RAHIEL*** *en étant président du jury et Mme* ***DJEBLI*** *d'avoir accepté d'examiner ce travail.*

Nos remerciements s'adressent à ***SAAD Ihsen*** *pour son aide pratique et son soutien moral et ses encouragements.*

On remercie aussi Mme ***KTAT Fatma*** *de l'université de Tunisie et notre ami l'étudiant* ***KHLIFA Fares*** *pour leurs aides et leurs encouragements.*

Nos remerciements s'adressent également à tout nos professeurs pour leurs générosités et la grande patience dont ils ont su faire preuve malgré leurs charges académiques et professionnelles.

Nos profonds remerciements vont également à tous nos amis et collègues et toutes les personnes qui nous ont aidés et soutenue de prés ou de loin, principalement à tous l'effectif de l'institut de maintenance et sécurité industrielle, université d'Oran 2.

DEDICACES

Je dédie ce travail :

Á ma chère mère

Qui a fait tant de sacrifices pour m'élever et m'instruire et qui m'a encouragé tout le long de mon parcours scolaire et académique.

Á mon cher père

Qui m'a encouragé, sué et a tant travaillé pour pouvoir m'instruire.

Á mes frères

Mon jumeau Raouf Et ma sœur qui m'a surtout soutenu sur le plan moral.

Je dédie aussi ce travail à mon frère Khaled.

Á ma grand-mère Et grand père

Tout en leur souhaitant bonne santé et longue vie.

***Á** mes chers enseignants qui m'ont dirigé et aidé et surtout soutenu.*

A tous mes amis et camarades d'études:

Lokman, Yanis, Zaghwa, Farouk, Ismail, daa'a.....

Aux étudiants GI et toute la promo 2019.

Table des matières

Liste des figures

Liste des Tableaux

Abbreviations

f.é.m : Force Electromotrice
GND : La masse (la terre)
VCC : Tension d'Entrée Analogique
OUT : Sortie
P : Proportionnel
PI : Proportionnel Intégral
PID : Proportionnel Intégral Dérivé
IDE : Integrated Development Environment
Adiosrv : Analog and Digital Input and Output Server for MATLAB
CAN : Convertisseur Analogique Numérique
CNA : Convertisseur Numérique Analogique
PWM : largeur d'impulsion modulée
SPI : Interface Série Périphérique
AREF. : Alimentation de Référence
ISP : In System Programmer : un programmateur AVR qui sert à injecter des programmes directement dans les puces AVR
ICSP : In- circuit Serial Programming

Introduction générale

Introduction générale

La température constitue une information importante dans plusieurs processus industriels, des laboratoires, dans la vie quotidienne et dans la surveillance des environnements, tels que les incubateurs dans les salles médicales. Elle intervient comme une grandeur principale dont la valeur doit être connue avec précision ou comme paramètre influant sur la qualité d'autres mesures. Sa valeur sera utilisée pour la correction ou la compensation.

Certains procèdes industriels ou biologiques favorisent des environnements de températures bien spécifiques, ainsi la régulation de température s'impose. Cette régulation passe par la mesure de température de manière continue afin de la maintenir à une certaine valeur souhaitée. Pour ce faire, plusieurs régulateurs existent et de toute évidence, le contrôleur PID reste le plus utilisé car c'est un processus simple, facile à comprendre et à mettre en œuvre de manière pratique. Le rapport coûts/ bénéfique est un des avantages qui a rendu les PID les outils de contrôle les plus fréquemment utilisés dans l'industrie. Ces régulateurs sont largement utilisés depuis les années 1980 pour la pratique de l'ingénierie de contrôle.

Dans ce projet de fin d'études, nous présentons la conception ainsi que la réalisation pratique du contrôle de la température pour un système thermique en utilisant des commandes P, PI et PID numériques. Notre choix est porté sur la commande numérique car elle est devenue très exploitée dans plusieurs applications industrielles. En effet, cette approche de commande possède plusieurs avantages, puisque la loi de commande est implémentée sur un système embarqué. Il est donc possible de modifier les paramètres du régulateur, de traiter les mesures entrées/sorties, d'ajouter une procédure de supervision du système,...etc. La commande numérique est caractérisée par : la flexibilité, l'adaptabilité et la simplicité de mise en œuvre.

Nous avons utilisé dans notre application un capteur intégré LM35 pour la mesure de température du système. Les calculs et la conversion nécessaires sont confiés à la carte Arduino, qui joue dans ce cas le rôle d'une interface, et les données sont traitées par l'environnement Matlab/Simulink en utilisant la liaison série USB. En effet, cette combinaison nous donne un outil efficace, à coût réduit, performant, innovant et extensible pour réaliser des prototypes de systèmes pluridisciplinaires.

Le plan de ce modeste travail s'articule comme suit :

- Le premier chapitre présente des généralités sur la mesure de la température par différents capteurs et quelques travaux de recherches sur le contrôle de température afin de situer notre projet dans son contexte technique et scientifique.

- Le deuxième chapitre donne d'abord une description de la carte d'acquisition de l'Arduino et plus précisément la gamme Arduino UNO qu'on va utiliser dans notre application. Une explication des différents étape à suivre pour la création d'une interface de communication Matlab/Arduino a été détaillée ensuite.

- Le troisième chapitre consiste à présenter la commande PID numérique que nous avons utilisé dans notre application et afin de mieux comprendre cette commande, nous allons d'abord donner une vue générale sur l'asservissement ainsi que la commande PID analogiques.

-Le quatrième chapitre est consacré en premier lieu à la conception et la réalisation de la maquette d'un système de régulation de température en présentant les différents composants électroniques ainsi que le logiciel utilisés. Nous allons passer ensuite à l'étude expérimentale en appliquant au système thermique des différents régulateurs afin de valider le bon fonctionnement de notre maquette.

- Enfin, nous terminons ce travail par des conclusions et des perspectives.

Chapitre I

Généralités et contexte de travail

I.1 Introduction

Dans la vie quotidienne, la notion de température est l'un des thèmes les plus abordés. Toutefois, avec l'évolution de la science et des technologies, nous avons été à plusieurs reprises confrontés à des problèmes liés à la température: dilatation des métaux, variation des résistances, problème des semi-conducteurs, variation de température dans des applications industrielles ou domestiques,...etc. Dans ce sens est né l'idée de mesurer la température. Plusieurs phénomènes font face à des variations de températures. Ainsi, de nos jours, l'on distingue plusieurs types de capteurs de température, selon le phénomène en présence. Ces mesures doivent être contrôlées, par la suite, avec précision.

Dans ce chapitre, il nous semble utile d'aborder la mesure de température par différents capteurs en présentant quelques travaux de recherches afin de situer notre application dans son contexte technique et scientifique.

I.2 Mesure de température

I.2.1 Définition de température

La température est une caractéristique qui exprime la valeur de la chaleur prise par un objet. Si ce dernier prend de la chaleur, sa température augmente; s'il dégage de la chaleur, sa température baisse. C'est une grandeur physique qui peut être contrôlée dont son unité de mesure est en degrés Celsius (°C).

Dans la vie courante, la température est reliée aux sensations de froid et de chaud, provenant du transfert de chaleur entre le corps humain et son environnement. En physique, elle se définit de plusieurs manières, comme fonction croissante du degré d'agitation thermique des particules, par l'équilibre des transferts thermiques entre plusieurs systèmes. La température est une variable importante dans d'autres disciplines.

I.2.2 Capteurs de température

La température est une grandeur physique mesurée à l'aide d'un capteur. Ce dernier est un dispositif qui permet de transformer une grandeur physique (Température) en une grandeur électrique (tension ou courant) et de la transférer vers un microcontrôleur. Sur cette base, le microcontrôleur contrôle le chauffage **[1].**

I.2.2.1 Classification de capteurs de température

Il est possible de mesurer la température de plusieurs façons différentes qui se distinguent par le coût des équipements et la précision ainsi que le temps de réponse**.**

En général, On peut classer les capteurs de plusieurs manières **[2]** :

- Par son rôle dans le processus industriel (contrôle de produit finis, de sécurité, etc)
- Par le signal qu'il fournit en sortie qui peut être numérique ou analogique.
- Par leur principe de traduction du mesurande (capteur résistif, piézoélectrique, etc)
- Par leur principe de fonctionnement : capteur Actif ou Passif

Toutes ces classifications permettent d'avoir une vue d'ensemble des capteurs et bien sur aucune des méthodes de classification n'est meilleure que l'autre car toutes présentent des avantages et des inconvénients.

a- Capteur actif de température

Ce capteur fonctionne comme un générateur; dès qu'il est soumis à l'action d'une mesurande (température) celui-ci transforme celle-ci en une grandeur directement exploitable à savoir en énergie électrique.

b- Capteur passif de température

Un capteur passif est considéré comme une impédance dont l'un des paramètres est sensible au mesurande (température). Cette impédance doit ensuite être intégrée dans un circuit pour pouvoir retrouver une grandeur électrique en sortie. Le montage qui permet ceci est appelé conditionneur. Il existe plusieurs sortes de conditionneur comme le montage potentiométrique, le pont de Wheatstone, les circuits oscillants ou les amplificateurs opérationnels.

c- Capteur intégré de température

Un capteur intégré est un capteur qui utilise la microélectronique. Ce capteur est constitué d'une plaque en silicium dans lequel on a fixé le capteur, le corps d'épreuve si besoin et d'autres composants électroniques qui peuvent servir à linéariser, amplifier, convertir le courant en tension, etc.

I.2.3 Mesure de température

La mesure de température, appelée aussi thermométrie peut être effectuée soit à distance ou par contact avec l'objet qu'on veut prendre sa température.

I.2.3.1 Mesure de température à distance

La mesure à distance ou sans contact peut avoir beaucoup d'applications mais elle est surtout utilisée pour mesurer ce qu'aucun autre capteur de contact arriverait à mesurer. C'est généralement le cas lorsque nous avons des températures trop élevées, lorsqu'il faut mesurer à très grandes distances comme la température de la lune ou tout autre astre, bien sur lorsque l'environnement est agressif. Il est aussi très utile pour mesurer des points chauds et pour mesurer la température de pièces en mouvements.

Figure I.1 : Capteur sans contacte.

I.2.3.2 Mesure de température par contact

La thermométrie par contact utilise les trois familles de capteurs cités ci-dessus (passifs, actifs et intégrés) et qui possèdent donc des caractéristiques différentes et permettent ainsi d'avoir une multitude de capteurs pour des applications variées. La thermométrie par résistance utilise des capteurs passifs alors que les thermocouples sont des capteurs actifs et bien évidemment la thermométrie par diode et transistor utilise des capteurs intégrés **[2].**

a- Thermométrie par résistance

La thermométrie par résistance utilise, comme son nom l'indique, la variation de la résistance d'un matériau en fonction de la température. Cette variation de résistance peut être faite aussi bien avec un métal (dans ce cas-là nous parlerons de résistance métallique) mais aussi avec des oxydes (dans ce cas-là nous parlerons de thermistances).

b- Thermométrie par thermocouple [3]

Les thermocouples sont des capteurs actifs qui délivrent une f.é.m (force électromotrice) lorsque ceux-ci sont soumis à une modification de la température. Il existe beaucoup de type de thermocouple qui sont pour la plupart repérée par une lettre ainsi un thermocouple de type J est constitué d'une jonction en fer et d'une jonction en constantan.

c- Thermométrie par diodes et transistors

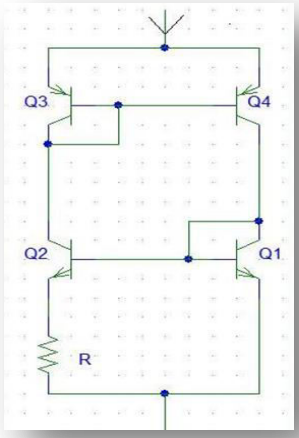
Les constructeurs proposent des capteurs de températures intégrés qui ressemblent à des transistors. Ces capteurs utilisent une propriété importante des diodes et des transistors: la tension à leurs bornes, lorsqu'ils sont traversés par un courant constant, dépend de la température. Ces capteurs ont l'avantage d'être simple à fabriquer et à mettre en œuvre, peu coûteux et très linéaire. Mais du fait de leur conception, ils ont une étendue de mesure limité (-50 °C , 150 °C) et sont affectés par un champs magnétique.

Deux exemples de capteur de température peuvent être pris comme le AD590 ou le LM 35 **[4].**

- **Capteur AD590**

Le transducteur de température AD590 génère un courant de sortie proportionnel à la température absolue. Avec une source de tension étendue de 4 V à 30 V, ce transducteur est idéal pour toutes les applications de détection de température qui requièrent une vaste plage de températures. L'AD590 est ajusté au laser pour étalonner une sortie de 298,2 µA à 298,2 K (25°C). Les dispositifs sont robustes et capables de résister à des pointes de tension directe jusqu'à 44 V et des tensions inverses de 20 V.

(a)



(b)

Figure I.2: Capteur AD590. **(a)** Photographie du AD590, **(b)** Schéma équivalent.

- **Capteur LM35**

Le capteur LM35 fait partie des capteurs de température électroniques de précision en structure intégrée. C'est un capteur analogique de température fabriqué par Texas Instruments **[5].** Il est extrêmement populaire en électronique, car précis, peu couteux, très simple d'utilisation et d'une fiabilité à toute épreuve

Les plus grandes forces du capteur LM35, qui font sa popularité sont:

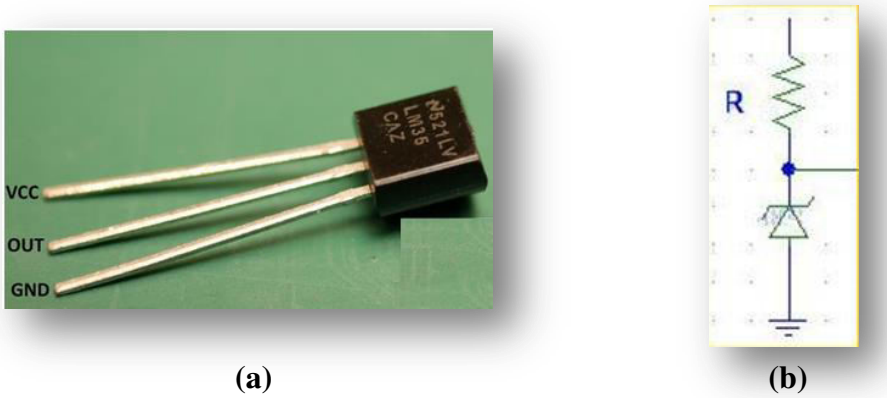
- Sa ***pré-calibration*** en sortie d'usine. Tous les capteurs LM35 sont calibrés en degré Celsius lors de la fabrication.

- Sa ***précision*** qui est garantie par le fabricant est de +/-1°C à 25°C et +/-1.5°C à -55°C ou +150°C pour la version la moins précise, ce qui est largement suffisant pour la plupart des applications. La version plus précise du LM35 (nommée "LM35A") a une précision garantie de +/-0.5°C à 25°C et +/-1°C à -55°C ou +150°C.

- Sa ***linéarité*** exemplaire: moins de 1°C d'erreur sur la plage complète de -55°C à +150°C. Comme chaque degré Celsius correspond à 10mV (soit 0.01 volt) et que la sortie du capteur est (quasi) parfaitement linéaire, convertir une mesure en température se résume à faire un bête produit en croix.

-Sa ***flexibilité de fonctionnement*** avec n'importe quelle tension d'alimentation comprise entre 4 volts et 30 volts, ce qui permet de l'utiliser dans virtuellement n'importe quel montage numérique ou analogique.

Le capteur LM35 est commercialisé dans boiter 3 broches classiques, comme illustré dans la **Figure I.3.**



(a) **(b)**

Figure I.3: Capteur LM35. **(a)** Photographie du LM35, **(b)** Schéma équivalent.

A partir de **Figure I.3,** on remarque que la capteur LM35 est constitué de trois pattes.

- **La patte VCC** du composant qui doit être branchée sur la broche 5V.
- **La patte GND** du composant doit être branchée sur une broche GND.
- La patte OUT (celle qui envoie les informations) doit être branchée sur une broche analogique.

I.3 Contrôle de température

Le contrôle de la température est un contrôle de paramètre fréquemment utilisé dans l'industrie et dans la vie quotidienne. Il est essentiel que ce contrôle soit effectué d'une manière simple et efficace, cela revient à maintenir la température ambiante d'un système à une certaine valeur souhaitée ou exigée.

Le contrôle de température peut toucher plusieurs domaines ou systèmes thermique comme dans les thermostats, dans la fabrication des produits chimiques, dans les applications industrielles qui nécessitent un contrôle très sensible de la température ambiante et surveillance des environnements tels que les incubateurs dans les salles médicales.

Le contrôle des systèmes thermiques a fait l'objet de nombreux sujets et recherches dans la littérature en utilisant différentes méthodes [6], [7] et [8]. Nous nous somme notamment intéressés, dans ce travail, par la commande de température en utilisant l'interface Arduino/Matlab (voir chapitre II). Sur cette base, nous présenterons quelques exemples précédemment réalisés avant d'aborder le contexte de travail de notre propre application.

I.3.1 Contrôle de température d'une ampoule

Le but de ce travail est de contrôler la température d'une ampoule **[9].** Cette température est mesurée à l'aide d'un capteur TMP36 qui est alimenté par une carte Arduino. Ce dernier est également utilisée pour générer la sortie numérique qui active et désactive le relais à semi-conducteurs afin d'allumer et d'éteindre l'ampoule. L'Arduino est utilisé comme une interface dont le rôle est de transférer les données acquises du capteur vers un PC muni d'un logiciel Matlab/Simulink permettant de visualiser la température de l'ampoule, de la commander en utilisant une commande Proportionnelle P et Proportionnelle Intégrale PI (voir chapitre III) et d'envoyer le signal de commande. La **Figure I.4** illustre cette expérience.

Figure I.4 : Vue de l'expérience de contrôle de température d'une ampoule.

I.3.2 Réalisation d'une chaine d'acquisition et d'asservissement de température

L'enseignant des travaux pratiques (TP) en génie électrique affronte toujours une situation problématique qui se traduit par le fait qu'il fait appel soit à des simulations, soit à des maquettes pédagogiques très spécifiques, intégrant des fonctions figées généralement non modifiables et peu extensibles.

Pour remédier à ce problème, l'auteur (enseignant) de ce travail a proposé une application pratique réalisée en collaboration avec ses étudiants de la deuxième année licence génie électrique de l'École Supérieure des Sciences et Techniques de Tunis (ESSTT). Cette application, dont la maquette est donnée par **Figure I.5**, consiste à réaliser un prototype pédagogique d'une chaine de mesure, traitement et d'asservissement de température qui sera le point de départ pour développer des séries de TP pluridisciplinaires [10]. Le choix s'est fixé sur la combinaison de l'environnement Simulink/Matlab et de la plateforme Arduino. Cette approche est très pertinente, car elle permet d'associer un puissant logiciel de simulation et une carte de prototypage permettant d'envisager des applications complexes, cela a rendu les TP plus interactifs.

Figure I.5 : Vue de la maquette d'une chaine d'acquisition et d'asservissement de empérature

I.3.3 Kit de TP de contrôle de température

Ce Kit représente une application de contrôle en boucle fermée de température d'un processus. Il est constitué d'un Arduino, une LED, deux chauffages et deux capteurs de température. La température issue des deux chauffages est ajustée pour la maintenir proche de la consigne (température souhaitée). L'énergie thermique de l'appareil de chauffage est transférée par conduction, convection et rayonnement au capteur de température. La chaleur est également transférée de l'appareil vers l'environnement. Ce kit permet l'identification de modèles et le développement de contrôleurs. Il s'agit d'un laboratoire de poche doté de logiciels en Python, MATLAB et Simulink, dans le but de renforcer la théorie du contrôle pour les étudiants. La **Figure I.6** présente un schéma de câblage de ce Kit [11].

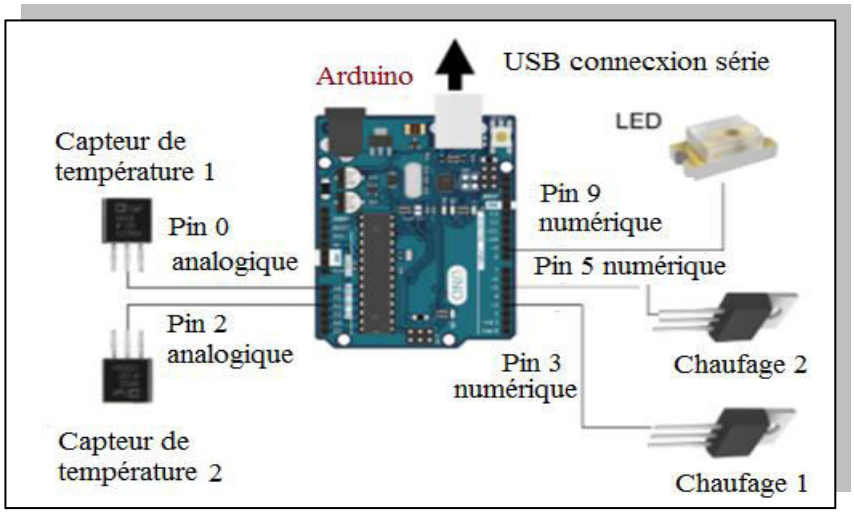


Figure I.6 : Schéma de câblage de Kit de contrôle de température

I.3.4 Modélisation et commande de température d'un four [12]

C'est un projet de fin d'études pour l'obtention du diplôme de Master en Electrotechnique, spécialité: Commande des systèmes électriques. Les auteurs de ce mémoire ont présenté la conception et la réalisation d'un prototype de contrôle de température d'un four utilisé pour le traitement thermique dans l'industrie, en utilisant des contrôleurs PI et PD (voir chapitre III). La maquette qu'ils ont réalisé est constituée d'une carte Arduino et un module de régulation de température B3510-A. L'Arduino permet d'établir la communication entre Matlab/Simulink et le module B3510. Cette communication facilité l'étude du système de régulation.

I.4 Contexte de travail

À l'heure actuelle, une grande partie de l'électricité est utilisée pour les systèmes de chauffage électrique des logements, les immeubles de bureaux ou les enceintes dans les processus de production technologique. Afin d'augmenter l'efficacité énergétique de l'utilisation de l'électricité pour le chauffage, divers systèmes de contrôle de la température sont utilisés dans les logements ou dans certaines enceintes dans les processus technologiques de production ou à l'industrie. Allant dans ce sens, le contexte de ce travail consiste à réaliser un prototype simple, facile à utiliser, performant et à un coût relativement faible permettant le contrôle de température. Pour cela:

- Nous faisons appel à l'environnement de programmation graphique Matlab/Simulink qui facilite le contrôle de la température en se servant de ses bibliothèques (toolbox control system)
- Nous utilisons la plateforme Arduino comme une carte d'acquisition ou interface permettant d'acquérir les mesures de température à partie d'un capteur intégré LM35, et d'envoyer ces données vers l'ordinateur afin de les contrôler sous Simulink.
- Nous présentons des exemples réels basés sur la simulation et la pratique qui expliquent l'avantage de la connectivité entre matériel didactique à bas coût et l'exécution des lois de commande sur un microcontrôleur.

I.5 Conclusion

Dans ce chapitre nous avons présenté quelque généralités sur les la mesure de température qui constitue un paramètre important et qui demande un contrôle adéquat. Pour cela, nous avons présenté en premier lieu les différents types de capteurs de température qui existent en s'intéressant au capteur LM35 qui sera utilisé dans notre application. Nous avons ensuite exposé le problème de contrôle de température qui est une étape crucial en donnant quelques exemples de projets de recherches qui ont étaient réalisés dans différents laboratoire. Nous avons enfin donné le contexte du travail de ce mémoire qui est basé sur une interface de communication Arduino/Matlab qui sera expliquée dans le prochain chapitre.

Chapitre II

Communication Matlab/ Arduino

II.1 Introduction

Le choix académique actuel en matière de logiciel de simulation multi-physique se porte actuellement sur Matlab. La carte Arduino assure la communication entre ce logiciel et d'autre module.

Il convient alors de s'ouvrir aux possibilités actuelles que cette solution logicielle, Matlab, permet à ce jour en matière de communication avec l'extérieur d'autant que la situation évolue de jour en jour.

Dans ce chapitre nous allons donner d'abord une description théorique sur le module Arduino et son environnement en s'intéressant à Arduino UNO qui a été utilisé dans notre projet. Nous expliquons ensuite les étapes à suivre pour la création d'une interface Matlab/Arduino.

II.2 L'univers de la carte Arduino

Arduino est un projet créé par une équipe de développeurs afin de permet aux débutants, amateurs ou professionnels de créer des systèmes électroniques plus ou moins complexes.

II.2.1 Bref historique de l'Arduino

Au début des années 2000, L'initiateur et cofondateur du projet, ***Massimo Banzi***, enseignait la physique informatique dans l'école Interaction Design Institute Ivrea (IDII) en Italie. Il apprenait aux étudiants comment utiliser l'électronique pour prototyper des objets de robotique en utilisant des cartes électroniques assez compliquées, conçues à l'origine pour des ingénieurs, et compatible uniquement avec Windows. [13]

Comme beaucoup de ses collègues, Banzi se reposait sur le BASIC Stamp [14] qui était un petit circuit, embarquant essentiellement : une alimentation, un microcontrôleur, de la mémoire et des ports d'entrée/sortie pour y connecter du matériel. Mais il n'avait pas assez de puissance de calcul, et il était relativement cher.

En 2001, ***Casey Reas*** et ***Ben Fry***, deux étudiants, avaient développé un langage de programmation 'Processing' [15] qui permettait aux programmateurs sans expérience de créer des infographies complexes dans un environnement de développement extrêmement facile à utiliser, Banzi se demanda s'il pourrait créer un logiciel similaire pour programmer un microcontrôleur, plutôt que des images sur l'écran. [13]

Un peu plus tard, en 2003, Hernando Barragan, pour sa thèse de fin d'étude, développa un prototype de plateforme, le Wiring [16], qui comprenait un environnement de développement

facile et une carte électronique prête à l'emploi. Mais Banzi pensait déjà plus grand, il voulait faire une plateforme encore plus simple, moins chère et plus facile à utiliser.

Il convia donc David Cuartielles, enseignant en Suède, a réalisé la première carte à usage pédagogique. Deux autres étudiants de Banzi se joignirent au projet, Nicholas Zambetti et David Mellis, ce dernier devient cofondateur et développeur en chef de la partie logicielle d'Arduino, et ensemble ils ont commencé à développer le logiciel en s'inspirant du langage Processing et de la carte Wiring dont l'objectif étant de mettre au point une plateforme rapide et facile d'accès [17].

Rapidement, l'histoire d'Arduino attira l'attention de Tim Igoe, connu pour être le premier à avoir mis l'ensemble de ses outils pédagogiques en ligne pour accès libre au public. Gianluca Martino a aidé à l'industrialisation de la production d'Arduino. Le premier prototype commercialisé était au cours de l'année 2005, un modèle basé sur l'ATMega8, un microcontrôleur Atmel [18] de la famille AVR.

La petite carte est désormais devenue le couteau suisse de nombreux artistes, passionnés, étudiants, et tous ceux qui rêvaient d'un tel gadget. Plus de 250 000 cartes Arduino ont été vendues à travers le monde sans compter celles construites à la maison.

II.2.2 Définition de l'Arduino

Le module ARDUINO est un circuit imprimé qui repose sur deux piliers dont le premier s'agit de la carte électronique programmable (Hardware), composée de plusieurs composants semi-conducteurs, de circuits intégrés et des périphériques, le deuxième s'agit de l'interface de programmation (Software), qui possède un langage de programmation très spécifique, basé sur les langages C et C++, adapté aux possibilités de la carte.

Chaque module Arduino possède au moins un régulateur linéaire 5V et un oscillateur à quartz 16 MHz Le microcontrôleur est préprogrammé avec un bootloader de façon à ce qu'un programmateur dédié ne soit pas nécessaire.

Les modules sont programmés au travers d'une connexion série RS-232, mais les connexions permettant cette programmation diffèrent selon les modèles. Les premiers Arduino possédaient un port série, puis l'USB est apparu sur les modèles UNO, tandis que certains modules destinés à une utilisation portable se sont affranchis de l'interface de programmation, relocalisée sur un module USB-série dédié (sous forme de carte ou de câble) , l'Arduino utilise des entrées/sorties du microcontrôleur pour l'interfaçage avec les autres circuits.

Ces cartes permettent un accès simple et peu couteux à l'informatique embarquée. De plus, elles sont entièrement libres de droit, autant sur l'aspect du code source (Open Source) que sur l'aspect matériel (Open Hardware). Ainsi, il est possible de refaire sa propre carte Arduino dans le but de l'améliorer ou d'enlever des fonctionnalités inutiles au projet. Le logiciel de l'Arduino est basé sur la puce ATMega [14].

Arduino est utilisé dans beaucoup d'applications comme l'électrotechnique industrielle et embarquée ; le modélisme, la domotique mais aussi dans des domaines différents comme l'art contemporain et le pilotage d'un robot, commande des moteurs et faire des jeux de lumières, communiquer avec l'ordinateur, commander des appareils mobiles (modélisme).

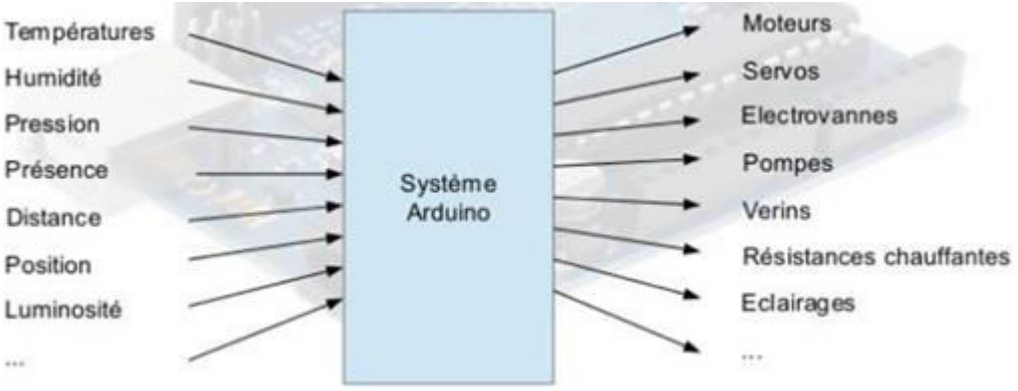


Figure II.1 : Exemples d'application de la Carte Arduino

II.2.3 Matériel Arduino

La carte Arduino repose sur un circuit intégré (un microcontrôleur) associée à des entrées et sorties qui permettent à l'utilisateur de brancher différents types d'éléments externes.

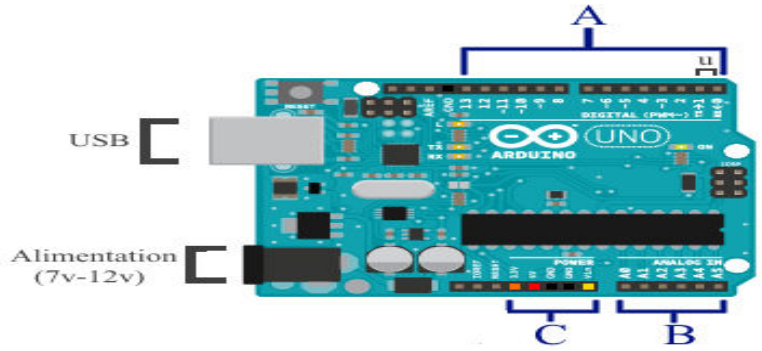


Figure II.2 : Carte Arduino

Les différentes versions des Arduino fonctionnent sous le même principe général.

Avec :

A : ce sont les broches dites numériques (0 ou 1) ou « tout ou rien » ; elles offrent en sortie du 5 V et acceptent en entrée du 5 V sur le même principe.

B : ce sont les broches dites analogiques, valeur entre 0 V et 5 V

C : les différentes broches d'alimentation : Rouge (sortie 5 V (+)), Orange (sortie 3,3 V (+)), Noire (les masses (−)) et Jaune (entrée reliée à l'alimentation (7 V-12 V))

Il y a des variations entre les différentes cartes (par exemple : UNO, la patte 13 est équipée d'une résistance)

Les différentes entrées de l'Arduino peuvent être connectées à: des capteurs qui collectent des informations sur leur environnement comme la variation de température via une sonde thermique, le mouvement via un détecteur de présence ou un accéléromètre, le contact via un bouton-poussoir, capteur qui détecte une fuite de gaz,..etc.

Figure II.3 : Différent types des capteurs pour l'Arduino

Les différentes sorties de l'Arduino peuvent être connectées à: des actionneurs qui agissent sur le mode physique telle une petite lampe, un moteur, haut-parleur,…etc.

Figure II.4: Différent actionneurs pour l'Arduino

Il existe plusieurs variétés de cartes Arduino. Ces cartes peuvent être autonome et fonctionner sans ordinateur ou servir d'interface avec celui-ci. Nous citons quelques-uns afin d'éclaircir l'évaluation de ce produit scientifique et académique [19]:

Il existe plusieurs variétés de cartes Arduino. Ces cartes peuvent être autonome et fonctionner sans ordinateur ou servir d'interface avec celui-ci.

Figure II.5 : Différente gammes de cartes Arduino

II.2.4 Logiciel Arduino

Le logiciel Arduino est gratuit (open source) et on peut le télécharger à partir du site officiel d'Arduino, à l'adresse http://Arduino.cc/en /Main/Software.

Plusieurs fichiers différents sont proposés en téléchargement. On doit faire notre choix en fonction du système d'exploitation de notre ordinateur : Windows, MacOs X, Linux.

Le logiciel Arduino et la programmation seront développés plus loin.

II.2.5 Carte Arduino UNO

C'est la carte idéale pour découvrir l'environnement ARDUINO. Elle permet à tout débutant de se lancer dans tous ses premiers petits projets.

Comme c'est la carte la plus utilisée, il est très facile de se référer aux tutoriels très nombreux sur le net et ainsi de ne pas rester seul dans son exploration.

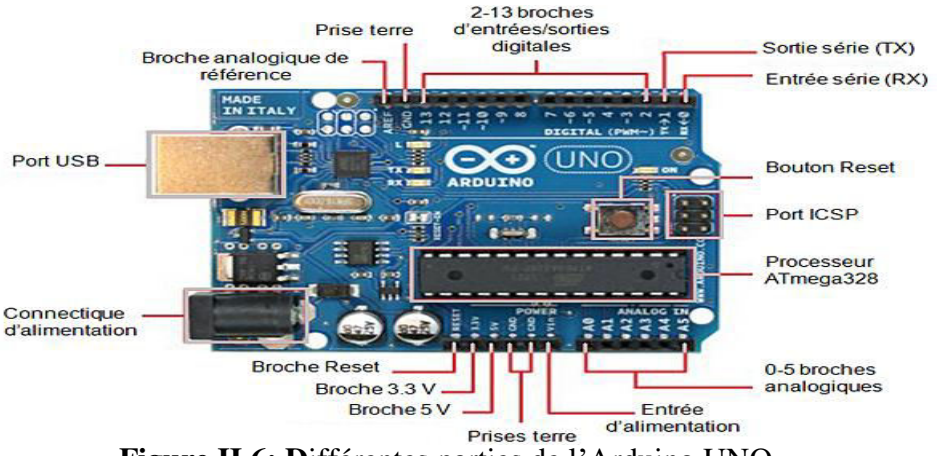


Figure II.6: Différentes parties de l'Arduino UNO

Cette carte possède la fiche technique suivante :

Microcontrôleur	ATmega328
Tension de fonctionnement	5V
Tension d'alimentation (recommandée)	7-12V
Tension d'alimentation (limites)	6-20V
Broches E/S numériques	14 (dont 6 disposent d'une sortie PWM)
Broches d'entrées analogiques	6 (utilisables en broches E/S numériques)
Intensité maxi disponible par broche E/S (5V)	40 mA (ATTENTION : 200mA cumulé pour l'ensemble des broches E/S)
Intensité maxi disponible pour la sortie 3.3V	50 MA
Intensité maxi disponible pour la sortie 5V	Fonction de l'alimentation utilisée - 500 mA max si port USB utilisé seul
Mémoire Programme Flash	32 KB (ATmega328) dont **0.5 KB** sont utilisés par le bootloader
Mémoire SRAM (mémoire volatile)	2 KB (ATmega328)
Mémoire EEPROM (mémoire non volatile)	1 KB (ATmega328)
Vitesse d'horloge	16 MHz

Tableau II.1 : Caractéristiques de l'Arduino UNO

II.2.5.1 Matériel Arduino UNO

Généralement tout module électronique qui possède une interface de programmation est basé toujours dans sa construction sur un circuit programmable ou plus.

a- Microcontrôleur ATMega328

C'est un circuit intégré (**Figure II.7**) construit autour d'un microcontrôleur Atmel (ici ATmega328) et de composants électroniques complémentaires (les transistors ; les résistances et les condensateurs) qui facilitent la programmation et l'interfaçage avec d'autres circuits.

Ce microcontrôleur peut être programmé de manière à effectuer des tâches très diverses comme la domotique (le contrôle des appareils domestiques - éclairage, chauffage…), le pilotage d'un robot, de l'informatique embarquée,...etc.

Figure II.7: Microcontrôleur ATMega328

b- Sources de l'alimentation

On peut distinguer plusieurs genres de sources d'alimentation (Entrée Sortie) et cela comme suit:

- **Entrée d'alimentation VIN**: c'est la tension d'entrée positive lorsque la carte ARDUINO est utilisée avec une source de tension externe (à distinguer du 5V de la connexion USB ou autre source 5Vrégulée). On peut alimenter la carte à l'aide de cette broche, ou, si l'alimentation est fournie par le jack d'alimentation, accéder à la tension d'alimentation sur cette broche.

- **Broche 5V**: c'est la tension régulée utilisée pour faire fonctionner le microcontrôleur et les autres composants de la carte. Cette tension peut donc provenir soit de la tension d'alimentation VIN via le régulateur de la carte, ou bien de la connexion USB (qui fournit du 5V régulé) ou de toute autre source d'alimentation régulée.

-**Broche 3V3**: c'est une alimentation fournie par le circuit intégré FTDI (circuit intégré faisant l'adaptation du signal entre le port USB d' ordinateur et le port série de l'ATMega de la carte est disponible).Ceci est intéressant pour certains circuits externes nécessitant cette tension au lieu du 5V.L'intensité maximale disponible sur cette broche est de 50mA.**[20].**

c- Entrées et sorties

- **14 broches numériques** qui peuvent être utilisées soit comme une entrée numérique, soit comme une sortie numérique.

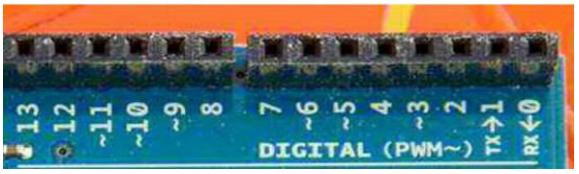


Figure II.8: Broches d'entrées/sorties numériques

- **Pins: 0 (RX) et 1 (TX);** sont utilisés pour la réception (RX) et d'émission (TX) des données en série TTL.

- **Pins 2 et 3**: peuvent être configurés comme déclencheurs pour des événements externes, tels que la détection d'un front montant ou descendant d'un signal d'entrée.

-**Pins 4, 5, 6, 9, 10 et 11:** peuvent être configurés via le logiciel avec la fonction analogWrite () pour générer des signaux PWM avec une résolution de 8 bits en l'utilisant comme CAN. On peut grâce à un simple filtre RC obtenir des tensions continues de valeur variable.

- Pins 10, 11, 12, 13: peuvent être programmés pour réaliser une communication SPI (Interface Série Périphérique), SPI utilise une bibliothèque spéciale.

- **Pin 13** est relié à une diode interne à la carte, utile pour les messages de diagnostic. Lorsque le niveau de la broche est HAUT, le voyant est allumé, quand le niveau de la broche est faible, il est éteint.

- 6 entrées analogiques

Figure II.9 : Broches d'entrées Analogiques

Ces entrées sont étiquetées de 0A à A5, dont chacun fournit 10 bits de résolution (en pratique 1024 valeurs différentes). Par défaut, on peut mesurer une tension de 5V à la masse, mais il est possible de changer l'extrémité supérieure de sa gamme en utilisant la broche AREF2. En outre, certaines broches ont des caractéristiques spéciales comme pour les broches numériques largeur d'impulsion. Il s'agit d'un artifice permettant de produire une tension variable à partir d'une tension fixe. La technique s'apparente approximativement à du morse : le signal de sortie est modulé sous forme d'un signal carré dont la largeur des créneaux varie pour faire varier la tension moyenne.

- 6 Broches pour l'alimentation

Elle comporte :

- **Une connexion Reset** pour la réinitialisation.

- **Une connexion 3.3V** qui permet à des circuits de puissance compatible de se connecter à la carte Arduino.

- **Une connexion 5V** fournit par le régulateur L7805CV. Cette tension est utile pour d'autres circuits électriques compatibles avec 5 volts.

- **Une connexion GND** pour la masse.

- **Une connexion Vin** qui renvoie la tension appliquée à partir de la prise d'alimentation et peut être utilisée pour alimenter d'autres circuits qui ont déjà un régulateur de

Figure II.10 : Les Broches d'Alimentation

- Un connecteur ICSP (programmation "in-circuit" est optionnel)

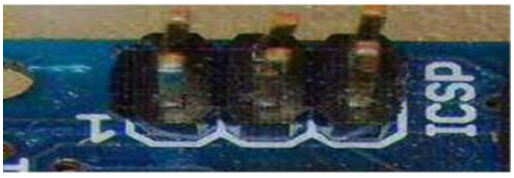


Figure II.11 : Connecteur ICSP

Dans le cas où le microcontrôleur de la carte Arduino viendrait à ne plus fonctionner, il est possible de remplacer la puce défectueuse. Ainsi le port ICSP permet de configurer une nouvelle puce pour la rendre compatible avec l'IDE Arduino. En effet le microcontrôleur de la carte Arduino possède un bootloader (ou bios) qui lui permet de dialoguer avec l'IDE. Il faut pour cela acheter un programmateur ISP2 ou il est aussi possible d'utiliser une autre carte Arduino comme programmateur ISP.

d- Circuit Additionnel

Il est possible de spécialiser la carte Arduino en l'associant avec des circuits additionnels ou modules 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. Ces circuits spécialisés apportent au système des fonctionnalités diverses et étendues dont voici quelques exemples :

- **Ethernet** : communication réseau ;
- **Bluetooth** : communication sans fil ;

- **Pilotage de moteurs** (pas à pas ou à courant continu) ;

-**Pilotage de matrices de LEDs** : pour piloter de nombreuses LEDs avec peu de sorties ;...etc.

Figure II.12 : Circuits Shields d'Arduino

II.2.5.2 Partie programme

Le logiciel de programmation des modules Arduino est une application Java, libre 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). Le langage de programmation utilisé est le C, et lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties.

Arduino IDE (Integrated Development Environment). Le logiciel est gratuit et open source dont la simplicité d'utilisation est remarquable. Ce logiciel va nous permettre de programmer la carte Arduino pour :

– Réaliser l'interfaçage avec Matlab/simulink

– Implémenter la commande directement sur la carte.

a- Interface de logiciel

Double-click sur l'icône IDE Arduino et on obtient la fenêtre vierge donnée par **Figure I.13**. Comme n'importe quel langage de programmation, *IDE ARDUINO* une interface souple et simple est exécutable sur n'importe quel système d'exploitation ARDUINO basé sur la programmation en C.

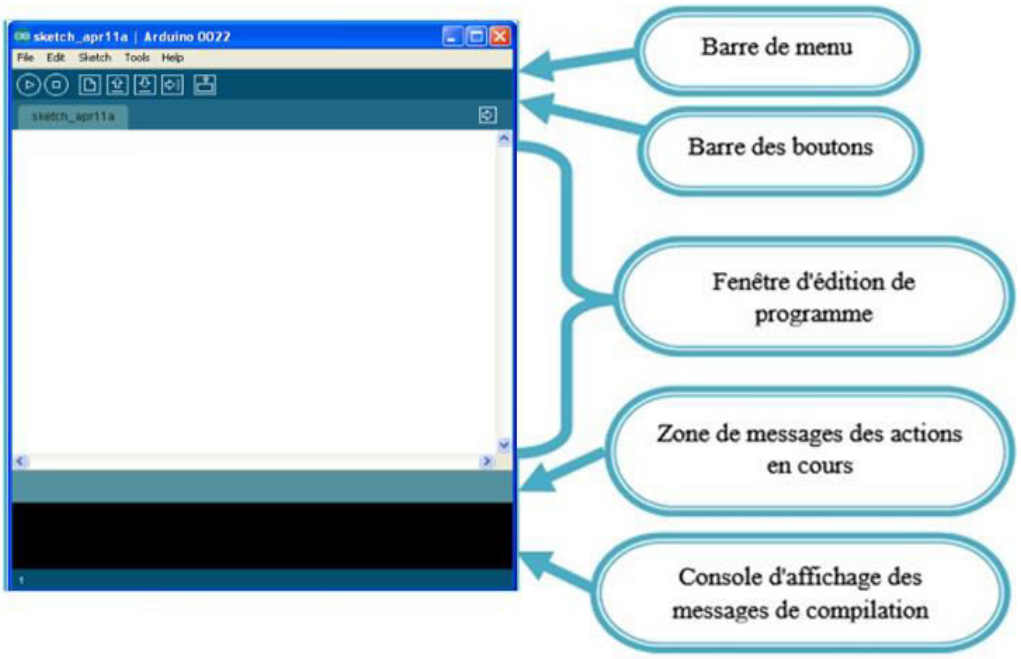


Figure II.13 : Interface de programmation d'ARDUINO

Cette fenêtre vide sera remplie de mots et de chiffres et d'autres textes. Cette fenêtre est comme tout autre logiciel contient des menus, des boutons, des alertes spéciales, et toutes sortes de contrôles : Ouvrir, Enregistrer, et le bouton du moniteur de série à l'extrême droite.

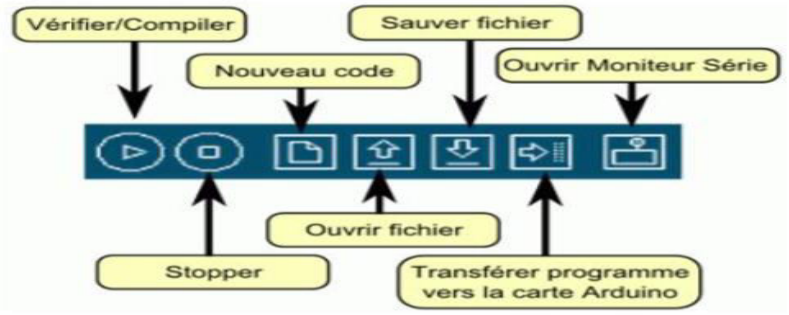


Figure II.14 : Détail de barre des boutons

b- Structure d'un programme Arduino

Le langage de programmation Arduino dérive du langage C++ et il en respecte les règles de syntaxe :

1. Une ligne qui commence par "//" est considérée comme un commentaire.
2. Un paragraphe qui commence par "/*" et qui se termine par "*/" est considéré comme un commentaire
3. Toute ligne d'instruction de code doit se terminer par un point-virgule ";"
4. Un bloc d'instructions (définition d'une fonction, boucle "while" ou "if"/"else"...) doit être délimité par des accolades ouvrantes "" puis fermantes "".
5. Toutes les variables doivent être déclarées, ainsi que leur type (int, float,...) avant d'être utilisées.

Un programme (ou "sketch") Arduino est constitué de 2 fonctions distinctes :

1. La fonction de configuration "void setup" exécutée une seule fois au lancement du programme.
2. La fonction "void loop" qui est ensuite exécutée indéfiniment en boucle.

Remarque

On peut relancer le programme en actionnant le bouton poussoir "reset" sur la carte.

Exemple (programme "Blink")

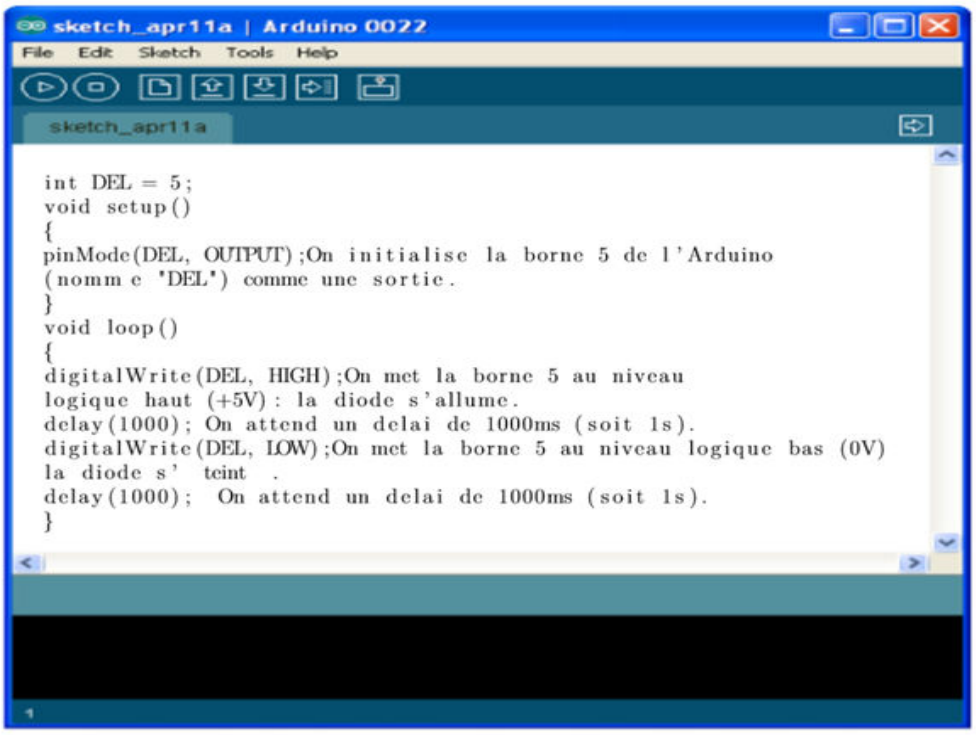


Figure II.15: Exemple de programme Blink

c- Etapes de téléchargement du programme

Une simple manipulation enchaînée doit être suivie afin d'injecter un code vers la carte ARDUINO via le port USB.

- On conçoit ou on ouvre un programme existant avec le logiciel IDE ARDUINO.
- On vérifie ce programme avec le logiciel ARDUINO (compilation)
-Si des erreurs sont signalées, on modifie le programme
- On charge le programme sur la carte.
- On alimente la carte soit par le port USB, soit par une source d'alimentation autonome (pile 9 volts par exemple).
- On vérifie si notre montage fonctionne.

II.2.6 Avantages de l'Arduino

Le système ARDUINO simplifie la façon de travailler avec les microcontrôleurs tout en offrant plusieurs avantages cités ci-dessus :

- **Prix réduit** : les cartes ARDUINO sont relativement peu coûteuses comparativement aux autres plates-formes
- **Multiplateforme :** le logiciel ARDUINO, écrit en JAVA, tourne sous les systèmes d'exploitation Windows, Macintosh et Linux, servant d'éditeur de code et de compilateur, et qui peut transférer le programme au travers de la liaison série (RS232, Bluetooth ou USB selon le module). La plupart des systèmes à microcontrôleurs sont limités à Windows.
- **Logiciel et matériel open source et extensible :** le logiciel ARDUINO et le langage ARDUINO sont publiés sous licence open source, disponible pour être complété par des programmateurs et les cartes ARDUINO sont basées sur les Microcontrôleurs ATMEL dont les schémas des modules sont publiés sous une licence et les concepteurs des circuits expérimentés peuvent réaliser leur propre version des cartes ARDUINO, en les complétant et en les améliorant.
- **Environnement de programmation clair et simple :** l'environnement de programmation ARDUINO (logiciel ARDUINO IDE) est facile à utiliser pour les débutants, tout en étant assez flexible pour que les utilisateurs avancés puissent en tirer profit également. il est à noter aussi qu'il y a de nombreuses librairies disponibles avec diverses fonctions implémentées.
- **Nombreux conseils, tutoriaux et exemples en ligne** (forums, site personnel etc...)

-**Existence de « shield » :** ce sont des cartes supplémentaires qui se connectent sur le module Arduino pour augmenter les possibilités comme par exemple : afficheur graphique couleur, interface Ethernet, GPS, ...etc.

II.3 Environnement Matlab/Simulink

Matlab/Simulink est un logiciel de calcul mathématique pour les ingénieurs et les scientifiques créé par Mathworks [21] [22].

MATLAB est un environnement de programmation pour le développement d'algorithme, d'analyse de données, de visualisation, et de calcul numérique. En utilisant MATLAB, la résolution des problèmes de calcul complexes se fait plus rapidement qu'avec des langages de programmation traditionnels, tels que C, C++, et le Fortran.

SIMULINK est un environnement pour la simulation, il fournit un environnement graphique interactif et un ensemble de bibliothèques de bloc qui permettent de concevoir, simuler, mettre en application, et examiner une variété de systèmes, tel que les systèmes de communications, de commandes, de traitement des signaux, de traitement visuel et d'image.

II.4 Interfaçage Arduino avec Matlab/ Simulink

Il existe trois possibilités d'interfacer la carte Arduino avec Matlab/Simulink, à savoir :

1. Programmation de la carte Arduino Uno comme une carte d'interface.
2. Utilisation du package ArduinoIO.
3. Utilisation du package Arduino Target.

II.4.1 Programmation de la carte Arduino Uno comme une carte d'interface

Cette solution consiste d'une part à utiliser les fonctions offert par le langage Arduino qui permet d'envoyer et d'acquérir des données binaires via le port série (USB) et d'autre part à développer sous Simulink un programme pour traiter ou visualiser ces données.

II.4.1.1 Configuration de la carte Arduino UNO

Les fonctions Arduino permettant cette configuration sont les suivantes :

- **Serial** : Cette fonction est utilisée pour la communication entre le la carte Arduino et un ordinateur ou un autre dispositif. Toutes les cartes Arduino ont au moins un port série (également connue sous le nom d'UART ou USART). Serial, communique sur les pins (0: RX) et 1 :(TX)) avec l'ordinateur par l'intermédiaire d'USB.

- **available()** : Permet d'obtenir le nombre de bit (caractères) disponibles pour lire du port série. Ces données sont stockées dans le buffer qui peut sauvegarder 64 bit.
- **read() :** Permet la lecture des bits entrants sur le port série (acquisition des données).
- **write()** : Permet l'écriture des bits sur le port série.(envoie des données)

Le programme suivant assure l'échange de données via le port série (USB) :

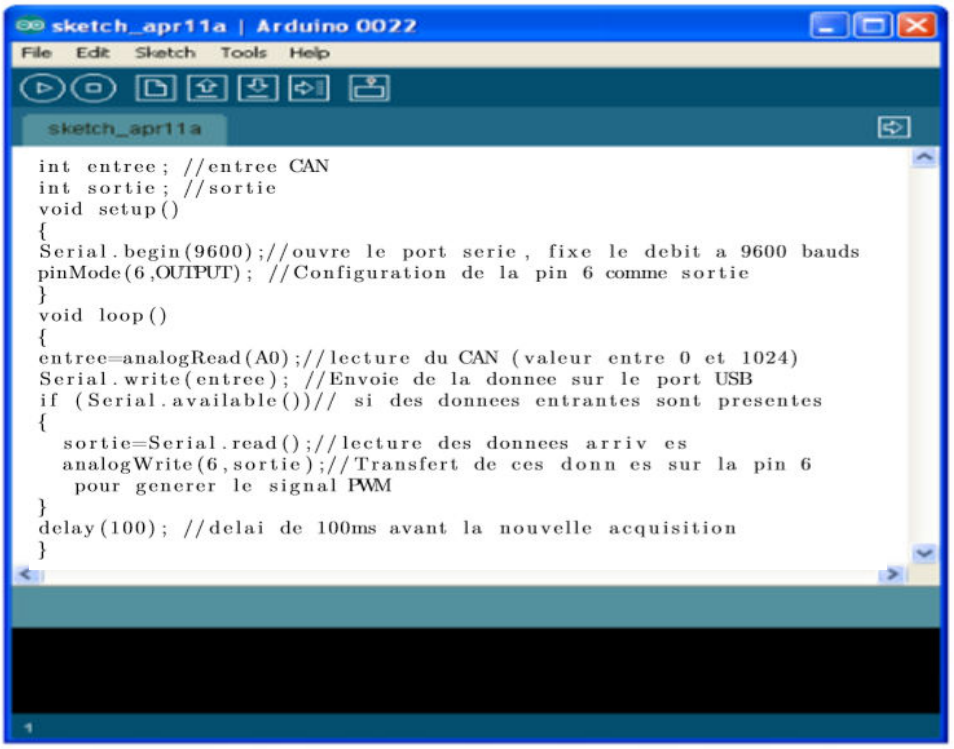


Figure II.16: Exemple de programme Configuration de la carte Arduino UNO

II.4.1.2 Traitement des données sous Simulink

La bibliothèque Instrument Control Toolbox offre les blocs qui permettent l'échange des données binaires.

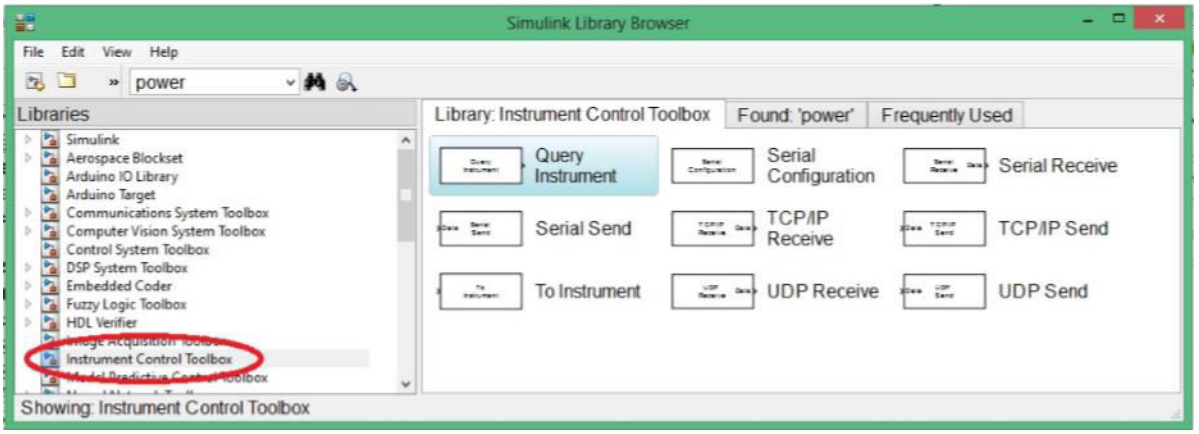


Figure II.17: Emplacement de la bibliothèque "Instrument Control Toolbox"

Ces blocs sont les suivants :

- **Serial Configuration :** Configuration des paramétrés du port série.
- **Serial Send :** Envoie des données binaires via le port série.
- **Serial Receive :** Acquisition des données binaires via le port série.

Les paramètres à configurer sont :

- **Communication Port:**

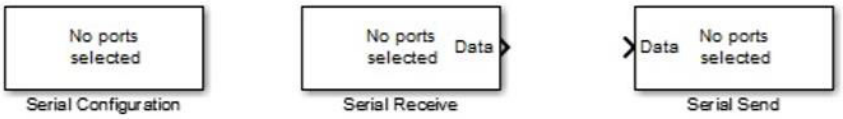


Figure II.18: Blocs pour la communication série.

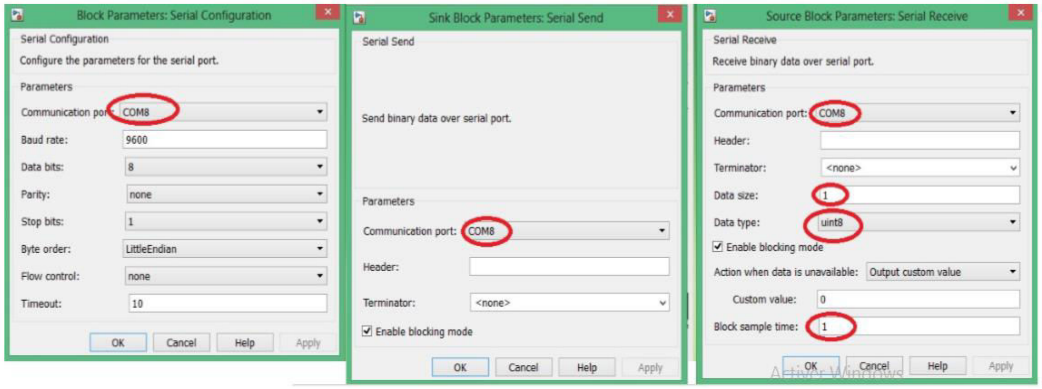


Figure II.19: Paramétrage des blocs pour la communication série

- **Data size**
- **Data type**
- **Block sample time**

Exemple II.1 Acquisition et d'envoie sous Simulink

Figure II.20 : Acquisition et envoie des données sous Simulink

II.4.2 Arduino IO

Cette solution consiste à utiliser la carte Arduino comme une interface d'entrées (Analog Input)/sorties (Analog/Digital Output). Ce package permet de communiquer Matlab ou Simulink avec la carte Arduino via un câble USB.

Elle consiste à pré-charger un programme dans la carte Arduino afin que celle-ci fonctionne en serveur : ce programme consiste à "écouter" les requêtes envoyées via la liaison série (USB) et de répondre à ces requêtes en renvoyant l'état d'une entrée ou en modifiant l'état d'une sortie. Ces mêmes entrées/sortie sont vues dans Matlab comme des entrées logiques ou analogiques (utilisation du CAN) ou des sorties analogiques (mode PWM).

II.4.2.1 Préchargement du programme dans la carte Arduino

Le préchargement du programme dans la carte Arduino se fait selon les étapes suivantes :

1. Télécharger le package ArduinoIO
2. Décompresser à la racine de votre disque dur, exemple E :\arduinoio
3. Ouvrir le dossier décompressé.
4. Aller vers : "ArduinoIO\pde\adiosrv"
5. Charger le fichier adiosrv.pde vers le logiciel Arduino.
6. Televerser: compiler et convertir en langage machine.

La carte Arduino UNO est maintenant configurée pour être utiliser comme une carte d'interface Entrées/Sorties.

II.4.2.2 Installation du package Arduino IO

L'installation du package Arduino IO se fait selon les étapes suivantes :

1. Lancer Matlab version 2013 et placer vous dans le répertoire E : \arduinoio.
2. Exécuter la commande : install-arduino
3. Fermer et relancer Matlab puis Simulink
4. Dans les bibliothèques se trouvent maintenant les blocs dans Arduino IO library

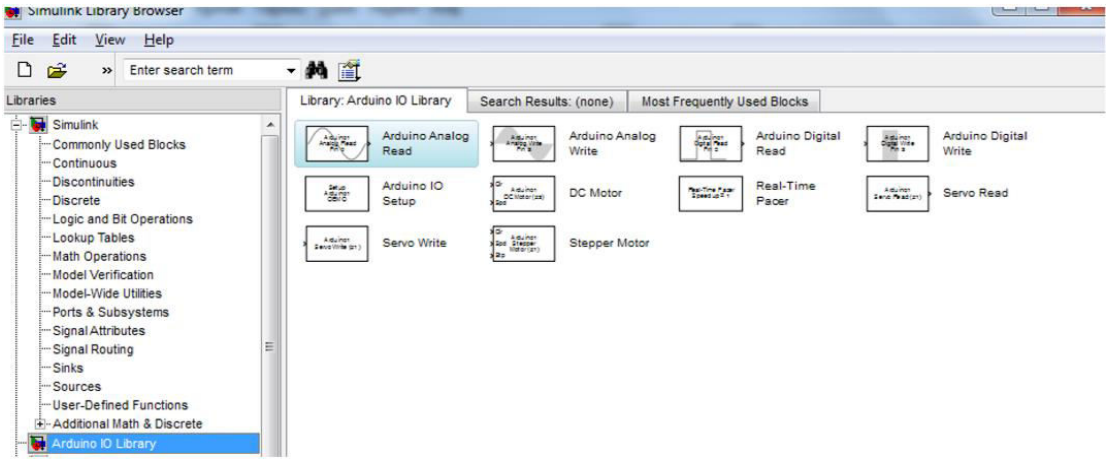


Figure II.21 : ArduinoIO Library

Il suffit alors d'utiliser les composants de la bibliothèque pour synthétiser ce que l'on souhaite réaliser.

II.4.2.3 Exploitation de la bibliothèque ArduinoIO sous Simulink

Les blocs nécessaires pour notre objectif d'asservissement sont les suivants :

Figure II.22: Blocs d'ArduinoIO nécessaires pour la commande

– Real-Time Pacer : Ce bloc permet de ralentir le temps de simulation de sorte qu'il synchronise avec le temps réel écoulé. Le coeffecient de ralentissement est contrôlable par l'intermédiaire du paramètre Speedup.

– Arduino IO Setup : Pour configurer sur quel port la carte Arduino UNO est connectée. Pour cela il suffit de voir dans Gestionnaire des périphériques.voir Figure 4.

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

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

II.4.2.4 Exploitation du package ArduinoIO sous Matlab

Le package ArduinoIO offre une panoplie de commandes permettant d'écrire un programme sous Matlab (M-file). Pour accéder à ces commandes il faut créer un objet Arduino dans l'espace de travail et spécifier le port sur lequel la carte arduino est connectée avec la commande

>> a = arduino('port');

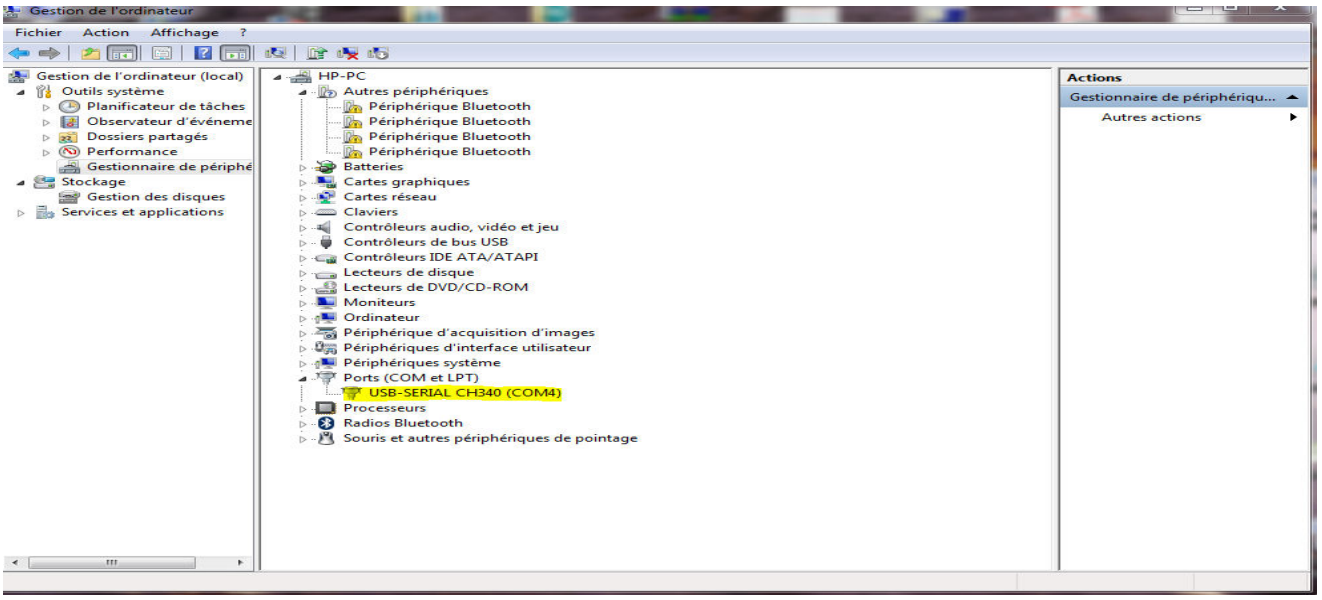


Figure II.23: Emplacement COM de la carte Arduino UNO

Parmi les commandes qui sont accessibles on retrouve :

– *pinMode*

Exemple : a.pinMode(11,'output') // configurer la pin 11 comme sortie.

– *digitalRead*

Exemple : val=a.digitalRead(4) ; // lecture de l'etat de la pin 4

– *digitalWrite*

Exemple : a.digitalWrite(13,0) ; // mettre la pin 13 à l'etat bas 0V

– *analogRead*

Exemple :val=a.analogRead(0) ; // lecture de la pin 0 de l'ADC

– *analogWrite*

Exemple :a.analogWrite(3,10) ; // envoyer sur la pin 10 un signal PWM de rapport cyclique 10/255

II.4.3 Arduino Target

Embedded Coder Support Package for Arduino permet de créer des applications Simulink qui vont fonctionner de façon autonome sur la carte Arduino. on dit que la carte Arduino est

devenue une cible (Target) et elle peut fonctionner d'une façon autonome (sans avoir recours Matlab/Simulink).

Dans la suite, on utilisera les blocs Simulink offert par le package ArduinoIO Library et la librairie Instrument Control Toolbox pour l'acquisition et l'envoie des données.

II.4.4 Acquisition et envoie des données

II.4.4.1 Présentation de l'ADC

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.

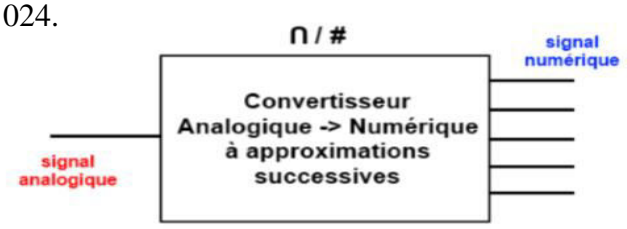


Figure II.24: Type du CAN de la carte Arduino UNO

Il n'est pas nécessaire d'initialiser ces entrées analogiques qui n'ont que cette seule fonction. La syntaxe de l'instruction permettant d'acquérir l'entrée analogique est la suivante :

analogRead(pin);

– pin : le pin sur laquelle on souhaite acquérir le signal analogique.

II.4.4.2 Acquisition de données

L'acquisition des données se faite en suivant les étapes données ci-dessous :

1. Branchement avec la carte Arduino UNO qui se fait selon le capteur utilisé. Ce paragraphe sera expliqué au chapitre 4.

2. Exploitation du package ArduinoIO Libraray, cela revient à faire :

-Un pré-chargement de adiosrv.pde sur la carte Arduino UNO

- Développement du modèle Simulink correspondant au montage utilisé

3. Exploitation de Instrument Control Toolbox, cela consiste à :

- Pré-programer la carte Arduino UNO ;
- Développer du modèle Simulink :

II.4.4.3 Envoie des données

a- Présentation des sorties analogiques (mode PWM)

La carte Arduino Uno dispose de 6 sorties (3, 5, 6, 9,10 et 11) qui peuvent être utilisées en mode PWM, c'est-à-dire en modulation de largeur d'impulsion. Ce sont des signaux logiques binaires de fréquence constante (500Hz) mais de rapport cyclique variable**.**

Figure II.25 : Description du signal PWM

Lorsqu' une lampe est alimenté par ce type de tension, tout se passe comme si il était alimenté par une tension continue ajustable entre 0V (rapport cyclique= 0) et 5V (rapport cyclique=255). Ces sorties doivent être initialisées comme des sorties digitales.

$$Vout = Vs \times \tau o/\tau c \qquad (II.1)$$

Avec : τ_c = 2ms

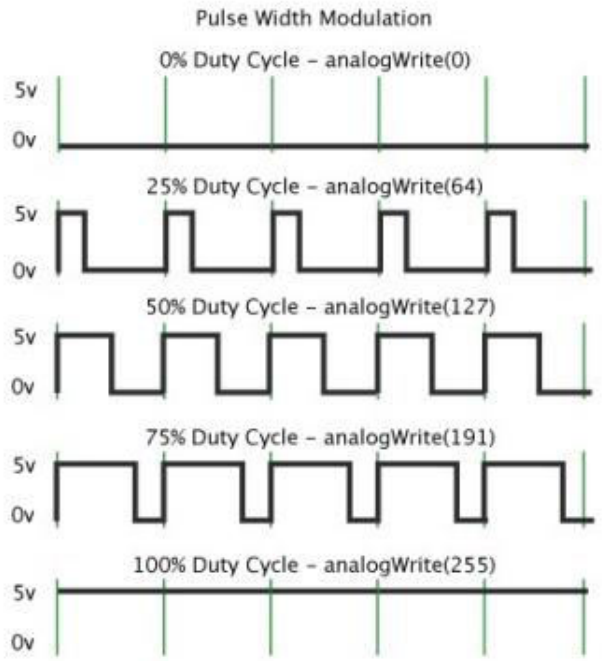


Figure II.26 : Exemples de variation du rapport cyclique

La syntaxe de l'instruction permettant de générer le signal PWM est la suivante :

analogWrite(pin, valeur) ;

– pin : la pin sur laquelle on souhaite envoyer le signal (3,5,6,9,10 ou 11).
– valeur : le rapport cyclique entre 0 et 255.

II.5 Conclusion

Dans ce chapitre nous avons expliqué la possibilité d'utiliser la carte Arduino comme une interface assurant la communication entre le logiciel Matlab/Simulink et d'autre module qui est un capteur de température dans notre cas.

Pour cela, nous avons projeté d'abord la lumière sur la carte d'acquisition l'Arduino et plus précisément la gamme Arduino UNO qu'on va utiliser dans notre application en détaillant la partie matérielle et la partie de programmation.

Nous avons expliqué enfin en détaille les différentes étapes qui permettent d'assurer l'acquisition et l'envoie des données entre Arduino et Matlab afin de visualiser l'évolution des données et de les contrôler par la suite.

Chapitre III

Commande PID

III.1 Introduction

La commande numérique des systèmes est devenue très exploitée dans plusieurs applications industrielles car elle a l'avantage d'être implémentée sur un système embarqué comme notre prototype. Il est donc possible de modifier les paramètres du régulateur, de traiter les mesures entrées/sorties, d'ajouter une procédure de supervision du système, ...etc.

Dans ce chapitre, nous allons présenter la commande PID numérique que nous avons utilisé dans notre application et afin de mieux comprendre cette commande, nous allons d'abord donner une vue générale sur l'asservissement ainsi que la commande PID analogique.

III.2 Asservissement analogique

III.2.1 Définition

L'asservissement est une partie de l'automatique dont l'objet principal est d'élaborer la commande d'un procédé afin de lui faire atteindre une grandeur physique souhaitée (voir Figure III.1). Le principe général est de comparer la consigne et l'état du système de manière à le corriger efficacement [23]. On parle également de système commandé par rétroaction négative ou en boucle fermée.

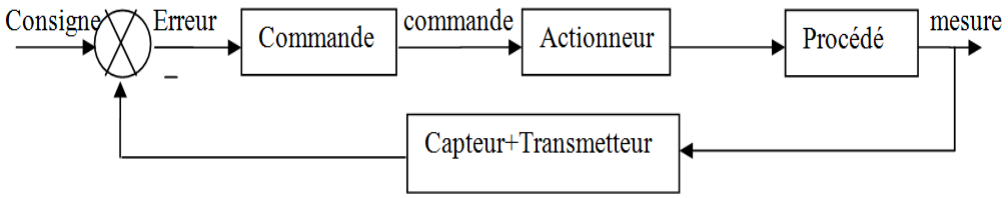


Figure III.1: Schéma de principe d'une boucle de régulation.

La commande, autrement dit régulateur, correcteur ou contrôleur, constitue l'organe principal dans la boucle de régulation. Il peut être classique comme il peut être robuste.

Dans la suite de ce paragraphe, nous allons présenter le contrôleur classique de type PID avec ses différentes actions ainsi que ses différentes formes.

III.2.2 Performances des systèmes asservis

Tout système asservi ou régulé doit posséder des performances. Celles-ci peuvent être résumées en trois points : la précision, la stabilité et la rapidité [24].

III.2.2.1 Rapidité des systèmes

Chercher à obtenir des systèmes asservis avec une réponse rapide par rapport aux variations de la consigne est une aptitude aide à effacer rapidement les perturbations.

Le temps de réponse à 5% de la réponse du système donne une bonne évaluation de sa rapidité. Cela exprime le temps mis par le processus soumis à un échelon pour atteindre sa valeur finale en régime permanent à ±5% près (et y rester). La F**igure III.2** illustre la mesure du temps de réponse à 5% pour un système 1er et un système 2ème ordre.

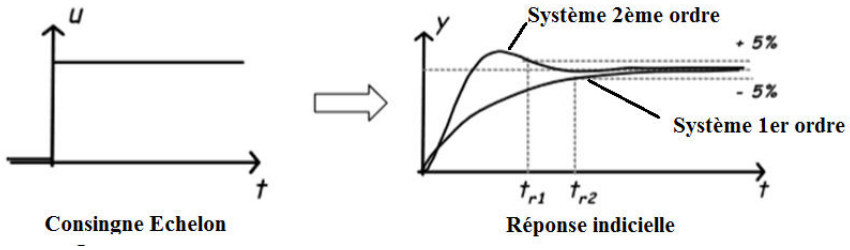


Figure III.2 : Illustration de la mesure du temps de réponse t_r à 5%.

Le temps de réponse t_r est lié à l'inertie du système considéré, c'est-à-dire à ses propriétés physiques (réaction rapide d'un micro moteur, dynamiques lentes d'un réacteur nucléaire). Elles ne sont pas modifiables.

III.2.2.2 Précision

Estimer la précision d'un système asservi c'est mesurer ou prédire l'évolution temporelle de l'écart entre la consigne d'entrée et la sortie du système. Le but étant de minimiser cette erreur statique. Plus l'écart entre ces grandeurs est petit, plus l'asservissement est précis. De même, la précision peut être étudiée vis-à-vis des perturbations dans le cas d'une boucle de régulation où il s'agit d'évaluer cet écart, suite à l'effet des perturbations.

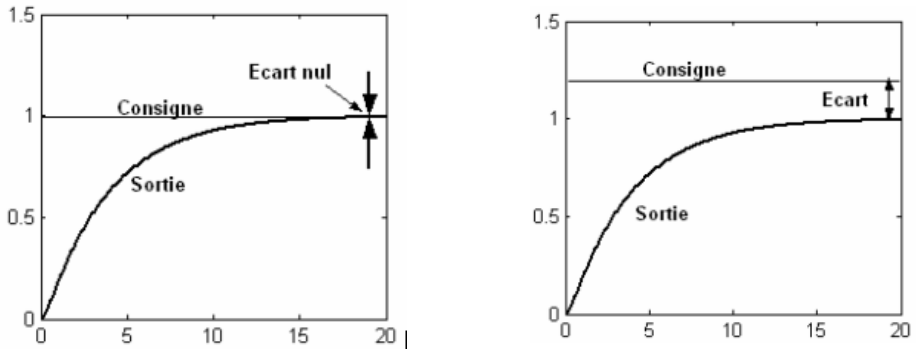


Figure III.3 : Evaluation de l'erreur statique d'un système asservi. **(a)** Précis, **(b)** Pas précis.

III.2.2.3 Stabilité

On dit qu'un asservissement est stable si pour une consigne bornée en amplitude, tous les signaux de sortie sont aussi bornés en amplitude.

La **Figure III.4** monte les réponses indicielles de trois systèmes dont la réponse 1 et 2 sont stable tan disque la réponse 3 est instable et elle diverge

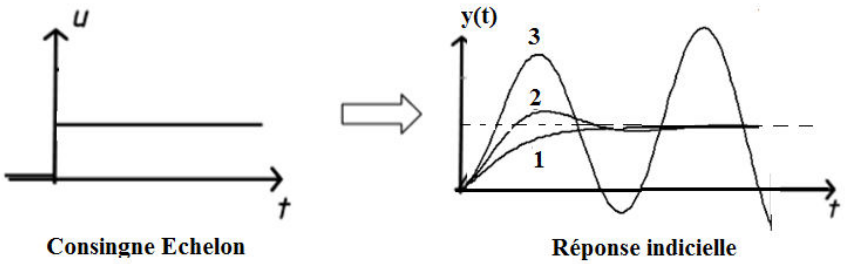


Figure III.4 : Evaluation de la stabilité d'un système asservi.

Un système asservi peut être instable s'il est dimensionné de manière incorrecte. Il est par conséquent important de s'assurer de la stabilité avant toute mise en marche : une boucle instable est une boucle inutilisable.

III.3 Commande PID

III.3.1 Définition

Les contrôleurs (régulateurs) Proportionnel-Intégral-Dérivé (PID) sont des organes de contrôle permettant d'effectuer une régulation en boucle fermée d'un système industriel. Ce sont les contrôleurs les plus utilisés dans l'industrie et permettent de contrôler la grande majorité des procédés.

III.3.2 Bref historique de la régulateur PID

Les régulateurs PID se révèlent suffisants pour résoudre un grand nombre de problèmes de contrôle surtout lorsque la dynamique du système n'est pas conditionnée et que les exigences en terme de performances ne sont pas exigées. Les régulateurs PID répondent à plus de 95% des besoins industriels. Malgré l'expérience acquise au fil des ans, les valeurs choisies, pour les paramètres P, I et D, ne sont pas toujours satisfaisantes, ni adaptées au processus à régler.

L'histoire des régulateurs est déjà longue et il peut être intéressant de rappeler quelques étapes importantes [24]. Les premiers régulateurs de type centrifuge apparaissent vers 1750 pour régler la vitesse des moulins à vent, suivi en 1788 du fameux contrôleur de vitesse d'une machine à vapeur de James Watt. En 1942, Ziegler et Nichols ont proposé deux démarches permettant de trouver facilement les paramètres optimums pour une installation donnée.

Au fil des ans, les propositions de Ziegler et Nichols ont été adaptées ou modifiées selon les besoins. En 1963, Horowitz a ajouté un degré de liberté supplémentaire au régulateur PID afin de mieux contrôler les dépassements obtenus lors d'une réponse indicielle. Ce nouveau degré de liberté consiste, en particulier, à ne réinjecter vers le terme proportionnel qu'une partie du signal de sortie.

Au début des années 1990 et dans le but de fournir des règles d'ajustement simples mais plus performantes que celles de Ziegler-Nichols, Åström et ses collaborateurs ont analysé le comportement dynamique d'un grand nombre de processus. Cette analyse a conduit à l'établissement de tableaux servant aux calculs des paramètres P, I et D à partir de mesures simples.

Actuellement les études se focalisent de plus en plus sur des techniques de l'amélioration des spécifications de la commande.

III.3.3 Principe de fonctionnement de la boucle de régulation

Un contrôleur PID calcule une valeur "d'erreur" comme étant la différence entre une variable de processus mesurée et un point de consigne souhaité. Le contrôleur tente de minimiser cette erreur en ajustant les entrées de contrôle de processus [25].

Un contrôleur PID est composé d'un élément proportionnel, d'un élément intégral et d'un élément dérivé, tous trois connectés en parallèle en série ou mixte. La **Figure III.5** montre un schéma d'un système avec un contrôleur PID. Ce dernier compare la valeur de processus mesurée avec une valeur de consigne de référence. La différence ou erreur est ensuite traitée pour calculer une nouvelle entrée de processus. Cette entrée essaiera de ramener la valeur de processus mesurée au point de consigne souhaité.

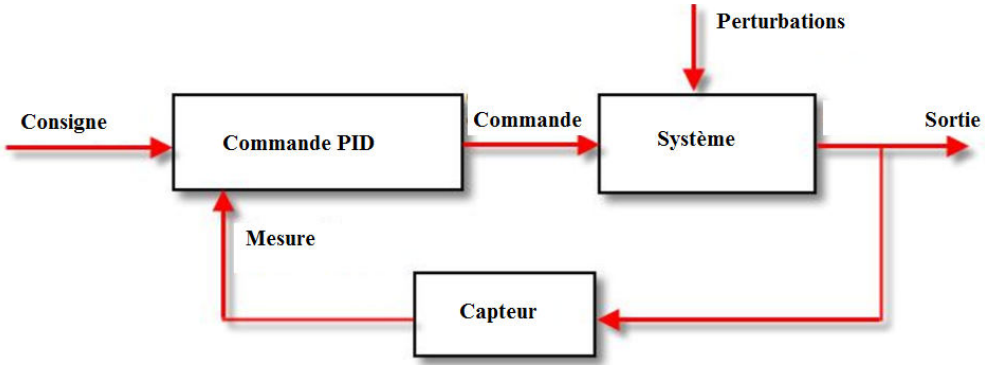


Figure III.5: Asservissement d'un système en utilisant la commande PID.

Le contrôleur PID est toujours utilisé en boucle fermée afin d'améliorer les performances du système [26].

Contrairement à un simple algorithme de contrôle proportionnel, le contrôleur PID est capable de manipuler les entrées de processus en fonction de l'historique et du taux de changement du signal. Cela donne une méthode de contrôle plus précise et plus stable [27]. L'idée de base est que le contrôleur lit l'état du système par un capteur. Ensuite, il soustrait la mesure d'une référence souhaitée pour générer la valeur d'erreur.

III.3.4 Commande PID analogique

Les contrôleurs PID analogiques sont des organes de contrôle permettant d'effectuer une régulation en boucle fermée d'un système en temps continu.

III.3.4.1 Commande proportionnelle P analogique

La commande P est un gain proportionnel qui joue le rôle d'un amplificateur appliqué au signal d'erreur de processus et qui influe sur la sortie de l'asservissement. La **Figure III.6** montre le schéma bloc du régulateur P.

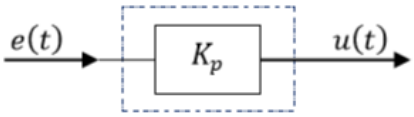


Figure III.6 : Schéma bloc de la commande P proportionnelle analogique

- La loi de commande du gain proportionnel est :

$$u(t) = K_p.e(t) \tag{III.1}$$

- La fonction de transfert du correcteur est donc :

$$C_p(p) = \frac{U(p)}{E(p)} = K_p \tag{III.2}$$

Avec :

$u(t)$: La commande.

K_p : Gain proportionnel.

$e(t)$: Signal d'erreur de processus (la consigne – la mesure).

III.3.4.2 Commande proportionnelle Intégrale PI analogique

L'action intégrale 'I' est en général associée au correcteur proportionnel 'P' dont la commande résultante est un proportionnel intégrale 'PI'. La **Figure III.7** illustre le schéma bloc correspondant au PI.

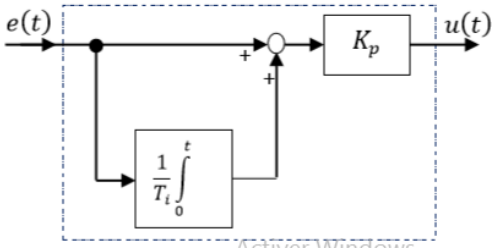


Figure III.7 : Schéma bloc de la commande PI Proportionnelle Intégrale analogique.

- La loi de commande du PI est :

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau\right) \quad \text{(III.3)}$$

- La fonction de transfert du correcteur est donc :

$$C_{PI}(p) = \frac{U(p)}{E(p)} = K_p\frac{1+T_i p}{T_i p} \quad \text{(III.4)}$$

Ou encor :

$$C_{PI}(p) = \frac{U(p)}{E(p)} = \frac{\frac{K_p}{T_i} + K_p}{p} \quad \text{(III.5)}$$

III.3.4.3 Commande proportionnelle Dérivée PD analogique

L'action dérivée 'D' est en général associée au correcteur proportionnel 'P' dont la commande résultante est un proportionnel Dérivé 'PD'. La **Figure III.8** illustre le schéma bloc correspondant au PD.

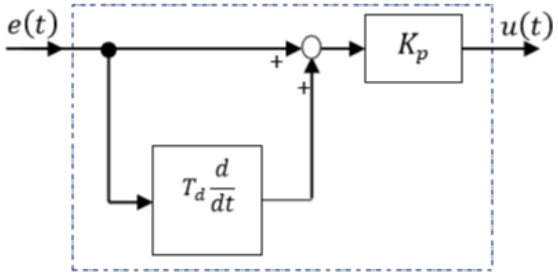


Figure III.8: Schéma bloc de la commande PD Proportionnelle Dérivée analogique.

- La loi de commande du PD est :

$$u(t) = K_p\left(e(t) + T_d\frac{de(t)}{dt}\right) \quad \text{(III.6)}$$

La fonction de transfert du correcteur PD est donc :

$$C_{PD}(p) = \frac{U(p)}{E(p)} = K_p(1+T_d p) \quad \text{(III.7)}$$

III.3.4.4 Commande proportionnelle Intégrale Dérivée PID analogique

L'association des trois actions Proportionnelle P, Intégrale I et Dérivée forme le régulateur PID dont l'intérêt est d'intégrer les effets positifs des trois correcteurs précédents.

Plusieurs implémentations des contrôleurs de ce type sont possibles, à savoir : structure série, structure parallèle et structure mixte.

a. Structure PID série

Le schéma bloc correspondant à l'architecture de PID série est donné par **Figure III.9**

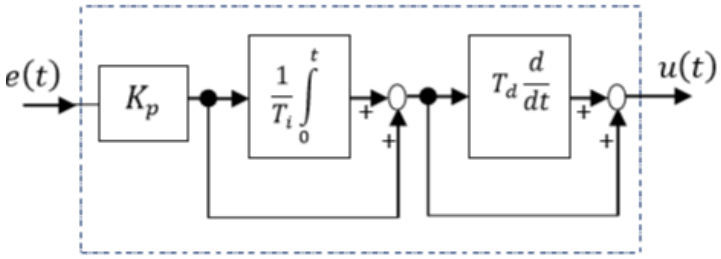


Figure III.9 : Schéma bloc de la commande PID série analogique.

- La loi de commande du PID série est :

$$u(t) = K_p\left(\frac{T_i + T_d}{T_i}e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \qquad \text{(III.8)}$$

- La fonction de transfert du correcteur PID série est donc :

$$C_{PID}(p) = \frac{U(p)}{E(p)} = K_p\left(1 + \frac{1}{T_i.p}\right)\left(1 + T_d.p\right) \qquad \text{(III.9)}$$

b. Structure PID parallèle

Le schéma bloc correspondant à l'architecture de PID parallèle est donné par **FigureIII.10**

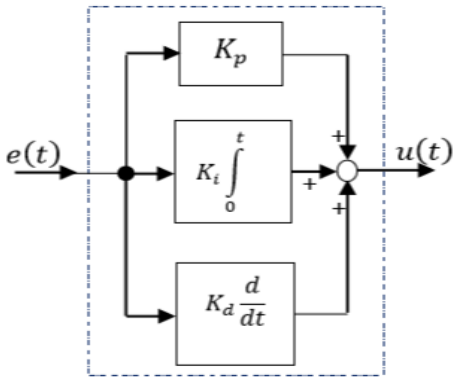


Figure III. 10: Schéma bloc de la commande PID parallèle analogique.

- La loi de commande du PID parallèle est :

$$u(t) = K_p.e(t) + K_i\int_0^t e(\tau)d\tau + K_d\frac{de(t)}{dt} \qquad \text{(III.10)}$$

- La fonction de transfert du correcteur PID parallèle est donc :

$$C_{PID}(p) = \frac{U(p)}{E(p)} = K_p + K_i.\frac{1}{p} + K_d.p \qquad \text{(III.11)}$$

c. Structure PID mixte

Le schéma bloc correspondant à l'architecture de PID mixte est donné par **Figure III.11.**

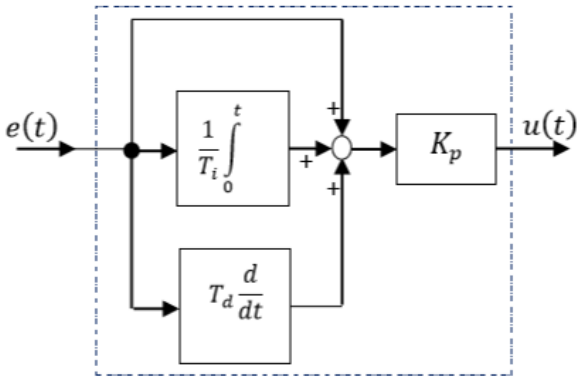


Figure III.11 : Schéma bloc de la commande PID mixte analogique.

- La loi de commande du PID mixte est :

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \qquad \text{(III.12)}$$

- La fonction de transfert du correcteur PID mixte est donc :

$$C_{PID}(p) = \frac{U(p)}{E(p)} = K_p\left(1 + \frac{1}{T_i.p} + T_d p\right) \qquad \text{(III.13)}$$

Remarque

Le terme dérivé incorporé dans le contrôleur PID est $T_d p$. En fait, cette fonction de transfert est irréalisable physiquement car elle est non causale. Dans la pratique, la dérivée est réalisée sous forme filtrée: $\frac{T_d p}{1+\frac{T_d p}{N}}$. Par conséquent, la fonction de transfert du contrôleur PID mixte devient sous la forme filtrée suivante :

$$C_{PID}(p) = K_p\left(1 + \frac{1}{T_i.p} + \frac{T_d p}{1+\frac{T_d p}{N}}\right) \qquad \text{(III.14)}$$

III.3.4.5 Avantages et inconvénients des actions d'un contrôleur PID analogique

a-Action proportionnelle P

- Permet de corriger les effets d'une perturbation ;
- Permet de diminuer l'erreur en régime permanant ;
- Permet d'augmenter la rapidité en régime transitoire ;
- Déstabilise le système quand on augmente le gain K_p.

b- Action intégrale I

- Annule l'erreur statique (erreur en régime permanant);
- Diminuer la rapidité et l'amortissement en régime transitoire ;
- Déstabilise le système quand on augmente le gain d'intégration K_i (T_i trop faible).

c-Action dérivée D

- Augmente la rapidité et l'amortissement en régime transitoire ;
- Stabilise plus rapidement le système (temps de réponse amélioré);
- Anticipe les erreurs futures ;
- N'annule pas l'erreur statique (aucun effet en régime permanant);
- Sensible aux parasites (perturbations extérieurs).

III.4 Asservissement numérique

III.4.1 Principe de l'asservissement numérique

Afin de mettre en œuvre les asservissements en milieu industriel, l'usage d'outils informatiques comme organes de contrôle des processus asservis est essentiel. C'est le cas par exemples des ordinateurs ou des microcontrôleurs qui peuvent, entre autre, assumer des fonctions de calculateurs numériques. Mais de tels instruments sont à base de composants électroniques (microprocesseurs, mémoires, ...) et fonctionnent avec des signaux binaires, porteurs d'informations numériques, on parle alors de signaux numériques. Se pose alors un problème fondamental, à savoir qu'un outil numérique ne peut s'accommoder de signaux analogiques, pourtant quasi exclusifs dans la majorité des systèmes physiques. En effet, le mode de traitement des informations imposé par un calculateur est de nature numérique et cadencé dans le temps de façon périodique grâce à une horloge. Le temps et l'amplitude du signal sont donc des grandeurs discrètes. Schématiquement, cela signifie que tout signal vivant dans l'ordinateur est une suite de nombres [28] [29]. Les problèmes qu'il s'agit de résoudre pour le contrôle des processus continus concernent les points suivants :

a- L'échantillonnage d'un signal continu

Cette opération consiste à relever les informations prises par un signal continu à intervalle de temps régulier, appelé période d'échantillonnage. On parle alors de signal échantillonné. Cela signifie que le calculateur ne tiendra compte que des échantillons, c'est-à-dire des valeurs prises par le signal aux instants d'échantillonnage.

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

Il s'agit de convertir la valeur prise par un signal analogique à l'instant d'échantillonnage en une valeur numérique afin qu'elle soit traitée par le calculateur. Un tel signal peut, par exemple, provenir d'un capteur. On parlera alors de signal de mesure.

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

Cette opération consiste à transformer le signal numérique issu du calculateur à l'instant d'échantillonnage (on parlera de signal numérique de commande), en signal analogique de commande existant sur toute la période d'échantillonnage, l'objectif étant de commander le système physique

d- La synthèse d'un algorithme de calcul

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

Remarque

- L'objet permettant de réaliser les opérations **(a)-(b)** s'appelle un Convertisseur-Analogique-Numérique (**CAN**).
- L'objet permettant de réaliser l'opération **(c)** s'appelle un Convertisseur Numérique-Analogique (**CNA**).

III.4.2 Structure d'un asservissement numérique

L'asservissement numérique se fait typiquement par le biais d'une structure schématisée par la **Figure III.12** et qui est composée à partir des objets fondamentaux suivants :

1. Un comparateur : celui-ci fournit un signal d'écart $\varepsilon(t)$ qui réalise la différence entre le signal analogique de référence e(t) et le signal analogique de mesure m(t).

2. Un CAN : celui-ci fonctionne à la période d'échantillonnage T > 0. Il fournit à sa sortie le signal numérique d'écart noté ε_n .

3. Un algorithme de commande : celui-ci manipule des suites de nombres et a pour fonction d'élaborer la loi de commande. Il délivre donc le signal numérique de commande u_n .

4. Un CNA : celui-ci fonctionne à la période d'échantillonnage T > 0. Il transforme le signal numérique de commande issu du calculateur en le signal analogique de commande correspondant.

5. **Des transmittances** *A(p)* et *C(p)* représentant respectivement la dynamique du système et celle du capteur.

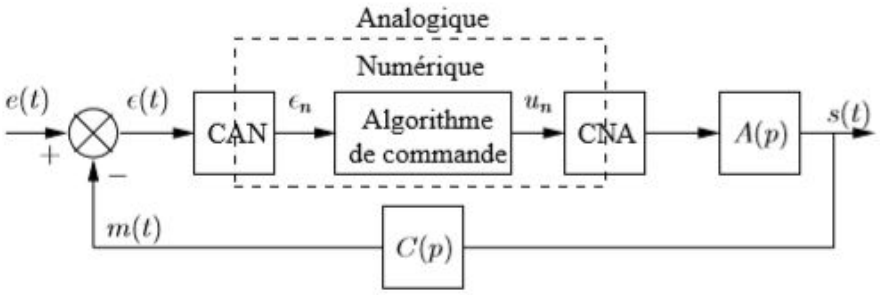


Figure III.12 : Structure typique de la réalisation d'un asservissement numérique

En pratique, l'opération de comparaison se fait également numériquement. Ainsi, une autre structure typique d'asservissement peut être schématisée par la **Figure III.13** où nous pouvons remarquer la présence d'un CAN supplémentaire.

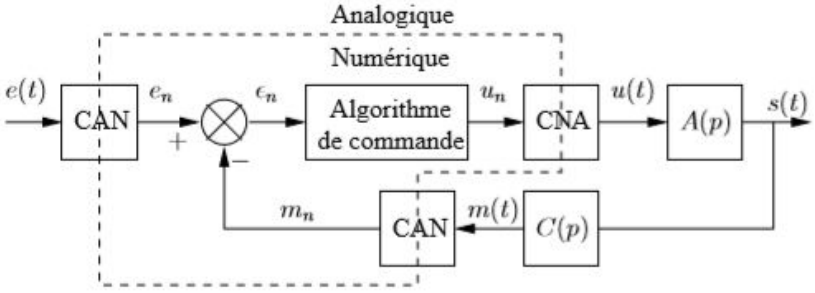


Figure III.13 : Autre structure typique réalisant un asservissement numérique

Dans toute structure d'asservissement est inséré un calculateur numérique réalisant, entre autre, les tâches de l'algorithme de commande. Un tel calculateur peut être à base de microprocesseurs et faire partie d'un microcontrôleur, d'une carte électronique dite d'acquisition et de traitement temps réel, type DSP, réalisant également les opérations de conversion.

III.4.3 Objectifs de l'asservissement numérique

Les performances naturelles d'un système (stabilité, précision, rapidité, etc...) peuvent ne pas correspondre à un degré d'exigence spécifié dans un cahier des charges. L'objectif de tout asservissement est de modifier ces performances afin qu'elles respectent au mieux ce cahier des charges.

De façon qualitative :

- **Pour rendre le système stable**, il faut rassembler tous les pôles du système dans le cercle unité et pour garantir une meilleure stabilité, il faut éloigner le plus possible les pôles de ce cercle en les rapprochant de l'origine.

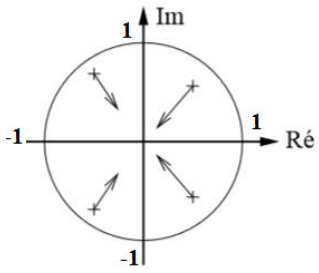


Figure III.14 : Plan des pôles numériques.

- Pour rendre le système plus précis, il faut augmenter le gain statique, voire ajouter des intégrateurs pour augmenter la classe du système.

- Pour améliorer la rapidité du système, il faut augmenter la bande passante de celui-ci, en augmentant la valeur de la fréquence de coupure.

Il existe différentes structures d'asservissement numérique. On s'intéresse au correcteur PID numérique qu'on va utiliser dans notre application.

III.4.4 Commande PID numérique

Comme en correction analogique, l'un des principaux correcteurs utilisés est un correcteur à action Proportionnelle, Intégrale et Dérivée, noté PID.

III.4.4.1 Transposition des contrôleurs PID analogiques

Afin de pouvoir importer un régulateur PID analogique dans un calculateur numérique, il faut le transposé (numérisé) en utilisant les différentes techniques de discrétisation.

Les régulateurs discrets ou numériques élaborent une grandeur de commande discrète $u(k)$ en fonction de l'écart de réglage discret $e(k)$ du système à commander. Selon la complexité du régulateur, la grandeur de commande à l'instant k est formée en fonction de la valeur de l'écart à cet instant, mais aussi instants précédents $(k-1)$, $(k-2)$,...etc.

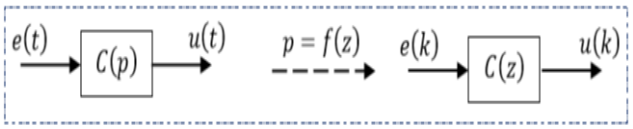


Figure III.15 : Approximation de la variable complexe 'p'

III.4.4.2 Différentes approximations de la dérivée et de l'intégrale continues

Les approximations les plus utilisées pour approcher la dérivée et l'intégrale continues sont résumées dans le tableau suivant [30]:

	Dérivation	***Intégration***
Méthode d'Euler arrière (Discrétisation arrière)	$p \to \frac{z-1}{Tz}$	$\frac{1}{p} \to \frac{Tz}{z-1}$
Méthode d'Euler avant (Discrétisation avant)	$p \to \frac{z-1}{T}$	$\frac{1}{p} \to \frac{T}{z-1}$
Méthode de Tustin (Transformation bilinéaire)	$p \to \frac{2}{T}\frac{z-1}{z+1}$	$\frac{1}{p} \to \frac{T}{2}\frac{z+1}{z-1}$

Tableau III.1 : Différentes approximations de la dérivée et de l'intégrale continues

III.4.4.3 Réalisation des contrôleurs PID numériques

Dans cette section, nous allons utiliser l'approximation d'Euler arrière pour discrétiser les différents contrôleurs numériques cités dans la section précédente [31].

a- Commande P numérique

Le régulateur P analogique a pour fonction de transfert $C_p(p)=\frac{U(p)}{E(p)}=K_p$. En utilisant la transformée en 'z', nous obtenons la fonction de transfert du contrôleur P numérique équivalent :

$$C_p(z)=\frac{U(z)}{E(z)}=K_p$$

(III.15)

Ce régulateur est décrit par l'équation récurrente suivante:

$$u(k)=K_p e(k) \qquad \text{(III.16)}$$

Son schéma fonctionnel est donné comme suit :

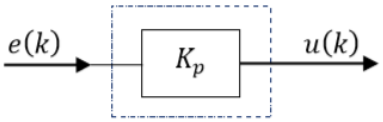


Figure III.16 : Schéma fonctionnel du contrôleur P numérique

b- Commande PI numérique

Soit un contrôleur PI analogique dont la fonction de transfert est de la forme $C_{PI}(p)=\frac{U(p)}{E(p)}=K_p\left(1+\frac{1}{T_i p}\right)$. En utilisant l'approximation d'Euler arrière, nous trouvons :

$$C_{PI}(z)=\frac{U(z)}{E(z)}=K_p\left(1+\frac{T}{T_i}.\frac{z}{z-1}\right)=\frac{r_0+r_1 z^{-1}}{1-z^{-1}} \qquad \text{(III.17)}$$

L'équation récurrente suivante permet de décrire ce régulateur :

$$u(k)=u(k-1)+r_0 e(k)+r_1 e(k-1)$$

(III.18)

Avec: $r_0=K_p\left(1+\frac{T}{T_i}\right)$ et $r_1=-K_p$

Son schéma fonctionnel peut être donné comme suit :

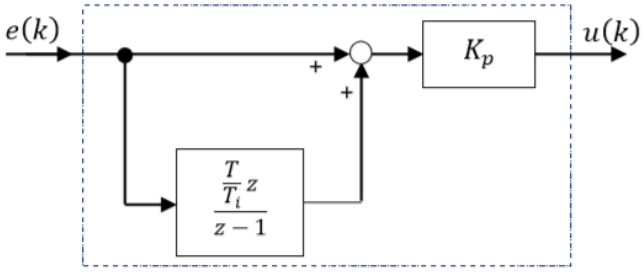


Figure III.17: Schéma fonctionnel du contrôleur PI numérique

c- Commande PD numérique

Un contrôleur PD analogique est décrit par la fonction de transfert: $C_{PD}(p) = K_p(1+T_d p)$.

Sa version numérique est obtenue en utilisant l'approximation d'Euler arrière $p = \frac{z-1}{T.z}$:

$$C_{PD}(z) = \frac{U(z)}{E(z)} = K_p(1+\frac{T_d}{T}\frac{z-1}{z}) \qquad \text{(III.19)}$$

ou bien :

$$C_{PD}(z) = r_0 + r_1 z^{-1}$$

(III.20)

Ce contrôleur est représenté par l'équation récurrente suivante :

$$u(k) = r_0 e(k) + r_1 e(k-1) \qquad \text{(III.21)}$$

Avec: $r_0 = K_p\left(1+\frac{T_d}{T}\right)$ et $r_1 = -K_p\frac{T_d}{T}$

Son schéma fonctionnel est montré par la figure suivante :

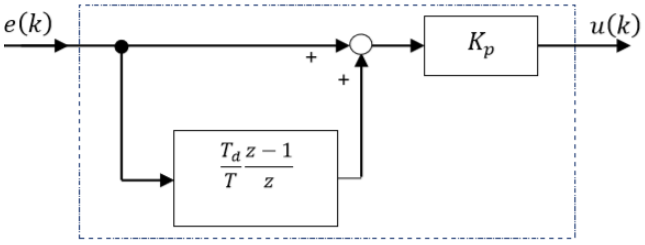


Figure III.18: Schéma fonctionnel du contrôleur PD numérique

d- Contrôleurs PID numériques

Dans cette section, les trois structures des contrôleurs PID analogiques étudiées dans la section précédente, sont discrétisées par la méthode d'Euler arrière. Par conséquent, leurs fonctions de transfert numériques équivalentes et leurs équations récurrentes sont calculées, et leurs schémas fonctionnels correspondants sont présentés.

- **Structure PID numérique série**

  La structure série du contrôleur PID numérique est définie par la fonction de transfert :

$$C_{PID}(z) = \frac{U(z)}{E(z)} = K_p\left(1+\frac{T}{T_i}\frac{z}{z-1}\right)\left(1+\frac{T_d}{T}\frac{z-1}{z}\right) \qquad \text{(III.22)}$$

Cette fonction de transfert peut être écrit :

$$C_{PID}(z) = \frac{r_0 + r_1 z^{-1} + r_2 z^{-2}}{1 - z^{-1}}$$

(III.23)

L'équation récurrente correspondante est donnée par

$$u(k) = u(k-1) + r_0 e(k) + r_1 e(k-1) + r_2 e(k-2) \quad \text{(III.24)}$$

Par définition : $r_0 = K'_p \left(T^2 + TT'_i + T'_d T'_i\right) / TT'_i$, $r_1 = -K'_p \left(T + 2T'_d\right) / T$ et $r_2 = K'_p T'_p / T_i$

Avec : $K'_p = K_p \frac{T_i + T_d}{T_i}, T'_i = T_i + T_d$ et $T'_d = \frac{T_i T_d}{T_i + T_d}$

Cette structure, apparaissant sous forme de schéma fonctionnel dans la figure suivante :

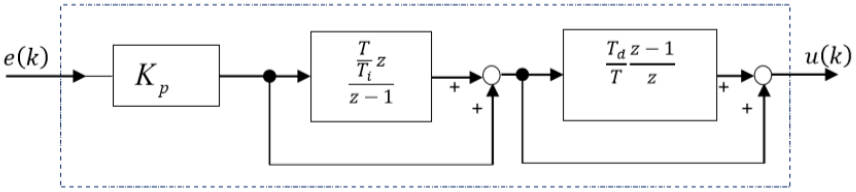


Figure III.19 : Schéma fonctionnel du contrôleur PID numérique série

- **Structure PID numérique parallèle**

  Une structure parallèle du contrôleur PID numérique s'obtient en posant $K_i = K_p / T_i$ et $K_d = K_p T_d$. Comme la structure série, ce contrôleur peut être décrit par :

- une fonction de transfert en 'z' :

$$C_{PID}(z) = \frac{U(z)}{E(z)} = K_p + \frac{K_i T}{z-1} z + \frac{K_d}{T} \frac{z-1}{z} \quad \text{(III.25)}$$

- une fonction de transfert en '$z-1$' :

$$C_{PID}(z) = \frac{U(z)}{E(z)} = \frac{r_0 + r_1 z^{-1} + r_2 z^{-2}}{1 - z^{-1}} \quad \text{(III.26)}$$

- une équation récurrente

$$u(k) = u(k-1) + r_0 e(k) + r_1 e(k-1) + r_2 e(k-2) \quad \text{(III.27)}$$

On définit :

$$r_0 = \frac{TK_p + K_d}{T}, r_1 = \frac{T^2 K_i - TK_p + 2K_d}{T} \quad \text{et} \quad r_2 = \frac{K_d}{T}$$

- un schéma fonctionnel:

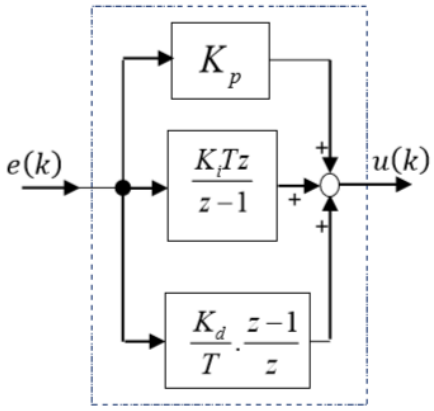


Figure III.20 : Schéma fonctionnel du contrôleur PID numérique parallèle

- **Structure PID numérique mixte**

  Le régulateur PID analogique mixte, quand la dérivation n'est pas filtrée, possède la fonction de transfert : $C_{PID}(p)=\frac{U(p)}{E(p)}=K_p\left(1+\frac{1}{T_i.p}+T_d p\right)$

Elle est non causale. Par contre, sa version numérique est causale :

$$C_{PID}(z)=\frac{U(z)}{E(z)}=K_p\left(1+\frac{T}{T_i.z}\frac{z}{z-1}+\frac{T_d}{T}\frac{z-1}{z}\right) \qquad \text{(III.28)}$$

Cette dernière peut être écrite sous la forme suivante :

$$C_{PID}(z)=\frac{U(z)}{E(z)}=\frac{r_0+r_1z^{-1}+r_2z^{-2}}{1-z^{-1}} \qquad \text{(III.29)}$$

L'équation récurrente de cette structure est donnée par:

$$u(k)=u(k-1)+r_0e(k)+r_1e(k-1)+r_2e(k-2) \qquad \text{(III.30)}$$

On définit : $r_0=K_p\left(1+\frac{T}{T_i}+\frac{T_d}{T}\right), r_1=-K_p\left(1+2\frac{T_d}{T}\right)$ et $r_2=K_d\frac{T_d}{T}$

Cette variante est représentée par le schéma fonctionnel suivant :

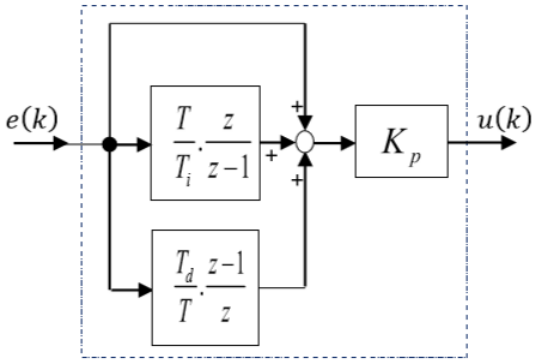


Figure III.21 : Schéma fonctionnel du contrôleur PID numérique mixte

Cette variante est dite forme non filtrée. En fait, il est préférable, pour réaliser un contrôleur PID numérique, de prendre la forme filtrée suivante :

$$C_{PID}(z)=\frac{U(z)}{E(z)}=K_p\left(1+\frac{T}{T_i}\frac{z}{z-1}+\frac{N(z-1)}{\left(1+N\frac{T_d}{T}\right)z-1}\right) \quad \text{(III.31)}$$

On peut écrire aussi la fonction de transfert de l'équation **(III.31)** sous la forme réduite suivante:

$$C_{PID}(z)=\frac{U(z)}{E(z)}=\frac{r_0+r_1z^{-1}+r_2z^{-2}}{(1-z^{-1})(1+s_1z^{-1})}$$

(III.32)

Dans ce cas l'équation récurrente est donnée comme suit

$$u(k)=(1-s_1)u(k-1)+s_1u(k-2)+r_0e(k)+r_1e(k-1)+r_2e(k-2)$$

(III.33)

avec:

$$s_1=-\frac{T_d}{T_d+NT},\ r_0=K_p\left(1+\frac{T}{T_i}-Ns_1\right),\ r_1=K_p\left(s_1\left(1+\frac{T}{T_i}+2N\right)-1\right) \text{ et } r_2=-K_d.s_1(1+N)$$

N : est un entier positif non nul.

D'autre part, le schéma fonctionnel de la **Figure III.21** devient :

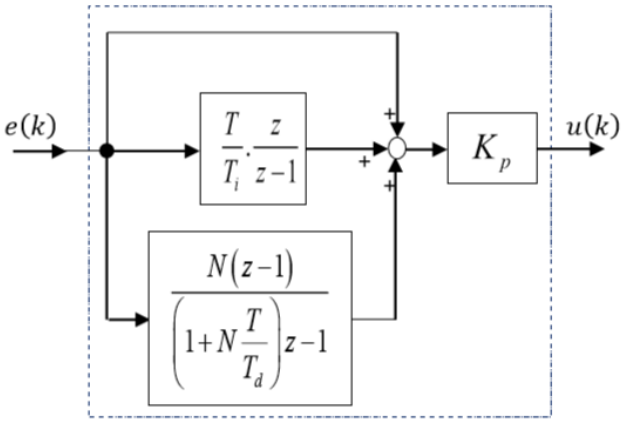


Figure III.22 : Schéma fonctionnel du contrôleur PID numérique filtré mixte

III.4.4.4 Choix du correcteur

La structure du correcteur PID fait apparaître trois actions : l'action Proportionnelle, l'action Intégrale et l'action Dérivée [32].

a- Action qualitative

- Action P : augmente la bande passante, donc la rapidité ; améliore la précision et dégrade la stabilité.

- Action I : ralentit le système ; améliore la précision en augmentant la classe du système et peut dégrader la stabilité. De plus, peu robuste aux perturbations basses-fréquences sur le signal de consigne.

- Action D : augmente la bande passante du système donc sa rapidité, permet d'améliorer la stabilité mais amplifie les bruits de mesure hautes fréquences.

L'action intégrale I présente d'autres inconvénients d'ordre pratique. Par exemple, l'intégrateur pur n'est pas réalisable technologiquement car cela signifierait un gain infini pour les basses fréquences. De manière similaire, l'action dérivée D présente un défaut analogue mais dans les hautes fréquences.

Il n'est pas toujours utile de faire une correction avec un correcteur qui combine toutes ces actions. Pour des améliorations spécifiques des performances, le correcteur peut être réduit :

b- Choix du correcteur

- Correcteur P : améliore la précision, sans changer la classe du système.

- Correcteur PI : augmente la classe du système et donc améliore la précision.

- Correcteur PID : augmente la classe du système, donc améliore sa précision, mais aussi la rapidité et la stabilité.

III.5 Conclusion

Dans ce présent chapitre nous avons expliqué les concepts de base de l'asservissement ainsi que les performances acquises. Nous avons ensuite mis en lumière la commande PID analogique afin de mieux maitriser le PID numérique. Ce dernier est un mécanisme de rétroaction de boucle de contrôle générique largement utilisé dans les systèmes de contrôle industriels et qui a prouvé son efficacité.

Le contrôleur PID numérique va assurer le contrôle de notre système thermique en temps réel. La réalisation pratique de cette régulation fera l'objet du prochain chapitre.

Chapitre IV

Conception et réalisation pratique

IV.1 Introduction

Ce chapitre est consacré en premier lieu à la conception et la réalisation de la maquette d'un système de régulation de température en se basant sur l'Implémentation de la carte Arduino UNO avec Matlab/Simulink.

Nous allons passer ensuite à l'étude expérimentale en appliquant au système thermique différents régulateurs afin de valider notre réalisation.

IV.2 Présentation de la maquette

La maquette est constituée d'un capteur de température LM35 et une Lampe halogène 12V/35W qui joue le rôle d'un élément chauffant, c'est le procédé thermique. Le capteur et la lampe sont installés dans une boite en bois avec un couvet en verre. Cette boite représente le système thermique qu'on veut contrôler sa température. La **Figure IV.1** schématise la connexion entre la carte Arduino UNO et le système thermique en passant par un circuit d'adaptation (résistance, transistor et transformateur) ainsi que la communication entre Arduino et le PC grâce au câble USB.

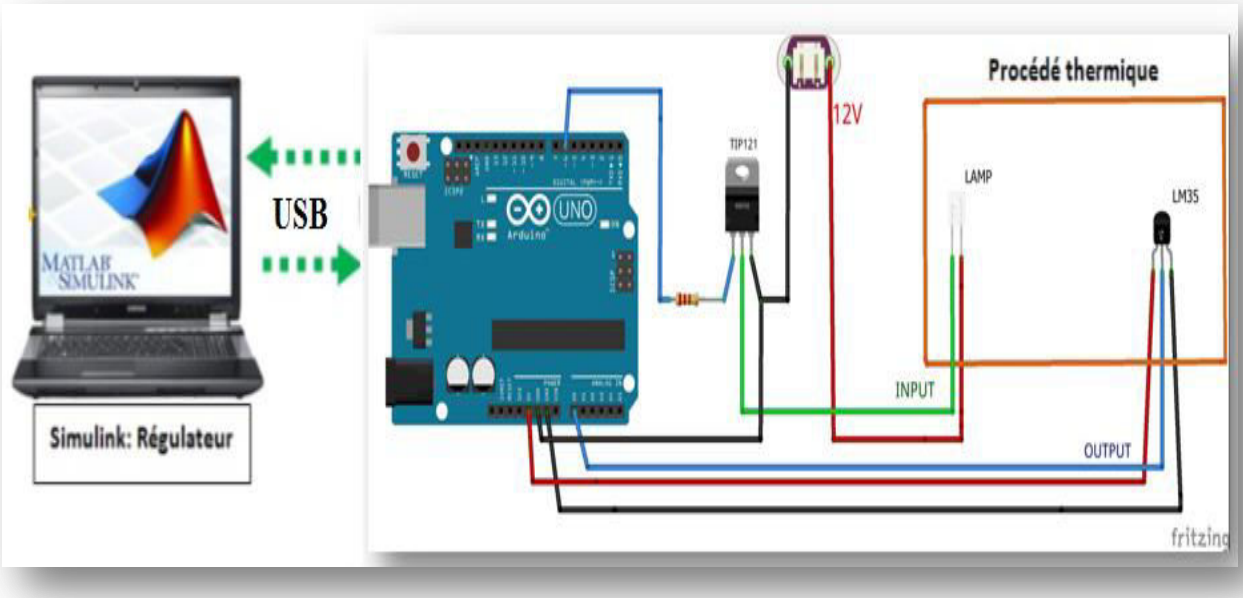


Figure IV.1: Câblage du procédé avec la carte Arduino et PC

Nous présentons aussi dans la Figure IV.2 une vue réelle de la maquette de la commande d'un système thermique que nous avons réalisé.

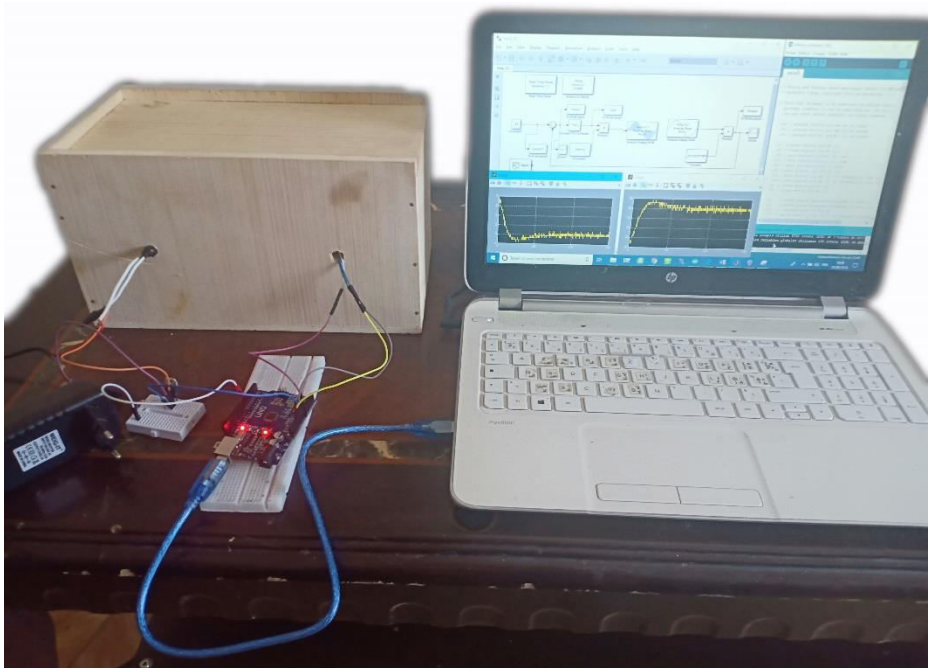


Figure IV.2: Une vue réelle de la maquette de commande du système thermique.

IV.2.1 Matériel utilisés

Pour la réalisation pratique de la commande d'un système termique dont la maquette est donnée par **Figure IV.1** et **Figure IV.2**, nous avons utilisé les composants electronique suivants:

IV.2.1.1 Carte Arduino UNO

Dans chapitre II, nous avons donné une déscription détaillée sur cette plateforme d'Arduino UNO qui est utilisée comme une interface perméttant l'écahange des données entre PC et le système thermique. .

IV.2.1.2 Un microportable PC

Nous avons utililisé un PC muni d'un logiciel Matlab/Simulink afin de commander le système thermique et visusaliser les résultats obtenus sous Simulink.

IV.2.1.3 Capteur de température LM35

Nous avons utilisé dans notre réalisation un capteur de température LM35 donnée par **Figure IV.4** dont sa définition et ses carctéristiques ont étet expliquées dans chapitre I.
D'après la fiche technique de ce capteur un volt correspond à 100 degrés Celsius. Cela représente la sensibilité du capteur.

La conversion de la tension de sortie du capteur Vout à une température est calculée par l'équation suivante:

$$Température(^0C) = V_{out} \times \left(100\ (^0C/V)\right) \quad \text{(IV.1)}$$

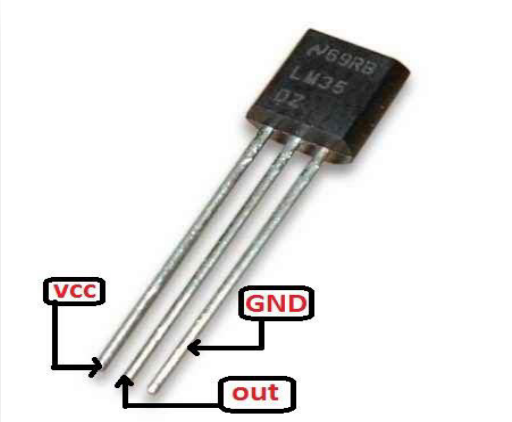


Figure IV.4: Capteur de température LM35.

a- Branchement de LM 35 avec la carte Arduino UNO

Pour exploiter le capteur LM35, il sufit (voir Figure IV.5 et Figure IV.6):

1. D'alimenter la patte VCC du composant en la branche sur la broche 5V de l'Arduino;
2. D'alimenter la patte GND du composant en la branche sur la broche GND du Arduino;
3. De brancher la patte centrale OUT (celle qui envoie les informations) à une entrée analogique d'Arduino (A0,..., A5).

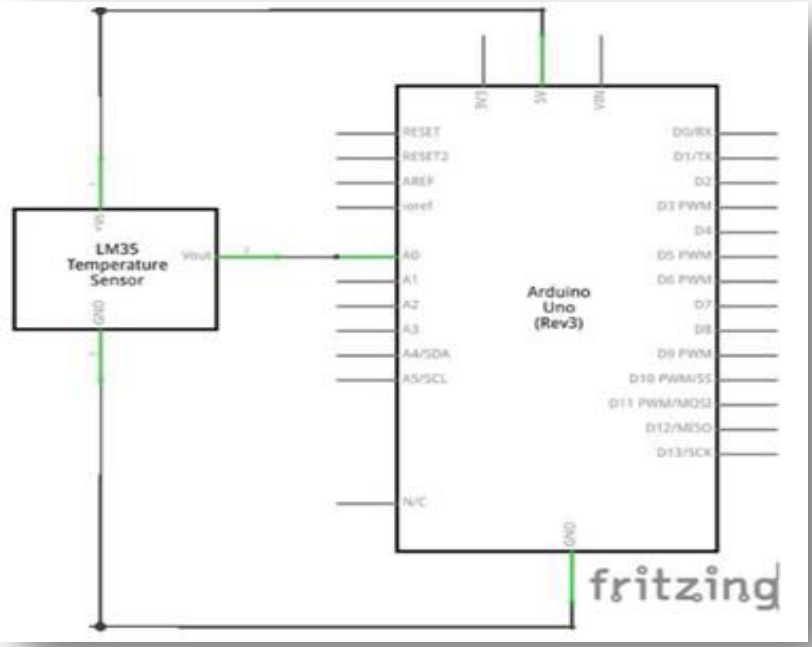


Figure IV.5: Schéma électronique du branchement du capteur LM35 avec Arduino.

La tension sur la sortie analogique (la broche « out » du capteur) est proportionnelle à la température mesurée. Lorsque on utilise la fonction analogRead() dans le programme d'Arduino afin de lire la valeur sur cette sortie analogique, cette fonction renvoie non pas une tension comprise entre 0 et 5V mais une valeur comprise entre 0 et 1023 une température comprise entre 0° jusqu'à 100°C. La valeur de la température sortant du capteur est calculée par la formule suivante :

$$Température(^0C) = V_{out} \times (5/1023) \times (100\ (^0C/V)) \quad \text{(IV.2)}$$

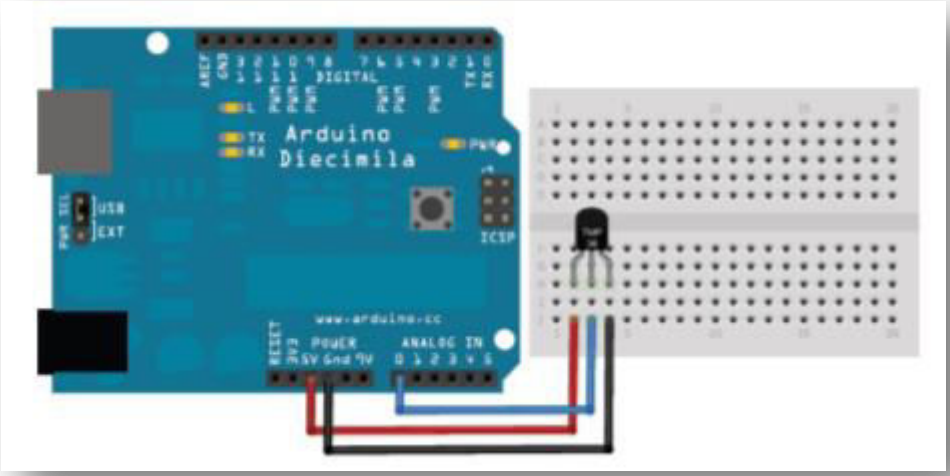


Figure IV.6: Câblage du capteur de température LM35 avec Arduino.

IV.2.1.4 Transistor TIP 122

Un transistor est un dispositif semi-conducteur à trois électrodes actives, qui permet de contrôler un courant (ou une tension) sur une des électrodes de sorties (le collecteur pour le transistor bipolaire et le drain sur un transistor à effet de champ) grâce à une électrode d'entrée (la base sur un transistor bipolaire et la grille pour un transistor à effet de champ).

Le transistor est utilisé comme interrupteur dans les circuits logiques, comme amplificateur de Signal, pour stabiliser une tension, moduler un signal ainsi que de nombreuses autres utilisations.

Dans notre projet, on a utilisé le transistor TIP121, donnée par Figure IV.7. C'est un transistor NPN utilisé pour les commutations de puissance linéaire moyenne et permet d'amplifier le courant jusqu'à 5A avec un gain d'amplification au minimum $\beta = 1000$.

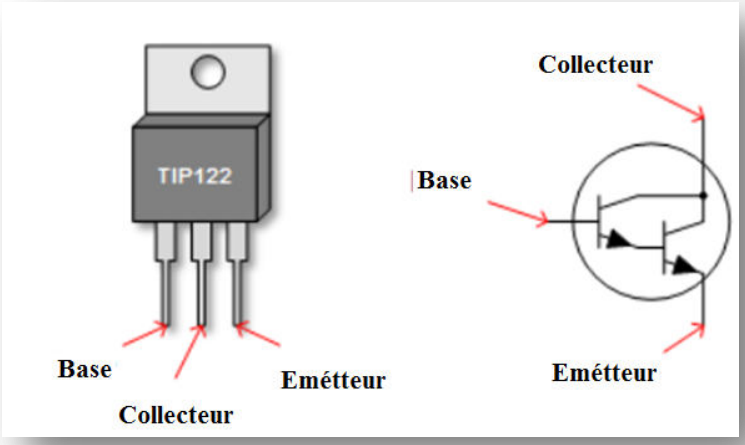


Figure IV.7: Transistor TIP 122.

IV.2.1.5 Lampe halogène

La lampe à luminosité halogène produit de la lumière qui ressemble à celle d'une lampe classique, en portant à sa luminescence un filment de tungestène et des gaz halogènes (iode et Brome) à base pression qui ont été introduits dans une ampoule en verre de quartz suppotant les hautes températures et permettant la régénération du filament, au moins partiellement, ce qui augmente la durée de vie de l'ampoule.

Dans notre projet, nous avons utilisé une lampe d'halogène 12V-35W illustrée par FigureIV.8 qui joue le rôle d'un élément chauffant.

Figure IV.8: Lampe halogène

La **Figure IV.9** montre le câblage de la lampe halogène avec Arduino et d'autres composants.

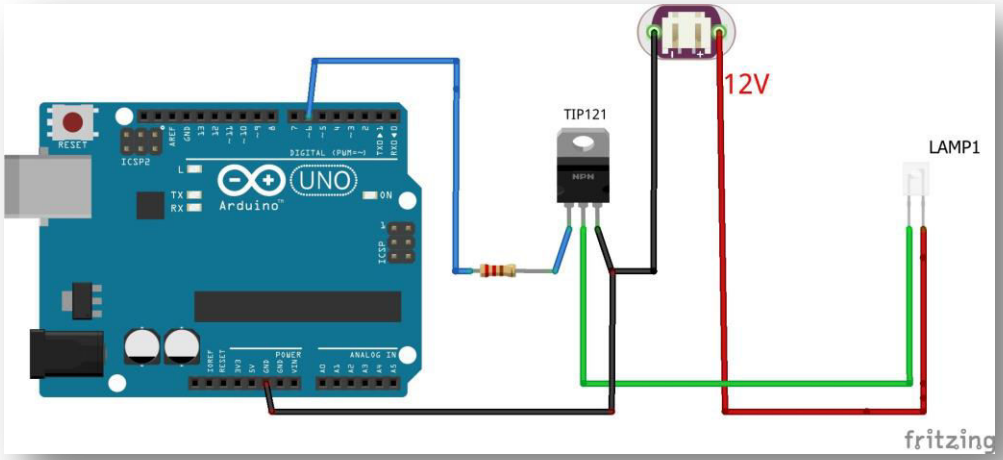


Figure IV.9: Câblage de la carte Arduino UNO avec la lampe d'halogène.

L'utilisation de la commande PWM à partir de la carte Arduino permet de faire varier la tension appliquée aux bornes de la lampe. Cela permet de contrôler l'intensité lumineuse de la lampe.

IV.2.1.6 Résistance

C'est un composant électronique ou électrique dont la principale caractéristique est d'opposer une plus ou moins grande résistance (mesurée en ohms) à la circulation du courant électrique. Dans notre application, nous avons utilisé une résistance de 1khom dont le branchement est donné par Figure IV.9.

IV.2.1.7 Transformateur

Un transformateur électrique est un outil qui sert à changer les valeurs d'intensité et de tension d'un courant électrique selon notre besoin.Cette machine permet en effet de modifier ces valeurs tout en gardant la même fréquence et la même forme. Dans notre projet, nous avons utilisé un transformateur de tension 220-12V, schématisé par Figure IV.10, dont le rôle est d'alimenter la lampe par 12 V. Le câblage de ce transformateur est donné par Figure IV.9.

Figure IV.10: Transformateur 220-12 V.

IV.2.2 Les étapes de développement

Les étapes à suivre pour pouvoir commander le système thermique à l'aide de l'interface Arduino/ Maltab sont les suivantes :

IV.2.2.1 L'interfaçage Arduino Matlab

Nous développons des blocs Simulink qui assurent les fonctions suivantes : acquérir, traiter et afficher les données depuis la carte Arduino Uno. En effet, à partir de la version 2012a de Matlab, la carte Arduino UNO ainsi qu'Arduino MEGA peuvent être installées automatiquement on peut accéder à plusieurs ressources à ce sujet [21] et [33].

a- Programmation d'Arduino UNO comme une carte d'acquisition

Au préalable, la carte Arduino est programmée pour être utilisée comme une carte d'interface Entrées/Sorties (Voir Chapitre II) :

- La référence est appliquée à la sortie PWM (Pin 5) de l'Arduino.

La température issue capteur est appliquée à l'entrée A0 de l'Arduino. b- Utilisation du package ArduinoIO (voir chapitre II)

IV.2.2.2 Acquisition et envoie des données

a- Acquisition des données à partir du capteur LM35

1- Branchement avec la carte Arduino UNO se fait selon la Figure IV.6

2- Exploitation du package Arduino IO Library

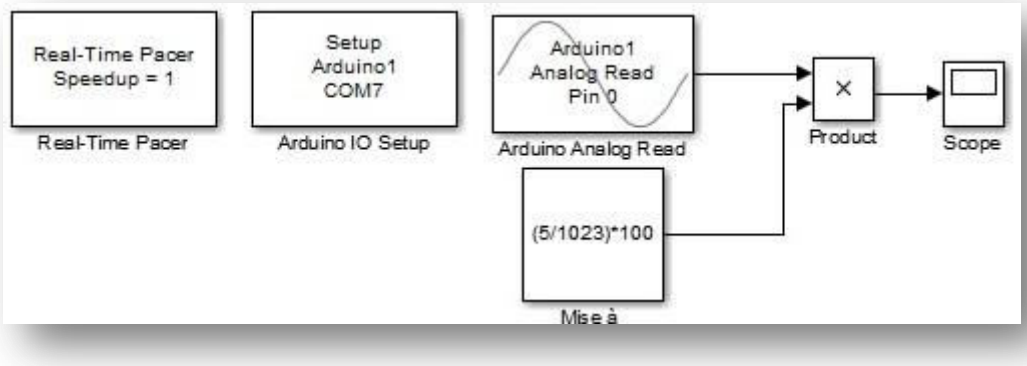


Figure IV.11 : Acquisition de la température sous ArduinoIO Library

b- Exploitation d'Instrument Control Toolbox

1- Pré-programmation de la carte Arduino UNO

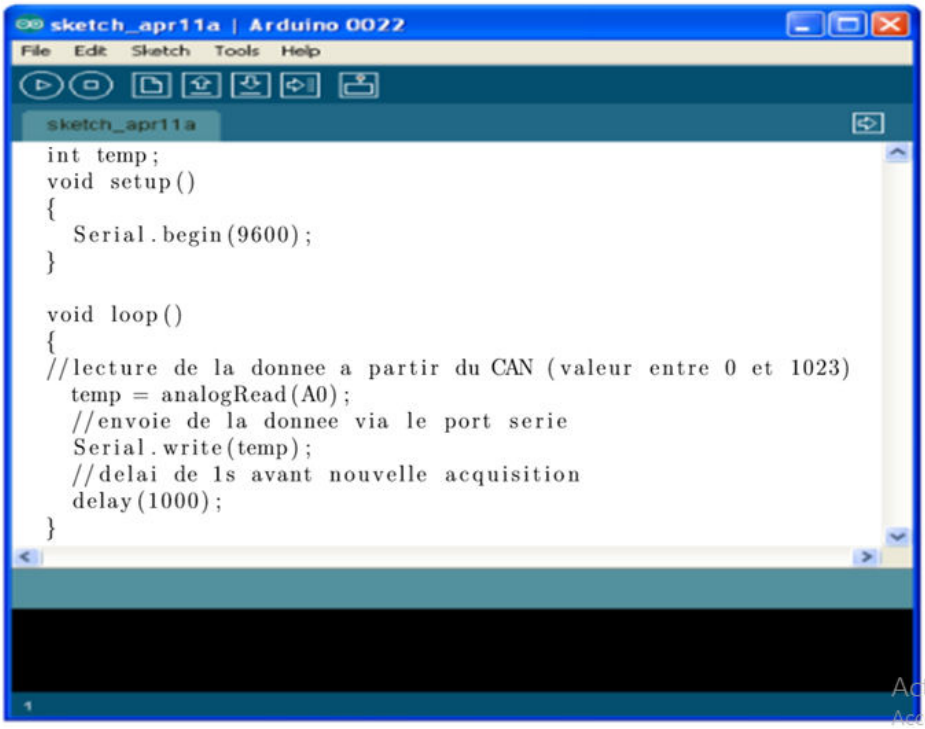


Figure IV.12: Pré-programme de la carte Arduino

2- Développement du modèle Simulink

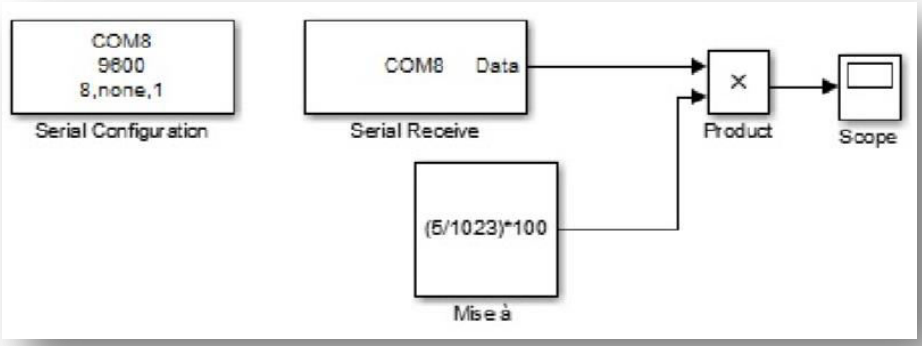


Figure IV.13: Acquisition de la température sous Instrument Control Toolbox

c- Exploitation du package ArduinoIO Libraray

1. Pré-chargement de adiosrv.pde sur la carte Arduino UNO
2. Développement du modèle Simulink

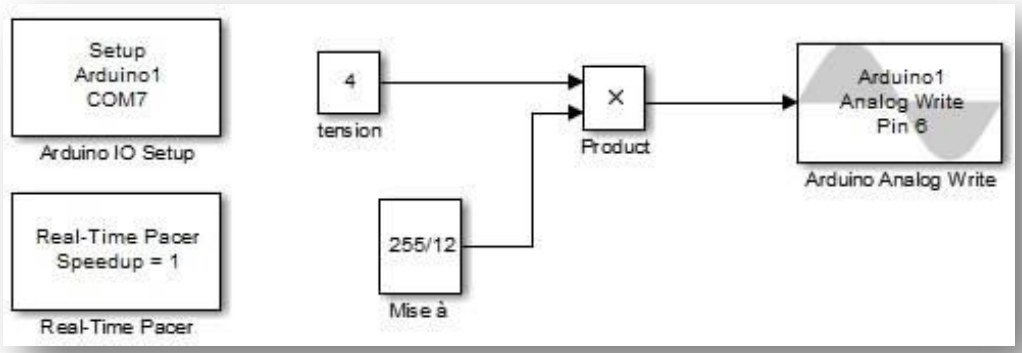


Figure IV.14 : Envoie de la commande PWM sous ArduinIO Library

IV.3 Modélisation du procédé thermique

La modélisation du procédé thermique qu'on a considéré dans notre application revient déterminer sa fonction de transfert échantillonnée en boucle ouvert notée G(z). L'entrée du système est la tension u(z) en volts et la sortie est la température T (z) de degré Celsius.

IV.3.1 Identification du système sous Matlab/Simulink

Cette étape est constituée de deux parties. La première est assurée 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 (voir Figure IV.15).

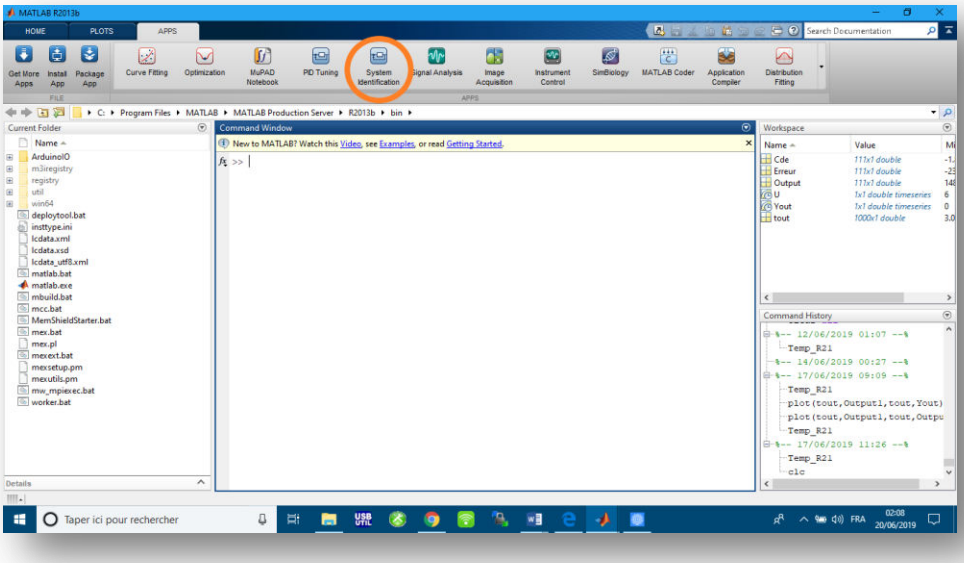


Figure IV.15 : L'utilisation de l'outil System Identification

IV.3.2 Acquisition de la réponse indicielle du système

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

Dans notre cas on va identifier notre système en étudiant la réponse de notre système à échelon de tension de 6V. Le modèle Simulink permettant de réaliser l'acquisition de la réponse du système à un échelon de tension est donnée par Figure IV.16.

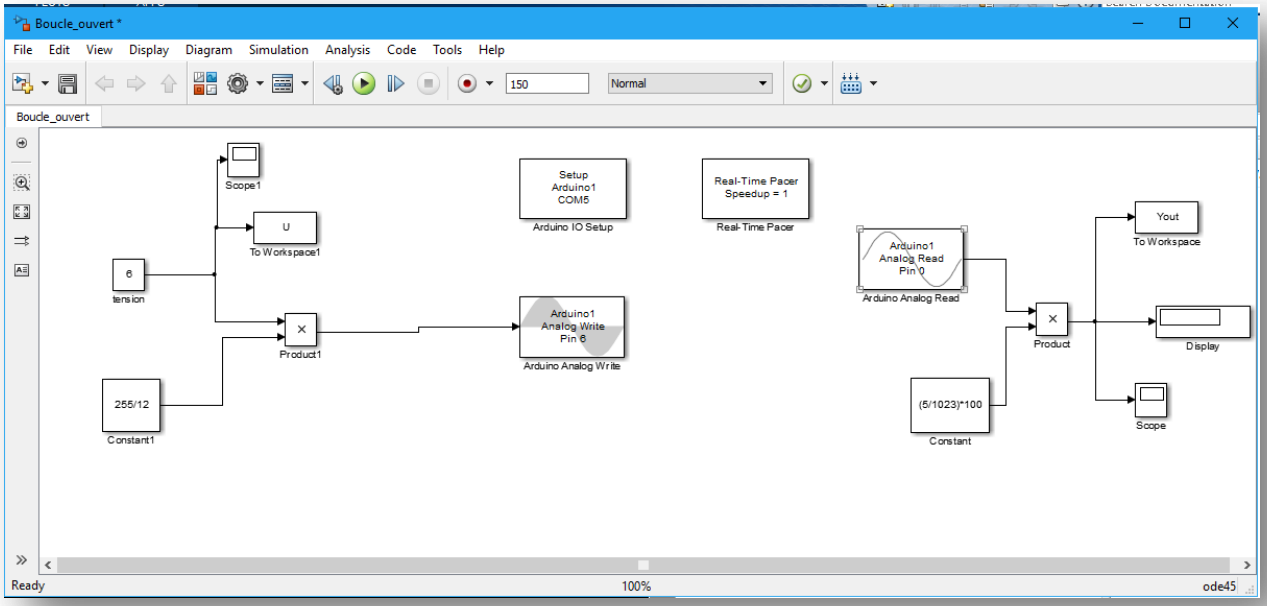


Figure IV.16: Modèle Simulink pour la détermination de la réponse indicielle en temps réel

IV.3.3 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) selon les étapes suivantes :

1- Ouvrir l'outil System identification Tool.

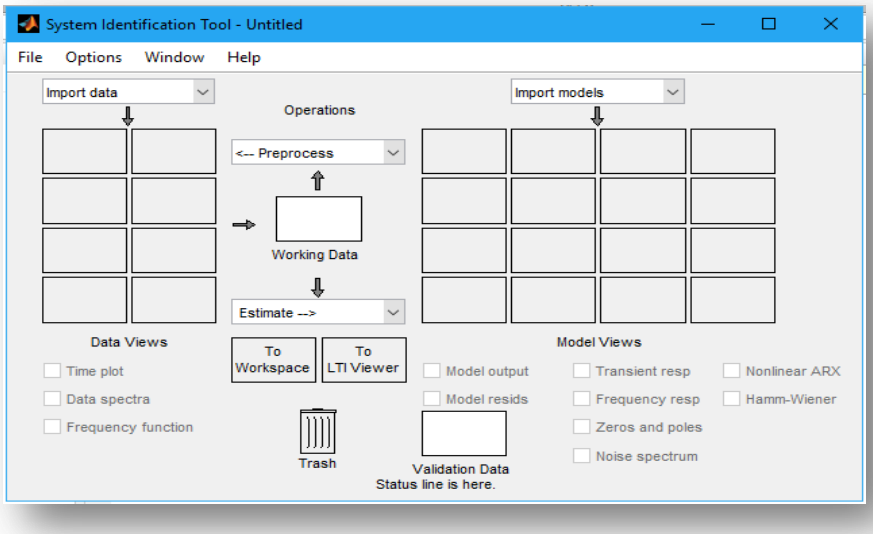


Figure IV. 17 : L'interface de l'outil *System identification.*

2- Cliquer sur import data et choisir *Time domain data*

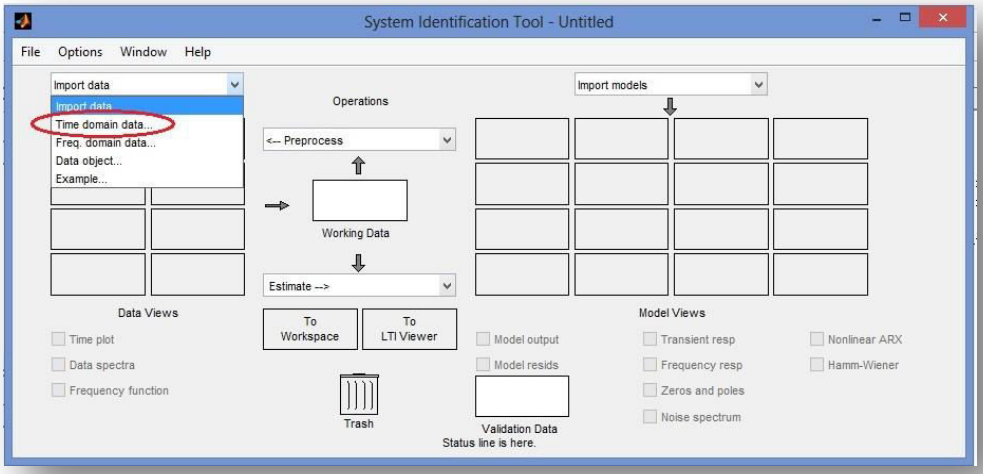


Figure IV.18: Choix des types des données "*Time domain Data*"

3- Entrer le nom de la variable *Input* et la variable *Output* ainsi que temps de *starting time* et *sample time* qu'on a utiliser lors de l'identification avec Simulink. Enfin cliquer sur Import.

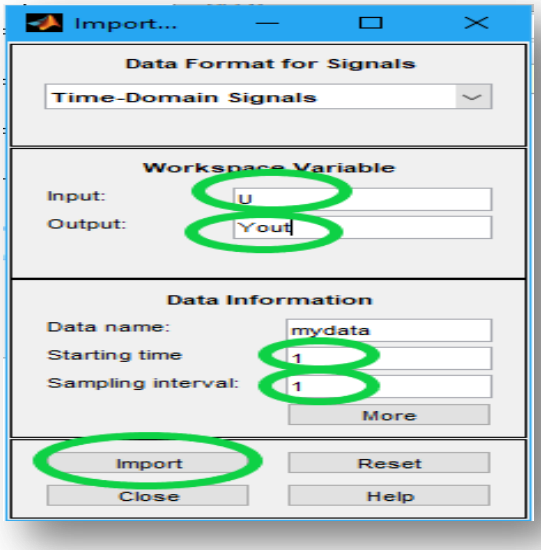


Figure IV.19 : Saisie des données relatives aux Input et Output du système.

4- Cliquer sur *Estimate e*t choisir Transfer *Function Models*

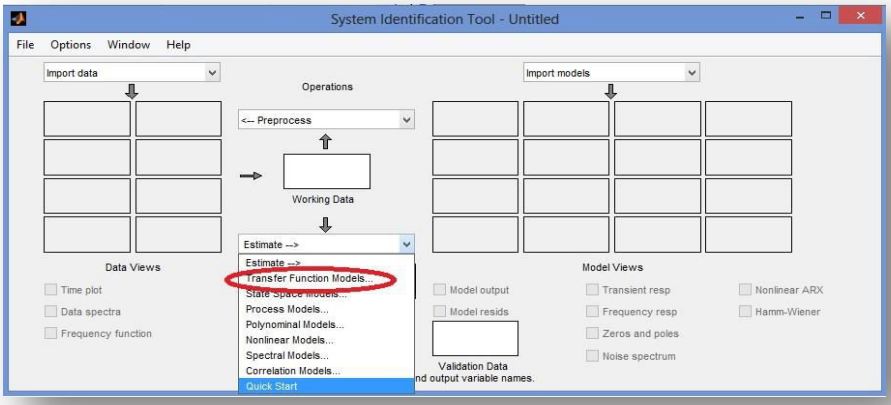


Figure IV.20: Choix de la description du système à estimer "*Tranfer Function*"

5- Entrer le nombre de pôles et de zéros et cliquer sur *Discrete-Time* ensuite cliquer sur *Estimate.*

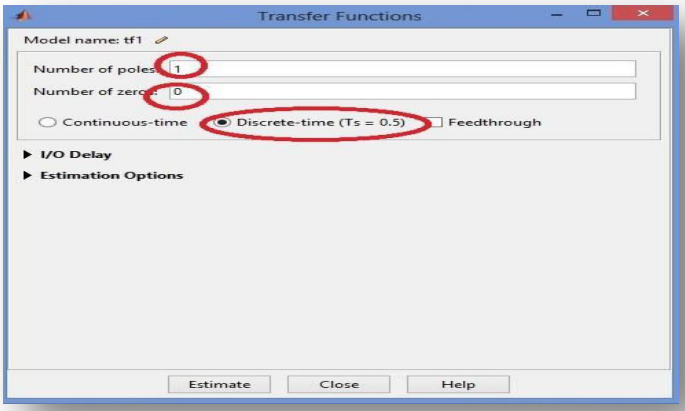


Figure IV.21: Choix du nombre des pôles et zéros de la fonction de transfert à estimer

6- Revenir à l'interface *System Identification Tool* et cliquer deux fois sur *tf1*.

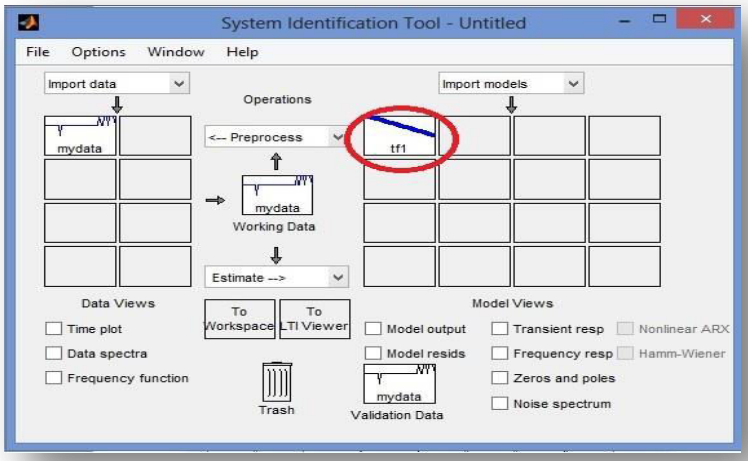


Figure IV.22: Visualisation du résultat de l'estimation

7- Une fenêtre apparait dans laquelle vous trouvez G(z).

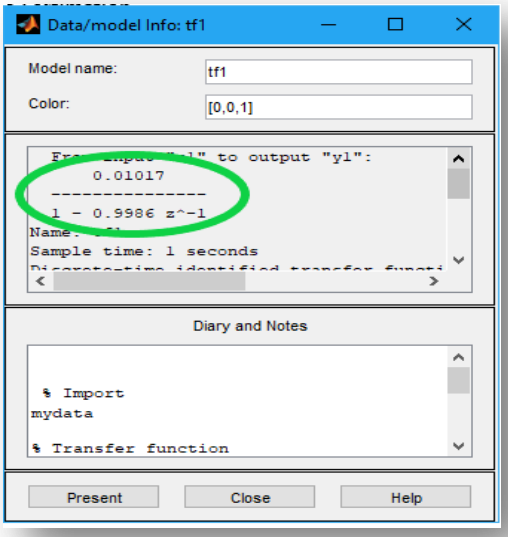


Figure IV.23: Récupération de la fonction de transfert estimée

IV.4 Commande du procédé thermique

L'étape de la commande du procédé thermique est constituée de deux parties.

- **La première partie** consiste à utiliser l'outil Matlab *PID Tuning* pour déterminer les différents paramétrés *Kp, Ki et Kd* des régulateurs P, PI et PID en fonction de notre objectif de commande.
- **La deuxième partie** consiste à implémenter sur Simulink puis sur la carte Arduino les correcteurs *P(z), PI(z)* et *PID(z)* .

  Cependant avant de commander le système thermique, il faut d'abord le tester en boucle ouverte afin d'analyser son comportement dans ce cas.

IV.4.1 Analyse du système en boucle ouverte

Nous allons analyser la réponse indicielle du système thermique en boucle ouverte pour une référence qui vaut 34°C. Le résultat obtenu est donné par **Figure IV.24.**

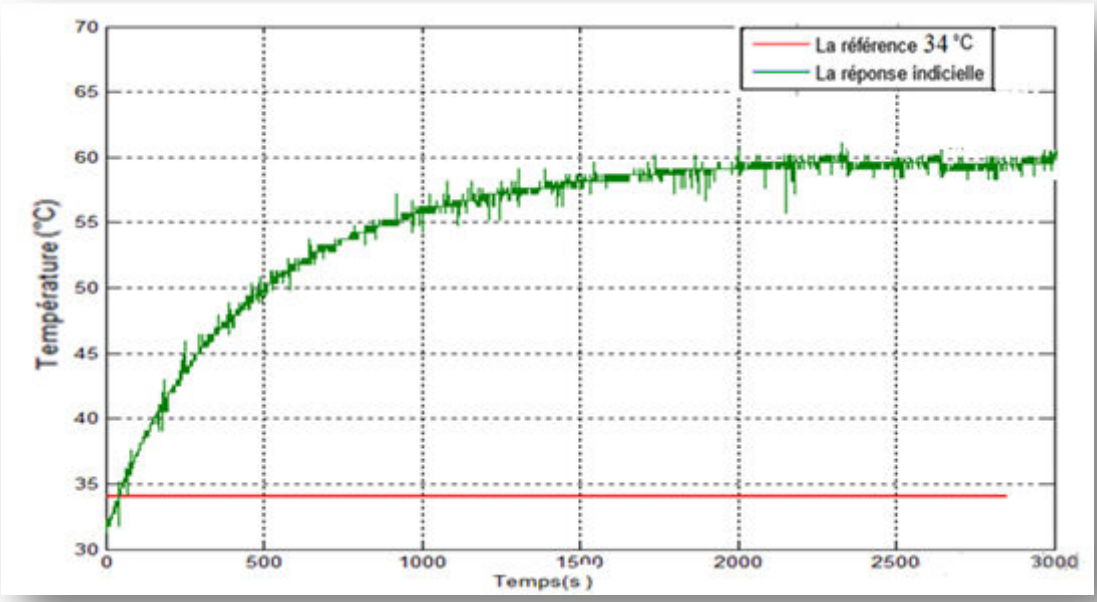


Figure IV.24: La réponse indicielle en boucle ouverte du système thermique en temps réel

A partir de Figure IV.24, on remarque que:

- L'expérience débute avec une température ambiante de 22°C.
- On voit bien que l'erreur statique est très grande (26°C !).
- Temps de réponse à 95% de la valeur finale de la température est égale à 1100s (18.5mn), donc le système est très lent pour qu'il se stabilise.

Toutes ces caractéristiques montrent que le système thermique a besoin d'etre controler.

IV.4.2 Commande du système thermique

Dans cette section, nous allons appliquer au système thermique des commandes numériques P, PI et PID apérs avoir expliquer comment les synthétiser dans Simulinket les implémenter dans Arduino.

IV.4.2.1 Synthèse du régulateur numérique

Nous allons expliquer les différentes étapes à suivre pour déterminer les gains Kp, Ki et Kd d'un régulateur PID en utilisant l'ogiciel Matlab PID Tuning et ça sera paraille pour la détermination des gains des commandes P et PI. Nous présentons d'abors, par Figure IV.25, la fenetre Matlab qui montre l'emplacement de l'outil PID tuning.

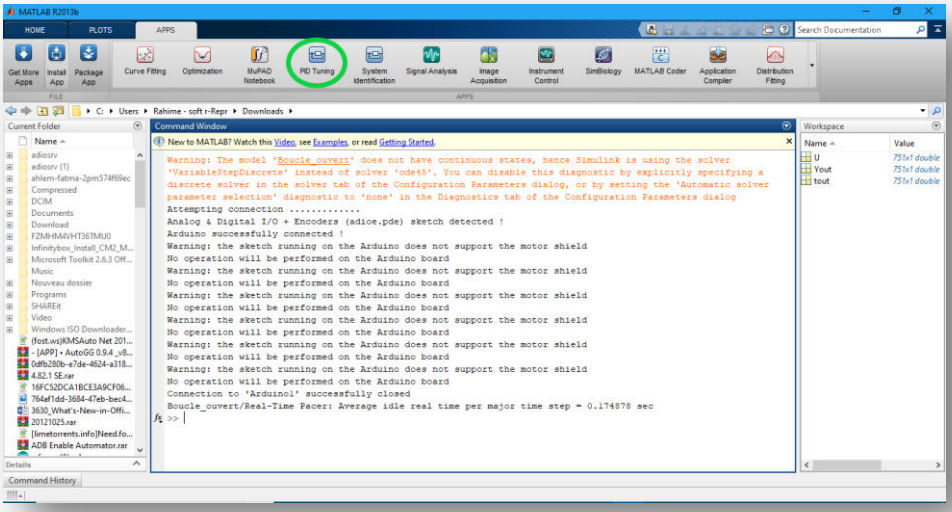


Figure IV.25: Emplacement de l'outil PID tuning.

1- Ouvrir l'outil PID Tuner et cliquer sur Import new plant

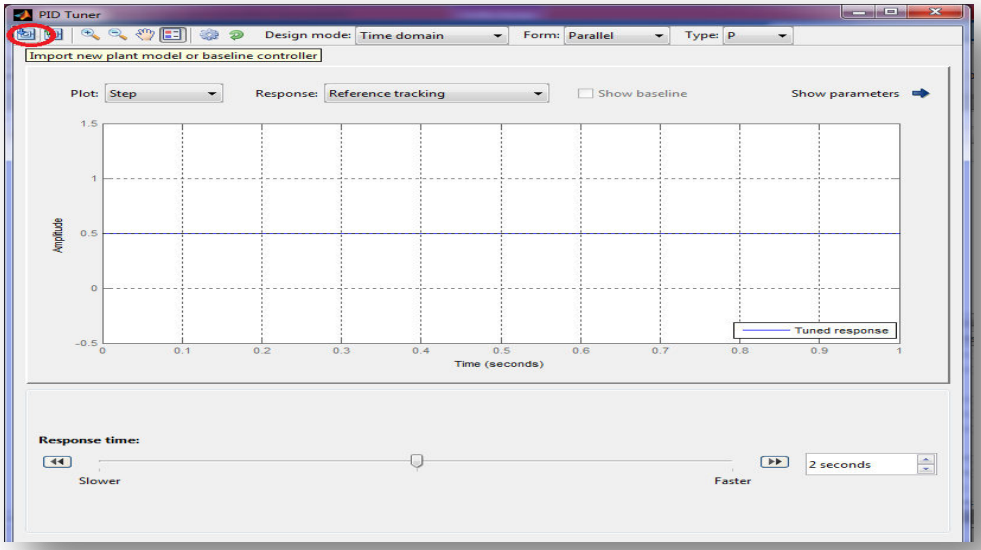


Figure IV.26: Interface de l'outil PID tuning

2- Une nouvelle fenêtre apparait dans laquelle on va sélectionner tf1 ensuite cliquer sur Import puis close.

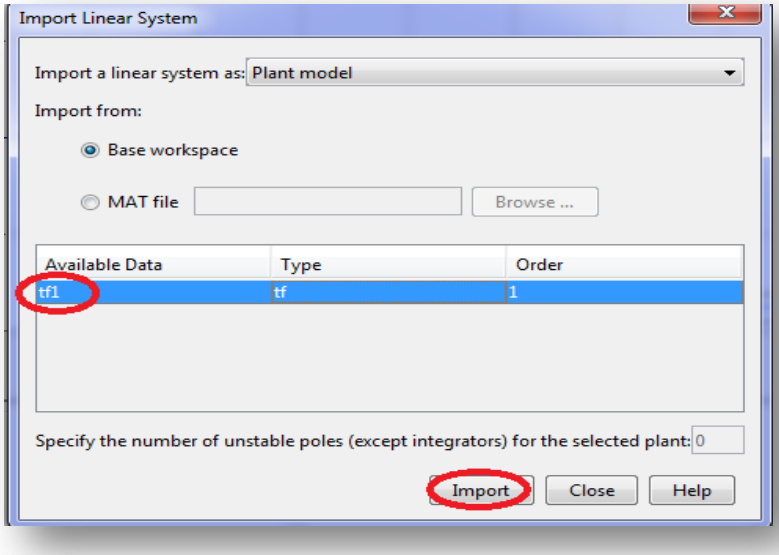


Figure IV.27 :Interface de l'outil « *Import Linear System* »

3- Revenir à la fenêtre *PID Tuner*, on peut choisir le type de régulateur à implémenter et les objectifs de la commande en boucle fermé et voir la réponse de la sortie du système.

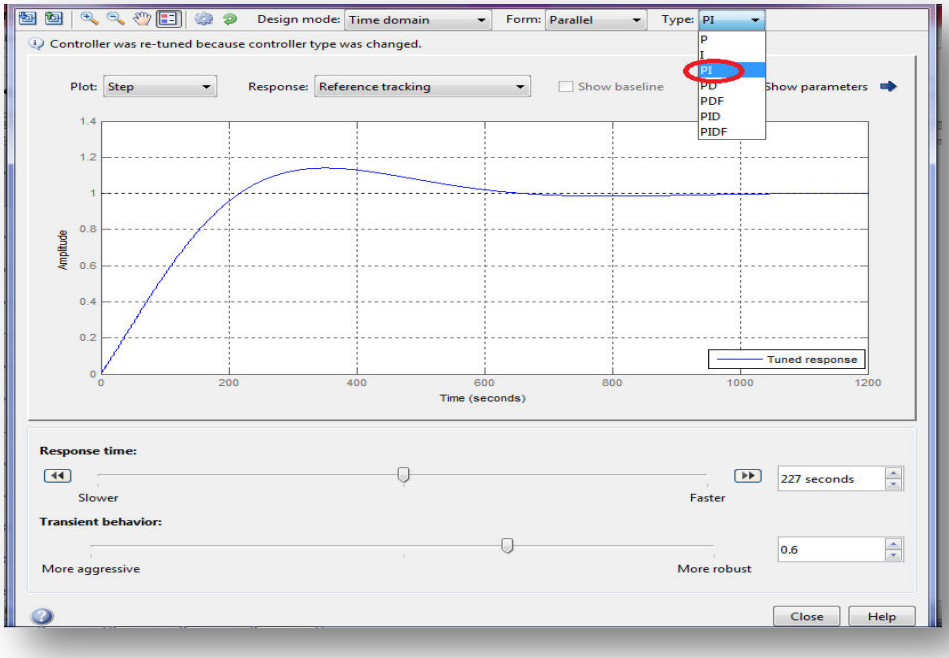


Figure IV.28: Choix de type de régulateur.

4- Cliquer sur la flèche de *show parameter* pour voir les paramétrés utilisés de notre régulateur ainsi que les performances du système en boucle fermée.

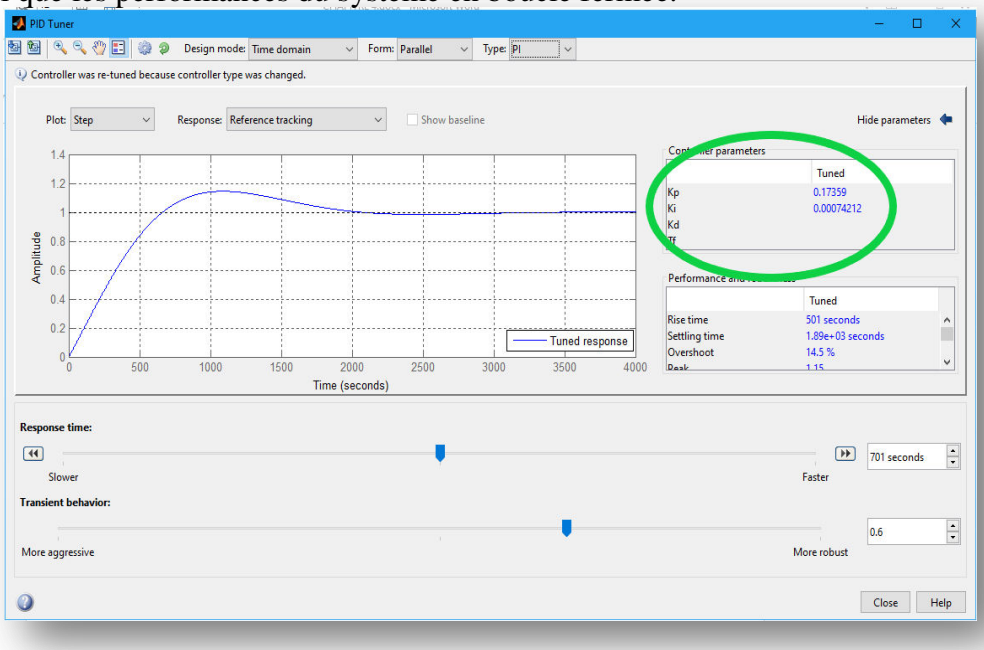


Figure IV.29 : Choix des paramètres du régulateur

IV.4.2.2 Impléntation de la commande sur la carte Arduino

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

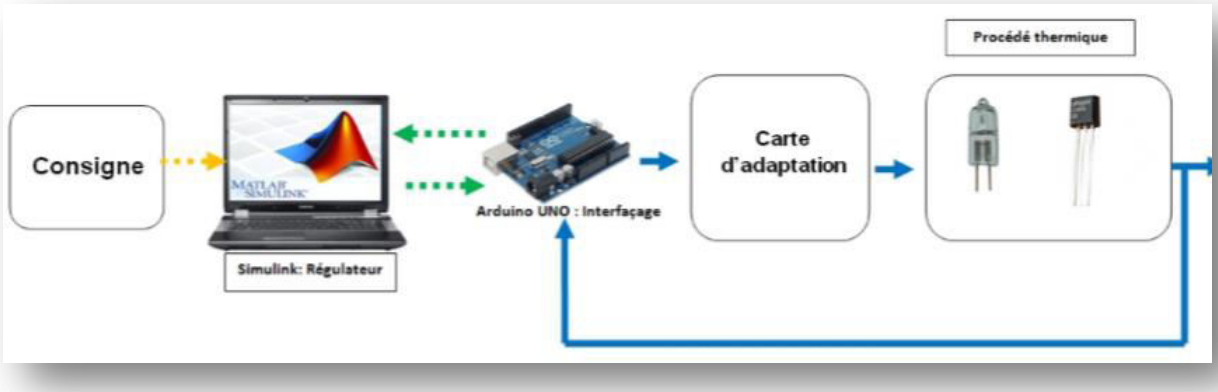


Figure IV.30 : Schéma Synoptique de la boucle d'asservissement à implémenter.

L'asservissement de notre procédé en temps réel est assuré par le schéma Simulink donné par Figure IV.31 qui regroupe la consigne de 34 C, la commande PID (z), le traitement de la température issue du capteur et l'envoie de la commande PWM.

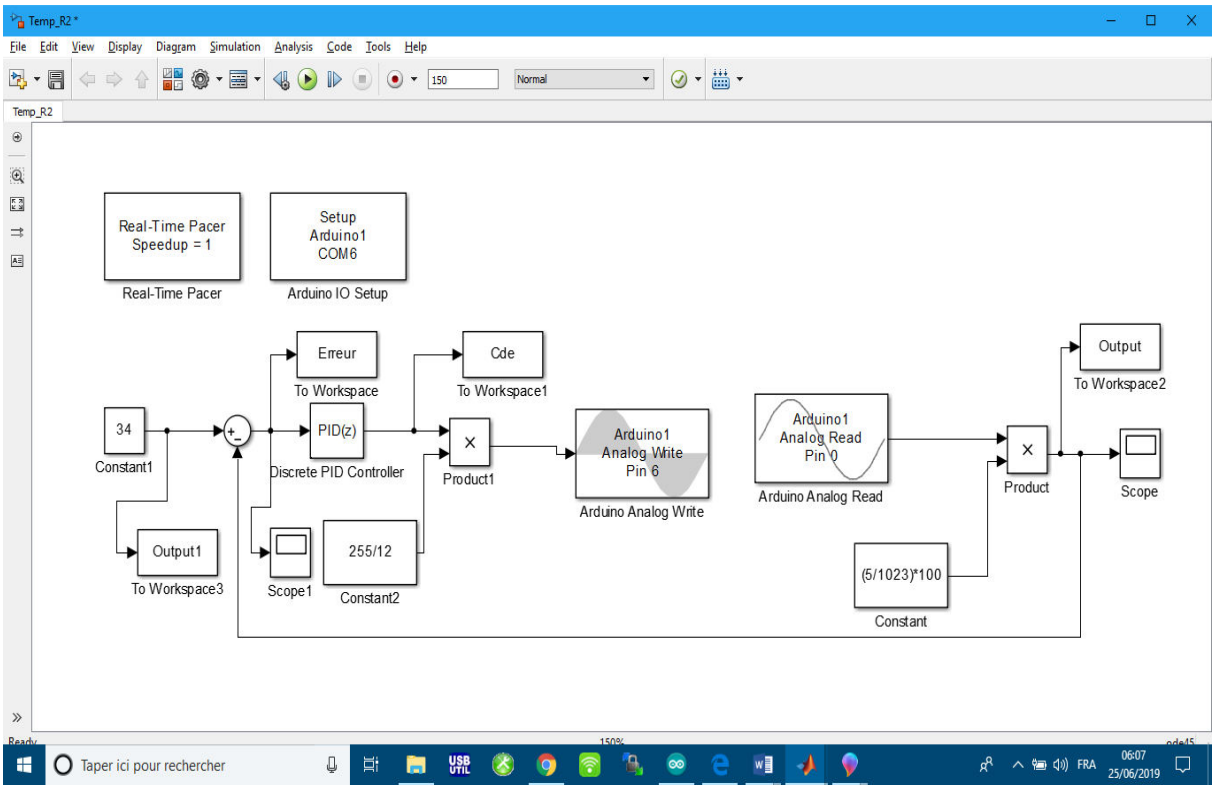


Figure IV.31 : Implémentation sur Simulink en temps réel

L'appui deux fois sur le bloc PID (z) permet d'introduire les paramétrés Kp, Ki et Kd et de configurer le régulateur selon notre objectif de commande.

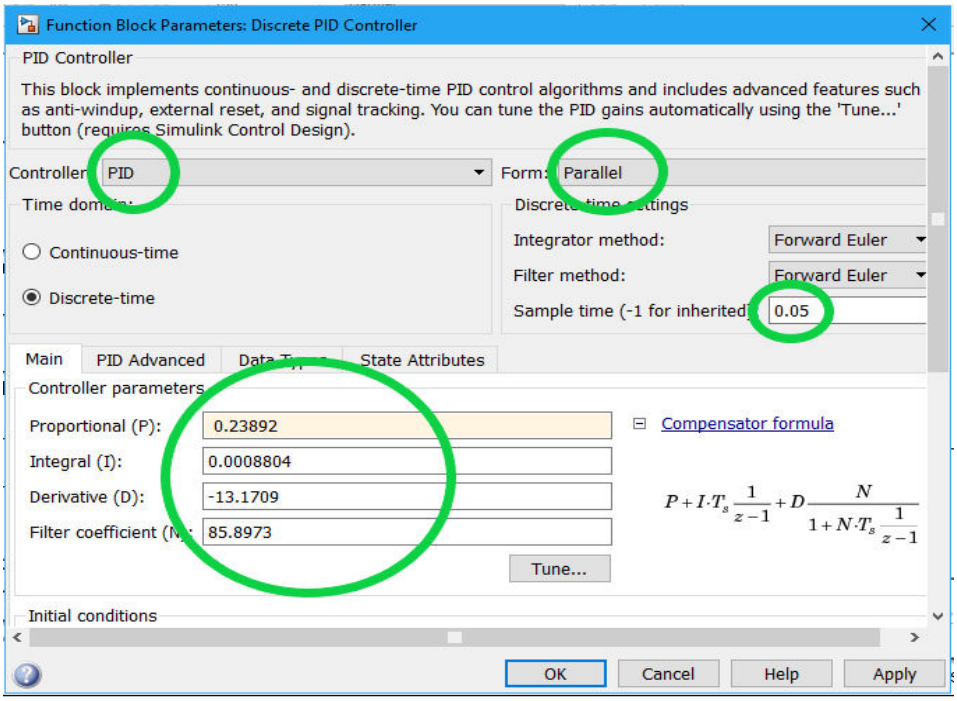


Figure IV.32: Configuration du régulateur et saisie de ses paramètres .

IV.4.2.3 Commade P appliquée au système thermique

Nous allons commander le système thermique par un régulateur Proportionel P numérique dont son equation est donnée par (III.15). Nous appliquons à ce système une consigne de 34°C (echelon) car on veut que la température reste autour de cette valeur. Pour cela Nous avons réalisé le schéma Simulink correspondant donné par Figure IV.33 dont le gain du P et donné par Figure IV.34.

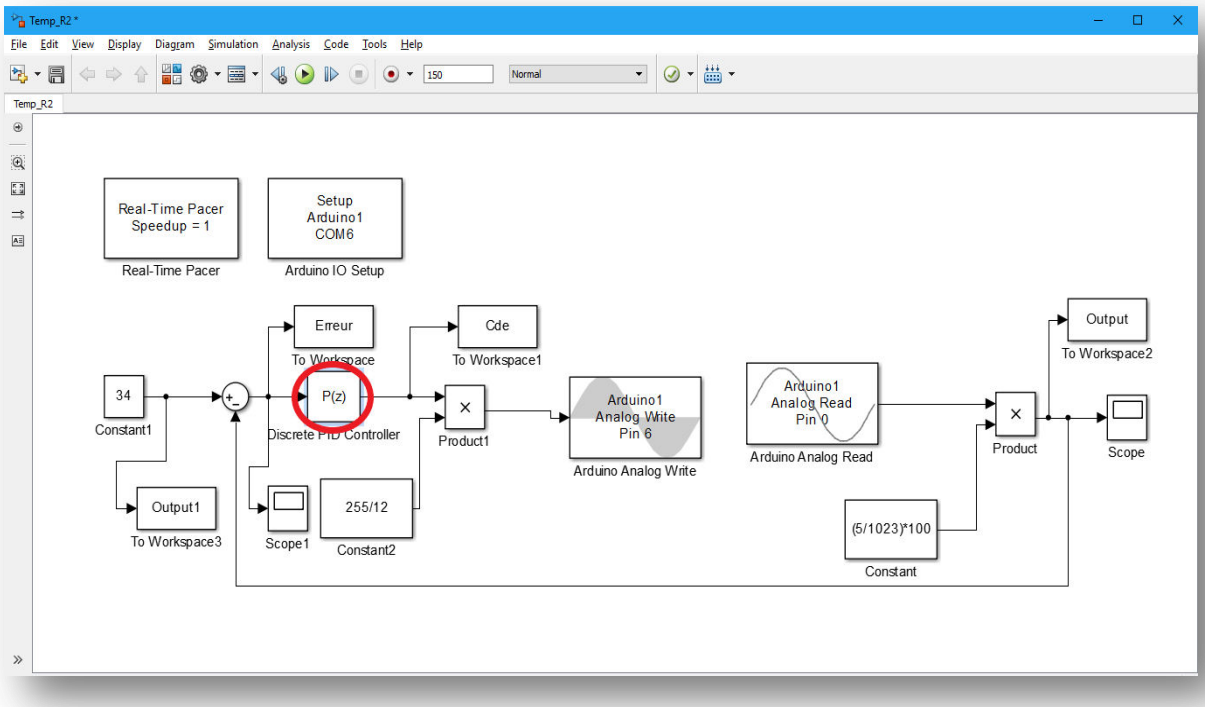


Figure IV.33: Implémentation de la commande P en temps réel du système sous Simulink

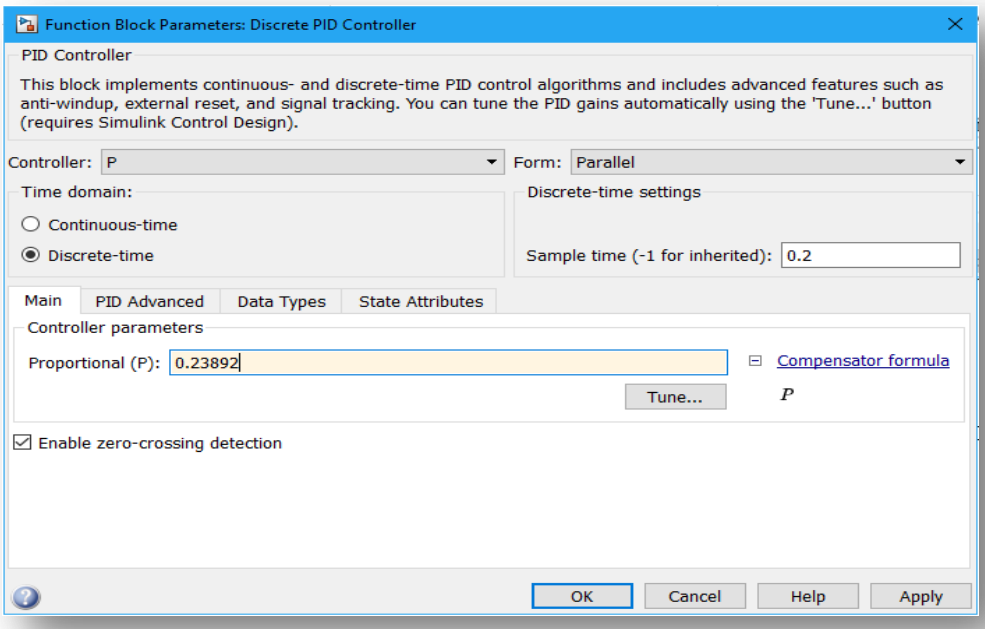


Figure IV.34: Choix du régulateur P et saisie de son gain

a- Expériences et résultats

Les résultats obtenus sont donnés par les Figures suivantes.

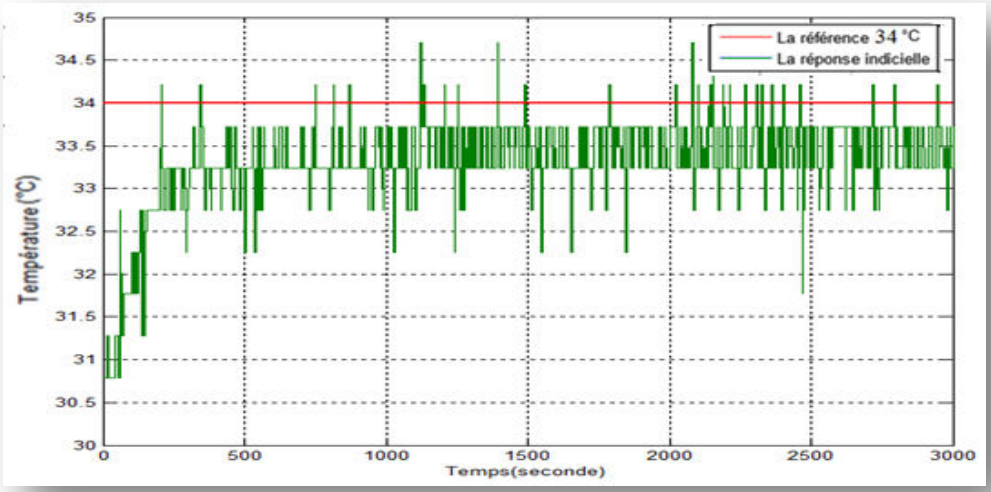


Figure IV.35 : Réponse indicielle du système thermique commandé par P.

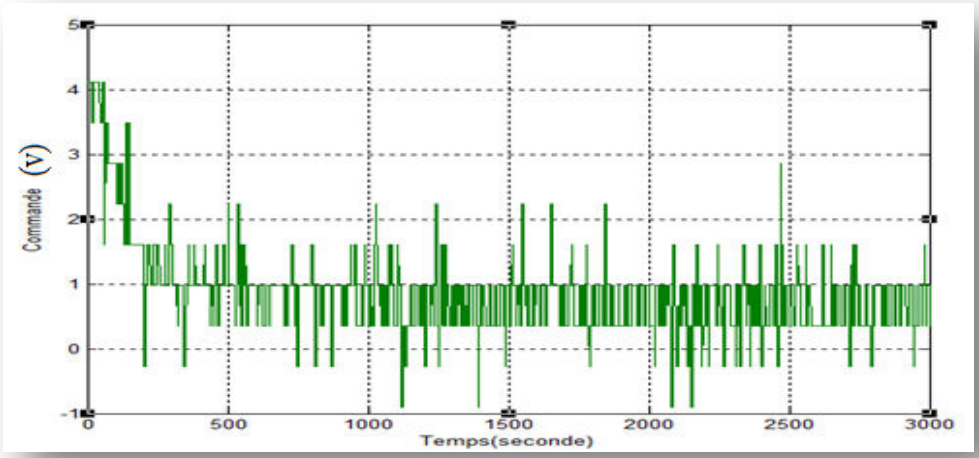


Figure IV.36 : Commande P appliquée au système thermique

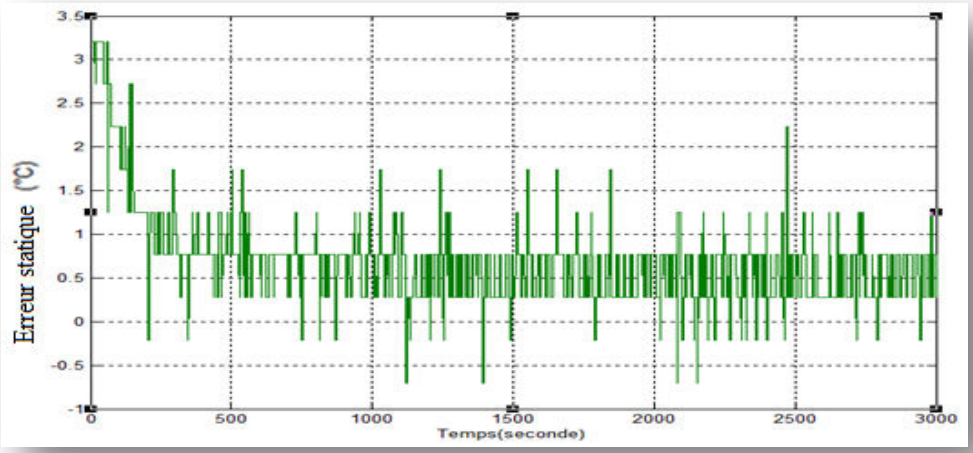


Figure IV.37 : Erreur statique de l'asservissement par P

b- Analyse des résultats

A partir des **Figures IV.35** on peut relever les remarques suivantes:

- L'expérience débute avec une température ambiante de 31°C.
- On voit bien qu'il y a une diminution de l'erreur statique (0.5°C) mais cette erreur ne s'annule pas au régime permanent et elle persiste. La **Figure IV.37** confirme ce résultat.
- Temps de réponse à 95% de la valeur finale de la réponse indicielle est égale à 166.6 s (2.7mn), donc le système est devenu plus au moins rapide garce au régulateur P si on le compare avec celui donné par Figure IV.24.
- La **Figure IV.36** représentant l'allure de la commande est proportionelle à celle de l'erreur statique donnée par **Figure IV.37**.

  On peut conclure donc que le régulateur P a juste diminué l'erreur de l'asservissement. Pour cela, nous allons ajouter à cette commande une action intégrale ce qui donne une commande PI (proportionnelle intégrale).

IV.4.2.4 Commade PI appliquée au système thermique

Nous allons commander maintenant le système thermique par un régulateur PI numérique dont son equation est donnée par (III.17). Nous appliquons à ce système une consigne de 34°C (echelon) car on veut que la température reste autour de cette valeur. Pour cela Nous avons éalisé le schéma Simulink en temps réel correspondant donné par Figure IV.38 dont les gains Kp et KI sont données par Figure IV.39.

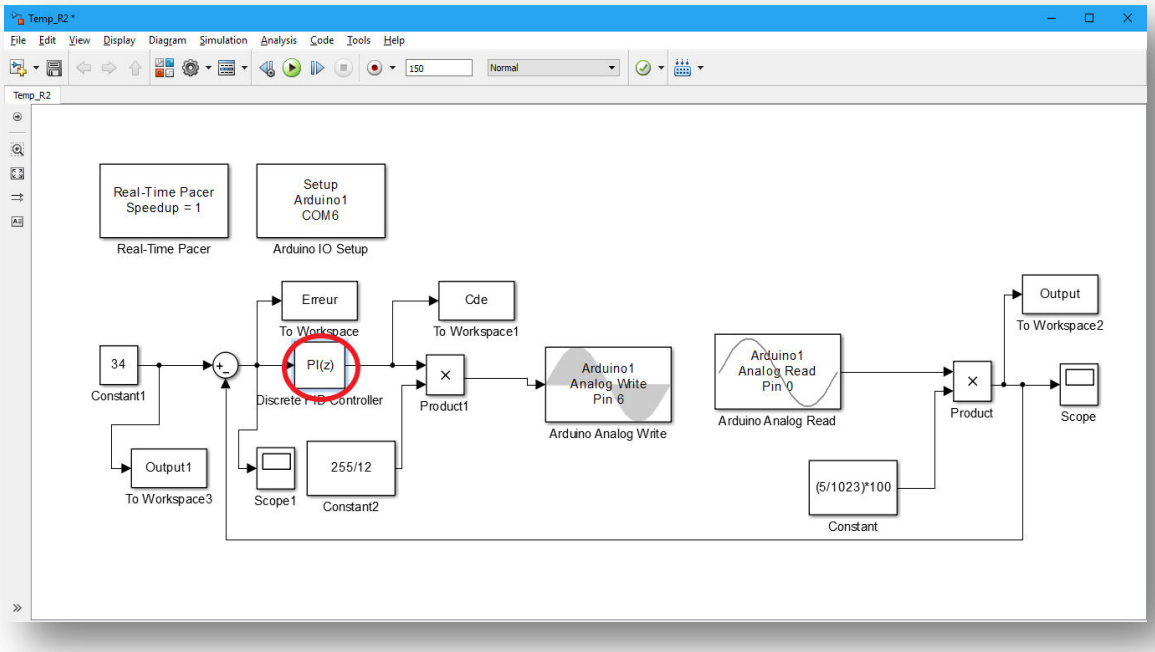


Figure IV.38: Implémentation de la commande PI en temps réel du système sous Simulink

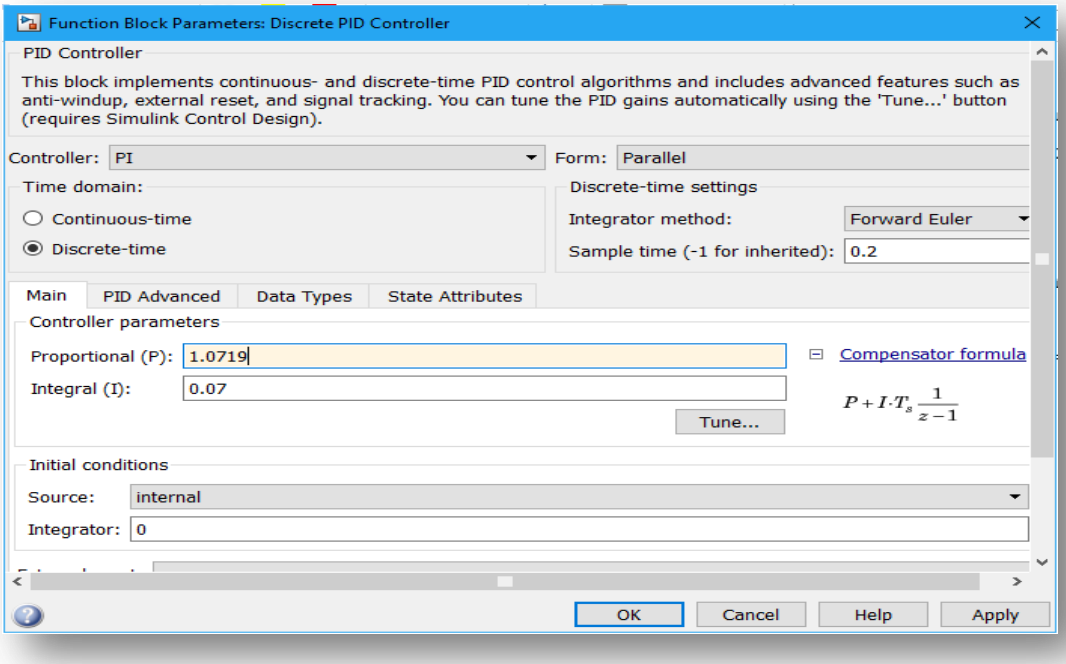


Figure IV.39 : Choix du régulateur PI et saisie de ses gains

a- Expériences et résultats

Les résultats obtenus sont donnés par les Figures suivantes.

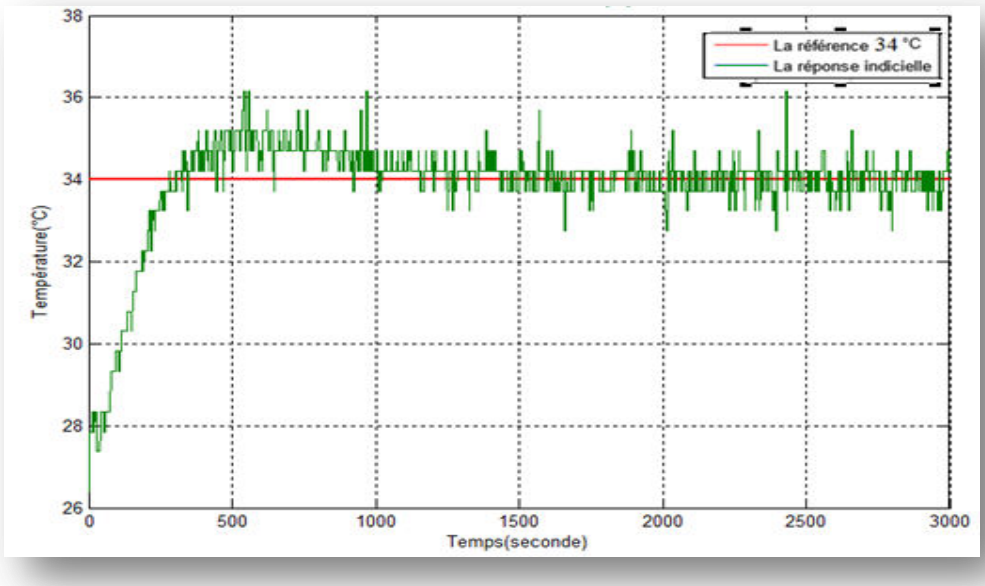


Figure IV.40 : Réponse indicielle du système thermique commandé par PI.

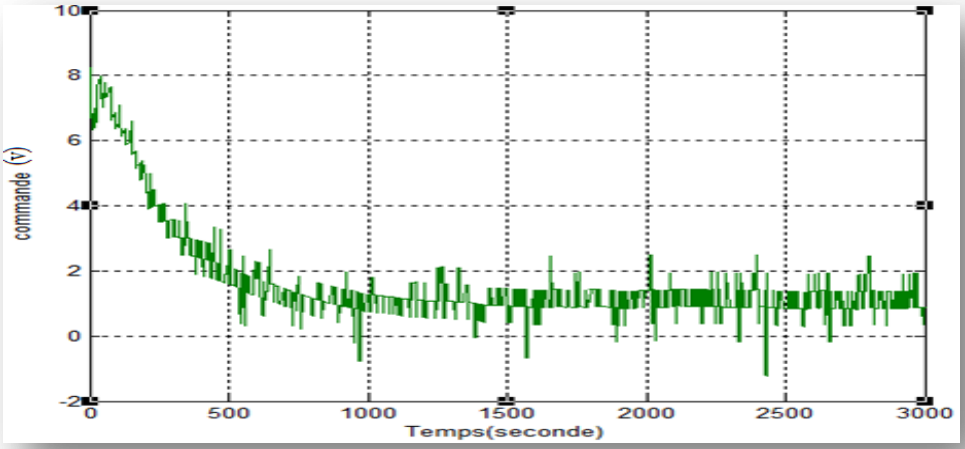


Figure IV.41 : Commande PI appliquée au système thermique

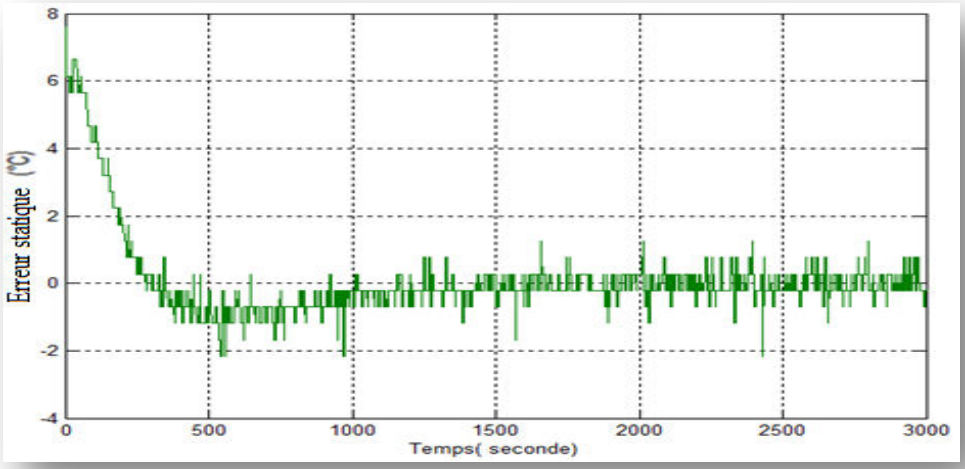


Figure IV.42 : Erreur statique de l'asservissement par PI

b- Analyse des résultats

A partir des **Figures IV.40** on peut relever les constations suivantes :

- L'expérience débute avec une température ambiante de 28°C.
- On voit bien qu'il y a une annulation de l'erreur statique (0°C) au régime permanent grace à l'effet de l'action intégrale de la commande PI. La **Figure IV.42** confirme ce résultat.
- Temps de réponse à 95% de la valeur finale de la réponse indicielle est égale à 300s (5mn), donc le système est devenu plus lent si on le compare avec celui donné par Figure IV.35 (2.7mn).
- On remarque aussi un legère dépassement de 1s à l'instant 500s .
- La **Figure IV.41** représentant l'allure de la commande PI est proportionelle à celle de l'erreur statique donnée par **Figure IV.41**.

On peut conclure donc que le régulateur PI a pu annuler l'erreur de l'asservissement au régime permanent et c'est son point fort, mais il a causé un petit retard de l'établissement du système asservis. Ce petit retard n'influe pas sur l'asservissement puisque les systèmes thermiques sont connus par leur lenteur. Nous avons relevé aussi le légère dépassement qui peut être annulé si on ajoute à cette commande l'action dérivée, ce qui donne le PID (Proportionnel, Intégral, Dérivé)

IV.4.2.5 Commade PID appliquée au système thermique avec consigne constante

Nous allons commander maintenant le système thermique par un régulateur PID parallèle numérique dont son equation est donnée par (III.25). Nous appliquons à ce système une consigne de 34°C (echelon) . Pour cela nous avons réalisé le schéma Simulink en temps réel correspondant donné par Figure IV.43 dont les gains Kp, KI et Kd sont données par Figure IV.44.

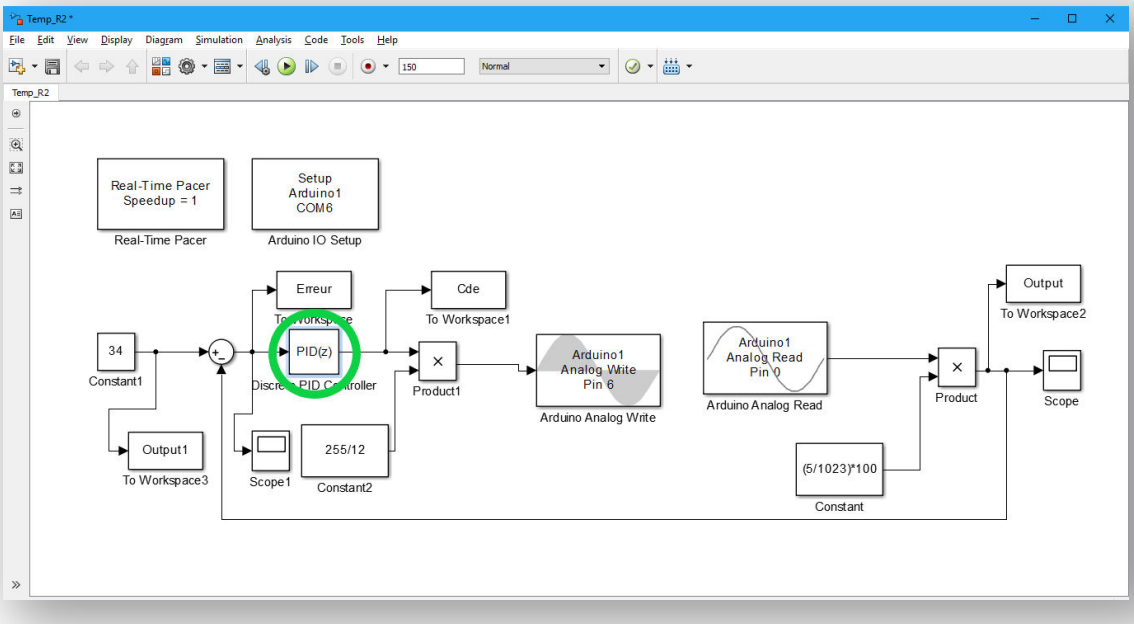


Figure IV.43: Implémentation de la commande PID en temps réel du système sous Simulink

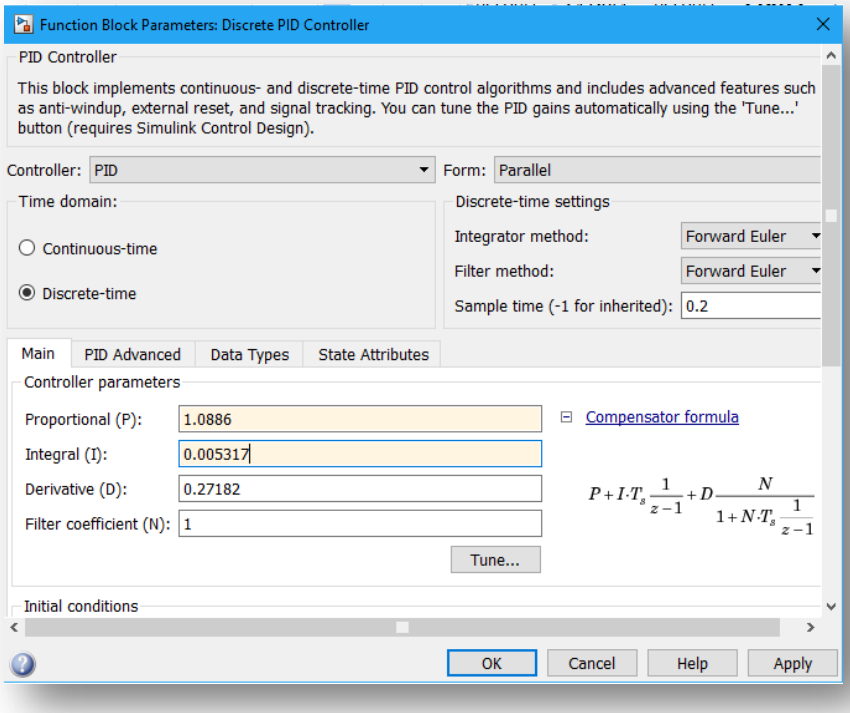


Figure IV.44 : Choix du régulateur PID et saisie de ses gains

a- Expériences et résultats

Les résultats obtenus sont donnés par les Figures suivantes.

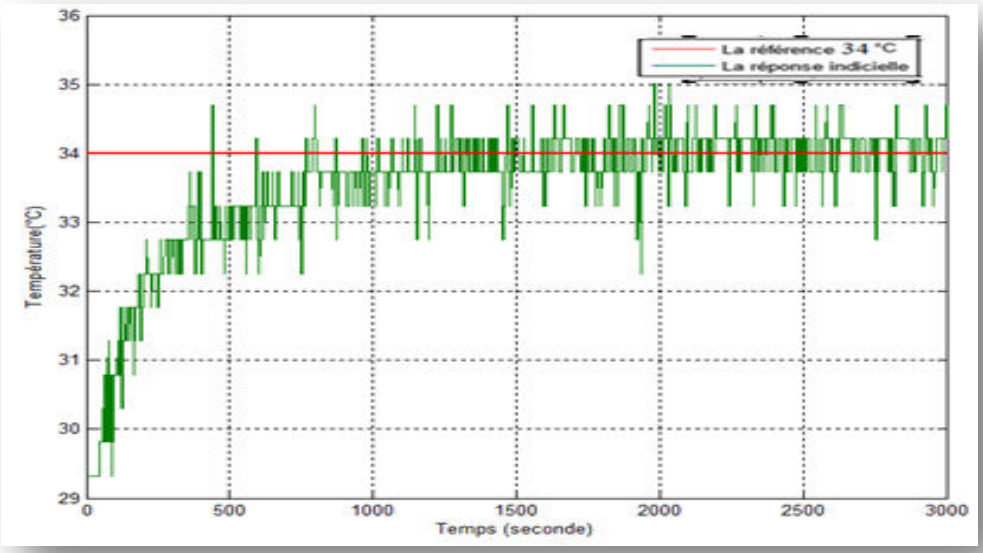


Figure IV.46 : Réponse indicielle du système thermique commandé par PID.

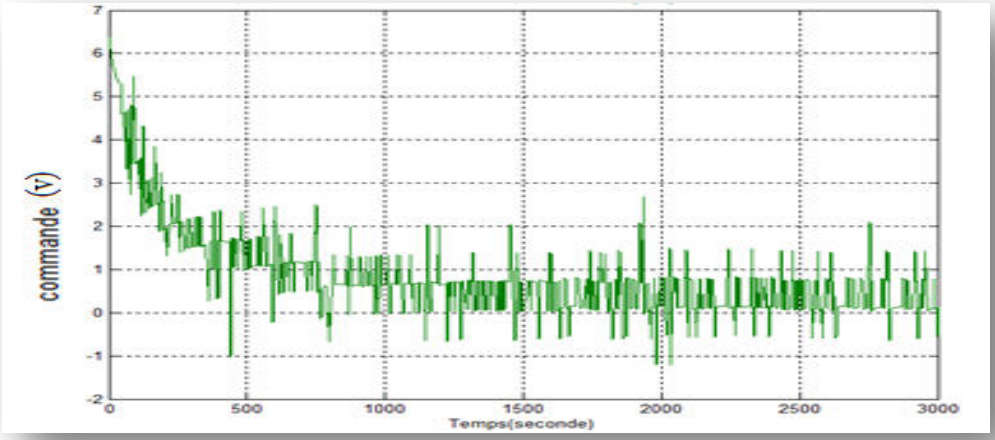


Figure IV.47 : Commande PI appliquée au système thermique

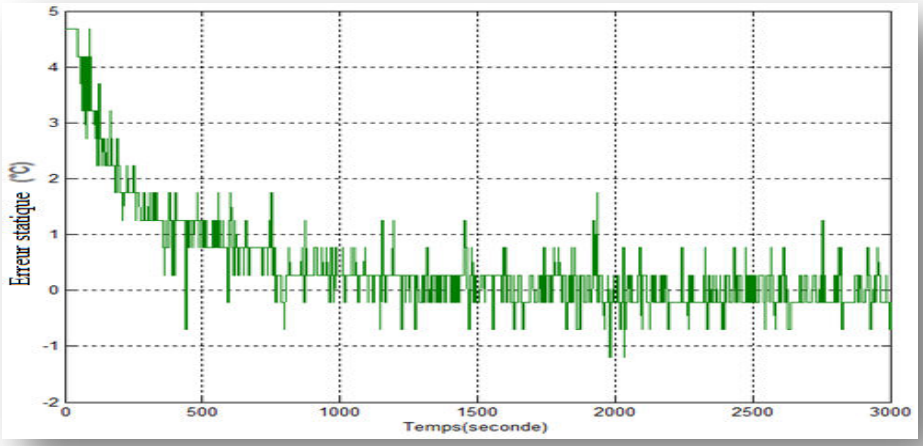


Figure IV.48 : Erreur statique de l'asservissement par PID

. b- Analyse des résultats

A partir des **Figures IV.46** on peut relever les remarques suivantes :

- L'expérience débute avec une température ambiante de 29.4°C.
- On voit bien qu'il y a une annulation de l'erreur statique (0°C) au régime permanent grace à l'effet de l'action intégrale de la commande PID. La **Figure IV.48** confirme ce résultat.
- Temps de réponse à 95% de la valeur finale de la réponse indicielle est égale à 250s (4.1mn), donc le système est devenu plus rapide si on le compare avec celui donné par Figure IV.43 (5mn).
- On remarque aussi qu'il n'y a pas de dépassement et ça grace à l'effet de l'action dérivée .

- La **Figure IV.47** représentant l'allure de la commande PID est proportionelle à celle de l'erreur statique donnée par **Figure IV.48**

  On peut conclure donc que le régulateur PID a amélioré les performances de l'asservissement par rapport aux expériences précédentes car il a pu annuler l'erreur statique et rend le temps de réponse acceptable.

IV.4.2.6 Commade PID appliquée au système thermique avec consigne variable

Nous allons commander maintenant le système thermique par un régulateur PID parallèle numérique dont son equation est donnée par (III.25) en appliquant à ce système une consigne variable. Cette dernière a été construit en utilisant le bloc Signal Builder de Simulink qui se trouve dans la bibliothèque Simulink/Source. Elle prend les valeurs suivantes:

- Durant l'intervalle de temps [0s, 1000s] la référence est contante et égale à 34°*C*.

- Durant l'intervalle de temps [1000s 2000s] la référence est contante et égale à 38°*C*.

- Durant l'intervalle de temps [2000s, 3000s] la référence est contante et égale à 36°*C*.

Pour cela nous avons réalisé le schéma Simulink en temps réel correspondant donné par Figure IV.49 dont les gains Kp, KI et Kd sont données par Figure IV.50.

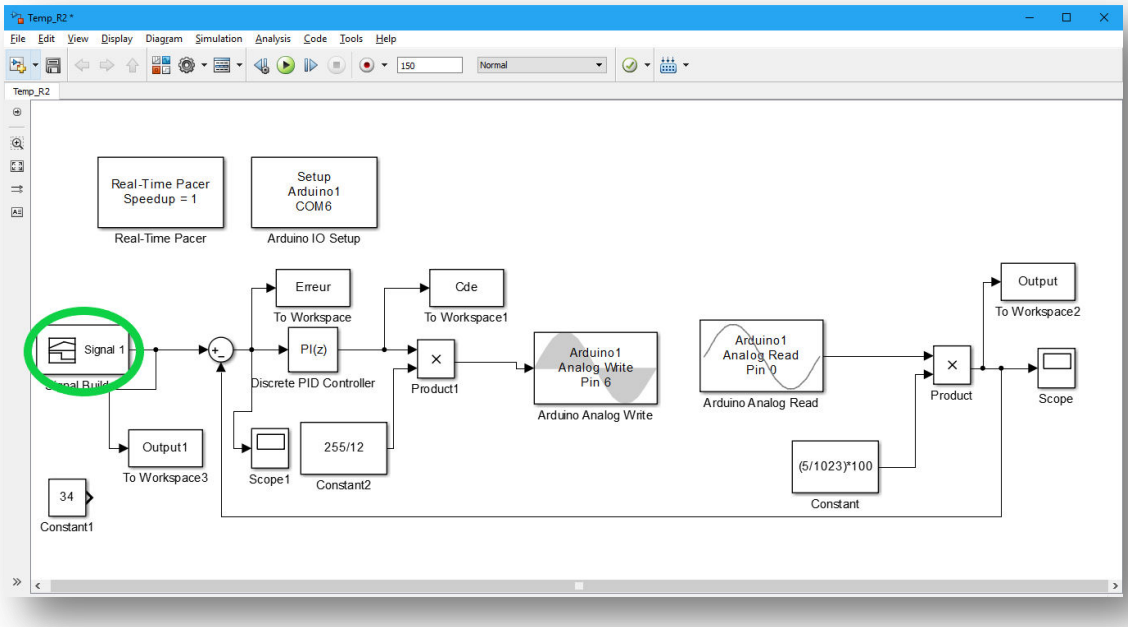


Figure IV.49: Implémentation de la commande PI en temps réel du système sous Simulink

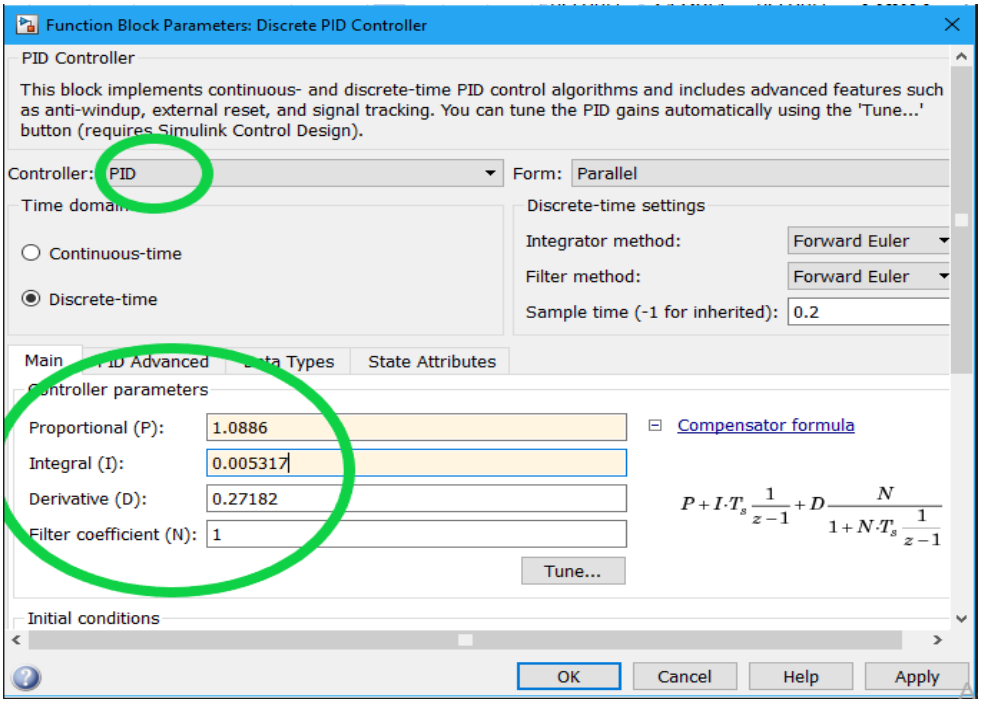


Figure IV.50: Choix du régulateur PID et saisie de ses gains (consigne variable)

a- Expériences et résultats

Les résultats obtenus sont donnés par les Figures suivantes.

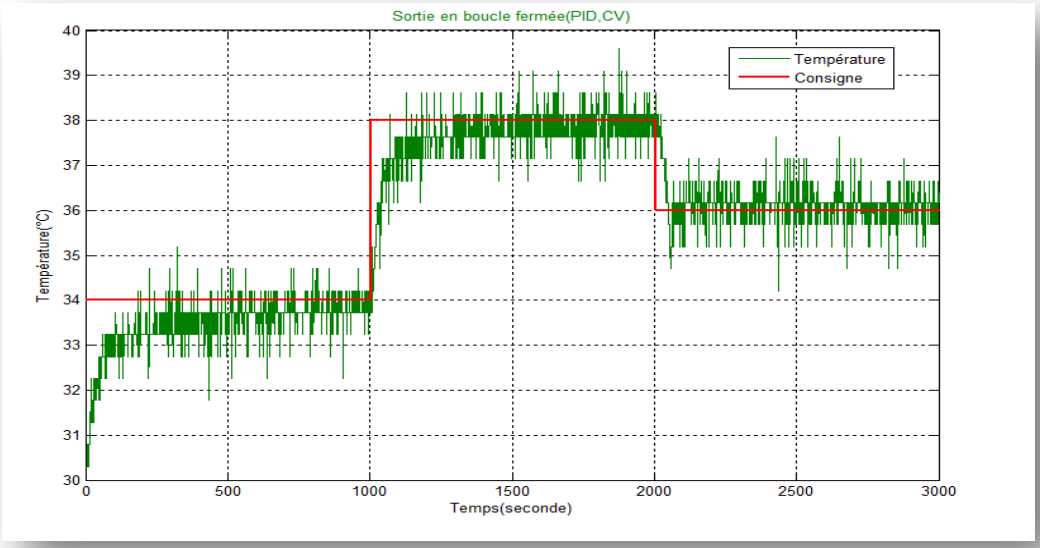


Figure IV.51: Réponse indicielle du système thermique commandé par PID (CV).

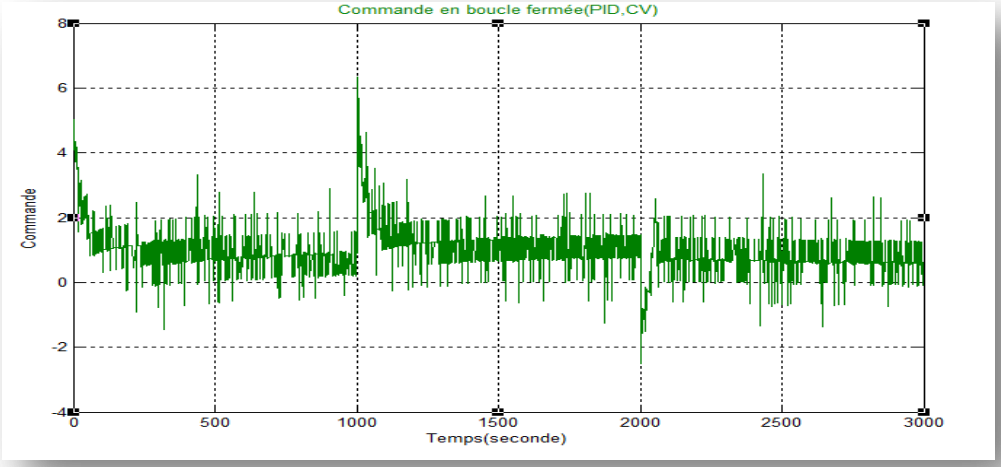


Figure IV.52: Commande PID appliquée au système thermique (Consigne Variable)

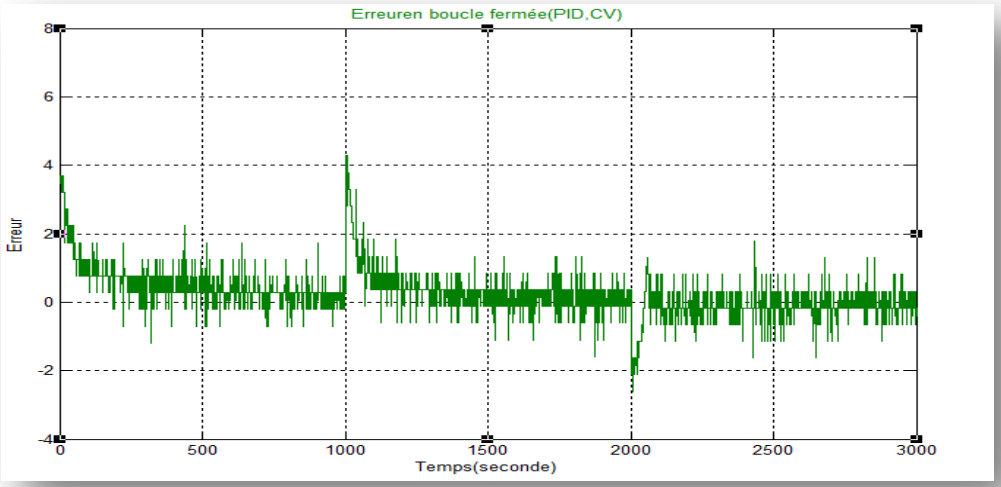


Figure IV.53: Erreur statique de l'asservissement par PID (consigne variable)

b- Analyse des résultats

A partir des **Figures IV.51** on peut relever les constations suivantes :

- L'expérience débute avec une température ambiante de 30.5°C.
- On voit bien qu'il y a une annulation de l'erreur statique (0°C) au régime permanent pendant les intervalles [1000s 2000s] et [2000s 3000s] et cela revient à l'effet de l'action intégrale de la commande PID. Cependant durant le temps [0s 1000s] on remarque qu'il y a une erreur de 0.3 °C. La **Figure IV.53** confirme ce résultat.

- On remarque qu'au niveau de variation de la consigne il y a des petites variations de la réponse indicielle et cela revient au temps (temps de réponse) qu'il le faut pour que la température mesurée ateind celle souhaitée (réference) à l'aide de PID .
- La **Figure IV.52** représentant l'allure de la commande PID est proportionelle à celle de l'erreur donnée par **Figure IV.53**.

On peut conclure donc malgré l'application d'une consigne variable au système, le régulateur PID a pu amélioré les performances de l'asservissement.

IV.5 Conclusion

Dans ce chapitre, nous avons présenté les différentes étapes à suivre pour l'implantation de notre système de régulation de température après avoir attribuer un modèle représentatif au système thermique en utilisant une méthode d'identification qui est intégrée dans Matlab. Nous avons expliqué aussi comment synthétiser les commandes P, PI et PID sous Simulink afin de contrôler la température de notre système.

Les résultats expérimentaux et de simulation que nous avons obtenus ont validé la maquette que nous avons réalisée et ils ont montré l'intérêt d'utiliser la carte Arduino Uno comme une interface entre Matlab et le procédé thermique. Ces résultats ont montré aussi que c'est l'application de PID au système chauffant qui a pu améliorer les performances de l'asservissement puisque il l'a rendu plus précis et plus rapide.

Conclusions et Perspectives

Conclusions et Perspectives

Le travail qui a été demandé dans le cadre de ce projet fin d'étude a été, pratiquement, accompli. En effet, nous avons achevé la réalisation d'une maquette et nous avons réalisé un dispositif permettant la mesure simultanée. Ce dispositif permet de faire la régulation de température d'un système thermique. L'acquisition des mesures de température a été assurée par le capteur de température LM35, le transfert des données a été assuré par la carte Arduino qui joue le rôle d'une interface qui est une plateforme matérielle très répandue et enfin la régulation a été réalisée à base de l'environnement Matlab/Simulink qui est un programme de modélisation/simulation professionnel bien connu.

Les résultats obtenus, à partir des différentes expériences, sont très satisfaisants. Cela valide les circuits électroniques qui composent la maquette et il montre l'intérêt de développement d'une telle communication Matlab/Arduino.

Au cours de notre projet, nous avons rencontré quelques difficultés techniques de la pratique, Nous commençons par évoquer l'utilisation du capteur de température LM35. Après plusieurs essais en changeant plusieurs fois ce capteur, on a découvert qu'il ne donne pas de vraies mesures de la température car ce sont des capteurs défaillants qui existent sur le marché et à cause de ça nous avons perdu beaucoup de temps pour savoir l'origine du problème. Nous avons donc changé ce capteur par un autre LM 35 qui admet un système de calibrage. Cela nous a assuré d'avoir des mesures plus précises. Cette première étape nous a bien servi d'excellentes expériences afin de mieux procéder avec ce genre d'instruments de mesures.
Une autre situation à noter, c'est la saturation ou la limitation du port analogique de la carte Arduino UNO. Les signaux d'entrée et de sortie sont limités entre 0V et 5V, et nous avions besoin de lire des signaux au-dessus de 5V car le système thermique fonctionne avec 12V. Cette étape nous a obligée à utiliser des circuits d'adaptations, on se rend compte que c'est une tâche très simple mais très couteuse en temps dont il faut tenir compte lors de l'étude.

Ce travail nous a permis donc d'enrichir nos connaissances dans plusieurs domaines, notamment dans le domaine de la régulation numérique et l'instrumentation de mesure de la température, la plate-forme Arduino et l'environnement Matlab/Simulink. Il nous a permis aussi d'approfondir nos connaissances théoriques et d'acquérir une bonne expérience au niveau de la réalisation pratique.

Enfin, nous pouvons dire que l'exemple pratique qui a été présenté dans ce mémoire, est considéré d'une part, comme étant un didacticiel permettant la mise en œuvre de cette nouvelle technologie et, d'autre part, un point de départ d'une série de prototypes pédagogiques pluridisciplinaires à échelonner sur une série de TP.

Ce travail est loin d'être terminé, il reste quelques points à examiner, à savoir :

- Ajouter à la maquette des dispositifs perturbateurs comme l'ouverture d'une porte ou des fenêtres.
- Placer des ventilateurs commandés par Arduino afin d'aider à refroidir le système thermique dans le cas où la température mesurée dépasse celle souhaitée.
- L'application de la commande adaptative au système thermique ou les gains du régulateur changent automatiquement en cas de besoin d'autres.
- L'application des commandes robustes au système thermique qui peuvent vaincre les différentes perturbations.

Références Bibliographique

[1] Ksatria. A et al: "Desing and Implementation of PID Control Based Incubator". Journal of Theoretical and Applied Information Technology. Vol.70 No.1, ISSN: 1992-8645, pp.19-24. 2014.

[2] Georges .A et al: "Les capteurs en instrumentation industrielle". 8ème édition. Paris, 2017.

[3] Ferguson .T:"Mesurer des températures par thermocouples"., Journal of National Instruments Corporation. 2014.

[4] Sintclair .R: "Sensors and transducers", NEWNES 2001

[5] Webster. J. G: "Measurement Instrumentation and sensors Handbook". Taylor&Francis Ltd, 2011.

[6] Bansal. H, Mathew.L and Gupta.A: " Controlling of Temperature and Humidity for an Infant Incubator Using Microcontroller".International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE), Vol. 4, Issue 6, 2015

[7] Muthuvignesh. M, Karthick.M and Nallakaruppan.M : " Temperature Control of Steam Using Microcontroller Arduino MEGA 2560". International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET). Vol. 6, Issue 3, March 2017

[8] Muslim .M.A, and Salmi. A: " Design and Implementation of Temperature Controller for a Vacuum Distiller". Proceeding of the Electrical Engineering Computer Science and Informatics. Vol 1, N 1, pp.198-201.2014.

[9] http://ctms.engin.umich.edu/CTMS/index.php?aux=Activities_Lightbulb

[10] ZERZRI. M.A: "Arduino et Simulink/Matlab un outil innovant à coût réduit pour le prototypage" . Journal sur l'enseignement des sciences et technologies de l information et des systèmes. Vol. 12, 2013.

[11] Hedengren. J.D: " Hands on Process Control", Brigham Young University. 2014

[12] Karaoui. L. A et Kedir. H: "Modélisation et commande d'un système de régulation de température". Master en Electrotechnique, spécialité : Commande des systèmes électriques. Centre Universitaire Belhadj Bouchaib d'Ain-Temouchent. 2017.

[13] Melgar. A and Diez. C: "Arduino and Kinect Projects: Design, Build, Blow Their Minds". 2014.

[14] Affagard. B and Géridan. J.M Projets: "créatifs avec Arduino"(. Lafargue, 20011

[15] Bartmann. E: "Le grand livre d'Arduino ". 3e édition, EYROLESS. 2018.

[16] https://www-soc.lip6.fr/trac/sesi-peri/chrome/site/cours/peri_2016_6_arduino-4p.pdf

[17] http://owni.fr/2011/12/16/arduino-naissance-mythe-bidouille/

[18] Juillot. G:" La programmation des ATMEL". support de cours. 2003.

[19] Lamri. O:" Commande des charges 220V par la reconnaissance vocale sous système Android et carte Arduino UNO"; Mémoire de Master en Automatique et Informatique Industriel . Université Mohamed Khider de Biskra. 2018

[20] Guenaoua. L: "Commande en position du a MCC par Arduino". Mémoire de Master en Instrumentation et contrôle industriel. Université Badji Mokhtar Annaba.2017.

[21] http://www.mathworks.com/academia/arduino-software/arduino-simulink.html

[22] Hoang. L.H "Introduction à MATLAB et Simulink" . Support de cours , Département de génie électrique et de génie informatique Université Laval Québec, CANADA. 2010.

[23] GRANJON. Y: "Automatique, Système linéaires, nonlinéaires, à temps continu, à temps discret, représentation d'état: Cours et exercices corrigés". 2ème édition, DUNOD, Paris, 2010

[24] TLIBA. S, JUNGERS.M et CHITOUR. Y:"Commande des Processus , Asservissements numériques". Polycopié de cours, Université Paris-Sud XI - ENS de Cachan, 2005.

[25] DETCHRAT, et. al. "IMC-Based PID Controllers Design for Two-Mass System", IMECS Journal, Vol. II, Hong Kong. 2012

[26] ZOUARI. F et.al.: “Adaptive Internal Model Control of a DC Motor Drive System Using Dynamic Neural Network" Journal of Software Engineering and Applications, Scientific Research, Vol. 5, p.168-189, 2012.

[27] IBRAHIM. k: “IMC based automatic tuning method for PID controllers in a smith predictor configuration" , Science Direct Journal, 2004

[28] RIVOIRE. M and FERRIER. J.L: "Cours d’Automatique Tome1: Signaux et Systèmes". Edition Eyrolles, 1995

[29] BORNE .G, et al: "Automatique Analyse et Régulation des Processus Industriels, Régulation Numérique" ,Tome 2. Méthodes et pratiques de l’ingénieur. Editions Technip, 1993.

[30] Crosnier. A, Abba. G, Jouvencel. B, and Zatata. R:" Automatique Ingénierie de la commande des systèmes techniques de base". Technosup. Ellipses, 2001.

[31] Sevely. Y, Abatut. J.L, and Roubellat. F: "Système et asservissement linéaires échantillonnés". DUNOD Université, 1989.

[32] Blanchet.G and Prado. J: "Elements d'automatique". Collection Pédagogique de Télécommunication. Ellipses, 1995.

[33] www.aduino.cc .