République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU**G**ENIE **E**LECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'ELECTRONIQUE

Mémoire de Fin d'Etudes
De MASTER PROFESSIONNEL

Domaine : **Sciences et Technologies** Filière : **Génie électrique**

Option : **Electronique industrielle.**

Thème :

Conception et réalisation d'une Commande d'un système thermique à base d'Arduino

Promoteur :

Mr. IDJERI B.

Etudié par :

Mr. BELABED Karim

Mr. ARAB SAID

Promotion 2017

Remerciements

En premier lieu, Dieu tout puissant de nous avoir donné le courage, la volonté et la patiente afin de munir à bien ce modeste ce travail.

Nous tenons à adresser nous vifs remerciements à notre promoteur ***Mr IDJERI. B*** *pour ses orientations et précieuse conseils tout le long de notre travail.*

Nous tenons également à remercier tous les membres du jury qui ont accepté d'évaluer notre modeste travail et qui nous ferons honneur de juger et d'enrichir notre travail par leurs proposition.

Un grand remerciement pour le PDG de L'Office National de Métrologie Légale ***Mr MESSILLI .R*** *et au Directeur régional centre de Gestionnaire du Réseau de Transport du Gaz* ***Mr HESSAS .L***

Dédicaces

Je tiens à dédier ce modeste travail à :
A mes très chers parents qui n'ont jamais cessé de nous procurer soutien et encouragements.
A ma femme et ma sœur.
A mes frères surtout AZIZ et MEHREZ.
A mes cousins & cousines.
A toute ma famille.
A tous mes amis (es) sans exception.

Saïd.

Je dédie ce travail :
A la mémoire de mon père
A ma grand-mère.
A ma mère
A mon frère.
A ma femme.
A mon fils.
A tous mes amis (es) sans exception

Karim.

Sommaire

Introduction générale

L'exploitation de la commande dans les systèmes automatisés tel que les systèmes thermiques joue un rôle très important dans le domaine industriel. Dans ce contexte, de nombreux travaux ont été réalisés ces dernières années, tentant d'augmenter leur performance, leur robustesse et leur fiabilité. Les évolutions conjuguées de l'électronique et de l'informatique ont révolutionné le domaine de la commande.

Dans ce projet nous allons présenter la conception d'un système thermique composé d'un capteur de température, une lampe et un ventilateur qu'on va tenter de commander avec un régulateur PID (Proportionnelle, Intégrale et Dérivée) conçu sur le logiciel Matlab. A cet effet nous avons opté pour l'usage d'une carte Arduino pour réaliser l'interfaçage entre le logiciel Matlab et le système thermique (acquisitions et envoi des données). Le logiciel Matlab est utilisé pour le traitement de données issues du système, en vue de d'assurer la régulation de la température selon la consigne désirée.

Le mémoire résumant le travail réalisé est organisé autour de trois chapitres :

Dans le premier chapitre nous allons présenter en premier lieu les différentes cartes Arduino, l'accent est porté sur la carte arduino UNO avec son logiciel de développement Arduino IDE. Ensuite nous allons introduire les outils qui peuvent être utilisés avec la carte.

Le deuxième chapitre, porte sur les outils théoriques exploités en commande numérique, tel que le pilotage, le guide…des processus physiques. Ainsi qu'une présentation sur le principe de la commande en boucle ouverte et fermée, à base du régulateur PID (Proportionnelle, Intégrale et Dérivée), Enfin les différentes fonctions qui seront utilisées dans des algorithmes de commande.

Le troisième chapitre est consacré à la conception et la réalisation d'un système de commande thermique à base de la carte arduino et le logiciel Matlab Simulink.

La conclusion synthétise le travail réalisé et donne un aperçu des perspectives qui peuvent être développées.

Chapitre I
Introduction à la carte programmable ARDUINO

1) Introduction :

Aujourd'hui, l'intégration des systèmes numériques dans le domaine de l'électronique tend à remplacer les systèmes analogiques, dans le but de leur miniaturisation formant ainsi des systèmes informatiques embarqués, toute en réduisant leur coût de fabrication. Il en résulte des systèmes plus complexes et performants pour un espace réduit. En fait depuis l'avènement de l'électronique, son avancée continuelle n'a cessé de progresser. Dans ce chapitre on va introduire une carte électronique de type Arduino qui combine les deux techniques de l'électronique numérique et analogique, utilisé comme un système embarqué avec quelques composants périphériques.

I.2) Le carte Arduino :

La Carte Arduino est un circuit imprimé en matériel libre (plateforme de contrôle) dont les plans de la carte elle-même sont publiés en licence libre dont certains composants de la carte: comme le microcontrôleur et les composants complémentaires qui ne sont pas en licence libre. Un microcontrôleur programmé peut analyser et produire des signaux électriques de manière à effectuer des tâches très diverses. Arduino est utilisé dans beaucoup d'applications comme l'électronique 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). Chaque module d'Arduino possède un régulateur de tension +5 V et un oscillateur à quartez 16 MHz. Pour programmer cette carte, on utilise l'logiciel IDE Arduino.

I.3 Les gammes de la carte Arduino :

Actuellement, il existe plus de 20 versions de carte Arduino, nous citons quelques un afin d'éclaircir l'évaluation de ce produit scientifique et académique (Voir Annexe N°A) [16]

Parmi ces types illustres dans l'annexe A, nous avons choisi une carte Arduino UNO (carte Basique). L'intérêt principal de cette carte est de faciliter la mise en œuvre d'une commande qui sera détaillée par la suite.

L'Arduino fournit un environnement de développement s'appuyant sur des outils open source comme interface de programmation. L'injection du programme déjà converti

par l'environnement sous forme d'un code «HEX» dans la mémoire du microcontrôleur se fait d'une façon très simple par la liaison USB. En outre, des bibliothèques de fonctions "clé en main" sont également fournies pour l'exploitation d'entrées-sorties. Cette carte est basée sur un microcontrôleur ATmega 328 et des composants complémentaires. La carte Arduino contient une mémoire morte EEPROM de 1 kilo octet. Elle est dotée de 14 entrées/sorties digitales (dont 6 peuvent être utilisées en tant que sortie PWM), 6 entrées analogiques et d'un quartz de 16 MHz, une connexion USB et possède un bouton de remise à zéro et une prise jack d'alimentation.

La carte est illustrée dans la figure ci-dessous. [14]

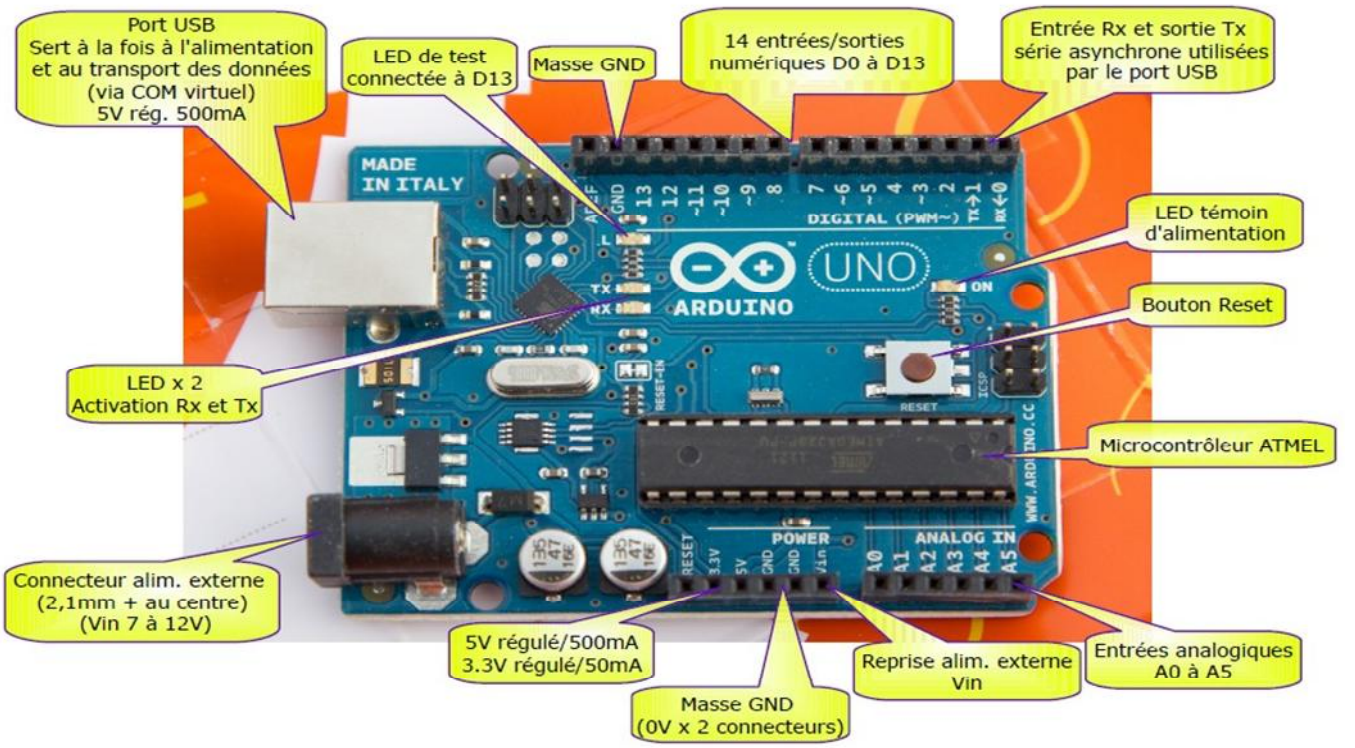


Figure I.1 La carte Arduino UNO

I.4 Pourquoi Arduino UNO :

Il y a de nombreuses cartes électroniques qui possèdent des plateformes basées sur des microcontrôleurs disponibles pour l'électronique programmée. Tous ces outils prennent en charge les détails de la programmation et les intègrent dans une présentation facile à utiliser. De la même façon, le système Arduino simplifie la façon de travailler avec les microcontrôleurs tout en offrant aux utilisateurs plusieurs avantages dont :

- **Le prix (réduits)**: les cartes Arduino sont relativement peu coûteuses comparativement aux autres plates-formes. La moins chère des versions du module Arduino peut être assemblée à la main.

- **Multi plateformes:** le logiciel Arduino, écrit en JAVA, tourne sous les systèmes d'exploitation Windows, Macintosh et Linux. La plupart des systèmes à microcontrôleurs sont limités à Windows.

- **Un environnement de programmation clair et simple:** l'environnement de programmation Arduino (le 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.

- **Logiciel Open Source et extensible:** le logiciel Arduino et le langage Arduino sont publiés sous licence open source, disponible pour être complété par des programmateurs expérimentés. Le logiciel de programmation des modules Arduino est une application JAVA multi plateformes (fonctionnant sur tout système d'exploitation), 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).

- **Matériel Open source et extensible**: les cartes Arduino sont basées sur les Microcontrôleurs Atmel ATMEGA8, ATMEGA168, ATMEGA 328, les concepteurs des circuits expérimentés peuvent réaliser leur propre version des cartes Arduino, en les complétant et améliorant. Même les utilisateurs relativement inexpérimentés peuvent fabriquer une version sur plaque d'essai de la carte Arduino, dont le but est de comprendre comment elle fonctionne pour économiser le coût.[17]

I. 5 La constitution de la carte Arduino UNO

Un module Arduino est généralement construit autour d'un microcontrôleur ATMEL AVR, et de composants complémentaires qui facilitent la programmation et l'interfaçage avec d'autres circuits. Chaque module possède au moins un régulateur linéaire 5V et un oscillateur à quartz 16 MHz (ou un résonateur céramique dans certains modèles). Le microcontrôleur est préprogrammé avec un boot loader de façon à ce qu'un programmateur

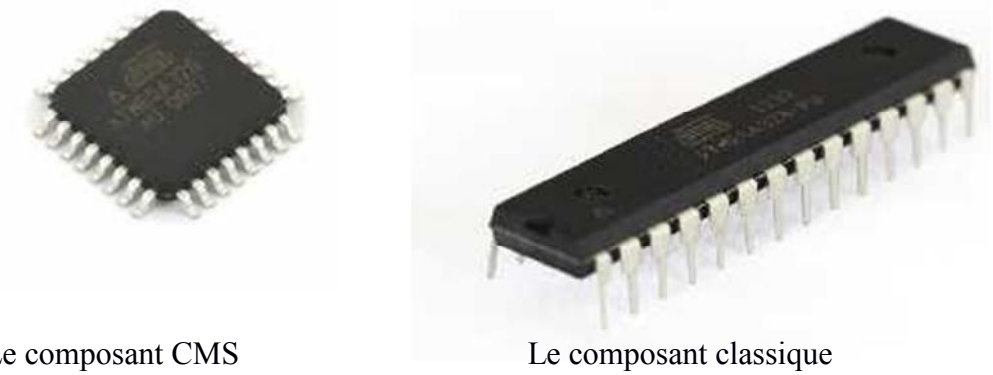
dédié ne soit pas nécessaire.

I.5.1 Partie matérielle

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

I.5.1.1 Le Microcontrôleur ATMega328

Un microcontrôleur ATMega328 est un circuit intégré qui rassemble sur une puce plusieurs éléments complexes dans un espace réduit au temps des pionniers de l'électronique. Aujourd'hui, avec l'avènement des circuits intégrés on n'a pas besoin de souder un grand nombre de composants encombrants; tels que des transistors; des résistances et des condensateurs car tout peut être logé dans un petit boîtier en plastique noir muni d'un certain nombre de broches dont la programmation peut être réalisée en langage C. La figure I.2 montre un microcontrôleur ATmega 328, qu'on trouve sur la carte Arduino.[17]



Le composant CMS Le composant classique

Figure I.2 Microcontrôleur ATMega328

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

- ***La mémoire Flash***: C'est celle qui contiendra le programme à exécuter. Cette mémoire est effaçable et réinscriptible mémoire programme de 32Ko (dont bootloader de 0.5 ko).
- ***RAM* :** c'est la mémoire dite "vive", elle va contenir les variables du programme.

Elle est dite "volatile" car elle s'efface si on coupe l'alimentation du microcontrôleur. Sa capacité est 2 ko.

- ***EEPROM*** : C'est le disque dur du microcontrôleur. On y enregistre des infos qui ont besoin de survivre dans le temps, même si la carte doit être arrêtée. Cette mémoire ne s'efface pas lorsque l'on éteint le microcontrôleur ou lorsqu'on le reprogramme. [17]

I.5.1.2 Les sources de l'alimentation de la carte :

On peut distinguer deux genres de sources d'alimentation et cela comme suit :

- ***VIN***. La tension d'entrée positive lorsque la carte Arduino est utilisée avec une source de tension externe (à distinguer du 5V de la connexion USB ou autre source 5V régulée). 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.

- 5V. La tension régulée utilisée pour faire fonctionner le microcontrôleur et les autres composants de la carte (pour info : les circuits électroniques numériques nécessitent une tension d'alimentation parfaitement stable dite "tension régulée" obtenue à l'aide d'un composant appelé un régulateur et qui est intégré à la carte Arduino). Le 5V régulé fourni par cette broche peut donc provenir soit de la tension d'alimentation VIN via le régulateur de la carte, ou bien de la connexion USB (qui fournit du 5V régulé) ou de tout autre source d'alimentation régulée.
- ***3V3***. Une alimentation de 3.3V fournie par le circuit intégré FTDI (circuit intégré faisant l'adaptation du signal entre le port USB de votre 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.

I.5.1.3 Les entrées & sorties

- **Entrées et sortie numériques :**

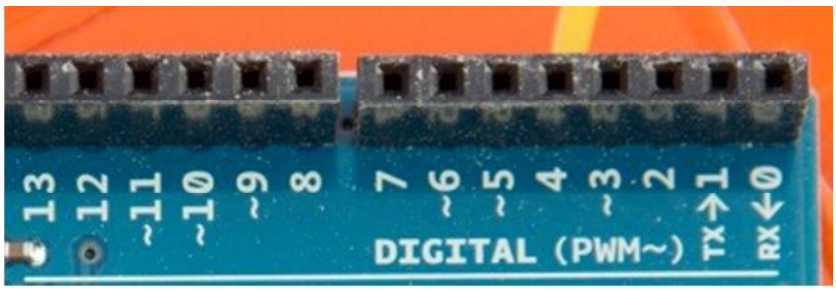


Figure I-3 Entrées et sortie numériques

Cette carte possède 14 broches numériques (numérotée de 0 à 13) peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique, en utilisant les instructions pinMode( ), digitalWrite( ) et digitalRead( ) du langage Arduino. Ces broches fonctionnent en 5V. Chaque broche peut fournir ou recevoir un maximum de 40mA d'intensité et dispose d'une résistance interne de "rappel au plus" (pull-up) (déconnectée par défaut) de 20-50 KOhms. Cette résistance interne s'active sur une broche en entrée à l'aide de l'instruction digital Write (broche, HIGH).

En plus, certaines broches ont des fonctions spécialisées :

- **Interruptions Externes**: Broches 2 et 3. Ces broches peuvent être configurées pour déclencher une interruption sur une valeur basse, sur un front montant ou descendant, ou sur un changement de valeur. -Impulsion PWM (largeur d'impulsion modulée):

Broches 3, 5, 6, 9, 10, et 11. Fournissent une impulsion PWM 8-bits à l'aide de l'instruction analog Write ( ).

- **SPI** (Interface Série Périphérique): Broches 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Ces broches supportent la communication SPI (Interface Série Périphérique) disponible avec la librairie pour communication SPI. Les broches SPI sont également connectées sur le connecteur ICSP qui est mécaniquement compatible avec les cartes Mega.

- **I2C**: Broches 4 (SDA) et 5 (SCL). Supportent les communications de protocole I2C (ou interface TWI (Two Wire Interface - Interface "2 fils"), disponible en utilisant la librairie Wire/I2C (ou TWI - Two-Wire interface - interface "2 fils").

- **LED**: Broche 13. Il y a une LED incluse dans la carte connectée à la broche 13.

Lorsque la broche est au niveau HAUT, la LED est allumée, lorsque la broche est au niveau BAS, la LED est éteinte.

➢ **Entrées et sortie Analogiques :**

La carte UNO dispose 6 entrées analogiques (numérotées de 0 à 5) ;

Figure I-4 Entrées Analogique

Chacune pouvant fournir une mesure d'une résolution de 10 bits (càd sur 1024 niveaux soit de 0 à 1023) à l'aide de la très utile fonction analogRead( ) du langage Arduino. Par défaut, ces broches mesurent entre le 0V (valeur 0) et le 5V (valeur 1023), mais il est possible de modifier la référence supérieure de la plage de mesure en utilisant la broche AREF et l'instruction analog Reference( ) du langage Arduino.

La carte Arduino UNO intègre un fusible qui protège le port USB de l'ordinateur contre les surcharges en intensité. Bien que la plupart des ordinateurs aient leur propre protection interne, le fusible de la carte fournit une couche supplémentaire de protection. Si plus de 500mA sont appliqués au port USB, le fusible de la carte coupera automatiquement la connexion jusqu'à ce que le court- circuit ou la surcharge soit stoppé.

I.5.1.4 Les ports de communications

La carte Arduino UNO a de nombreuses possibilités de communications avec l'extérieur. L'Atmega328 possède une communication série UART TTL (5V), grâce aux broches numériques 0 (RX) et 1 (TX).

On utilise (RX) pour recevoir et (TX) transmettre (les données séries de niveau

TTL). Ces broches sont connectées aux broches correspondantes du circuit intégré ATmega328 programmé en convertisseur USB vers série de la carte, un composant qui assure l'interfaçage entre les niveaux TTL et le port USB de l'ordinateur.

Comme un port de communication virtuel pour le logiciel sur l'ordinateur, La connexion série de l'Arduino est très pratique pour communiquer avec un PC, mais son inconvénient est le câble USB, pour éviter cela, il existe différentes méthodes pour utiliser ce dernier sans fil:

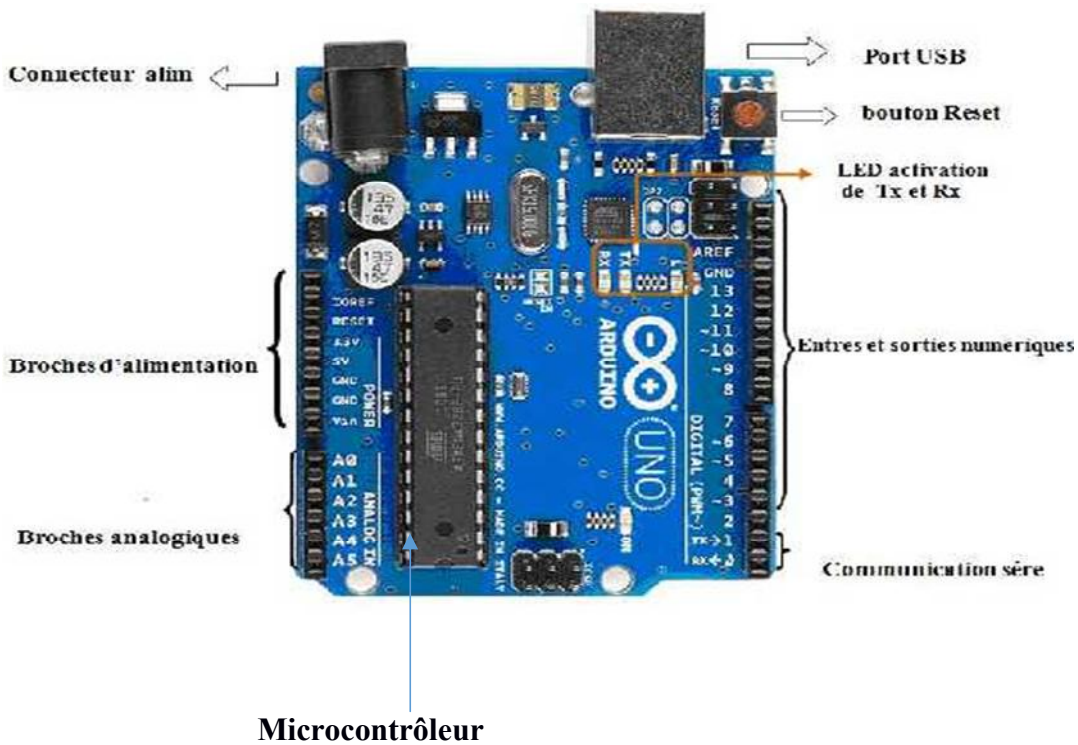


Figure I.5 Constitution de la carte Arduino UNO

I.5.2) Partie programme (Programmation Arduino) :

I.5.2.1) Présentation du logiciel de programmation (IDE Arduino) :

Un **IDE** (environnement de développement) libre et gratuit est distribué sur le site internet d'Arduino. D'autres alternatives existent pour développer pour Arduino (extensions pour CodeBlocks, Visual Studio, Eclipse, XCode, etc…).

L'interface de l'IDE Arduino offre une interface minimale et épurée pour développer un programme sur les cartes Arduino. Il est doté d'un éditeur de code avec **coloration**

syntaxique et d'une **barre d'outils rapide**. Ce sont les deux éléments les plus importants de l'interface. On retrouve aussi une barre de menus qui est utilisé pour accéder aux fonctions avancées de l'IDE. Enfin, une **console** affichant les résultats de la compilation du code source.

I.5.2.2 Langage de programmation :

Un langage de programmation est un langage permettant à un être humain d'écrire un ensemble d'instructions (code source) qui seront directement converties en langage machine grâce à un compilateur (c'est la compilation). L'exécution d'un programme Arduino s'effectue de manière séquentielle, c'est-à-dire que les instructions sont exécutées les unes à la suite des autres.

I.5.2.3 Structure d'un programme Arduino :

Le langage Arduino est inspiré de plusieurs langages. On retrouve notamment des similarités avec le C, le C++, le Java et le Processing. Le langage impose une structure particulière typique de l'informatique embarquée. Un programme Arduino doit se structurer en 4 parties :

- la partie Commentaires ;
- la partie déclaration des variables ;
- la partie initialisation et configuration des entrées/sorties ;
- La partie Programmation des interactions.

Le code du syntaxe est structuré par une ponctuation stricte :

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

I.5.2-3-1 Les commentaires :

La première partie du programme, doit impérativement débuter par des commentaires permettant de définir ce que doit faire le programme. Ceci permet de savoir en quoi consiste son fonctionnement, pourquoi pas sa date de création, sa version …etc.

Mais les commentaires ne se limitent pas à la première partie. Les commentaires doit agrémenter le programme afin de le faire comprendre par une personne tiers, essayant de le lire.

- Une ligne qui commence par "//" est considérée comme un commentaire.
- Un paragraphe qui commence par "/*" et qui se termine par "*/" est considéré comme un commentaire

I.5.2-3-2 les variables :

Une variable est un espace de stockage nommé qui permet de stocker une valeur utilisable par la suite dans la boucle d'un programme. Une variable peut aussi bien représenter des données lues ou envoyées sur un des ports analogiques ou numériques, une étape de calcul pour associer ou traiter des données, que le numéro 'physique' de ces entrées ou sorties sur la carte. Une "variable" n'est donc pas exclusivement un paramètre variant dans le programme.

La nature de la variable définit les limites de celle-ci, ainsi que la place nécessaire dans la mémoire de la carte Arduino. Ceci a de l'importance lorsque vous créez de gros programme impliquant beaucoup de variables qui prennent de la place en mémoire. Cette définition permet de limiter la place mémoire. De plus, il faut savoir que plus la variable prend de place, plus les opérations de calcul seront long et inversement.

Il est donc nécessaire de définir les variables au plus juste en fonction de ce que le programme doit faire ; parmi les plus utilisées sont :

- **byte** : nombre de 0 à 255.
- **char** : définit un caractère alphanumérique.
- **int** : entier décimal allant de -32 768 à 32 767. Va de paire avec **unsigned int** qui lui ira de 0 à 65 535 (il augmente les positifs et élimine les négatifs).
- **string** : suite de caractères alphanumérique. Pas de limite, sauf celle de la mémoire.
- **long** : entier décimal allant de -2 147 483 648 à 2 147 483 648. Va de paire avec
- **unsigned long** qui lui ira de 0 à 4 294 967 295 (il augmente les positifs et élimine les négatifs).
- **array** : définition d'un tableau de variable.

I.5.2-3-3 Configuration des entrées/sorties.

La fonction setup permet dans la plus part du temps de configurer les différentes broches du microcontrôleur.

Dans cette partie on défini la nature des entrées/sorties matérielles utilisées (si telle ou telle broche est utilisé en entrée ou en sortie, de nature numérique (binaire : 0 ou 1) ou

analogique. Cette phase doit être précédée par la commande "void setup() " suivie d'une accolade "{" et se terminer d'une accolade "}".

Dans cette configuration on doit définir comment doivent se comporter les différentes broches de l'ARDUINO. Comme aussi définir les broches numérique, utilisant que du binaire, analogique servant soit aux convertisseurs ou à la PWM, ainsi qu'aux broches de communications.

I.5.2-3-3-1 Numérique :

pinMode (Broche, état) ;

La première variable à mettre est le numéro de broche que l'on essai de configurer

La deuxième variable est l'état de configuration, soit " INPUT " (entrée, il s'agit de broche recevant des informations), soit " OUTPUT " (sortie, il s'agit de broche envoyant des informations).

I.5.2-3-3-2 Analogique.

Pour la configuration de broche analogique, seul une sortie analogique doit être configurée. On doit simplement ajouter pinMode (numéro de broche, OUTPUT).

Pour les entrées analogiques, pas besoin de les définir, elles se mettront automatiquement en entrée lorsqu'on les sollicitera.

I.5.2-3-3-3 Communication :

Il existe plusieurs moyens pour faire communiquer des systèmes électroniques entre eux et la carte arduino. Certaine broches pourront être utilisé pour des type de communication, tel que liaison RS232, I2C ou autre. Ces broches ont des configurations particulières.

- Serial.begin(X): définit la vitesse X (en bits par seconde) de transfert des données. Cette fonction doit apparaitre dans le setup du programme. Il existe plusieurs vitesse de transmission des données : 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 250000
- serial.print(X) : permet d'envoyer l'information X.
- serial.println(X) : même action que print mais avec un retour à la ligne à la fin du message

I.5.2-3-4 Programme principal.

void loop( ) : cette instruction définit le début du programme principal, celui qui tournera en permanence sur la carte. La fonction loop permet justement de faire tourner en boucle le programme.

Sans oublier biensur les accolades :

- en début de programme : " {"
- en fin de programme : " }"

Ces deux balises définissent le début et la fin du programme.

I.5.2-3-4-1 Lecture et écriture numérique et analogique :

- digitalRead (Broche) : cette instruction permet de lire la valeur d'une entrée. Comme cette valeur, pour être utilisable, doit être sauvegardée, nous allons utiliser une équation qui devra ressembler à ça : variable = digitalRead (n° de broche)
- Ainsi dans la variable sera enregistrée la valeur présente au n° de broche.
- digitalWrite(Broche, état) : cette instruction permet d'écrire sur une broche la valeur de l'état. Cet état sera soit HIGH (niveau haut : 1), soit LOW (niveau bas : 0).
- Par exemple :

digitalWrite (15,HIGH); // la broche n°15 sera mise au niveau logique haut (1)

- analogRead(Broche) : cette instruction permet de lire une valeur analogique sur une des broches dédiées. Tout comme digitalRead, cette valeur doit être sauvegardée, il va donc falloir utiliser une équation de ce style :

variable = analogRead(0); //sauvegarder la valeur du convertisseur lié à la broche A0 (entrée analogique 0) dans la variable

Chaque conversion se fera par rapport à une tension par défaut de 5V maxi. La valeur convertit sera une image de la tension qui va évoluer entre 0 et 5V, par tranche de 4,88 mV.

Par exemple si la tension sur A0 est de 2,5 V, la valeur obtenu après l'instruction analogRead sera : 2,5/0,00488 = 512

- analogWrite(Broche,valeur) : cette instruction permet de créer une PWM :

Cette fonction appelée PWM (Pulse Width Modulation, Modulation à Largeur d'Impulsion), permet de générer un signal rectangulaire périodique de fréquence fixe (490 Hz, soit une période d'environ 2ms) mais de rapport cyclique variable (le temps au niveau haut

évolue sans changer la période). Le rapport cyclique est réglé par la "valeur". Celle-ci peut être entre 0 et 255. Cette valeur divisé par 255, donnera le pourcentage de temps que prendra le niveau haut par rapport au niveau bas.

Exemple : analogWrite(5,128); // Un signal de 490 Hz sera présent sur la broche 5 de l'ARDUINO avec un rapport cyclique de 128/255 = 50%. Soit un temps identique entre l'état haut et l'état bas.

I.5.2-3-4-2 La manipulation des ports à l'aide des registres de port :

I.5.2-3-4-2-1 Description :

Les registres de port permettent une manipulation de bas-niveau (c.-à-d. au niveau du matériel lui-même) et plus rapide des broches d'entrée/sortie du microcontrôleur de la carte Arduino. Les microcontrôleurs utilisés pour les cartes Arduino (l'ATmega8 et l'ATmega 168, 328) ont trois ports :

- Port B (broches numériques de 8 to 13)
- Port C (broches analogiques/numériques)
- Port D (broches numériques 0 to 7).

Chaque port est contrôlé par 3 registres, qui sont également défini en tant que variable dans le langage Arduino.

I.5.2-3-4-2-2 But de la manipulation de port :

Il es important de savoir les avantages de manipulation de port :

- Etre capable de mettre un niveau HAUT ou BAS sur une broche très rapidement, en une fractions de microsecondes. Alors que les instructions digitalRead() et digitalWrite prennent chacune environ une douzaine de ligne code, lesquelles seront compilées en quelques instructions machine. Chaque instruction machine demande 1 cycle d'horloge à 16 Mhz, ce qui peut être ajouté dans les applications sensibles au temps. L'accès direct au port permet de réaliser la même chose en quelques cycles d'horloge seulement.
- Parfois, on peut avoir besoin de mettre au même niveau plusieurs broches en sortie exactement au même moment. En appelant l'instruction digitalWrite(10,HIGH); suivie par une autre instruction digitalWrite(11,HIGH]]; entraînera une mise au niveau HAUT de la broche 10 plusieurs microsecondes avant la broche 11, ce qui pourrait perturber certains circuits externes sensibles au temps qui sont connectés. Par contre,

on peut mettre au même niveau exactement au même moment les broches en utilisant PORTB |= B1100;

- Si nous somme limité en programme mémoire, on peut utiliser cette méthode pour rendre votre code plus petit. Cela requiert un peu moins d'octets de code compilé d'écrire simultanément sur plusieurs broches via le registre de port que d'utiliser une boucle for pour chaque broche séparément. Dans certains cas, ceci fera la différence pour réussir à programmer ou non le programme dans la mémoire flash du microcontrôleur.

I.5.2-3-4-3 Gestion du temps:

Il est souvent nécessaire d'effectuer des pose dans le temps, de temporiser des programmes. Ceci est souvent utilisé dans la gestion de signaux numérique. La procédure avec l'ARDUINO est assez simple, il existe deux instructions permettant d'effectuer des pause:

- delay (ms): cette instruction permet d'effectuer des pauses d'une durée en milli seconde inscrite entre parenthèses.

  Exemple : delay(250); //une pause de 250 ms est effectuée.

- delayMicroseconds(µs) : cette instruction permet d'effectuer des pauses d'une durée en micro seconde inscrite entre parenthèses.

  Exemple : delay(150); //une pause de 150 µs est effectuée.

I.5.2-3-4-3-4 Les structures de contrôle :

Les structures de contrôle sont des blocs d'instructions qui s'exécutent en fonction du respect d'un certain nombre de conditions. Il existe quatre types de structure :

- if...else : exécute un code si certaines conditions sont remplies et éventuellement exécutera un autre code avec sinon ;
- while : exécute un code tant que certaines conditions sont remplies ;
- for : exécute un code pour un certain nombre de fois ;
- switch/case : fait un choix entre plusieurs codes parmi une liste de possibilités.

Une telle carte d'acquisition qui se base sur sa construction sur un microcontrôleur doit être dotée d'une interface de programmation comme est le cas de notre carte. L'environnement de programmation open-source pour Arduino peut être téléchargé gratuitement (pour Mac OS X, Windows, et Linux).

I.6 Les Accessoires de la carte Arduino :

La carte Arduino généralement est associée aux accessoires qui simplifient les réalisations.

I.6.1 Communication :

Le constructeur a suggéré qu'une telle carte doit être dotée de plusieurs ports de communications ; on peut éclaircir actuellement quelques types.

I.6.1.1 Le module Arduino Bluetooth :

Le Module Microcontrôleur Arduino Bluetooth est la plateforme populaire Arduino avec une connexion sérielle Bluetooth à la place d'une connexion USB, très faible consommation d'énergie, très faible portée (sur un rayon de l'ordre d'une dizaine de mètres), faible débit, très bon marché et peu encombrant.

Figure I.6 Type de modules Bluetooth

I.6.1.2 Le module shield Arduino Wifi :

Le module Shield Arduino Wifi permet de connecter une carte Arduino à un réseau internet sans fil Wifi.

Figure I.7 Module shield wifi

I.6.1.3 Le Module XBee

Ce module permet de faire de la transmission sans fil, faible distance /consommation /débit/ prix. [6]

Figure I.8 Module XBee

I.6.2 Les capteurs

Un capteur est une interface entre un processus physique et une information manipulable. Il ne mesure rien, mais fournit une information en fonction de la sollicitation à laquelle il est soumis. Il fournit cette information grâce à une électronique à laquelle il est associé. [3]

Figure I.9 Capteur Arduino

I.6.3 Les drivers

Il existe plusieurs drivers comme des cartes auxiliaires qui peuvent être attachées avec l'Arduino afin de faciliter la commande ; on peut citer quelques types[8].

1 Des moteurs électriques

Figure I.10 Moteurs électriques

2 Les afficheurs LCD

Les afficheurs LCD sont devenus indispensables dans les systèmes techniques qui nécessitent l'affichage des paramètres de fonctionnement.

Ces Afficheurs permettent d'afficher des lettres, des chiffres et quelques caractères spéciaux. Les caractères sont prédéfinis. [3]

Figure I.11 Afficheurs LCD

3 Le relais

C'est un composant qui possède une bobine (électro-aimant) qui est parcourue par un courant électrique agissant sur un ou plusieurs contacts. Le relais est une solution à la commande en puissance. Il assure en outre une isolation galvanique en mettant en oeuvre un mouvement mécanique. [8]

Figure I.12 Relais

I.7 -Conclusion :

Dans ce chapitre, nous avons projeté la lumière sur une carte d'acquisition qui est l'Arduino donnant ainsi les raisons pour lesquelles on l'a choisie, puis nous avons cité des différents types de cette dernière. Ensuite, nous avons expliqué les deux parties essentielles de l'Arduino; (la partie matérielle et la partie de programmation) plus précisément. Nous avons également expliqué le principe de fonctionnement de la carte Arduino sans oublier ses caractéristiques.

Chapitre II

Commande numerique et notions de régulation de processus

I Introduction :

Les performances des systèmes industriels passe par la maîtrise et la mise en œuvre de l'ensemble des moyens théoriques et pratiques pour maintenir les grandeurs physiques influentes à des valeurs désirées, appelées consignes, par action des grandeurs réglantes, et ce malgré l'influence des grandeurs perturbatrices.

Dans ce chapitre nous allons faire une présentation sur le principe de la régulation ensuite donner des généralités sur la commande numérique, et enfin une présentation du régulateur PID (Proportionnel Intégrale Dérivé).

II Commande en boucle ouverte et fermée :

Un système de commande permet d'exercer des actions entrainant une amélioration du comportement du système et de ses performances. L'ensemble des méthodes permettant l'analyse du comportement d'un système donné et la synthèse d'un système de commande satisfaisant des spécifications de performance. Alors, l'objectif principal d'une commande est de maîtriser l'évolution d'une ou plusieurs grandeurs physiques (température, pression, vitesse, PH ...) à partir d'une ou plusieurs variables de contrôle et ceci dans un environnement perturbé.

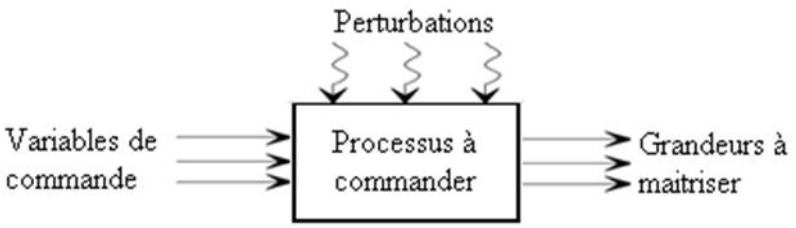


Figure II.1 Système de commande

Quelle que soit la nature du système `a commander, il est toujours possible de classer les différentes structures de commande en deux grandes familles. Les structures de commande en boucle ouverte et les structures de commande `a contre - réaction appelées également structures de commande en boucle fermée.

II-1 Commande en boucle ouverte :

En l'absence d'entrées perturbatrices et en supposant que le modèle mathématique du système est parfait, il est imaginable de générer un signal de commande produisant le signal de sortie souhaite. Cela constitue le principe de la commande en boucle ouverte.

Les signaux d'entrée e(t) ne sont pas influences par la connaissance des signaux de sortie s(t).

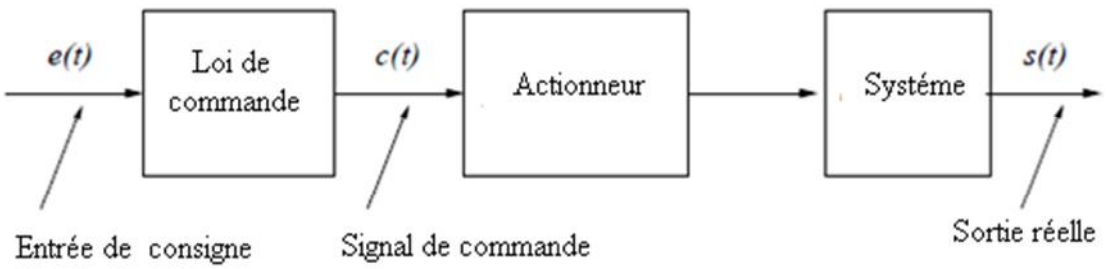


Figure II.2 Commande en boucle ouverte

II- 2 Commande en boucle fermée :

L'introduction d'un retour d'information sur les sorties mesurées s'avère alors nécessaire.

Le principe de commande en boucle fermée est illustré sur la figure suivante et définit la structure de commande `a contre réaction (feedback en anglais). On parle alors de système bouclé

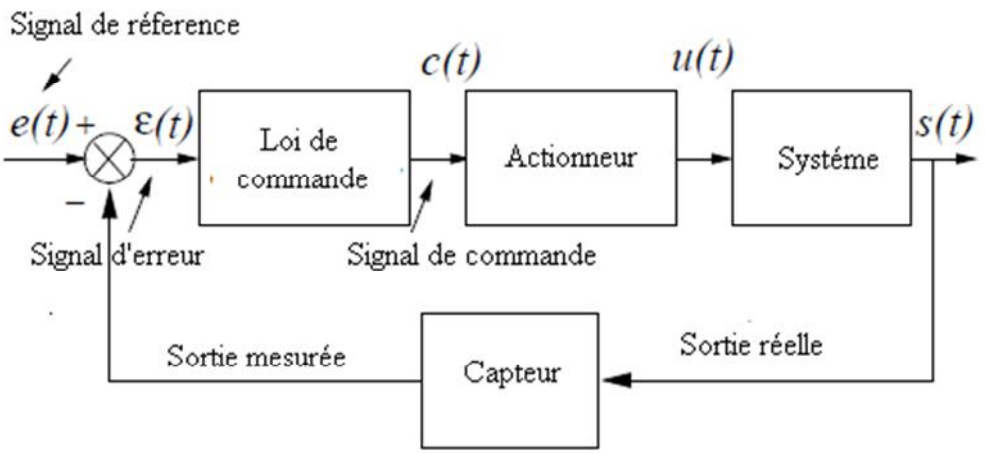


Figure II.3 Commande en boucle fermée

Un système bouclé vérifie en quelque sorte que la réponse du système correspond à l'entrée de référence tandis qu'un système en boucle ouverte commande sans contrôler l'effet de son action. Les systèmes de commande en boucle fermée sont ainsi préférables quand des perturbations non modélisables et/ou des variations imprévisibles des paramètres sont présentes. Cette structure de commande permet ainsi d'améliorer les performances dynamiques du système commandé (rapidité, rejet de perturbation, meilleur suivi de consignes, moindre sensibilité aux variations paramétriques du modèle, stabilisation de systèmes instables en boucle ouverte).

Il est toutefois important de remarquer que cette structure de commande ne présente pas que des avantages. Elle nécessite l'emploi de capteurs qui augmentent le cout d'une installation. D'autre part, le problème de la stabilité et de la précision des systèmes `a contre-réaction se pose de manière plus complexe.

III Commande numérique :

III-1 Structure technologique :

La commande des systèmes était dans le passé l'apanage des techniques analogiques de traitement du signal. Maintenant la vulgarisation de l'électronique numérique fait que la quasi-totalité des commandes sont faites à l'aide de DSP, micro contrôleurs, calculateurs... Ce saut technologique n'est pas sans incidences conceptuelles : En effet, le traitement numérique de l'information à des instants discrétisés implique l'utilisation d'outils tels, la transformation en z ou les variables d'état discrètes.

Cependant, les approches continues, transformation de Laplace, variables d'état continus, gardent leurs lettres de noblesses car elles sont tout d'abord le fondement des techniques discrètes et dans un contexte numérique elles offrent souvent des moyens d'analyses pertinents.

La commande par une unité de calcul, d'un processus à l'allure suivante:

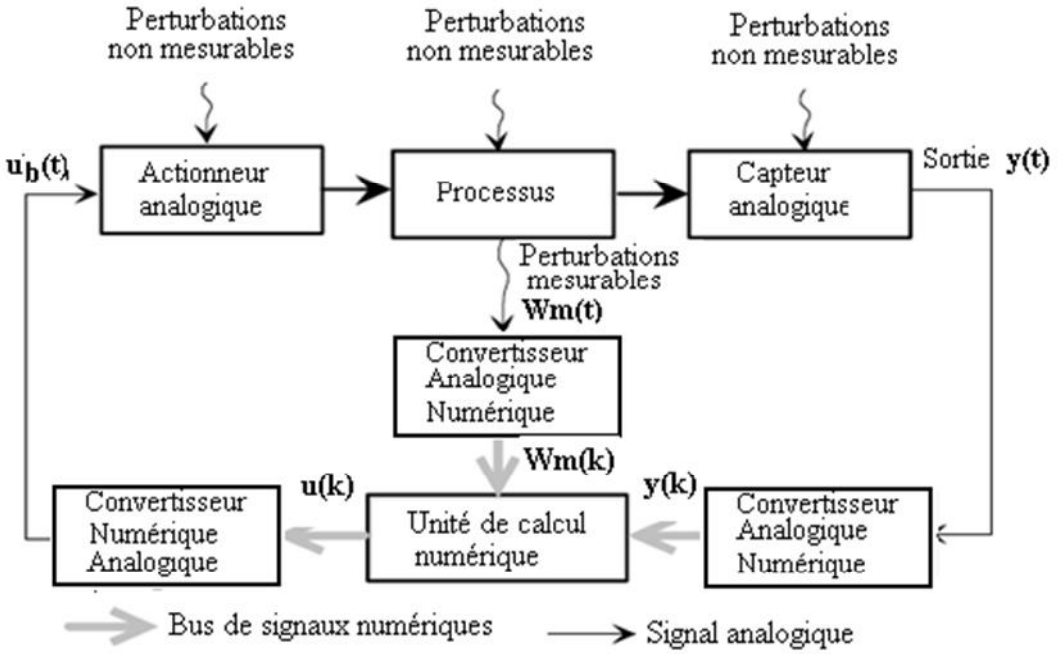


Figure II.4 Structure de la commande

Dans ce schéma technologique il apparaît des signaux de nature différente. La sortie du processus est un signal continu qui est ensuite converti sous forme numérique et se retrouve discrétisé et représenté par une suite de nombres.

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.

Intéressons nous maintenant au procédé. Il est illusoire de vouloir décrire par un seul type de schéma bloc la diversité des systèmes qu'on peut rencontrer. Cependant, nous allons en définir un qui met en évidence les principales grandeurs altérant la sortie :

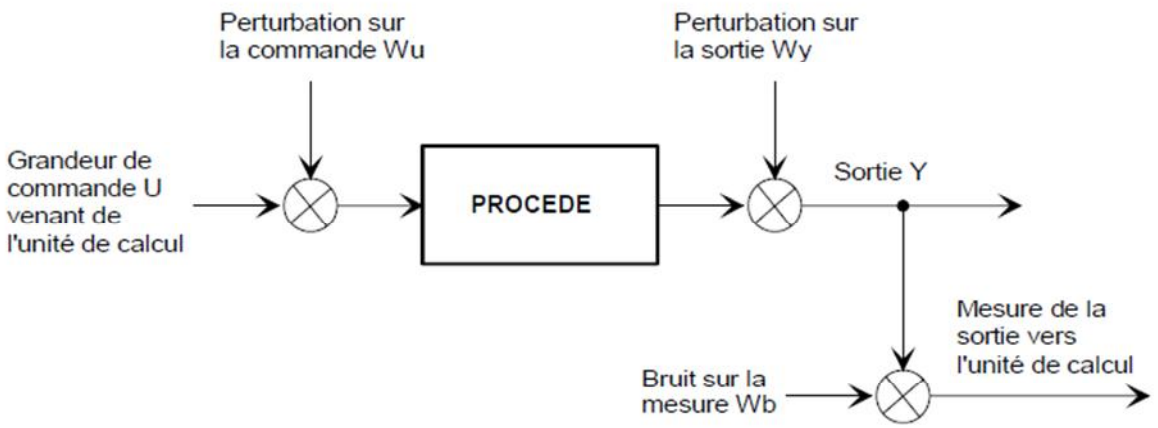


Figure II.5 principales grandeurs altérant la sortie

Wu	bruit sur l'entrée	Wy bruit sur la sortie	Wb bruit sur la mesure
Y	sortie de procédé	U	grandeur de commande

A partir de cette analyse, il sera possible de compléter l'approche en ajoutant des transferts sur les perturbations, de dissocier celles inconnues de celles qui sont mesurables, etc. Par soucis de concision nous restreindrons notre étude au schéma fonctionnel de la (figure II.5).

Pour bien contrôler un système, il faudra s'attacher à faire une analyse fonctionnelle de l'ensemble et d'identifier les principaux transferts reliant la commande et les perturbations à la grandeur à contrôler.

Le plus souvent le processus est non linéaire ou considéré comme linéaire autour d'un point de fonctionnement. Pour le caractériser, il sera indispensable d'utiliser les méthodes modernes d'identification pour représenter le comportement dynamique du système à contrôler.

Il es important de contrôler au mieux la sortie sachant que le modèle du processus dont il dispose est imprécis et peut évoluer dans le temps. En outre la mesure de la grandeur à contrôler est bruitée et la commande peut l'être aussi.

Il va donc s'agir de définir un correcteur ou un algorithme de commande, ayant des performances satisfaisantes, et s'affranchissant des méconnaissances du procédé, des imprécisions de la mesure, et des limitations de la commande.

III -2 Structure d'un asservissement numérique :

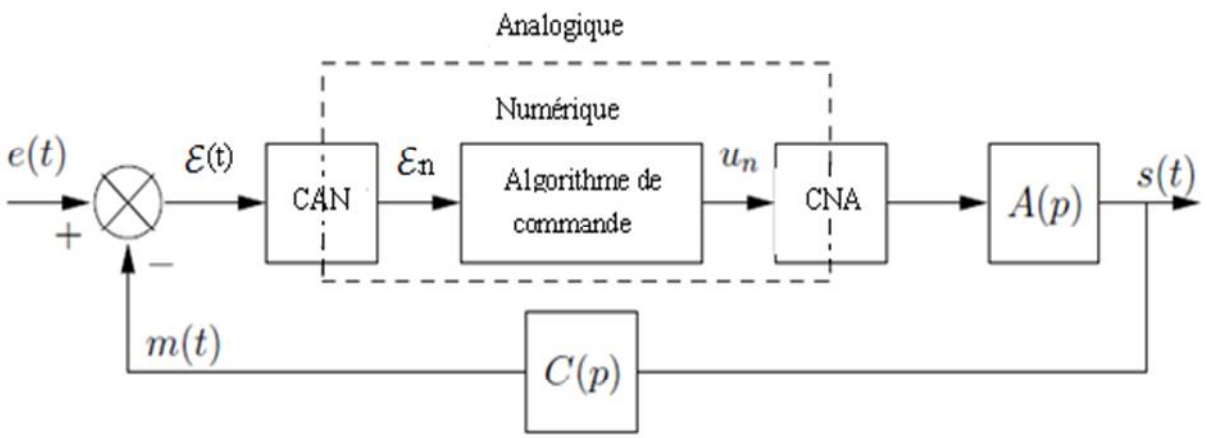


Figure II.6 Structure typique de la réalisation d'un asservissement numérique.

L'asservissement numérique se fait typiquement par le biais d'une structure schématisée par la figure II.6 et composée des objets fondamentaux suivants :

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

- un CAN : celui-ci fonctionne à la période d'échantillonnage T >0. Il fournit à sa sortie le signal numérique d'écart noté $\mathcal{E}n$.

- 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 Un.
- 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.

- des transmittances A(p) et C(p) représentant respectivement la dynamique du système et celle du capteur.

En pratique, l'opération de comparaison se fait également numériquement. Ainsi, une autre structure typique d'asservissement peut être schématisée par la figure II.7 où nous pouvons remarquer la présence d'un CAN supplémentaire.

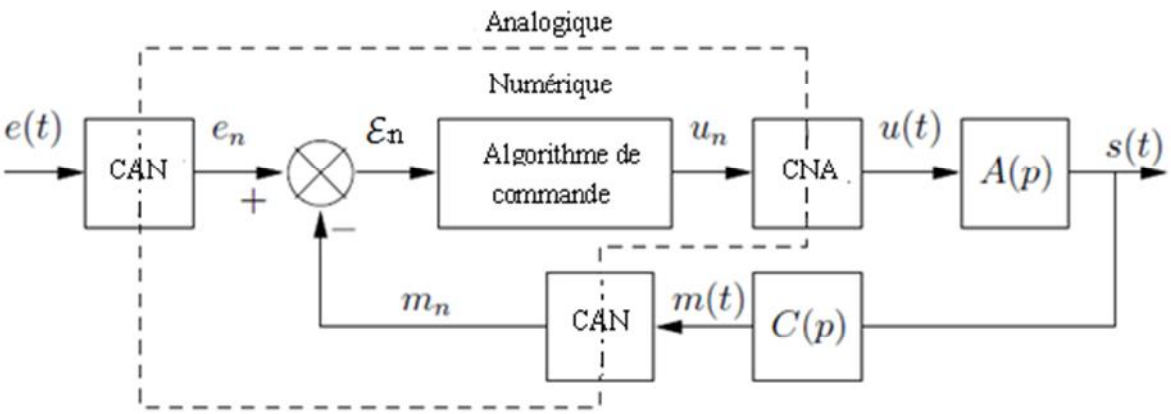


Figure II.7 – Autre structure typique réalisant un asservissement numérique.

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.

IV Principe de régulation PID :

Le régulateur standard le plus utilisé dans l'industrie est le régulateur PID (proportionnel intégral dérivé), car il permet de régler à l'aide de ses trois paramètres les performances (amortissement, temps de réponse,...) d'un processus modélisé par un deuxième ordre. Nombreux sont les systèmes physiques qui, même en étant complexes, ont un comportement voisin de celui d'un deuxième ordre. Par conséquent, le régulateur PID est bien adapté à la plupart des processus de type industriel et est relativement robuste par rapport aux variations des paramètres du procédé, quand on n'est pas trop exigeant sur les performances de la boucle fermée par rapport à celles de la boucle ouverte (par exemple, accélération très importante de la réponse ou augmentation très importante de l'amortissement en boucle fermée).

Si la dynamique dominante du système est supérieure à un deuxième ordre, ou si le système contient un retard important ou plusieurs modes oscillants, le régulateur PID n'est plus adéquat et un régulateur plus complexe (avec plus de paramètres) doit être utilisé, au dépend de la sensibilité aux variations des paramètres du procédé.

IV -1 Les actions PID :

En pratique, à une catégorie donnée de systèmes à asservir correspond un type de correcteur adopté. Pour effectuer un choix judicieux, il faut connaître les effets des différentes actions : proportionnelle, intégrale et dérivée.

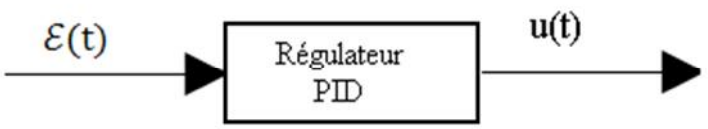


Figure II.8: Schéma synoptique d'un régulateur PID

Un régulateur PID est obtenu par l'association de ces trois actions et il remplit essentiellement les trois fonctions suivantes :

- Il fournit un signal de commande en tenant compte de l'évolution du signal de sortie par rapport à la consigne
- Il élimine l'erreur statique grâce au terme intégrateur
- Il anticipe les variations de la sortie grâce au terme dérivateur.
- La commande $U(t)$ donnée par le régulateur PID, dans sa forme classique est décrite par:

$$U(t) = Kp\left|\varepsilon(t) + \frac{1}{Ti}\int_0^t \varepsilon(\tau)d\tau + Td\frac{d\varepsilon(t)}{dt}\right| \quad (1)$$

Elle est la somme de trois termes :

— le terme proportionnel :

$$P = Kp\varepsilon(t) \quad (2)$$

— le terme intégral :

$$I = Kp\frac{1}{Ti}\int_0^t \varepsilon(\tau)d\tau \quad (3)$$

— le terme dérivatif :

$$D = Kp Td\frac{d\varepsilon(t)}{dt} \quad (4)$$

Les paramètres du régulateur PID sont le gain proportionnel K_p, le temps intégral T_i et le temps dérivatif T_d , les temps étant exprimés en secondes.

IV -2 Les caractéristiques régulateur PID :

IV -2 -1 L'action proportionnelle[7] :

La sortie U (t) du régulateur proportionnel est donnée en fonction de son entrée ε(t) qui représente l'écart entre la consigne et la mesure par la relation :

$$u(t) = Kp.\varepsilon(t) \quad (5)$$

Pour le cas discret, cette relation reste la même telle que :

$$u(k) = Kp.\varepsilon(k) \quad (6)$$

Le rôle de l'action proportionnelle est de minimiser l'écart ε entre la consigne et la mesure et elle réduit le temps de monter et le temps de réponse. On constate qu'une augmentant du gain K_p du régulateur entraîne une diminution de l'erreur statique et permet d'accélérer le comportement global de la boucle fermée. On serait tenté de prendre des valeurs de gain élevées pour accélérer la réponse du procédé mais on est limité par la

stabilité de la boucle fermée. En effet, une valeur trop élevée du gain augmente l'instabilité du système et donne lieu à des oscillations.

IV -2- 2 L'action dérivée :

Elle est une action qui tient compte de la vitesse de variation de l'écart entre la consigne et la mesure, elle joue aussi un rôle stabilisateur, contrairement à l'action intégrale. En effet, elle délivre une sortie variant proportionnellement à la vitesse de variation de l'écart ε :

$$u(t) = Td\frac{d\varepsilon(t)}{dt} \tag{7}$$

Avec T_d le dosage de l'action dérivée, exprimé en minutes ou en secondes.

L'action dérivée va ainsi intervenir uniquement sur la variation de l'erreur ce qui augmente la rapidité du système (diminution des temps de réponses). L'action dérivée permet aussi d'augmenter la stabilité du système par apport de phase ($+\pi/2$ qui augmente la marge de phase). L'annulation de cette action en régime statique impose donc de ne jamais l'utiliser seule : l'action dérivée n'exerce qu'un complément à l'action proportionnelle.

En pratique, il est souhaitable de limiter l'action dérivée afin de ne pas amplifier les bruits haute fréquence et de limiter l'amplitude des impulsions dues aux discontinuités de l'écart. Lorsque la période d'échantillonnage Te est petite, la différence vers l'arrière (Approximation d'Euler rétrograde) nous permet d'approcher la dérivée d'un signal à temps continu par :

$$\frac{df(t)}{dt} = \frac{f(t)-f(t-T_e)}{T_e} \tag{8}$$

L'opération de dérivation se traduisait par une multiplication par la variable de Laplace P en continu. Dans le cas discret, en appliquant la transformée en Z on obtient :

$$Z\left\{\frac{f(t)-f(t-T_e)}{T_e}\right\} = \frac{1-Z^{-1}}{T_e}F(z) \tag{9}$$

Ceci conduit à établir l'équivalence linéaire entre la variable de Laplace P et la variable *z* :

$$p \leftarrow\rightarrow \frac{1}{T_e}\cdot(1 - z^{-1}) \tag{10}$$

Donc en discret, le terme dérivé peut être remplacé par :

$$u(k) = \frac{T_d}{T_e}.\left(\varepsilon(k) - \varepsilon(k-1)\right) \quad (11)$$

IV -2 -3 L'action intégrale[7] :

L'action intégrale agit proportionnellement à la surface de l'écart entre la consigne et la mesure, et elle poursuit son action tant que cet écart n'est pas nul. On dit que l'action intégrale donne la précision statique (Elle annule l'erreur statique). L'action intégrale est conditionnée par le temps d'intégrale Ti.

$$u(t) = \frac{1}{T_i}\int_0^t \varepsilon(t).d\tau \quad (12)$$

Comme dans le cas de l'action proportionnelle, un dosage trop important de l'action intégrale engendre une instabilité de la boucle de régulation. Pour son réglage, il faut là aussi trouver un compromis entre la stabilité et la rapidité.

L'ajout du terme intégral permet d'améliorer la précision mais en contrepartie, il

Introduit malheureusement un déphasage de$-\pi/2$ ce qui risque de rendre le système instable du fait de la diminution de la marge de phase.

Enfin, le correcteur intégral présente le défaut de saturer facilement si l'écart ne s'annule pas rapidement ce qui est le cas des systèmes lents. En effet, tout actionneur est limité : un moteur est limité en vitesse, une vanne ne peut pas être plus que totalement ouverte ou totalement fermée. Il se peut que la variable de commande amène l'actionneur à sa limite ce qui suspend la boucle de retour et le système aura une configuration assimilable à une boucle ouverte puisque l'actionneur demeurera saturé indépendamment de la sortie du système.

Quand l'erreur est réduite (action intégrale non saturée), il se peut qu'il faille un temps important pour que les valeurs des variables ne soient correctes de nouveau: on appelle ce phénomène l'emballement du terme intégral.

Pour l'éviter, on peut :

- soit suspendre l'action intégrale quand la commande est saturée ;

- soit appliquer une méthode d'anti-saturation, qui consiste à recalculer le terme intégral pour ne pas saturer.

Pour le cas discret, le terme intégral peut être remplacé par la somme des écarts et la différentielle *dt* par *Te* ce qui nous donne le résultat suivant

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

Ti : Constante de temps intégrale et s'exprime en (s).

Le problème est que lorsque $^{\varepsilon}$ redevient nul, le signal U peut avoir atteint une valeur trop élevée pour qu'il y ait équilibre. Autrement, elle permet d'éliminer l'erreur résiduelle en régime permanent.

L'action intégrale est utilisée lorsqu'on désire avoir en régime permanent, une précision parfaite, en outre, elle permet de filtrer la variable à régler d'ou l'utilité pour le réglage des variables bruités telles que la commande.

IV -3 Le régulateur PI :

Le correcteur intégral est en général associé au correcteur proportionnel, il élabore alors une commande qui peut être donnée par la relation suivante :

$$u(t) = K_p\left(\varepsilon(t) + \frac{1}{T_i}\int_0^t \varepsilon(\tau)d\tau\right) \quad (14)$$

La fonction de transfert du correcteur est alors donnée par :

$$C(p) = K_p\frac{1+T_ip}{T_ip} \quad (15)$$

Pour un régulateur intégral pur, le régime dynamique est relativement long. D'un autre côté, le régulateur proportionnel réagit immédiatement aux écarts de réglage mais il n'est pas en mesure de supprimer totalement l'erreur statique. La combinaison des actions proportionnelle et intégrale permet d'associer l'avantage du régulateur P, c'est-à-dire la réaction rapide à un écart de réglage, à l'avantage du régulateur I qui est la compensation exacte de la grandeur pilote.

La transposition de correcteurs continus consiste à discrétiser un correcteur continu afin de l'utiliser dans une commande numérique.

En utilisant cette équivalence dans l'équation, on obtient le correcteur PI discret :

$$C(z) = \frac{U(z)}{\varepsilon(z)} = \frac{(K_p + K_i) - K_p . Z^{-1}}{1 - Z^{-1}} \quad (16)$$

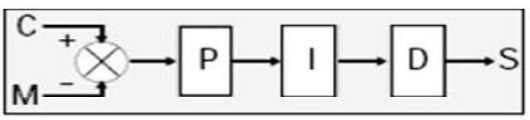
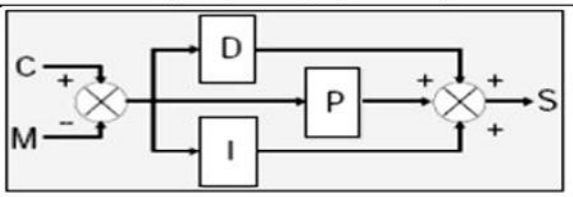
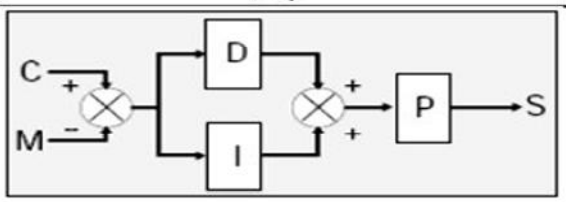
D'où on obtient l'algorithme de commande du régulateur PI :

$$u(k) = u(k-1) + (K_p + K_i).\varepsilon(k) - K_p.\varepsilon(k-1) \quad (17)$$

IV -4 Le régulateur PID [10] :

L'action conjuguée PID permet une régulation optimale en associant les avantages de chaque action : la composante P réagit à l'apparition d'un écart de réglage, la composante D s'oppose aux variations de la grandeur réglée et stabilise la boucle de régulation et la composante I élimine l'erreur statique. Et c'est pour cela que ce type de correcteur est le plus utilisé en milieu industriel

Dans un régulateur PID, il existe plusieurs façons d'associer les paramètres P, I et D. en effet, le correcteur PID peut avoir une structure série, parallèle ou mixte [7] :

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

La discrétisation du correcteur PID parallèle par l'approximation d'Euler rétrograde nous donne :

$$C(z) = \frac{U(z)}{\varepsilon(z)} = K_p + \frac{T_e}{T_i}.\frac{1}{1-Z^{-1}} + \frac{T_d}{T_e}.(1 - z^{-1}) \quad (18)$$

Ce qui nous donne l'algorithme de commande suivant :

$$u(k) = u(k-1) + (K_p + K_i + K_d).\varepsilon(k) - (K_p + 2K_d).\varepsilon(k-1) + K_d.(k-2) \quad (19)$$

V Conclusion :

Dans ce chapitre, nous avons présenté les différentes méthodes de régulations en boucle ouverte et en boucle fermée, une description générale des systèmes industriels ainsi que le principe de la régulation, qui se résument dans l'exploitation des régulateurs PID qui sera l'objet du chapitre suivant.

Chapitre III
Conception et réalisation de la commande de systeme thermique

I-Introduction :

Dans ce chapitre on va introduire la commande du système thermique à base de la carte Arduino. En premier lieu nous allons présenter les composants utilisés, ensuite on présentera la méthode adoptée pour l'interfaçage entre la carte arduino et le module Simulink de Matlab. Enfin, nous allons procéder à la modélisation du procédé thermique et le commander par un régulateur numérique PID.

II-Présentation de la maquette :

La maquette est constituée d'un capteur de température LM35 et une Lampe halogène 12V/21W qui joue le rôle d'un élément chauffant. Le capteur et la lampe sont installés dans une boite en bois avec un couvercle en plexiglass qui représente le système thermique à commander. La figure III.1 ci-dessous illustre les connexions entre la carte Arduino UNO et l'entrée/sortie du système thermique.

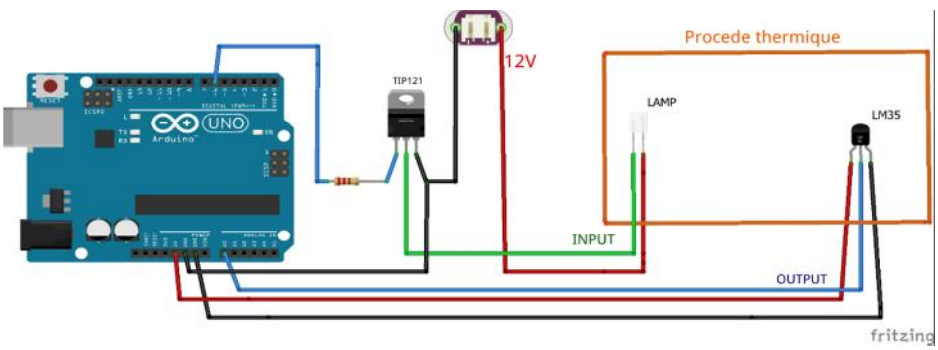


Figure III.1 Branchement du procédé avec la carte Arduino

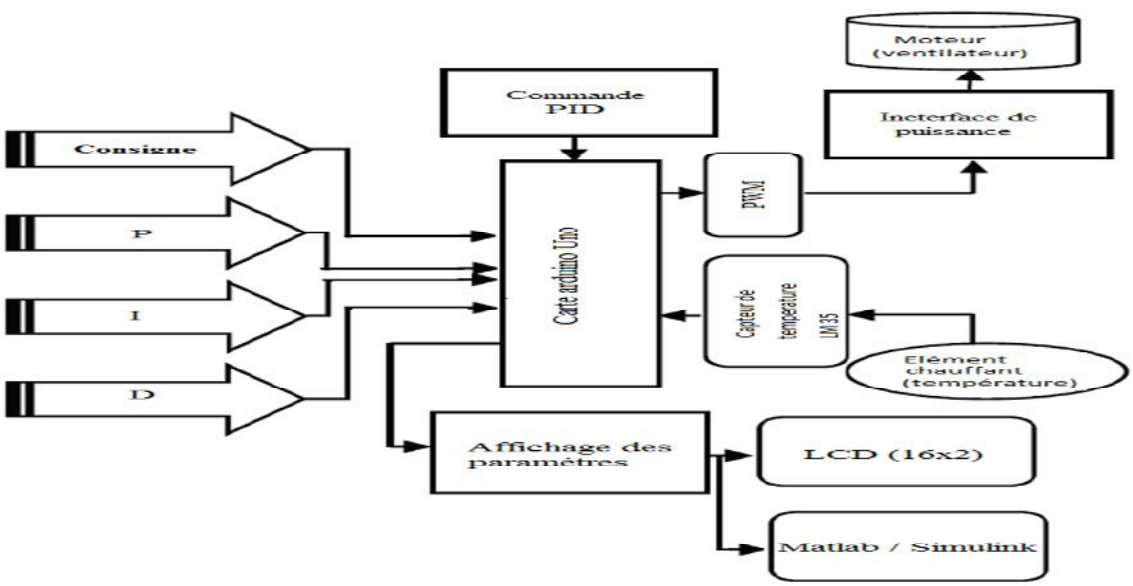


Figure III-2 Schémas Synoptique de système de commande thermique

Le système thermiques est constitué de :

- Une carte Arduino.
- Capteur de température LM35.
- Transistor TIP122.
- Diode 1N4004.
- Batterie 12v.
- Lampe halogène 12v (élément chauffant
- Logiciel IDE.
- Logiciel Matlab/ Simulink.

II-1) La carte Arduino :

La carte Arduino utilisée est la UNO R3 pour :

- l'acquisition des données provenant du capteur de température (donnée reçu sur l'entrée analogique A_0).
- Commander un système de régulation de la température via la pin numérique numéro 7.

II-2) Présentation du capteur de température [21]:

Le capteur de température LM35 est capable de mesurer des températures de 0 à 100 °C avec une précision de 1.5 °C. Il est représenté par la figure III-2 ci-dessous fait partie des capteurs de température électroniques de précision en structure intégrée.

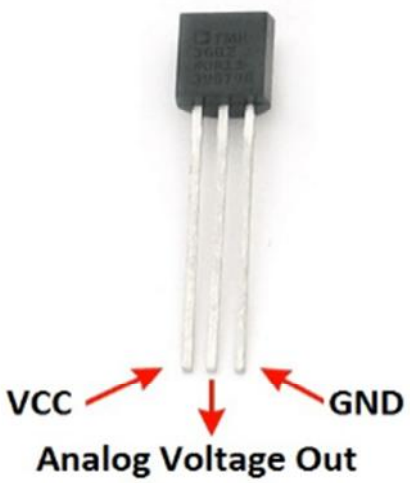


Figure III-3 – Capteur de température LM35

D'après la fiche technique : $10mV \rightarrow C^{o}$

Autrement un volt correspond à 100 degrés Celsius.

La lecture analogique d'un signal de 0 à 5V étant codée de 0 à 1023.

La formule descriptive de la température en fonction de la tension à l'entrée du capteur est donnée par la formule suivante la suivante :

$$Temperature[°C] = 5 \times 100 \times \frac{Analog\ Pin}{1023} \quad (1)$$

5 : Représente la tension d'alimentation du capteur en Volt.

100 : La valeur maximale mesurable par le capteur de température

1023: Représente résolution de l'entrée analogique

Analog Pin : Représente la tension délivrée par capteur de température en Volt.

II-3) Le transistor TIP121 [21]:

C'est un transistor Darlington NPN qui permet d'amplifier le courant jusqu'à 5A avec son gain d'amplification "au minimum" $\beta = 1000$.

II-4) La diode 1N4004 [21] :

Elle joue le rôle d'une diode à roue libre, permettant la protection contre les surtensions.

II-5) Afficheur LCD 16x2 :

Dont le rôle est l'affichage de la température à l'intérieur du système thermique. Cet afficheur constitué de 2 lignes de 16 caractères (16×2).

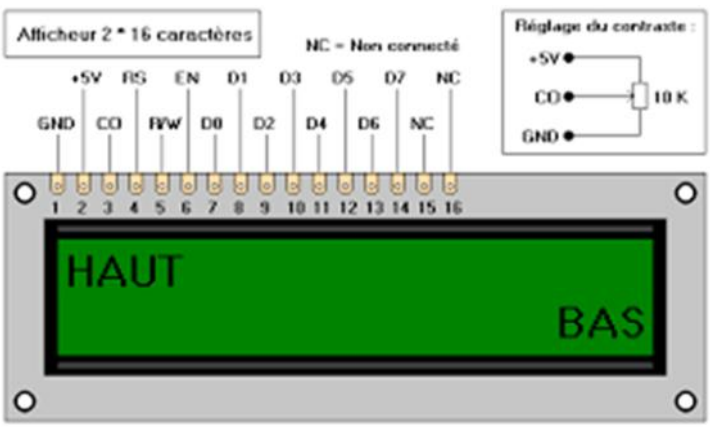


Figure III-4 afficheur LCD16x2

II.6) Moteur à courant continue :

Ce moteur est représenté par la figure III-4est commandé avec la carte Arduino qui fonctionne

avec une tension de 3 à 6V.

Figure III-5 : Connexion d'un moteur CC sur le pin 7 de l'Arduino (Voir Annexe B)

III) L'environnement Matlab/Simulink et interfaçage Arduino :

III-1) L'environnement Matlab/Simulink :

C'est un logiciel de calcul mathématique pour les ingénieurs et les scientifiques créé par Math Works :

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 module de Matlab permettant la simulation multi-domaine. Il fournit un environnement graphique interactif grâce à un ensemble de bibliothèques intégrées et extensible, permettant la simulation d'une variété de systèmes.

III-2) L'interfaçage Arduino et Matlab/Simulink :

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

1. Programmation de la carte Arduino Uno comme une carte d'interface.
2. Utilisation du package Arduino I/O (INPUT/OUTPUT).
3. Utilisation du package Arduino Target.

Dans notre cas nous avons opté pour la deuxième méthode, pour réaliser l'interfaçage Input/Output, qui permet de faire communiquer 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 permet de recevoir les requêtes envoyées via la liaison série (USB) et de répondre en sortie selon ce qui est reçu. Les étapes de configuration de cette méthode sont structurées comme suit :

III.2.1) Pré-chargement du programme dans la carte Arduino :

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 : "Arduino IO/pde'/adiosrv"
5. Charger le fichier adiosrv.pde (adiosrv: Analog and Digital Input and Output Server for MATLAB).

vers le logiciel Arduino.

6. Televerser .

La carte Arduino UNO est maintenant configurée pour être utilisé pour assurer l'interfaçage Entrées/Sorties.

III.2.2)Installation du package Arduino IO :

Les étapes résumant l'installation sont indiquées ci-dessous :

1. Lancer le logiciel Matlab et se placer dans le répertoire E :/arduinoio.
2. Exécuter la commande : Install-arduino.
3. Relancer le module simulink.

Après l'exécution de toutes ces étapes, la fenêtre ci-dessous se présente dont la bibliothèque Arduino IO library.

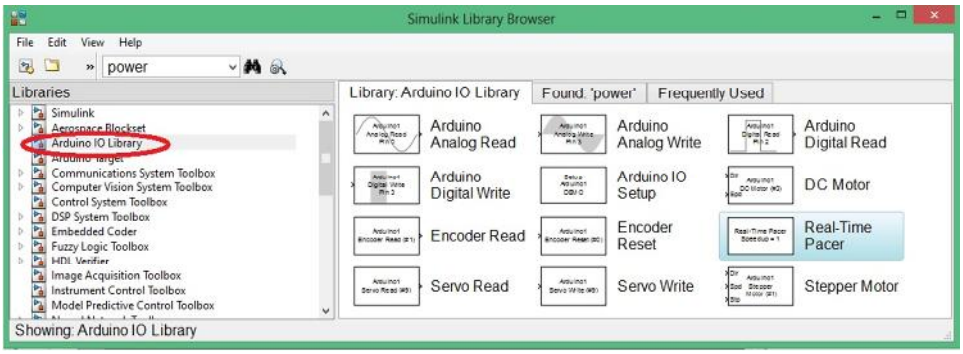


Figure III-6 – ArduinoIO Library

III.2.3) Exploitation de la bibliothèque ArduinoIO sous Simulink :

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

Figure III-7 – Les Blocs d'ArduinoIO nécessaires pour la commande

- **Real-Time Pacer**: Ce bloc permet de ralentir le temps de simulation. Le coeffecient de ralentissement est contrôlé par le paramètre *Speedup*.
- – **Arduino IO Setup**: Ce bloc a pour rôle de sélectionner le numéro du port de la carte Arduino assurant la communication.
- **Arduino Analog Read**: Ce bloc a pour rôle de sélectionner le pin recevant les données issues du capteur.
- **Arduino Analog Write**: Ce bloc assure la configuration en PWM à envoyer vers l'actionneur.

III.2.4) Exploitation du package ArduinoIO sous Matlab :

Le package Arduino I/O 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 sera connectée avec la commande :

```
>> a = arduino( ''port );                    (2)
```

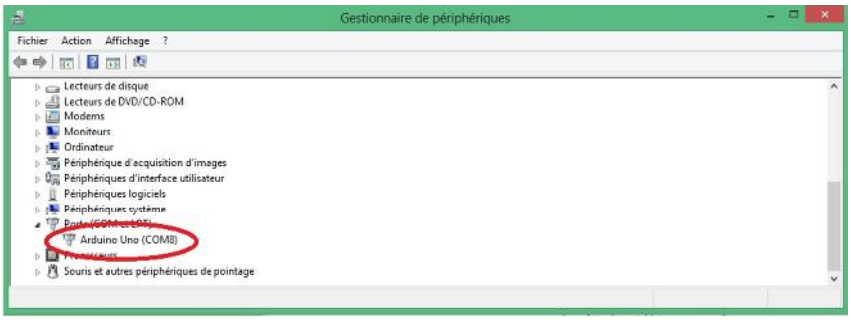


Figure III-8 – Emplacement du port de communication de la carte Arduino.

IV) Modélisation du procédé thermique :

Le but de cette partie est de déterminer la fonction de transfert échantillonnée de notre procédé thermique 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.1) Présentation de l'étape d'identification avec Matlab :

Cette étape est constituée de deux parties. La première est assurée par l'environnement Simulink et le package Arduino I/O pour l'envoie et l'acquisition des données. La deuxième partie est assurée par l'outil *System identification* sous l'environnement Matlab représentée par la figure III-8 ci-dessous:

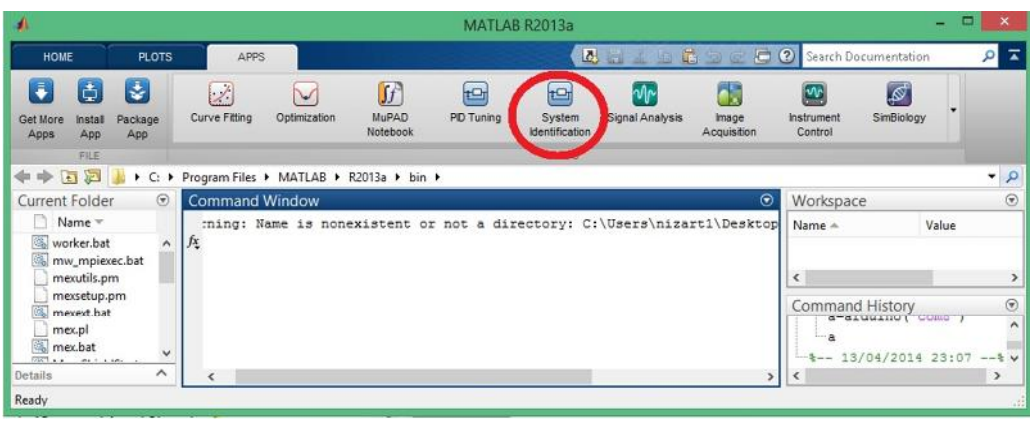


Figure III-9 – L'utilisation de l'outil System Identification

IV.2) Acquisition de la réponse indicielle du système :

L'étude de la réponse d'un système dont l'entrée est un échelon unitaire en tension permet d'identifier la réponse de notre système, ayant pour objectif la détermination de l'équation physique descriptive du système.

Le modèle Simulink permet de visualiser la réponse du système en injectant un échelon unitaire à l'entrée selon le schéma suivant :

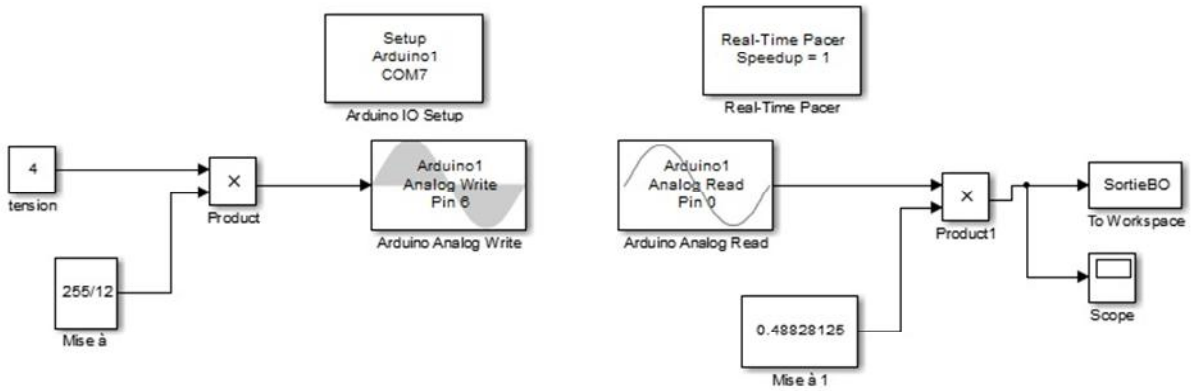


Figure III-10 – Modèle Simulink pour la visualisation de la réponse indicielle

IV.3) Détermination de la fonction de transfert G(z) :

Après avoir visualisé la réponse du système, on passe à la détermination de la fonction de transfert *G(z) en important les données provenant du système*, dont les étapes sont indiquées ci-dessous :

1. Ouvrir l'outil *System identification Tool représenté par la figure III-10 ci-dessous :*

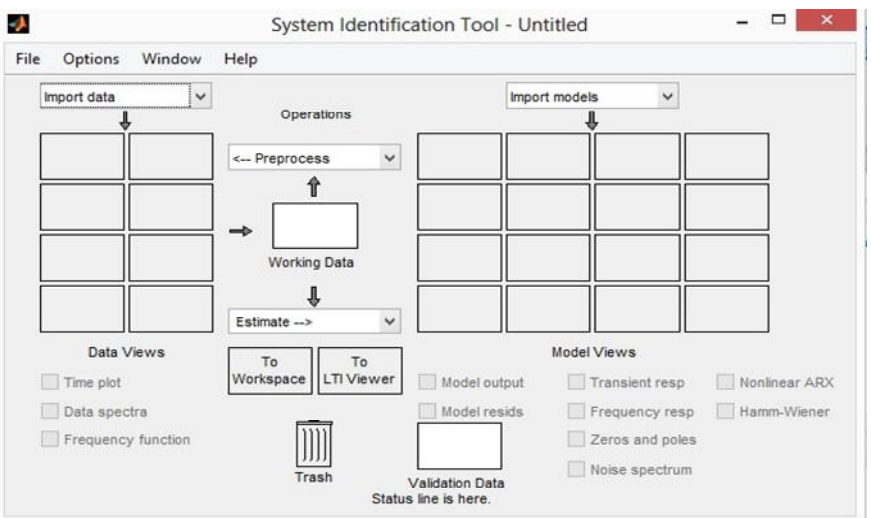


Figure III-11 – L'interface de l'outil System identification

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

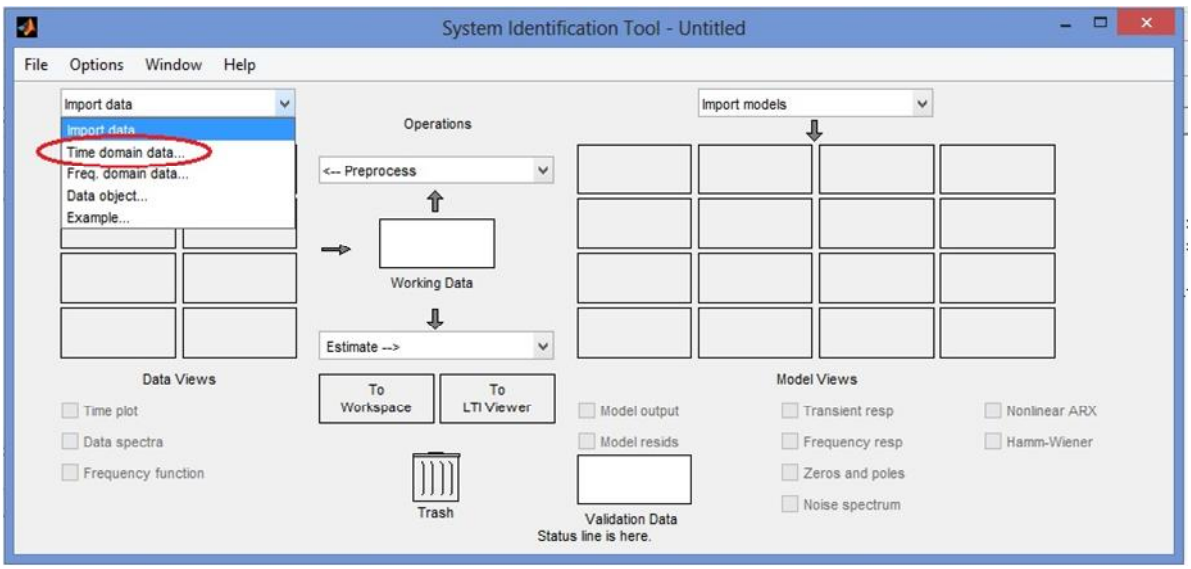


Figure III-12 – Choix des types des données "Time Domain Data"

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

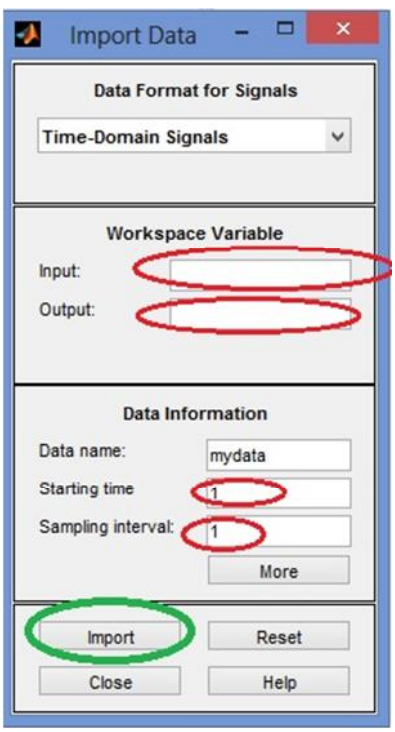


Figure III-13 – Saisie des données relatives aux Input et Output du système

4. Cliquer sur *Estimate* et choisir *Transfer Function Models*

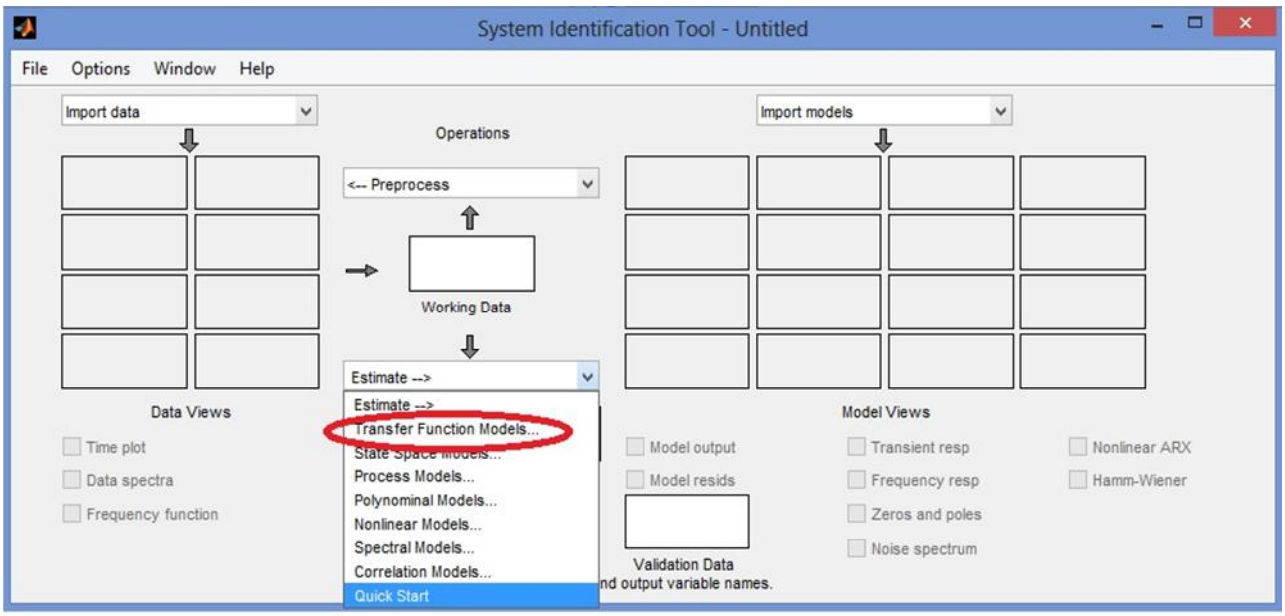


Figure III-14 – Choix de la description du système à estimer "Tranfer Function"

5. Entrer le nombre de pôle et de zéro et cliquer sur *Discrete-Time* ensuite cliquer sur *Estimate*.

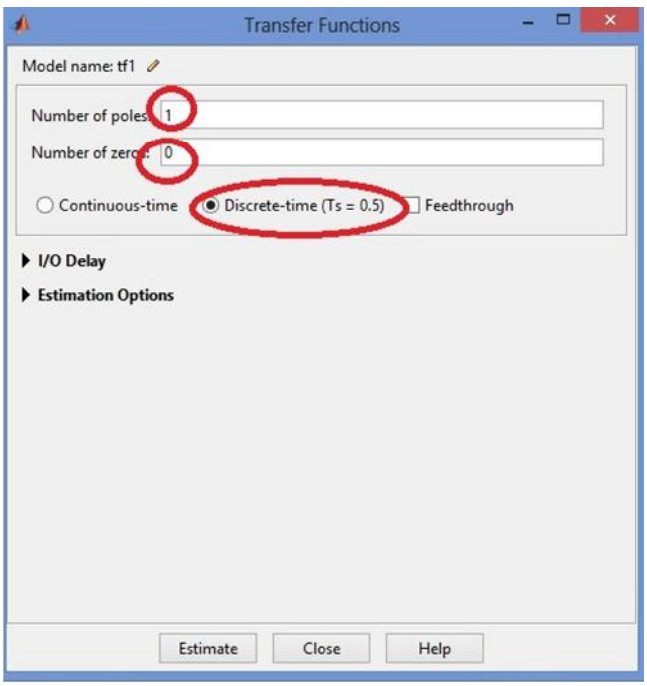


Figure III-15 – Choix du nombre des pôles et zéros de la fonction de transfert à estimer.

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

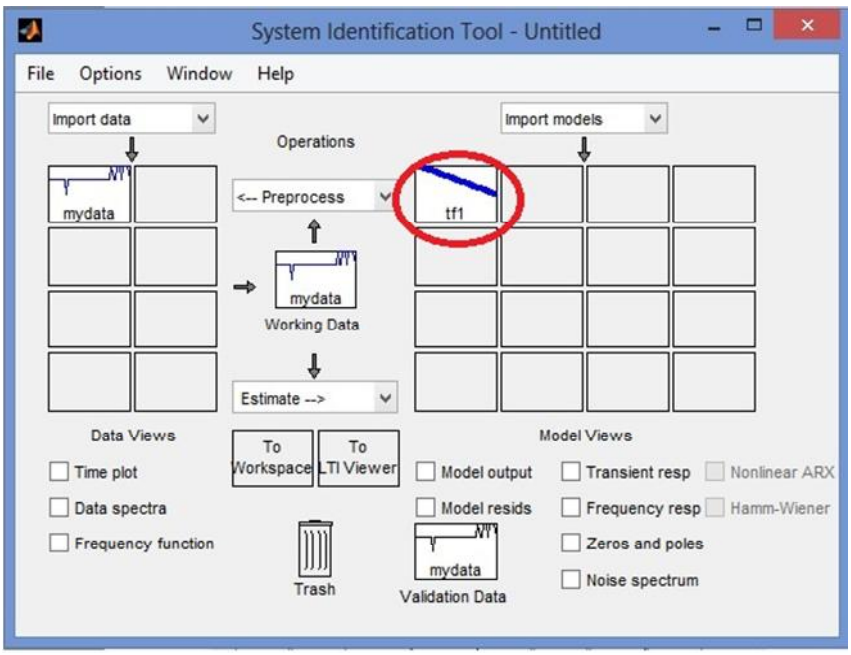


Figure III-16 – Visualisation du résultat de l'estimation

7. Une fenêtre apparait dans laquelle on trouve *G(z)*.

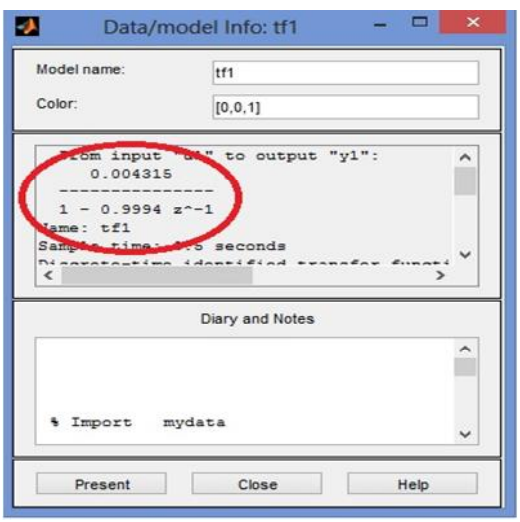


Figure III-17 – Récupération de la fonction de transfert estimée.

V)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 *PID Tuning* pour déterminer les différents paramètres de notre régulateur K_p, K_i et K_D en fonction de la commande à utiliser, ces paramètres sont choisis empiriquement (voir annexe D).
- La deuxième partie consiste à implémenter le correcteur *PID(z)* sur Simulink à téléverser sur la carte Arduino.

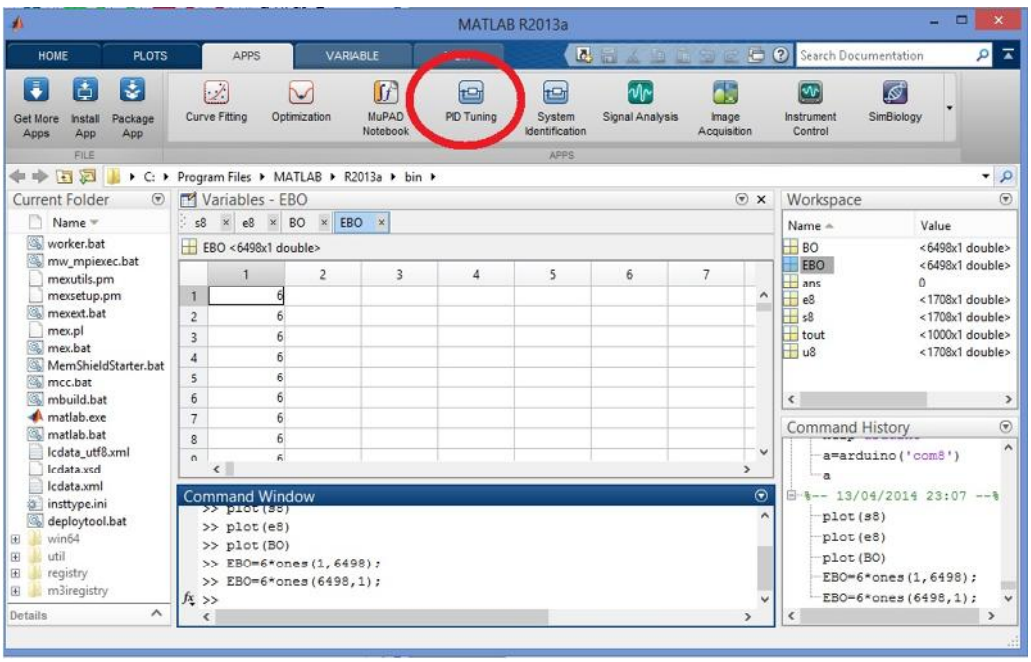


Figure III-18 – Emplacement de l'outil PID tuning

V.1) Synthèse du régulateur numérique :

Les étapes de la synthèse du régulateur PID sont résumées comme suit :

1. Ouvrir l'outil *PID Tuner*

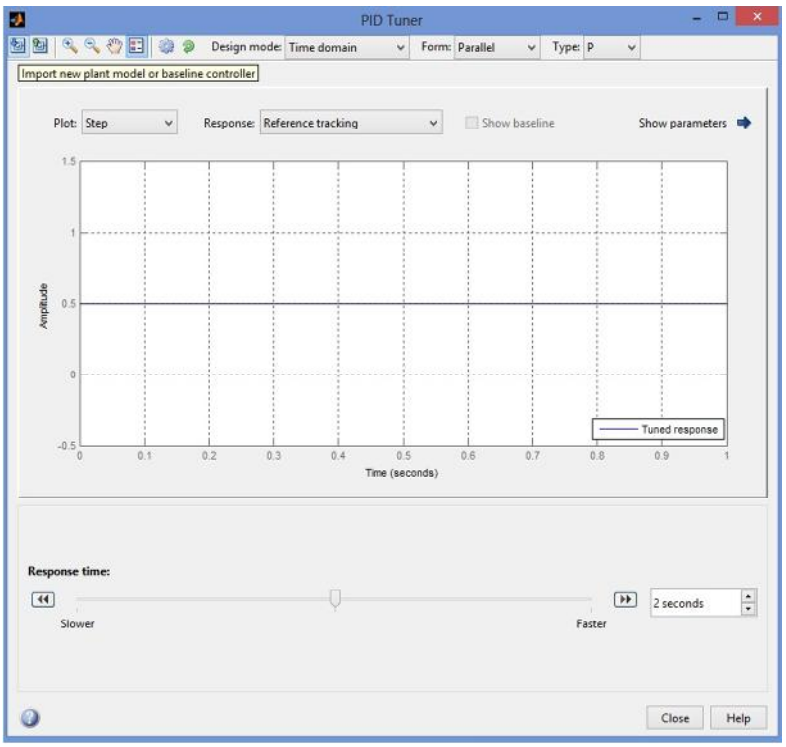


Figure III-19 – Interface de l'outil "PID tuning"

2. Cliquer sur *Import new plant*, une nouvelle fenêtre apparaît.

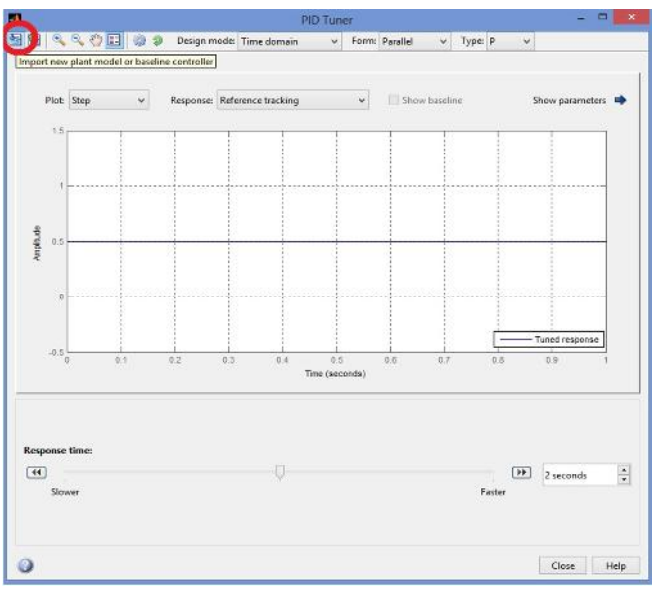


Figure III-20 – Importation du modèle estimé.

3. Une nouvelle fenêtre apparaît dans laquelle on va sélectionner **tf1** ensuite cliquer sur *import* puis *close*.

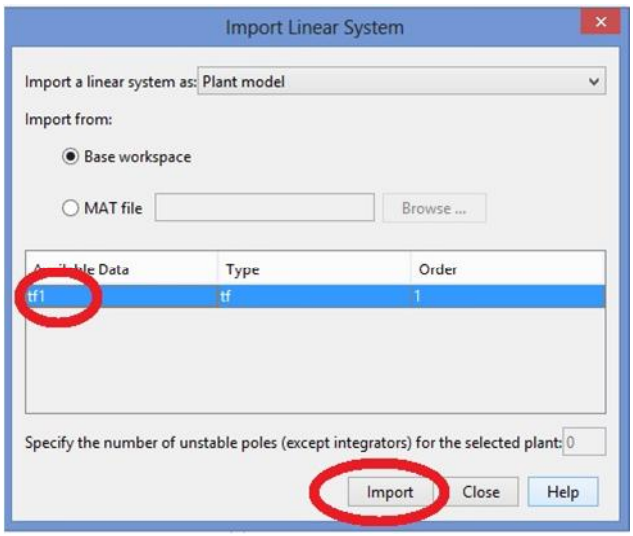


Figure III-21 – Interface de l'outil "Import Linear System"

4. Revenir à la fenêtre *PID Tuner*, dans la quelle on peut choisir le type de régulateur à implémenter et visualiser la réponse du système.

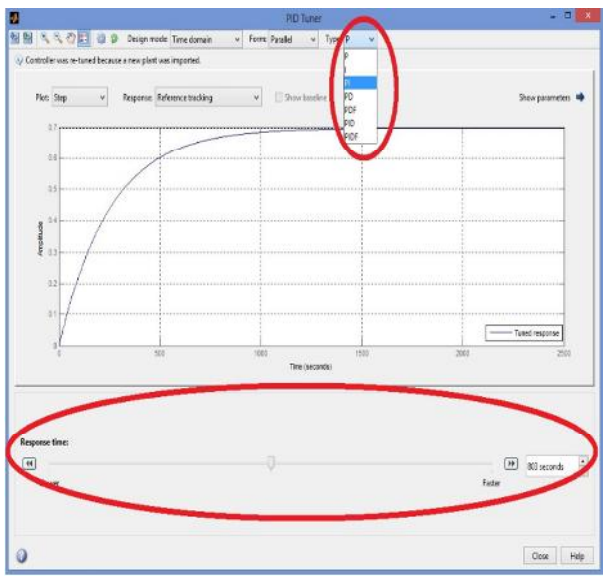


Figure III-22 – Choix du régulateur à implémenter

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

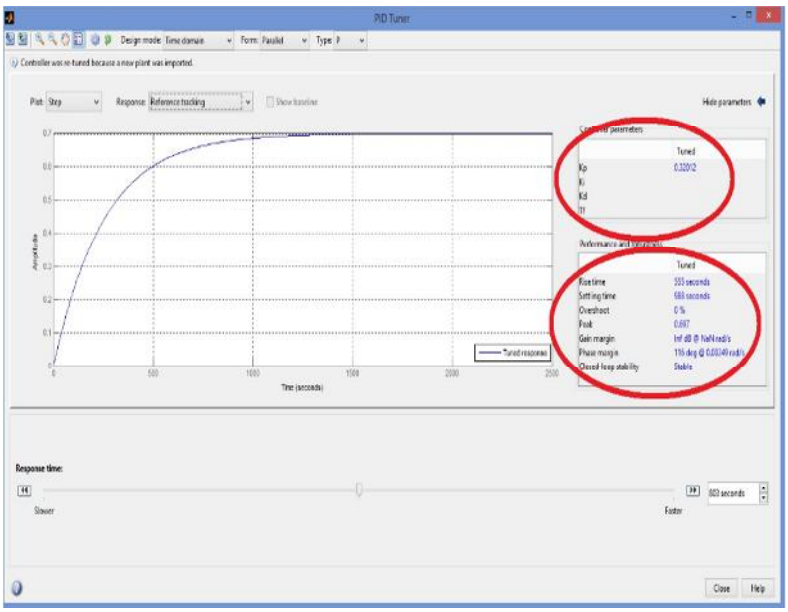


Figure III-23 – Récupération des paramétrés du régulateur

V.2) Implémentation de la loi commande sous Simulink :

La boucle du système asservi à implémenter sur Simulink est représentée par la figure suivante :

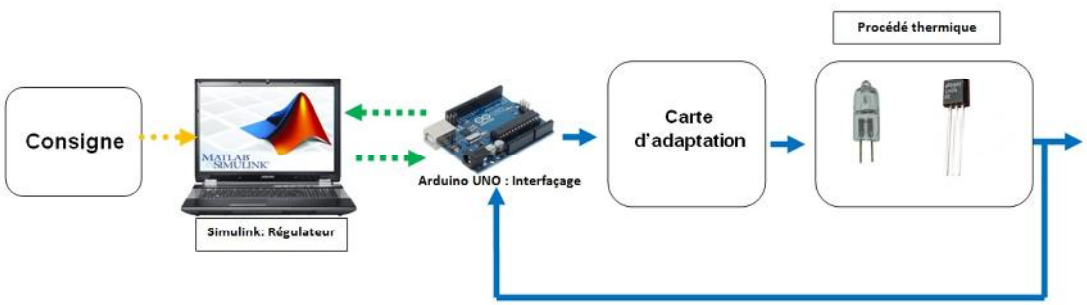


Figure III-24 – Synoptique de la boucle d'asservissement à implémenter

L'asservissement de notre procédé est assuré par le modèle Simulink représenté par la figure ci-dessous qui regroupe la consigne, le comparateur, le correcteur *PID(z)*, le traitement de la température issue du capteur et l'envoie de la commande PWM.

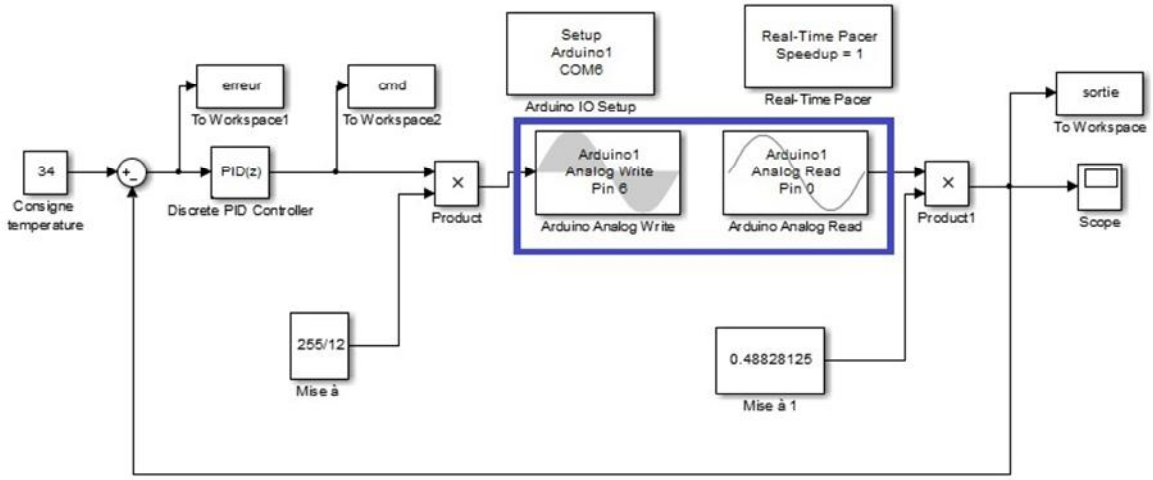


Figure III-25 – Modèle Simulink du système asservi en température

L'appui deux fois sur le bloc *PID(z)* permet d'introduire les paramétrés K_p K_i K_d et de configurer le régulateur selon l'objectif de commande.

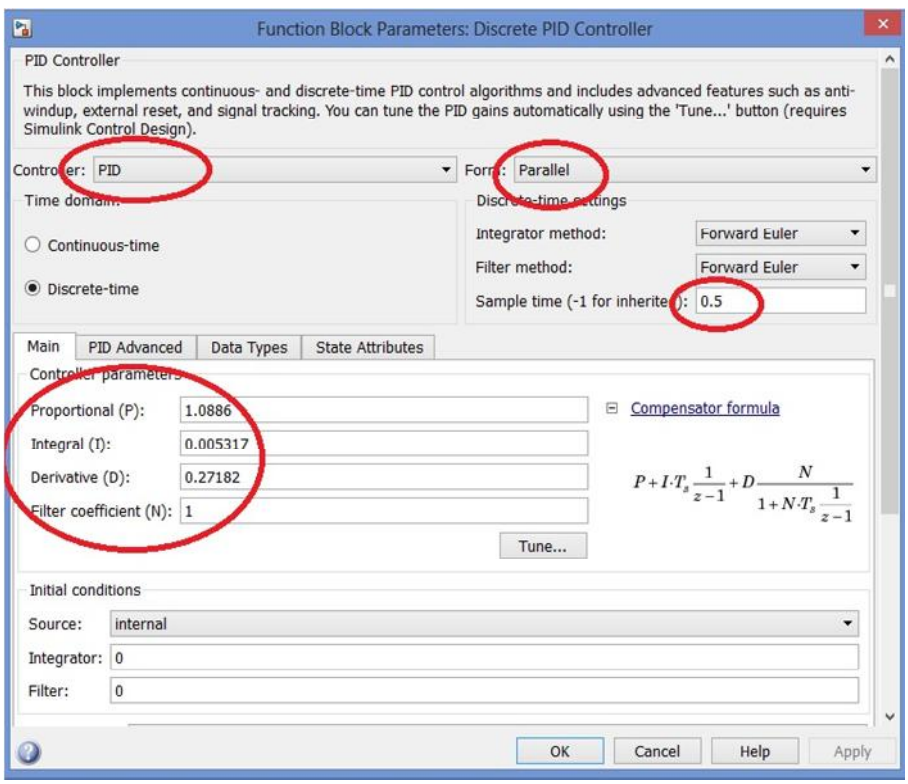


Figure III-26 – Recueil des paramètres du régulateur

V.3) Implémentation de la commande sur la carte Arduino :

Dans cette partie on va utiliser les fonctions présentées dans l'Arduino pour envoyer (la commande) et acquérir (la température instantanée). La loi de régulation est introduite directement sur la carte Arduino. La figure ci-dessous illustre le schéma synoptique de la loi de régulation implémentée sur la carte Arduino.

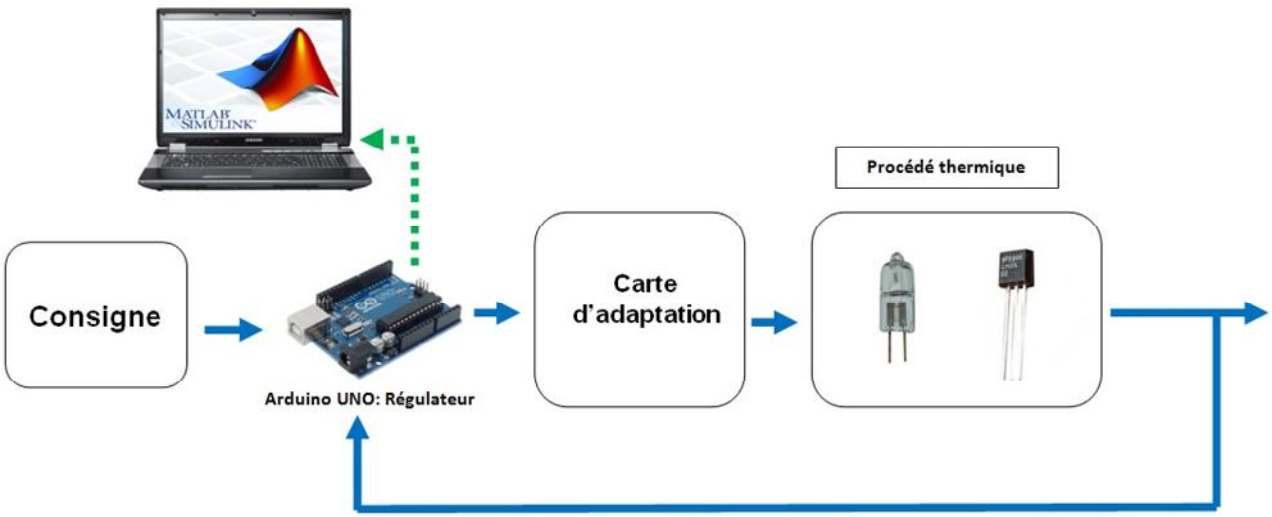


Figure-III-27 – Schéma synoptique de la loi de régulation implémentée sur la carte ARDUINO

V.3.1) Le régulateur Numérique PID:

Ce type de régulateur est défini par l'équation différentielle suivante :

$$U_{PID}(t) = Kp \times e(t) + Ki \times \int_0^t e(\tau)\, d(\tau) + Kd \times d/dt\, e(t) \qquad (3)$$

L'équation de contrôle PID peut être exprimée de plusieurs manières, mais une formule générale est donnée par l'équation suivante [PID Control : A brief introduction and guide, using Arduino.] :

$$PID = Kp \times erreur + Ki \times \Sigma(erreur \times \Delta t) \times Kd\, \frac{(erreur - erreur\ precedente)}{\Delta t} \qquad (4)$$

V.3.2) L'implémentation du régulateur PID :(voir le code dans l'annexe D) :

VI) Conclusion :

Dans ce chapitre, nous avons réalisé une commande d'un système thermique à base d'une carte arduino. L'avantage tiré est de connaître et réguler en temps réel les données acquis du système via un régulateur PID. Les résultats obtenus confirment la faisabilité du système conçu.

Conclusion générale

Dans ce mémoire nous avons présenté la conception et réalisation d'un système thermique composé d'un capteur de température et un ventilateur dont le rôle est de faire diminuer la température du système. Le refroidissement est commandé par une consigne (valeur de la température désirée). A cet effet nous avons fait appel au logiciel Matlab pour l'élaboration de la loi de la commande et la visualisation des résultats obtenus. Le fonctionnement du système confirme la faisabilité de la matérialisation du régulateur PID sur Matlab.

Ensuite nous avons utilisé une carte Arduino pour assurer l'interfaçage entre le système thermique et le Logiciel Matlab. Les données reçues du capteur de température par la carte sont traitées et envoyées pour commander le refroidissement du système selon valeur de consigne fixée par l'utilisateur. Dans cette application nous avons essayé plusieurs valeurs de la température dont le fonctionnement répond aux diverses conditions.

Comme perspectives de ce travail d'autres lois de commandes à base d'autres grandeurs physiques peuvent être adoptés tel que : la pression, l'humidité, le débit …etc. En outre les méthodes et outils utilisés au cours de ce mémoire peuvent être étendus à d'autre système pour améliorer leurs performances. L'exploitation des capteurs à distance ou des outils de communications sans fils tel que le Wifi, le Bluetooth et le Xbee en vue d'assurer la commande à distance peut être adoptée.

Nous espérons que ce modeste travail sera bénéfique pour les promotions futures et sera utilisé pour être amélioré.

Bibliographie

Bibliographie

Ouvrages :

- [1] PSCAL MASSON «Electronique avec Arduino », École Polytechnique Universitaire de Nice Sophia-Antipolis, Edition 2015-2016-V32
- [2] ERIC BARTMANN «LE GRAND LIVRE D'ARDUINO» EYROLLES 2015
- [3] TERO KARVINEN, KIMMO KARVINEN, VILLE VALTOKARI traduit par DOMINIQUE MANIEZ «Les capteurs pour Arduino et Raspberry PI Tutoriels et projets
- [4] Patrick Prouvost, «Automatique - Contrôle et régulation Cours, exercices et problèmes corrigés», DUNOD
- [5]Michel Grout, Patrick Salaun, «Instrumentation industrielle spécification des capteurs et vannes de regulation», DUNOD 2015
- [6] J.M RETIF, «Automatique Régulation», Institut National des Sciences Appliquees de Lyon ; 2008
- [7] ALINA BESANÇON-VODA et SYLVIANNE GENTIL «Régulateurs PID analogique et numérique», Technique de l'ingénieur : R7416 (03/1999)
- [8] GEORGE ASCH ET COLL «les capteur capteurs en instrumentation industrielle», DUNOD 2010
- [9] FRANCIS COTTET, EMMANUEL GROLLEAU «SYSTÈMES TEMPS RÉEL DE CONTRÔLE-COMMANDE Conception et implémentation », DUNOD 2005
- [10] CHELLY NIZAR, CHARED AMINE « FORMATION ARDUINO MATLAB/ SUMILINK Commande d'un système thermique à l'aide de la carte arduino UNO» Hammamet 2014
- [11] J.M RETIF «AUTOMATIQUE REGULATION» Institut National des sciences Appliquées de Lyon, Edition 2008

Sites internet:

- [12] https: //chellynizarblog.files.wordpress.com/2015/10/report1.pdf
- [13] https://www.carnetdumaker.net/articles/tags/capteur/
- [14] http: //www.mathworks.com/
- [15] http: //www.arduino.cc/
- [16] https:// www.txrobotic.com

- [17] X.HINAULT. www.mon-club-elec.fr.

Mémoires :

- [18] Sakli MOUADH – Génie Électrique & Automatique; Ingénieur diplômé de l'École Nationale d'Ingénieurs de GABÈS, TUNISIE «Régulation Industrielle de Processus».
  http://www.bhautomation.fr/Download/Automaticiens/Projets_automatisme_2007_S_MOUADH.pdf
- [19] G.ZIDANI, M.R.REZOUG- Génie Électrique «étude et réalisation d'une carte de contrôle par Arduino via le système Androïde».
- [20] Emmanuel DE GEEST, Mémoire fin d'études réalisé en vue de l'obtention du grade d'Ingénieur Civil Electricien (Electronique), «Méthodes d'optimisation pour le réglage de contrôleurs PID», Université de Liège, Faculté des Sciences Appliquées, Année Académique 2000-2001.
- [21]datasheetLM35SNIS159G–AUGUST 1999–REVISED AUGUST 2016www.Ti .com

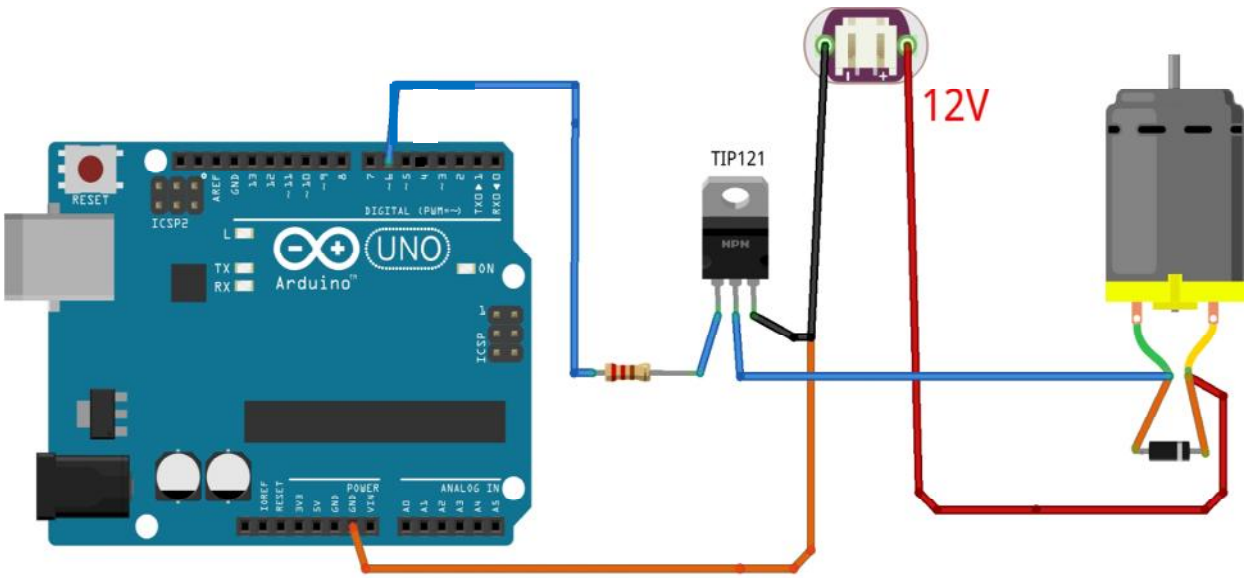
Annexes

Annexe N°A

Tableau comparatif des différentes cartes Arduino

Cartes Arduino / Caractéristiques	UNO R3 (classique & CMS)	UNO R3 Ethernet (classique & POE)	Leonardo	Mega 2560	Mega ADK	DUE	Esplora	Mini	Nano	Yun (classique & POE)	Zero PRO
Microcontrôleur	ATmega328P	ATmega328P	ATmega32u4	ATmega2560	ATmega2560	AT91SAM3X8E	ATmega32u4	ATmega328P	ATmega328P	ATmega32u4	ATSAMD21G18
Cadencement Horloge	16 MHz	16 MHz	16 MHz	16 MHz	16 MHz	84 MHz	16 MHz	16 MHz	16 MHz	16 MHz	48 Mhz
Tension d'entrée	7 - 12V	7 - 12V	7 - 12V	7 - 12V	7 - 12V	7 - 12V	7 - 12V	7 - 9V	7 - 9V	5V	5V
Tension de fonctionnement	5V	5V	5V	5V	5V	3,3V	5V	5V	5V	5V	3,3V
Entrée/Sortie Numérique	14/6	14/4	20/7	54/15	54/15	54/12	✗	14/6	14/6	20/7	14/12
Entrée-Sortie (PWM) Analogique	6/0	6/0	12/0	16/0	16/0	12/2 (DAC)	✗	8/0	8/0	12/0	6/1 (DAC)
Mémoire vive (Flash)	32 Ko	32 Ko	32 Ko	256 Ko	256 Ko	512 Ko	32 Ko	32 Ko	32 Ko	32 Ko	256 Ko
Mémoire vive (SRAM)	2 Ko	2 Ko	2,5 Ko	8 Ko	8 Ko	96 Ko	2,5 Ko	2 Ko	2 Ko	2,5 Ko	32 Ko
Mémoire morte (EEPROM)	1 Ko	1 Ko	1 Ko	4 Ko	4 Ko	✗	1 Ko	1 Ko	1 Ko	1 Ko	16 Ko
Interface USB	USB-B mâle	USB-B mâle	Micro-USB	USB-B mâle	USB-B mâle & USB-A pour Androïd	2 ports micro-USB (Native et programming)	Micro-USB	✗	Mini-USB	Micro-USB	2 ports micro-USB (Native et programming)
Port UART	1	1	1	4	4	4	✗	✗	1	1	2
Carte SD	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗
Ethernet	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗
Wi-Fi	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Dimensions	68x53mm	68x53mm	68x53mm	101x53mm	101x53mm	101x53mm	165x60mm	30x18mm	45x18mm	68x53mm	68x53mm

Annexe B



Branchement de la carte Arduino UNO avec un moteur DC(ventilateur).

Annexe C

Méthodes de réglage de la boucle PID

IL existe plusieurs méthodes pour régler une boucle PID. Le choix de la méthode dépendra en grande partie de la possibilité ou non de mettre la boucle hors production pour la mise au point ainsi que de la vitesse de réponse du système. Si le système peut être mis hors production, la meilleure méthode de réglage consiste souvent à soumettre le système à un changement de consigne, à mesurer la réponse en fonction du temps et à l'aide de cette réponse à déterminer les paramètres de la régulation.

- La première méthode de réglage consiste à mettre les valeurs I et D à zéro. Augmenter ensuite le gain P jusqu'à ce que la sortie de la boucle oscille. Puis, augmenter le gain I jusqu'à ce que cesse l'oscillation. Enfin, augmenter le gain D jusqu'à ce que la boucle soit suffisamment rapide pour atteindre rapidement sa consigne. Le réglage d'une boucle PID rapide provoque habituellement un léger dépassement de consigne pour avoir une montée plus rapide, mais certains systèmes ne le permettent pas.

Paramètre	Temps de montée	Dépassement	Temps de réglage	S.S. Erreur
P	Augmente	Augmente	Chang. faible	Diminue
I	Diminue	Augmente	Augmente	Elimine
D	Chang. faible	Diminue	Diminue	Chang. faible

Effets de l'augmentation des paramètres

- La 2eme méthode de réglage est la méthode de "Ziegler-Nichols", introduite par John G. Ziegler et Nathaniel B. Nichols. Elle commence comme la méthode précédente: réglage des gains I et D à zéro et accroissement du gain P jusqu'à ce que la sortie du procédé commence à osciller. Noter alors le gain critique (K_c) et la période d'oscillation de la sortie (P_c). Ajuster alors les termes P, I et D de la boucle comme sur la table ci-dessous:

Type de régulation	**P**	**I**	**D**
P	$.5K_c$		
PI	$.45K_c$	$1.2/P_c$	
PID	$.6K_c$	$2/P_c$	$P \times P_c/8$

Annexe D

L'implémentation du régulateur PID

Le code suivant permet d'implémenter le régulateur PID sur la carte Arduino.

```
float delta _ erreur =0;
float somme _ erreur = 0;        // Somme des erreurs pour l'intégrateur
float kp = 1.0886; // Coefficient proportionnel
float ki = 0.005317; // Coefficient intégrateur
float kd = 0.27182; // Coefficient dérivateur
float ep, up, temp, u, e, integral, cmd ;
float derive;
intconsigne;
void setup ( ) {
Serial. begin (9600) ;
}
void loop ( )
{
temp=a n a l o g R e a d (0);
temp=temp* 0. 8828125;
e=35-temp;
delta_erreur = (e-ep) / 0.5;
somme_erreur=somme_erreur+e*0.5;
cmd = kp*e + ki*somme_erreur + kd*delta_erreur;
analog Write (6, cmd*(255/12));
ep=e ;
delay (500);// periode d'echantillonage
Serial. write (analog Read (0)); // envoie de la donnee sur le portserie}
```