République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université 8Mai 1945 – Guelma

Faculté des sciences et de la Technologie

Département d'Electronique et Télécommunications

Mémoire de fin d'étude

Pour l'obtention du diplôme de Master Académique

Domaine : **Sciences et Technologie**

Filière : **Electronique**

Spécialité : **Instrumentation**

Conception et réalisation d'un Régulateur PID numérique de température avec ARDUINO

Présenté par :

Alaa-eddine Bordjiba

Sous la direction de :

Pr. Djalil Boudjehem

Juillet 2019

Remerciement

Tout d'abord, je tiens à remercier le bon Dieu le tout Puissant de m'avoir donné la force et le courage de mener à bien ce modeste travail.

Je tiens à remercier mon directeur de recherche

monsieur Dr. Boudjahem abdeldjalil

j'ai l'honneur de bénéficier de votre riche enseignement, merci pour votre soutien, votre aide, votre orientation et vos conseils si précieux.

Un énorme merci aux membres de jury, vous me faites un grand honneur en acceptant de juger ce travail.

Chaleureux remerciement à ma famille qui m'a soutenus moralement, je remercie mes parents, qui mon encouragé et aidé à arriver à ce stade de ma formation.

je tiens vivement à remercier tous ceux et celles qui m'ont soutenus dans la réalisation de ce modeste travail.

Alaa-eddine Bordjiba

Dédicace

Je dédie ce modeste travail au meilleur des pères

à ma très chère maman qu'ils trouvent en moi la source de leur fierté

qui ne cessent de me donner avec amour le nécessaire pour que je puisse arriver à ce que je suis aujourd'hui.

Que dieux vous protège et que la réussite soit toujours à ma portée pour que je puisse vous combler de bonheur.

à mes soeurs Bouchra et khawter à qui je souhaite un avenir radieux plein de réussite

à mes amis et mes collègues,

à tous ceux qui me sont chers, aux personnes qui m'ont toujours aidé et encouragé, qui étaient toujours à mes côtés

et qui m'ont accompagnaient durant mon chemin d'études.

Alaa-eddine Bordjiba

Sommaire

Liste de figure

Chapitre I : régulateur PID6

Chapitre II : étude de L'ARDUINO

Chapitre III : régulateur PID de température avec ARDUINO

Liste de tableaux

Chapitre I : régulateur PID

Chapitre III : régulateur PID de température avec ARDUINO

Introduction générale

Le terme de régulation est employé lorsqu'on cherche à combattre des perturbations afin de garder une valeur constante par exemple, une température, une pression, un débit ou une hygrométrie...

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

En industrie, les régulateurs PID répondent à plus de 90% de ces besoins. Par exemple, le nombre de régulateurs installés dans une usine industrielle se compte par milliers.

D'abord 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. Ensuite En 1942, Ziegler et Nicols ont proposé deux démarches expérimentales permettant de trouver facilement des paramètres optimums pour une installation donnée. Au fil des années, les propositions de Ziegler et nicols ont été adaptées ou modifiées selon les besoins. Au milieu des années 50, les contrôleurs automatiques PID étaient largement adoptés pour une utilisation industrielle.

Au début des années 1990 et dans le but de fournir des règles d'ajustement simples mais plus performantes que celles de Zeigler-Nichols, Dans ce chapitre on présente les régulateurs PID classiques et les méthodes de synthèse de Ziegler et nicols. A ce stade, il est important d'insister sur le fait que les méthodes présentées ci-après ne sont applicables qu'à des processus non oscillants que l'on trouve surtout dans l'industrie chimique, alimentaire, pétrolière, pharmaceutique, etc [2][5] .

Un contrôleur de température est un dispositif utilisé pour maintenir une température souhaitée à une valeur spécifiée.
L'exemple le plus simple d'un contrôleur de température est un thermostat commun trouvé dans les maisons. Par exemple, un chauffe-eau utilise un thermostat pour contrôler la température de l'eau et la maintenir à une certaine température commandée. Les régulateurs de température sont également utilisés dans les fours.

Lorsqu'une température est définie pour un four, un contrôleur surveille la température réelle à l'intérieur du four.

S'il tombe en dessous de la température définie, il envoie un signal pour activer le chauffage afin de ramener la température au point de consigne. Les thermostats sont également utilisés dans les réfrigérateurs. Donc, si la température devient trop élevée, un contrôleur déclenche une action pour abaisser la température

Chapitre I :

Les Régulateurs PID

1. Introduction

Le terme de régulation est employé lorsqu'on cherche à combattre des perturbations afin de garder une valeur constante par exemple, une température, une pression, un débit ou une hygrométrie...

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

En industrie, les régulateurs PID répondent à plus de 90% de ces besoins. Par exemple, le nombre de régulateurs installés dans une usine industrielle se compte par milliers.

Dans ce chapitre on présente les régulateurs PID classiques et les méthodes de synthèse de Ziegler et nicols. A ce stade, il est important d'insister sur le fait que les méthodes présentées ci-après ne sont applicables qu'à des processus non oscillants que l'on trouve surtout dans l'industrie chimique, alimentaire, pétrolière, pharmaceutique, etc [2][5] .

2. Structure d'un régulateur PID

L'idée de base d'un contrôleur PID est de lire un capteur, puis de calculer la sortie souhaitée de l'actionneur en calculant les réponses proportionnelle, intégrale et dérivée, puis en faisant la somme de ces trois composants pour calculer la sortie. Avant de définir les paramètres d'un contrôleur PID.

2.1. Régulateur PID parallèle :

L'équation utilisée pour décrire le PID parallèle est la forme la plus simple, parfois appelée équation parallèle, car chaque action (P, I et D) se produit en termes distincts de l'équation, l'effet combiné étant une simple somme[6] [3] :

$$m = K_p \left(e + \frac{1}{\tau_i} \int e\, dt + \tau_d \frac{de}{dt} \right) + b$$

Dans l'équation parallèle, chaque paramètre d'action (Kp, τi, τd) est indépendant des autres. Au début, cela peut sembler être un avantage, car cela signifie que chaque réglage effectué sur le contrôleur ne devrait affecter qu'un aspect de son action. Cependant, il est parfois préférable que le paramètre de gain affecte les trois actions de contrôle (P, I et D). Nous pouvons

montrer l'indépendance des trois actions, en divisant l'équation en trois parties différentes, chacune décrivant sa contribution au résultat (Δm) [6] :

$$\Delta m = K_p \left(\frac{\tau_d}{\tau_i} + 1 \right) \Delta e$$

$$\Delta m = \frac{K_p}{\tau_i} \int e \, dt$$

$$\Delta m = K_p \tau_d \frac{de}{dt}$$

Comme vous pouvez le constater, les trois parties de cette équation PID sont complètement séparées, chaque paramètre de réglage (Kp, τi, τd) agissant indépendamment dans le cadre de son propre terme.

2.2. L'action proportionnelle

L'action proportionnelle applique une correction instantanée pour tout écart entre la mesure et la consigne, plus la perturbation est grande, plus la correction apportée est grande. Cette composante seule ne permet pas une grande précision surtout dans les systèmes à faible inertie, comme dans le traitement de l'air, cette rapidité d'action engendre un phénomène appelé le pompage [1].

La bande proportionnelle : C'est l'écart entre la valeur mesurée et la valeur de la consigne notant que si la bande proportionnelle est égale à zéro la régulation fonctionne en tout ou rien.

L'action proportionnelle du régulateur s'exprime de différentes manières, soit par le gain (coefficient multiplicateur de l'écart), soit en % à appliquer pour que la sortie varie 0 à 100 %, soit encore en ***K*** ici on parlera d'écart pour passer de 0% à 100 %[1].

En % : Température mesurée 26 °C, température consigne 24 °C, BP 20 % on aura en sortie de régulateur : (26-24)/ (26*20%) = 38%

En gain : Température mesurée 20°C, température consigne 22°C, pour un gain de 20 on aura ici en sortie : (**22-20)*20=40%**

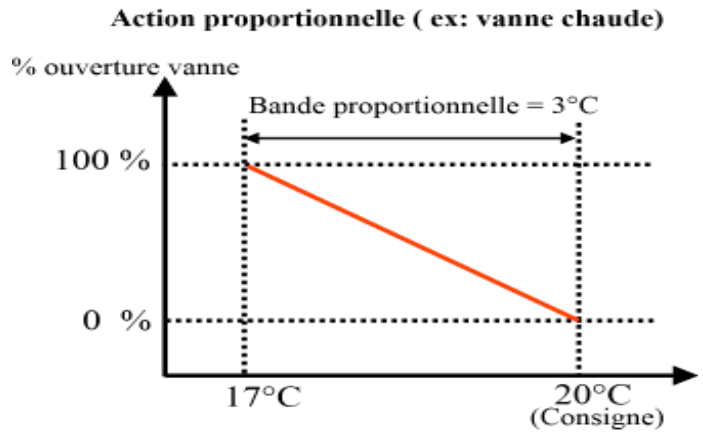

Figure I.1 la courbe de l'action proportionnelle

2.3. L'action intégrale :

Cette action apporte une notion de temps d'intégration à la correction, cette notion de temps s'exprime généralement en seconde.

Cette action est complémentaire à l'action proportionnelle, elle permet de stabiliser dans le temps l'action proportionnelle, plus l'erreur mesurée est constante plus la correction est constante.

2.4. L'action dérivée :

Cette action permet d'anticiper la réponse de la régulation en cas de perturbation rapide ou de modification de consigne ce qui améliore la stabilité du système. On peut donc dire que cette composante permet de compenser tout dépassement excessif de la consigne.

3. Réglage d'un PID :

Le réglage d'un PID consiste à déterminer les coefficients K_p, T_i et T_d afin d'obtenir une réponse adéquate du procédé et de la régulation. Les objectifs sont d'être robustes, rapide et précis.

La robustesse est sans doute le paramètre le plus important. On dit qu'un système est robuste si la régulation fonctionne toujours même si les paramètres de modèle change un peu (varier). Par exemple, les fonctions de transfert de certains procédés peuvent varier en fonction de la température ambiante ou de l'hygrométrie ambiante relativement à la loi de Pascal. Un régulateur doit être capable d'assurer sa tâche même avec des perturbations afin de s'adapter à

des usages non prévus/testés (dérive de production, vieillissement mécanique, environnements extrêmes...)[12] .

La rapidité du régulateur dépend du temps de montée et du temps d'établissement du régime stationnaire.

Le critère de précision est basé sur l'erreur statique La réponse type d'un procédé stable est la suivante :

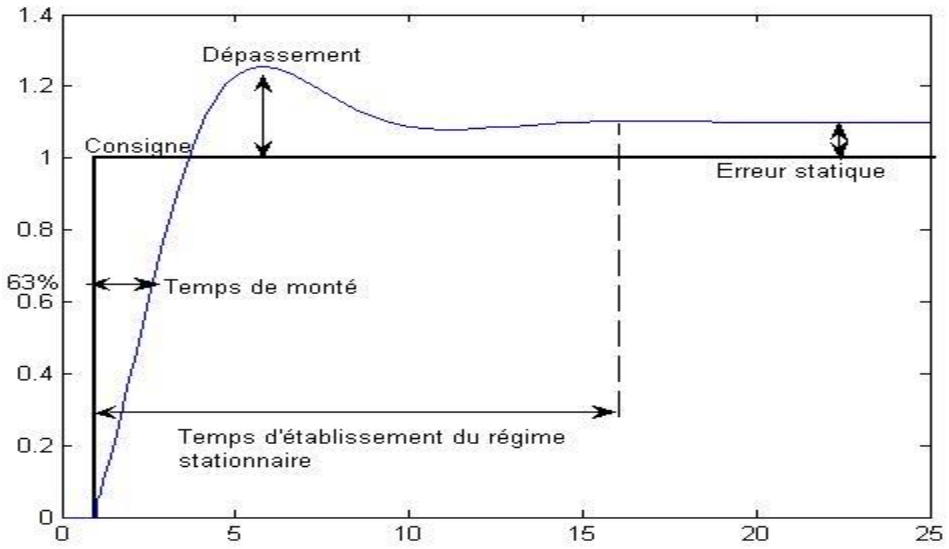

Figure I.2 la courbe de paramètre de réglage PID

Dans le cas des systèmes simples, les paramètres du PID influencent la réponse du système de la manière suivante :

Le gain K : lorsque K augmente, le temps de montée est plus court et l'erreur statique est réduite, mais il provoque un dépassement plus important.

Ti : lorsque 1/ Ti est présent l'erreur statique est annulée. Quand il augmente, la valeur finale est plus rapidement atteinte pour les systèmes présentant de grandes marges de stabilité. Le temps d'établissement en régime stationnaire s'allonge pour les autres systèmes qui vont davantage osciller. Le réglage de ce paramètre dépend donc du comportement dynamique du système et influe sur son amortissement et son temps de réponse.

Td : lorsque ***Td*** augmente, le temps de montée diminue et le dépassement diminue ce qui améliore la stabilité. Toutefois il n'influence pas l'erreur statique. Si ce paramètre est trop élevé dans un premier temps il stabilise le système avec des réactions violentes pouvant saturer le signal de commande sortant du correcteur, et dans un deuxième temps il amplifie de manière exagérée des perturbations brèves[13].

Pour ces trois paramètres, le réglage au-delà d'un seuil trop élevé a pour effet.

D’engendrer une oscillation du système de plus en plus importante menant à l'instabilité : un ***Kp*** trop important rend le système trop sensible, un ***1\Ti*** trop important provoque une intégration trop rapide, un ***Td*** trop important accentue la sensibilité aux bruits de fréquence élevée. C'est pourquoi d'autres régulateurs dits à "avance de phase" ou "retard de phase" sont utilisés pour atteindre les performances désirées.

L'analyse du système avec un PID est simple mais sa conception peut être délicate, voire difficile, car il n'existe pas de méthode unique pour résoudre ce problème. Il faut trouver des compromis, le régulateur idéal n'existe pas. En général, on se fixe un cahier de charge à respecter sur la robustesse, le dépassement et le temps d'établissement du régime stationnaire.

Les méthodes de réglage les plus utilisées en théorie sont les méthodes de Ziegler-Nichols (en boucle ouverte et boucle fermée), la méthode de P. Naslin (polynômes normaux à amortissement réglable), la méthode du lieu de Nyquist inverse (utilise le diagramme de Nyquist).

De manière plus pratique et rapide les professionnels utilisent soit l'identification par modèle de Broïda pour les systèmes stables ou le modèle intégrateur retardé pour les systèmes instables, soit la méthode par approches successives qui répond à une procédure rigoureuse : on règle d'abord l'action P seule pour avoir un dépassement de 10 à 15 % puis l'action dérivée de façon à « raboter » au mieux le dépassement précédent, enfin on ajuste si nécessaire l'action intégrale en se fixant un dépassement final compris entre 5 et 10 %.

Il existe aussi une méthode qui, en supposant connue la fonction de transfert H(p) du système, permet de déterminer un régulateur PID robuste dans le sens où la marge de phase et la pulsation au gain unité (donc la marge de phase/retard) sont fixées à l'avance

Dans certains cas, les performances d'un PID peuvent devenir insuffisantes, en raison par exemple de la présence d'un retard trop important ou d'un procédé à phase non minimale, posant des problèmes de stabilité. On fait alors appel à d'autres algorithmes de réglage.

Avant que le contrôleur PID fonctionne, il doit être adapté à la dynamique du processus à contrôler. Dans la plupart ses cas, les concepteurs donnent les valeurs par défaut pour les termes P, I et D. Ces valeurs ne peuvent pas donner les performances souhaitées et entraînent parfois une instabilité et des performances de contrôle lentes. Différents types de méthodes de réglage sont développés pour régler les contrôleurs PID et nécessitent une attention particulière

de la part de l'opérateur pour la sélection des meilleures valeurs de gains proportionnel, intégral et dérivé.

3.1. Technique de courbe de réaction de processus :

Initialement, nous devons appliquer manuellement certaines sorties de contrôle au système et enregistrer la courbe de réponse. Après cela, nous devons calculer la pente, le temps mort, le temps de montée de la courbe et enfin substituer ces valeurs aux équations P, I et D pour obtenir les valeurs de gain en termes de PID. C'est une technique de réglage en boucle ouverte.

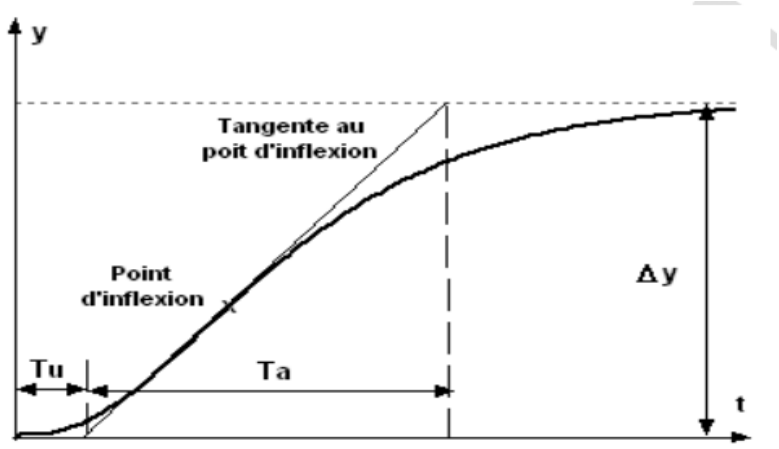

Figure I.3 Courbe de réaction du processus

K étant le gain statique du système : $\frac{\Delta y}{\Delta u}$ en régime statique

3.2. Méthode de Ziegler&Nichols en boucle ouverte

Ziegler&Nichols proposent de calculer les paramètres du régulateur P, PI ou PID à l'aide des recommandations suivantes [4] :

Réglage des paramètres			
Régulateur	Kp	Ti	Td
P: $R(p) = Kp$	$\frac{Ta}{Tu} \frac{1}{K}$		
PI: $R(p) = Kp(1+\frac{1}{T_i p})$	$\frac{Ta}{Tu} \frac{0.9}{K}$	$\frac{3.33}{Tu}$	
PID: $R(p) = K_p (1+ T_d p + \frac{1}{T_i p})$	$\frac{Ta}{Tu} \frac{1.2}{K}$	$\frac{2.0}{Tu}$	$\frac{0.5}{Tu}$

Tableaux I.1 paramètre de régulateur P Pi et Pid

3.2.1. Avantages

- La méthode est facile à mettre en œuvre physiquement et au point de vue calcul
- Elle peut être appliquée à un système déjà en production et permet une adaptation automatisée du régulateur pour s'adapter à l'évolution des paramètres intérieurs (usure) et extérieurs (environnement) au système.

3.2.2. Inconvénients

- Le système peut devenir instable ou passer dans des états dangereux (par exemple pour les systèmes chimiques)
- La méthode peut nécessiter beaucoup de temps si le système réagit très lentement (jours, semaine dans le cas de certaines réactions chimiques) Heureusement de nombreux systèmes ont des temps caractéristiques faibles (systèmes électroniques ou mécaniques).

4. Régulateur PID numérique :

Dans la plupart des documents sur le contrôle numérique, on s'initie rapidement à la transformation z et au plan z, par opposition à au plan s, que est principalement utilisé pour le lieu des racines dans la conception des contrôleurs PID.

Avec le contrôle numérique, nous connectons un calculateur (microcontrôleur) au monde extérieur (analogique).

L'actionneur, l'installation et le capteur sont de véritables appareils dans le monde réel (analogique). La valeur réelle étant contrôlé (C) est détecté par un capteur qui, ici, est modélisé comme un simple gain. Habituellement, le capteur produit un signal électrique ; souvent une tension ; qui est ensuite envoyé au calculateur via un convertisseur analogique-numérique (A / D). Là, le signal original, que ce soit la température, la pression, la position, la vitesse ou que ce soit, est récupéré sous forme numérique à l'intérieur de calculateur en divisant la tension représentative grâce au gain du capteur. Ceci est représenté par Cd dans le diagramme, le «d» indiquant que c'est la représentation numérique de la variable réelle détectée C, à ce stade, l'algorithme pour la comparaison avec la valeur souhaitée (R) peut avoir lieu, l'erreur déterminée en format numérique et l'algorithme de contrôleur exécuté pour arriver à la version numérique de la sortie analogique à envoyer à l'actionneur. Ceci est converti par le convertisseur N / A en un signal réel (U, généralement une tension), qui démarre la chaîne d'actionnement.

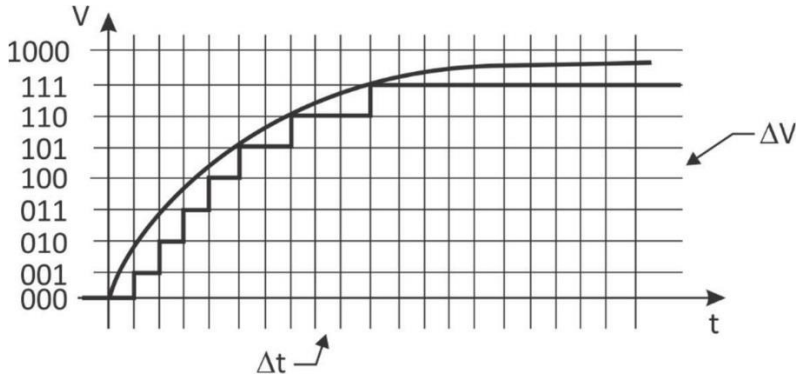

Figure I.4 conversion analogique numérique d'un signale

Comme on peut le voir avec une comparaison minutieuse, aux instants de temps indiqués, les bits seront activés pour représenter que la tension analogique réelle est arrivée au niveau de tension correspondant à un bit. Ainsi, la tension détectée est représenté graphiquement comme forme de courbe en escalier indiquée sous la courbe de tension analogique. Bien sûr, 3 bits ne donnent pas une bonne résolution de la courbe analogique. Trois bits binaires donnent $2^3 = 8$ valeurs possibles.

Généralement, plus de bits sont utilisés pour donner une résolution plus élevée des valeurs analogiques. Dix, par exemple, donnerait $2^{10} = 1024$ niveaux différents, et il est facile de voir que 1024 niveaux permettraient au courbe pour être représenté avec une meilleure fidélité. Trois bits ont été utilisés ici simplement pour montrer la nature du problème de la représentation numérique d'une courbe analogique.

5. La structure de régulateur PID numérique :

Le contrôle numérique est implémenté avec des microcontrôleurs spécialisés développés pour une utilisation industrielle renforcée. Avec l'arrivée de petits microprocesseurs qui sont peu coûteux, la réalisation des régulateurs PID adéquat pour de nombreuses applications mécaniques, thermiques, ou des systèmes de fluide est disponible.

Voici comment. Le programme de contrôle du microcontrôleur s'exécute souvent dans une boucle à pas de temps fixe. Cette boucle est très rapide, particulièrement comparé aux temps de réaction de la plupart des systèmes mécaniques. Les boucles typiques s'exécutent une fois toutes la 1-10 milliseconde. Généralement, cela suffit pour contrôler la plupart des systèmes mécaniques par exemple.

La commande numérique présente de nombreux avantages par rapport à la commande analogique (commande avec des composants électriques standards). Mais le contrôle

analogique bat toujours le contrôle numérique dans cette zone critique de vitesse. Les contrôleurs analogiques répondent pratiquement sans délai, ce qui est nécessaire pour les systèmes très rapides [4].

Comme indiqué ci-dessus, les contrôleurs numériques exécutent un programme en boucle qui fournit la commande. Le schéma figure I.5 montre la structure approximative de ce programme informatique. Ces trois étapes sont effectuées, l'un après l'autre, chaque fois que l'automate tourne en mode automatique.

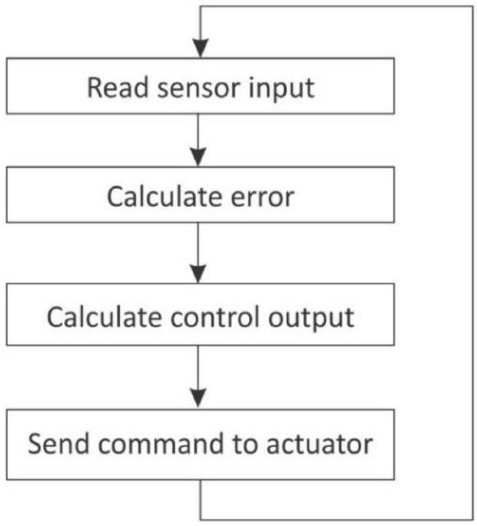

Figure I.5 la structure approximative de régulateur PID

6. Problèmes rencontrés avec le régulateur numérique :

Un problème inhérent au contrôle numérique est que le taux de balayage du régulateur peut être trop lent pour le processus qu'il essaie de contrôler. Les processus thermiques sont généralement lents, donc ce problème ne se rencontre pas là. Mais les processus d'actionnement hydraulique peuvent être très rapides et là, le régulateur doit suivre l'évolution de la nature du processus. Si la fréquence de balayage du contrôleur est trop lente, des changements dans le processus peuvent être omis entre les instances d'échantillonnage du capteur. La figure illustre un scénario possible de ce type.

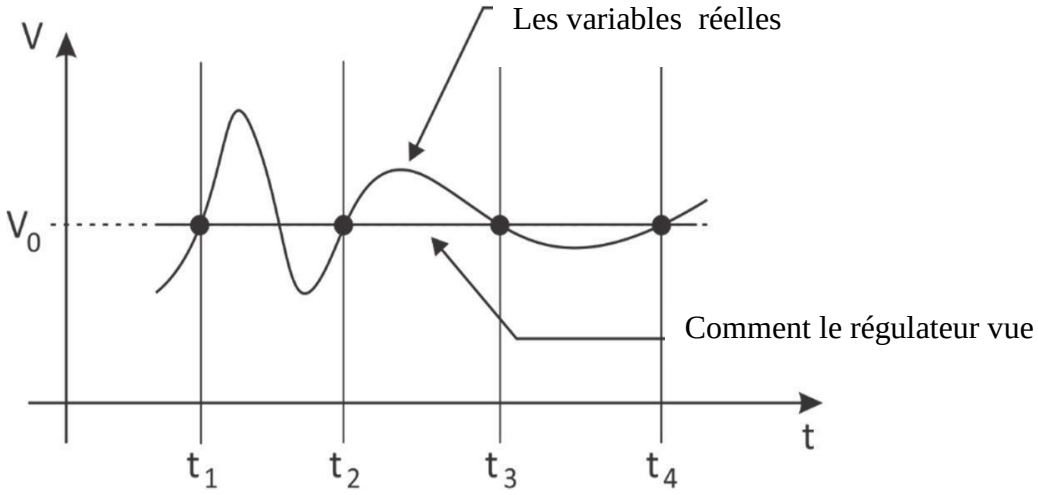

Figure I.6 comparaison entre la variation d'signal analogique et ces enchantions

Dans ce cas, même si la variable échantillonnée change de haut en bas, il semble qu'elle se trouve à niveau constant parce que les instances d'échantillonnage l'attrapent alors qu'elle croise une valeur constante. Le tableau de bord affichant la valeur de V, et ne montrerait donc aucun changement de V, même si V est en fait éprouvant une excursion assez active de V0.

Bien que ce soit un peu un exemple artificiel, c'est facile de voir que la première valeur de crête serait oubliée ou mesurée comme étant inférieure à moins que l'exemple vienne de coïncider avec l'occurrence du pic. De toute évidence, la solution à ce Le problème est de réduire, le taux de balayage du microprocesseur.

7. Les avantages et les inconvénients :

7.1. Les avantages

Il agit uniquement sur l'erreur entre le signal souhaité et le signal commandé. Par conséquent, aucune mesure supplémentaire des états internes n'est nécessaire, ce qui constitue l'un des principaux avantages des applications, car plus de mesures signifie plus de capteurs, plus de conditionnement du signal, plus de maintenance, plus de coût.

Hormis certaines propriétés structurelles de l'installation du processus, il n'est pas nécessaire d'avoir une connaissance approfondie de l'installation. Le réglage peut se faire par essais et erreurs ou par une table de correspondance. Il n'y a pas besoin de beaucoup d'expertise

pour la mise au point, de sorte que peu de techniciens moyennement qualifiés peuvent facilement s'acquitter de cette tâche…

Il est efficace et robuste contre certaines incertitudes courantes s'il est correctement réglé (autour d'une région d'exploitation).

Facile à implémenter dans le matériel (à travers des filtres) et également facile à implémenter via des microcontrôleurs, automates, etc. Aucun code sophistiqué à concevoir… ne peut être écrit par un programmeur de niveau moyen.

7.2. Inconvénients :

Le régulateur n'est pas vraiment adapté aux installations non linéaires en général ou en langage profane, le contrôleur peut ne pas assurer les performances souhaitées pour un environnement des points de fonctionnement changeants. Dans les applications haut de gamme comme les avions de combat, les sous-marins, la robotique de précision, les modèles économiques (prévisions boursières), etc., il s'agit simplement de réaliser une performance stable autour d'une zone d'exploitation fixe. Le régulateur devrait pouvoir suivre un signal de référence dans diverses conditions (même certaines d'entre elles peuvent ne pas être connues).

Dans l'ensemble, le régulateur fonctionne bien pour la plupart des applications industrielles procès mais au contraire pas vraiment adapté aux applications avancées pour la défense, la robotique, les modèles financiers etc.

8. CONCLUSIONS

Dans chaque système de la vie quotidienne, les systèmes de contrôle ont l'essentiel pour gérer leurs différents appareils et environnements. En général, les régulateurs numériques sont utilisés dans presque toutes les applications de l'entreprise avec d'autres périphériques essentiels tels que les microprocesseurs, les ordinateurs et l'électronique numérique. Ensuite, nous pouvons trouver un système de contrôle numérique dans des systèmes d'application électriques, mécaniques ou chimiques. Dans ce chapitre, une étude sur les contrôleurs PID a été introduite pour explorer leurs structures avec le développement nécessaire, que ce soit dans le processus de réglage de contrôleur PID ou sur leurs modifications de structure.

Chapitre II :

Système d’ARDUINO

1. INTRODUCTION

L'Arduino est une plateforme électronique (une carte à microcontrôleur à source ouverte qui se base sur des logiciels et du matériel faciles dans leur utilisation. Arduino peut lire les entrées comme elles peuvent en faire des sorties.

Au paravent, le cerveau penseur de milliers de projets était Arduino, commençant par des objets quotidiens vers les instruments scientifiques complexes. Des milliers de toutes les catégories mondiales : programmation, décideurs… et même des étudiants se sont disposés autour de cette plateforme à code source ouvert. La contribution de la part de cette communauté a permis d'aider les novices et les experts grâce aux multiples connaissances qu'elle a soumis.

Elle a pris sa naissance en Italie à la ville de l'Ivrea (l'Ivrea Interaction Design Institute), un programme de conception supérieure dans le domaine de la conception d'interaction mis en place, considérée comme un simple outil de prototypage rapide pour les étudiants qui n'ont plus de formation en électronique et en programmation. Dès qu'elle a réalisé une large place dans la communauté, la plateforme Arduino s'est évaluée pour s'occuper aux nouveaux besoins, son offre est différente par rapport aux simples cartes 8 bits aux produits pour applications. Toutes les cartes Arduino sont à source ouverte, on peut les construire indépendamment

2. Matériel et logiciel :

Les cartes Arduino sont généralement basées sur des microcontrôleurs Atmel Corporation, tels que des microcontrôleurs AVR 8, 16 ou 32 bits (ATmega328, ATmega32u4 ou ATmega2560). La caractéristique importante des cartes Arduino est les connecteurs standards. À l'aide de ces connecteurs, nous pouvons connecter la carte Arduino à d'autres périphériques tels que des LED ou des add-ons appelés Shields[8].

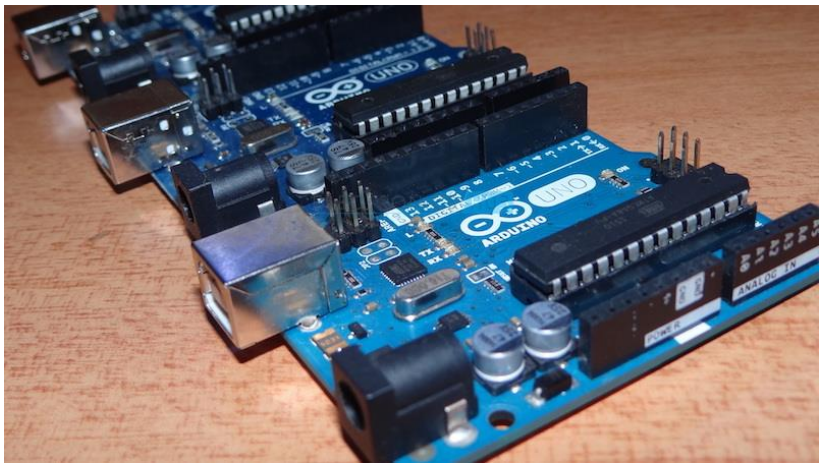

Figure II.1 Arduino

Les cartes Arduino comprennent également un régulateur de tension intégré et un oscillateur à cristal. Ils consistent également en un adaptateur USB série avec lequel la carte Arduino peut être programmée via une connexion USB. Pour programmer la carte Arduino, nous devons utiliser l'IDE fourni par Arduino. L'IDE Arduino est basé sur le langage de programmation Procession et prend en charge les langages C et C ++.

3. Types de cartes Arduino :

Il existe de nombreux types de cartes Arduino disponibles sur le marché, mais toutes les cartes ont un point commun: elles peuvent être programmées à l'aide de l'IDE Arduino. Les raisons pour différents types de cartes sont différentes exigences d'alimentation, options de connectivité, leurs applications, etc. Les cartes Arduino sont disponibles en différentes tailles, formes. Les cartes Arduino connues et fréquemment utilisées sont Arduino UNO, Arduino Mega, Arduino Nano, Arduino Micro et Arduino Lilypad [8] .

3.1. Arduino Uno (R3) :

Le Uno est une énorme option pour l'Arduino initial. Il se compose de 14 broches d'entrées/sorties numériques, où 6 broches peuvent être utilisées en tant que sorties de modulation de largeur d'impulsion (PWM), 6 entrées analogiques, un bouton de réinitialisation, une prise d'alimentation, une connexion USB. Il comprend tout ce qui est nécessaire pour tenir le microcontrôleur en place. Connectez-le simplement à un PC à l'aide d'un câble USB et donnez le matériel nécessaire pour commencer à utiliser un adaptateur CA/CC ou une batterie [8] .

Figure II.2 Arduino UNO

4. Structure d'Arduino :

Nous avons appris qu'il existe de nombreuses variantes pour expérimenter les types Arduino. Maintenant, nous allons en apprendre davantage sur la structure matérielle d'Arduino Uno.

4.1. Structure de l'Arduino Uno :

La principale raison d'utiliser Arduino Uno est que l'ONU est la meilleure carte pour se lancer dans l'électronique et le codage. l'ONU est la carte la plus robuste sur lequel on peut commencer à bricoler. L'ONU est la carte la plus utilisé et documenté de toute la famille Arduino & Génuine.

Venir à l'origine d'Arduino Uno "Uno" signifie un en italien et a été choisi pour marquer la sortie d'Arduino Software (IDE) 1.0. La carte Uno et la version 1.0 du logiciel Arduino (IDE) étaient les versions de référence d'Arduino, qui ont maintenant évolué vers de nouvelles versions. La carte Uno est la première d'une série de cartes USB Arduino et le modèle de référence pour la plate-forme Arduino.

5. Schématique de la carte Arduino:

A Partir du schéma ci-dessus, nous pouvons obtenir l'aperçu schématique de la conception des cartes Arduino. Et la photo suivante spécifie clairement le schéma des broches d'Atmega168 IC, qui est l'essentiel pour contrôler le fonctionnement de l'Arduino Uno Board. Voir le mappage entre les broches Arduino et les ports ATmega328P. La cartographie des Atmega8, 168 et 328 est identique [10].

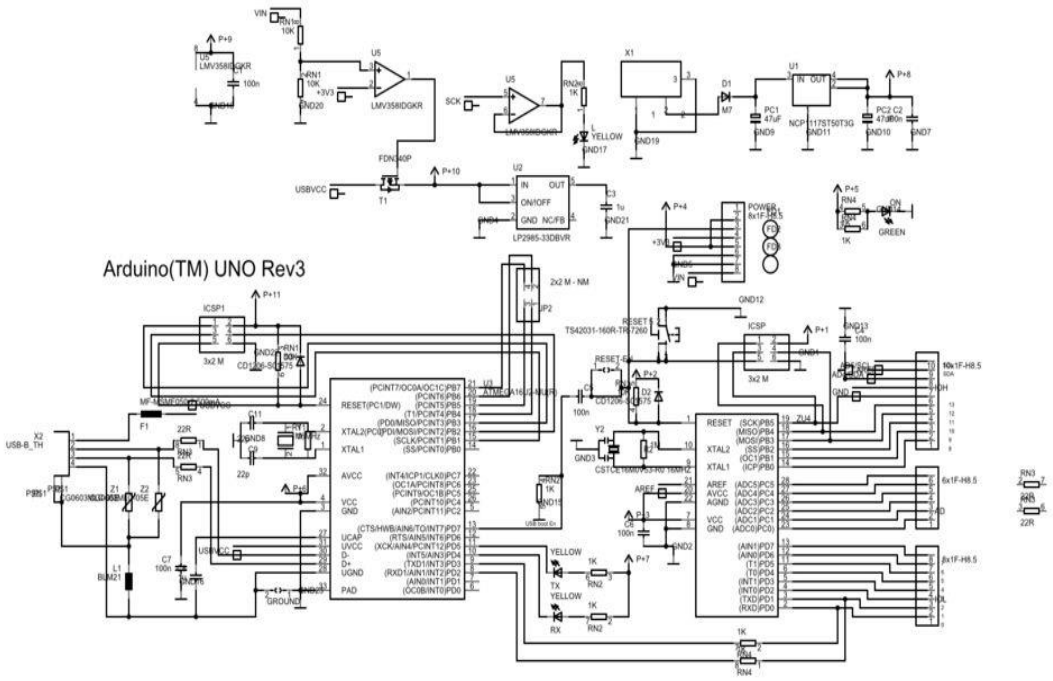

Figure II.3 schématique de la carte Arduino

5.1. Entrées et Sorties :

Chacune des 14 broches numériques de l'Uno peut être utilisée comme entrée ou sortie à l'aide des fonctions pinMode(), analogWrite(), analogread(), digitalWrite () et digitalRead (). Ils fonctionnent à 5 volts. Chaque broche peut fournir ou recevoir 20 mA dans les conditions de fonctionnement recommandées et possède une résistance de tirage interne (déconnectée par défaut) de 20 à 50 000 ohms. Un maximum de 40 mA est la valeur qui ne doit pas être dépassée sur une broche d'entrée / sortie pour éviter des dommages irréversibles au microcontrôleur. De plus, certaines broches ont des fonctions spécialisées : Série : 0 (RX) et 1 (TX). Utilisé pour recevoir (RX) et transmettre (TX) des données sérient TTL. Ces broches sont connectées aux broches correspondantes de la puce série ATmega8U2 USB-to-TTL. Interruptions externes : 2 et 3.

Ces broches peuvent être configurées pour déclencher une interruption sur une valeur basse, un front montant ou descendant ou une modification de la tension. Valeur. Voir la fonction attachInterrupt () pour plus de détails. PWM: 3, 5, 6, 9, 10 et 11. Fournissez une sortie PWM 8 bits avec la fonction analogWrite ().SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Ces broches prennent en charge la communication SPI à l'aide de la bibliothèque SPI.LED: 13. Une DEL intégrée est pilotée par la broche numérique 13.

Lorsque la broche a la valeur HIGH, elle est allumée, lorsque la broche est LOW, elle est éteinte. Le Uno a 6 entrées analogiques, appelées A0 à A5, chacune offrant une résolution de 10 bits (soit 1024 valeurs différentes). Par défaut, ils mesurent la masse à 5 volts, mais il est possible de modifier l'extrémité supérieure de leur plage à l'aide de la broche AREF et de la fonction analogReference (). Il existe deux autres broches sur la carte: AREF. Tension de référence pour les entrées analogiques. Utilisé avec analogReference ()[10].

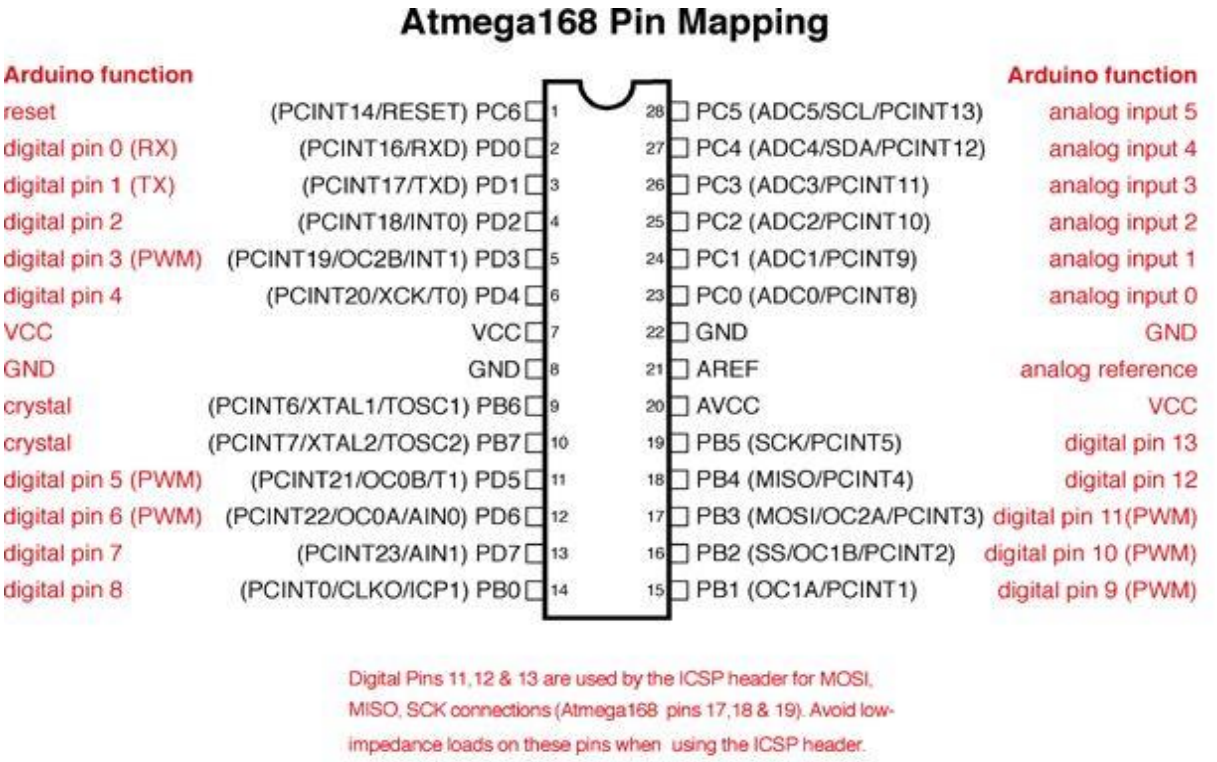

Figure II.4 Entrées et Sorties de microcontrôleur (Arduino)

6. Programmation de l'Arduino :

En ce qui concerne sa partie programmation, la principale chose que nous devons avoir en tête est le langage de programmation que nous allons utiliser pour la programmation.

Le langage Arduino est simplement un ensemble de fonctions C/C++ pouvant être appelées à partir d'un code. L'esquisse subit des modifications mineures (par exemple, la génération automatique de prototypes de fonctions) et est ensuite transmis directement à un compilateur C/C++. Et pour le compilateur, voici le nouveau terme appelé Carte de développement intégré Arduino (Arduino IDE). L'Arduino / Genuino Uno peut être programmé

avec le (logiciel Arduino (IDE)). Sélectionnez "Arduino / Genuino Uno dans le menu Outils> Carte (en fonction du microcontrôleur de la carte utiliser).

L'ATmega328 de l'Arduino / Genuino Uno est préprogrammé avec un chargeur de démarrage qui nous permet de télécharger du nouveau code sans l'aide d'un programmateur matériel externe.

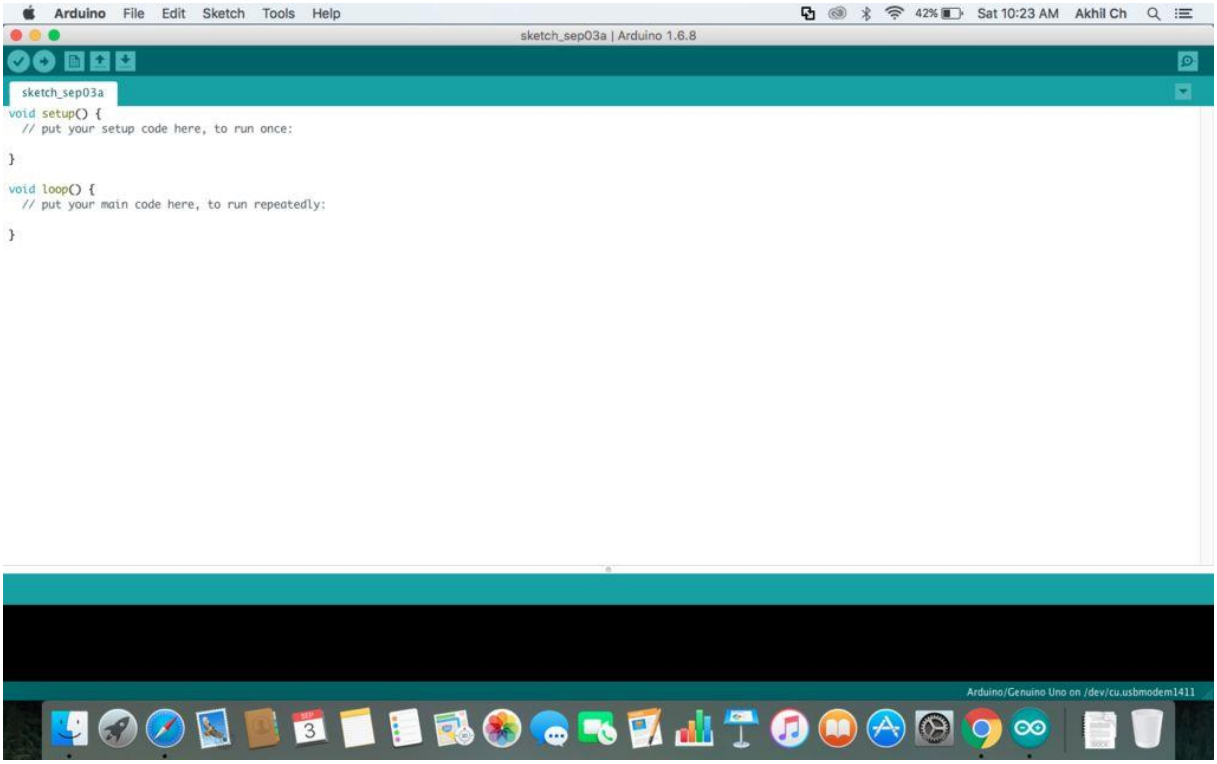

Figure II.5 exemple de démarrage de program Arduino

6.1. Structure d'un programme Arduino :

//Déclaration des variables ;

ex : **int led = 13**; *// fixer la variable led à 13 byte; entier prendre des valeurs de 0 à 255*

int integer1 ; // *entier prendre des valeurs de* -32,768 à 32,767

unsigned int uInteger1 ; *// prendre des valeurs de 0 à 65,535*

float mesure1 ; *// reel*

boolean flag=true; *// booleen true ou false*

Int tableauInt[6] ; *//tableau d'entiers a 6 entrées, numérotées de 0 à 5*

Initialisation des ports utilisés de la carte et initialisation du port série

void setup () { // cette séquence est réalisée une seule fois, au début

Serial. Begin(9600); // *initialise le port série*

PinMode (3, OUTPUT); // *initialise la voie 3 comme sortie*

PinMode (4, INPUT); // *initialise la voie 4 comme entrée*

PinMode (led, INPUT); // *si led est une constante de valeur 13, initialise la voie 13 comme entrée*

} //**fin de la procédure setup ()**

// Programme principal : une boucle qui tourne à l'infini :

void loop () {*instructions du programme à mettre ici ;*

}

6.1.1. Imposer des tensions aux sorties de la carte (Ecrire) :

digitalWrite(3, HIGH); // *impose la valeur HIGH au port 3 défini comme OUTPUT*

(*autrement dit, envoie 5 V à la sortie du port D3). On peut mettre LOW et dans ce cas, c'est 0 V).*

analogWrite(6,100) ; *// impose la valeur 100 au port D6 défini comme OUTPUT**(il s'agit d'une sortie analogique PWM). Cette valeur est définie entre 0 et 255 (255 vaut 5 V et 0 vaut 0 V donc 100 correspondra ici à une tension moyenne en sortie de D6 de 1,96 V)*.

6.1.2. Lire des tensions aux entrées de la carte :

buttonState = digitalRead(4); // *lit la valeur HIGH ou LOW du port D4 défini comme Input **(ici buttonState a été défini comme boolean)***

val = analogRead(5); *// lit la valeur analogique au port 5 défini comme INPUT **(valeur entière comprise entre 0 et 1023)***

Affecter une valeur à un tableau

readings[3] = 1 ;

Transformation d'un entier en réel ;

nombrereel = float(nombreentier) ;

6.1.3 Exemples de calculs simples :

Dans ce cas il faut faire attention aux types réel ou entier :

Vmes= average*Vref/1024;

temp = (Rmes/100-1)/3.85e-3;

Attente un milliseconde:

delay(1);

6.1.4 Mesure du temps écoulé depuis le lancement du programme en millisecondes:

time = millis(); *(définir time comme unsigned long)*

6.1.5 Envoi sur port série de chaine de caractère :

Serial.print("Mesure : ");

Serial.print(variable) ; *// ça envoie la valeur de variable*

Serial.println(" toto "); *// Envoi d'une chaine avec retour à la ligne (car « println » et pas « print » indispensable en fin de ligne:*

6.1.6 Une boucle for (ici de 0 à 99) :

for (int index=0; index < 100; index++){

instructions de la boucle à executer ;

}

for (int x = 2; x < 100; x = x * 1.5){

instructions de la boucle à executer ;

}

une boucle **while**

(ici qui s'execute tant que index est <3, et à chaque tour on ajoute 1)

while (index< 3) {

index = index + 1 ;

}

do {

instruction à faire;

}

une condition if (ici par exemple sur la valeur d'un boolean buttonState:

if (buttonState == HIGH) {

instruction à écrire ici si buttonState vaut HIGH ;

}

else {

instruction à écrire ici si buttonState vaut LOW ;

}

7. Les tensions de référence :

La carte Arduino fournit des ports permettant d'accéder à certaines tensions de référence. GND est la référence de la carte Arduino par rapport à laquelle toutes les différences de tension sont mesurées. Si la carte est reliée à l'ordinateur par un câble USB, cette tension est celle de la terre.

Les ports 5V et 3V3 donnent accès aux tensions de 5 V et de 3.3 V. Ces tensions sont normalement régulées et précises. Une exception : quand la carte est branchée sur un port USB sans alimentation externe, le port 5 V ne provient plus de la carte Arduino mais directement du câble USB, la tension de référence 5 V n'est alors plus aussi bien régulée. VIN est la tension de l'alimentation externe, quand il y en a une. Attention : si on relie directement le port 5 V au port GND (ou le port 3V3 au port GND, ou le port 5V au port 3V3), on provoque un court-circuit qui endommagera la carte !

8. Les avantages et les inconvénients de la carte Arduino :

Au cours des dernières années, l'utilisation d'Arduino a augmenté de façon exponentielle en raison de sa lisibilité et de sa facilité. L'utilisation d'Arduino présente certains avantages et inconvénients [7].

8.1 Les avantage :

a. Prêt à l'emploi : Le plus gros avantage d'Arduino est sa structure prête à l'emploi. Comme Arduino est livré dans un emballage complet qui comprend le régulateur 5V, un brûleur, un oscillateur, un microcontrôleur, une interface de communication série, une LED et des en-têtes pour les connexions. On n'a pas à penser aux connexions de programmeurs pour la programmation ou toute autre interface. Il suffit de le brancher sur le port USB de l'ordinateur et le tour est joué. L'idée révolutionnaire va changer le monde après quelques mots de codage[7].

b. Exemples de codes : Un autre grand avantage d'Arduino est sa bibliothèque d'exemples présents dans le logiciel Arduino. Pour expliquer cet avantage on présente l'exemple de mesure

de tension suivant. Par exemple, si on souhaite mesurer la tension à l'aide du microcontrôleur ATmega8 et que nous souhaite afficher la sortie sur un écran d'ordinateur, nous devenons suivre l'ensemble du processus. Le processus commencera par l'apprentissage des CAN du microcontrôleur pour la mesure, passera par l'apprentissage de la communication série pour l'affichage et se terminera par les convertisseurs USB - série.

D'autre part, si nous voulons mesurer la tension en utilisant Arduino. Il nous suffit de brancher l'Arduino et d'ouvrir l'exemple ***ReadAnalogVoltage*** comme illustré à la figure II.6 [7].

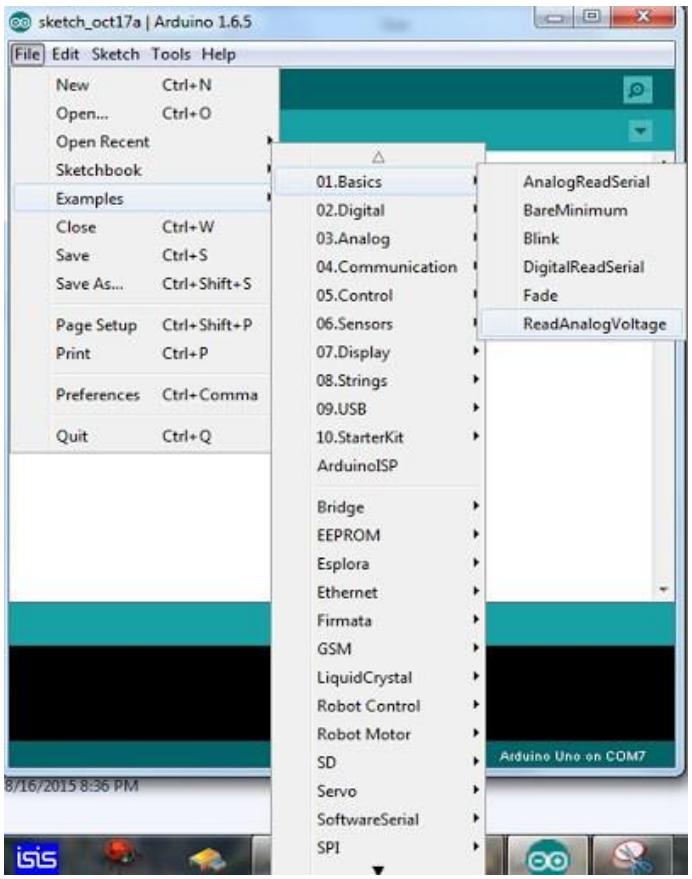

Figure II.6 exemple de code Arduino

c. Fonctions sans effort : Pendant le codage d'Arduino, nous remarquerons certaines fonctions qui facilitent la vie. Un autre avantage d'Arduino est sa capacité de conversion automatique d'unités. Nous pouvons dire que pendant le débogage, nous n'avons pas des conversions d'unités. Utilisez simplement vos efforts sur les parties principales de vos projets. Nous n'avons pas à nous soucier des problèmes secondaires.

d. Grande communauté : Il existe de nombreux forums présents sur Internet dans lesquels les gens parlent de l'Arduino. Ingénieurs, amateurs et professionnels font leurs projets à travers

Arduino. Nous pouvons facilement trouver de l'aide sur tout. De plus, le site Web Arduino lui-même explique toutes les fonctions d'Arduino.

Nous devrions donc conclure l'avantage d'Arduino en disant que lorsqu'on travaille sur différents projets, il suffit de s'inquiéter de son idée novatrice. Le reste sera géré par l'Arduino lui-même.

8.2. Les inconvénients

a. La structure : La structure d'Arduino est aussi son inconvénient. Lors de la construction d'un projet, nous devons réduire autant que possible sa taille. Mais avec les grandes structures d'Arduino, nous devons nous en tenir à des circuits imprimés de grande taille. Si nous travaillons sur un petit microcontrôleur comme ATmega8, nous pouvons facilement réduire la taille de la carte de circuit imprimé [7].

b. Coût : Le facteur le plus important qu'on ne peut pas nier est le coût. C'est le problème auquel tout amateur, ingénieur ou professionnel doit faire face. Maintenant, nous devons considérer que l'Arduino est rentable ou non.

c. Facile à utiliser : on peut remarquer qui, si nous commençons notre expérience avec Arduino, il nous sera très difficile de réaliser les circuits intelligents complexes à l'avenir. Le matériel / logiciel facile à utiliser d'Arduino ne permet pas à une personne d'apprendre les bases de nombreuses choses comme les communications série, ADC, I2C, etc.

Conclusion

Arduino est un outil de programmation qui nous permet de mettre des programmes de circuits électroniques pour obtenir des interactions avec des composants extérieurs (on obtient comme exemple les moteurs, les capteurs...).

Généralement, Arduino est utilisé pour la domotique et la robotique. Ses multiples entrées et sorties nous permettent de faire plusieurs choses à faible coût. Il donne un ensemble très large de cartes, chaque carte a une taille différente que l'autre avec plus ou moins d'entrées et de sorties. On peut rajouter aussi plusieurs modules sur chaque carte pour rajouter des fonctionnalités et des rôles, prenant par exemple le Bluetooth, le wifi...

CHAP III

Réalisation du régulateur PID numérique

1. Introduc²tion :

Un contrôleur de température est un dispositif utilisé pour maintenir une température souhaitée à une valeur spécifiée.

L'exemple le plus simple d'un contrôleur de température est un thermostat commun trouvé dans les maisons. Par exemple, un chauffe-eau utilise un thermostat pour contrôler la température de l'eau et la maintenir à une certaine température commandée. Les régulateurs de température sont également utilisés dans les fours. Lorsqu'une température est définie pour un four, un contrôleur surveille la température réelle à l'intérieur du four.

S'il tombe en dessous de la température définie, il envoie un signal pour activer le chauffage afin de ramener la température au point de consigne. Les thermostats sont également utilisés dans les réfrigérateurs. Donc, si la température devient trop élevée, un contrôleur déclenche une action pour abaisser la température.

Dans cette partie nous proposons un régulateur PID numérique pour la régulation de la température d'une plaque chauffante, en utilisant la carte Arduino avec le microcontrôleur ATMEga 328P.

2. Schéma général de régulateur PID :

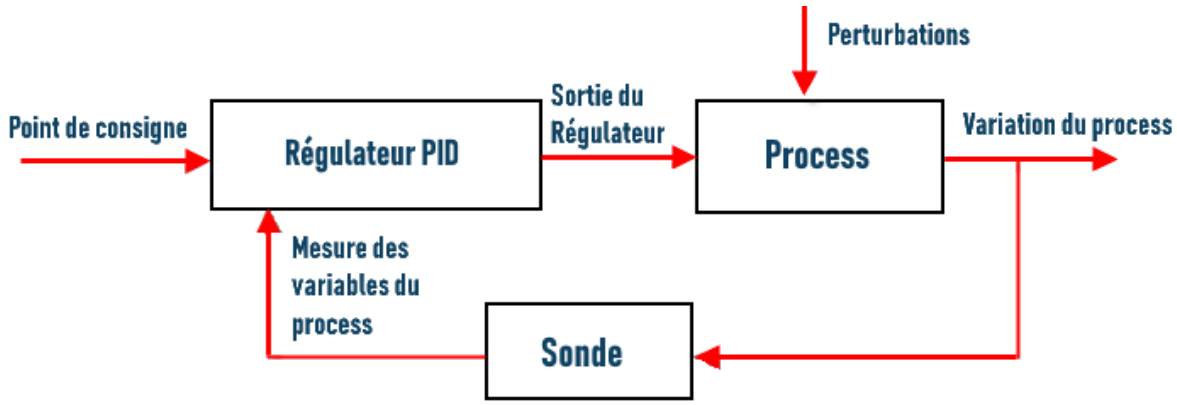

Figure III.1 schéma de fonctionnement de régulateur PID

Dans ce projet, on va régler la température en utilisant le PID numérique dont l'utilisation de cette dernière nous donne un haut degré de flexibilité lors de son utilisation.

Afin de maintenir un degré de température constant pendant une période de temps spécifiée, nous établions une routine PID pour le réglage de la température. Ça pourrait être

efficace pour la cuisson des aliments par exemple ou tout ce qui demande un contrôle de la température

La description du régulateur PID ainsi que son structure, composition et réglage sont décrit dans le chapitre 01. En générale, notre régulateur de température prend des mesures de l'état actuel du système en utilisant une sonde de température a coefficient négatif CTN, puis utilise ces données pour déterminer s'il faut maintenir l'élément chauffant ou non

C'est pourquoi dans notre projet on a abordé 3 étapes nécessaires et évidentes :

- mesurer la température
- Calculer les valeurs nécessaires pour activer l'action
- conception d'un système actionnaire qui va faire le chauffage selon le modèle représenté dans le schéma suivant :

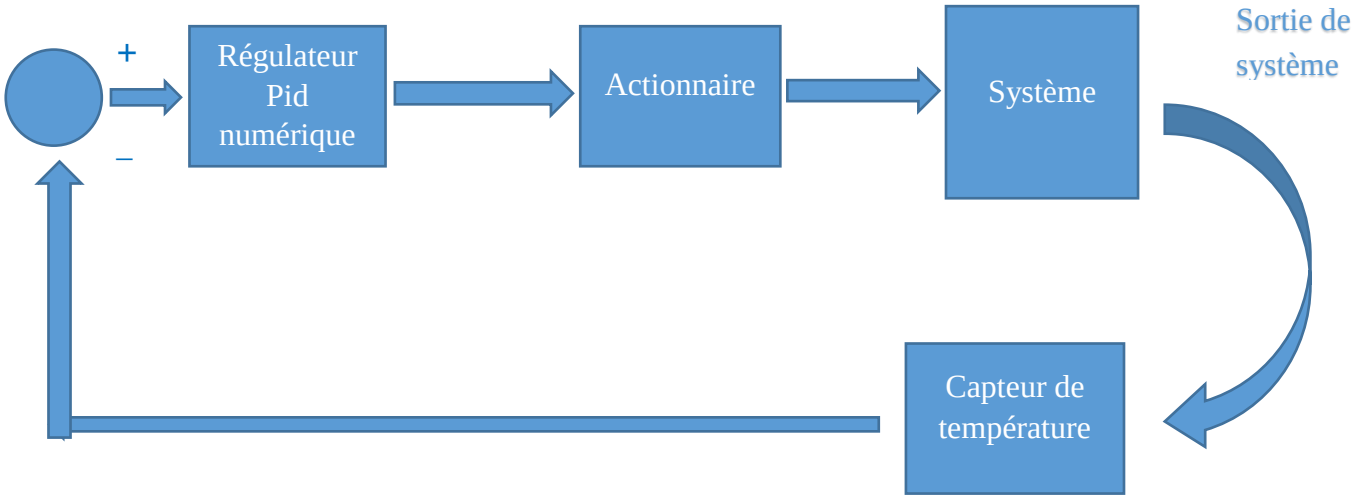

Figure III.2 schéma de fonctionnement de régulateur PID numérique de température

2.1. Mesures de température

A partir du schéma précédant, on passe à la première étape celle de mesurer la température de la résistance de chauffage et pour mesure sa température, en utilisant le thermomètre par Arduino comme suit :

2.1.1. Le capteur de température CTN :

Dans ce projet nous avant utiliser une thermistance de type CTN. En général, Les thermistances sont des résistances variables qui changent leur résistance avec la température. Ils sont classés en fonction de la façon dont leur résistance répond aux changements de température. Dans les thermistances à coefficient de température négatif (CTN), la résistance

diminue avec l'augmentation de la température. Les thermistances CTN sont les plus courantes et c'est le type que nous allons utiliser dans ce projet. Les thermistances CTN sont fabriquées à partir d'un matériau semi-conducteur (tel qu'un oxyde de métal ou une céramique) qui a été chauffé et comprimé pour former un matériau conducteur sensible à la température.

Figure III.3 capteur CTN

2.1.2. Fonctionnement :

Construisons un circuit de thermistance de base pour voir comment cela fonctionne. Vous pourrez ainsi l'appliquer ultérieurement à d'autres projets. La thermistance étant une résistance variable, nous devons mesurer la résistance avant de pouvoir calculer la température. Cependant, l'Arduino ne peut pas mesurer directement la résistance, il ne peut mesurer que la tension

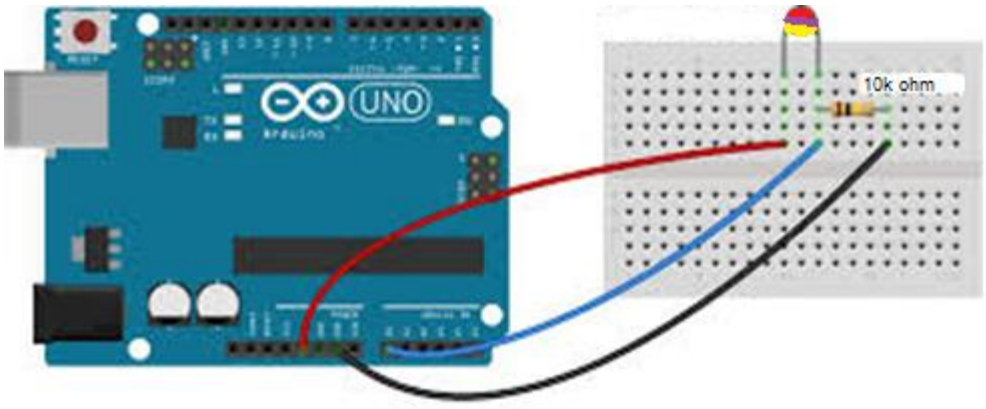

Figure III.4 branchement de capteur CTN

L'Arduino mesurera la tension en un point situé entre la thermistance et une résistance connue. Ceci est connu comme un diviseur de tension. L'équation pour un diviseur de tension est la suivante :

$$v_{out} = v_{in} \times (\frac{R2}{R1 + R2})$$

Cette équation peut être réorganisée et simplifiée pour résoudre pour R2, la résistance de la thermistance :

$$R2 = R1 \times (\frac{V_{IN}}{v_{out}} - 1)$$

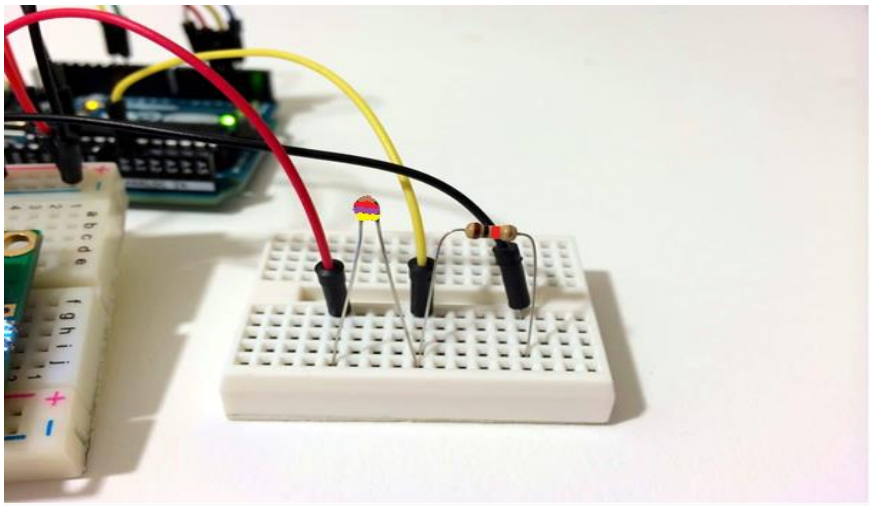

Figure III.5 fonctionnement de capteur CTN

Enfin, l'équation de Steinhart-Hart est utilisée pour convertir la résistance de la thermistance en lecture de température. Une version de celui-ci est montrée ci-dessous (d'autres versions utilisent le terme carré ainsi que le terme cube) :

$$\frac{1}{T(kelvin)} = A + Bln(R) + C(\ln(R))^3$$

Où R est la résistance de la thermistance à la température T (en kelvins). Il s'agit d'une équation d'ajustement de courbe générale destinée à toutes les résistances de type CTN.

L’approximation de la relation de température et de résistance est «assez bonne» pour la plupart des applications.

Notez que l'équation nécessite les constantes A, B et C. Celles-ci sont différentes pour chaque thermistance et doivent être données ou calculées. Comme il y a trois inconnues
Ces constantes peuvent être identifiées en utilisant la Datasheet du constructeur et dans notre cas les valeurs que nous avons utilisées dans notre projet sont :

- A= 3.354016E-03ss
- B= 2.569850E-04
- C= 6.383091E-08

Après l’implémentation de circuit et le programme nous avons test ce dernier pour vérifier que notre Project est capable de mesurer la valeur réel de la température, et pour mieux indiquer la température nous l’avons affiché sur le LCD.

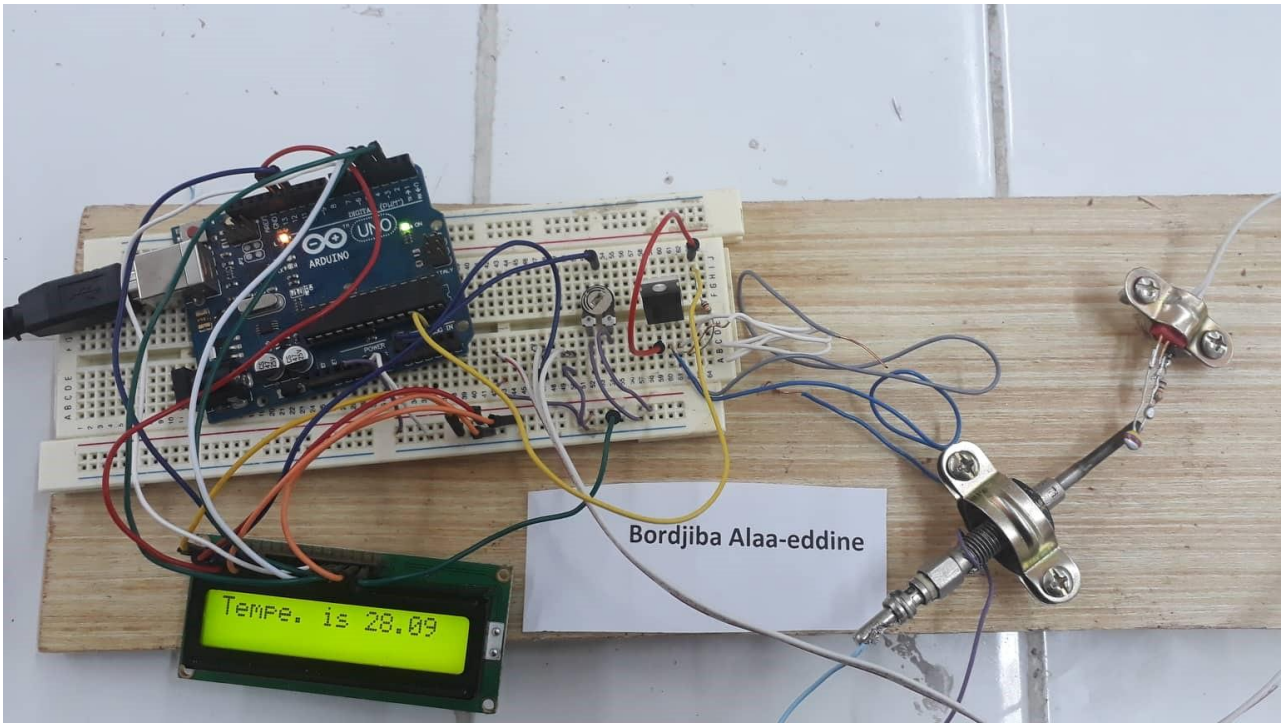

Figure III.6 Mesure et affichage de la température ambiante

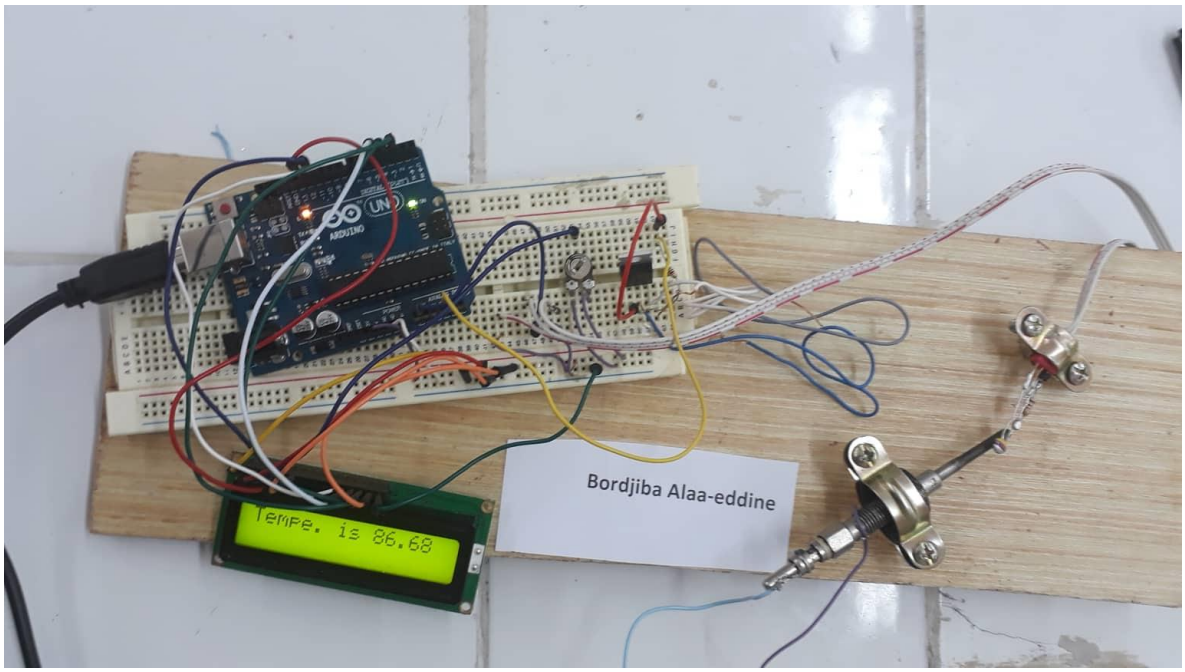

Figure III.7 Exemple de la mesure et l'affichage de la température du système après une excitation quelconque

2.1.3. Branchement de l'afficheur LCD :

Voici un schéma des broches de l'écran LCD que j'utilise. Les connexions de chaque broche à l'Arduino seront les mêmes[9] , mais vos broches pourraient être disposées différemment sur l'écran LCD

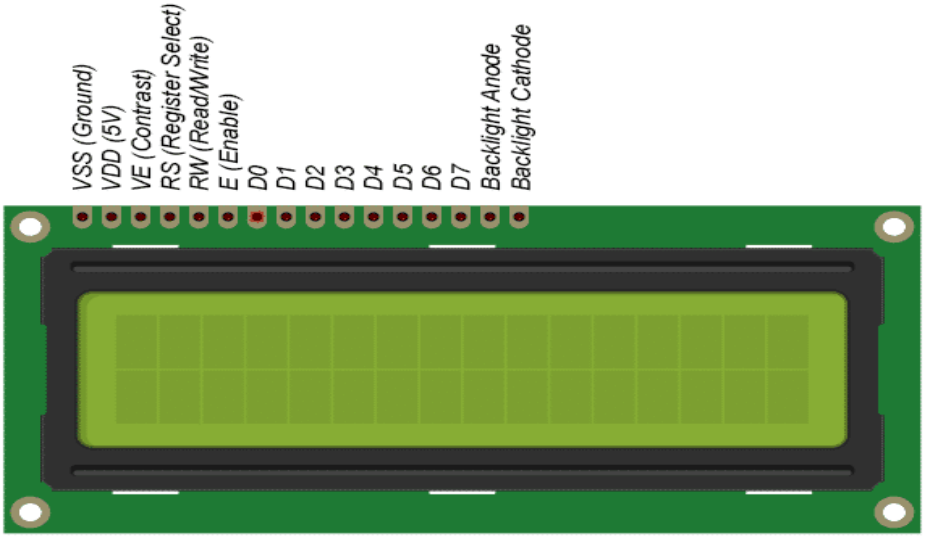

Figure III.8 afficheur LCD 16*2

En outre, nous devrons peut-être souder un en-tête à 16 broches à l'écran LCD avant de le connecter à une plaque d'essai. Suivant le schéma ci-dessous pour connecter l'écran LCD à l'Arduino [9]:

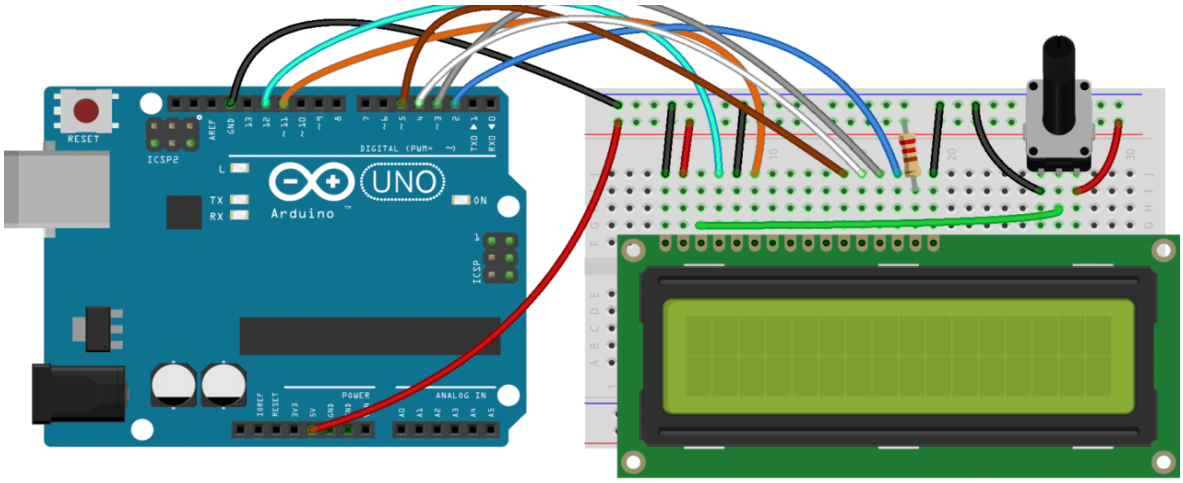

Figure III.9 branchement de l'afficheur LCD

2.2. Le Calcule de régulateur PID

Après cette étape que nous l'avons annoncé précédemment, on passe à l'étape suivante celle du calcul des valeurs nécessaires pour activer l'action, alors nous avons implémenté le PID numérique que l'on a expliqué précédemment. Les trois actions de ce contrôleur sont :

- **Action Proportionnel :** $P = Kp \cdot e(t)$
- **Action Intégral :** $I = Ki \int_0^t e(\tau) d\tau$
- **Action Dérivée :** $D = Kd \frac{d}{dt} e(\tau)$

Nous avons implémentés ces trois action dans un algorithme puis les injectés dans l'Arduino.

Pour les actions intégral et dérivation, nous avons utilisé les algorithmes de calcule numérique pour la réalisation de ces action selon l'organigramme suivant

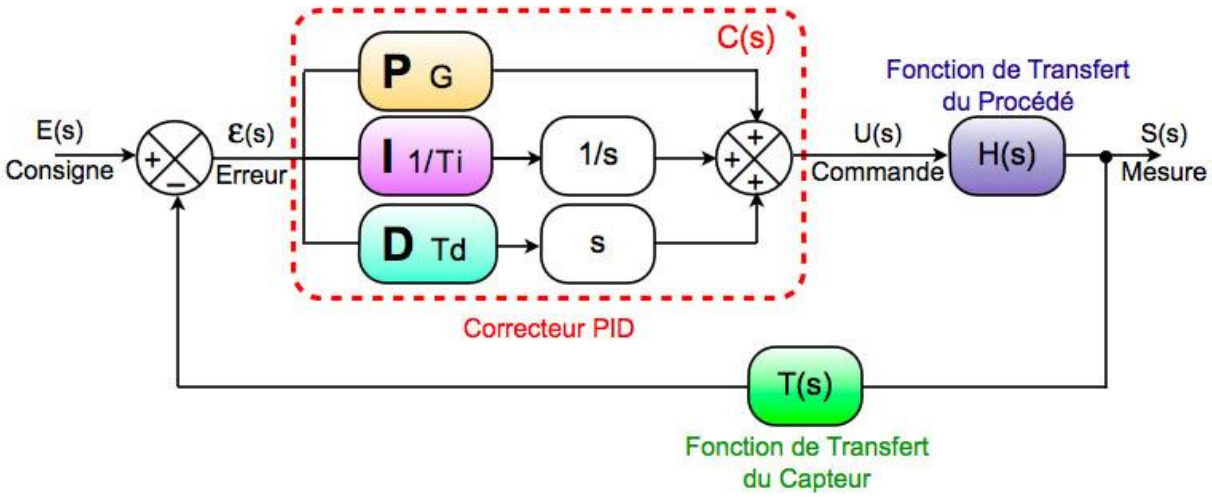

Figure III.10 schéma de calcule de régulateur PID

Il est important de mentionner que nous avons utilisé la structure parallèle de PID.
Elle est donnée par l'équation suivante :

$$u(t) = kp\,.\,e(t) + \ Ki\int_0^t e(\tau)d\tau + Kd\frac{d}{dt}e(\tau)$$

Afin de compléter la réalisation de la régulation on passe à l'étape suivante qui est la détermination des trois paramètres de notre régulateur (les constants KP KI KD).

2.2.1. Calcule des paramètres du Régulateur :

Pour calculer les paramètres de notre Régulateur, nous avons utilisé la méthode de Ziegler-Nichol en utilisant la réponse du système en boucle ouvert.
Le principe donc, est d'exciter le système avec un échelon, puis tracer la réponse du système. Nous enregistrer la température du système utilisant la sonde CTN.

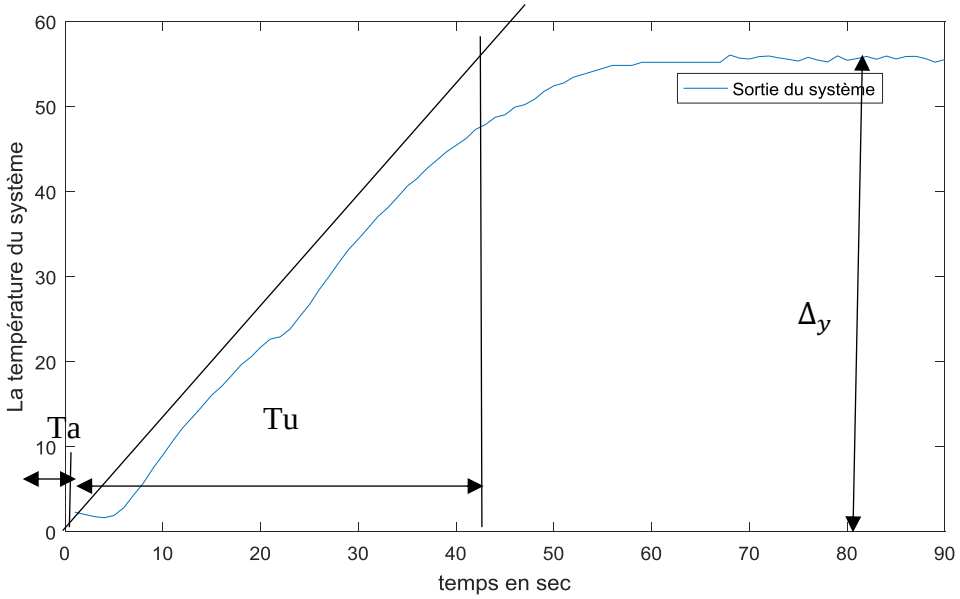

Figure III.11 Réponse indicielle du système en boucle ouvert

Le principe : tracer la pente au point d'inflexion, puis déterminer les paramètres Ta, Tu et K,

Réglage des paramètres			
Régulateur	Kp	Ti	Td
PID: $R(p) = K_p\,(1+ T_d\,p + \frac{1}{T_i\,p}\ )$	$\frac{\text{Ta } 1.2}{\text{Tu . K}}$	$\frac{2.0}{\text{Tu}}$	$\frac{0.5}{\text{Tu}}$

Tableaux III.1 réglage du paramètre de régulateur PID

2.3. Conception d'un système actionnaire pour l'excitation

Après la conception du contrôleur PID et son activation pour faire les calculs nécessaires pour activer l'action, on doit faire maintenant la conception d'un système actionnaire qui va faire le chauffage selon la température prévue, En utilisant dans ce dernier le MOSFET IRFZ44N.

2.3.1. MOSFET IRFZ44N :

L'IRFZ44N est un MOSFET à canal N avec un courant de drain élevé de 49A et une valeur de faible R_{DS} de 17,5 mΩ. Il a également une tension de seuil basse de 4 V à laquelle le MOSFET commencera à conduire. Par conséquent, il est couramment utilisé avec des

microcontrôleurs pour conduire avec 5V. Cependant, un circuit de commande est nécessaire si le MOSFET doit être complètement connecté [11].

Figure III.12 le MOSFET IRFZ44N

Le nombre de pin	Le nom de pin	La description
1	Source	Le courant sort par la source
2	Gate	Contrôle la polarisation du MOSFET
3	Drain	Le courant entre par le drain

Tableau III.2 description des pats de MOSFET IRFZ44N

2.3.2. Caractéristiques :

- MOSFET à canal N à petit signal Le courant de drain continu (ID) est de 49A à 25 ° C
- Le courant de drain pulsé (ID-pic) est de 160A
- La tension de seuil minimale de la porte (VGS-th) est de 2V
- La tension de seuil maximale de la porte (VGS-th) est de 4V
- La tension source-porte est (VGS) est ± 20V (max)
- La tension maximale de drain-source (VDS) est de 55V
- Le temps de montée et le temps de chute sont d'environ 60ns et 45ns respectivement.
- Il est couramment utilisé avec Arduino, en raison de son faible courant de seuil. Disponible en paquet To-220

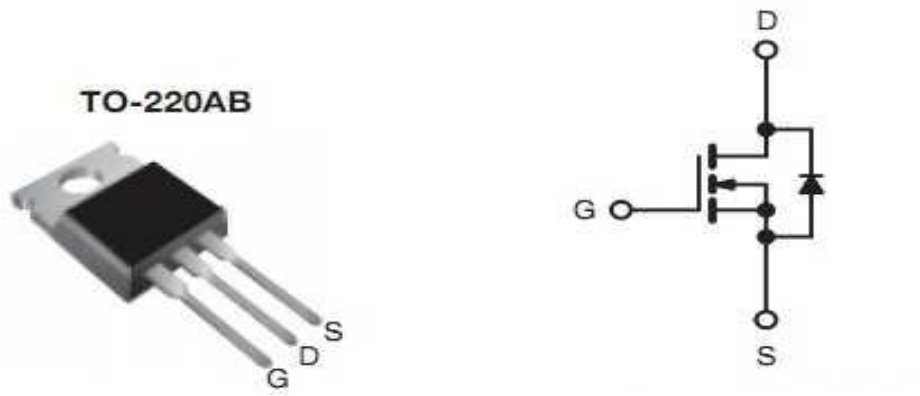

Figure III.13 schéma intérieure de L'IRFZ44N

L'IRFZ44N est connu pour son courant de drain élevé et sa vitesse de commutation rapide. En plus de cela, il a également une faible valeur Rds qui aidera à augmenter l'efficacité des circuits de commutation. Le MOSFET commencera à s'allumer avec une petite tension de grille de 4V, mais le courant de drain ne sera maximum que si une tension de grille de 10V est appliquée. Si le mosfet doit être piloté directement à partir d'un microcontrôleur comme Arduino, essayez alors le mosfet de version logique IRLZ44N [11].

2.3.3. Utilisation :

Et en utilise ce MOSFET IRFZ44N dans notre projet comme le schéma ci-dessus :

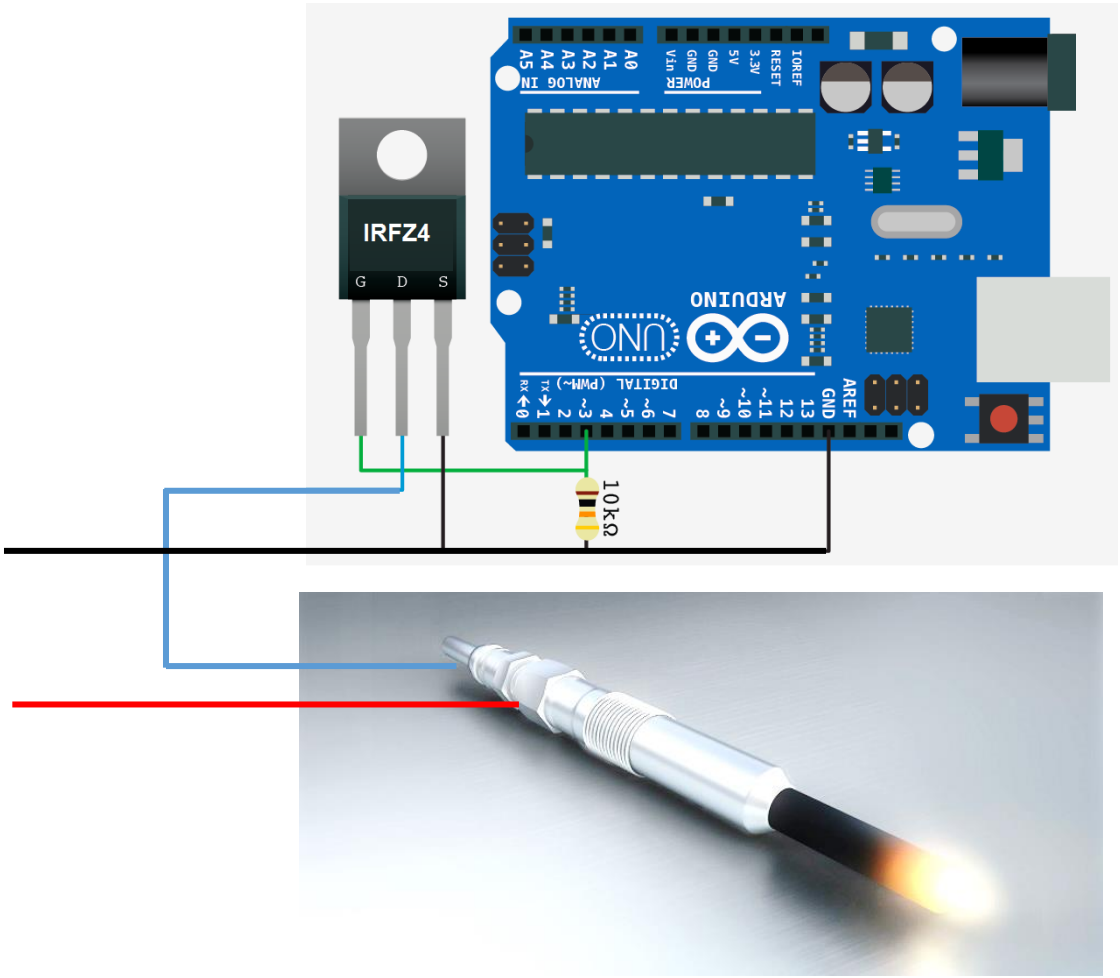

Figure III. 14 le branchage de mosfet IRFZ44N dans notre projet

Le circuit final de notre projet est illustré dans la figure suivante :

- Le composant :
  - La carte Arduino Uno
  - Afficheur LCD
  - Bougie de chauffage
  - Capteur CTN
  - MOSFET IRFZ44N
  - Résistance 10 k
  - Potentiomètre 10 k

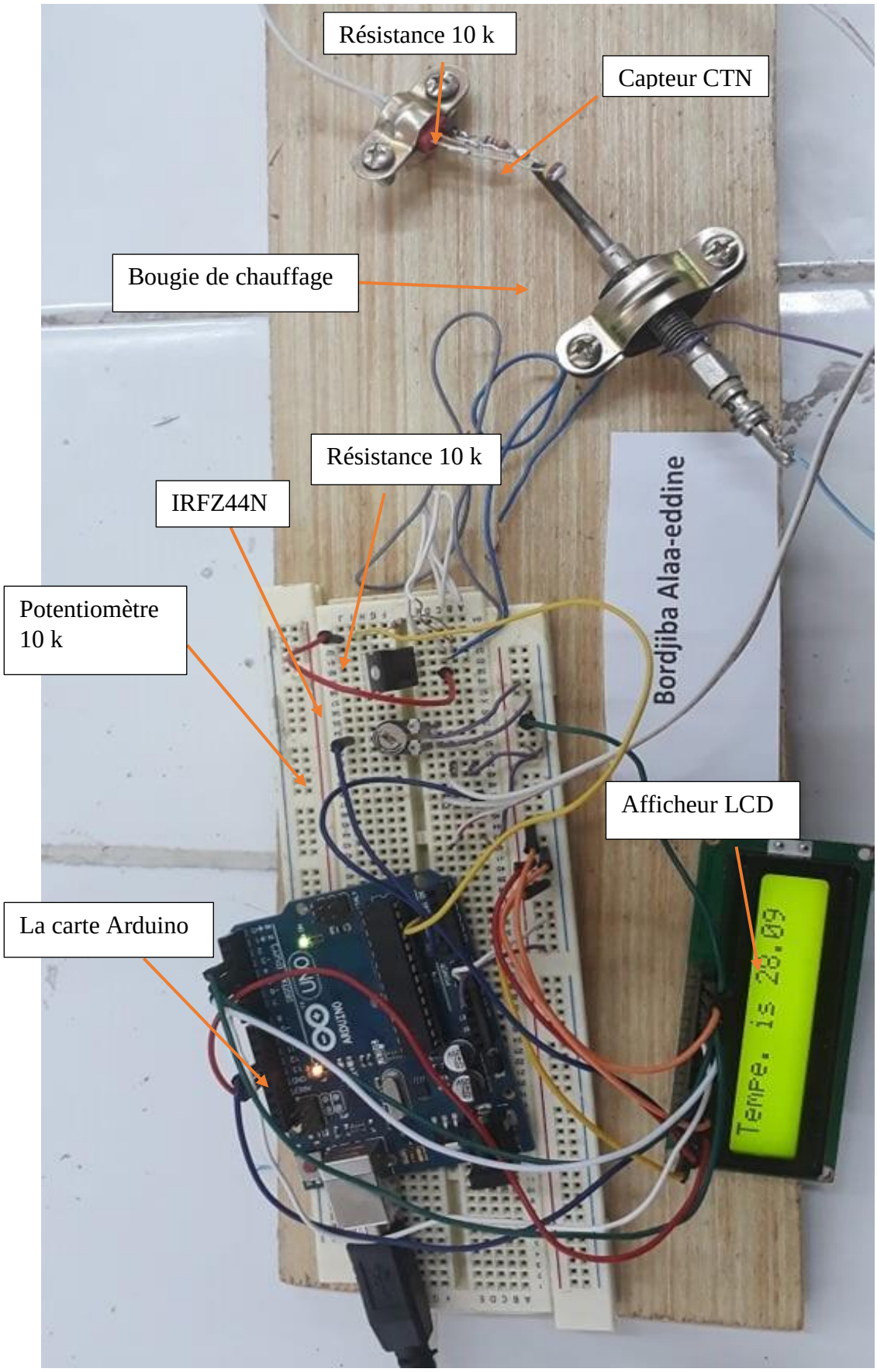

Figure III .15 Représente le circuit final de notre projet

Conclusion :

Nous avons récemment configuré le régulateur PID numérique avec une carte Arduino pour contrôler les réglages de température. L'objectif est de maintenir une température constante pendant une certaine période de gradient, de la maintenir pendant une période de trempage spécifique, puis de la refroidir à une certaine vitesse.

Cela peut être utile pour le soudage, la cuisson, la préparation ou tout ce qui nécessite une minuterie pour contrôler la température.

Conclusion générale

De nos jours, l'asservissement par le régulateur PID est très utilisé dans l'industrie car, ce dernier nous permet de contrôler un grand nombre de procédés ainsi d'effectuer une régulation en boucle fermée d'un système industriel.

C'est le régulateur le plus répandu compte tenu de sa simplicité. Notre premier objectif est de montrer le rôle principal du PID de satisfaire les spécifications de performances qui traduisent les performances relatives à la stabilité, à la précision et à la rapidité Nous avons abordé dans le premier chapitre la structure détaillée du régulateur PID ainsi que son réglage. Nous avons aussi abordé le régulateur PID numérique, sa structure et les problèmes rencontrés avec ce dernier.

Le second chapitre est consacré à la présentation de la carte Arduino qui a le rôle de stocker le programme et de le faire fonctionner avec ses types, sa structure ainsi que sa programmation, les tensions de références, les avantages et les inconvénients.

L'application sur tout ce qu'on a abordé dans les deux premiers chapitres feront l'objectif du troisième chapitre. Nous avons appliqué le régulateur PID de température avec Arduino en présentant un schéma général de régulateur PID, les mesures de température, les calculs de régulateur PID ainsi que la conception d'un système actionnaire qui va faire le chauffage.

Nous avons réalisé successivement notre projet après plusieurs essais et épreuves afin d'obtenir un régulateur PID de température qui permet d'avoir une mesure de température désirée et indiquée, de régler à l'aide de ses trois paramètres, les performances d'un processus modélisé par un deuxième ordre avec une carte Arduino qui le rend facile à utiliser.

Bibliographies

[1] https://www.abcclim.net/regulation-p-pi-pid.html

[2] Benabdi Sofiane & Saoudi Abd Erraouf MASTER optimisation d'un régulateur PID par les colonies de fourmis artificielle (ACO) Université Mohamed Khider Biskra

[3] http://www.acsysteme.com/fr/pid-serie-ou-parallele

[4] https://www.elprocus.com/the-working-of-a-pid-controller/

[5] Aidan O'Dwyer 2005-01-01 PID control: the early years Technological University Dublin, aidan.odwyer@dit.i

[6] https://instrumentationtools.com/pid-controllers/

[7] http://engineerexperiences.com/advantages-and-disadvatages.html

[8] https://www.electronicshub.org/arduino-introduction/#Introduction

[9] http://www.circuitbasics.com/arduino-thermistor-temperature-sensor-tutorial/

[10] https://www.instructables.com/id/Hardware-Structure-of-ARDUINO-UNO/

[11] https://components101.com/mosfets/irfz44n-datasheet-pinout-features

[12] F. Mudry, Ajustage des Paramètres des Régulateurs PID , fmy / mars 2006, eivd

[13] http://linuxcnc.org/docs/html/motion/pid_theory_fr.html

Annexes

NTC Thermistors, Radial Leaded, Standard Precision

QUICK REFERENCE DATA		
PARAMETER	VALUE	UNIT
Resistance value at 25 °C	3.3 to 470K	Ω
Tolerance on R_{25}-value	± 2; ± 3; ± 5	%
$B_{25/85}$-value	2880 to 4570	K
Tolerance on $B_{25/85}$-value	± 0.5 to ± 3	%
Operating temperature range: At zero power dissipation; continuously At zero power dissipation; for short periods	 -40 to +125 ≤ 150	°C
Response time (in oil)	≈ 1.2	s
Thermal time constant τ (for information only)	15	s
Dissipation factor δ (for information only)	7 8.5 (for R_{25}-value ≤ 680 Ω)	mW/K
Maximum power dissipation at 55 °C	500	mW
Climatic category (LCT / UCT / days)	40 / 125 / 56	-
Weight	≈ 0.3	g

FEATURES

- Accuracy over a wide temperature range
- High stability over a long life
- Excellent price/performance ratio
- RoHS compliant, available with or without exemption
- UL recognized, file E148885
- Mounting: radial
- Material categorization: for definitions of compliance please see www.vishay.com/doc?99912

APPLICATIONS

- Temperature measurement, compensation, sensing and control in consumer and industrial applications

DESCRIPTION

These thermistors have a negative temperature coefficient. The part consists of a NTC chip, soldered between two tin plated copper wires. It has a gray base coating and is color band coded. The coating has no specified insulation properties.

PACKAGING

The thermistors are packed in bulk or tape on reel; see part numbers and relevant packaging quantities.

DESIGN-IN SUPPORT

For complete Curve Computation, visit:
www.vishay.com/thermistors/ntc-curve-list/

MARKING

The thermistors are marked with colored bands; see dimensions drawing and "Electrical data and ordering information".

MOUNTING

By soldering in any position.
Not intended for potted applications.

Datasheet de notre capteur CTN

PARAMETER FOR DETERMINING NOMINAL RESISTANCE VALUES											
NUMBER	$B_{25/85}$ (K)	NAME	TOL. B (%)	A	B (K)	C (K^2)	D (K^3)	A_1	B_1 (K^{-1})	C_1 (K^{-2})	D_1 (K^{-3})
1	2880	Mat O. with Bn = 2880K	3	- 9.094	2251.74	229098	- 2.744820E+07	3.354016E-03	3.495020E-04	2.095959E-06	4.260615E-07
2	2990	Mat P. with Bn = 3990K	3	- 10.2296	2887.62	132336	- 2.502510E+07	3.354016E-03	3.415560E-04	4.955455E-06	4.364236E-07
3	3041	Mat Q. with Bn = 3041K	3	- 11.1334	3658.73	- 102895	5.166520E+05	3.354016E-03	3.349290E-04	3.683843E-06	7.050455E-07
4	3136	Mat R. with	3	- 12.4493	4702.74	- 402687	3.196830E+07	3.354016E-03	3.243880E-04	2.658012E-06	- 2.701560E-07
5	3390	Mat S. with Bn = 3390K	3	- 12.6814	4391.97	- 232807	1.509643E+07	3.354016E-03	2.993410E-04	2.135133E-06	- 5.672000E-09
6	3528 (1)	Mat I. with Bn = 3528K	0.5	- 12.0596	3687.667	- 7617.13	- 5.914730E+06	3.354016E-03	2.909670E-04	1.632136E-06	7.192200E-08
	3528 (2)			- 21.0704	11903.95	- 2504699	2.470338E+08	3.354016E-03	2.933908E-04	3.494314E-06	- 7.712690E-07
7	3560	Mat H. with Bn = 3560K	1.5	- 13.0723	4190.574	- 47158.4	- 1.199256E+07	3.354016E-03	2.884193E-04	4.118032E-06	1.786790E-07
8	3740	Mat B. with Bn = 3740K	2	- 13.8973	4557.725	- 98275	- 7.522357E+06	3.354016E-03	2.744032E-04	3.666944E-06	1.375492E-07
9	3977	Mat A. with Bn =3977K	0.75	- 14.6337	4791.842	- 115334	- 3.730535E+06	3.354016E-03	2.569850E-04	2.620131E-06	6.383091E-08
10	4090	Mat C. with Bn = 4090K	1.5	- 15.5322	5229.973	- 160451	- 5.414091E+06	3.354016E-03	2.519107E-04	3.510939E-06	1.105179E-07
11	4190	Mat D. with Bn = 4190K	1.5	- 16.0349	5459.339	- 191141	- 3.328322E+06	3.354016E-03	2.460382E-04	3.405377E-06	1.034240E-07
12	4370	Mat E. with Bn = 4370K	2.5	- 16.8717	5759.15	- 194267	- 6.869149E+06	3.354016E-03	2.367720E-04	3.585140E-06	1.255349E-07
13	4570	Mat F. with Bn = 4570K	1.5	- 17.6439	6022.726	- 203157	- 7.183526E+06	3.354016E-03	2.264097E-04	3.278184E-06	1.097628E-07

Notes
(1) Temperature < 25 °C
(2) Temperature ≥ 25 °C

Les coefficients A-B-C de CTN qu'ils on a utilisé dans notre projet