République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de JIJEL

Faculté des sciences et de la Technologie

Département d'Electronique

Mémoire de fin d'étude

Pour l'obtention du diplôme de Master Académique

Domaine : **Sciences et Technologie**

Filière : **Electronique**

Spécialité : **Electronique des systèmes embarqués**

Régulation de la température et supervision à distance

Présenté par :

BENZAID Baya

DERF Sabrina

Sous la direction de :

Mr : SANTOUH Zoubir

Saison universitaire:

2019/2020


بسم الله الرحمن الرحيم

Remerciement

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

Nous tenons à remercier notre directeur de recherche

Monsieur Santouh.Z

Nous avons 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 nous faites un grand honneur en acceptant de juger ce travail.

Chaleureux remerciement à nos familles qui nous ont soutenues moralement, nous remercions nos parents, qui nous ont encouragées et aidées à arriver à ce stade de notre formation.

Nous tenons vivement à remercier tous ceux et celles qui nous ont soutenues dans la réalisation de ce modeste travail.

BENZAID BAYA

DERF SABRINA

Dédicaces

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 sœurs, mes frères, mes amis et mes collègues à qui je souhaite un avenir radieux plein de réussite aux personnes qui m'ont toujours aidé et encouragé, qui étaient toujours à mes côtés et qui m'ont accompagné durant mon chemin d'études.

Benzaid Baya

Dédicaces

Je dédie ce modeste travail à ma grande famille surtout ma mère, mon mari et mes enfants je vous remercie de votre patience le long de cette année interminable, de votre soutien précieux.

À ms sœur, mes frères, et belle famille, mes collègues de travail qui m'ont aidé et encouragé.

Que dieu vous protège.

Derf Sabrina

Résumé

L'évolution de la technologie crée continuellement des défis que l'homme est toujours prêt à les prendre. Un de ces défis est la commande à distance qui simplifie des contrôles qui ont été difficile voire impossible. La commande à distance de la température est un secteur très important dans le domaine industriel ou dans la vie quotidienne. Ce projet consiste à réaliser un régulateur de température d'un four électrique à distance. Ça implique un développement d'un système de contrôle numérique basé sur une carte Arduino et supervisé par une interface LabVIEW depuis un PC, qui communique avec la carte Arduino via le module Bluetooth, pour commander le système de chauffage (four électrique).

Mots clés : régulation industrielle, contrôle de température à distance, connexion Bluetooth, supervision LabVIEW.

ملخص

يؤدي التطور المستمر للتكنولوجيا إلى خلق تحديات دائمًا ما يكون الإنسان على استعداد لمواجهتها .أحد هذه التحديات هو التحكم عن بعد بالأنظمة التي من الصعب إن لم يكن من المستحيل ضبطها. يعد التحكم في درجة الحرارة عن بعد قطاعًا مهمًا للغاية في المجال الصناعي أو في الحياة اليومية .يعنى هذا المشروع بدراسة تطبيق مراقبة وتحكم عن بعد بدرجة حرارة فرن كهربائي يشرف عليه واجهة غرافيكية لتطبيقLabVIEW من جهاز كمبيوتر يتصل بلوحة Arduino عبر وحدة Bluetooth للتحكم في نظام تدفئة (فرن كهربائي).

الكلمات الرئيسية: التحكم الصناعي، التحكم في درجة الحرارة عن بعد، اتصال Bluetooth، الإشراف لابفيو.

Table des matières

Remerciement……..…i
Dédicace…….ii
Résumé.....iii
Table des matières.....iv
Liste des figures.....vii
Liste des tableaux.....ix
Liste d'abréviation.....ix

Introduction générale 1

Chapitre1 : La régulation

I Introduction ……..……2
II La régulation industrielle……..…….2
II 1 Définition d'un systeme.....2
II 2 Identifications d'un système……..……2
II 3 Identification dans le domaine temporel…….……..3
II 3 1 Système du premier ordre ……..…….…….....3
II 3 1 1 Définition……..…….……..……3
II 3 2 Réponse d'un système du premier ordre.……..……3
II 3 3 Système du second ordre…….…….……..……4
II 4 Méthode de Broïda…….…….……..…5
II 5 Modèle d'ordre supérieur ‘’Méthode de Strejc”…….……..…6
III Objectif de la régulation automatique…….……..……7
IV Chaines de régulations……..…….....7
IV 1 Régulation en boucle ouverte……..……7
IV 2 Régulation en boucle fermée…….…..……..…..8
V Eléments et signaux caractéristiques d'un système de régulation……...…….…..8
V 1 Constitution d'une régulation……..…..…….....8
V 1 1 Partie commande……...……..……..…..8
V 1 2 Actionneur……...…..……..8
V 1 3 Capteur……...……..……..…..9

V 2 Formes fondamentales de régulation……..…….9
V 2 1 Régulation tout ou rien……..……9
V 2 2 Régulation avec PID……...…….……11
V 2 2 1 principe de régulateur PID……...…….……11
V 2 2 2 Foncions de régulateur PID……..…….……11
V 2 2 3 Rôle des actions dans PID…….……12
V 2 2 4 Méthodes de réglage des actions…….…….....12
V 2 2 5 Méthodes de Ziegler et Nichols (ZN)……...…….……...13
VII Conclusion…….....14

Chapitre2 : Implémentation matérielle et logicielle du projet

I Introduction……..……...…..……15
II Implémentation matérielle……..……...15
II 1 La carte Arduino UNO……..…….15
II 1 1 Microcontrôleur……..…… …16
II 1 2 Entrées/Sorties numériques…….……17
II 1 3 Entrées analogiques…….…….……17
II 2 Le module Bluetooth……..…….……17
II 2 1 Présentation du module……...…….…….17
II 2 2 Branchement du module à Arduino…….…….……18
II 3 Afficheur LCD…….……... ……19
II 3 1 Définition …….……...……19
II 3 2 Schéma de câblage…….…… ….……19
II 3 3 Code programmation……...……...…..20
II 3 4 MOSFET de puissance IRF520 Canal N……..21
II 3 5 la connexion RS232……...…21
III Implémentation logiciel ……22
III 1 Logicielles utilisés……...…..……...…..22
III 1 1 Arduino IDE……...…..22
III 1 2 PROTEUS…….……...…22
III 1 3 LabVIEW……..……23
III 1 4 Pilote de Port Série Virtuel ‘VSPD’ (Virtual Serial Port Driver).....23
III 2 Langage de programmation……...……24

III 2 1 Structure générale d'un programme en langage Arduino...……...24
III 2 2 Etapes de l'implémentation sur le microcontrôleur...……25
III 3 Raisons de choisir Arduino ……..……..…25
III 4 Leurs applications…….……26
IV Conclusion…….……26

Chapitre3 : LabVIEW et Arduino et communication bidirectionnelle

I Introduction……...……..……27
II Définition……...……..……27
III La programmation graphique……...……..……27
III 1 Les palettes……...……28
III 1 1 Une palette de commande……..……28
II 1 2 Une palette de fonctions……..……29
II 1 3 Une palette d'outils (Tools)…….……30
III 2 Structures de données variables……31
III 3 Structures de programmation dans LabVIEW……..……31
III 3 1 Structure séquence……...……31
III 3 2 Structure Conditionnelle……..……31
III 3 3 Structures d'itération……31
III 3 4 La boucle conditionnelle : TANT QUE (While Loop) ……...…32
III 4 Graphes dans LabVIEW……...33
IV Transmission des données……..……34
V L'interface de programmation NI-VISA……..…..34
VI Conclusion……..……...…35

Chapitre4 : Réalisation et identification du système

I Introduction……..…….……36
II Réalisation des programmes …….…….……36
II 1 Arduino……...……..…….……...…36
II 1 1 Organigramme du programme d'Arduino……..……...…37
II 1 2 Liste des commandes et leurs procédures……...…….....……37
II 2 Programme sous LabVIEW……..…….....…38
II 2 1 Organigramme du programme LabVIEW……..……...…38

II 2 2 Explication de l'organigramme……39
II 2 3 Front panel……40
III Identification et configuration du régulateur……41
III 1 La régulation tout ou rien……42
III 1 1 Implémentation et simulation logiciels……43
III 1 2 Interprétation des résultats……44
III 2 La régulation PID……44
III 2 1 Implémentation et simulation logiciels……47
III 2 2 Interprétation des résultats……47
IV Création de l'application final……48
VI Conclusion……48
Conclusion générale ……49
Bibliographie……49

Liste des figures

Figure I.1. Schéma synoptique d'un système industriel.

Figure I.2. Système du premier ordre.

FigureI.3. Réponse d'un système du deuxième ordre à un échelon pure**.**

Figure I.4. Système sans dépassement avec un point d'inflexion.

Figure I.5. Réponse d'un système sans dépassement.

Figure I.6. Structure d'un système de régulation.

Figure I.7. Commande en boucle ouverte.

Figure I.8. Commande en boucle fermée.

Figure I.9. Constitution d'une chaine fermée.

Figure I.10. Caractéristique d'un régulateur TOR.

Figure I.11. Evolution des grandeurs dans le temps.

Figure I.12. Evolution des grandeurs dans le plan XY.

Figure I.13. Schéma synoptique du régulateur PID.

Figure II.1 Photo réelle du module Arduino.

Figure II.2 Description de la carte Arduino UNO.

Figure II.3 Photo réelle du module Bluetooth CH-05.

Figure II.4 Branchement du module Bluetooth CH-05.

Figure II.5 Afficheur 16x2 LCD

FigureII.6 Branchement d'LCD avec Arduino

FigureII.7 MOSFET IRF520.

Figure II.8 Structure de RS232

Figure II.9 Chronologie des langages de programmation

Figure II.10 Figure montre le rôle d'VSPD.

Figure II.11 Structure d'un programme Arduino.

Figure III.1 Panneau-avant et bloc de diagramme

Figure III.2 Palette de commandes.

Figure III.3 Les quatre types de contrôleur numérique disponibles

Figure III.4 Palette de fonctions

Figure III.5 Palette d'outils.

Figure III.6 Choix 'Structures' de la palette Fonctions

Figure III.7 Terminaison de la boucle 'While'

Figure III.8 Choix des Graphes de la palette 'Commandes'

Figure III.9 L'interface d'E/S VISA

Figure III.10 Exemple de connexion RS232 dans LabVIEW

Figure IV.1 Illustration d'un message et de son accusé entre LabVIEW et Proteus.

Figure IV.2 Fonctionnement du programme Arduino.

Figure IV.3 Fonctionnement du programme LabVIEW

Figure IV.4 Schéma bloc descriptive.

Figure IV.5 Panneau avant d'application.

FigureIV.6 Réponse du système à un échelon.

FigureIV.7 Identification des paramètres du système.

FigureIV.8 la commande TOR

FigureIV.9 Schéma de simulation sous Proteus.

FigureIV.10 Graphe de température régulée.

FigureIV.11 Graphe du signal de commande en PMW.

FigureIV.12 Graphe de la loi de commande et la température mesurée.

FigureIV.13 Création de l'application final.

Liste des tableaux

Tableau I.1 Réglage du régulateur.

Tableau I.2 Paramètres PID obtenus à partir d'une réponse indicielle (ZNt)

Tableau I.3 Paramètres PID obtenus à partir d'une réponse indicielle (ZNt) aux point critique.

Tableau I.4 Règles et conditions flou.

Tableau II.1 Configuration des broches du IRF520.

Tableau III.1 Fonctionnalités de LabVIEW.

Tableau IV.1 Règles de détermination des paramètres du PID

Liste d'abréviations

TOR	régulation tout ou rien.
PID	proportionnel, intégral, dérivé.
ZN	Ziegler et Nichols.
Kr	le gain critique
Tc	la période d'oscillation.
Kp	gain proportionnel
Ki	gain intégrale
Kd	gain du dérivé**.**
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
GPIB	général purpose interface bus
E/S	entrés /sortis
VI	Virtual Instrument.
ASCII	American Standard Code for Information Interchange
VISA	Virtual Instrument Software Architecture
API	Application Programming Interface
EEPROM	Electrically Erasable Programmable Read-Only Memory
PWM	Modulation à largeur d'imputions
UAL	unité arithmétique et logique.
A/N	conversion analogique/numérique.
PCB	Printed Circuit Board
LCD	Liquid Crystal Display
IDE	Integrated Développement Environement.
PROTEUS	Processor pour Text Easy to USE

Introduction générale

Historiquement, 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 régulateurs automatiques PID étaient largement adoptés pour une utilisation industrielle.

C'est dans ce cadre que se situe notre projet de fin d'études intitulé : « régulation de température et supervision à distance ».

La problématique abordée consiste à réaliser un système de régulation de température avec deux types de régulation : PID et tout ou rien (TOR). De ce fait, nous avons exécuté les tâches suivantes : étude détaillée des différents organes d'un système automatique, conception et simulation d'un système à base de kit Arduino, programmation du kit, et enfin développement d'une interface graphique sous LabVIEW assurant ainsi le dialogue entre Arduino et LabVIEW.

Lorsqu'une température est définie pour un four ainsi que les divers paramètres de régulation, par notre logiciel de supervision « LabVIEW », le régulateur surveille la température réelle à l'intérieur du four. Notre but principal dans cette étude est d'obtenir une application **exécutable** dans n'importe quel écran PC, et supervisée à distance via le module Bluetooth.

Ce mémoire porté sur vos mains s'articule autour de quatre chapitres suivants :

Tout d'abord, nous commençons par une présentation générale et théorique des divers types de régulation de systèmes thermique existant, une description de ces caractéristiques et équations mathématiques. Le **deuxième chapitre** sera consacré au matériel et logiciels utilisés.

Le troisième chapitre une brève description du logiciel LabVIEW, son architecture de ses différentes fenêtres et leurs rôles, ainsi que son principe de fonctionnement. Nous présenterons, dans le même volet, l'interfaçage avec la carte ARDUINO

Finalement, nous enrichissons notre travail par la mise en marche finale du système **:** présenter la procédure de développement d'une interface LabVIEW permettant la visualisation des résultats obtenus et les graphes, …Nous achèverons ce mémoire par une conclusion et des perspectives.

I : la régulation

I Introduction

La régulation est une discipline technique destiné à analyser et concevoir des systèmes de commande pratiques et autres dispositifs technologiques.

Réguler une grandeur, c'est obtenir d'elle un comportement donné, dans un environnement susceptible de présenter des variations.

L'objectif global de la régulation est de maintenir une grandeur physique à une valeur constante quelques soient les perturbations extérieures selon le protocole : Mesurer, Comparer et Corriger. Nous sommes donc amenés à effectuer des mesures pour obtenir certaines connaissances avant d'entreprendre une action. Ces mesures seront obtenues par l'intermédiaire d'appareillages spécifiques. Parmi les exemples de procédé de régulation on cite : la régulation de la température. Notre objectif et maintenir la température à une valeur désirée (consigne).

II La régulation industrielle

II 1 Définition d'un système

Un système est un ensemble d'appareils utilisé pour obtenir un produit déterminé. L'évolution de ce dernier dépend d'une ou plusieurs grandeurs incidentes, d'ailleurs il est caractérisé par une ou plusieurs grandeurs physiques mesurables et maîtrisables qui vont permettre de contrôler l'objectif fixé, comme il peut être simple ou complexe.

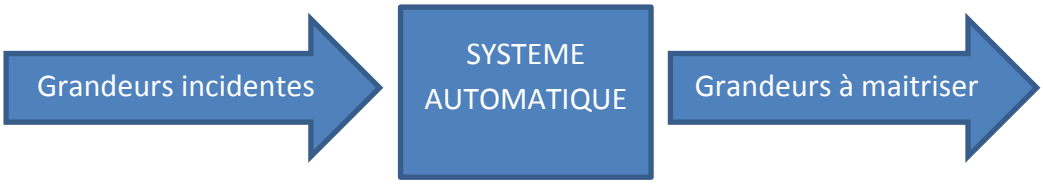


Figure I.1 : Schéma synoptique d'un système industriel.

Afin de pouvoir contrôler (régler, asservir) un système, il est nécessaire de connaître un certain nombre de ses propriétés :

- nombre et nature des entrées et des sorties.
- comportement statique et dynamique (temps de montée, nombre et période des oscillations).
- linéarité ou non-linéarité.
- stabilité, …

II 2 Identification d'un système

L'identification d'un système est la détermination de son modèle mathématique sur la base des observations expérimentales entrées/sorties ce qui détermine sa particularité.

Le traitement mathématique des réponses graphiques du système est appelé : **IDENTIFICATION**. Le modèle obtenu est dit de conduite ou représentation [1].

II 3 Identification dans le domaine temporel

Cette technique consiste à attaquer le système par un échelon d'amplitude donnée et s'intéresser par la suite à l'évolution dans le temps de sa sortie. Ensuite nous comparons la réponse obtenue avec une fonction de transfert normalisée. Il faut donc utiliser une méthode d'identification qui ne demande aucune connaissance préalable si ce n'est l'enregistrement de la réponse indicielle. Le modèle que l'on obtiendra alors sera un modèle de représentation qui peut n'avoir aucune correspondance physique avec le système considéré, mais si l'identification est bonne, les réponses indicielles ou fréquentielle seront aussi proches que possible de celles du système étudié.

II 3 1 Système du premier ordre

II 3 1 1 Définition

On appel système du premier ordre tout système dont le fonctionnement est décrit par une équation différentielle du premier ordre.

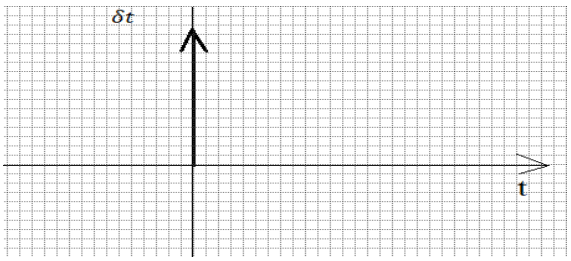
$$\tau \frac{ds(t)}{dt} + s(t) = e(t) \Rightarrow \tau pS(p) + S(p) = E(p) \Rightarrow T(p) = \frac{S(p)}{E(p)} = \frac{1}{1+\tau p} \qquad (1.1)$$

Fonction de transfert du système du premier ordre [5].

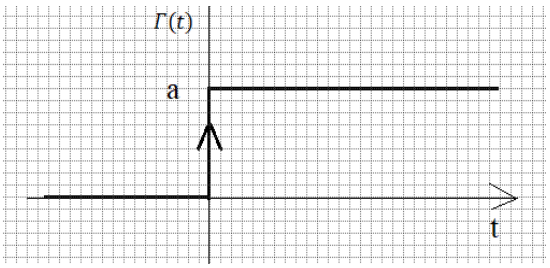
II 3 2 Réponse d'un système du premier ordre

On appel réponse d'une fonction la réponse s(t) à une entrée e(t) connu et non périodique. Les entrées donnant des réponses indicielles :

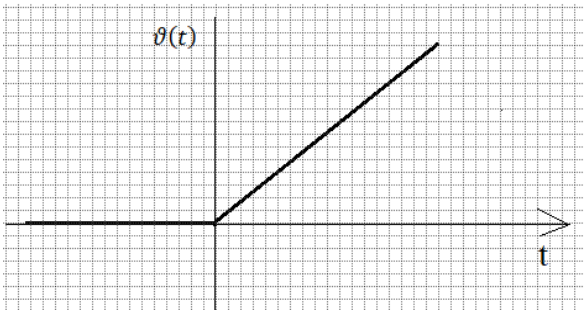
a. Entrée impulsion : Pic de Dirac : $e(t) = \begin{cases} 0 & t \neq 0 \\ \delta t & t = 0 \end{cases}$; $L[\delta t] = 1$ (1.2)



b. Entrée échelon : $e(t) = \Gamma(t) = \begin{cases} 0 & t < 0 \\ a & t > 0 \end{cases}$; $L[\Gamma(t)] = \frac{a}{p}$ (1.3)



c. Entrée rampe :$e(t) = \vartheta(t) = \begin{cases} 0 & t < 0 \\ at & t \geq 0 \end{cases}$; $L[\vartheta(t)] = \frac{a}{p^2}$ (1.4)



La réponse d'un système du premier ordre soumis à un échelon, est donnée comme suit :

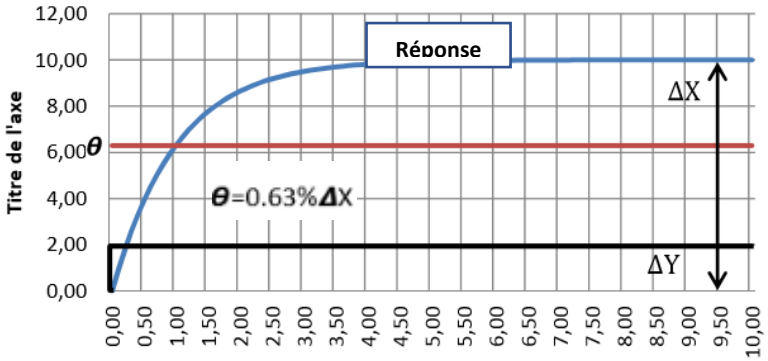


Figure I.2 : Système du premier ordre

Le modèle du premier ordre est de la forme : $H(P) = \frac{X(P)}{Y(P)} = \frac{Gs.e^{-\tau P}}{(1+\theta P)}$ (1.5)

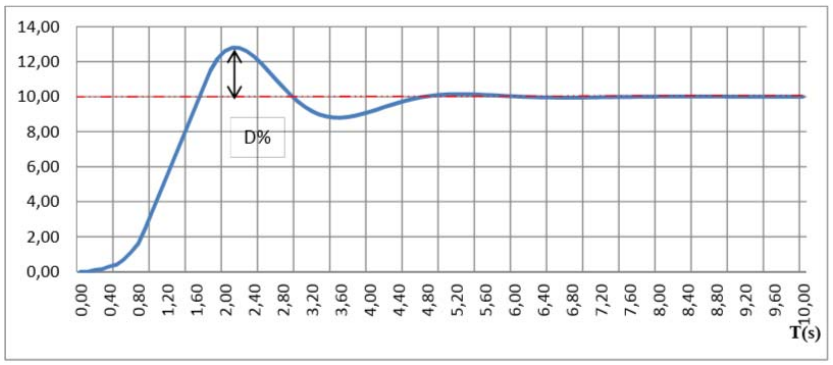
La détermination des paramètres de modèle se fait comme suit :

• Le gain statique est mesuré directement par : $Gs = \frac{\Delta x}{\Delta y}$ (1.6)

• La constante de temps $\boldsymbol{\theta}$: on trace conjointement la droite d'ordonnée (0.63 Δx) parallèle à l'axe des abscisses.

II 3 3 Système du second ordre

La réponse d'un système du second ordre soumis à un échelon, est donnée comme suit :



FigureI.3 : Réponse d'un système du deuxième ordre à un échelon pure

Sa forme canonique est : $G(p) = \frac{G(s)}{1+\frac{2h}{Wn}p+p^2+\frac{p^2}{Wn^2}}$ (1.7)

- Le facteur d'amortissement 'h' se détermine à l'aide de la mesure du dépassement :

$$D\% + 100e^{-\frac{\pi h}{\sqrt{1-h^2}}} \quad (1.8)$$

- La pulsation propre Wn: $Tpic = \frac{\pi}{Wn\sqrt{1-h^2}}$ (1.9)

II 4 Méthode de Broïda

Broida a estimé qu'en passant d'un ordre 'n' à un 1er ordre que la tangente au point d'inflexion était une source d'erreur importante et que la durée des essais pouvait être longue sur les systèmes lents avec le risque d'avoir une entrée qui varie pendant l'essai [2] .

Le modèle proposé pour approcher le comportement du système est un premier ordre avec retard, sa fonction de transfert est : $G(p) = \frac{Gs.e^{-\tau P}}{(1+\theta P)}$ (1.10)

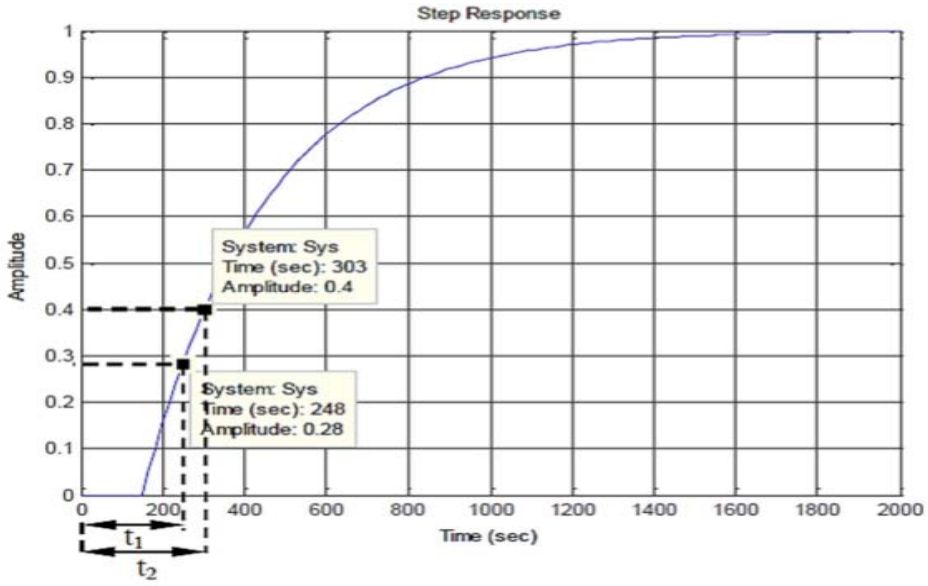


Figure I.4 : Système sans dépassement avec un point d'inflexion

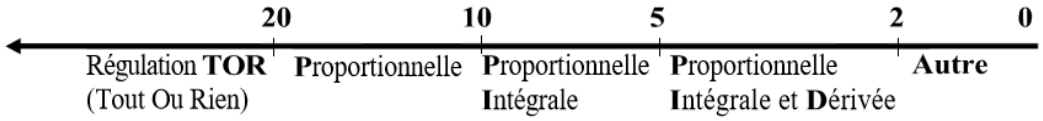
Ces calculs montrent que pour obtenir les temps t1 et t2, on prend respectivement pour Y(t) les valeurs 0.28 Δx et 0.40 Δx

- Le gain statique Gs : $Gs = \frac{\Delta x}{\Delta y}$ (1.11)
- La constante de temps (θ) : $\theta = 5{,}5 * (t40\% - t28\%)$ (1.12)
- Le retard (τ) : $\tau = (2.8t28\% - 1.8t40\%)$ (1.13)

• Choix de la régulation – indice de réglabilité

La difficulté de réguler un procédé est d'autant plus grande, pour une constante θ donnée, que le retard τ est grand. Il est donc naturel de mettre en œuvre un régulateur d'autant plus riche en action que le procédé comporte un rapport $\frac{\theta}{\tau}$ petit. Le modèle de Broïda ait été établi par calcul

ou par identification expérimentale, le graphe suivant guide sur le choix de la régulation en fonction du rapport $\frac{\theta}{\tau}$.



• Réglage du régulateur : Une fois la régulation choisie, le tableau ci-dessous conduit au réglage du régulateur à appliquer selon Broïda.

Tableau I.1 : Réglage du régulateur

	P	PI série	PI parallèle	PID série	PID parallèle	PID mixte
Kc	$\frac{0,78}{Gs}\cdot\frac{\theta}{\tau}$	$\frac{0,78}{Gs}\cdot\frac{\theta}{\tau}$	$\frac{0,78}{Gs}\cdot\frac{\theta}{\tau}$	$\frac{0,83}{Gs}\cdot\frac{\theta}{\tau}$	$\frac{0,83}{Gs}\cdot(\frac{\theta}{\tau}+0,4)$	$\frac{0,83}{Gs}\cdot(\frac{\theta}{\tau}+0,4)$
Ti	sans	$\boldsymbol{\theta}$	$\frac{\tau.Gs}{0,78}$	$\boldsymbol{\theta}$	$\frac{\tau.Gs}{0,75}$	$\boldsymbol{\theta}$+0,4τ
Td	0	0	0	0,42τ	$\frac{0,35.\theta}{Gs}$	$\frac{\theta.\tau}{\tau+2.5\theta}$

II 5 Modèle d'ordre supérieur ''Méthode de Strejc".

Cette méthode permet l'identification d'un processus dont la réponse à l'échelon n'a pas de dépassement.

La méthode de Strejc consiste à caractériser le procédé par un modèle de la forme :

$$G(p) = \frac{Gs.e^{-\tau P}}{(1+\theta P)^n} \qquad (1.14)$$

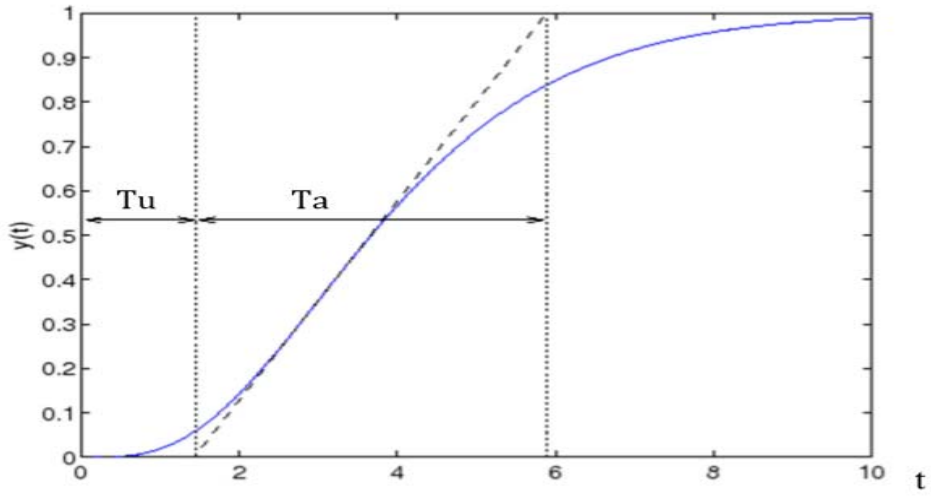


Figure I.5 : Réponse d'un système sans dépassement

• Le principe de la méthode de Strejc :

Quatre éléments sont essentiels pour la détermination d'un modèle selon Strejc :

- Tracer la tangente au point d'inflexion, qui permet de définir les deux grandeurs (Tu) et (Ta), comme dans la Figure (I.5)
- On calcule le rapport η : $\eta = \frac{Tu}{Ta}$ (1.15)
- Dans le tableau (I.1) on cherche la valeur immédiatement inférieure à la valeur (η) dans la colonne Tu/Ta, en suivant la ligne correspondante, on obtient l'ordre (n) du modèle, mais aussi la 3éme colonne nous permet de calculer la constante de temps θ .

Le retard est égal à $\tau = Tu - \frac{Tu}{Ta}$ (1.16) [1].

III Objectif de la régulation automatique

La régulation regroupe l'ensemble des moyens matériels et techniques mis en œuvre pour maintenir une grandeur physique à réguler, égale à une valeur désirée appelée consigne.

Le fonctionnement de l'installation ne requiert que certaines grandeurs physiques y1(t), y2(t)…, d'un système aient un comportement fixé par les consignes c1(t), c2(t)…, malgré la présence de perturbations p1(t), p2(t)…, d'origine externe et imprévisible. A cet effet, les grandeurs physiques sont mesurées, traitées puis une action correctrice est entreprise sur le système au moyen des commandes u1(t), u2(t). Pour cela, le but de la régulation est d'ajuster la 'puissance' à apporter en fonction des besoins [1].

Suivant les procédés et les objectifs à réaliser, il existe une grande variété de matériels et de techniques utilisés en régulation ou en commande.

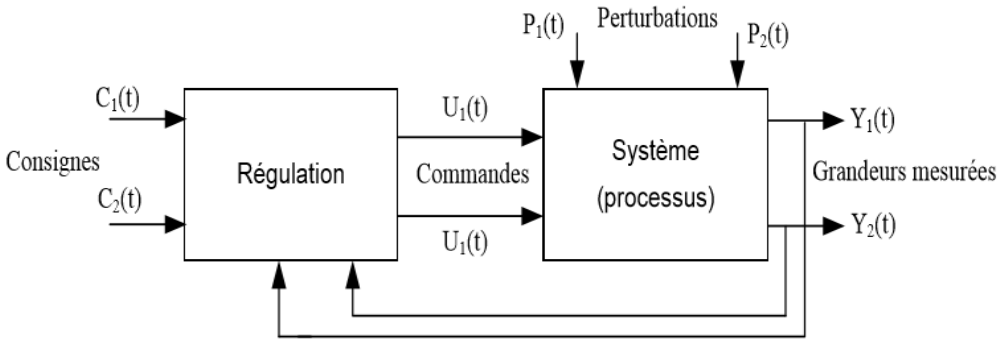


Figure I.6 : Structure d'un système de régulation.

IV Chaines de régulations

IV 1 Régulation en boucle ouverte

Une régulation est dite en boucle ouverte lorsque les signaux d'entrées e(t) ne sont pas influencés par les signaux de sorties s(t). Dans ce cas l'action ne modifie pas la grandeur observée (la grandeur à maîtriser). La mise en œuvre d'une telle régulation nécessite la connaissance des lois régissant le fonctionnement du processus, c'est-à-dire la corrélation entre la valeur mesurée et la grandeur régulant. La figure I.7 illustre la structure d'une commande en boucle ouverte. L'inconvénient majeur est que l'objectif fixé n'est généralement pas atteint complètement, (réside dans ses limites).

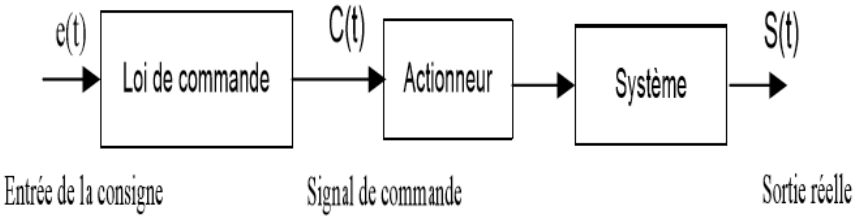


Figure I.7 : Commande en boucle ouverte.

IV 2 Régulation en boucle fermée

La grandeur réglant exerce une influence sur la grandeur réglée pour la maintenir dans des limites définies malgré les perturbations. Cette chaîne de régulation est dite fermée car l'action modifie la grandeur observée. Le principe de commande en boucle fermée est illustré sur la figure I.8 et définit la structure de commande. L'avantage d'une chaîne fermée est qu'une variation de la grandeur observée (la grandeur à maîtriser) entraîne une variation de l'action donc l'objectif fixé peut alors être atteint.

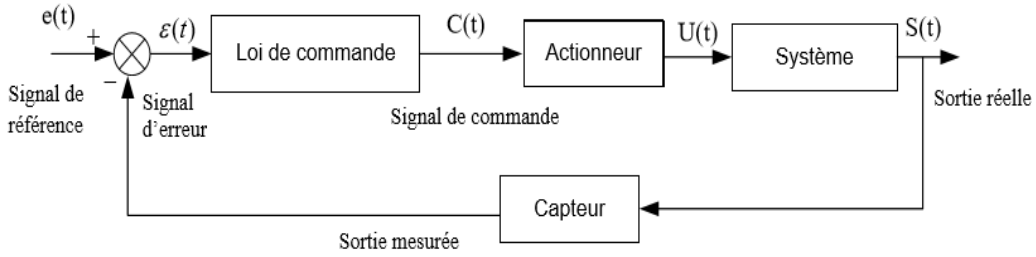


Figure I.8 : Commande en boucle fermée.

V Eléments et signaux caractéristiques d'un système de régulation

V 1 Constitution d'une régulation

V 1 1 Partie commande

Le régulateur est composé d'un comparateur qui détermine l'écart (ou l'erreur) entre la consigne et la mesure et d'un correcteur, qui élabore à partir du signal d'erreur l'ordre de commande.

V 1 2 Actionneur

C'est l'organe d'action qui apporte l'énergie au système pour produire l'effet souhaité. Il est en général associé à un pré-actionneur qui permet d'adapter l'ordre (basse puissance) et l'énergie.

V 1 3 Capteur

Le capteur prélève sur le système la grandeur réglée (information physique) et la transforme en un signal compréhensible par le régulateur. La précision et la rapidité sont deux caractéristiques importantes du capteur.

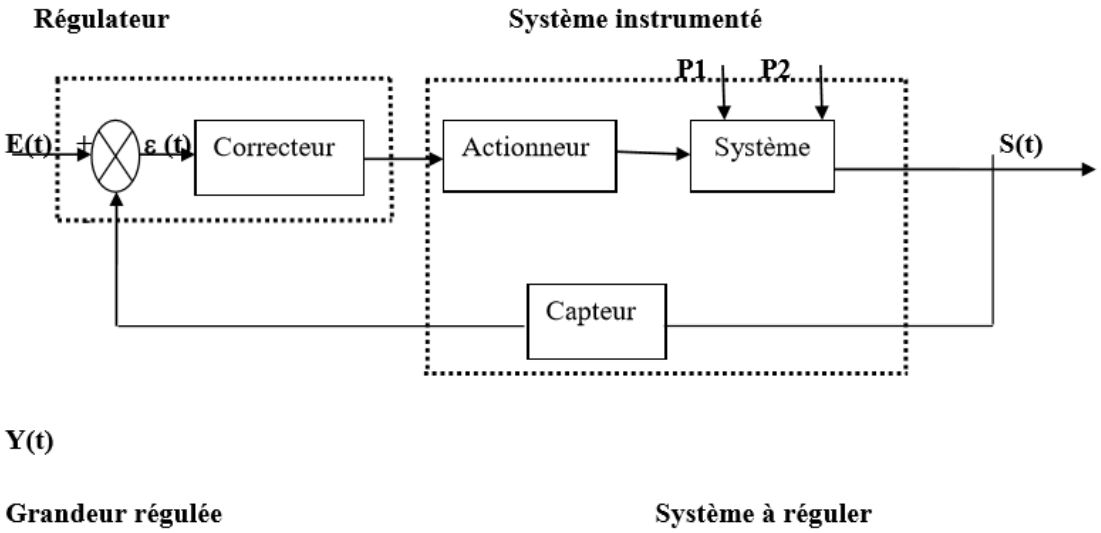


Figure I.9 : Constitution d'une chaine fermée.

- La consigne noté E(t) : est la valeur que l'on désire atteindre, c'est la grandeur régulant du système.
- La mesure de la grandeur à maîtriser est noté S(t) : représente le phénomène physique que doit réguler le système.
- L'écart ε(t) ou l'erreur entre la consigne et la mesure est représenté par l'équation :

$$\varepsilon(t) = e(t) - s(t) \qquad (1.17)$$

- P1 et P2 grandeurs non contrôlées sont les grandeurs perturbatrices appelées perturbations.

Dans l'étude d'un système de régulation, le système est défini par sa fonction de transfert en utilisant la transformée de 'Laplace' pour les systèmes analogiques et la transformée en 'Z' pour les systèmes numériques.

V 2 Formes fondamentales de régulation

V 2 1 Régulation tout ou rien (TOR)

Cette régulation a deux états possibles pour la commande ; Celui qui correspond à la commande maximale (100%) et celui qui correspond à la commande minimale (0%), avec un seuil de commutation [2] .

Le réglage du régulateur se fait à l'aide de deux paramètres : La consigne W, et le seuil DIFF.

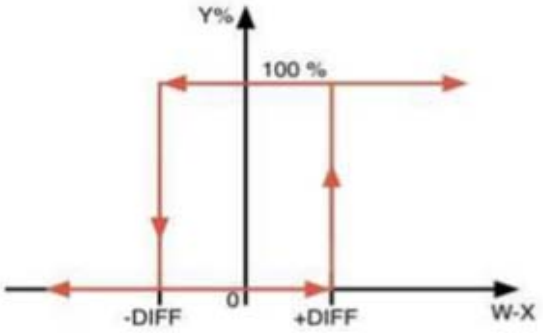


Figure I.10: Caractéristique d'un régulateur TOR

La grandeur réglée oscille autour du point de fonctionnement (figure I.11 et figure I.12). À chaque dépassement des seuils de commutation, la sortie du régulateur change d'état.

Compte tenu de l'inertie du système, la valeur absolue de l'erreur |{E}| peut dépasser le seuil DIFF. Sauf exception, la mesure ne peut pas être constante dans ce type de régulation. Le système est en régime d'instabilité entretenue.

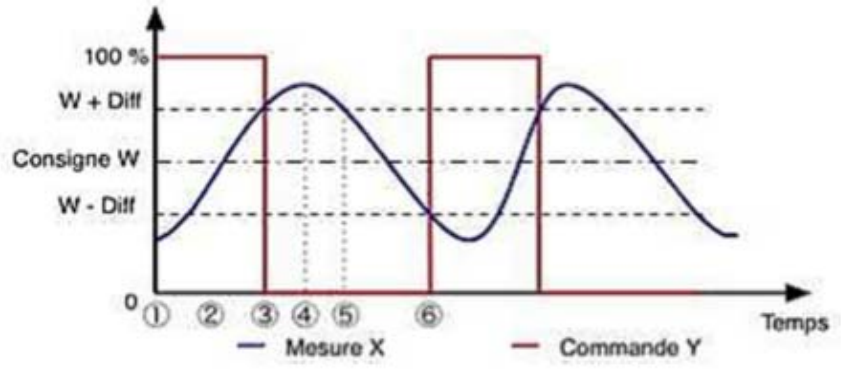


Figure I.11 : Evolution des grandeurs dans le temps

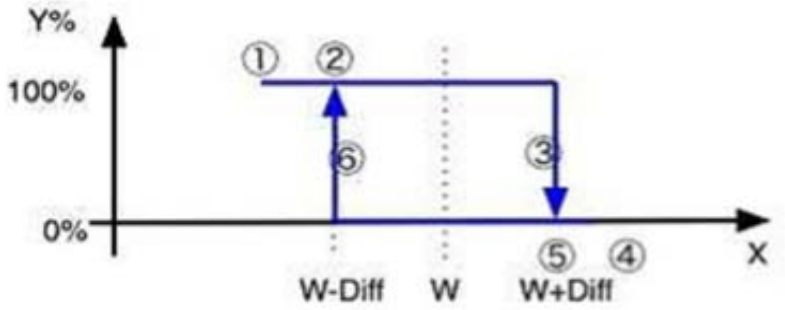


Figure I.12 : Evolution des grandeurs dans le plan XY

Cette régulation est utilisée quand la dynamique du système est très lente (grande constante de temps). Elle est acceptable pour les systèmes thermiques stables de faible puissance ou de forte inertie thermique.

Cette méthode de régulation est considérée comme une régulation discontinue car la commande envoyée aux actionneurs varie instantanément.

V 2 2 Régulation avec 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'une large gamme de processus industriels.

V 2 2 1 principe de régulateur PID

Le régulateur PID forme un signal de commande u(t) (grandeur de réglage) qui va faire varier la puissance de réglage l'erreur ε(t) par l'intermédiaire d'un actionneur (organe de réglage).

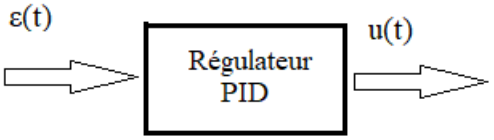


Figure I.13 : Schéma synoptique du régulateur PID

La commande u(t) donnée par le régulateur PID, dans sa forme Classique est décrite par :

$$u(t) = K_p \left[\varepsilon(t) + \frac{1}{T_i}\int_0^t \varepsilon(\tau)d\tau + T_d \frac{d\varepsilon(t)}{dt}\right] \quad (1.18)$$

Elle est la somme de trois termes :

- Le terme proportionnel : $P = K_p\varepsilon(t)$ (1.19)
- Le terme intégral : $I = K_p \frac{1}{T_i}\int_0^t \varepsilon(\tau)d\tau$ (1.20)
- Le terme dérivatif : $D = K_p T_d \frac{d\varepsilon(t)}{dt}$ (1.21)

Les paramètres du régulateur PID sont le gain proportionnel Kp, le temps intégral Ti et le temps dérivatif Td, les temps étant exprimés en secondes.

V 2 2 2 Foncions de régulateur PID

Un régulateur PID remplit essentiellement trois fonctions :

1. Il fournit un signal de commande u(t) en tenant compte de l'évolution du signal de sortie y(t) par rapport à la consigne w(t).
2. Il élimine l'erreur statique grâce au terme intégrateur.
3. Il anticipe les variations de la sortie grâce au terme dérivateur.

Il existe trois types de régulation PID, le PID série, le PID parallèle et le PID mixte.

P = Kp : est l'action proportionnelle, sur la plupart des régulateurs, on règle la Bande Proportionnelle au lieu de régler le gain du régulateur.

I = 1/Ti (min-1 en général) : est l'action intégrale

D = Td (s en général) : est l'action dérivée [3].

V 2 2 3 Rôle des actions dans PID

a. Rôle de l'action proportionnelle (P)

L'action proportionnelle vise à accélérer la réponse de la mesure, pour réduire l'écart entre la mesure et la consigne, en multipliant l'erreur par un gain, plus l'écart est important plus la correction est grande.

L'étude de l'action proportionnelle sur un procédé instable (aussi appelé intégrateur), montre que lors d'une variation de consigne, la mesure rejoint la consigne dans tous les cas, mais en cas de perturbation, la mesure s'écarte de la consigne, la régulation proportionnelle tend à la ramener tout en laissant subsister un écart résiduel, lorsque le régime permanent est atteint [4] .

b. Rôle de l'action Intégrale (I)

Le rôle de l'action intégrale est d'annuler l'écart entre la mesure et la consigne. Le signal de sortie du régulateur en intégrateur seul est proportionnel à l'intégrale de l'écart mesure-consigne. L'action intégrale est généralement associée à l'action proportionnelle.

Une augmentation excessive de l'action intégrale (diminution de Ti) peut être source d'instabilité. Le comportement de l'action intégrale sur un procédé instable, est sensiblement le même que sur un procédé stable. Il faut noter que l'action intégrale est nécessaire pour annuler l'écart, suite à des perturbations. Lors de changement de consigne, son intérêt est moindre car l'écart s'annule naturellement du fait que le procédé est lui-même intégrateur, l'action intégrale donne une réponse plus rapide qu'en régulation à action proportionnelle seule. [2]

c. Rôle de l'action dérivée (D)

Cette action compense les effets du temps mort (retard) du procédé. Elle a un effet stabilisateur mais une valeur excessive peut entraîner l'instabilité. Son rôle est identique quelle que soit la nature du procédé. La sortie du dérivateur est proportionnelle à la vitesse de variation de l'écart.

Dans le cas d'un signal de mesure bruité, la dérivée amplifie le bruit, ce qui rend son utilisation impossible.

La solution à ce problème consiste, soit à filtrer le signal de mesure, soit à utiliser un module de dérivée filtrée avec un gain transitoire réglable.

V 2 2 4 Méthodes de réglage des actions

Avant de commencer les réglages d'une boucle de régulation, il faut s'assurer que le sens d'action du régulateur est correct, quelle que soit la méthode de réglage utilisée, les réglages ne sont adaptés qu'au point de fonctionnement.

Il existe différentes méthodes de réglage des actions d'un régulateur PID suivant le type de procédé et les contraintes de fabrication[4] .

a. Méthode par approches successives

Elle consiste à modifier les actions du régulateur et à observer les effets sur la mesure enregistrée, jusqu'à obtenir la réponse optimale. On règle l'action proportionnelle, puis l'action dérivée et l'intégrale.

Cette technique est simple et utilisable sur n'importe quel type de système. Néanmoins du fait de son caractère itératif, son application devient longue sur des procédés à grande inertie.

b. Méthode nécessitant l'identification du procédé

Il est possible de calculer rapidement les paramètres de réglage, si l'on connaît les paramètres du procédé, suite à une modélisation de sa fonction de transfert, et si l'on est en possession de la structure du régulateur. Il est alors possible d'affiner ces suites à des essais, afin d'obtenir la réponse souhaitée.

Cette méthode nécessite un enregistreur à déroulement rapide. Elle est de préférence utilisée sur des procédés à grande inertie.

c. Méthode de Ziegler et Nichols

En 1942, Ziegler et Nichols ont proposé deux approches expérimentales destinées à ajuster rapidement les paramètres des régulateurs P, PI et PID. La première nécessite l'enregistrement de la réponse indicielle du système à régler seul, alors que la deuxième demande d'amener le système en boucle fermée à sa limite de stabilité [1].

V 2 2 5 Méthodes de Ziegler et Nichols (ZN)

a. Méthode de la réponse indicielle

Pour obtenir les paramètres du régulateur **PID**, il suffit d'enregistrer la réponse indicielle du processus seul (c'est-à-dire sans le régulateur), puis de tracer la tangente au point d'inflexion de la courbe. On mesure ensuite sa pente **p**, le retard apparent **L** correspondant au point d'intersection de la tangente avec l'abscisse et le gain $K_0 = Y(\infty)/E$. On peut alors calculer les coefficients du régulateur choisi à l'aide du tableau I.2.

Généralement, les gains **Kp** proposés par Ziegler-Nichols sont trop élevés et conduisent à un dépassement supérieur à 20%. Il ne faut donc pas craindre de réduire Kp d'un facteur 2 pour obtenir une réponse satisfaisante.

Tableau. I.2: Paramètres PID obtenus à partir d'une réponse indicielle (ZNt)

Type	Kp	Ti	Td
P	1/(pLK0) = 1/(aK0)		
PI	0.9/(pLK0) = 0.9/(aK0)	3L	
PID	1.2/(pLK0) = 1.2/(aK0)	2L	0.5L

b. Méthode du point critique

Cette méthode est basée sur la connaissance du point critique du processus. Expérimentalement, on boucle le processus sur un simple régulateur proportionnel dont on augmente le gain jusqu'à amener le système à osciller de manière permanente ; on se trouve ainsi à la limite de stabilité. Après avoir relevé le gain critique **Kcr** du régulateur et la période d'oscillation **Tcr** de la réponse, on peut calculer les paramètres du régulateur choisi à l'aide du tableau 2. Ici également, les valeurs proposées conduisent à un temps de montée relativement court malheureusement assorti d'un dépassement élevé. Cette situation n'étant pas toujours satisfaisante, on peut être amené à corriger les coefficients proposés et, en particulier, à diminuer le gain Kp. On notera que les paramètres Ti et Td proposés par les deux méthodes de Ziegler-Nichols sont dans un rapport constant égal à 4. Le régulateur possède donc deux zéros confondus valant :

$$-1/(2Td) = -2/Ti.$$

Tableau. I .3 : Paramètres PID obtenus à partir d'une réponse indicielle (ZNt) aux point critique.

Type	Kp	Ti	Td
P	0.5Kcr		
PI	0.4Kcr	0.8Tcr	
PID	0.6Kcr	0.5Tcr	0.125Tcr

VI Conclusion

A travers ce chapitre nous avons cité les différents composants d'une étude d'un système ainsi que les éléments caractérisant ce dernier, comme nous avons abordé une brève description des commandes à utiliser (PID, TOR, flou) soit en boucle ouverte ou fermée afin d'entamer une étude théorique qui nous permettras d'associer une meilleure régulation à notre propre système. Cette partie feras l'objet du prochain chapitre

II : Implémentation matérielle et logicielle du projet

I Introduction

Pour étudier le comportement d'un processus il est évident qu'on a besoin d'un montage électronique où implémenter l'algorithme développé. A défaut de la réalisation physique de ce montage on a recours à la simulation par ordinateur qui aide plus ou moins malgré la non prise en considération des conditions environnante réelles.

Dans ce chapitre nous allons présenter les matériel et logiciels utilisés dans la simulation de notre travail.

II Implémentation matérielle

II 1 La carte Arduino UNO

Arduino est un outil permettant de construire des dispositifs qui peuvent interagir avec l'environnement qui les entoure. On peut s'en servir pour y relier des capteurs détectant le son, la lumière ou les vibrations, qu'il utilisera alors pour allumer une lumière, changer sa couleur, mettre en route un moteur, et bien d'autres choses. Arduino est un système magique, qui se situe au cœur de toutes ces actions. Arduino se présente généralement sous la forme d'une carte électronique bleue, qui a à peu près la taille de la main. Cette carte comporte des inscriptions en blanc qui permettent de repérer ses différents éléments. Tous les composants et les circuits de la carte sont visibles et accessibles. Arduino est un microcontrôleur, autrement dit un ordinateur très simple. Il ne peut pas faire beaucoup de choses en même temps, mais ce qu'on lui dit de faire, il le fait très bien. Il existe de nombreux types de microcontrôleurs mais ce qui est particulier avec Arduino, c'est qu'il est conçu pour être utilisé par les débutants, mais il peut aussi s'adapter à de gros projets.

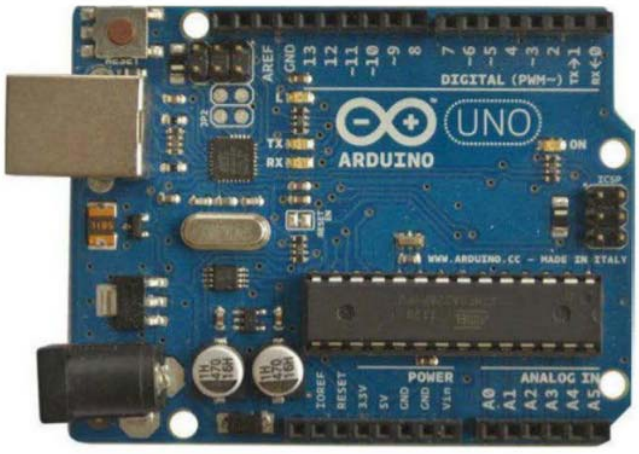


Figure II.1 : Photo réelle du module Arduino.

La carte à microcontrôleur Arduino UNO est la première version stable de cartes Arduino. Elle utilise un microcontrôleur AVR d'architecture ATmega328p cadencé à 16Mhz. Elle possède

32ko de mémoire flash destinée à recevoir le programme, 2ko de SRAM (mémoire vive) et 1 ko d'EEPROM (mémoire morte destinée aux données). Elle contient 14 broches d'entrée/sortie numériques dont 6 peuvent être utilisées comme sorties PWM (Modulation à largeur d'imputions) et elle permet aussi de mesurer des grandeurs analogiques grâce à ces 6 entrées analogiques, la figure II.2 illustre une carte Arduino UNO [7][10].

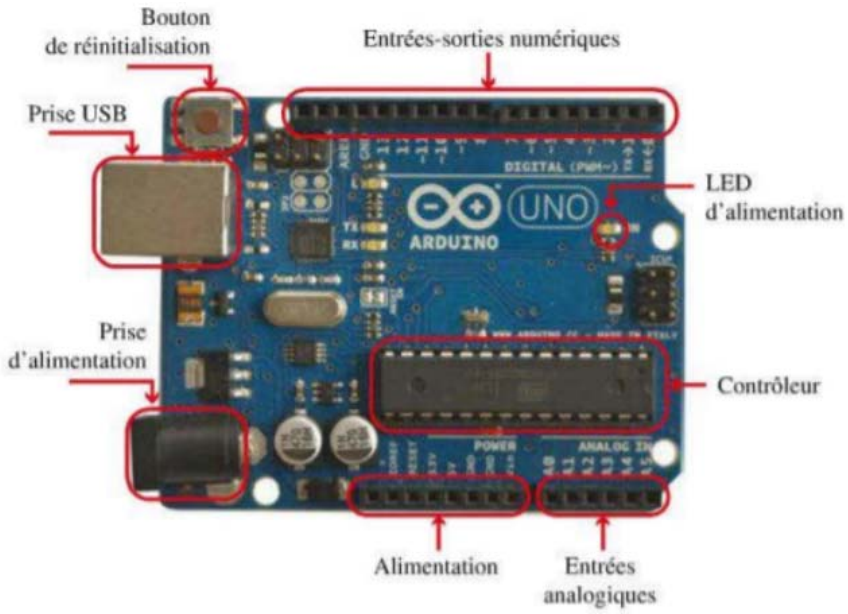


Figure II.2 : Description de la carte Arduino UNO.

II 1 1 Microcontrôleur

C'est un circuit intégré qui rassemble sur une puce plusieurs éléments dans un espace réduit, qui est un système à microprocesseur contenant des périphériques intégrés tel que mémoires de donnés, des programmes pouvant être utilisés comme un système embarqué. L'architecture de la carte Arduino a été publiée en open-source, et toute sa philosophie s'appuie sur le monde du libre, au sens large.

Les microcontrôleurs sont de plus en plus utilisés dans les applications embarquées tels les téléphones, les automobiles…

Le microcontrôleur est composé de quatre parties principales :

- Un microprocesseur dont la fonction est le traitement des informations, composé d'une unité arithmétique et logique (UAL), d'un bus de données, d'adresse et de commande, ayant pour tâche l'exécution du programme embarqué dans le microcontrôleur.
- Une mémoire de données (RAM ou EEPROM) dans laquelle seront stockées les données temporaires nécessaires aux calculs qui est une mémoire de travail volatile.
- Une mémoire de programme (flash), contenant les instructions du programme à exécuter. Il s'agit ici d'une mémoire non volatile.

- La dernière partie correspond aux ressources auxiliaires qui sont :
- Des ports d'entrées / sorties (parallèle ou série).
- Des Timers servant à générer ou mesurer des signaux.
- Des convertisseurs A/N pour le traitement des signaux analogiques.

II 1 2 Entrées/Sorties numériques

Cette carte possède 14 broches numériques (numérotée de 0 à 13) qui sont programmables par le biais d'instructions pinMode (), digitalWrite () et digitalRead (). Ces broches fonctionnent à 5V, chacune peut fournir ou recevoir un courant maximal de 40mA disposant d'une résistance interne de (rappel au plus) (pull-up) (déconnectée par défaut) de 20-50KOhms. Cette résistance interne s'active sur une broche en entrée à l'aide de l'instruction digitalWrite (broche, HIGH). En plus, certaines broches ont des fonctions spécialisées par exemple :

- Transmission et réception série : les broches 0 (RX) et 1 (TX) permet de recevoir et transmettre les données séries de niveau TTL et ces pins sont connectés aux pins correspondants de l'USB-TTL puce Serial ATmega8U2.
- Interruptions Externes : les broches 2 et 3 peuvent être configurées pour déclencher une interruption sur un niveau bas, sur un front montant, descendant et sur un changement de valeur.
- Modulation à largeur de bande (PWM : Pulse Width Modulation) : les broches 3, 5, 6, 9, 10,11 peuvent fonctionner en mode PWM pour faire varier la puissance du signal envoyé sur 8 bits à l'aide de l'instruction analogWrite ().

II 1 3 Entrées analogiques

La carte UNO dispose de 6 entrées analogiques (A0 à A5), où chacune peut fournir une mesure avec fonction analogRead () du langage Arduino sur une résolution de 10 bits (de 0 à 1023). Par défaut, ces broches mesurent une tension comprise entre le 0V correspondant au niveau 0 et le 5V correspondant au niveau 1023. Notons qu'il est possible de modifier le niveau supérieur de la plage de mesure en modifiant la tension sur la broche AREF ou en utilisant l'instruction 'analogReference' () du langage Arduino.

II 2 Le module Bluetooth

II 2 1 Présentation du module

Depuis son apparition dans les années 1990, le Bluetooth a su révolutionner la communication sans fil. C'est Ericsson qui, à l'origine, souhaitait trouver un moyen de supprimer les fils et rendre la communication entre les différents outils numériques plus faciles. Il est possible aujourd'hui de voir le Bluetooth à l'œuvre sur tous les supports : casques audio, console de jeu,

téléphone portable, enceinte… Ici nous allons parler de la voie Bluetooth série à ordinateur pour le logiciel Arduino. Les fils USB empêchent une certaine mobilité pour y résoudre il suffit de paramétrer un module Bluetooth (Bluetooth HC-05). Ce module communique avec une carte Arduino par une liaison série. Elle s'établit sur deux broches RX et TX. Ce programme permet la réception et l'envoi de données par le moniteur de série vers un autre périphérique Bluetooth (PC, Smartphone ou autre logiciel Arduino ouvert). Voici le comment connecter le module Bluetooth HC-05 Arduino.

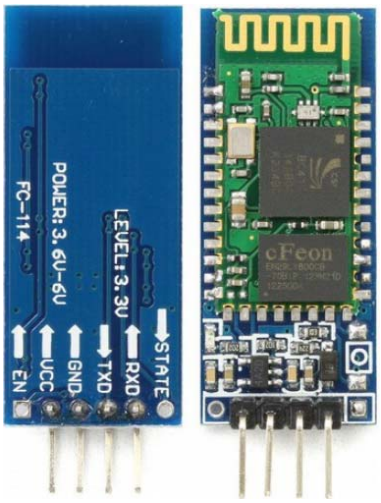


Figure II.3 : Photo réelle du module Bluetooth CH-05.

Le module Bluetooth HC-05 se trouve partout sur internet. Il fait la taille d'un doigt. Il se présente comme un montage de module Bluetooth sur un PCB.

Il évite donc de souder le module, opération délicate entraînant parfois la perte du matériel si ce n'est pas bien maîtrisé. Pas besoin non plus d'ajouter des « LEDs » de signal qui sont directement intégrées dans le dispositif.

Le dispositif est composé de plusieurs broches : GND et VCC concernant l'alimentation, RX/TX pour le pôle communication. La broche « Key » permet de gérer ses envois de commandes de configuration au module. La dernière broche, « LED », permet d'obtenir le signal du module.

II 2 2 Branchement du module à Arduino

1. Relier le VCC au 5V de l'Arduino et le GND à la masse.
2. Connecter les broches de transmission RX et TX aux broches de liaison 0 et 1 d'Arduino.
3. Brancher la broche LED et si besoin utiliser le mode « commande » du module. On pout reconfigurer la parité, la vitesse, et autre liée au Bluetooth. On peut ainsi connecter n'importe quelle sortie numérique de l'Arduino.
4. Quand le module est connecté, la connexion entre le logiciel et le HC-05 est établie. Rien de plus simple : simuler une voie de série afin de pouvoir utiliser deux broches

numériques. Vous pouvez coder la nouvelle voie série vous-même ou utiliser une existante. ‘SoftwareSerial’ est le meilleur exemple, puisqu’il est reconnu à travers le monde.

5. La LED permet de savoir si la connexion et la communication s’est bien faite entre les deux supports [9].

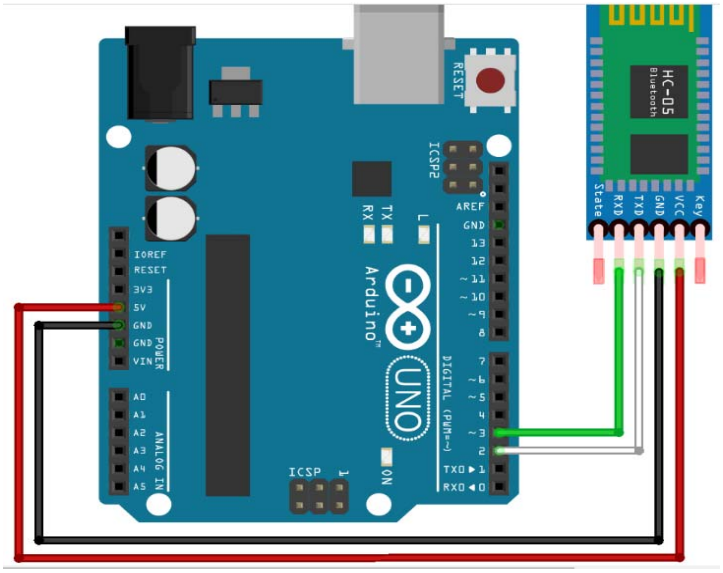


Figure II.4 : Branchement du module Bluetooth CH-05.

II 3 Afficheur LCD

II 3 1 Définition

Afficheurs LCD (LiquidCrystal Display) sont des modules compacts intelligents et nécessitent peu de composants externes pour un bon fonctionnement. Il est constitué de deux lames de verre, distantes de 20 µm environ, sur lesquelles sont dessinées les mantisses formant les caractères. L'application entre les deux faces d'une tension alternative basse fréquence de quelques volts (3 à 5 V) le rend absorbant. Un afficheur à cristaux liquides réflectif ne peut être utilisé qu'avec un bon éclairage ambiant. Sa lisibilité augmente avec l'éclairage.

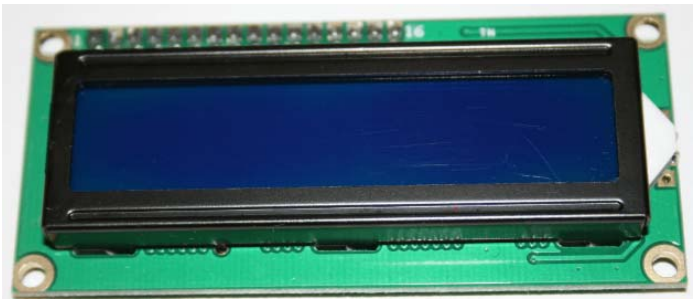
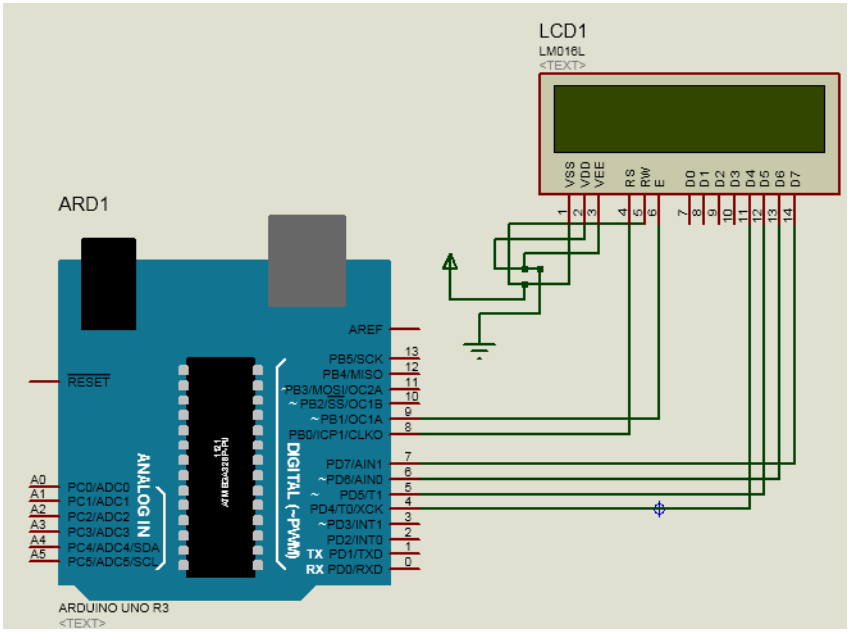


Figure II.5 : Afficheur 16x2 LCD

II 3 2 Schéma de câblage

L'écran LCD 16×2 présente 16 broches pour permettre la gestion de l'affichage et du contraste.

- VSS Relier à la masse de l'écran
- VDD Broche d'alimentation. Typiquement connectée à la broche 5V de l'Arduino.
- V0 Broche de contraste. Connecté à une sortie PWM ou à un potentiomètre.
- RS Register Select. Permet de sélectionner la zone mémoire.
- RW Read or Write. Toujours à la masse.
- E enable. Active ou non l'affichage
- D0 mode 8bits.4 bits de poids fort de la communication I2C
- D1 mode 8bits.4 bits de poids fort de la communication I2C
- D2 mode 8bits.4 bits de poids fort de la communication I2C
- D3 mode 8bits.4 bits de poids fort de la communication I2C
- D4 4 bits de poids faible de la communication I2C
- D5 4 bits de poids faible de la communication I2C
- D6 4 bits de poids faible de la communication I2C
- D7 4 bits de poids faible de la communication I2C
- A anode. borne + de la LED de rétroéclairage
- K katode. borne – de la LED de rétroéclairage



FigureII.6 : Branchement d'LCD avec Arduino

II 3 3 Code programmation

Une fois votre module correctement branché, vous pouvez modifier le code suivant pour obtenir la fonctionnalité désirée. Dans l'exemple suivant, nous créons un compte à rebours.

Pour gérer l'écran LCD 16×2 dans le programme, la librairie utilisée est « LiquidCrystal.h. »

- LiquidCrystal LCD (rs, en, d4, d5, d6, d7) pour définir la communication i2c 4bits
- LCD. Begin (16, 2) ; affiche le texte sur les deux lignes
- lcd.print () pour afficher une chaine de caractères en ASCII
- lcd.write() pour afficher des données, un octet à la fois.
- lcd. SetCursor (x, y) pour placer le curseur (colonne x : 0-16, lignes-y :0-2)
- lcd. clear() effacer ce qui est affiché à l'écran

II 3 4 MOSFET de puissance IRF520 canal N

L'IRF**520** est un '**Power Mosfet'** avec un courant de collecteur de 9,2 A et une tension de claquage de 100 V. Le MOSFET a une tension de seuil de grille basse de 4V et est donc couramment utilisé avec des microcontrôleurs comme Arduino pour la commutation de charges à courant élevé.

FigureII.7 Mosfet IRF520.

Le tableau suivant montre la configuration des broches du mosfet : [8]

Tableau II.1 Configuration des broches du IRF520

Code PIN	Nom de la broche	La description
1	La source	Le courant s'écoule par la source
2	Porte	Contrôle la polarisation du MOSFET
3	Drainer	Le courant passe par le drain

II 3 5 La connexion RS232

La communication Série (ou RS-232) est un type de communication très commun et utilisé en électronique. Anciennement, on utilisait ce type de communication sur les

ordinateurs de bureau (connecteur de type DB-9) pour communiquer avec les différents périphériques externes. Contrairement au port parallèle, le port série utilise seulement deux fils, soit RX et TX pour transférer de l'information.

TX : Transmetteur

RX : Récepteur

Le protocole de la communication série est très simple à comprendre et à programmer. La transmission de données commence par un "Start Bit" qui annonce l'envoi des bits de données. Pour terminer l'échange, il y a un "Stop Bit" qui conclut la communication de l'information. La plupart du temps, on transmet 8 bits de données (1 octet) à la fois, malgré que le protocole nous permette d'envoyer de 5 à 8 bits. Il faut toujours envoyer le bit le moins significatif (LSB) en premier et finir par le bit le plus significatif (MSB). On peut aussi ajouter un bit de parité après l'envoi des données pour assurer la validité de l'information. Finalement, on peut mettre 1 ou 2 "Stop Bit" tout dépendant de votre application. La figure suivante montre la structure de la connexion RS232.

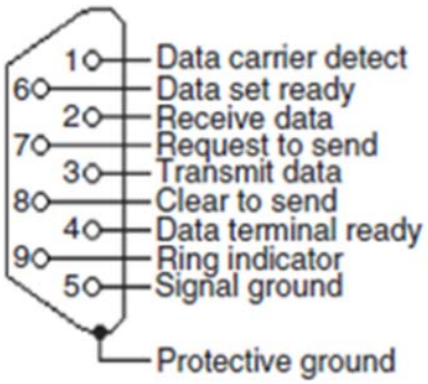


Figure II.8: Structure de RS232 [11].

III Implémentation logiciel

III 1 Logicielles utilisés

III 1 1 Arduino IDE (Integrated développement environnement)

C'est un langage de programmation Open Source écrit en ''C''. Il a toujours été gratuit à Télécharger de l'Internet pour aider les étudiants et les ingénieurs à réaliser leurs projets.

III 1 2 PROTEUS

Proteus (Processor *TExt Easy to Use*) est un langage de programmation procédural entièrement fonctionnel créé en 1998 par 'Simone Zanella'.

Proteus est une suite de logicielle permettant la CAO électronique éditée par la société « Labcenter Electronics ». Proteus est composé de deux logiciels principaux : ISIS, permettant

entre autres la création de schémas et la simulation électrique, et ARES, dédié à la création de circuits imprimés [13].

Cette fonctionnalité permet de réaliser des circuits de faible complexité en plaçant les composants et en traçant les pistes directement**.** Une fois les connections établies, il est possible d'effectuer un routage automatique des pistes.

Dans ce logiciel vous pouvez également créer de nouveaux boitiers et les placer dans une bibliothèque.

III 1 3 LabVIEW

Langages graphiques de haut niveau (ex : Simulink, Flow Code, **LabVIEW**)

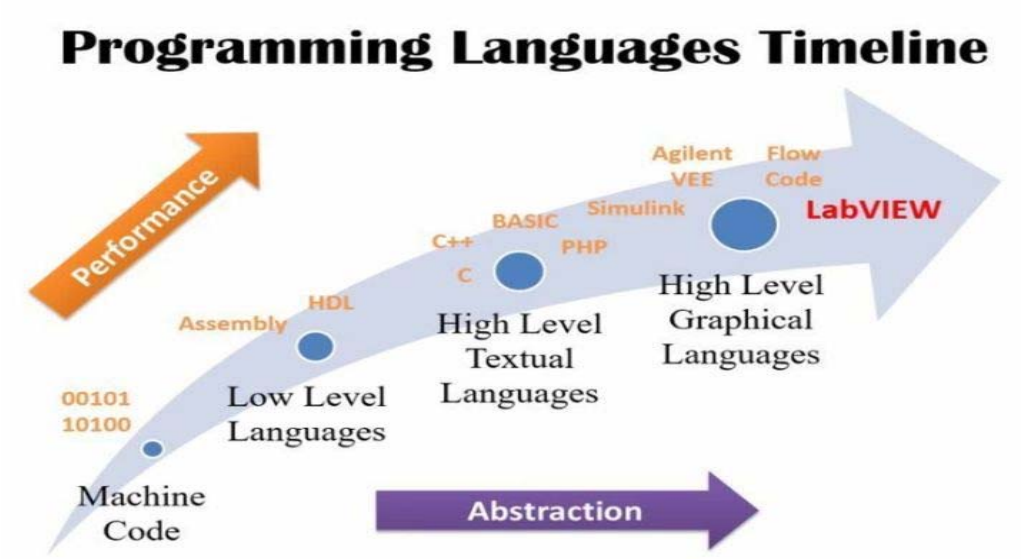


Figure II.9: Chronologie des langages de programmation

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) est un Environnement de développement graphique qui permet de créer des applications modulaires (notion de sous VI) et extensibles pour la conception d'applications, le contrôle et le test.

La description détaillée de ce logiciel est citée dans le chapitre trois.

III 1 4 Pilote de Port Série Virtuel 'VSPD' (Virtual Serial Port Driver)

Le simulateur de port série permet de créer un grand nombre de ports COM virtuels et offre la possibilité de simuler totalement le comportement des ports séries.

VSPD (Virtual Serial Port Driver) est plus qu'un simple simulateur de ports COM, il permet :

- La création flexible des ports.
- Supporte les lignes de contrôle et le transfert de données à haut débit via les ports virtuels.
- Les applications peuvent échanger des données sur des ports virtuels via un câble virtuel de type nul modem.

- Les données envoyées d'un port à l'autre seront transmises instantanément.[5]

On lui utilise pour connecter LabVIEW avec Proteus et faire les divers échanges de donnés. Le simulateur nous affiche le nom et le numéro de port utilisé …il suffit juste affecter le numéro au port correspondant et régler le baud rate (vitesse de transmission).

Figure II.10: Figure montre le rôle du VSPD.

III 2 Langage de programmation

Un langage de programmation est un langage permettant d'écrire un ensemble d'instructions qui seront converties en langage machine grâce à un compilateur. L'avantage du langage Arduino est qu'il est basé sur les langages C/C++ qui supporte toutes les syntaxes standards du langage C et quelques-unes des outils du C++. En plus, la disponibilité des librairies permet de faciliter la communication avec le matériel connecté à la carte (Afficheurs LCD, Afficheurs 7 segments, capteurs, servomoteurs… etc.). Pour écrire un programme avec le langage Arduino, il faut respecter certaines règles. Notons que l'exécution d'un programme Arduino s'effectue d'une façon séquentielle, c'est-à-dire que les instructions sont exécutées les unes à la suite des autres. Avant tout le compilateur doit vérifier l'existence de deux structures obligatoires à tout programme en langage Arduino qui sont :

- La partie initialisation et configuration des entrées/sorties (la fonction setup ()).
- La partie principale qui s'exécute en boucle (la fonction Loop ()).

III 2 1 Structure générale d'un programme en langage Arduino

Pour utiliser l'IDE standard Arduino (arduino.exe), il suffit de saisir le code dans la fenêtre dédiée, de compiler et de téléverser le programme sur la carte Arduino. La carte doit être reliée à l'ordinateur via un câble USB. Le modèle de la carte Arduino ainsi que le port série sur lequel elle est branchée doivent être déclarés dans le menu de l'IDE Outils/type de carte et Outils/port série. Une fois le programme compilé téléversé dans le microcontrôleur, son exécution sera lancée. La fonction setup () s'exécute une seule fois après la mise sous tension et après chaque redémarrage. Par contre la fonction Loop () s'exécute en boucle. La figure I.3 illustre la structure générale d'un programme IDE.

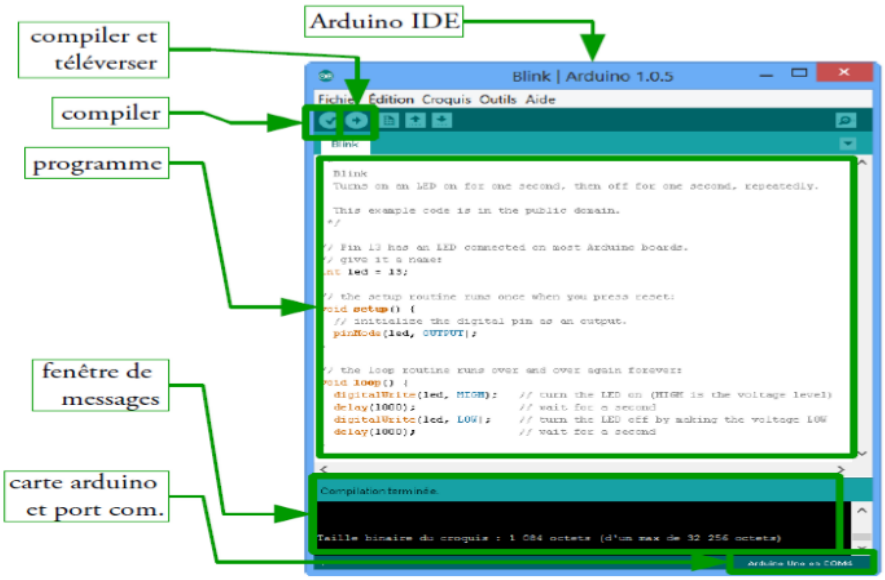


Figure II.11 : Structure d'un programme Arduino.

Un programme Arduino comporte trois parties principales :

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

III 2 2 Etapes de l'implémentation sur le microcontrôleur

1. La création d'un projet.
2. L'écriture du programme ensuite enregistrement.
3. La vérification de la syntaxe et correction d'éventuelles erreurs.
4. Le téléversement vers le microcontrôleur.

III 3 Raisons de choisir Arduino

- **Le prix :** en vue des performances qu'elles offrent, les cartes Arduino sont relativement peu couteuses.
- **La liberté :** C'est un bien grand mot, mais elle définit de façon assez concise l'esprit de l'Arduino. Elle constitue en elle-même deux choses :
    - Le logiciel : gratuit et open source, développé en Java, dont la simplicité d'utilisation relève du savoir cliquer sur la souris.
    - Le matériel : cartes électroniques dont les schémas sont en libre circulation sur internet.
- **La compatibilité** : le logiciel, tout comme la carte, est compatible sous les plateformes les plus courantes (Windows, Linux et Mac), contrairement aux autres outils de programmation du commerce qui ne sont, en général, compatibles qu'avec

Windows. La communauté ARDUINO est impressionnante et le nombre de ressources à son sujet est en constante évolution sur internet. De plus, on trouve les références du langage ARDUINO ainsi qu'une page complète de tutoriels sur le site arduino.cc.

III 4 Leurs applications

Le système Arduino nous permet de réaliser une multitude des projets, qui ont une application dans tous les domaines, l'étendue de l'utilisation de l'ARDUINIO est gigantesque. Des exemples pour leur utilisation :

- Contrôler les appareils domestiques
- Fabriquer votre propre robot.
- Faire un jeu de lumières.
- Communiquer avec l'ordinateur.
- Télécommander un appareil mobile (modélisme).
- Acquisition et contrôle des donnés, c'est notre cas.
- etc.

Note : La carte UNO peut-être alimentée soit via le port USB ou à l'aide d'une alimentation externe qui consiste à brancher une batterie au connecteur qui s'appelle (prise jack).

IV Conclusion

La réalisation d'un travail passe d'abord par la description du matériel utilisé, dans ce chapitre nous avons mis au point les caractéristiques de chaque composant et le rôle qu'il doit jouer dans la réalisation de notre projet ainsi que les logiciels permettent le bon déroulement de cette simulation, nous pouvons ensuite élaborer une stratégie de conception matériel et logiciel que nous aborderons dans le chapitre suivant.

III : Arduino et LabVIEW et communication bidirectionnelle

I Introduction

Le logiciel LabVIEW est une "Plate-forme Expérimentale d'Instruments Virtuels pour Laboratoire : « Laboratory Virtual Instrument Engineering Workbench ». C'est un environnement de programmation commercialisé par la société « National Instruments » permettant de coder à l'aide de diagramme. On peut grâce à ce logiciel créer des outils de mesure et de contrôle souvent très utilisés dans des projets de recherches. On peut contrôler et commander un processus physique externe allant du simple capteur ou actionneur à la chaîne de fabrication.

II Définition

Le LabVIEW est un outil d'acquisition, d'analyse et de présentation de données comme il est illustré dans le tableau suivant :

Tableau III .1 : Fonctionnalités de LabVIEW.

Acquisition	Analyse	Présentation
Contrôle d'instruments	Traitements numériques	Affichage de données
GPIB IEEE 488 RS 232 VXI	-Génération des signaux -Filtrage, fenêtrage -Analyse fréquentielle	-Interface interactive -Graphiques et courbes
Acquisition de données	Traitement statistique	Stockage de données
E/S analogiques E/S numériques	-Régression, lissage -Moyenne, écart type	-Archivage -Impression

Il possède un langage graphique (le langage G), des bibliothèques de fonctions et de sous-programmes, ainsi que des outils de développement.

III La programmation graphique

Contrairement à la nature séquentielle des langages textuels, LabVIEW est basé sur un environnement de programmation graphique utilisant la notion flot de donnée pour ordonnancer les opérations, où le flot de donnée détermine l'ordre d'exécution du programme. Dans LabVIEW, on crée une interface utilisateur en utilisant un ensemble d'outils et d'objets. L'interface utilisateur ou interface interactive du programme est connue sous le nom de panneau-avant où apparaissent

des objets sous forme de commandes d'entrée ou contrôleurs (Controls) ou d'indicateurs de sortie (Indicators), ces objets peuvent prendre à l'écran l'apparence d'un appareil de mesure, d'où vient l'appellation 'Instrument Virtuel' (Virtuel Instrument en anglais, ou « VI »)

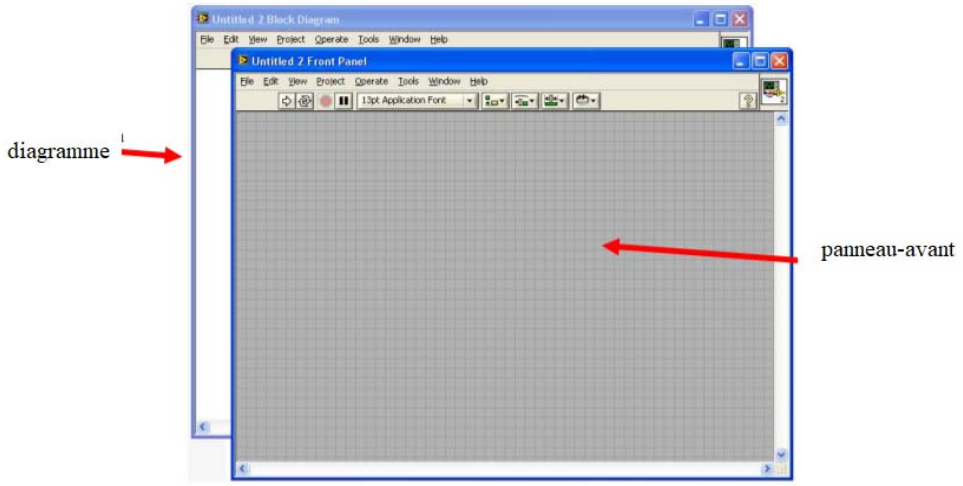


Figure III.1 : Panneau-avant et bloc de diagramme

Un programme écrit en langage graphique sera utilisable de façon interactive dès que son panneau-avant sera visible.

Le schéma fonctionnel contient ce programme. À certains égards, le diagramme ressemble à un organigramme.

Les utilisateurs peuvent contrôler le programme, modifier les entrées et voir les données mises à jour en temps réel grâce à l'environnement à multiples fenêtres. Chaque commande ou indicateur du panneau avant a une borne correspondante sur le schéma fonctionnel. Lorsqu'un VI est exécuté, les valeurs des contrôles circulent dans le diagramme, où elles sont utilisées dans les fonctions, et les résultats sont transmis à d'autres fonctions ou indicateurs via des fils.

III 1 Les palettes : Un clic sur le bouton droit de la souris fait apparaitre ces palettes de chaque fenêtre du VI.

III 1 1 Une palette de commande : accessible depuis le panneau avant contient toutes les commandes et indicateurs nécessaires à la création de la face-avant. Ils correspondent à des types prédéfinis du langage (Numérique, Booléen, chaines, Listes, tableaux & Matrices, Graphe, …), à des symboles de décoration divers (Décorations) ou à des types définis par l'utilisateur (Select controls …) et des types prédéfinis du langage.

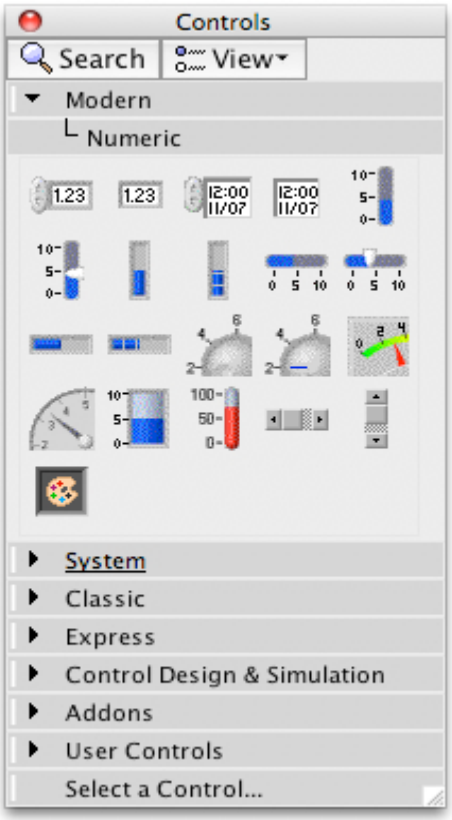


Figure III.2 : Palette de commandes

En fait, sur cette palette, les objets sont représentés plusieurs fois. La forme la plus synthétique de la Palette de commandes est représentée sur la Figure II.2. Schématiquement, on peut dire que les quatre choix « Modern », « System », « Classic » et « Express » représentent quatre styles graphiques différents, mais rien de plus. Les objets LabVIEW peuvent être pris dans l'un des quatre, seule leur apparence physique sur le « panneau avant » sera différente. Leurs fonctions seront identiques. Ceci est souligné sur la figure III.3 qui représente quatre objets identiques (des contrôleurs numériques) d'apparence différente, mais leur fonction intrinsèque reste la même.

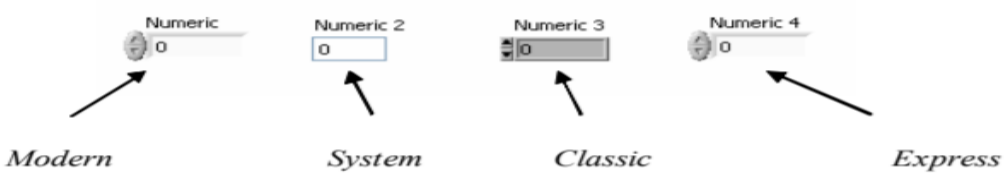


Figure III.3: Les quatre types de contrôleur numérique disponibles

III 1 2 Une palette de fonctions

Accessible depuis la fenêtre du diagramme contient tous les éléments (fonctions de base, VI Express, etc.) nécessaires à la création du code graphique. Les principaux choix offerts par cette

palette correspondent à des opérateurs ou des fonctions prédéfinies du langage. Sans panneau avant ni diagramme, les fonctions fournissent du code machine en ligne et sont pour la plupart polymorphes, c'est à dire qu'elles s'adaptent aux types (réels, entiers, tableaux de réels, tableaux d'entiers, …) et représentations des données (décimales ou hexadécimales par exemple). La forme la plus synthétique de la Palette de fonctions est représentée sur la figure III.4. Cette palette regroupe toutes les fonctions qui peuvent être effectuées par LabVIEW.

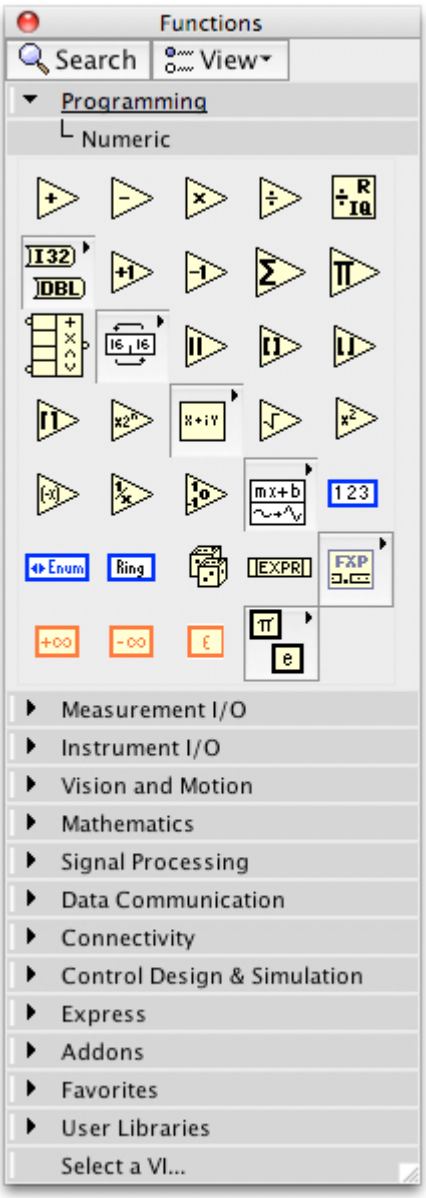


Figure III.4: Palette de fonctions

III 1 3 Une palette d'outils (Tools)

Utilisable aussi bien pour le diagramme que pour le panneau avant apparaît par défaut lorsqu'on ouvre un VI et reste néanmoins toujours accessible par le choix Tools Palette du menu 'View' si on l'a supprimée. Elle permet de définir divers modes de fonctionnement du curseur :

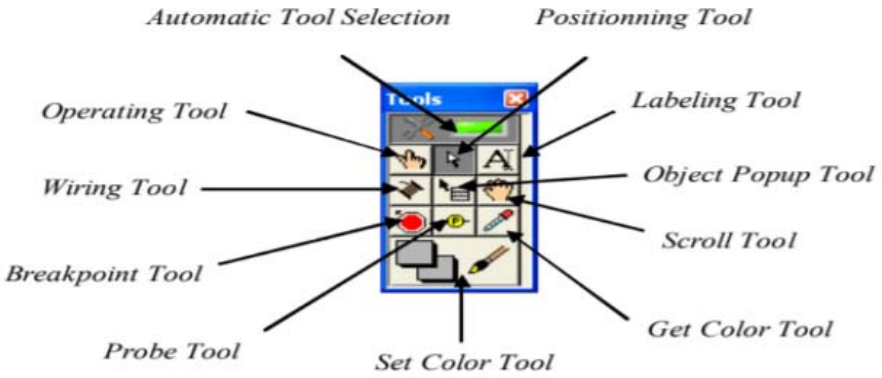


Figure III.5: Palette d'outils

III 2 Structures de données variables

Afin de les identifier plus facilement dans les diagrammes, chaque type de variables possède une couleur Existante :

III 3 Structures de programmation dans LabVIEW

Les structures de programmation sont des représentations graphiques de boucles et de conditions. Elles sont utilisées pour répéter des blocs de code et exécuter le code de manière conditionnelle ou dans un ordre spécifique. Ces structures se divisent en trois types : séquence, condition, itération.

III 3 1 Structure séquence

C'est un moyen pour imposer l'ordre d'exécution des tâches, leurs cadres ressemblent à des diapositives placées les uns derrières les autres ; l'exécution commence par le code contenu dans la première (N°1) puis continue dans l'ordre 1, 2, etc.

III 3 2 Structure Conditionnelle

Elle peut correspondre à un if … then … else dans les langages textuels. LabVIEW représente l'alternative à l'aide de plusieurs diagrammes alternatifs

III 3 3 Structures d'itération

La boucle « while » (tant que) permet de répéter une ou plusieurs instructions N fois (N nombre connu). A l'intérieur de la boucle while se trouve un terminal d'entrée local générant l'entier indiquant l'indice d'itération de la boucle. Un terminal de sortie de type booléen permet d'arrêter la boucle lorsque la valeur du stop soit égale à 1.

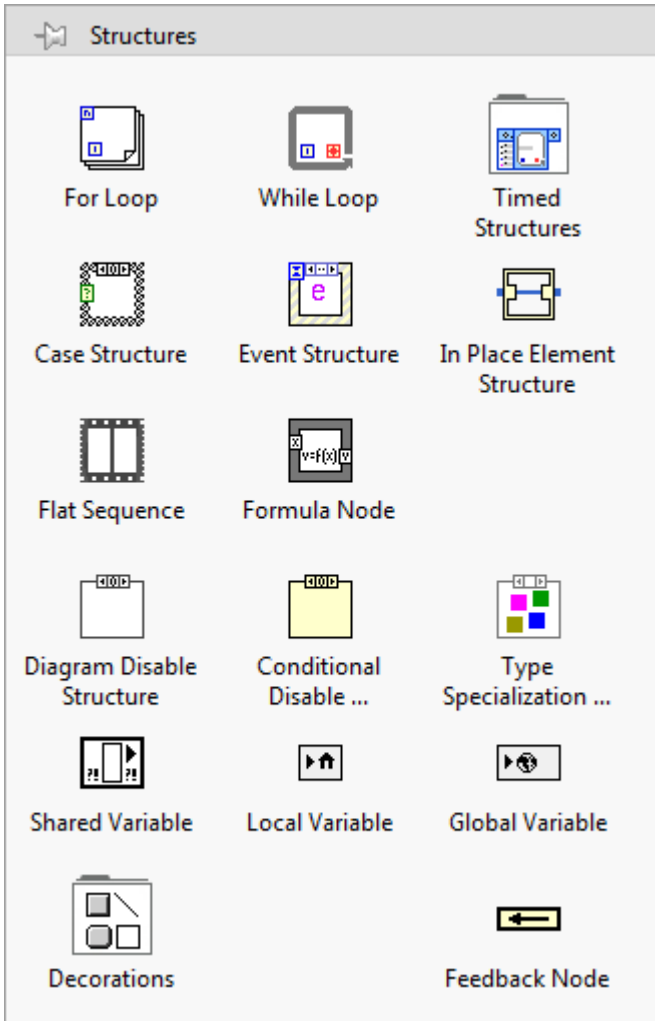


Figure III.6 : Choix 'Structures' de la palette Fonctions

III 3 4 La boucle conditionnelle : TANT QUE (While Loop)

C'est une structure permettant d'exécuter une partie de programme tant qu'une condition reste vraie. Elle correspond à la boucle do … while (condition) du Pascal par exemple. On l'utilise par exemple lorsqu'on effectue des mesures, sans savoir a priori quand on doit s'arrêter. On y accède par le choix While Loop de la sous-palette Structures de la palette Fonctions. Elle est formée d'un cadre rectangulaire que l'on étire de la même façon que les boucles décrites précédemment. L'intérieur du cadre est un sous-diagramme destiné à recevoir la partir du programme à exécuter tant que la condition de sortie le permet.

Figure III.7 : Terminaison de la boucle « While »

La boucle « While » contient deux terminaisons par défaut : une terminaison itérative fournissant à l'intérieur de la boucle le numéro de l'itération en cours (0 pour la 1ère) et une terminaison conditionnelle, testée à la fin de chaque tour de boucle et provoquant l'arrêt des itérations et la sortie de la boucle si son état est False [5].

III 4 Graphes dans LabVIEW

LabVIEW, dans son langage graphique G, offre à l'utilisateur des objets extrêmement performants, correspondant à des types de données numériques inexistants dans les autres langages, permettant l'affichage et la manipulation aisée des graphiques.

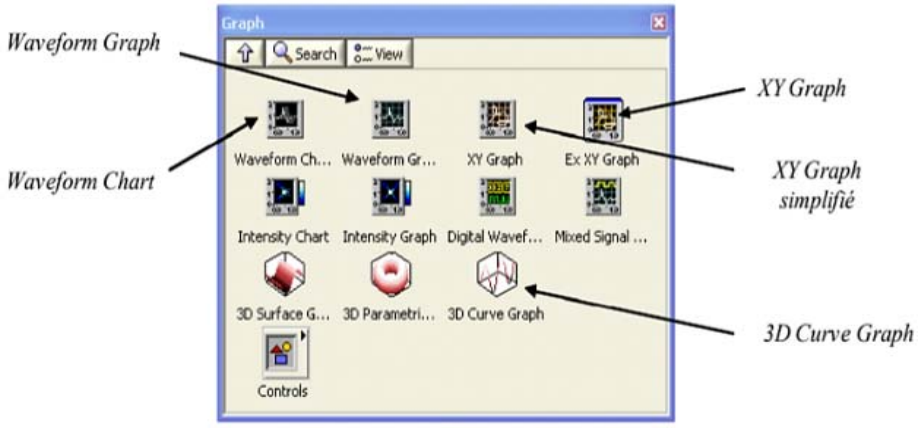


Figure III.8: Choix des Graphes de la palette 'Commandes'

Les graphes sont accessibles dans la palette de commande 'Graph Indicators'. Les graphes inscrivent les points en donnant une abscisse initiale et un incrément à chaque nouveau point. LabVIEW propose 3 principaux types de graphes :

• **Graphe déroulant** : c'est une sortie 2D représentant une courbe (ou plusieurs courbes) dont les points sont donnés point par point.

• **Graphe XY** : permet de tracer des courbes paramétriques, en annulant la base de temps. Pour ce faire, il faut envoyer un cluster contenant les points pour X, et les points pour Y sur chaque composante du cluster en entrée du graphe.

• **Graphe 3D :** comme son nom l'indique c'est une représentation tridimensionnelle de sortie.

A la création, ces afficheurs graphiques présentent sur le panneau avant une zone d'affichage de courbes, pourvue d'échelles verticale et horizontale, et proposent d'entrer immédiatement le nom (Label) du graphique à l'emplacement du curseur. Des options par défaut sont fournies pour les échelles et le style d'affichage des points, mais l'utilisateur peut personnaliser le graphique grâce aux fenêtres accessibles par le menu surgissant.

IV Transmission des données

Pour contrôler, réguler ou agir efficacement sur un processus physique, chimique ou biologique, naturel ou industriel, il faut avoir un système de transmission tel que le RS232. LabVIEW dispose de VI spéciales appelés VISA RS232, elle permet de transmettre le signal acquis via port série en code ASCII de l'ordinateur vers l'interface de commande pour un traitement ou une visualisation. [6]

V L'interface de programmation NI-VISA

LabVIEW permet de piloter des instruments GPIB ou série en allant rechercher les fonctions associées dans la palette E/S d'instruments comme représenté ci-dessous. Néanmoins, dans la plupart des cas, on utilisera les fonctions NI-VISA qui permettent de dialoguer avec un instrument en utilisant des fonctions de haut-niveau indépendantes du moyen et protocole de communication utilisé.

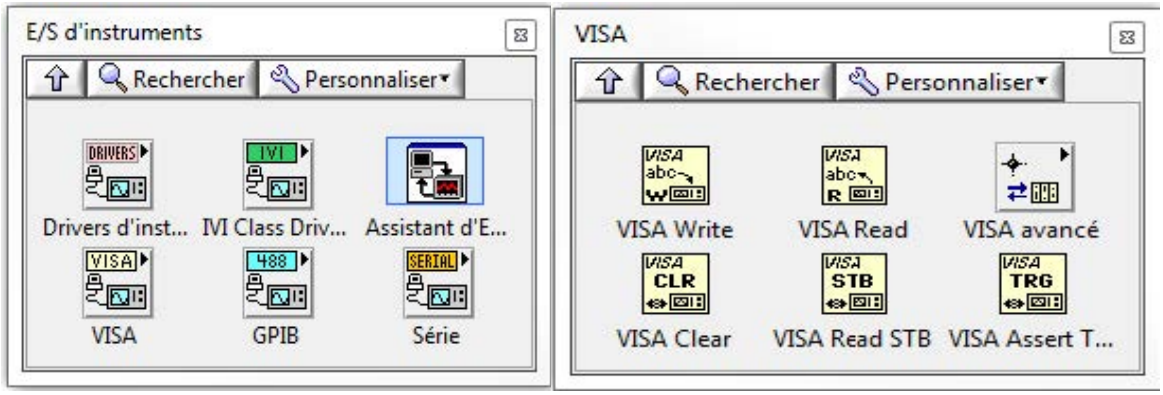


Figure III.9: L'interface d'E/S VISA

VISA est une API standard qu'on utilise pour contrôler une vaste gamme d'instruments. Cette interface est en effet capable d'appeler le driver approprié suivant le type d'instrument utilisé, on n'a donc pas à appréhender le protocole de communication utilisé par l'instrument.

La figure III.10 représente un exemple où on peut voir que la transmission RS232 se fait avec des VI bien spécifiques, tel que l'ouverture se fait par un VISA OPEN, et fermé par un VISA CLOSE, configuré avec un VISA CONTROL SERIAL PORT, on envoi des données via la VISA WRITE, et on peut lire avec un VISA READ.

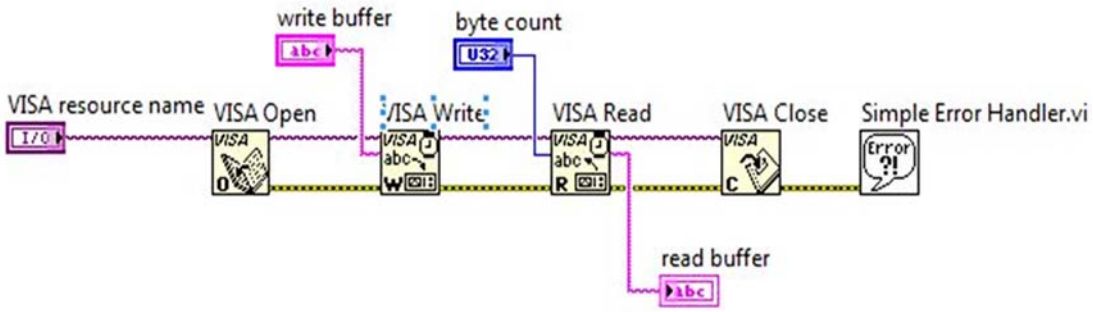


Figure III.10: Exemple de connexion RS232 dans LabVIEW

VI Conclusion

De façon générale, LabVIEW est un océan assez vaste, nous avons essayé d'aborder le minimum qui touche à notre projet, puisque l'implémentation de l'algorithme de régulation est faite au niveau de la carte Arduino, l'interface LabVIEW sert à introduire les données et à visualiser les sorties.

IV : Simulation et identification du système

I Introduction

Après conception et simulation du système, il ne reste qu'à entamer sa réalisation, mais à cause de l'épidémie de COVID _19 qui présente un obstacle contre sa réalisation pratique nous allons se contenter par la simulation logicielle qu'on va l'expliquer bien dans ce chapitre.

II Réalisation des programmes

II 1 Arduino

Nous avons utilisé l'IDE Arduino pour programmer notre carte Arduino en langage C. Ce code Arduino s'exécute en synchronisation avec le programme LabVIEW (envoi/réception de données en deux sens) commandant le circuit simulé par le logiciel Proteus, le programme fonctionne comme suit :

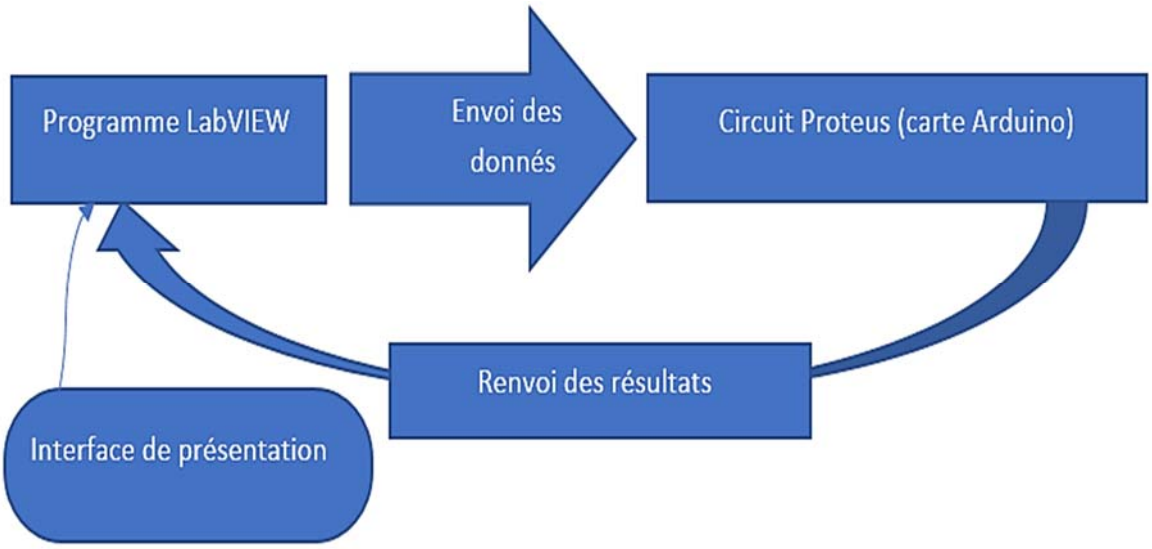


Figure IV.1 : Illustration d'un message et de son accusé entre LabVIEW et Proteus.

II 1 1 Organigramme expliquant le programme Arduino :

L'organigramme suivant explique l'algorithme du code écrit sur l'IDE d'Arduino.

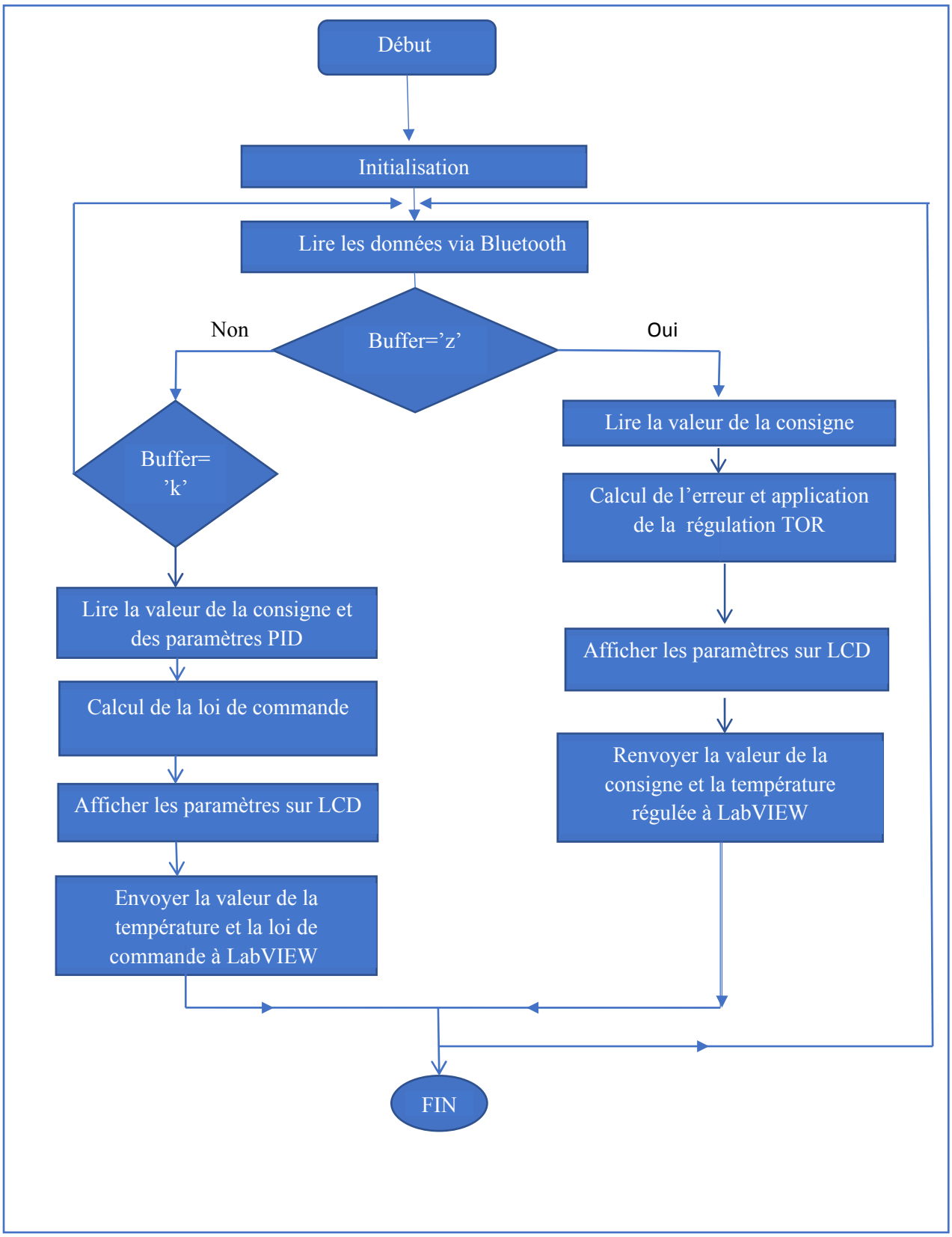


Figure IV.2 : Fonctionnement du programme Arduino.

II 1 2 Liste des commandes et leurs procédures

- **Initialisation et déclaration des constantes**

  Ici on déclare les constantes et les variables du programme.

  - ✓ Les broches d'LCD (4 5 6 7 8 9).
  - ✓ Tx :la broche numérique (0).
  - ✓ Rx :la broche numérique (1).

- **Déclaration des entrées et sorties**

  Dans cette partie on a déclaré les variables d'entrée et de sortie ainsi que la vitesse de transmission. (Baud rate).

  - ✓ ControlPin : la sortie du signal de commande (la broche numérique 3).
  - ✓ Y1 : l'entrée du capteur de température (la broche analogique A1).
  - ✓ LED d'indice :la broche numérique (13).

- **La boucle (Void)**

Partie de régulation TOR

Si le caractère envoyé par LabVIEW est 'Z' alors le processeur exécute la partie du programme 'processus TOR' (vérifier le caractère 'Z' : lire la consigne et la renvoyer comme accusé de réception faire la comparaison entre la consigne et la valeur mesurée, générer la correction adéquate et envoyer la valeur finale à l'interface d'affichage LabVIEW). Ce code va être exécutée continuellement jusqu'à changement du caractère envoyé ou arrêt du système.

Partie de régulation PID

Si le caractère envoyé par LabVIEW est 'K' alors le processeur exécute la partie du programme 'processus PID' : lire la consigne, lire les paramètres PID, mesurer la température d sortie, calculer l'erreur entre la consigne et la température mesurée, calculer la loi de commande selon l'algorithme implémenté et envoyer la température régulée et la loi de commande à LabVIEW. Bien sûr cette partie du programme va être exécuté continuellement jusqu'à changement du régulateur.

II 2 Programme sous LabVIEW

II 2 1 Organigramme expliquant le programme LabVIEW

L'organigramme suivant explique l'algorithme de mise en marche du programme LabVIEW :

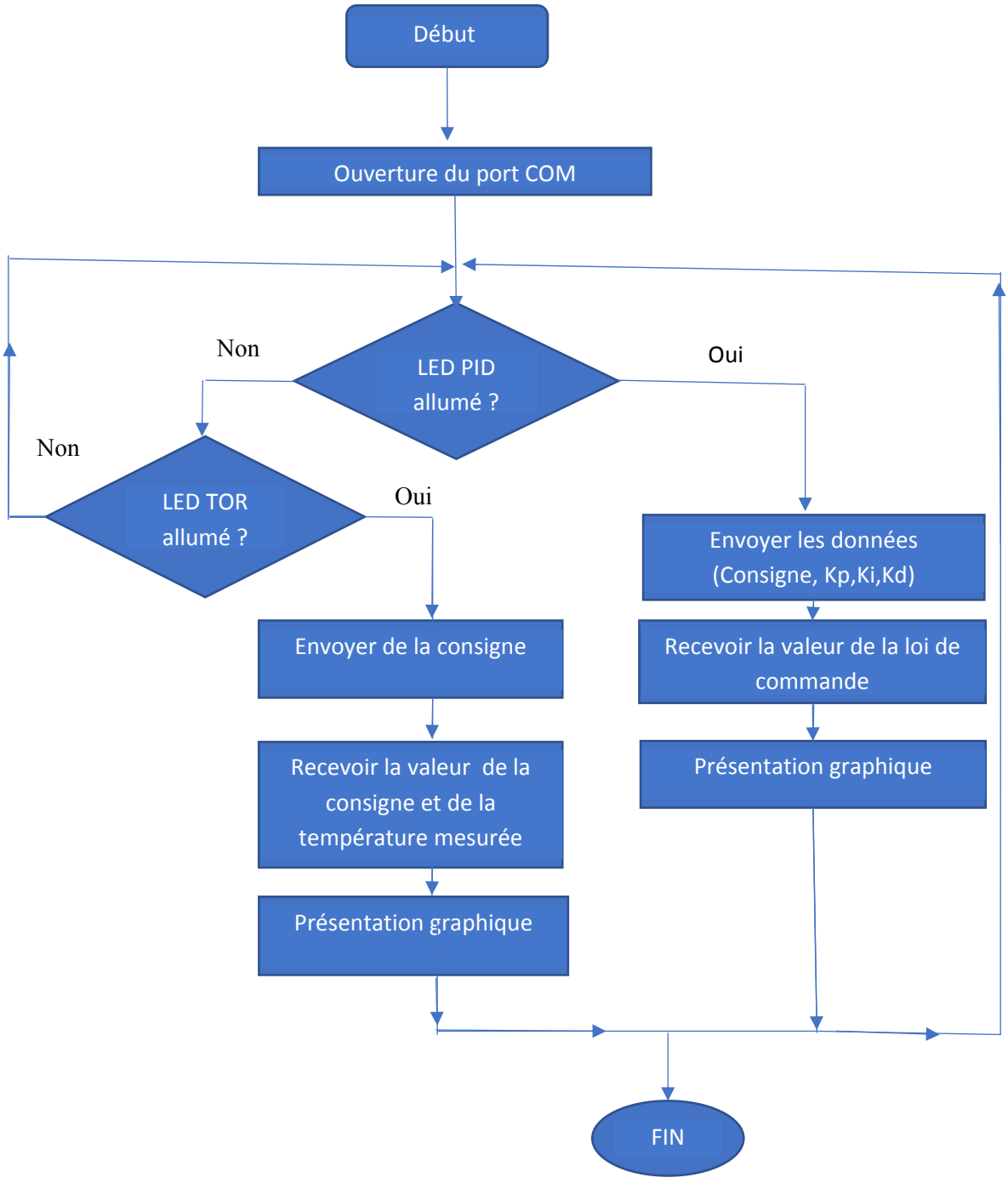


Figure IV.3 : Fonctionnement du programme LabVIEW.

II 2 2 Explication de l'organigramme

- La communication est faite via une paire de ports série COM créé par le simulateur de ports virtuels, le programme commence par l'ouverture du port associé au LabVIEW, (Ce n'est pas le cas s'il y a une réalisation pratique du projet parce qu'un un seul port suffira).
- Le choix de la régulation désirée, ce fait en appuyant sur le bouton correspondant dans le panneau avant.

- Après l'appui sur un de ces boutons, on envoie les paramètres de régulation (consigne choisie, kp, ki et kd déjà calculés) via le panneau avant.
- Le calcul se fait dans le kit Arduino, qui par la suite affichera tous les paramètres sur un écran LCD.
- L'envoi permanant des résultats de calcul par le Bluetooth vers l'interface LabVIEW qui sont : la température réelle et la valeur de la loi de commande et enfin leurs présentations graphiques. Le schéma bloc suivant montre la procédure de communication :

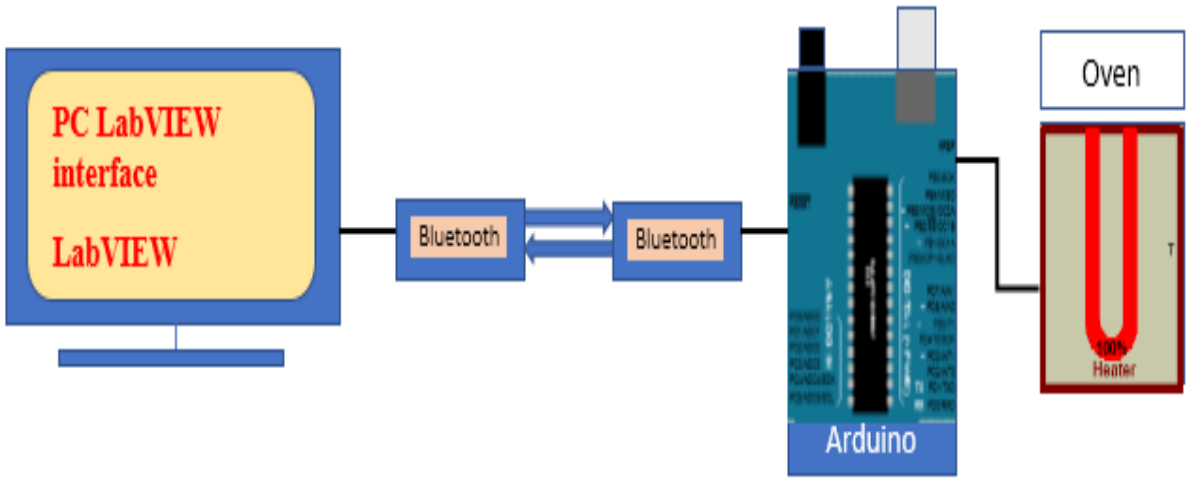


Figure IV.4 : Schéma bloc descriptive.

II 2 3 Panneau avant

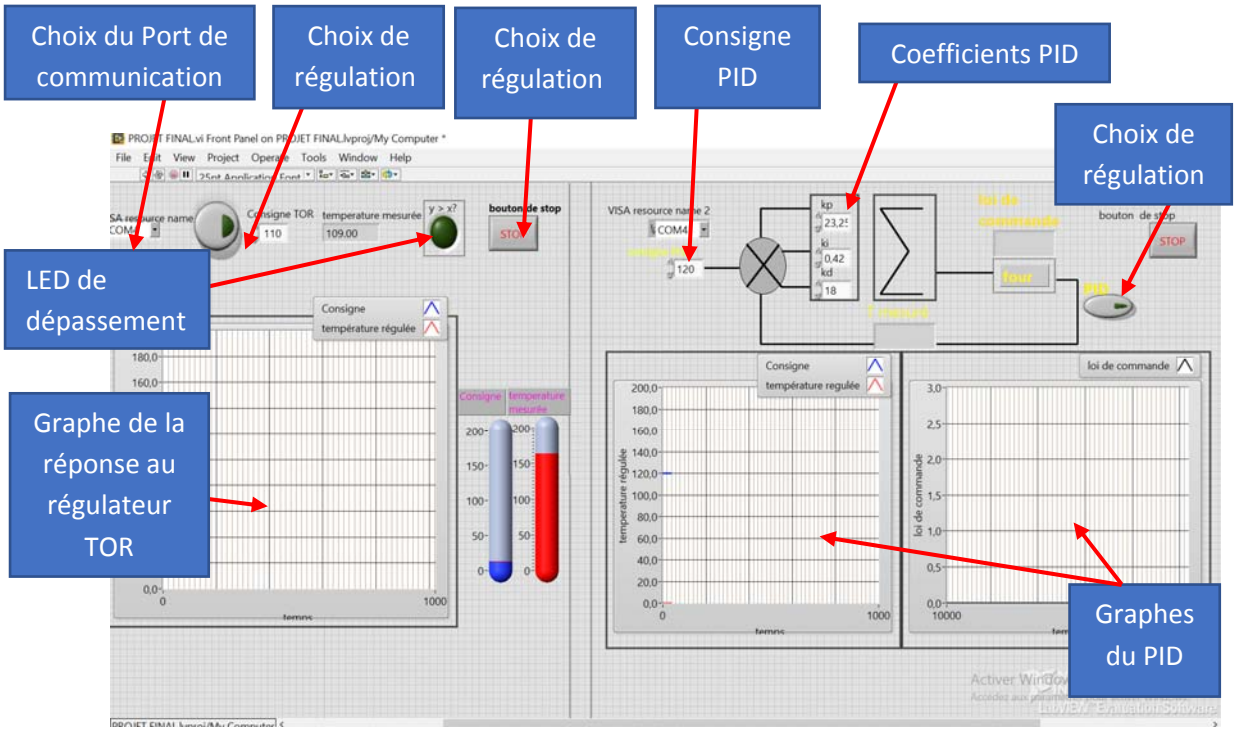


Figure IV.5 : Panneau avant d'application

Il faut tout fois prendre en compte que le programme Arduino sera en synchronisation totale avec le programme sous LabVIEW, donc il faudra respecter une certaine structure de communication (envoi → reception → renvoi d'un accusé de réception), et cette structure est appliquée par les deux programmes en marche.

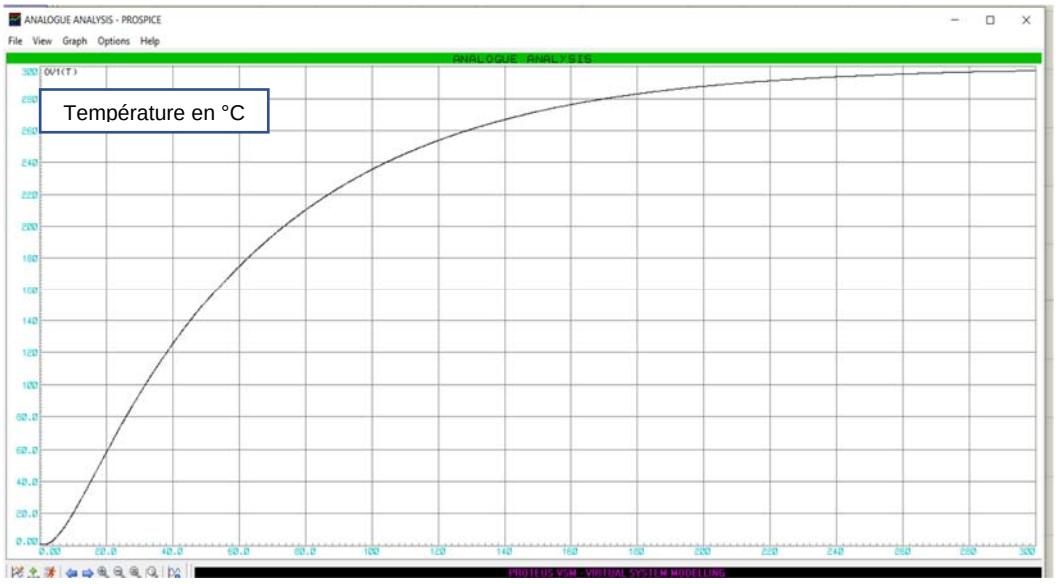
Le programme fonctionne comme suit :

III Identification et configuration du régulateur

Dans cette partie, nous allons identifier le système à réguler, en effet la mise en place finale de ce régulateur ne se fera qu'après la détermination des paramètres de ce dernier, et leur implémentation dans le programme.

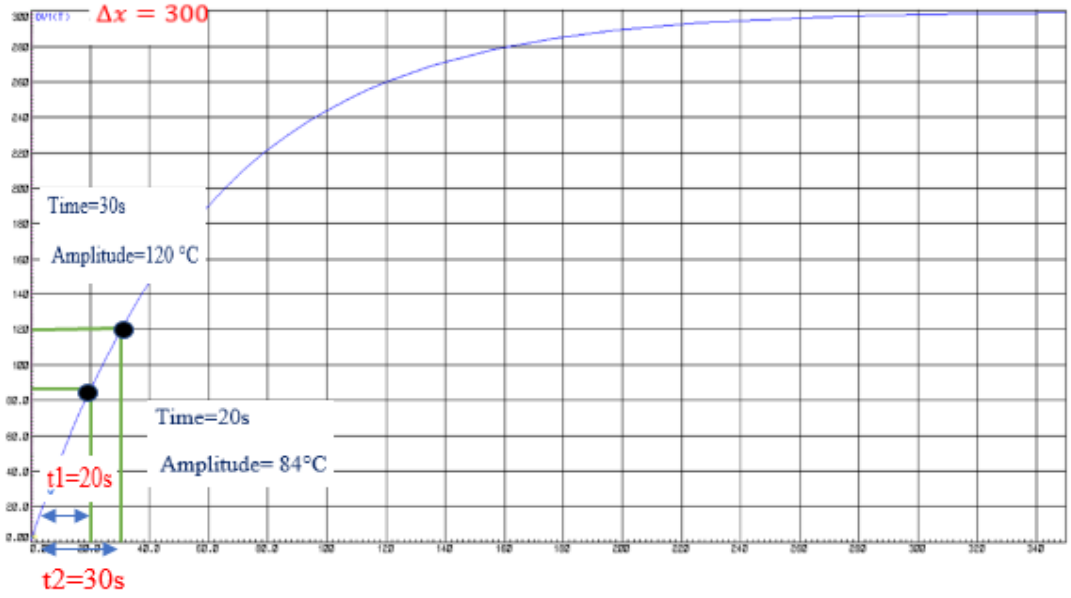
Considérons le système qu'on a pour étude : un four (OVEN), l'entrée du système est la tension ov1(t) en volts et la sortie est : la température T(t) en degré Celsius

En appliquant un signal échelon sur le système non-régulé (en boucle ouverte), nous constatons l'évolution du système comme suit :



FigureVI.6 : Réponse du système a un échelon.

C'est un système de premier ordre, en appliquant la méthode de Broïda, on peut définir sa fonction de transfert.



FigureVI.7 : Identification des paramètres du système.

Détermination du modèle de Broïda

Le model de Broïda est sous la forme : H(P)= $\frac{X(P)}{Y(P)} = \frac{Gs.e^{-\tau P}}{(1+\theta P)}$ (4.1)

Le gain $Gs = \frac{\Delta x}{\Delta y} = \frac{x(\infty)}{y(\infty)} = \frac{300}{300} = 1$ (4.2)

La constante de temps : $\theta = 5{,}5 * (t2 - t1) = 5{,}5 * (30 - 20) = 55s$ (4.3)

Le retard : $\tau = (2{,}8 * t1) - (1{,}8 * t2) = 56 - 54 = 2s$ (4.4)

- On obtient la fonction de transfert :$H(P) = \frac{X(P)}{Y(P)} = \frac{Gs.e^{-\tau P}}{(1+\theta P)} = \frac{e^{-2P}}{(1+55P)}$ (4.5)

$$H(P) = \frac{e^{-2P}}{(1+55P)} \quad (4.6)$$

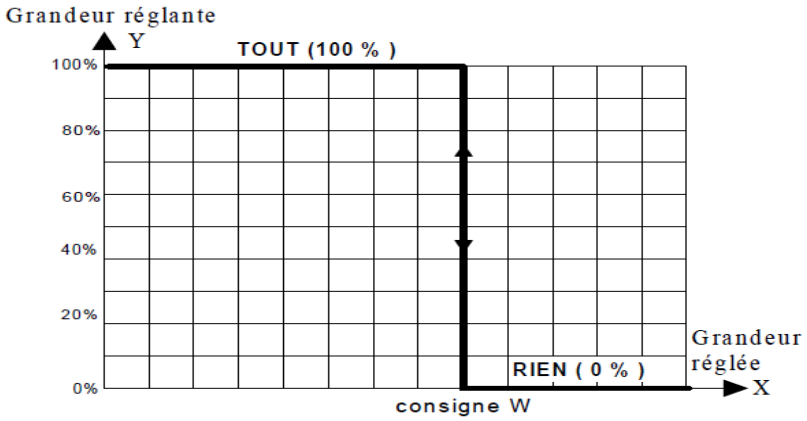
III 1 La régulation tout ou rien (BANG_BANG)

Dans ce cas, le régulateur commande le système instantanément (TOUT ou RIEN), ça veut dire la sortie du régulateur ne prend que deux valeurs 0 ou 1.

Le principe consiste à comparer la grandeur à regler avec une valeur donnée à l'avance(consigne), et le processus de reglage commence lorsque la grandeur à regler s'écarte de la valeur imposée.

$$\boldsymbol{u(t) = 100\%\ si\ y(t) < Tc}$$

$$\boldsymbol{u(t) = 0\%\ si\ y(t) > Tc}$$



FigureVI.8 : La commande TOR

Y=u(t), X=y(t)

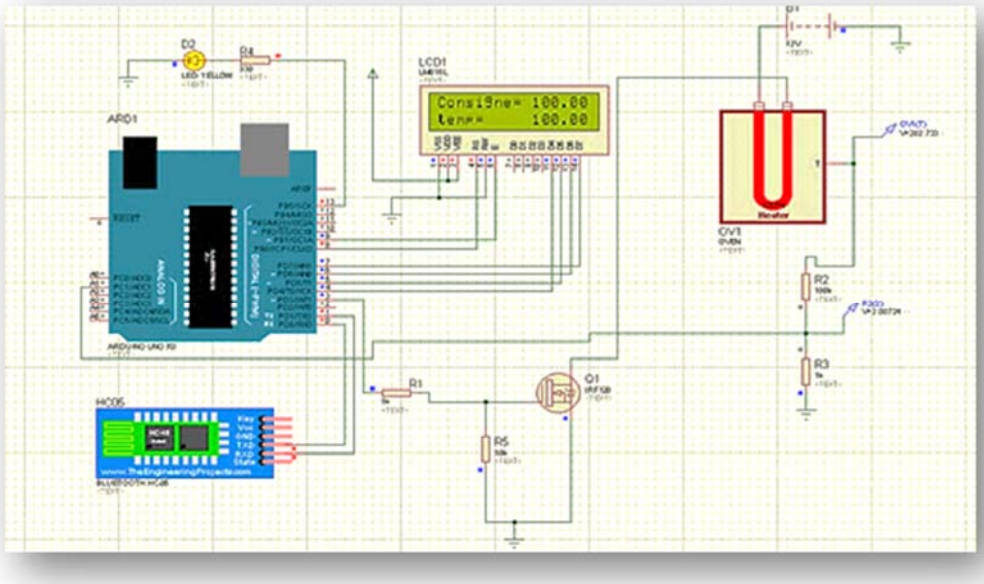
Tant que la valeur de la mesure y(t) est Inférieure à la consigne Tc, la commande u(t) est de 100 % (TOUT). Dès que la mesure y(t) atteint et dépasse la consigne Tc, la commande u(t)est de 0 % (RIEN).

III 1 1 Implémentation et simulation logiciel

Dans un premier temps, nous nous contenterons de développer sous Arduino IDE un programme qui va implémenter cet algorithme de régulation.

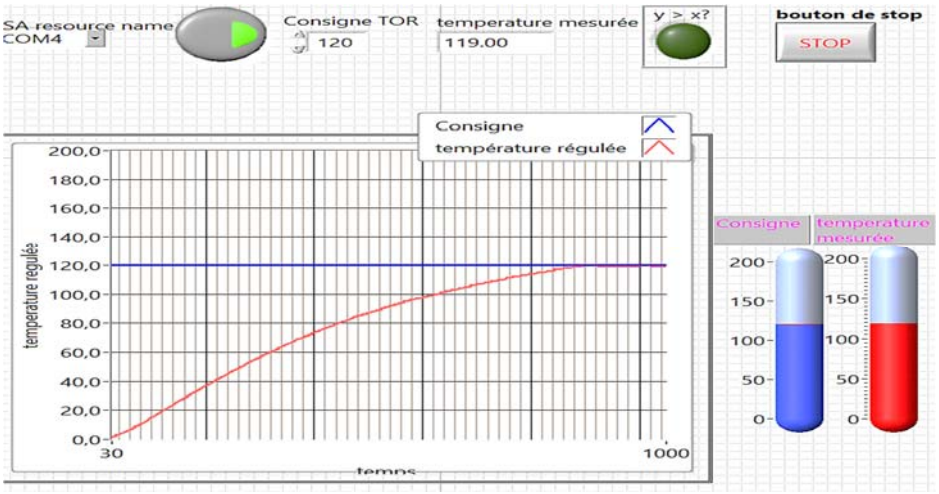
Arduino fait une lecture analogique de la température du four (OVEN) et fait une comparaison avec la température de consigne envoyée depuis LabVIEW. Par la suite il va générer une commande TOR (marche/arrêt), qui commande un relais de puissance activant le four (OVEN).

Le schéma de la simulation sous Proteus est donné par la figure suivante :



FigureVI.9 : Schéma de simulation sous Proteus.

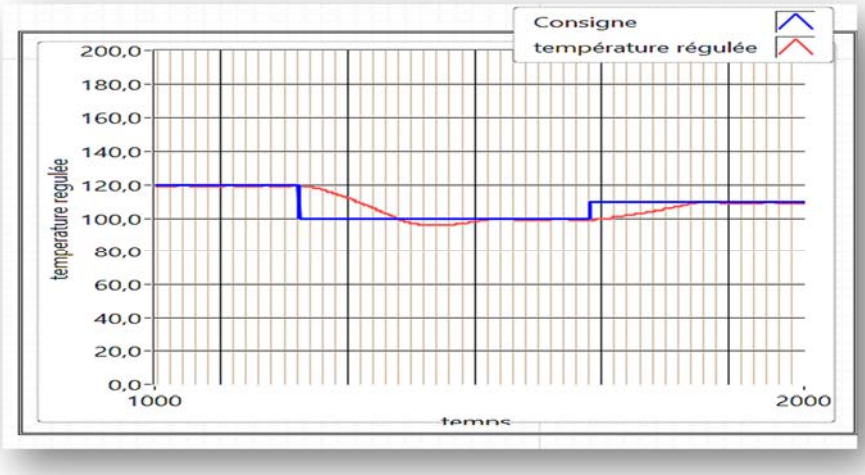
Tant que la valeur de la mesure ‘’ temp’’ est inférieure à la consigne ‘’Consigne’’, la commande est de 100 % (TOUT). Dès que la mesure atteint la consigne, la commande est de 0 % (RIEN), la LED est l’indice de dépassement s’il aura lieu et on peut la remplacer par une alarme. On obtient le graph de simulation de la commande sous le panneau avant comme suit :



FigureVI.10 : Graphe de la température régulée.

III 1 2 Interprétation des résultats

Lorsque on impose une consigne (température de référence) au système et on lance le programme, la température du four augmente jusqu’à atteindre la valeur imposée, le changement de cette valeur (la consigne) entrainerait le changement de la réponse du système qui suit la consigne.



III 2 La régulation PID

Dans ce cas, le régulateur commande le système selon une loi de commande c’est-à-dire on peut changer l’état de l’intensité de l’actionneur. La sortie du régulateur peut prendre des valeurs intermédiaires.

La régulation consiste à comparer la consigne (température de référence) imposé par le superviseur depuis l'interface LabVIEW et la température du four (obtenue grâce au capteur reliée à la broche A1 de l'Arduino). Le régulateur fait ceci de façon numérique c'est à dire échantillonné (temps discrets et grandeurs quantifiées). Il va donc périodiquement faire la différence entre la température consigne (xx) et la température réelle (yy) puis ajuster la tension aux bornes du four en fonction de cette erreur (e), il utilise 3 corrections différentes :

- ✓ Le premier moyen consiste à dire que plus l'erreur est grande et positive et plus il faut augmenter la tension aux bornes du moteur (pour rattraper cette erreur). La tension aux bornes du moteur est alors directement proportionnelle à l'erreur :

$$v = u(t) = Kp * e \quad (4.7)$$

'P' étant un coefficient de proportionnalité.

- ✓ On voit cependant apparaître des problèmes, quand l'erreur s'annule ou prend des valeurs négatives la loi de commande s'annulerait et reprenait une valeur importante assez rapidement que la réponse du système oscille autour de la valeur de référence. Il y aura donc une erreur qui subsistera et on n'atteindra pas exactement la température souhaitée. C'est pourquoi, on ajoute un autre élément qui à chaque période, fait la somme de l'erreur (intégration de l'erreur). Donc, si l'erreur reste constante, sa somme va augmenter au fil du temps, On a alors :

$$v = u(t) = Kp * e + Ki * (e(t) + e(t-1)) \quad (4.8)$$

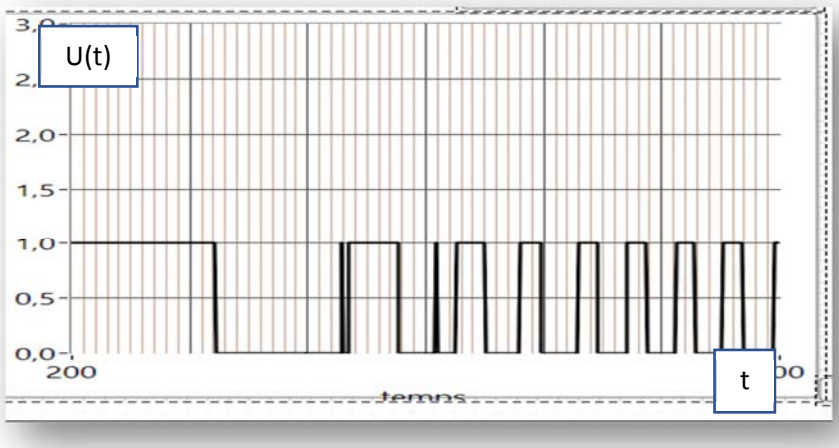
La valeur Ki est un coefficient qui va déterminer l'influence de ce paramètre intégral.

- ✓ Pendant ce temps l'erreur va continuer à augmenter et l'on va encore accroître la tension aux bornes du four alors que cela n'est pas nécessaire. Pour diminuer ce phénomène "d'emballement", on va introduire un 3ème élément qui va permettre d'en "calmer" les effets, on se base sur la différence (la variation) de l'erreur entre 2 mesures. Il s'agit de la dérivée de l'erreur. On obtient alors au final :

$$u(t) = Kp * e + Ki * \big(e(t) + e(t-1)\big) + Kd * (e(t) - e(t-1)) \quad (4.9)$$

Le régulateur commande donc le four en prenant en compte trois aspects de l'erreur entre la température consigne et la température réelle. Chacune des influences peut être réglée par la constante appropriée : Kp (proportionnel), Ki(intégral), Kd (dérivée). Il s'agit de l'utilisation d'un filtre PID numérique.

L'actionneur est commandé par le signal de sortie du régulateur numérique la troisième broche numérique de l'Arduino qui est un signal modulé en largeur d'impulsion dont l'allure est la suivante :



FigureVI.11 : Graphe du signal de commande en PMW.

A fréquence fixe le rapport cyclique est en fonction du résultat l'algorithme de calcul. Le rapport cyclique de signal de commande a une valeur 'α' avec (0<α<1).

Identification des paramètres PID

Le régulateur PID forme un signal de commande (grandeur de réglage) qui va faire varier la puissance de réglage par l'intermédiaire d'un actionneur (organe de réglage).

Nous avons déjà calculé la fonction de transfert de notre système à réguler :

$$H(P) = \frac{e^{-2P}}{(1+55P)} \qquad (4.10)$$

Les paramètres PID sont déduits à partir du tableau suivant :

Tableau : Règles de détermination des paramètres du PID

	Régulateurs P	Régulateurs PI	Régulateurs PID
Kc	$\frac{0.8\tau}{a\theta}$	$\frac{0.8\tau}{a\theta}$	$\frac{1}{1.2a}\left(\frac{\tau}{\theta}+0.4\right)$
Ti		τ	$\tau+0.4\theta$
Td			$\frac{\tau\theta}{\theta+2.5\tau}$

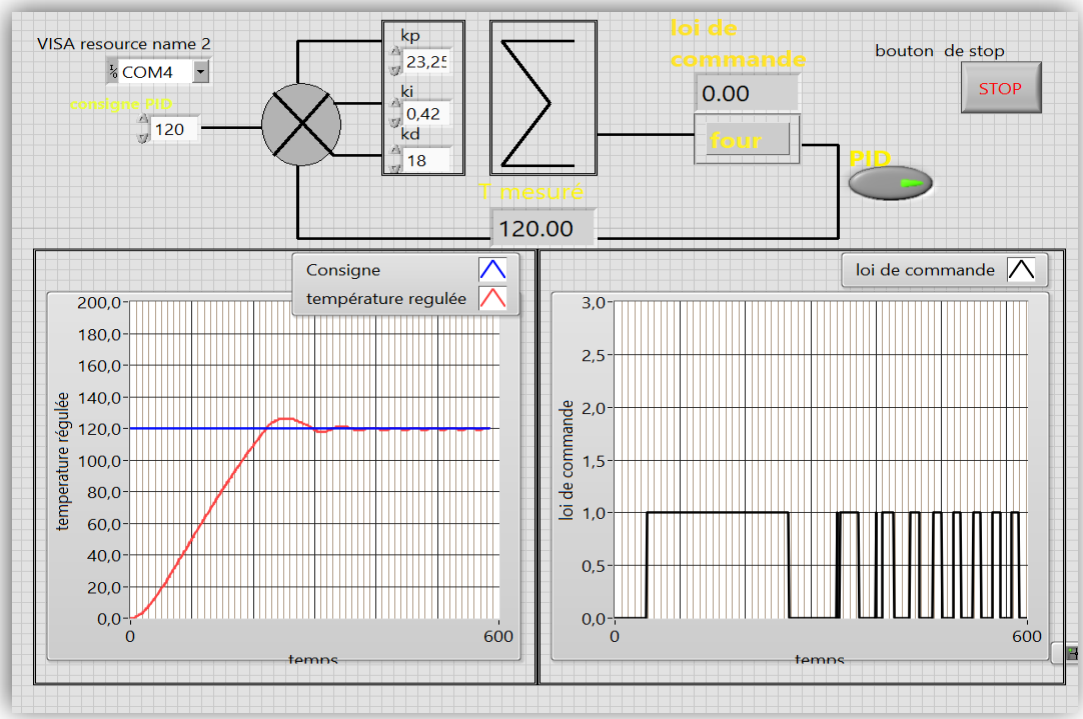
- $Kp = Kc = \frac{1}{1.2a}\left(\frac{\tau}{\theta}+0.4\right) = \frac{1}{1.2*1}\left(\frac{55}{2}+0.4\right)$ Kp=23.25 (4.11)
- $Ki = \frac{Kp}{Ti} = \frac{Kp}{\tau+0.4\theta} = \frac{23.25}{55+0.4*2}$ Ki=0.42 (4.12)

- $Kd = Kp * Td = Kp * \frac{\tau\theta}{\theta+2.5\tau} = 23.25 * \frac{55*2}{2+2.5*55}$ Kd=18.3 (4.13)

III 2 1 Implémentation et simulation logiciels

Pour activer la régulation PID il suffit d'appuyer sur le bouton PID monté sur le panneau avant du LabVIEW, Lorsque les données sont envoyées de LabVIEW à notre Arduino, ce dernier va mesurer la température du four, calculer l'erreur et générer la loi de commande, les paramètres du système s'afficheront sur l'écran LCD, le Bluetooth renvoi les résultats à LabVIEW, ces derniers seront visualisés sous forme de graphes.

III 2 2 Interprétation des résultats



FigureVI.12 : Graph de la loi de commande et la température régulée.

La figure (IV.11) représente la réponse indicielle du système corrigé par un régulateur PID.

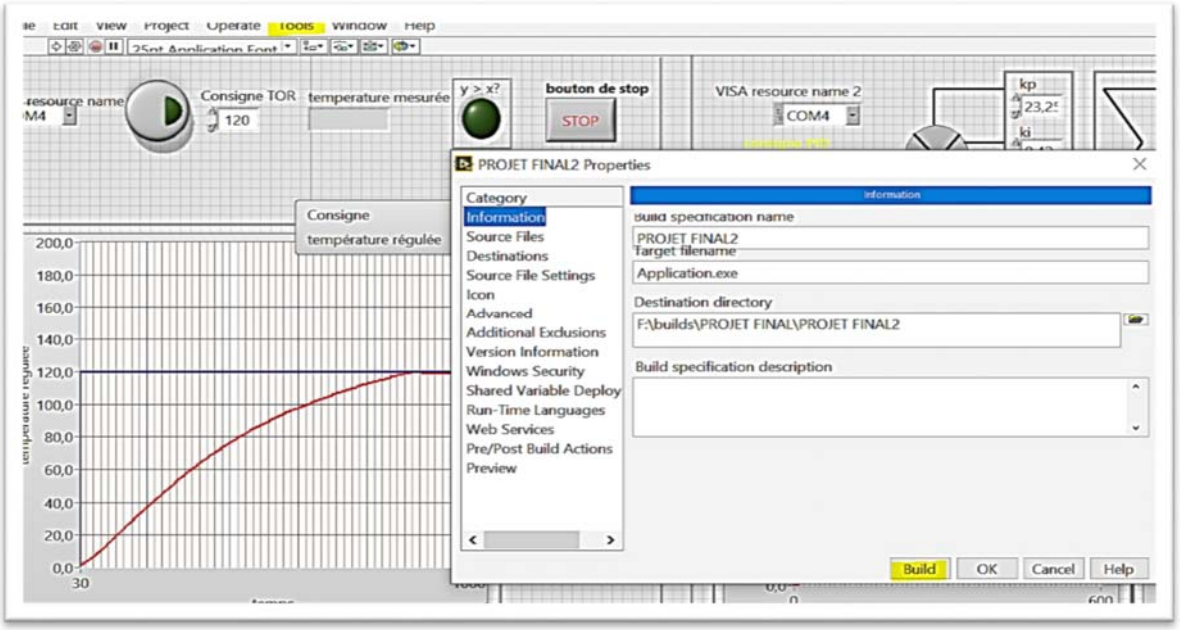
Après un temps de retard estimé par 200 ms pris par le système pour établir la connexion entre ses deux postes, la température augmente suivant la valeur de la consigne. On constate un petit dépassement ensuite la température garde une valeur constante vis-à-vis de la consigne. Grâce au régulateur PID et la commande u(t) généré en PWM. Si la température descend au-dessous de la consigne, la commande envoie une impulsion de largeur déterminée par le régulateur PID vers le four.

L'avantage de ce régulateur par rapport au précèdent est l'amélioration du **temps de réponse.**

IV Création de l'application final

LabVIEW a la fonctionnalité de construire une application du projet qui est exécutable sur n'importe quel PC, alors nous avons construit l'application de notre projet en suivant les étapes suivantes :

- Dans le menu « Tools » cliquer sur « build Application(exe) from VI… » une boite de dialogue s'ouvrira ;
- Spécifier l'emplacement et le nom du fichier, ensuite cliquer sur « Build ».



FigureVI.13 : Création de l'application final.

IV Conclusion

Notre but de régulation de température à distance est atteint, la supervision par LabVIEW s'avère très pratique et très satisfaisante, la synchronisation entre les deux postes du système est assurée par la transmission sans fil grâce au module Bluetooth. Le cahier de charge étant respecté nous sommes arrivés à terme dans notre projet il faut toujours prendre en considération que les résultats obtenus de la simulation sont un peu différents de celles du monde réel, et cela dépend de la précision des modèles, des composants et de la complication des montages.

Conclusion générale

La commande à distance de température est un secteur très important dans le domaine industriel ou dans la vie quotidienne.

Ce projet de fin d'étude a pour objectif « la régulation de température à distance et la supervision avec LabVIEW ».

Dans le premier chapitre nous avons présenté les divers types de régulation, Nous avons expliqué le principe de fonctionnement et ces équations mathématique.

Dans le deuxième chapitre Nous avons présenté le matériel électronique utilisé, logiciels et applications qui permettent de réaliser ce travail.

Le troisième chapitre on a parlé des deux logiciels principaux qui présentent le plan sur lui on fait notre application finale. Le dernier chapitre, on a expliqué les étapes de réalisation et l'identification du projet.

Ce travail nous a permis d'enrichir nos connaissances dans plusieurs domaines notamment dans le domaine de la régulation numérique, et de commande à distance.

Comme perspectives ce système pourra être développé à l'avenir vers un système de supervision basé sur le web en utilisant l'internet des objets (IOT) qui facilitera l'accès du superviseur sur le système à distance via internet. Aussi avec plus de capteurs l'interface pourra superviser un système contrôlant non seulement la température mais aussi l'humidité, la qualité de l'air et plus de paramètres seront analysés.

À l'avenir, ce système serait mis à niveau vers un système de surveillance basé sur le Web en utilisant le GPS qui faciliterait l'accès de l'utilisateur sur le système à distance via Internet.

Bibliographie

[1] PROUVOST Patrick, « Instrumentation et régulation en 30 Fiches », Edition DUNOD, 2010, 156p.

[2] www.electroschematics.com, (consulté le 30 juin 2020).

[3] JEAN- MARIE FLAUS « La régulation industrielle » HERMES science publications, Paris 2000. (Consulté le 04 avril 2020).

[4] René PRIGENT, Mathieu AUCLERC, « Régulation et automatisme des systèmes frigorifiques 2e Edition », Edition DUNOD, 2013, 207p. (Consulté le 30 juin 2020).

[5] Ricardo Dunia (NI), Eric Dean (NI), and Dr. Thomas Edgar (UT) «Introduction to LabVIEW for Control Design & Simulation »Reference Text: «Process Dynamics and Control 2nd edition», by Seborg, Edgar, Mellichamp, Wiley 2004(Consulté le 15 juillet 2020).

[6] http://www.ni.com/pdf/manuals/372946d.pdf, (Consulté le 2 juin 2020).

[7] Erik Bartman « le grand livre d'Arduino »2éme Edition Eyrolles_2018. Disponible sur : https://www.editions-eyrolles.com/ (Consulté le 12 aout 2020).

[8] www.components101.com (Consulté le 1 mars 2020).

[9] https://www.arduino-france.com/tutoriels/bluetooth-hc-05/ (Consulté le 15 octobre 2020).

[10] « Arduino Based Automatic Plant Watering System ». Devika, S. International Journal of Advanced Research in Computer Science and Software Engineering. 2014. ISSN : 2277 128X.

[11] S. Sumathi and P. Surekha _livre pdf : « LabVIEW Based Advanced Instrumentation Systems »

[12] Aghiles Abed, Hassina Kacimoussa, « Conception et réalisation d'un système de régulation à base d'un microcontrôleur », université Mouloud Ammari_Tizi_Ouzou département d'Automatique. Mémoire de Master2,Disponible sur :

https://dl.ummto.dz/handle/ummto/7394?fbclid=IwAR1-U-RqJo_Jgh2Y0DWvFYaiNU8gtn5uFHenjrrSiMArGO_Gk4EUz9fIKV0

[13] www.ISIS_proteus.odt

Résumé

L'évolution de la technologie crée continuellement des défis que l'homme est toujours prêt à les prendre. Un de ces défis est la commande à distance qui simplifie des contrôles qui ont été difficile voire impossible. La commande à distance de la température est un secteur très important dans le domaine industriel ou dans la vie quotidienne. Ce projet consiste à réaliser un régulateur de température d'un four électrique à distance. Ça implique un développement d'un système de contrôle numérique basé sur une carte Arduino et supervisé par une interface LabVIEW depuis un PC, qui communique avec la carte Arduino via le module Bluetooth, pour commander le système de chauffage (four électrique).

Mots clés : régulation industrielle, contrôle de température à distance, connexion Bluetooth, supervision LabVIEW.

ملخص

يؤدي التطور المستمر للتكنولوجيا إلى خلق تحديات دائمًا ما يكون الإنسان على استعداد لمواجهتها .أحد هذه التحديات هو التحكم عن بعد بالأنظمة التي من الصعب إن لم يكن من المستحيل ضبطها. يعد التحكم في درجة الحرارة عن بعد قطاعًا مهمًا للغاية في المجال الصناعي أو في الحياة اليومية .يعنى هذا المشروع بدراسة تطبيق مراقبة وتحكم عن بعد بدرجة حرارة فرن كهربائي يشرف عليه واجهة غرافيكية لتطبيقLabVIEW من جهاز كمبيوتر يتصل بلوحة Arduino عبر وحدة Bluetooth للتحكم في نظام تدفئة (فرن كهربائي).

الكلمات الرئيسية: التحكم الصناعي، التحكم في درجة الحرارة عن بعد، اتصال Bluetooth، الإشراف