REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOUHMED BOUDIAF-M'SILA

Faculté des Sciences ET Technologies

Département Electronique

N° Ordre : 2020/ ESEM /

Domaine des Sciences ET Technologies

Filière : Electronique

Spécialité : Electronique des systèmes Embarquées

Mémoire de fin d'étude

En vue de l'obtention du diplôme

MASTER

Présenté par

DEGHFEL A.ERRAZAK
HERIZI A.ELGHANI

Thème

Réalisation d'un régulateur industrielle PID à base Arduino et LABVIEW. Application au Régulation de Température et Niveau

Membres de jury :

Dr ATTALLAH BILAL	Président
Dr Y.BRIK.	Promoteur
Dr KHENNOUF SALEH	Examinateur
Dr DJERIOUI MOUHAMED	Examinateur

Année Universitaire : 2019/2020

Remerciements

*Tout d'abord, nous remercions le **bon Dieu** le tout Puissant de nous a donné la force et le courage de mener à bien ce modeste travail et qui nous a permis de vivre ce bonheur.*

*Nous tenons à remercier notre encadreur Monsieur **Dr. Y BRIK**, pour le suivi de notre travail et pour l'encouragement constant qu'il n'a cessé de prodiguer tout au long de la duré de cette travaille.*

Ensuite nos remerciements s'adressent aux membres du jury qui nous a faits l'honneur de participer à notre soutenance, qui a bien voulus examiner et valoriser notre travaille.

Egalement nos remerciements les plus vifs reviennent aux enseignants du département d'électronique et étudiants de la classe électronique des systèmes embarqués sans exception pour leurs collaborations et la communication durant cette année.

Nous tenons à présenter nos vifs remerciements à nos familles et nos proches pour leur soutien et leurs encouragements.

Enfin un grand merci pour tous ceux qui ont collaboré de près ou de loin à la réalisation de ce modeste travail.

M.DEGHFEL ABD ERRAZAK

M.HERIZI ABD ELGHANI

Dédicace

Je dédie cet humble travail à ma petite famille, à ma femme spécialement pour son accompagnement et soutien Inconditionnel.

*Je dédie également ce travaille à mes enfants : **M**aram, **Y**oucef et **I**yad, et je demande l'excuse pour le temps pris de leurs Temps, et je reste toujours à leurs dispositions au monument voulu.*

A mes parents, qui ont un grand plaisir quand je réussite à ma vie personnel et professionnel.

A mes chères frères et sœurs.

A mes amis et mes collègues de travaille,

A tous ceux qui me sont chers, aux personnes qui m'ont toujours aidé et encouragé, qui étaient toujours à mes côtés et qui m'ont accompagnaient durant mon chemin de vie.

Merci à tous

DEGHFEL ABD ERRAZAK

Dédicace

Je dédié ce travail à ma très chère famille pour toute l'affection qu'ils m'ont donné, leur soutien moral et accompagnement.

A ma femme et à la petite fille

A mes chers parents qui ont participés au parcours et étapes de ma vie.

A tous mes amis et collèges de travaille.

HERIZI ABD ELGHANI

Sommaire

Chapitre I

Commande numérique et notions de régulation PID du processus

Chapitre II

Description le dispositif programmable ARDUINO MEGA 2560

Chapitre III

Commande, Contrôle et régulation dans Labview

Chapitre IV

Etude et Réalisation Pratique du Régulateur et Contrôleur PID

Liste des figures

Liste des tableaux

Résumé

Ce projet fin d'étude à pour but d'implanter un régulateur numérique de type PID, ce régulateur industrielle sera réaliser à base d'une carte d'acquisition Arduino pour la commande et contrôle de la régulation de niveau d'un réservoir d'eau et de la température ambiante. La transmission et échange de données (affichage de la mesure, la commande, transmission de consigne et paramètres PID de l'utilisateur…etc.) fera grâce à une interface HMI du poste de commande doté par un logicielle Labview, qui permet de l'affichage et la supervision de déroulement du processus en temps réel.

Introduction Générale

Dans la majorité des procès industrielle de production, les performances sont jugées selon des critères de qualité du produit, de sécurité, de sureté de fonctionnement et la fiabilisation surtout des procès, l'automatique apport des solutions pratiques pour remplir au mieux l'ensemble de ces critères (satisfaction du cahier des charges). La mise en œuvre de ces solutions consiste à conduire – ou piloter – le procédé.

La conduite des procédés consiste a recouvre des activités telles que la planification, l'ordonnancement, la surveillance, la supervision et commande en temps réel.

Le principe général de la commande automatique et manuel des procédés est connu sous le terme de commande en boucle ouvert où fermée (feedback control).

La régulation regroupe l'ensemble des moyens matériels et techniques mis en œuvre pour maintenir la grandeur réglée à une valeur égale à celle de la valeur désirée qui est la consigne malgré les perturbations indésirables.

La régulation de la température, le niveau d'eau ou autre liquide sont parmi les procédés utilisés au monde industriel, laboratoire et à usages dans la vie journalière, ce qui nous oblige de l'implémentation d'une stratégie technique et efficace pour le bon contrôle et régulation automatique de ces procédés en utilisant les différentes techniques efficaces de traitement de l'information.

Ce mémoire traite une commande numérique, régulation manuel et automatique d'un réservoir d'eau et également un contrôle de la température en utilisant un calculateur ARDUINO et un PC doté par le logicielle de commande, contrôle et supervision le LABVIEW.

Dans ce travaille les objectifs ont été visés :

- Le premier chapitre sera regroupe des notions de bases sur la commande numérique et le régulateur PID.
- Le deuxième est de regrouper suffisamment d'informations sur une grande catégorie de cartes d'interfaçage Arduino : son langage de programmation, sa construction, son principe de fonctionnement.
- Nous abordons dans le troisième chapitre la programmation graphique à base du LABVIEW à plusieurs plateformes de l'Arduino et la communication via le port série où USB.
- Dans le dernier chapitre, nous intéressons essentiellement à la réalisation d'un régulateur et contrôleur PID à base Arduino et une interface de commande et contrôle

par LABVIEW. Ainsi l'élaboration des interfaces graphiques, les graphes résultants et les interprétations des résultats.

CHAPITER I

Généralité sur la Commande Numérique Et Régulation PID des Processus

I.1 Introduction

La régulation est au cœur de toutes nos actions : conduire sa voiture, régler la température de sa douche le matin, réaliser une recette de cuisine… La régulation (ou asservissement) consiste à agir de façon à ce que une mesure soit égale à une consigne. Si l'on cherche à atteindre une consigne, on parlera de poursuite ou asservissement, si l'on cherche à éliminer des perturbations pour qu'une valeur reste constante (ex : garder température intérieure de la voiture constante quelle que soit température extérieure), on parlera de régulation. L'industrie utilise à foison des systèmes d'asservissement ou de régulation pour gérer le débit d'un fluide dans une conduite, la température d'un produit, la hauteur d'un niveau de cuve… Historiquement, les régulateurs n'étaient pas intégrés dans l'unité centrale des Automates programmables Industriels, mais se présentaient sous forme de modules autonomes gérant leur environnement propre (acquisition, calcul, commande…). De plus en plus, les automates intègrent les régulateurs, Soit sous la forme de module autonome émulant un régulateur externe au sein de l'UC (évitant ainsi la redondance de câblage qu'imposait l'utilisation de régulateur externe), soit sous la forme de blocs primitifs intégrables au sein du cadeau même titre qu'un bloc temporisateur.

Les régulateurs permettent ainsi de lier plus simplement les parties séquentielles et continues du procédé. La régulation fait partie intégrante de la qualité de production : c'est donc un point non négligeable de la chaîne de valeurs d'une installation.

I.2 Objectif de la régulation automatique

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

On ne peut pas parler de principe de régulation sans parler des lois de commandes. En faite, les grandeurs physiques commandes varient continuent dans le temps. Pour celles qui ne présentent que 2 états (système binaire ou «tout ou rien», tel les feux de signalisation, les commandes d'ascenseurs, de transfert de pièces par convoyeurs, etc..) en utilise une autre approche différente a la structure de boucle utilisée dans la plupart des systèmes [1].

Les systèmes automatiques assurent en fait deux types de fonctions :

- Maintenir la grandeur commandée, ou grandeur réglée, à une valeur de référence malgré les variations des conditions extérieurs ; on parle de la régulation en sens strict.
- Répondre à des changements en temps réel d'objectif, ou à un objectif variable tel que la poursuite de cible, on parle d'un fonctionnement d'asservissement.

I.3 Principe générale de la régulation

Dans la plupart des appareils et installations industrielles, et mêmes domestiques, il est nécessaire de maintenir des grandeurs physiques à des valeurs déterminées, en dépit des variations externes ou internes influant sur ces grandeurs. Le niveau d'un réservoir d'eau, la température d'une étuve, le débit d'une conduite de gaz, étant par nature variables, doivent donc être règles par des actions convenables sur le processus considère. Si les perturbations influant sur la grandeur à contrôler sont lentes ou négligeables, un simple réglage (dit en boucle ouverte) permet d'obtenir et de maintenir la valeur demandée (par exemple : action sur un robinet d'eau). Dans la majorité des cas, cependant, ce type de réglage n'est pas suffisant, parce que trop grossier ou instable. Il faut alors comparer, en permanence, la valeur mesurée de la grandeur réglée a celle que l'on souhaite obtenir et agir en conséquence sur la grandeur d'action, dite grandeur réglant. On a, dans ce cas, constitue une boucle de régulation et plus généralement une boucle d'asservissement.

Cette boucle nécessite la mise en ouvre d'un ensemble de moyens de mesure, de traitement de signal ou de calcul, d'amplification et de commande d'actionneur, constituant une chaine d'éléments associes : la chaine de régulation (ou d'asservissement).

Toute chaine de régulation (ou d'asservissement) comprend trois maillons indispensables : l'organe de mesure, l'organe de régulation et l'organe de contrôle.

Il faut donc commencer par mesurer les principales grandeurs servant à contrôler le processus. L'organe de régulation récupère ces mesures et les compare aux valeurs souhaitées, plus communément appelées valeurs de consigne. En cas de non concordance des valeurs de mesure et des valeurs de consigne, l'organe de régulation envoie un signal de commande a l'organe de contrôle (vanne, moteur, etc.), afin que celui-ci agisse sur le processus. Les paramètres qui régissent le processus sont ainsi stabilises en permanence a des niveaux souhaites [2].

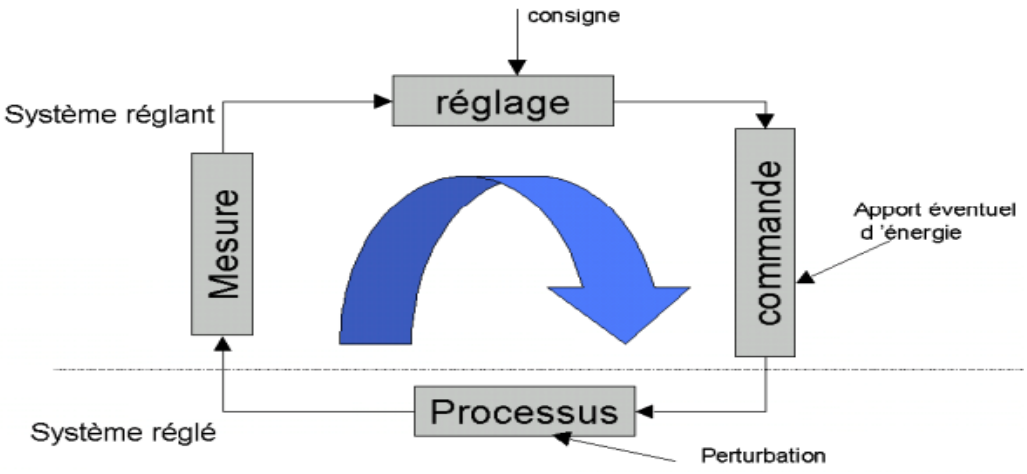


Figure I.1 Schéma de traitement de l'information d'une chaine de régulation

I.4 Structure de commande ET éléments constitutifs d'une chaine de régulation

La commande d'un processus consiste à détermine la commande appropriée, de manière à assurer aux variables à contrôler (sorties) un comportement défini. L'action de la commande est une action susceptible de changer l'état du système à commander. Ces commandes sont délivrées par un organe de commande ; le processus et son organe de commande constituent le système de commande [3].

Il existe quatre grandeurs qui jouent un rôle essentiel dans la régulation :

- **La grandeur réglée** : C'est la grandeur physique que l'on désire contrôler. Elle donne son nom à la régulation. Par exemple : régulation de température, humidité, niveau d'eau …etc.
- **La consigne** : C'est la valeur désirée que doit avoir la grandeur réglée.
- **Les grandeurs perturbatrices :** ce sont les grandeurs physiques susceptibles d'évoluer au cours du processus et d'influencer la grandeur réglée. Par exemple : l'influence de la température extérieure sur le système de chauffage interne.
- **La grandeur réglant :** c'est la grandeur de commande qui a été choisie pour contrôler la grandeur réglée. Par exemple le débit d'eau dans le système de chauffage.

Pour réguler un système physique, il faut :

- Mesurer la grandeur réglée avec un capteur.
- Réfléchir sur l'attitude à suivre : c'est la fonction du régulateur. Le régulateur compare la grandeur réglée avec la consigne et élabore le signal de commande.
- Agir sur la grandeur réglant par l'intermédiaire d'un organe de réglage.

Le schéma ci-dessous montre le déroulement

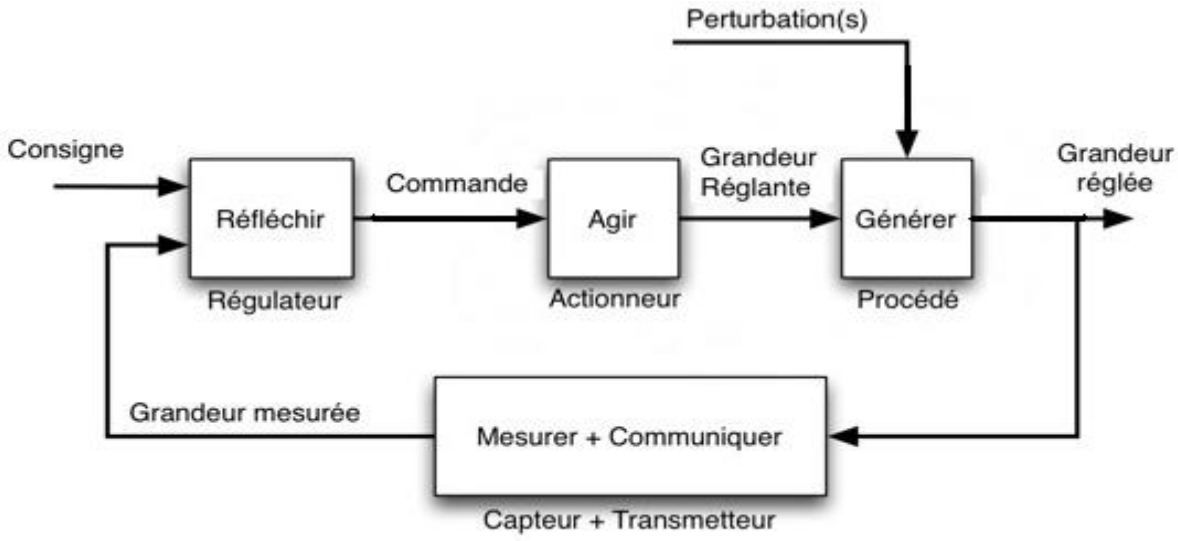


Figure I.2 Structure de commande et principe d'une chaine de régulation

I.5 Comportement des systèmes

Le choix des éléments de la chaine de régulation est dicte par les caractéristiques du processus à contrôler, ce qui nécessite de bien connaitre le processus en question et son comportement.

I.5.1 Comportement en Régulation

La consigne est maintenue constante et il se produit sur le procède une modification (ou une variation) d'une des entrées perturbatrices.

L'aspect régulation est considère comme le plus important dans le milieu industriel, car les valeurs des consignes sont souvent fixes. Néanmoins, pour tester les performances et la qualité d'une boucle de régulation, on s'intéresse à l'aspect asservissement.

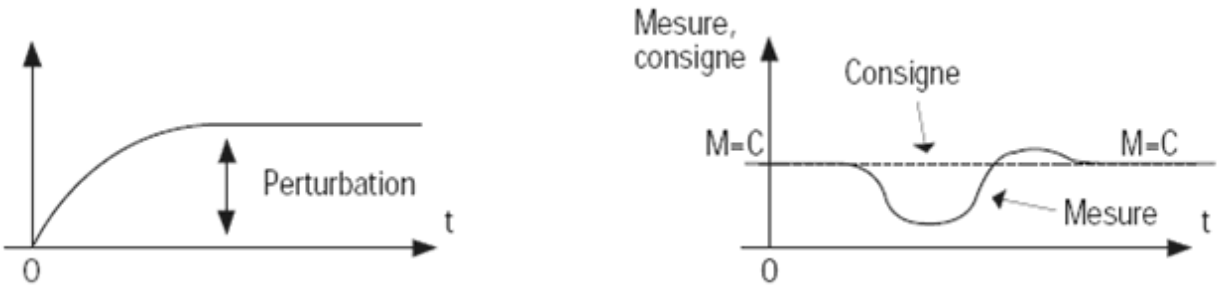


Figure I.3 Comportement des systèmes en régulation

I.5.2 Comportement en asservissement

L'operateur effectue un changement de la valeur de la consigne, ce qui correspond a une modification du point de fonctionnement du processus.

Si le comportement en asservissement est correct, on démontre que la boucle de régulation réagit bien, même lorsqu'une perturbation se produit.

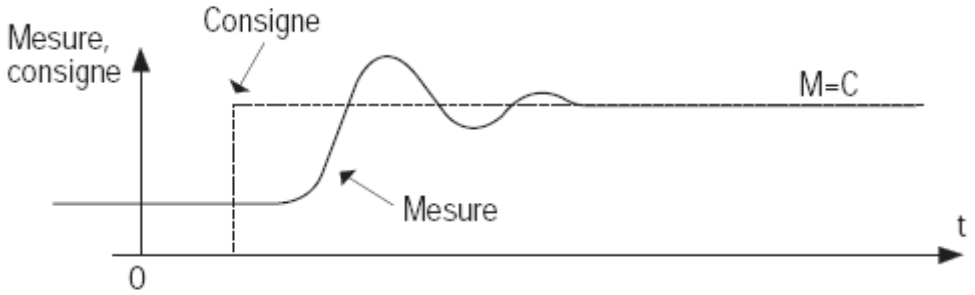


Figure I.4 Comportement des systèmes en asservissement

I.6 Performances des systèmes réglés [4]

Il s'agit maintenant d'analyser la réponse d'un système à un signal, que ce soit lors d'une **expérimentation** ou d'une **simulation**.

Les critères permettant de qualifier et quantifier les performances du système sont :

- ✓ La Stabilité.
- ✓ La précision.
- ✓ La Rapidité.
- ✓ La Amortissement.

L'asservissement idéal est un système ayant une bonne stabilité et bonne précision, le régime transitoire doit être rapide et bien amorti. Ces critères de performances ne sont pas toujours compatibles. Par exemple en mécanique, un processus rapide est léger, il a ainsi une faible inertie et risque d'être peu amorti. D'autre part si on veut améliorer la précision, on raidit l'asservissement et on risque de tomber alors sur un phénomène d'instabilité. Tout l'art de l'automaticien est de réaliser une partie commande permettant de respecter au mieux ces critères.

I.6.1 La Stabilité

On dit qu'un système est stable lorsque celui-ci tend à revenir à son état d'équilibre pour une consigne constante, la sortie doit être constante.

On peut dire aussi qu'un système est stable si sa sortie tend vers zéro lorsque son entrée s'annule c.-à-d. il ne présente pas de divergence temporelle à la sortie.

Les courbes ci-dessous illustrent les deux cas de comportement stable et instable.

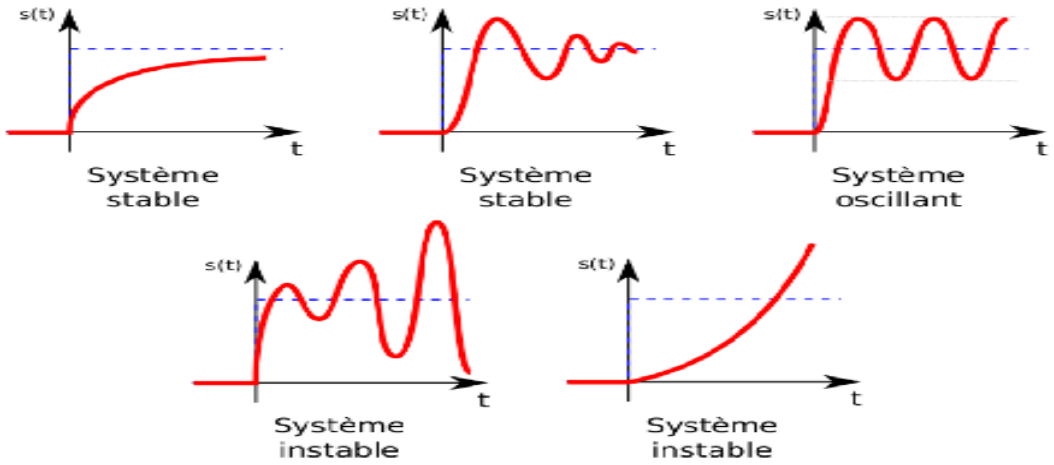


Figure I.5 Stabilités des systèmes asservis (sollicitation échelon)

I.6.2 La précision

La précision quantifié l'erreur lorsque l'équilibre est atteint, Avec l'entrée e(t) et la sortie s(t) de même nature. Autrement, un système est précis si la sortie suit la consigne en toutes circonstances avec un écart inférieur à la valeur définie dans un cahier des charges.

$\varepsilon(t) = e(t) - s(t)$ et on envisage la valeur de ε pour $t \to \infty$ (régime permanent) [5].

Cette erreur dépend de l'entrée (consigne où valeur désirée) et du gain de la fonction de transfert, car l'augmentation du gain permet d'avoir une meilleure précision, On distingue différents types d'erreur, en fonction du signal d'entrée [4]. :

A. **Précision où erreur statique :** l'erreur statique est l'erreur qui subsiste en régime permanent, c'est à-dire lim $\varepsilon(t)$ quand t tend vers l'infini.

La réponse indicielle (échelon d'amplitude E0) permet la mise en évidence de l'erreur statique. Suivant la nature du système cette erreur peut ne pas être nulle, et que des corrections (augmentation du gain, du nombre d'intégrations...) peuvent réduire ou annuler cette erreur.

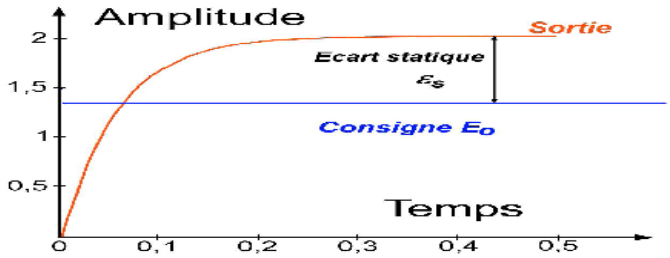


Figure I.6 L'écart (erreur) statique d'une réponse indicielle à échelon

B. **Précision dynamique ou erreur de vitesse :** tant que la partie transitoire de la réponse n'est pas devenue négligeable, $\varepsilon(t)$ caractérise l'erreur dynamique.

La réponse à une rampe permet la mise en évidence de l'erreur en poursuite d'un système suiveur.

On caractérise comme indiqué sur la figure ci-contre, l'erreur de traînage, que l'on observe en régime permanent. L'erreur de traînage ou erreur en vitesse, participe aussi à la précision d'un système, que l'on peut améliorer par des corrections.

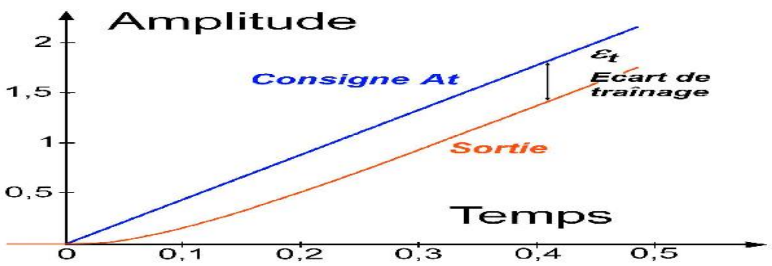


Figure I.7 L'erreur de trainage (vitesse) dans une réponse à une rampe

I.6.3 La rapidité

La rapidité quantifie le temps de réponse du système. Elle correspond au temps de réaction de la sortie par rapport à la consigne.

En pratique et dans les systèmes industriels, on utilise le critère de rapidité de temps de réponse à **5%** (Tr5%) appelé aussi temps d'établissement, c'est le temps mis par le système pour atteindre sa valeur de régime permanent à ± 5% près et y rester.

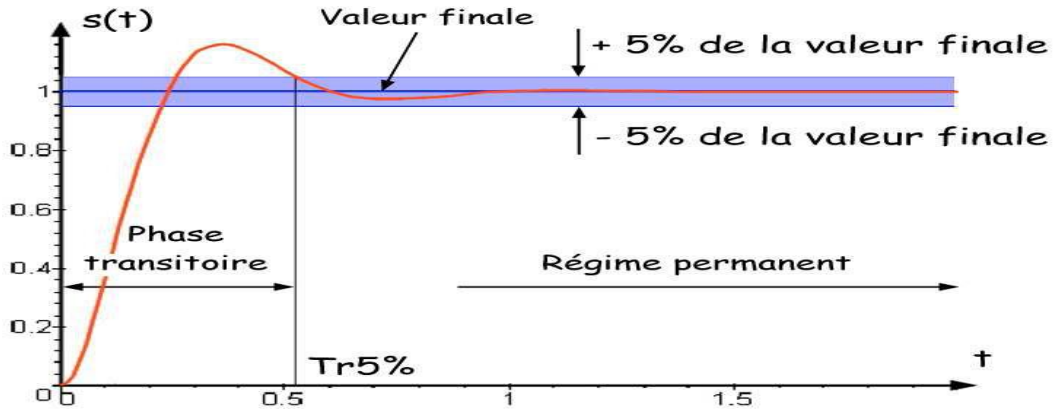


Figure I.8 Tolérance et critère de l'erreur dans une réponse à échelon

I.7 Notion de Boucle ouverte/Fermée (Les déférents types de boucles de régulation) [6]

On distingue en générale deux structures de commande : la commande en boucle ouverte et la commande en boucle fermée appelée également commande à contre-réaction

I.7.1 Système en boucle ouvert

Un système de commande est en boucle ouverte lorsqu'aucune mesure de sortie ***S(t)*** n'est utilisée (ne comporte pas de contre-réaction) pour élaborer la commande ***U(t)*** (Figure I.9). Cela nécessite la connaissance d'un modèle de fonctionnement du système à commander, par exemple la connaissance d'un modèle de fonctionnement d'un moteur à courant continu permettra de connaître la tension d'entrée qu'il faudra lui appliquer pour obtenir la vitesse de rotation désirée, le schéma bloc de système en boucle ouverte est donné par la figure suivante.

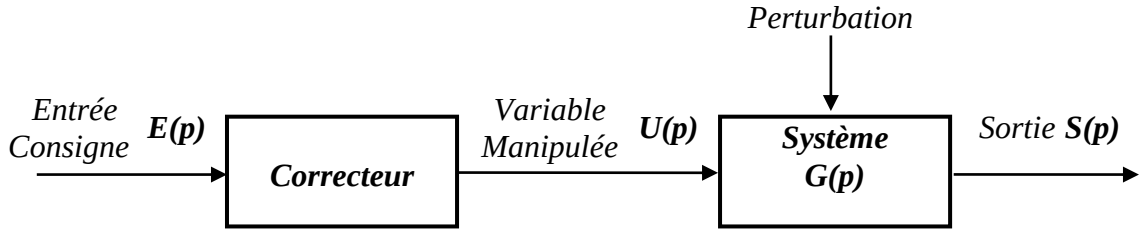


Figure I.9 Schéma bloc de la Commande en boucle ouverte

Cette solution est envisageable dans le cas où le système est parfaitement connu et modélisé et dans le cas où l'obtention d'une mesure de la sortie n'est pas économiquement possible.

I.7.2 Commande en boucle fermé

Afin de résoudre les problèmes de la commande en boucle ouverte et d'automatiser le système (supprimer l'action humaine) on introduit une boucle de retour (ou rétroaction). Dans cette stratégie de commande, une mesure de la sortie est utilisée et comparée avec la consigne par le correcteur (Figure I.10).

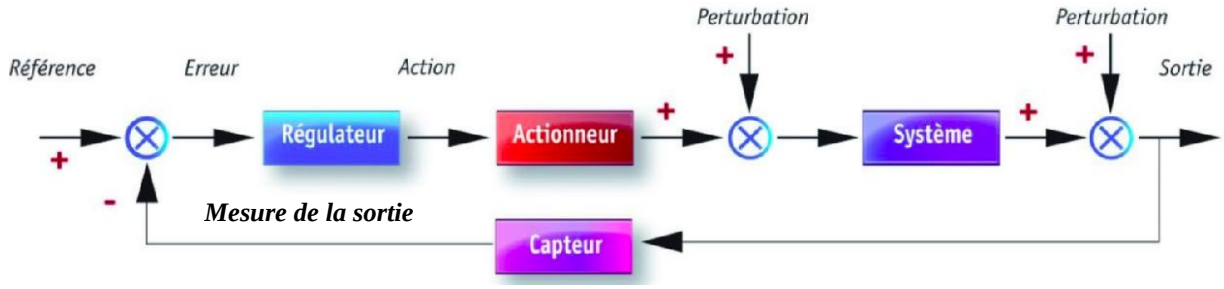


Figure I.10 Schéma bloc de la Commande en boucle fermée

Avec:

- **la consigne ($y_r(t)$ ou $y_c(t)$) :** elle correspond à la valeur souhaitée en sortie,
- **la sortie commandée $y(t)$ (asservie) :** représente le phénomène physique que doit asservir le système de commande (l'asservissement).
- **Écart ou erreur ($\varepsilon(t)$) :** représente la différence entre la consigne et la sortie.
- **Comparateur :** compare en permanence ce que l'on obtient à ce que l'on souhaite obtenir en sortie (élabore l'écart $\varepsilon(t)$).
- **Correcteur :** le correcteur (contrôleur) traite le signal d'écart et détermine le signal de commande.
- **Actionneur :** il reçoit du correcteur le signal de commande et agit directement sur le système à commander. Il représente le "muscle" qui va piloter l'évolution du système (par exemple : amplificateur de puissance, moteur, vérin, vanne, etc..).
- **Capteur :** organe de mesure qui donne une image aussi fidèle que possible de la sortie $y(t)$ et la transforme en un signal compréhensible par le comparateur (le plus souvent électrique). La sensibilité du capteur impose donc les limites de la précision de l'asservissement (par exemple : accéléromètre, thermomètre, gyroscopes, …).

On peut donc définir un **asservissement (système asservi)** comme un système bouclé (boucle fermée) dont le fonctionnement tend à annuler l'écart entre une grandeur commandée (la sortie $y(t)$) et une grandeur de commande (la consigne $y_r(t)$)**.**

I.8 Les régulateurs PID

I.8.1 Définition

Le **régulateur PID**, appelé aussi **correcteur où contrôleur PID** (proportionnel, intégral, dérivé) est un système de contrôle permettant d'améliorer les performances d'un asservissement, c'est-à-dire un système ou procédé en boucle fermée. C'est le régulateur le plus utilisé dans l'industrie et même dans des systèmes complexe où ses qualités de correction s'appliquent à de multiples grandeurs physiques [7].

I.8.2 Principe générale

Un régulateur est constitué d'un comparateur pour observer l'écart entre la mesure et la consigne (référence), et d'un correcteur, dont l'algorithme qui délivre un signale de commande à partir de l'écart et d'obtenir une loi d'évolution de la mesure du procédé conformément au cahier de charge.

Le correcteur PID agit de trois manières :

- ✓ Action **proportionnelle** : l'erreur est multipliée par un gain ***G***.
- ✓ Action **intégrale** : l'erreur est intégrée et divisée par un gain ***Ti***.
- ✓ Action **dérivée** : l'erreur est dérivée et multipliée par un gain ***Td***.

Il existe plusieurs architectures possibles pour combiner les trois effets (série, parallèle ou mixte), on présente ici la plus classique : une structure PID parallèle qui agit sur l'Erreur [7].

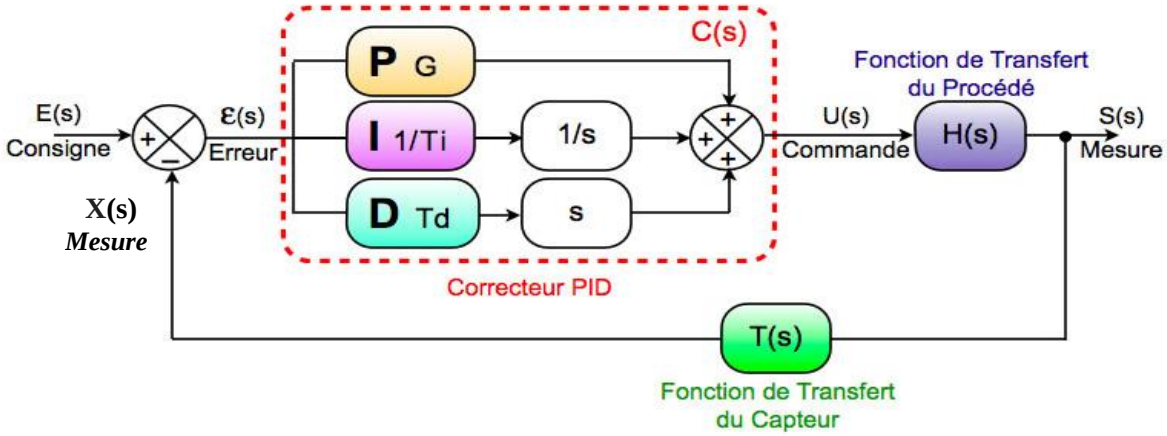


Figure I.11 Schéma bloc d'un système avec correcteur PID (boucle fermé)

Notation :

T(s) signale de mesure, **E(s)** consigne (valeur d'entré où désirée), **ε(s)** signale d'écart (où d'erreur) sont exprimés de la même unité.

Le signale **U(s)** élaboré par le régulateur à partir de l'écart est le signale de commande du procès.

Ecart = mesure – consigne soit **ε = X- E**

La commande sortie d'un régulateur PID, dans le domaine temporel est représenté par l'équation suivante :

$$S(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de}{dt} \quad \text{(I.1)}$$

Avec :

ε(i): Représente l'erreur de suivi, la différence entre la valeur d'entrée désirée (v) et la sortie réelle*(s);*

e(t) : Signal de commande ;

$K_p = G$: Gain Proportionnel ;

$K_i = 1/T_i$ *:* Gain Intégral ;

$K_d = T_d$ *:* Gain Dérivative ;

La fonction de transfert exprimée dans le domaine P du régulateur PID parallèle est la somme des trois actions

$$C(P) = G + (1/T_i) * (1/P) + T_d * P \quad \text{(I.2)}$$

En régulation des procédés, on préfère implanter la fonction de transfert du PID sous la forme mixte

$$C(P) = G\,(1 + (1/P*t_i) + (t_d * P)) \quad \text{(I.3)}$$

Où t_i et t_d sont des constantes de temps (différentes de T_i et T_d dans la formulation précédente) et est le gain de la partie proportionnelle.

Les différents paramètres à trouver sont G, t_i et t_d pour réguler la grandeur physique du procédé ayant pour fonction de transfert $H(s)$. Il existe de nombreuses méthodes pour trouver ces paramètres. Cette recherche de paramètre est communément appelée synthèse [7].

I.8.3 Régulateur physique

Le régulateur se présente sous plusieurs formes : est un appareil indépendant, un module intégré dans l'unité centrale d'un automate programmable (API), ou un bloc fonctionnel de la bibliothèque de cet API. En bus de terrain, une fonction PID est intégrée aussi dans la technologie des capteurs, transmetteurs et des cartes électroniques des actionneurs autonomes.

I.8.4 mode de fonctionnement du régulateur

Deux modes de fonctionnement du régulateur sont disponibles :

I.8.4 .1 Le mode automatique

C'est fonctionnement en contre réaction (boucle fermée), et est le mode normale de fonctionnement de la régulation. La valeur de la sortie Y est calculée par le régulateur et dépend de la valeur de l'écart ε à ce même instant. Le technicien ne peut donc pas modifier la valeur de sortie.

I.8.4 .2 Le mode manuel

C'est le fonctionnement en boucle ouverte et est un mode qui permet au technicien de commander directement le procédé en fixant lui-même la valeur de la sortie. Le correcteur n'agit plus, le régulateur devient alors une station de commande manuelle. D'ailleurs le démarrage des installations se fait très souvent en mode manuel**.**

I.8.5 Les déférents actions du régulateur PID

I.8.5.1 Action proportionnelle P : *(kp de 0.1 à 1000)*

Le régulateur à action proportionnelle, ou régulateur P, a une action simple, puisqu'il construit une commande u(t) proportionnelle à l'erreur e (t). Le rapport erreur/command désigne le gain du système asservi Kp. L'augmentation du gain Kp influe sur le système, il peut accélérer la réponse du système mais, il peut aussi dégrader la stabilité (risque d'instabilité) [8].

- Loi de commande du régulateur P :

$$u(t) = K_p . e(t) \tag{I.4}$$

- Fonction de transfert du régulateur **P** :

$$G_c(p) = \frac{U(p)}{E(p)} = K_p \tag{I.5}$$

- La représentation d'un régulateur **P** par son schéma fonctionnel est

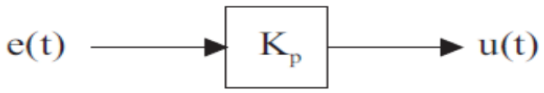


Figure I.12 Schéma fonctionnel du régulateur P

- Réponse indicielle du régulateur **P** :

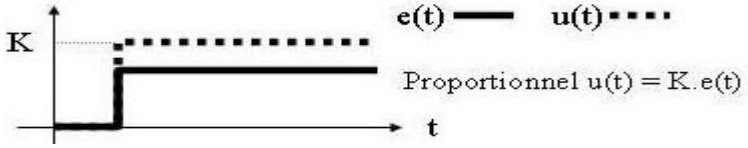


Figure I.13 Action proportionnelle P

On voit que le régulateur P assure une transmission instantanée du signal d'erreur, dans ce sens, son action est relativement dynamique : sa commande ne dépend pas du passé, ni d'une tendance, mais simplement de ce qui se passe à l'instant actuel [9].

I.8.5.2 Action intégrale I : *(T_i de 0.02 à 200min)*

Le signale de sortie est proportionnelle à l'intégrale du signale d'entrée (erreur). L'Anomalie d'entrée et sortie du système, même contre-réfactionnée par un régulateur P, pouvait présenter une erreur permanente en régime permanent. L'intervention de cette erreur lorsque les signaux d'entrée sont constants est désignée par erreur statique. Pour remédier au problème du statisme la solution consiste à intégrer l'erreur [9].

L'action intégrale permet d'annuler l'erreur de position (statique), mais elle peut provoquer des oscillations et des dépassements en régime permanent et même diminuer la stabilité du système [8].

Dans l'industrie le régulateur I est utilisé souvent pour des raisons technologie, d'avoir une précision parfait, exemple : régulation de pression ou de la température d'un réacteur nucléaire. De plus ce régulateur est considéré un filtre donc est utiliser pour le réglage des paramètres dynamiques tels que la pression.

- Loi de commande du régulateur **I** :

$$u(t) = \frac{1}{T_i}\int_0^t e(t)dt \tag{I.6}$$

- Fonction de transfert du régulateur **I** :

$$G_c(p) = \frac{U(p)}{E(p)} = \frac{1}{T_i\, p} \tag{I.7}$$

- La représentation d'un régulateur **I** par son schéma fonctionnel est

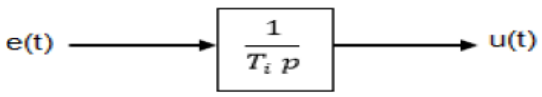


Figure I.14 Schéma fonctionnel du régulateur I

- Réponse indicielle du régulateur **I** :

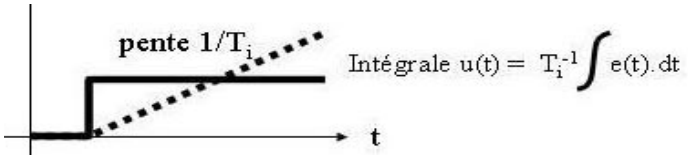


Figure I.15 Action intégrale I

I.8.5.3 Action intégrale D : *(T_d de 0 à 2000s)*

L'action du régulateur D n'intervient que sur la dérivée de l'erreur, c'est à dire qu'elle est sensible à la variation de l'erreur et non à l'erreur elle-même. Lorsque celle-ci est constante (régime statique) le dérivateur n'a aucun effet [9].

L'Avantage de ce type de correcteur est d'améliorer la stabilité du système [8].

- Loi de commande du régulateur **D** :

$$u(t) = T_d \frac{\mathrm{d}\, e(t)}{\mathrm{dt}} \tag{I.8}$$

- Fonction de transfert du régulateur **D**:

$$G_c(p) = \frac{U(p)}{E(p)} = T_d\, p \tag{I.9}$$

- La représentation d'un régulateur **D** par son schéma fonctionnel est

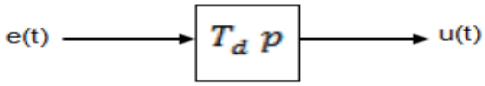


Figure I.16 Schéma fonctionnel du régulateur D

- Réponse indicielle du régulateur **D** :

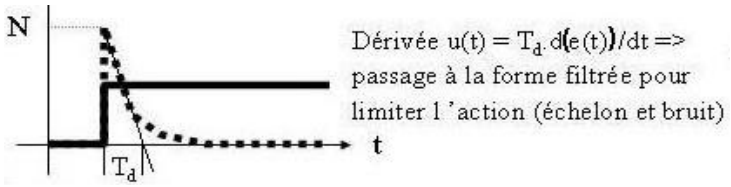


Figure I.17 Action intégrale I

I.9 Les déférents régulateurs (Correcteurs) PID

I.9.1 Régulateur Proportionnelle P

Le rôle de l'action proportionnelle est de minimiser l'écart ε entre la consigne et la mesure, et elle réduit le temps de monter et le temps de réponse.

On constate qu'une augmentation du gain Kp du régulateur entraine une diminution de l'erreur statique et permet d'accélérer le comportement global de la boucle fermée.

$$P(t) = K_p \varepsilon(t) \tag{I.10}$$

On serait tente de prendre des valeurs du gain élevées pour accélérer la réponse du procédé mais on est limite par la stabilité de la boucle fermée.

En effet, une valeur trop élevée du gain, augmente l'instabilité du système et donne lieu à des oscillations. En cas d'un régulateur P [10].

La figure ci-dessous montre un exemple d'un asservissement en position de la réponse d'une entré échelon prennent déférents valeurs du paramètre K_p [11].

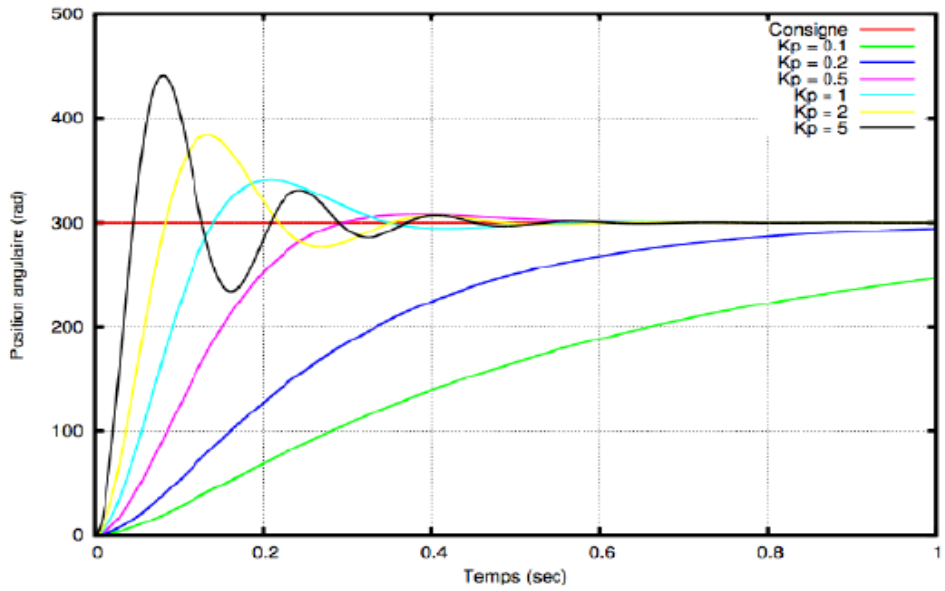


Figure I.18 Modélisation de la réponse à un échelon dans un asservissement en position (Correcteur P)

Effets du correcteur :

- ✓ Diminution de temps de montée.
- ✓ Diminution de l'erreur statique.
- ✓ Augmentation de temps de stabilisation
- ✓ Augmentation du dépassement.

I.9.2 Régulateur Proportionnelle-Intégrale PI

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

La fonction de transfert du correcteur est alors donnée par :

$$u(t) = K_p\left(\varepsilon(t) + \frac{1}{T_i}\int_0^t \varepsilon(t)d\tau\right) \quad \rightarrow \quad C(p) = K_p\frac{1+T_iP}{T_iP} \qquad \text{(I.11)}$$

Le terme intégral complète l'action proportionnelle puisqu'il permet de compenser l'erreur statique et d'augmenter la précision en régime permanant. L'intégrale agissant comme un filtre sur le signale intégré, elle permet de diminuer l'impact des perturbations (bruit, parasites), et il en résulte alors un système plus stable.

Malheureusement, un terme intégral trop important peut lui aussi entrainer un dépassement de la consigne et une stabilisation plus lente.

La figure suivante mentionne l'effet d'un correcteur PI en fixant le K_p et en variant le K_i dans un asservissement en vitesse à un échelon [11].

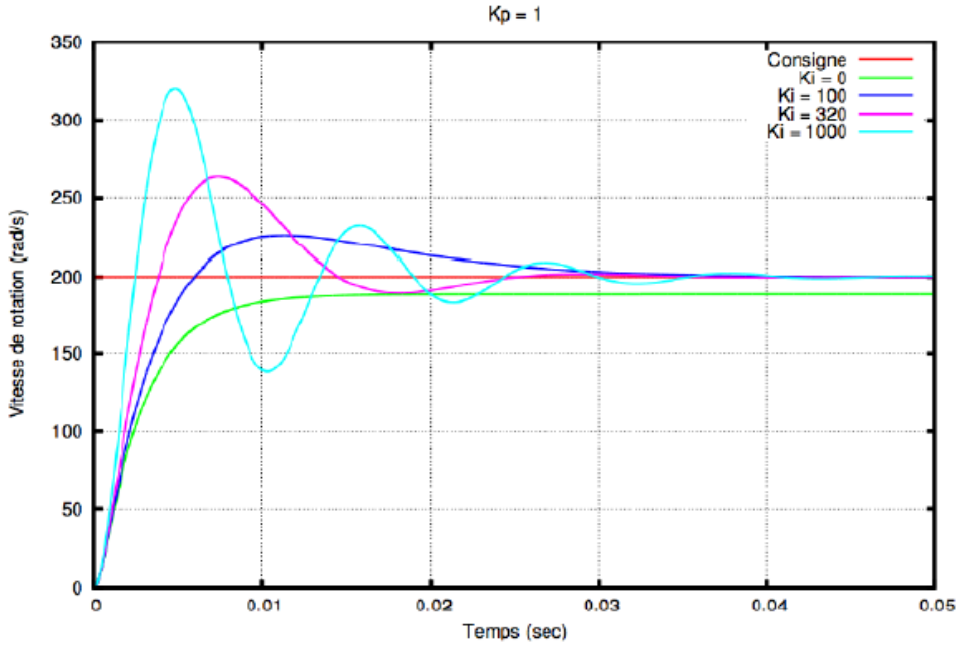


Figure I.19 Modélisation de la réponse à un échelon dans un asservissement en vitesse (Correcteur PI)

Effets du correcteur :

- ✓ Diminution de temps de montée.
- ✓ Elimination de l'erreur statique.
- ✓ Augmentation de temps de stabilisation
- ✓ Augmentation du dépassement.

I.9.3 Régulateur Proportionnelle-Intégrale-Dérivé PID

Le régulateur standard le plus utilisé dans l'industrie, est le régulateur PID (proportionnel intégral dérivé), car il permet de régler à l'aide de ses trois paramètres, les performances (amortissement, temps de réponse, ...) d'un processus modélisé par un deuxième ordre.

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

$$u(t) = K_p[\varepsilon(t) + \frac{1}{T_i}\int_0^t \varepsilon(t)d\tau + T_d\frac{d\varepsilon(t)}{dt} \quad \text{(I.12)}$$

L'action conjuguée PID permet une régulation optimale en associant les avantages de chaque action : la composante P réagit à l'apparition d'un écart de réglage, la composante D s'oppose aux variations de la grandeur réglée et stabilise la boucle de régulation ; et la composante I

Élimine l'erreur statique. Et c'est pour cela que ce type de correcteur est le plus utilise en milieu industriel [11].

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

- ✓ Il fournit un signal de commande en tenant compte de l'évolution du signal de sortie par rapport à la consigne.
- ✓ Il élimine l'erreur statique grâce au terme intégrateur.
- ✓ Il anticipe les variations de la sortie grâce au terme dérivateur.

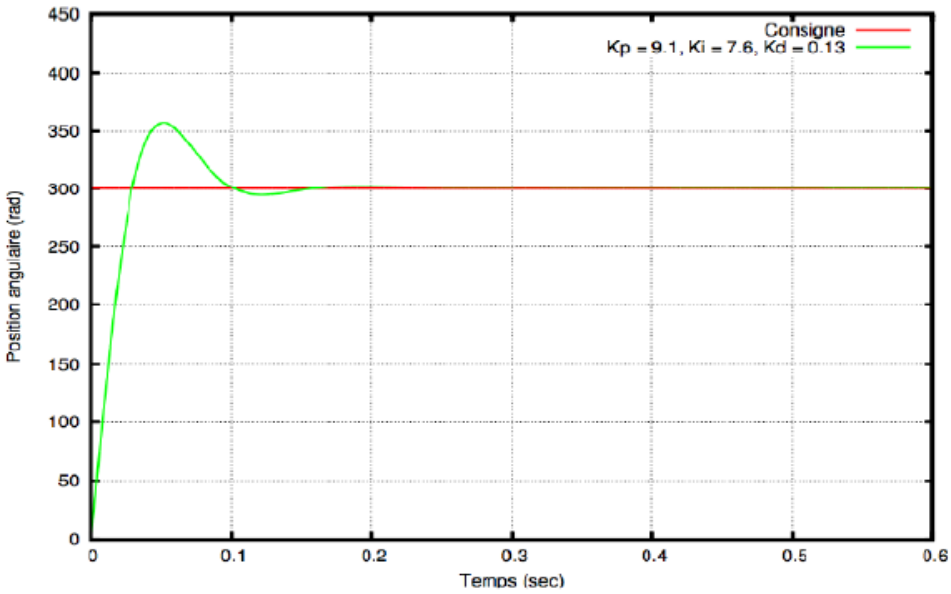


Figure I.20 Modélisation de la réponse à un échelon dans un asservissement en vitesse (Correcteur PID)

Effets du correcteur :

- ✓ Diminution de temps de montée.
- ✓ Elimination de l'erreur statique.
- ✓ Diminution de temps de stabilisation
- ✓ Diminution du dépassement.

Nota

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

I.9.4 Résumés des actions PID et domaine d'utilisation

Ce tableau (Tableau I.1) montre l'influence d'un PID série sur le système qu'il corrige si l'on augmente séparément l'action proportionnelle (P), intégrale(I), ou dérivée(D).

Action	Précision	Stabilité	Rapidité	Rôle et domaine d'utilisation
P				l'action proportionnelle agit de manière instantanée, donc rapide. A fin de diminuer l'écart de réglage et rendre le système plus rapide, on augmente le gain mais, on est limité par la stabilité du système. Le régulateur Pest utilisé lorsqu'on désire régler un paramètre dont la précision n'est pas important, exemple : régler le **niveau** dans un bac de stockage.
I				l'action intégrale complète l'action proportionnelle. Elle permet d'éliminer l'erreur résiduelle (statique) en régime permanent. A fin de rendre le système plus dynamique (diminuer le temps de réponse). L'action intégrale est utilisée lorsque on désire avoir en régime permanentent, une précision parfaite.
D				L'action dérivée, en compensant les inerties dues au temps mort, accélère la réponse du système et améliore la stabilité de la boucle, en permettant notamment un amortissement rapide des oscillations dus à l'apparition d'une perturbation ou à une variation subite de la consigne. L'action D est utilisée dans l'industrie pour le réglage des variables lents telles que la **température**, elle n'est pas recommandée pour le réglage d'une variable bruitée ou trop dynamique (**la pression**,)

Tableau I.1 les effets des actions du correcteur PID et domaine d'utilisation

I.10 Méthode de Réglage de la commande PID [12]

Le réglage d'un PID consiste à déterminer les coefficients Kp, Ki et kd afin d'obtenir une réponse adéquate du procédé et de la régulation. L'objectif est d'être **robuste**, **rapide** et **précis**. Il faut pour cela limité le/ou les éventuels **dépassements** (*over shoots*).

- La robustesse est sans doute le paramètre le plus important et délicat. On dit qu'un système est robuste si la régulation fonctionne toujours même si le modèle change un peu. Un régulateur doit être capable d'assurer sa tâche même avec ces changements afin de s'adapter à des usages non prévus.
- La rapidité du régulateur dépend du temps de montée et du temps d'établissement du régime stationnaire.
- Le critère de précision est basé sur l'erreur statique.

Il existe plusieurs méthodes pour calculer les paramètres du régulateur, pour que le système en boucle fermée ait la réponse désirée. Ces méthodes peuvent être empiriques ou analytiques.

Les méthodes de réglage PID les plus utilisées dans le contrôle industrielles. La méthode universelle de réglage n'existe pas, et il est important de comprendre que c'est la connaissance comportementale du procédé et des interventions autorisées par la production qui induisent la méthode à appliquer pour obtenir les paramètres PID d'un réglage satisfaisant.

I.10.1 Approches possibles

- **Méthode en boucle ouverte**

  Le régulateur PID est en ***mode manuel***. Le technicien agit directement sur la commande de l'actionneur. L'avantage et de pouvoir modéliser le procédé par une fonction de transfert en adéquation avec le comportement observé.

  L'inconvénient est que le procédé évolue librement sans contrôle automatique, et présente donc un certain danger : la production est arrêtée ou perturbée pendant l'essai.

- **Méthode en boucle fermée (Contre réaction)**

  Le régulateur, mis en ***mode automatique***, contrôle le procédé, limitant ainsi les risques dangereux d'emballement. Lorsque la méthode impose une mise en oscillation entretenue, la production est perturbée puisque la grandeur réglée évolue avec l'amplitude inconnue. Un second inconvénient est que la modélisation éventuelle du procédé ne correspond pas en nombre et en valeur aux constantes de temps ou retard réels.

I.10.2 Méthodes expérimentales en boucles ouverte

- **Identification du procédé**

  L'essai en boucle ouverte le plus facile à faire est de créer un incrément, ou échelon de commande et d'enregistrer la réponse.

  La forme de la courbe obtenue conduit au choix d'un modèle le plus représentatif du procédé industriellement, le modèle le plus employé est celui de ***Broida***.

- **Réglage empirique temporel**

  Le réglage préconisé concerne une régulation de poursuite : la réponse prévue à un échelon de consigne est pseudopériodique avec un premier dépassement d'environs 25% à 30%.

  La méthode est empirique et les valeurs de K_P, T_i et T_d du régulateur PID demandent un ajustement éventuel en boucle fermée pour obtenir la réponse finale désirée.

I.10.3 Méthodes expérimentales en boucles fermée

- **Méthode du régleur**

  Applicable à tous les procédés, elle ne nécessite aucune connaissance du procédé.

  C'est une méthode de réglages par approches successives : le technicien règle un à un les paramètres de chaque action PID du régulateur pour obtenir la réponse souhaitée à un échelon de consigne.

- **Méthode de Zeigler et Nichols**

  Le technicien cherche expérimentalement à mettre en oscillation entretenu le procédé en modifiant le gain du régulateur. Deux relevés sur la sinusoïde obtenue permettent de déterminer la valeur des paramètres PID du régulateur.

  La méthode dite du pompage n'est pas applicable si le procédé n'arrive pas à entrer en oscillation entretenu : c'est un moindre mal, puisque cela signifie qu'il n'y a pas de problème de stabilité.

  Les étapes à suivre pour cette méthode de calcule sont :

  1. On trace la réponse indicielle G(s).
  2. On trace la tangente qui passe par le point d'inflexion.
  3. On calcule les paramètres T_a, T_u et K.

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

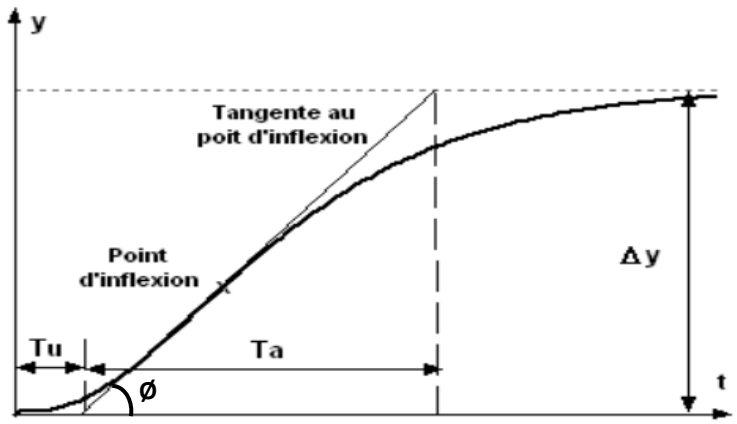


Figure I.20 Courbe de réaction du processus

Tang(Ø) = Δy/ Δu

K étant le gain statique du système K= Tang(Ø) = Δy/ Δu

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

Tableau I.2 Paramètres P, PI, PID par méthode Zeigler et Nichols

I.10.4 Méthodes analytiques

- **Méthode du modèle de référence**

L'approche est ***temporelle*** : on impose la réponse en boucle fermée à un échelon de consigne. L'avantage est de maîtriser la qualité de la régulation de poursuite.

$$F(p) = \frac{X(p)}{W(p)} = \frac{C(p)H(p)}{1+C(p)H(p)} \qquad (I.13)$$

On en déduit la fonction de transfert du régulateur nécessaire et les valeurs des paramètres :

$$C(p) = \frac{F(p)}{H(p)[1-F(p)]} \qquad (I.14)$$

Cette méthode, pouvant s'appliquer aussi à la régulation de maintien, se limite à des fonctions de transfert relativement simple, le régulateur PID n'ayant que trois paramètres.
La fiche 24 décrit cette méthode pour une fonction de transfert du premier ordre.

- **Méthode des marges de stabilité**

  L'approche est fréquentielle : on impose une ***marge de gain*** G_m ou ***de phase*** ϕ_m à la fonction de transfert isochrone en chaine ouverte *C (jw) H (jw).*

  L'avantage est de garantir un degré donné de stabilité et donc une qualité définie du régime transitoire. Par rapport à une approche temporelle empirique, la connaissance du comportement fréquentiel du procédé permet de mieux maitriser le degré d'amortissement par un choix des valeurs T_i et T_d.

  Pour la marge de gain comme pour la marge de phase, on obtient deux équations :

$$Gm = 20 \lg \left[\frac{1}{C(jw)H(jw)}\right] \quad \text{(I.15)}$$

Avec $\phi = -\pi$

$$\phi m = \pi + \operatorname{Arg}[C(j\omega)H(j\omega)] \quad \text{(I.16)}$$

Pour

$$|C(j\omega)H(j\omega)| = 1. \quad \text{(I.17)}$$

La marge de gain est plus utilisée que la marge de phase, car elle garantit mieux les variations éventuelles des constantes de temps et des retards du procédé.

Les inconnues sont K_P, T_i et T_d et w : il faut donc, fixer deux d'entre elles pour déterminer les valeurs de réglages PID.

I.11 Conclusion

Dans ce chapitre, nous avons présenté la composition des déférents systémes de régulation industrielle en boucle ouverte et fermées, une description et mise en évidence qu'une régulation le plus utilisé au monde de l'industrie qui est de type PID est préférentiel pour stabiliser de façon efficace les systèmes automatisés.

Nous avons détaille aussi le fonctionnement de chaque éléments du correcteur PID, précisant le rôle et effet de chacun. Ensuite, on a présenté les déférentes façons d'associer les paramètres P, I et D. En effet, le correcteur PID peut avoir une structure série, mixte ou parallèle.

En fin on a détaille certains nombres d'approches pour calculer les actions d'un régulateur PID.

Nous présentons dans le chapitre suivant l'élément essentielle Arduino MEGA 2560 afin de réaliser une commande PID assisté par un PC.

CHAPITER II

Description le dispositif programmable ARDUINO MEGA 2560

II.1 Introduction

Aujourd'hui, l'électronique est de plus en plus remplacée par de l'électronique programmée. On parle aussi de système embarquée ou d'informatique embarquée. Son but est de simplifier les schémas électroniques et par conséquent réduire l'utilisation de composants électroniques, réduisant ainsi le coût de fabrication d'un produit. Il en résulte des systèmes plus complexes et performants pour un espace réduit.

Depuis que l'électronique existe, sa croissance est fulgurante et continue encore aujourd'hui. L'électronique est devenue accessible à toutes personnes en ayant l'envie : ce que nous allons apprendre dans ce travail est un mélange d'électronique et de programmation.

On va en effet parler d'électronique embarquée qui est un sous-domaine de l'électronique et qui a l'habileté d'unir la puissance de la programmation à la puissance de l'électronique.

II.2 Définition du module Arduino

Le module Arduino est un circuit imprimé en matériel libre (plateforme de contrôle) dont les plans de la carte elle-même sont publiés en licence libre dont certains composants de la carte : comme le microcontrôleur et les composants complémentaires qui ne sont pas en licence libre. Un microcontrôleur programmé peut analyser et produire des signaux électriques de manière à effectuer des tâches très diverses. Arduino est utilisé dans beaucoup d'applications comme l'électrotechnique industrielle et embarquée ; le modélisme, la domotique mais aussi dans des domaines différents comme l'art contemporain et le pilotage d'un robot, commande des moteurs et faire des jeux de lumières, communiquer avec l'ordinateur sans et avec des interfaces HMI, commander des appareils mobiles (modélisme). Chaque module d'Arduino possède un régulateur de tension +5 V et un oscillateur à quartez 16 MHz (ou un résonateur céramique dans certains modèles). Pour programmer cette carte, on utilise l'logiciel IDE Arduino.

II.3 Les gammes de la carte Arduino

Actuellement, il existe plus de 20 versions de module Arduino, nous citons quelques un afin d'éclaircir l'évaluation de ce produit scientifique et académique :

- **Le NG d'Arduino**, avec une interface d'USB pour programmer et usage d'un
- ATmega8.
- **L'extrémité d'Arduino**, avec une interface d'USB pour programmer et usage d'un Microcontrôleur ATmega8.
- **L'Arduino Mini**, une version miniature de l'Arduino en utilisant un microcontrôleurATmega168.

- **L'Arduino Nano**, une petite carte programme à l'aide porte USB cette version utilisant un microcontrôleur ATmega168 (ATmega328 pour une plus nouvelle version).
- **Le LilyPad Arduino**, une conception de minimaliste pour l'application wearable en utilisant un microcontrôleur ATmega168.
- **Le NG d'Arduino plus**, avec une interface d'USB pour programmer et usage d'unATmega168.
- **L'Arduino Bluetooth**, avec une interface de Bluetooth pour programmer en utilisant un microcontrôleur ATmega168.
- **L'Arduino Diecimila**, avec une interface d'USB et utilise un microcontrôleur ATmega168.
- **L'Arduino Duemilanove** ("2009"), en utilisant un microcontrôleur l'ATmega168 (ATmega328 pour une plus nouvelle version) et actionné par l'intermédiaire de la puissance d'USB/DC.
- **L'Arduino Mega**, en utilisant un microcontrôleur ATmega1280 pour I/O additionnel et mémoire.
- **L'Arduino UNO**, utilisations microcontrôleur ATmega328.
- **L'Arduino Mega2560**, utilisations un microcontrôleur ATmega2560, et possède toute la mémoire à 256 KBS. Elle incorpore également le nouvel ATmega8U2 (ATmega16U2 dans le jeu de puces d'USB de révision 3).
- **L'Arduino Leonardo**, avec un morceau ATmega3Ù4 qui élimine le besoin de raccordement d'USB et peut être employé comme clavier.
- **L'Arduino Esplora** : ressemblant à un contrôleur visuel de jeu, avec un manche et des sondes intégrées pour le bruit, la lumière, la température, et l'accélération.

Parmi ces types, nous avons choisi une carte Arduino MEGA 2060. L'intérêt principal de cette carte est de faciliter la mise en œuvre d'une telle commande qui sera détaillée par la suite.

L'Arduino fournit un environnement de développement s'appuyant sur des outils open source comme interface de programmation. L'injection du programme déjà converti par l'environnement sous forme d'un code « HEX » dans la mémoire du microcontrôleur se fait d'une façon très simple par la liaison USB. En outre, des bibliothèques de fonctions "clé en Main" sont également fournies pour l'exploitation d'entrées-sorties. Cette carte est basée sur un microcontrôleur **ATMEGA2560** et des composants complémentaires.

II.4 Pourquoi Arduino Méga

Il existe plusieurs modèles fabriqués par : ATMEL. Le choix dépend de plusieurs critères de sélection dont le développeur doit tenir compte du :

- Type du microcontrôleur.
- Nombre d'entrées/sorties.
- Liaison d'entrées/sorties.
- Conversion analogique numérique et numérique analogique.
- Mémoire RAM, ROM, EPROM interne ou externe, sa taille.
- Vitesse d'horloge, temps d'exécution d'une multiplication, d'une division.
- Bus de données 8bits /16bits.
- Les logiciels de programmation (assembleur, c, micro,…).
- Les émulateurs pour la mise au point des applications.
- Les évolutions prévisibles du composant, son prix, les sources.

II. 5 Constitution de la carte Arduino MEGA 2560

La carte Arduino Mega 2560 est basée sur un ATMega2560 cadencé à 16 MHz. Elle dispose de 54 E/S dont 14 PWM, 16 analogiques et 4 UARTs. Elle est idéale pour des applications exigeant des caractéristiques plus complètes que l'Uno. Des connecteurs situés sur les bords extérieurs du circuit imprimé permettent d'enficher une série de modules complémentaires. Elle peut se programmer avec le logiciel Arduino. Le contrôleur ATMega2560 contient un boot loader qui permet de modifier le programme sans passer par un programmateur. Le logiciel est téléchargeable gratuitement. La carte est illustrée dans la figure si dessous.

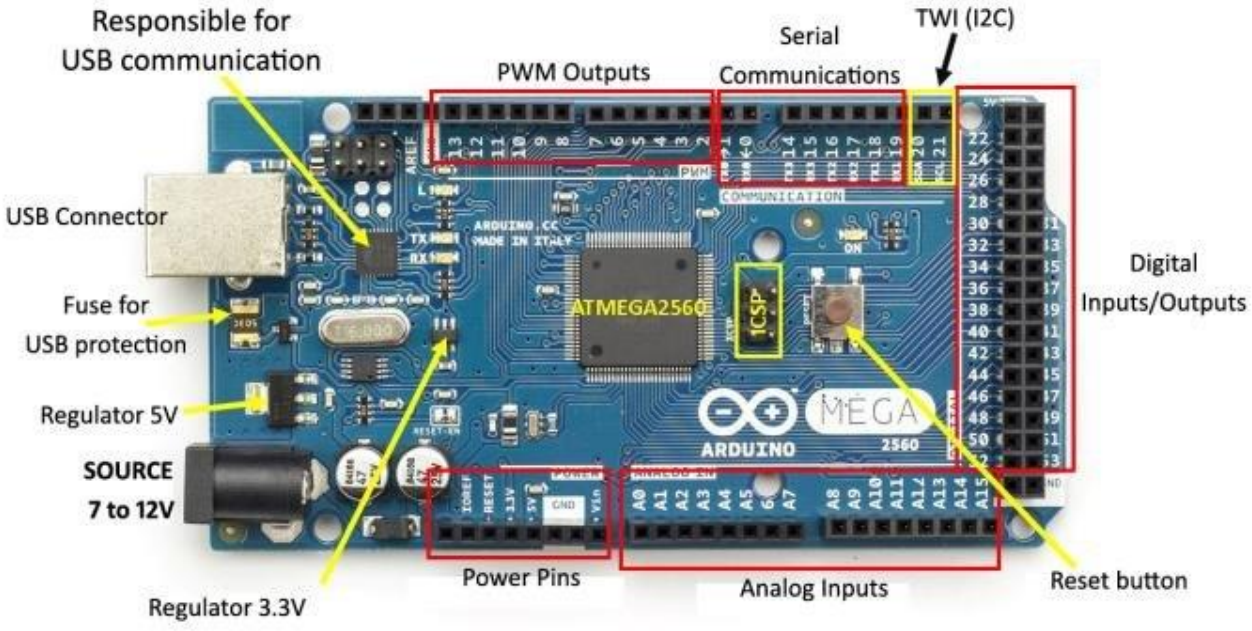


Figure II.1 Présentation de la carte Arduino Mega 2560

II. 6 Les caractéristiques de la carte Arduino Mega 2560 [15]

Le tableau ci-dessous résume les principales caractéristiques de la carte Arduino

Version	Rev. 3.
Dimensions :	107x53 x 15 mm
Tension d'Alimentation	▪ 5V Via port USB où ▪ Tension d'entrée de 7 à 12 V sur connecteur d'alimentation.
Microprocesseur	ATMega2560
Broches E/S Numériques	▪ 54 broches d'E/S dont 14 PWM
Broches d'entrées analogiques	▪ 16 entrées analogiques 10 bits.
Intensité maximale disponible par broche E/S (5V)	40 mA
Intensité maximale disponible la sortie (3.3V)	50 mA
Intensité maximale disponible la sortie (5V)	Fonction de l'alimentation utilisée 500 mA max si port USB utilisé seul.
Mémoire programme flash	256 KB dont les 8 KB utilisé pour boot loader
Mémoire SRAM	8 KB
Mémoire EEPROM	4 KB
Vitesse de l'horloge (Cadencement)	16 MHz
Nombre de port série	3 ports séries
Bus et protocole de communication	Bus I2C et SPI
Gestion des interruptions	
Fiche USB B.	

Tableau II.1 Les caractéristiques de la carte Arduino Méga

II.7 Le Microcontrôleur ATMega2560

L'Atmel ATMEGA2560-16AU est un microcontrôleur 8 bits CMOS basse puissance basée sur architecture RISC améliorée des AVR. En exécutant des instructions puissantes en un seul cycle d'horloge, l'ATMEGA2560-16AU atteint des débits approchant les 1MIPS par MHz permettant aux concepteurs de système d'optimiser la consommation d'énergie par rapport à la vitesse de traitement. [16]

Le microcontrôleur de cette carte est un objet capable de traiter, de stocker et de restituer de l'information. Il est en particulier constitué d'un microprocesseur. C'est la partie centrale qui

Permet le traitement de l'information : il comprend des milliers voir des millions de transistors. Ils réalisent les fonctions logiques suivantes :

- ❖ Calculs arithmétiques et logiques.
- ❖ Mémorisation.
- ❖ Interface de communication.

Les Atmel sont des composants dits RISC (**R**educed **I**nstructions **S**et **C**omputer), ou encore (composants à jeux d'instructions réduit). Ces microcontrôleurs utilisent l'architecture de Harvard : c'est-à-dire le programme et les données sont stockés dans des mémoires physiquement séparées. (Deux bus de données. Un bus est utilisé pour les données et un autre pour les instructions).

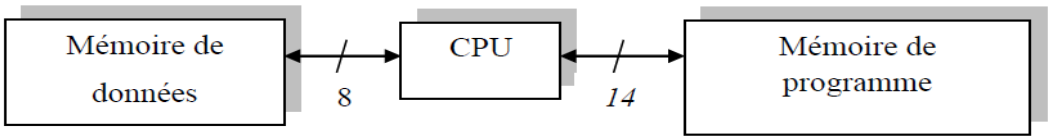


Figure II.2 Architecture de Harvard pour les Atmel

La figure (II.3) montre un microcontrôleur ATmega2560, qu'on trouve sur la carte Arduino

Figure II.3 Microcontrôleur Atmega2560

Architecture RISC avancée ;

- 135 Instructions puissantes, la plupart, avec une exécution en un seul cycle d'horloge.
- Registres 32 × 8 à usage général.
- Fonctionnement entièrement statique.
- Jusqu'à 16 MIPS à 16 MHz.
- Multiplicateur en 2 cycles sur la puce.
- Segments de mémoire non-volatile à haute endurance.
- 256 Ko de Flash dans le système auto programmable.
- EEPROM 4Ko.
- SRAM 8Ko Interne.

- Flash EEPROM 10 000 cycles de Lecture/Ecriture.
- Conservation des données : 20 ans à 85°C / 100 ans à 25°C.
- Section de code de démarrage en option avec bits de verrouillage indépendants.
- Programmation IN SITU par le programme de démarrage sur puce.
- Opération de lecture écriture réelle.
- Verrou de programmation pour la sécurité du logiciel.
- Endurance : Jusqu'à 64 K octets d'espace de mémoire externe en option.
- Supporte la librairie Atmel® QTouch®.
- Boutons tactiles capacitifs, curseurs et molettes.
- Acquisition QTouch et QMatrix®. [16]

II. 8 Description générale de la carte Arduino Mega 2560

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

II. 8.1 Le microcontrôleur [14]

Le microcontrôleur ATMega2560 est constitué par un ensemble d'éléments qui ont chacun une fonction bien déterminée. Il est en effet constitué des mêmes éléments que sur la carte mère d'un ordinateur. Globalement, l'architecture interne de ce circuit programmable se compose essentiellement sur :

- **La mémoire Flash :** C'est celle qui contiendra le programme à exécuter. Cette m moire est effaçable et réinscriptible mémoire programme de 256 KB (dont boot loader de 8 KB).
- **RAM :** c'est la mémoire dite "vive", elle va contenir les variables du programme. Elle est dite "volatile" car elle s'efface si on coupe l'alimentation du microcontrôleur.
- **EEPROM :** C'est le disque dur du microcontrôleur. On y enregistre des infos qui ont besoin de survivre dans le temps, même si la carte doit être arrêtée. Cette mémoire ne s'efface pas lorsque l'on éteint le microcontrôleur ou lorsqu'on le reprogramme.
- **Le registre :** c'est un type de mémoire utilisé par le processeur.
- **La mémoire cache :** c'est une mémoire qui fait la liaison entre les registres et la RAM.

II. 8.2 Les sources d'alimentation de la carte

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

- **VIN :** La tension d'entrée positive lorsque la carte Arduino est utilisée avec une source de tension externe (à distinguer du 5V de la connexion USB ou autre source 5V régulée). On peut alimenter la carte à l'aide de cette broche, ou, si l'alimentation

Est fournie par le jack d'alimentation, accéder à la tension d'alimentation sur cette broche.

- **USB :** Pour fonctionner, la carte a besoin d'une alimentation. Le microcontrôleur fonctionnant sous 5V, la carte peut être alimentée en 5V par ce le port
- **Fiche JACK :** Alimentation externe qui est comprise entre 7V et 12V. Cette tension doit être continue et peut par exemple être fournie par une pile 9V. Un régulateur se charge ensuite de réduire la tension à 5V pour le bon fonctionnement de la carte.

II. 8.3 Les portes entrés sorties et pins de la carte

- **GND :** La carte Arduino Mega2560 se compose de cinq ports de GND.
- **POWER (Puissance) :** la carte Arduino Mega2560 se compose de quatre portes de POWER, et sont comme suit :
- **IOREF : c**ette broche sur la carte fournit la référence de tension avec laquelle le microcontrôleur fonctionne. Un écran correctement configuré peut lire la tension de la broche IOREF et sélectionner la source d'alimentation appropriée ou activer des traducteurs de tension sur les sorties pour travailler avec le **5V** ou **3,3V**.
- **3.3V.** Une alimentation de **3,3 volts** générés par le régulateur à bord, la consommation de courant maximal est de 50 mA.
- **5V.** La tension régulée utilisée pour faire fonctionner le microcontrôleur et les autres composants de la carte (pour info : les circuits électroniques numériques nécessitent une tension d'alimentation parfaitement stable dite "tension régulée" obtenue à l'aide d'un composant appelé un régulateur et qui est intégré à la carte Arduino). Le 5V régulé fourni par cette broche peut donc provenir soit de la tension d'alimentation VIN via le régulateur de la carte, ou bien de la connexion USB (qui fournit du 5V régulé) ou de tout autre source d'alimentation régulée.
- **VIN.** La définition est déjà citée.
- **RESET (Réinitialiser) :** Mettre cette broche au niveau BAS entraîne la réinitialisation (= le redémarrage) du microcontrôleur. Typiquement, cette broche est utilisée pour ajouter un bouton de réinitialisation sur le circuit qui bloque celui présent sur la carte.
- **Broches E/S Digitales et PWM (0-21):**peuvent être configurées en sortie PWM. La configuration est effective dès que le programme utilise la fonction analogWrite
- (...) sur une des broches configurables en PWM. Le rapport cyclique de la PWM est réglé via cette même fonction entre 0, le signal est constamment à l'état bas, 0V, à 255, le signal est constamment à l'état haut, 3,3V ou 5V selon l'Arduino, soit un rapport cyclique de 100%.

La PWM trouve son emploi lorsqu'il s'agit de commander un moteur électrique à courant continu, comme celui qui équipe notre locomotive, ou une intensité lumineuse, par exemple dans une DEL. Si le courant nécessaire à une DEL est compatible avec ce que peut fournir une sortie PWM, il n'est est pas de même pour la commande d'un moteur. Dans ce dernier cas, il est nécessaire d'amplifier le signal via un transistor qui permettra également de passer des 3,3V ou 5V de l'Arduino aux 12V nécessaires pour nos engins.

- **Broches E/S Digitales (22-53):** Chacune des 32 broches numériques (numérotée de 22 à 53) sur le Mega2560 peut être utilisé comme une entrée ou une sortie, en utilisant *Pinmode (...)*pour définir le mode du pin adéquat, *digital Write (...)*, pour écrire dans une sortie soit l'état 0 ou 1 binaire qui est traduit comme tension 5V ou 0v,*digitalRead (...)*permet de lire l'état présente à la broche connectée, toutes ces fonctions. Ils fonctionnent à 5 volts. Chaque broche peut fournir ou recevoir 20 mA en état de fonctionnement recommandée et a une résistance pull-up interne (déconnectée par
- défaut) de 20-50 k ohm. Un maximum de 40mA est la valeur qui ne doit pas être dépassée pour éviter des dommages permanents au microcontrôleur.
- **Broches Entrées analogiques (de A0 à A15) :** La carte MEGA 2560 dispose **16 entrées analogiques** (numérotées de 0 à 15), chacune pouvant fournir une mesure d'une résolution de 10 bits (c.-à-d. sur 1024 niveaux soit de 0 à 1023) à l'aide de la très utile fonction analogRead () du langage Arduino. Par défaut, ces broches mesurent entre le 0V (valeur 0) et le 5V (valeur 1023), mais il est possible de modifier la référence supérieure de la plage de mesure en utilisant la broche AREF et l'instruction *analogRead (...)* du langage Arduino.

ADC. Un signal analogique est celui qui peut prendre un certain nombre de valeurs, contrairement à un signal numérique qui n'a que deux valeurs : HAUT et BAS. Pour mesurer la valeur des signaux analogiques, l'Arduino a intégré analogique-numérique (ADC). L'ADC transforme la tension analogique en une valeur numérique. La fonction qu'on peut utiliser pour obtenir la valeur d'un signal analogique est analogRead (broche). Cette fonction convertit la valeur de la tension sur une broche d'entrée analogique et renvoie une valeur numérique de 0 à 1023, par rapport à la valeur de référence. La référence est 15V sur Arduino Mega2560. Il a un paramètre qui est le nombre de broches Il y a quelques autres broches analogiques de la carte : AREF. Tension de référence pour les entrées analogiques (si différent du 5V). Utilisé avec l'instruction analogReference (). [14]

II.8.4 Les portes de communication

- **Communication série :** Port Série Serial : 0 (RX) et 1 (TX) ; Port Série Serial 1: 19 (RX) et 18 (TX) ; Port Série Serial 2: 17 (RX) et 16 (TX) ; Port Série Serial 3: 15 (RX) et 14 (TX). Utilisées pour recevoir (RX) et transmettre (TX) les données sériées de niveau TTL. Les broches 0 (RX) et 1 (TX) sont connectées aux broches correspondantes du circuit intégré ATmega8U2 programmé en convertisseur USB vers-série de la carte, composant qui assure l'interface entre les niveaux TTL et le
- Port USB de l'ordinateur.
- Ces broches également peuvent être connectées avec des accessoires d'exploitation d'un réseau sans fil et ne sont pas visibles comme : **wifi, GPS, Bluetooth**…
- **Serial Périphérie Interface (SPI) :** est un protocole de données série synchrone utilisées par les microcontrôleurs pour communiquer avec un ou plusieurs périphériques rapidement sur de courtes distances. Il peut également être utilisé pour la communication entre les deux microcontrôleurs. Avec une connexion **SPI**, il y a toujours un dispositif maître (généralement un microcontrôleur) qui contrôle les périphériques. En règle générale, il y a trois lignes communes à tous les appareils :
- **MISO** (port 50) - La ligne esclave pour envoyer des données au maître.
- **MOSI** (port 51) - La ligne principale pour l'envoi de données vers les périphériques.

Et une ligne spécifique pour chaque appareil :

- **SS (**port 53) - la broche sur chaque périphérique que le maître peut utiliser pour activer et désactiver des dispositifs spécifiques.
- **SCK** (port 52) - Les impulsions d'horloge qui synchronise la transmission de données générées par le maître.
- **I2C / TWI broche:** mental (SDA). Est sur la broche numérique 20. (SCL). Est sur la broche numérique 21. Ces broches peuvent être connectées avec des accessoires d'exploitation d'un réseau sans fil FM.

II.9 Les fonctions IDE de l'environnement Arduino

Nous citons quelques fonction de base manipulons les bibliothèques de l'Arduino

Fonctions d'initialisation

- begin () : initialise communication avec Arduino "maître"**.**
- begin (adresse) : initialise communication avec Arduino "esclave".

Fonctions mode maitre

- requestFrom (adresse, quantité) : demande de données à un esclave.
- beginTransmission (adresse) : débute communication avec un esclave (ouvre stockage données à envoyer avec write)

- endTransmission () : envoie des données vers esclave
- write () : écrit les données à envoyer vers esclave
- available () : teste ces données disponibles en provenance esclave (cf-requestFrom).
- read () : lit les données en provenance de l'esclave.

Fonctions mode esclave :

- write () : envoie les données vers le maître après requête.
- available () : test ces données disponibles en provenance du maître (cf onReceive).
- read () : lit données en provenance maître.
- onReceive (fonction) : définit la fonction à appeler sur réception de données en provenance du maître.
- onRequest (fonction) : définit la fonction à appeler sur requête du maître.

Fonctions obsolètes

- send () : obsolète.
- receive () : obsolète.

PWM : 2 à 13 et 44 à 46. Fournissent une impulsion PWM 8-bits à l'aide de l'instruction analogWrite (). [15]

II.10 Fonctionnement et utilisation

L'utilisation des cartes Arduino est très simple : connecter la carte à l'ordinateur (via le câble USB, en général), lancer « Arduino IDE » (c'est le logiciel qui permet de programmer la carte), mettre en place les composants, les relier à la carte, coder le programme, le charger sur la carte grâce au bouton téléviser et le programme s'exécutera ensuite en boucle.

II. 11 Conclusion

Dans ce chapitre nous avons détaille l'élément électronique de base Arduino méga 2560 utilisé dans notre réalisation du module de régulation de type PID, et qui donne vraiment un potentiel de création quasi infini pourvu que l'on dispose d'un matériel approprié. Il est possible de réaliser aussi des dispositifs électroniques, de commande et contrôle des dispositifs électrique et actionneurs divers en tout domaine d'utilisation pratique.

Il est clair que l'Arduino permet d'assurer les principaux protocoles d'interfaçage : communication bidirectionnel, acquisition (conversion) et traitement des différents types de signaux avec une bonne précision et rapidité avec une mémoire de programme de données importantes.

CHAPITER III

Commande, contrôle et régulation dans Labview

III.1 Introduction à la programmation graphique

Labview (Laboratory Virtual Instrument Engineering Workbench) est un langage de programmation dédié au contrôle d'instruments et l'analyse de données. Contrairement à la nature séquentielle des langages textuels, Labview est basé sur un environnement de programmation graphique utilisant la notion de flot de données pour ordonnancer les opérations. [17]

Labview est entièrement équipé pour faciliter la communication avec du matériel tel que le GPIB, VXI, PXI, RS-232, RS-485 et des périphériques d'acquisition de données enfichables. Labview dispose aussi de fonctions intégrées pour connecter votre application à Internet à l'aide du serveur Web de Labview et gère des protocoles standards tels que TCP/IP et ActiveX. . [18]

Grâce à Labview, nous prouvent créer des applications compilées 32 bits qui nous offrir des vitesses d'exécution rapides nécessaires pour des solutions personnalisées d'acquisition de données, de test, de mesure et de contrôle.

Labview contient des bibliothèques détaillées pour l'acquisition, l'analyse et traitement des données à base des fonctions, la présentation et le stockage des données. Labview comprend aussi toutes les structures de programmation des langages traditionnels. Vous pouvez placer des points d'arrêt, animer l'exécution d'un programme et effectuer une exécution pas à pas pour faciliter la mise au point et le développement. [18]

Etant donné que Labview possède la capacité de communiquer avec des périphériques externes, il peut donc naturellement servir à envoyer des données vers ces périphériques afin de les contrôler (un robot par exemple) même si ce n'est pas son objectif premier.

Un programme Labview est appelé instrument virtuel ou VI (pour Virtual Instrument)

III.2 Domaine d'application

Les utilisations de Labview sont de plus en plus diversifiées. A la base, Labview était destiné au contrôle d'instruments de laboratoire.

Labview offre à des milliers d'utilisateurs satisfaits un moyen plus rapide de programmer l'instrumentation, l'acquisition de données et les systèmes de commande. En utilisant Labview pour réaliser le prototype, la conception, les tests et la mise en application de nos systèmes d'instrumentation, nous pouvant réduire aussi le temps de développement du système et en augmenter la productivité de 4 à 10 fois. [18]

Cependant, les possibilités de communications offertes (RS232, USB, Ethernet, GPIB...), la diversité des bibliothèques disponibles, la possibilité de créer très rapidement des

Interfaces graphiques efficaces en font un outil de plus en plus utilisé par les industriels. [19]

Figure III.1 Schéma synoptique de domaine d'application du Labview

III.3 Environnement Labview

Les programmes Labview sont appelés instruments virtuels ou VIs, car leur apparence et leur fonctionnement s'apparentent aux instruments réels, tels que les oscilloscopes et les multimètres. Chaque VI utilise des fonctions qui manipulent les entrées de l'interface utilisateur ou d'autres sources et qui affichent ces informations ou les déplacent vers d'autres fichiers ou ordinateurs.

Un VI contient les trois composants suivants :

- **Face-avant** : Sert d'interface utilisateur.
- **Diagramme** : Contient le code source graphique du VI qui définit sa fonctionnalité.
- **Cadre connecteur et icône** : identifie le VI pour que vous puissiez l'utiliser dans un autre VI. Un VI dans un autre VI est appelé un sous-VI. Un sous-VI correspond à un sous-programme dans des langages de programmation textuels.

Pour écrire un programme sur Labview, on à besoins des Palettes qui nous offrent la possibilité de modifier la face avant et le digramme de Labview, on trouve trois palettes flottantes respectivement nommées: **Outils**, **Commandes**, et **Fonctions**.

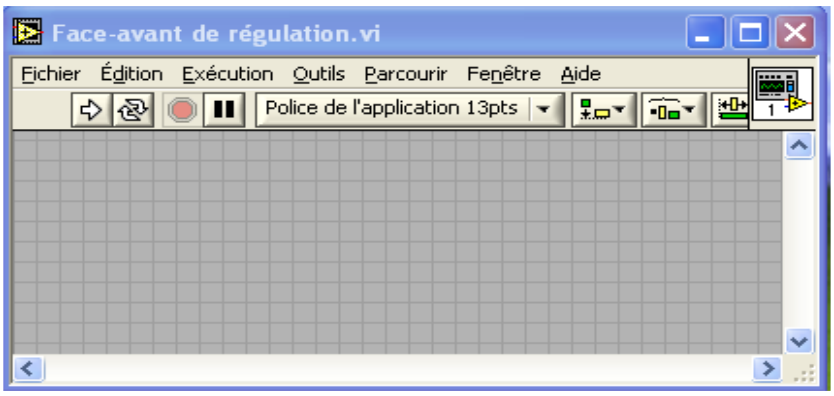


Figure III.2 La face avant (interface utilisateur)

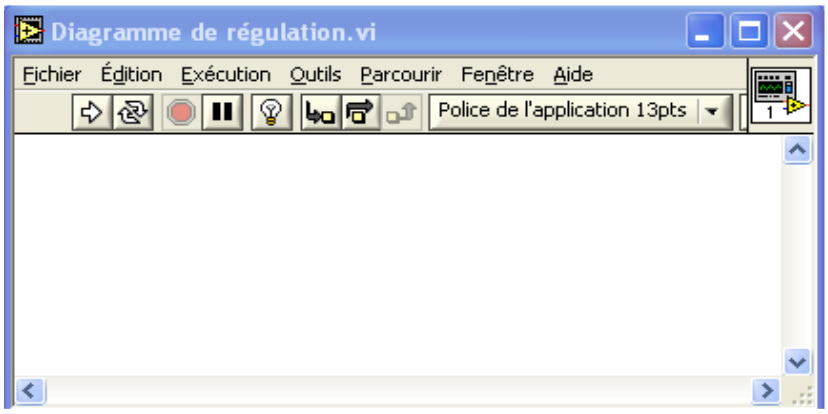


Figure III.3 Diagramme (Code source)

A. Palette d'outils

Elle est disponible sur la face-avant et sur le diagramme, elle contient les outils nécessaires pour faire fonctionner et modifier la face avant et les objets du diagramme.

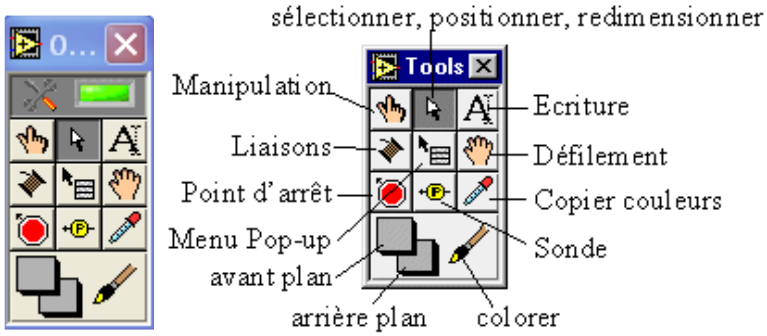


Figure III.4 Palette d'outils

B. Palette de commandes

Elle est disponible uniquement sur la face-avant, elle contient les commandes, les éléments graphiques et indicateurs de la face-avant nécessaire pour créer l'interface utilisateur. Ceux-ci sont hiérarchisés par type de données ou par grandes familles d'objets. [20]

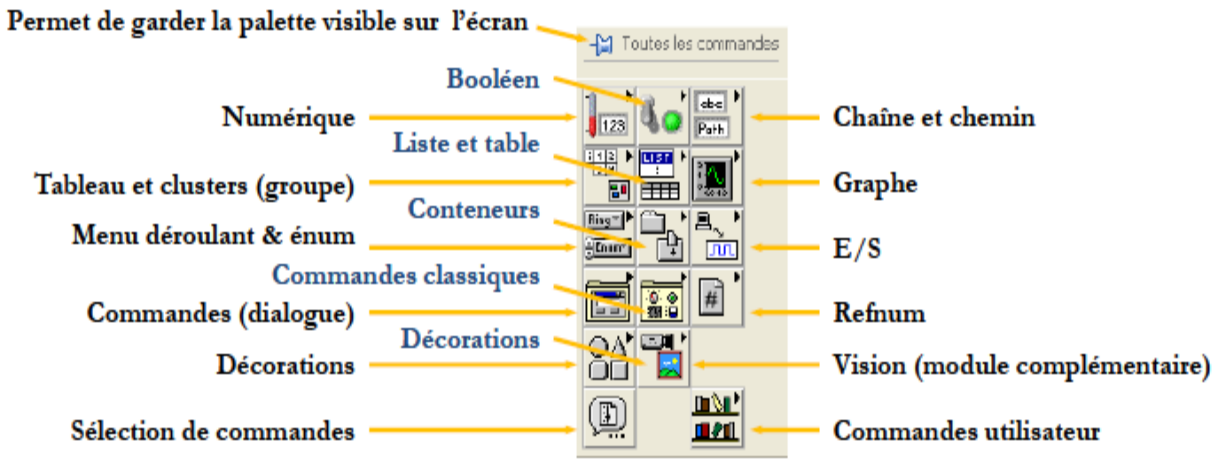


Figure III.5 Palette de commandes

C. Palette de fonctions

Elle est accessible uniquement dans le diagramme par les mêmes méthodes que celle de commandes. Elle contient l'ensemble des fonctions nécessaire de Labview regroupées par type de fonctionnalités (Programmation, acquisition de données, traitement mathématiques et arithmétique, connectivité….). [20]

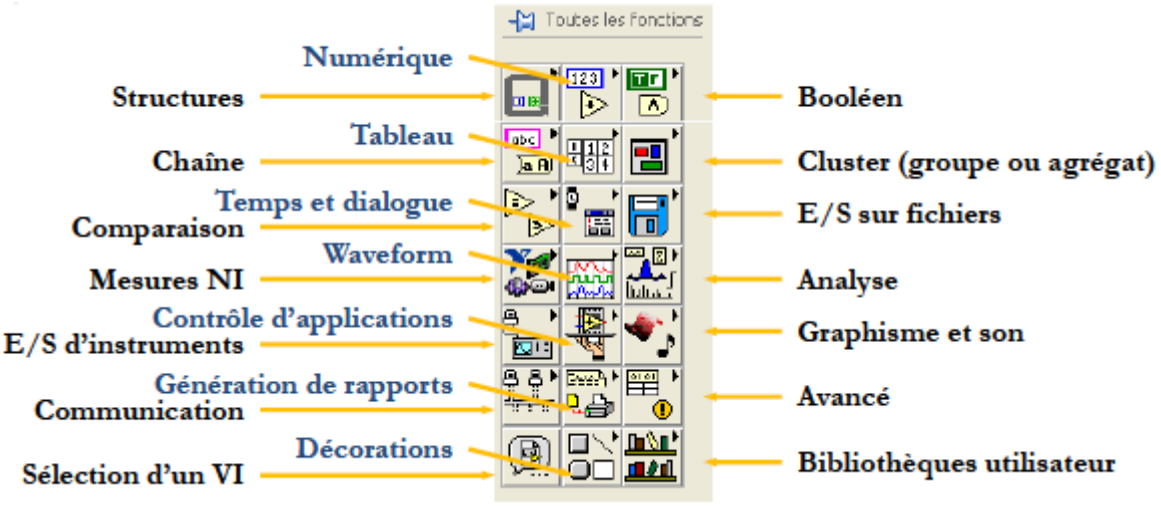


Figure III.6 Palette de fonctions

III.4 Description des sous menus de commande et de fonctions

A traverse les sous menus de commande et de fonctions le programmeur peut manipuler le graphique afin de réaliser n'import quel 'interface de commande et contrôle, donc sont des éléments nécessaire qui vont permettre de crée l'interface utilisateur.

III.4.1 Sous menus de commande

Il s'agite à les commandes et indicateurs formulant l'interface utilisateur tel que, de voltmètre, de boutons poussoirs, indicateurs numériques et analogique, boutons rotatifs etc. Tous ces éléments sont à placer sur la face avant et sont accessibles via la palette de commandes [20].

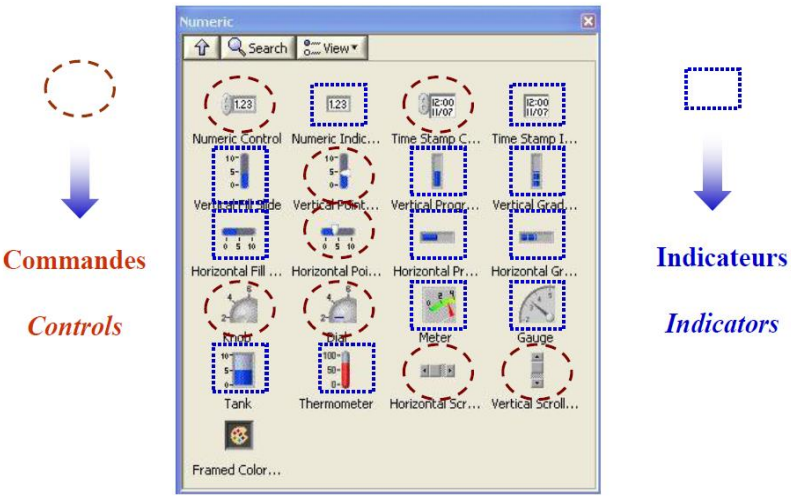


Figure III.7 sous menu de commande

III.4.2 Sous menus de fonctions

Labview dispose les bibliothèques des fonctions logiques et arithmétiques, qui manipulant les bits et facilitent les opérations de calcul au niveau des équations simple et complexe.

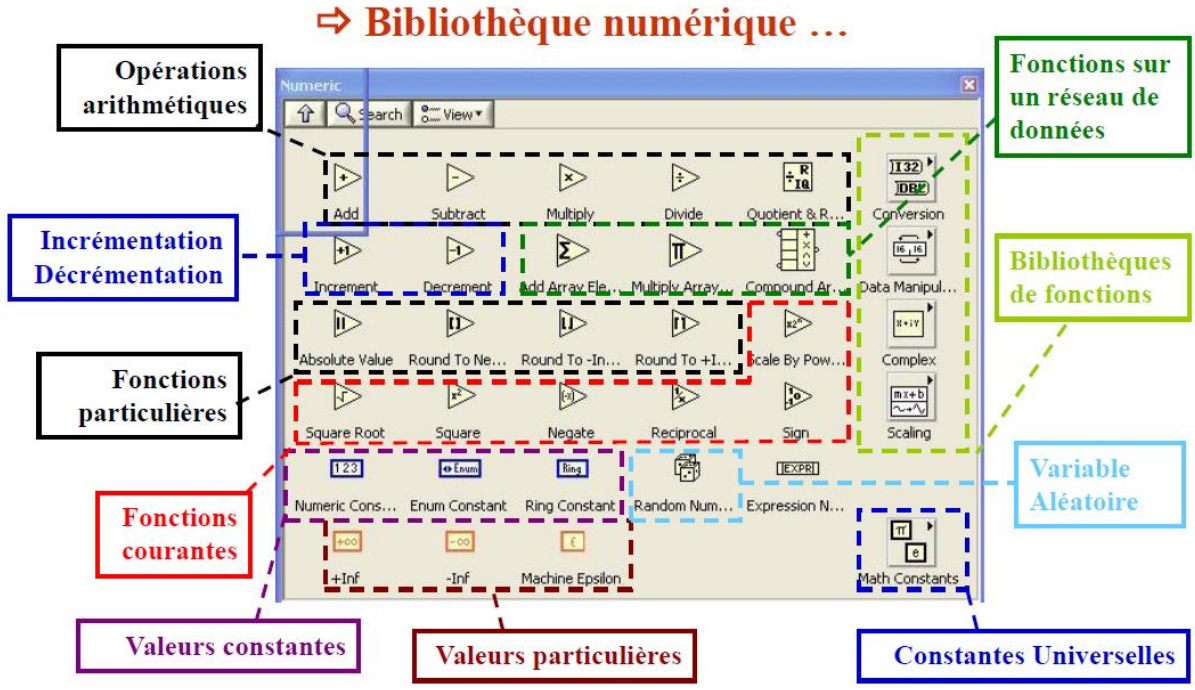


Figure III.8 Sous menu de fonction numérique

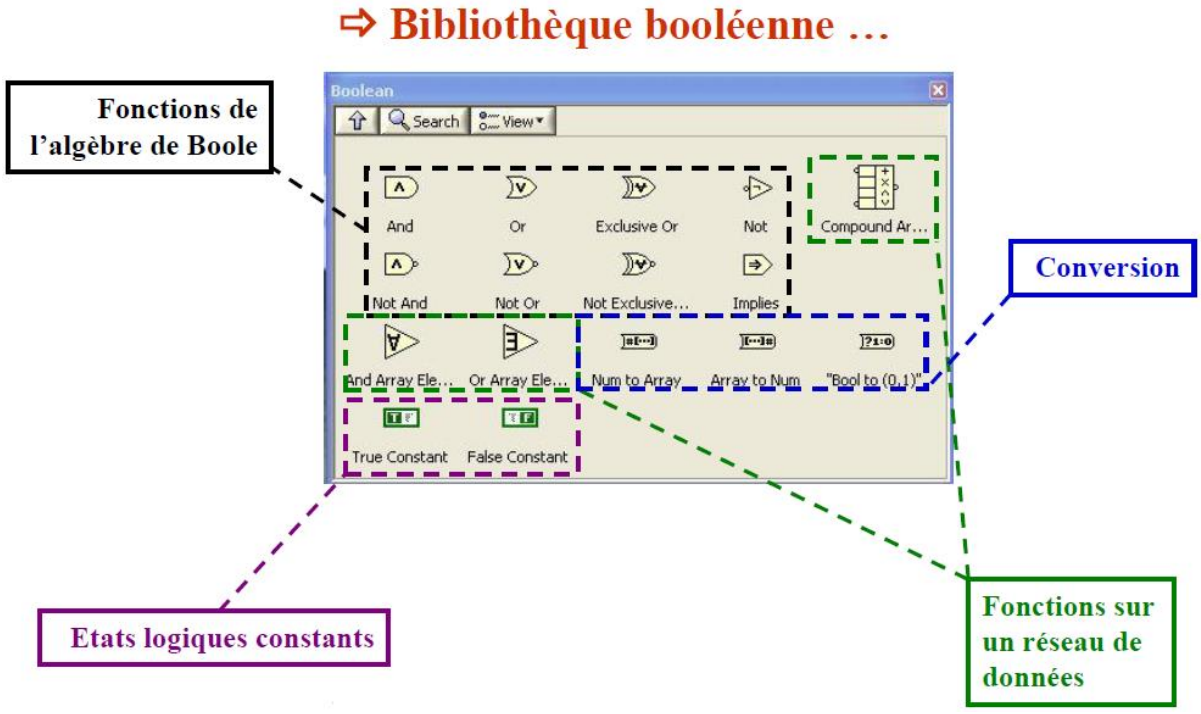


Figure III.9 Sous menu de fonction logique (booléenne)

III.5 Structure dans Labview

III.5.1 Structure de données

Labview utilise un langage fortement typé et toutes données ou structure de données ne peuvent être manipulées qu'avec des fonctions admettant ce type, en faite dans Labview on trouve les types de base scalaire, les types entiers (signes ou non, codes sur 8, 16 ou 32 bits), le type réel (code sur 16, 32 ou 64 bits), le type booléen, le type chaine de caractères, le type tableau (matrice) et cluster. Il est important de noter que les éléments représentant ces données, ainsi que les liaisons issues de ces éléments, sont de forme et de couleur différente.

Ces fils associés aux variables servent à l'acheminement des données entre terminaux vis les nœuds du V.I. [22]

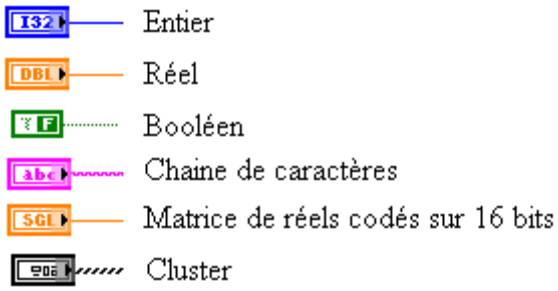


Figure III.10 différents types de structures de données dans Labview

Le langage permet aussi de créer des structures de données plus élaborées.

III.5.1.1 Le type tableau (structure de données homogènes)

Comme tous les langages de programmations la présence de notion de tableau est obligatoire, puisqu'elle définit un outil de base pour le stockage et les traitements des informations.

La bibliothèque tableau dans Labview est riche en fonctions prédéfinit, tel que, l'initialisation, l'indexation, l'inversement d'un tableau, la concaténation de deux tableau, et plusieurs autre fonctions, bien sur, la notion des tableaux peut être élargie pour contenir les matrices. Et comme toutes les autres bibliothèques, on peut l'enrichir avec des fonctions ou des procédures créent à partir des langages de bas niveau comme le langage C. [22]

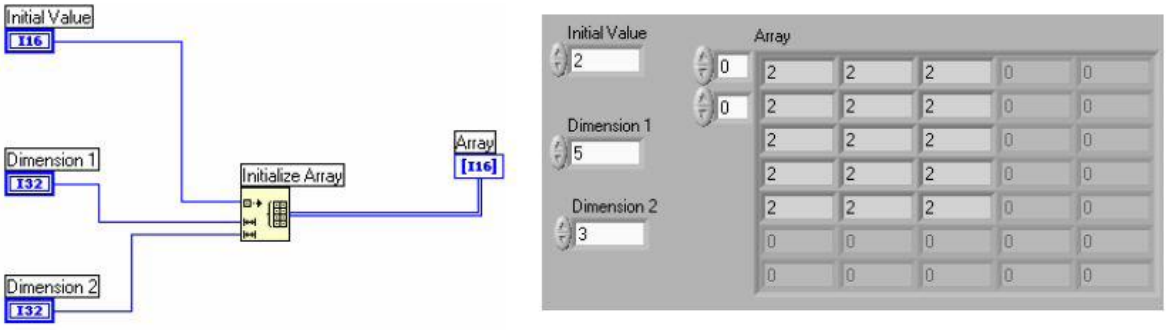


Figure III.11 Exemple d'initialisation d'un tableau a 2 dimensions (matrice)

III.5.1.2 Le type cluster (structure de données hétérogènes)

Les clusters peuvent être comparés aux ensembles en mathématiques (record en Pascal, et struct en C), ce sont des ensembles pouvant englober tout type de donnée, c'est-à-dire des valeurs numériques, booléennes, chaîne de caractères mais aussi des tableaux, des graphes …

Ainsi donc les clusters seront très utiles pour transférer de nombreuses données d'une partie du code à une autre ou d'un VI à un autre.

Des fonctions existent sur les clusters notamment pour les créer et pour désassembler les données du cluster. [22]

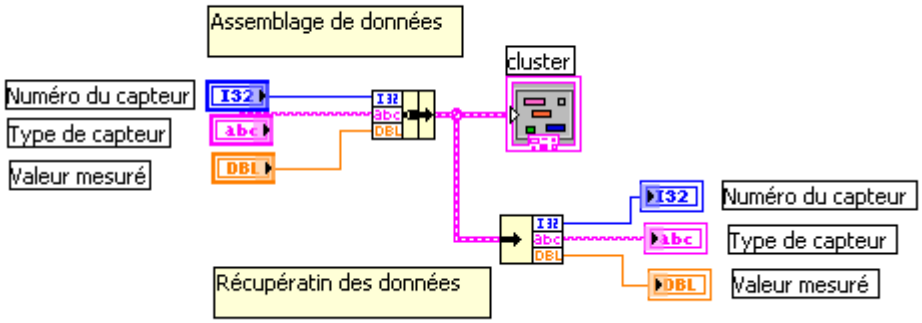


Figure III.12 Exemple de manipulation des données avec le type cluster

III.5.2 Structure de programme

Labview utilise un langage flot de données pur qui a été enrichi de quatre types de structures : la séquence, deux structures d'itération (la boucle « Pour où FOR » avec un nombre d'itérations fixe et la boucle «Tant Que où While » avec un nombre d'itérations soumis a condition) et la structure de choix.

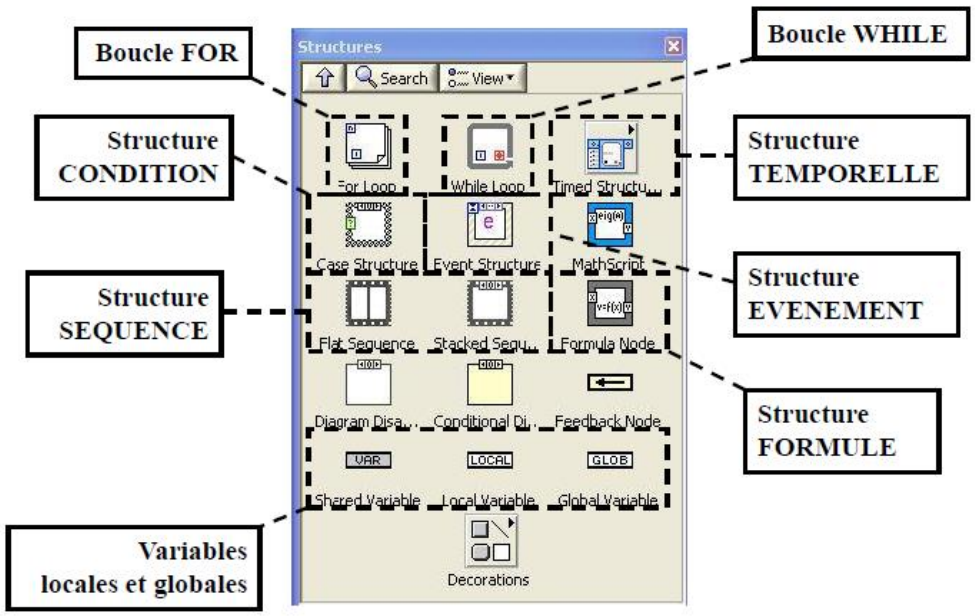


Figure III.13 Bibliothèque boucles et structures données

III.5.2.1 Structure séquence

La structure de « séquence » permet de spécifier l'ordre d'exécution de flots de données et séparer des parties de code qui devront s'exécuter séquentiellement. Cette structure se présente sous la forme d'un cadre et a le statut d'un nœud. [22]

- **Exemple :**

  Sur la figure (III.14), l'utilisation de la séquence s'impose : il s'agit d'initialiser un port série puis d'envoyer une chaine de caractères. Le premier flot de données risque de provoquer des erreurs car rien n'empêche le nœud d'écriture sur le port série de s'exécuter avant le nœud d'initialisation de ce port : l'utilisation de la séquence permet d'ordonner correctement les opérations.

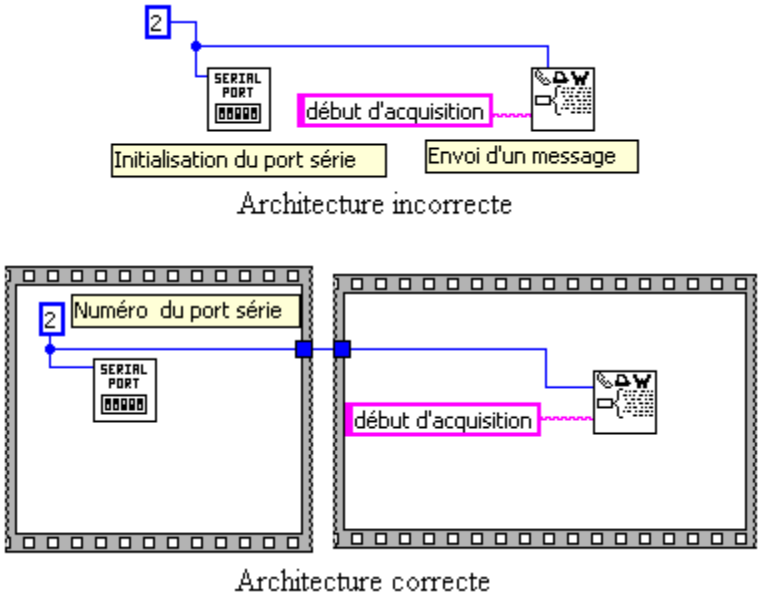


Figure III.14 Exemple d'utilisation de la structure de séquence

III.5.2.2 Les Structures Itératives (Boucles For ET While)

Les deux structures itératives, la boucle «Pour/ For» et la boucle «Tant que/ While», ont aussi le statut d'un nœud ordinaire. Les boucles en générale permettent l'exécution d'un programme, d'un sous-programme ou d'une partie de programme jusqu'à une action ou une valeur définie par l'opérateur.

La boucle « Pour/ For » permet d'exprimer la répétition (ou itération) pour un nombre de fois prédétermine défini par une connexion d'entrée obligatoire: le nombre d'itérations à effectuer N. à l'intérieur de la boucle « Pour » se trouve un terminal d'entrée local générant l'entier indiquant l'indice d'itération de la boucle (i varie de 0 à N–1).

- **Exemple :** L'utilisation de cette structure, présentée sur la figure (III.15), concerne l'acquisition de N mesures avec un temps fixe entre chacune, défini par un nœud de

Temporisation Les N données de sortie sont assemblées pour former un vecteur avec une auto-indexation.

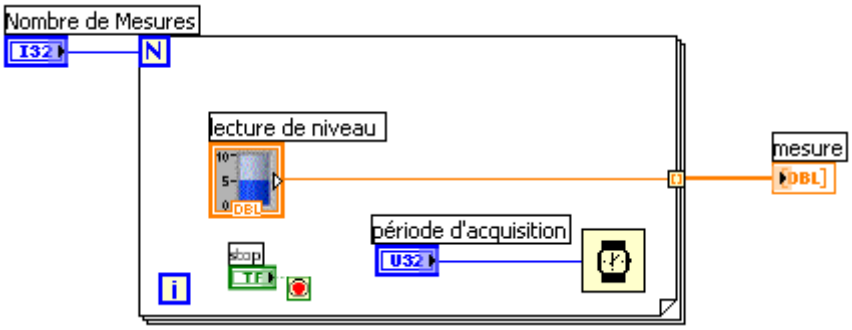


Figure III.15 Exemple d'utilisation de la structure itérative «pour»

La boucle « Tant que / While» permet d'exprimer la répétition pour un nombre de fois non connu à l' avance. A l'intérieur de la boucle « Tant Que » 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 « False » lui est envoyée.

❖ **Exemple :**

L'utilisation précédente décrite sur la figure III.15, est reprise avec cette structure « Tant Que » en arrêtant l'acquisition pour une valeur trop grande de la mesure (figure III.16).

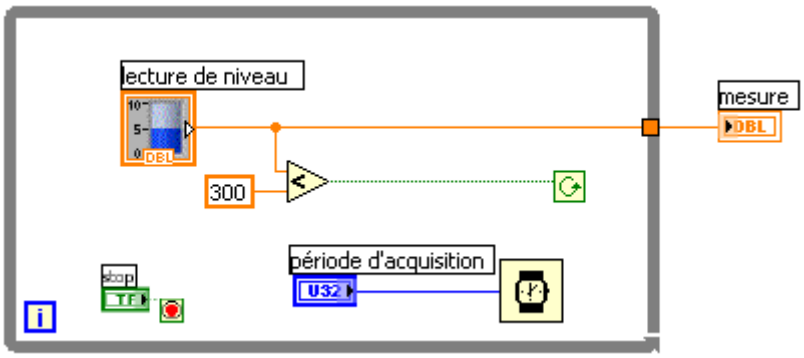


Figure III.16 Exemple d'utilisation de la structure itérative « tant que »

Dans les deux cas de structure itérative lorsque des tableaux de données arrivent ou partent des tunnels, ceux-ci ont la possibilité d'être automatiquement indexes : récupération un par un des éléments du tableau ou concaténation pour créer un tableau. Dans le cas des deux structures, il est possible de mémoriser des résultats produits lors d'itérations antérieures et de les récupérer ensuite. Ces registres à décalage sont des variables locales à une boucle.

❖ **Exemple :**

La figure (III.17) présente une utilisation de ces registres a décalage dans le cas de la boucle « Pour ». Il est important de remarquer que ces registres à décalage constituent

Un des effets de bord du langage. Il est de plus nécessaire d'initialiser ces mémoires locales pour réaliser une utilisation contrôlée et cohérente.

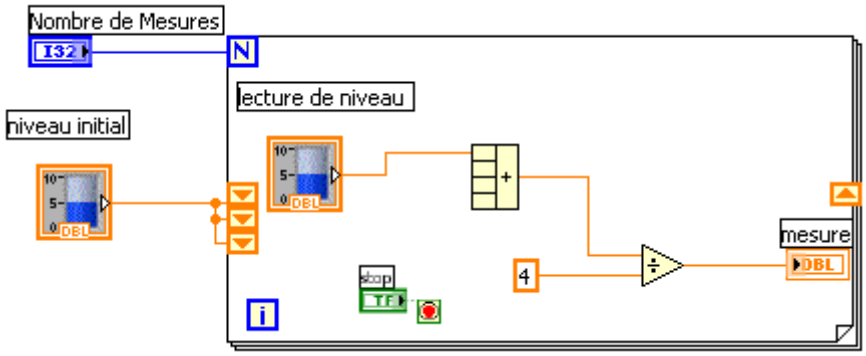


Figure III.17 Exemple d'utilisation des registres à décalage dans une boucle « pour »

N.B : Dans des structures itératives nous somme obligés de créer une commande STOP pour pouvoir arrêter la boucle. Tant que STOP est faux la boucle continue. Les boutons STOP possèdent différentes actions mécaniques.

III.5.2.3 La Structure de choix

La dernière structure nécessaire est celle qui va permettre d'exprimer l'alternative (le choix peut être considéré comme un **IF** ou un **CASE**). La sélection du cas exécute est faite par la valeur de la variable connectée a l'entrée représentée par un point d'interrogation (figure III.18). L'identifiant du cas représente est indique en haut de la structure. [22]

- **Exemple :**

  La figure (III.18) présente une utilisation de la structure de choix. Si nous avons une action « vrais » alors il' y a une acquisition de donne sur le port série, et si l'action est « faux », alors il' y aura pas d'acquisition.

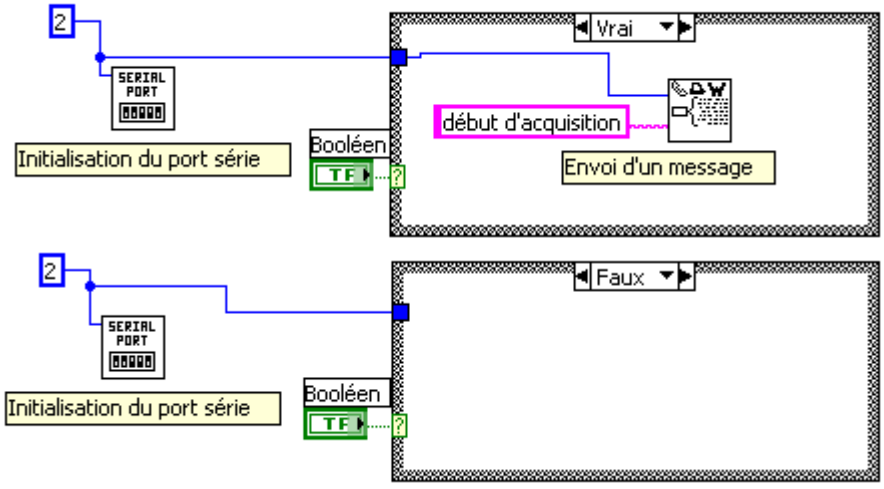


Figure III.18 Exemple d'utilisation de structure de choix.

III.6 Traitement numérique dans Labview

III.6.1 Les fonctions prédéfinies

Labview possède les instructions de base d'un langage de programmation permettant de traiter les différents types de données. Ainsi, nous avons des fonctions liées aux variables numériques (entiers, réels et complexes), aux variables booléennes, aux chaines de caractères, et aux tableaux.

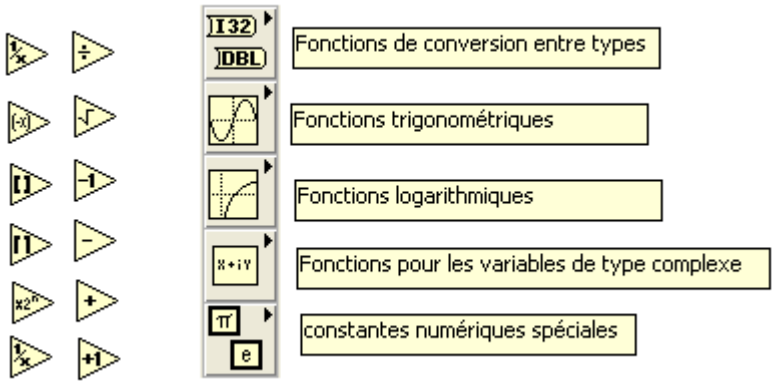


Figure III.19 Quelques instructions de traitement de données numérique dans Labview.

Nous trouvons aussi les operateurs de test et de comparaisons liées à ces différents types de données. A partir des structures de contrôle, des fonctions et des operateurs, de base, il est alors possible de traduire un algorithme quelconque et d'enrichir la bibliothèque des fonctions en utilisant le mécanisme d'encapsulation. Un diagramme complet est alors réduit a un nœud qui peut être ensuite réutilise.

III.6.2 Boites à outils mathématique

Labview contient aussi des boites de calcul mathématiques qui servent a introduire des commandes complexes tel que (linspace, ode45, sin, cos, etc.)

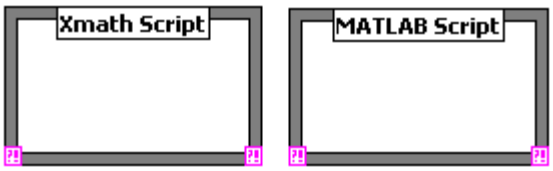


Figure III.20 Exemple de boite de calcul mathématique dans Labview.

III.7 Communication Arduino et Labview (pilot E/S instruments)

Dans cette partie nous allons découverte comment établir la communication entre l'interface Labview et une carte Arduino via un protocole de communication par exemple série (UBS), et en utilisent différentes bibliothèque et plateformes permettant le contrôle et pilotage d'instruments E/S et ainsi l'échange de donnés en temps réel.

Les différents interfaces et bibliothèques utilisées sont :

- ✓ Interface et Bibliothèque **VISA** (NI Serial Communication).
- ✓ Interface Arduino-**LIFA** (Labview Interface for Arduino).
- ✓ Interface ET platform Arduino-**Maker hub** (LINX).

Donc L'objectif est de présenter des techniques d'interfaçage de la carte Arduino avec Labview, pour cela il est nécessaire de connecter une carte Arduino (Exemple MEGA 2560) via USB à l'ordinateur PC doté par un logicielle Labview, afin d'installer un paquet spécial (Package Manager) pour Labview et contrôler l'Arduino directement à partir de Labview.

La procédure ci-dessous explique comment installer, configurer et intégrer Arduino Board à Labview. [19]

Figure III.21 Interface Labview-Arduino.

Donc les éléments nécessaire pour l'installation et afin d'établir la communication sont :

- Installation du Labview.
- Installation de l'Arduino IDE.
- Installation NI-VISA.
- Gestionnaire de paquets VI (Package Manager (VIPM)).
- Après l'utilisateur doit être installé les Interfaces Labview pour Arduino a partir du Package Manager La bibliothèque Arduino-Labview(LINX) ou la bibliothèque (LIFA BASE), et d'autres si nécessaire.

III.7.1 Interface et Bibliothèque VISA

Nous utilisons pour les communications avec les instruments principalement les fonctions VISA (Visual Instrument Software Architecture) utilisées sous Labview pour les drivers d'instruments.

Les fonctions VISA vont nous permettre de communiquer avec les pilotes d'E/S installés sur le système tels que les liaisons série RS232, USB, GPIB, Ethernet, VXI ou PXI dont les drivers sont écrits pour différents types de communication

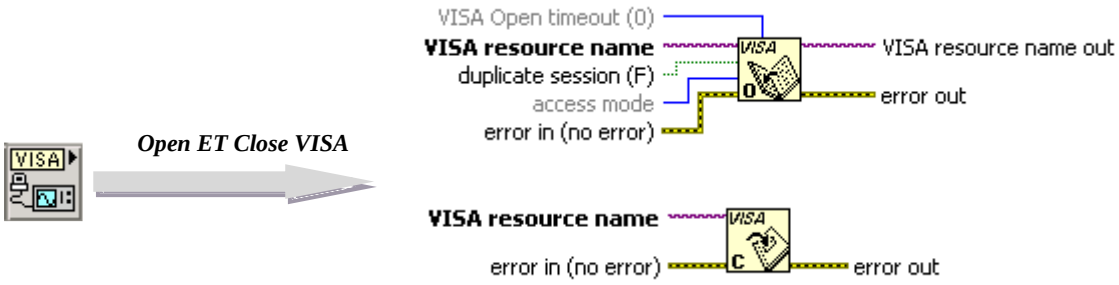


Figure III.22 Ouverture et fermeture d'une VISA pilote d'instrument.

Les fonctions VISA les plus couramment utilisées pour communiquer avec les instruments de mesure sont les fonctions VISA : VISA Write et VISA Read.

- VISA Write écrit le contenu d'un tampon vers le périphérique désigné par le nom de la ressource VISA.
- VISA Read lit les données en provenance du périphérique. Le VI demande le nombre de caractères à lire. Cette valeur doit être supérieure ou égale au nombre d'octets à transmettre. Si la fin de message est implémentée par le matériel (GPIB ou RS232 par exemple) la lecture s'arrête sur le terminateur, sinon elle s'arrête au nombre de caractères ou au time out.

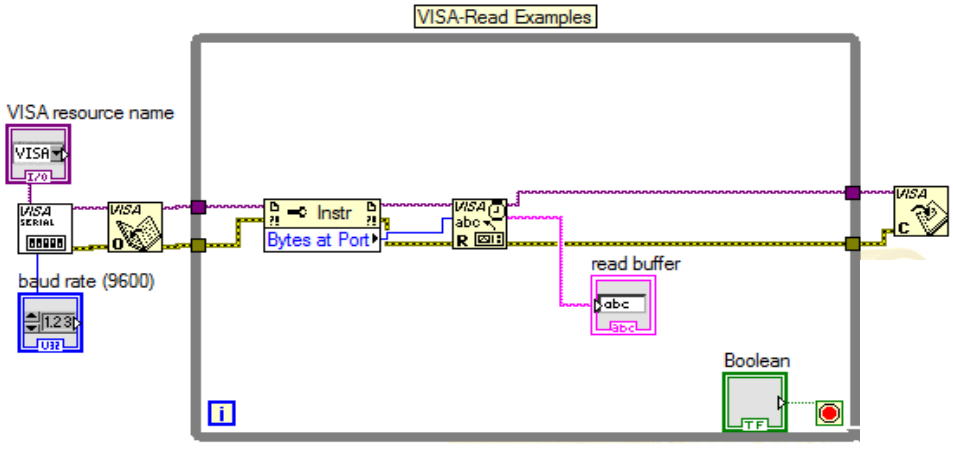


Figure III.23 Exemple VISA Lecture.

III.7.2 Interface Labview Arduino-LIFA

LIFA signifie Labview Interface for Arduino, l'interface Labview pour Arduino (LIFA) c'est une boite à outils permet aux développeurs d'acquérir des données à partir du microcontrôleur Arduino.

Afin de réaliser un programme et établir la communication en utilise l'interface Labview pour Arduino LIFA, il est important de suivre les étapes suivantes pour une fois:

- Installation dans Labview la bibliothèque ou le module Arduino LIFA via le VI Package Manager (VIPM).
- Téléchargement via l'IDE Arduino le **Firmware LIFA base** (fichier LIFA base.ino) sur la carte Arduino cible, ce programme Firmware est disponible par défaut aux libraires de l'Arduino dans le fichier LIFA_base.
- Lorsque le module Arduino est installé, la palette Arduino et ces bibliothèques est disponible dans la palette des fonctions du Labview. La figure ci-dessous présente le module Arduino.

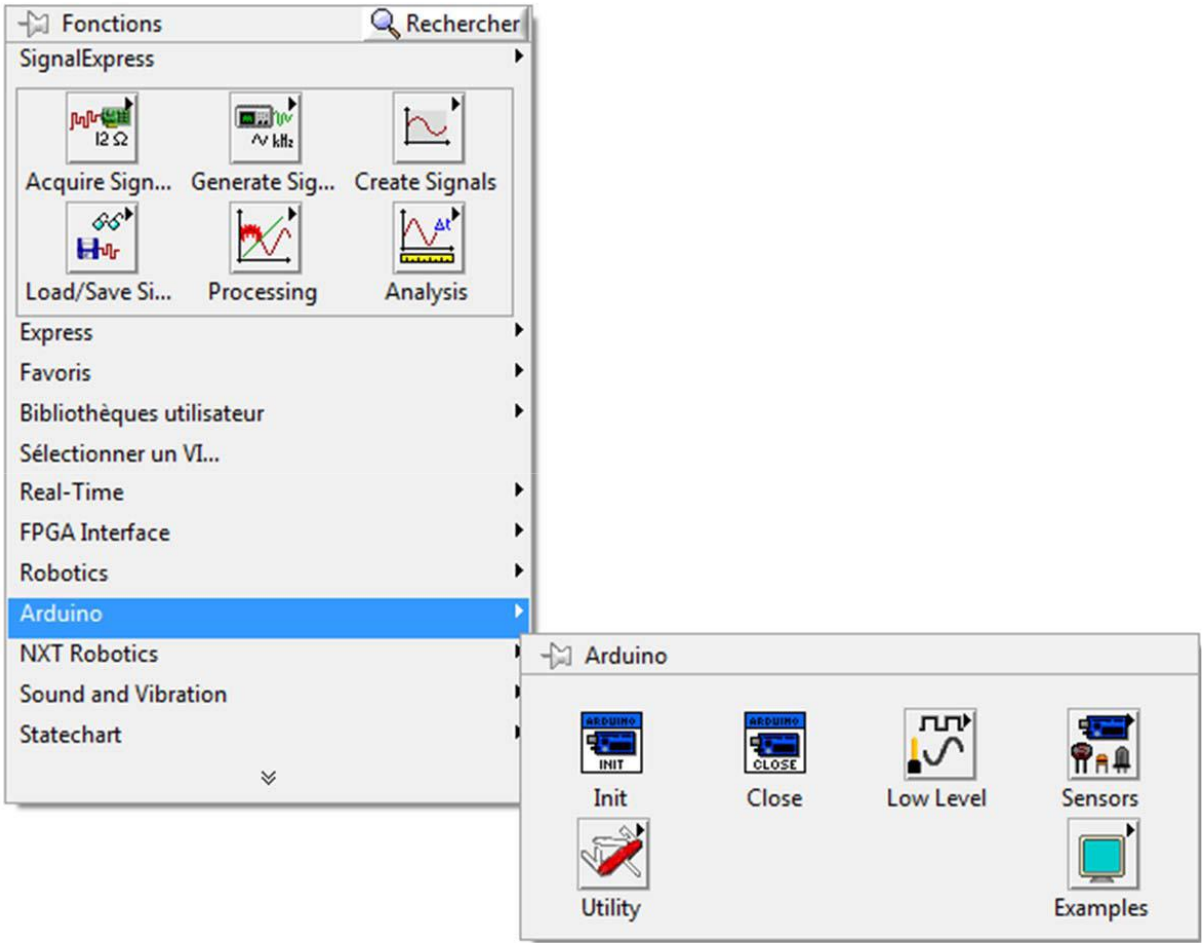


Figure III.24 Interface Labview Arduino.

Pour expliquer quelques fonctions utilisés par l'interface Labview Arduino (LIFA) nous allons procéder à la réalisation d'un ou deux programmes exemplaire, le première montre comment crée un programme de configuration des pins, écrire sur sortie digitale de l'Arduino. Et le deuxième montre comment lire une entré analogique via un potentiomètre.

❖ **Exemple 1**

La carte Arduino possède une LED connectée au Pin13. Aussi, sans rien brancher, il est possible de créer Un premier programme : l'allumage de la Led L disponible sur la carte Arduino. A l'aide de la palette Arduino.

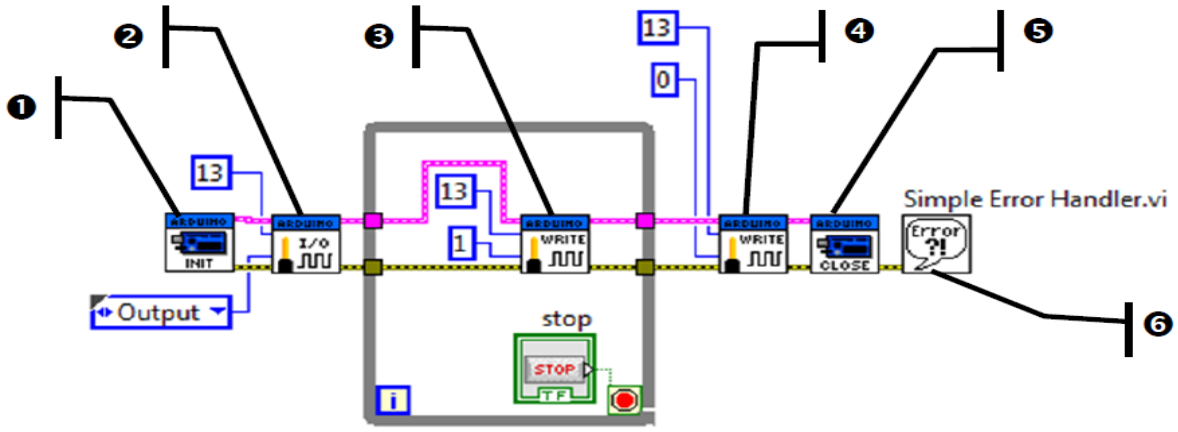


Figure III.25 Exemple VI Labview Arduino.

❶

❷ : Init – Initialise la carte Arduino.

: Set Pin Mode - Déclare si le Pin est une entrée ou une sortie. Dans notre cas, le *Pin 13* est une sortie.

❸

❹ : Digital Write - Ecrit la valeur indiquée - *ici 1* - sur le Pin digital concerné – *ici 13* -.

: Digital write. Ecrit *0* sur le *Pin 13*.

❺ : Arduino Close – ferme la carte Arduino.

❻ : Gestionnaire d'erreurs : Se trouve dans la palette Programmation – Dialogue/IU (Interface utilisateur) – Gestionnaire d'erreurs simples. Affiche une erreur avec son code s'il y a lieu.

❖ **Exemple 2**

Branchez un potentiomètre rotatif entre 0V et 5V, et de manière à ce que la tension variable soit envoyée sur l'entrée analogique 0 de l'Arduino. A l'aide de la palette Labview, le monitoring de cette entrée analogique au moyen du **VI** suivant :

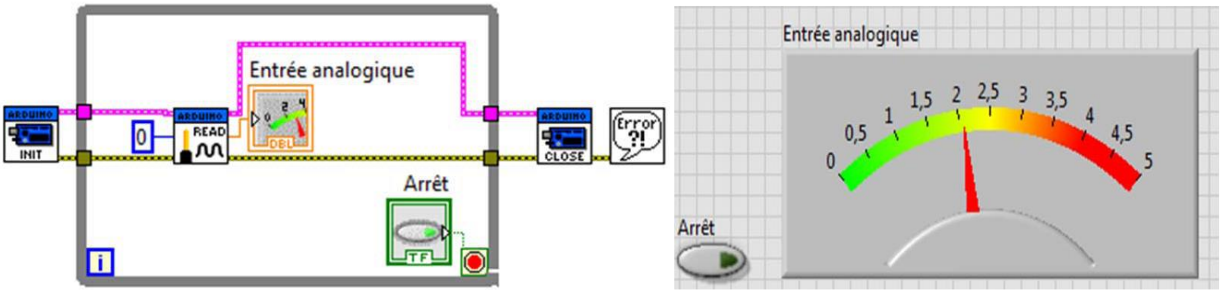


Figure III.26 Exemple VI d'une entré analogique à base LIFA.

III.7.3 Interface Arduino-Maker hub (LINX).

Le package Labview Interface for Arduino (LIFA) a été remplacé par **LINX**.

LINX fournit des VIs Labview faciles à utiliser pour interagir avec des plates-formes embarquées courantes comme Arduino, chip KIT et myRIO. Avec LINX 3.0, le logiciel a été étendu pour prendre en charge le Raspberry Pi et le BeagleBone Black, permettant aux cartes d'exécuter du code Labview. **LINX** vous permet d'utiliser les VIs de capteur intégrés pour commencer à envoyer des données à votre PC en quelques secondes ou d'utiliser les VIs périphériques pour accéder aux E/S numériques, E/S analogiques, SPI, I2C, UART, PWM et plus encore.

En son cœur, **LINX** est une couche d'abstraction matérielle qui vous permet d'avoir une seule interface Labview avec une variété de périphériques matériels différents. Selon le périphérique, nous pouvons accéder de deux manières: E/S distantes ou locales.

Pour pouvoir travaille avec les interfaces de LINX, sera indispensable au première lieu installer via le VIPM (VI Package Manager) le package de Maker hub-LINX. Le LINX après nous offrir et intègre dans sa boite d'outils plusieurs fonctionnalités présenté parmi sur la figure suivant [23]:

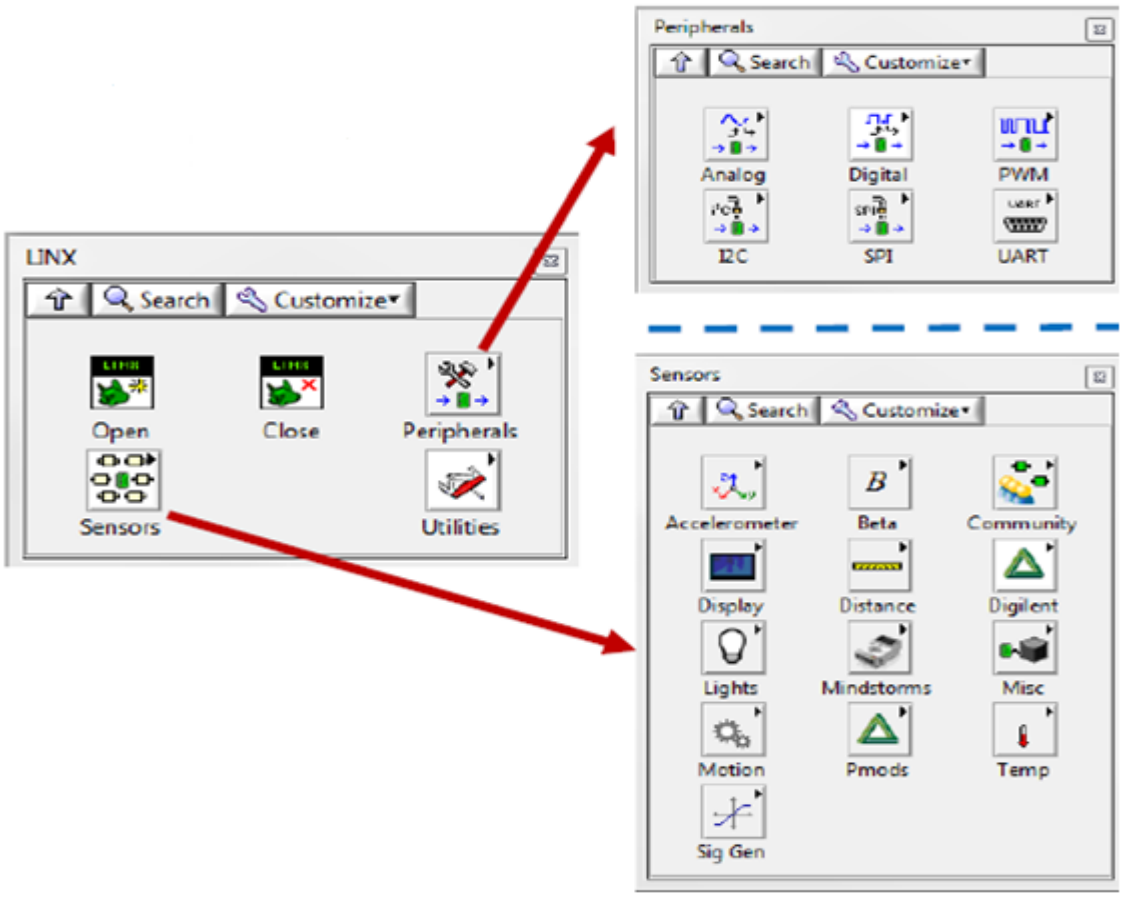


Figure III.27 Interface Labview Arduino-Maker hub LINX.

❖ **Exemple**

Dans cet exemple nous allons explique comment écrire une valeur sur un canal de sortie digitale de l'Arduino connecter à une LED, A l'aide de la palette Labview Arduino LINX, le monitoring de la sortie au moyen du **VI** suivant :

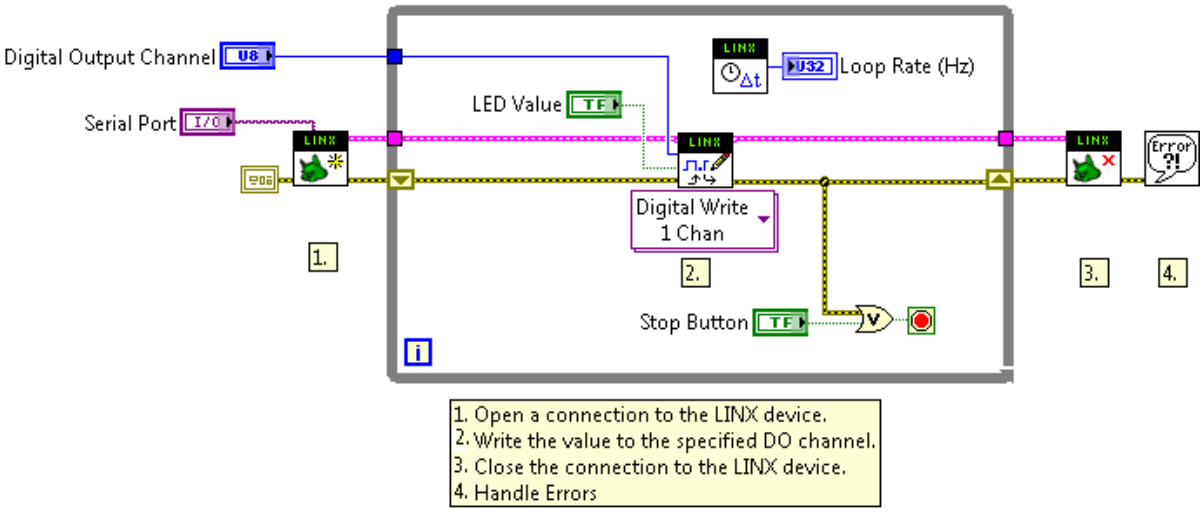


Figure III.28 Exemple sortie digital utilisent interface Labview Arduino-LINX.

N.B : Dans notre réalisation nous avons utilisé cette technique d'interface Labview Arduino LINX qui sera détaille dans le chapitre suivante.

III.8 Bibliothèque spécialisés dans Labview

Labview possède des bibliothèques de fonctions spécialisées dans le domaine de la mesure, du test et du contrôle-commande. Ces bibliothèques de fonctions peuvent être réalisées soit à partir des fonctions et operateurs de base vus précédemment, soit directement par des programmes écrits en langage de haut niveau, compile et ensuite intègre sous la forme d'un nœud graphique.

Dans notre projet on a besoin des bibliothèques spécifiques qui n'existent pas dans la version standard de Labview, mais dans des versions nommée (Labview RT) c.-a-d. Real time comme les bibliothèques de commande et de contrôle, en faite nous avons utilise des bibliothèques nommées « PID control toolset » pour la commande et la régulation de notre système. Il existe d'autre bibliothèque de commande telle que « fuzzy control toolset », « DSP control toolset » et quelques autres commandes avancées.

On va présenter d'abord la bibliothèque de commande que nous avons utilisée dans notre projet :

III.8.1 Bibliothèque de Commande et contrôle PID dans Labview

Elle ajoute aux fonctions standards de Labview un ensemble d'algorithmes de régulation P, PI, PD et PID, avec toutes les options que l'on trouve généralement dans les systèmes de régulation complets.

Tous les régulateurs dans cette bibliothèque ont une structure parallèle de la forme :

$$C(p) = \frac{U(p)}{\varepsilon(p)} = K_P + \frac{1}{pT_i} + pT_d \qquad \text{(III.1)}$$

Et voici un exemple d'une bibliothèque d'un régulateur PID.

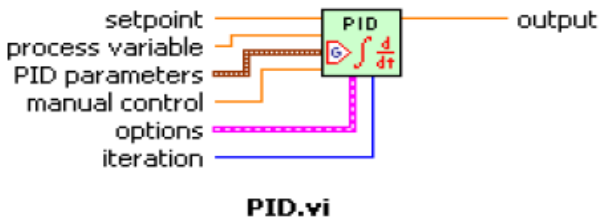


Figure III.29 Exemple de bibliothèque de régulation PID.

La composition ou structure interne d'une bibliothèque de l'algorithme PID est figure dans le schéma suivante :

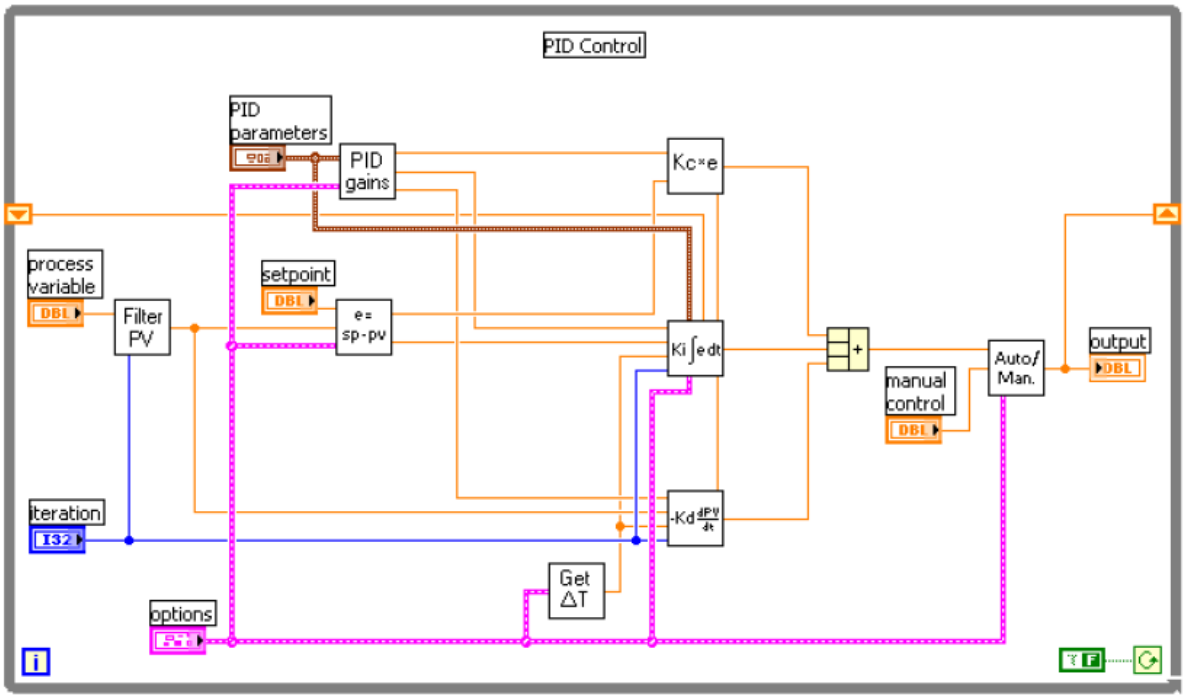


Figure III.30 structure interne de la bibliothèque PID.

III.8.2 Bibliothèque de Réglage PID automatique de la température

Cette bibliothèque du Labview nous donne la possibilité de contrôle et règle directement un système de régulation de température dans notre projet.

Cette dernière se base sur VI ci-dessous pour améliorer les performances de systèmes de Température mais aussi d'autres types de systèmes qui contiennent des temps morts. Ce VI utilise l'interne pour compenser le temps mort et régler le système.

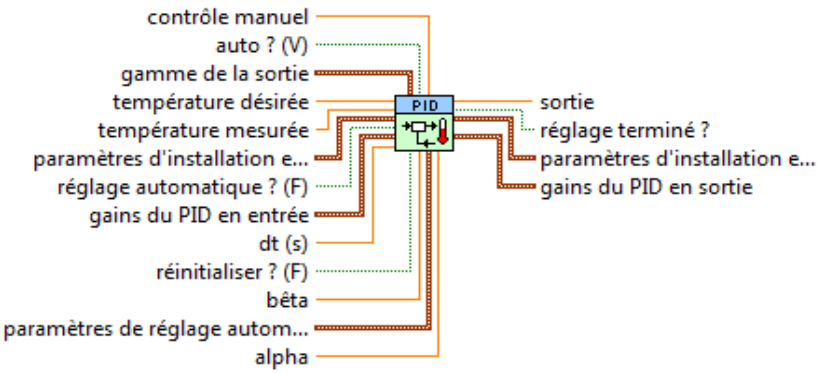


Figure III.31 Bibliothèque de régulation PID automatique de la Température.

III.8.3 Bibliothèque PID Advanced

Dans la régulation de niveau d'eau de notre projet nous avons utilisé un contrôleur PID avec des fonctionnalités optionnelles avancées. L'algorithme PID avancé comprend les fonctionnalités de l'algorithme utilisé par le VI PID, ainsi que le contrôle du mode manuel avec des transitions manuelles à automatiques sans à-coups, une action intégrale non linéaire, un contrôle à deux degrés de liberté et un contrôle au carré des erreurs. Utilisez l'instance DBL de ce VI pour implémenter une seule boucle de contrôle. Utilisez l'instance DBL Array pour implémenter un contrôle multi-boucle parallèle.

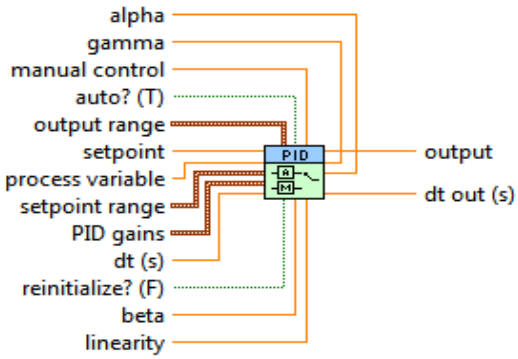


Figure III.32 Bibliothèque de régulation PID Advanced.

III.9 Conclusion

Dans ce chapitre nous avons exposé la partie logicielle que nous avons utilisée dans la programmation des interfaces de commande et contrôle de notre travail. La partie logicielle est dédiée à la présentation du logiciel Labview, ses fonctionnalités, sa structure et ses bibliothèques les plus utilisés.

CHAPITER IV

Etude et Réalisation Pratique du Régulateur et Contrôleur PID

IV.1 Introduction

Dans ce chapitre, nous étudions la partie Hardware et soft du projet en présentant la structure générale de notre maquette et les déférents composants électronique et mécanique utilisées pour la réalisation pratique du module de contrôle et commande PID d'un processus industrielle (régulation de niveau et température), ainsi que leurs schémas électronique de brochages d'une manière générale.

L'objectif principal de ce chapitre est de réalisé un module prototype, effectuant les fonctions de corrections numérique PID d'un signal analogique avec technique de supervision et assistance graphique par un HMI basé sur le mesure, affichage et commande par Labview.

Afin de mener à bien le projet de contrôle de niveau et température, il est nécessaire de procéder à une étude électromécanique et du logicielle afin de réaliser une série d'étapes pour garantir le bon développement optimal et pouvoir réaliser le fonctionnement du système contrôlé.

La réalisation pratique est divisée en deux grandes parties, une première application de la régulation de niveau d'eau d'un réservoir et une deuxième application sert au contrôle et régulation de la température.

IV.2 Description de système de la régulation de niveau

On se propose dans cette première application de présenter le système de régulation du niveau d'eau dans un réservoir. Ceci permettra de mettre en évidence les différents composants d'une boucle de régulation et de comprendre les actions PID du régulateur pour atteindre des performances désirées.

IV.3 Présentation le synoptique de la maquette

La maquette de régulation de notre projet est constituée de des éléments mécaniques et électroniques sont :

- Un réservoir haut de remplissage d'eau (réservoir de mesure).
- Un réservoir bas (réservoir de stockage).
- Un capteur de niveau à ultrason HC-SR04.
- Une pompe d'eau.
- Une vanne manuelle en haute pour Remplissage d'eau.
- Un actionneur (Servomoteur MG996R + Vanne manuelle) pour conditionnement de débit et décharge d'eau.
- Un module électronique contient deux cartes, une de l'acquisition, traitement et commande (Carte microcontrôleur Arduino MEGA 2560), et une autre carte d'interfaçage E/S.
- Un module d'alimentation standard ATX.

- Un kit (capteur LM35 + lampe 220V) pour la régulation de température.
- Des LED$_s$ de visualisation d'états de fonctionnement du système.

La régulation est assurée par un programme élaborer avec le logiciel LABVIEW, qu'on présentera après, la régulation est faite avec un PID industriel.

Le schéma synoptique de la structure de la maquette conçue est donne par la figure suivante.

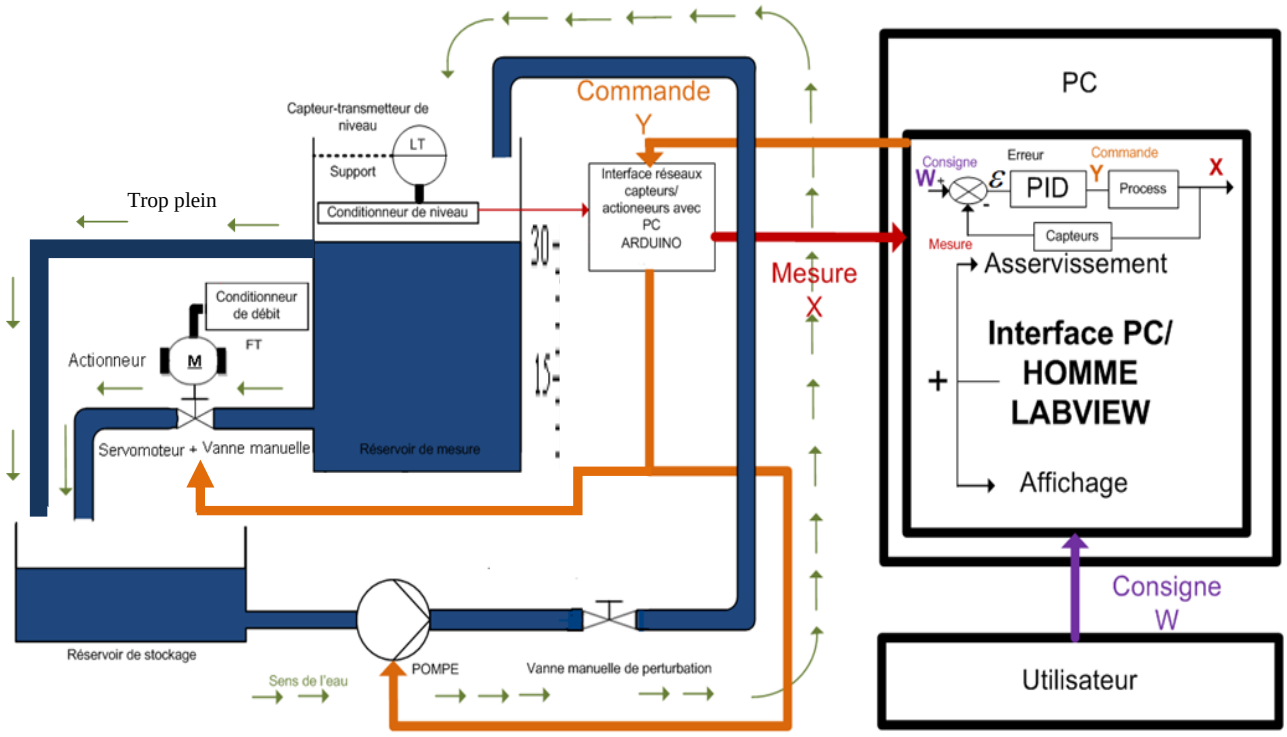


Figure IV.1 Schéma synoptique de la maquette régulation de niveau.

Nous expliquent brièvement la composition de notre maquette :

- Le réservoir haut (réservoir de mesure) est en plexiglas transparent permettant l'observation de l'évolution du niveau d'eau. Le réservoir haut est de forme parallélépipédique ou rond, de 30 cm de hauteur et de largeur, et de 12 cm de profondeur, une règle graduée permet de lire directement le niveau d'eau en cm. Ce réservoir présente une entrée de remplissage d'eau, une sortie d'évacuation ou d'usage.
- Le réservoir bas représente la source d'eau (stock), il est naturellement plus volumineux que le réservoir haut. Il est aussi de forme parallélépipédique ou rond, de plus de 30 cm de hauteur et de largeur, et de 15 à 20 cm de profondeur. Ce réservoir est aussi en plexiglas de préférence.
- La pompe permet le remplissage du réservoir haut, elle est composée d'une entrée d'aspiration de 8 mm de diamètre, d'une sortie de refoulement de 5 mm de diamètre,

La pompe est alimentée par une tension continue variable entre 0 à 24Vdc, elle consomme en fonctionnement nominal un courant de 1.5 A.

- Le capteur ultrason, de référence **HC-SR04** (voir Annexe A)**,** c'est un émetteur récepteur utilise les ultrasons pour déterminer la distance d'un objet (surface de l'eau), et afin de lire le niveau réel de notre réservoir d'eau de mesure.
- L'actionneur composé d'un servomoteur (de type MG996R) rotatif d'un angle de **0°** à **180°** et une vanne manuelle, l'ensemble du mécanisme compose une vanne régulatrice sert au conditionnement de débit d'eau et pilotage proportionnellement de l'ouverture et fermeture d'eau de vidange. L'actionneur est considéré l'élément principale de maintenir le niveau d'eau (régulation de niveau) dans le réservoir de mesure.
- La vanne manuelle placée sur le départ de refoulement d'eau permet de varier le débit de remplissage du réservoir haut.
- Le trop-plein a un diamètre suffisamment grand pour garantir deux fonctions : prévenir le débordement du réservoir haut, en cas de disfonctionnement du système ou en cas d'un dépassement important, et maintenir le réservoir bas a la pression atmosphérique empêchant ainsi la création d'une dépression lorsque la pompe aspire de l'eau.
- Un module électronique d'automation à base d'une carte Arduino Mega 2560 et une carte d'interface E/S, le module permet principalement le traitement des signaux provenant des capteurs, commande des actionneurs, et la communication avec la station de supervision et commande PC- HMI sous Labview.
- Un poste de commande PC contient l'interface graphique Labview de programmation de la carte Arduino, L'affichage des mesures, des paramètres de commande PID et l'enregistrement des donnés reçues du module via le port série USB.

IV.4 Développement du modèle ou prototype

Après avoir eu l'idée, nous avons procédé à l'obtention et choix du matérielle de la construction du modèle, ce qui a pris des heures de travail dans la conception de la pièce mécanique afin qu'un modèle homogène soit laissé, et le plus important était qu'il était fonctionnel; Nous avons pris en compte tous les détails mécaniques tels que le fluide du liquide, l'assemblage de la valve, l'installation du capteur et actionneurs, quelque chose qui nous préoccupait que le système était très hermétique et qu'il n'y avait pas de fuites, puis la

Partie électronique et de contrôle qui sera détaille dans le point suivant, enfin le modèle avait une ingénierie détaillée.

IV.5 Développement de la carte électronique du module (Partie matériel)

Dans cette partie nous détaillons les blocs constituent le module électronique (bloc d'alimentation, carte Arduino Mega 2560 et une carte d'interface E/S) qui permet l'acquisition des signaux des capteurs à l'unité de traitement à base microcontrôleur, la commande des actionneurs, la communication et échange de donnés avec la station PC- HMI. Ainsi que délivre les niveaux de tension nécessaire pour l'alimentation du module et nos périphériques.

Dans notre partie du matériel utilisé dans ce projet nous pouvons divisés en trois catégories :

- Bloc d'alimentation AC/DC.
- Carte Arduino et périphériques (Capteurs et Actionneurs).
- Carte d'interface E/S.

IV.5.1 Bloc d'alimentation

L'ensemble des dispositifs de notre projet nécessite une Alimentation stabilisée de plusieurs niveaux de tension telle que (0V, + 3.3V, +5V, +12V et +24V) de tensions continues, et une alimentation alternative 220AC. Pour cette raison nous avons l'idée de choisir la modification et mise en place d'un bloc d'alimentation utilisé dans le PC standard model ATX, qui possède une puissance suffisante pour alimenter nous périphériques et offrir d'autres services utiles dans notre maquette.

Le tableau ci-dessous présente le brochage des pins du bloc d'alimentation.

Couleur	Signal	Pin	Pin	Signal	Couleur
Orange	+3.3 V	1	13	+3.3 V	Orange
				+3.3 V sense	Brun
Orange	+3.3 V	2	14	−12 V	Bleu
Noir	Masse	3	15	Masse	Noir
Rouge	+5 V	4	16	Power on	Vert
Noir	Masse	5	17	Masse	Noir
Rouge	+5 V	6	18	Masse	Noir
Noir	Masse	7	19	Masse	Noir
Gris	Power good	8	20	Réservé	N/C
Violet	+5 V standby	9	21	+5 V	Rouge
Jaune	+12 V	10	22	+5 V	Rouge
Jaune	+12 V	11	23	+5 V	Rouge
Orange	+3.3 V	12	24	Masse	Noir

Tableau IV.1 Connexion alimentation 24 pins model ATX.

IV.5.2 Carte Arduino et périphériques (Capteurs et Actionneurs)

Pour pouvoir commander le procède (dans notre cas c'est un niveau d'eau dans un réservoir et température ambiante) avec une loi de commande numérique a travers un PC, il est nécessaire d'utiliser une carte de traitement CPU à base microcontrôleur Arduino et qui a pour but principale de transformer les signaux numérique a des signaux analogique équivalente et les signaux analogiques en des signaux numérique équivalente.

En plus et afin de contrôler le déroulement d'un processus industriel, il est obligatoire d'utilisé des capteurs de phénomène physique et des actionneurs de commande vis vers sa.

IV.5.2.1 Carte Arduino Mega 2560

Le bloc de traitement et de commande se résume au module Arduino Mega 2560 décrit précédemment au chapitre II. Le bloc de traitement et commande est relié essentiellement via une carte d'interface E/S à un :

- ✓ Capteur ultrason HC-RS04 dans le but de mesure le niveau d'eau (ou liquide).
- ✓ Un capteur de température LM35 pour pouvoir capter la température.
- ✓ Un servomoteur MG996R utilisé comme actionneur et pilote de la vanne de régulation de niveau.

La station PC-HMI à est relié avec la carte acquisition Arduino à travers le COM/USB pour envoi des mesures et reçois les commandes utilisateurs.

IV.5.2.2 Les Capteurs utilisées

- **Capteur ultrason HC-RS04 :**

  Ce module dispose simplement de 4 pins de sortie : VCC, TRIG, ECHO, GND.

Figure IV.2 Capteur ultrason HC-SR04.

Il est donc très facile de l'interfacer à un microcontrôleur. Le processus complet est le suivant: Mettre le pin "TRIG" une impulsion de niveau haut (5V) durant au moins 10µs et le module démarre sa lecture; A la fin de la mesure, s'il détecte un objet devant lui, le pin "ECHO" passe au niveau haut (5V) "Pulse In ()". Et, la distance où se situe l'obstacle est proportionnelle à la durée de cette impulsion Il est donc très

Facile de calculer cette distance On utilise la vitesse moyenne que parcourt le son (dans l'air) soit 340m/s ce qui donne 29.1 us/cm.
Pour calculer la distance on pose :

Distance = Temps mesuré / Vitesse

Distance = Temps mesuré / 29.1 us

Mais ici on a la distance aller retour donc on va devoir diviser par 2 la distance ce qui nous donne : **Distance = distance / 2**

Et en simplifiant le calcul on aura : **Distance = temps mesuré / 58**

Caractéristiques HC-SR04:

- Alimentation: 5 Vcc.
- Consommation: 15 mA.
- Fréquence: 40 kHz.
- Portée: de 2 cm à 4 m.
- Déclenchement: impulsion TTL positive de 10µs.
- Signal écho: impulsion positive TTL proportionnelle à la distance.
- Calcul: distance (cm) = impulsion (µs) / 58.
- Dimensions: 45 x 21 x 18 mm

- **Capteur LM35 :**

Le LM35 est un capteur de température permet d'acquérir une température ambiante, une fois que alimenté, il délivre une tension analogique proportionnelle à la température. La variation de la tension est très faible, c'est pourquoi nous utilisons un amplificateur opérationnel. Sa représentation graphique est une droite linéaire

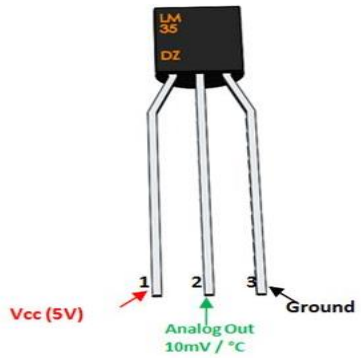


Figure IV.3 Capteur température LM35DZ.

Le capteur LM35 est capable de mesurer aussi des températures allant de -55°C à +150°C dans sa version la plus précise et avec le montage adéquat.

Le LM35 est un capteur de température de précision qui peut être facilement étalonnée. Il fonctionne comme un Zener 2-terminale et la tension de claquage est directement proportionnelle à la température absolue à l'10mV / ° K.

Il fournit 0 V pour 0 degré, 250 mV pour 25 degrés, etc. et un maximum de 1 volt pour 100 degrés.

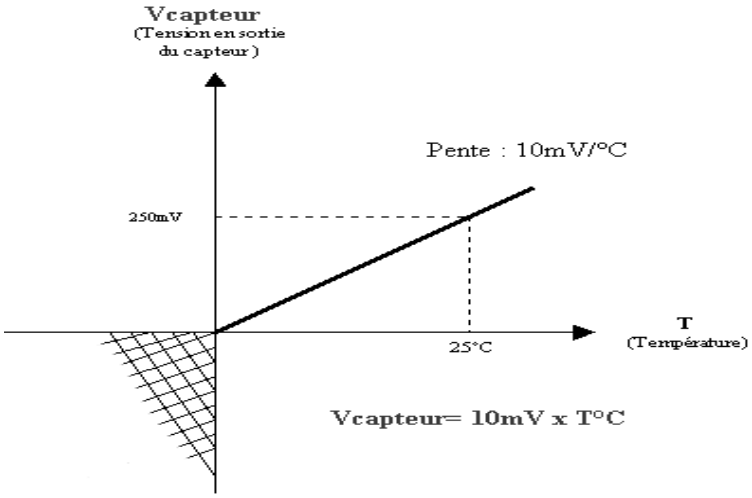


Figure IV.4 Tension en fonction de la Température.

En utilisant une entré analogique du Arduino **A0** la température est donnée par la relation suivant :

Température (C°) = 100 * tensions

Caractéristiques LM35 DZ:

- Tension d'alimentation : 4V à 30V.
- Etendu de mesure : 0°C à 100°C.
- Précision : ±0,75 °C (typique).
- Echelle : 10mV/°C.
- Calibration : 0mV à 0°C, 1000mV à 100°C

IV.5.2.3 Les Actionneurs

- **Servomoteur MG 996R :**

Le servomoteur est moteur un peu particulier, qui peuvent tourner avec une liberté d'environ 180° et garder de manière relativement précise l'angle de rotation que l'on souhaite obtenir.

Dans notre réalisation nous avons utilisé le servomoteur pour piloter et contrôle le mécanisme mécanique d'ouverture et fermeture de la vanne de régulation de niveau d'eau.

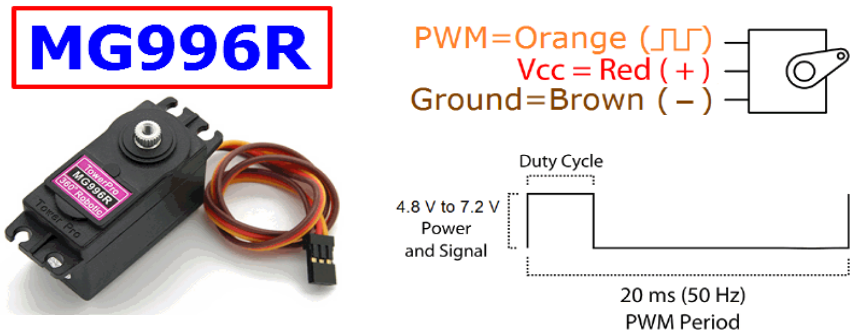


Figure IV.5 Servomoteur MG996R et pins de connexions.

Spécification MG996R:

- Poids: 55g.
- Longueur du fil servomoteur: 32cm.
- Couple de décrochage: 9,4 kg / cm (4,8 V); 11 kg / cm (6,0 v).
- Vitesse de fonctionnement: 0,19 s / 60 degrés (4,8 v); 0,15 s /60 degrés (6,0 v).
- Tension de fonctionnement: 4,8 ~ 6,6 v.
- Type d'engrenage: engrenage en métal.
- Plage de température: 0- 55deg.
- Prise servomoteur: JR (pour JR et Futaba).
- Largeur de bande morte: 1us.

- **Pompe à eau (moteur DC) :**

La pompe électrique est l'élément essentielle d'actionnement dans notre circuit de fluide, elle à une fonction d'aspiration de l'eau et injection dans le réservoir de mesure. Nous avons choisir une pompe à eau basse pression et faible tension de consommation DC.

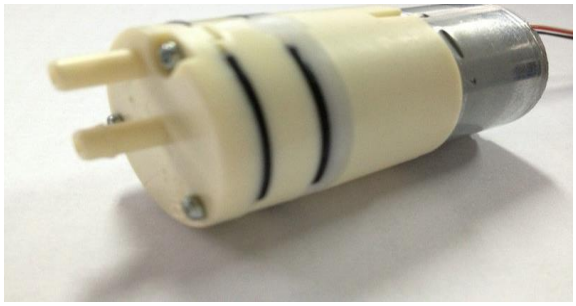


Figure IV.6 Pompe à eau électrique basse pression.

Spécification:

- Tension nominale: DC 24V.
- Charge: Eau.
- Courant (avec charge): moins de 150mA.
- Débit: 2,0 LPM.
- Taille: D27 x 65mm.
- Pression maximale: plus de 600 mmHg.
- Bruit: moins de <60 dB

IV.5.3 Carte interface E/S

Afin de relier et protéger les commandes entrées sorties de la carte Arduino et ces périphériques, nous avons réalisé une carte d'interface entre l'unité d'acquisition et traitement et tous les périphériques (capteurs de mesure + actionneurs), voir figure ci-dessous.

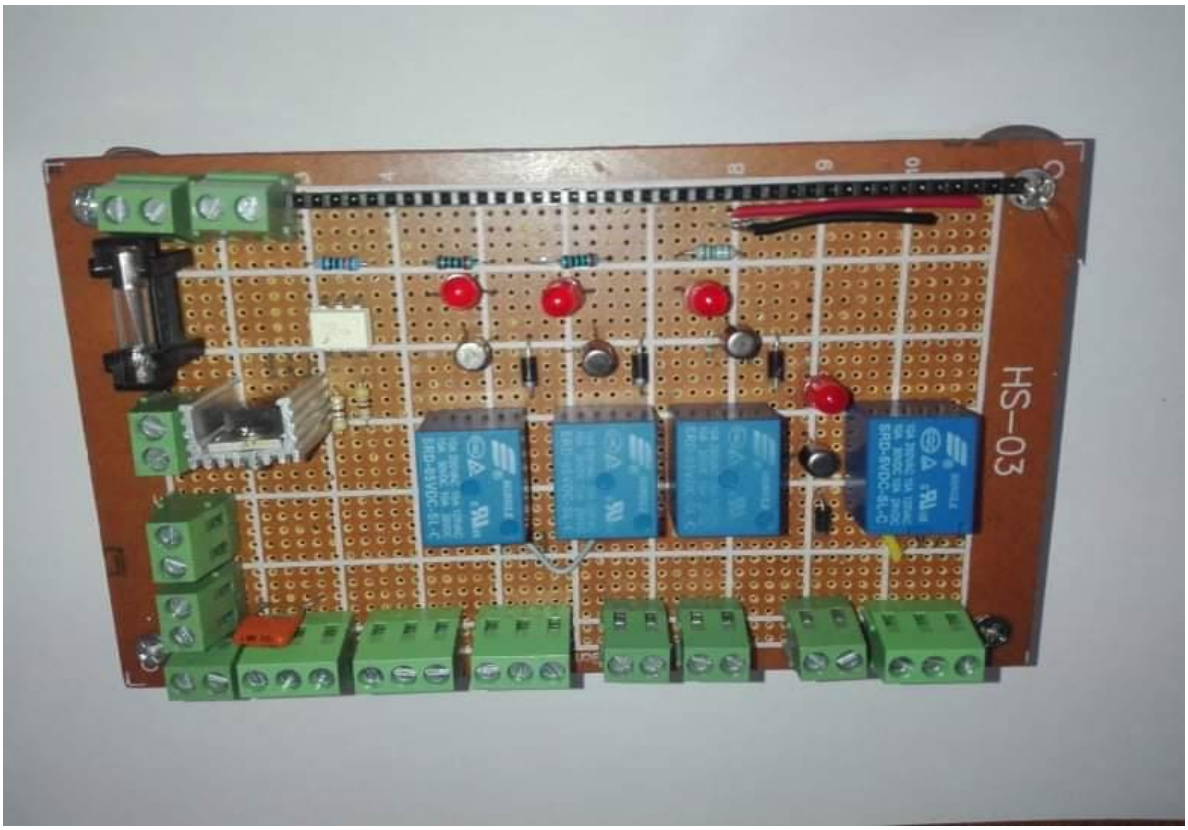


Figure IV.7 La carte d'interface E/S réalisée

Pour la réalisation de cette carte on utilise les composants électroniques présentés dans le tableau IV.2 ci-dessous :

Tableau IV.2 Composants utilisés pour réaliser la carte de commande E/S [24]

Désignation	Description
 Relais SDR-5V	Un relais est un commutateur électrique qui permet de commander un second circuit utilisant généralement une tension et un courant bien supérieur à ce que l'Arduino pourrait accepter 10A en 220V / 10A en 24V / 10A en 110V.
 Transistor NPN : BC 107	Transistor BC107 : est un transistor à jonction bipolaire NPN. Un transistor, signifiant transfert de résistance, est couramment utilisé pour amplifier le courant. Un petit courant à sa base contrôle un courant plus important aux bornes du collecteur et de l'émetteur.
 Diode 1N4007	La diode est un composant électronique qui ne laisse passer que le courant dans un sens. C'est le sens passant, ou direct. Le sens ou aucun courant ne passe est le sens bloqué, ou inverse.
 TRIAC BT 139	Le triac est composé de deux thyristors opposés commandés par une seule gâchette, et est un composant électronique utilisé comme interrupteur commandé. Il est adapté aux tensions alternatives et peut commander des moteurs électriques, des variateurs de lumière ou toute autre charge secteur. Contrairement au thyristor qui ne peut conduire que dans un sens, le triac peut conduire dans les deux sens. Il est bidirectionnel
 Opto-coupleur MOC3023 double protection	Un Opto-coupleur est un dispositif composé de deux éléments électriquement indépendant, mais optiquement couplés, un photo-triac dans notre cas (LED photo + triac). Le rôle d'un Opto-coupleur est soit d'assurer une isolation galvanique (aucune liaison électrique) entre deux systèmes électriques pour des utilisations diverses comme ▪ Interface pour la transmission de données. ▪ Commande de structures Basse Tension (Secteur EDF). ▪ Variation de puissance

 Des Résistances	Résistance: est un composant électronique ou électrique dont la principale caractéristique est d'opposer une plus ou moins grande résistance. Nous avons utilisés plusieurs valeurs de résistance tel que (100, 220,330 ,470) Ω et 1K Ω
 es Condensateurs	Les condensateurs de 10uF, 100Nf que nous avons utilisé principalement pour : ▪ Stabiliser une alimentation électrique (il se décharge lors des chutes de tension et se charge lors des pics de tension). ▪ Traiter des signaux périodiques (filtrage…). ▪ Séparer le courant alternatif du courant continu, ce dernier étant bloqué par le condensateur.
 Des LED	LED 5v : Les diodes électroluminescentes de différentes couleurs sont parfaites pour constituer une voie de retour visuelle dans nos projets, nous avons utilisées pour la visualisation des états de fonctionnement de notre système de régulation.
 Bornier électrique	Bornier électrique : est un dispositif permettant d'assurer la continuité électrique entre un câble et une autre partie de l'installation.
 files de connections	C'est un fil électrique qui relie les dipôles d'un circuit entre eux. Son rôle est de permettre de circulation le courant électrique entre ces dipôles.
 Barrette 20pins de connections double	C'est une baratte de connexion et liaison entres pins qui assure le contact et la conduction du courant électrique.
 Disipateur termique (Radiateur)	Un dissipateur thermique est un dispositif destiné à favoriser l'évacuation des pertes dissipées par les éléments semi-conducteurs de puissance. Il s'agit de dispositifs généralement munis d'ailettes, qui doivent de préférence être montées verticalement pour faciliter le refroidissement par convection.

IV.5.4 Le schéma général du module électronique

Le schéma de La figure (IV.8) représente la configuration générale de notre module et la connexion avec tous les périphériques de la maquette en utilisant fritzing.

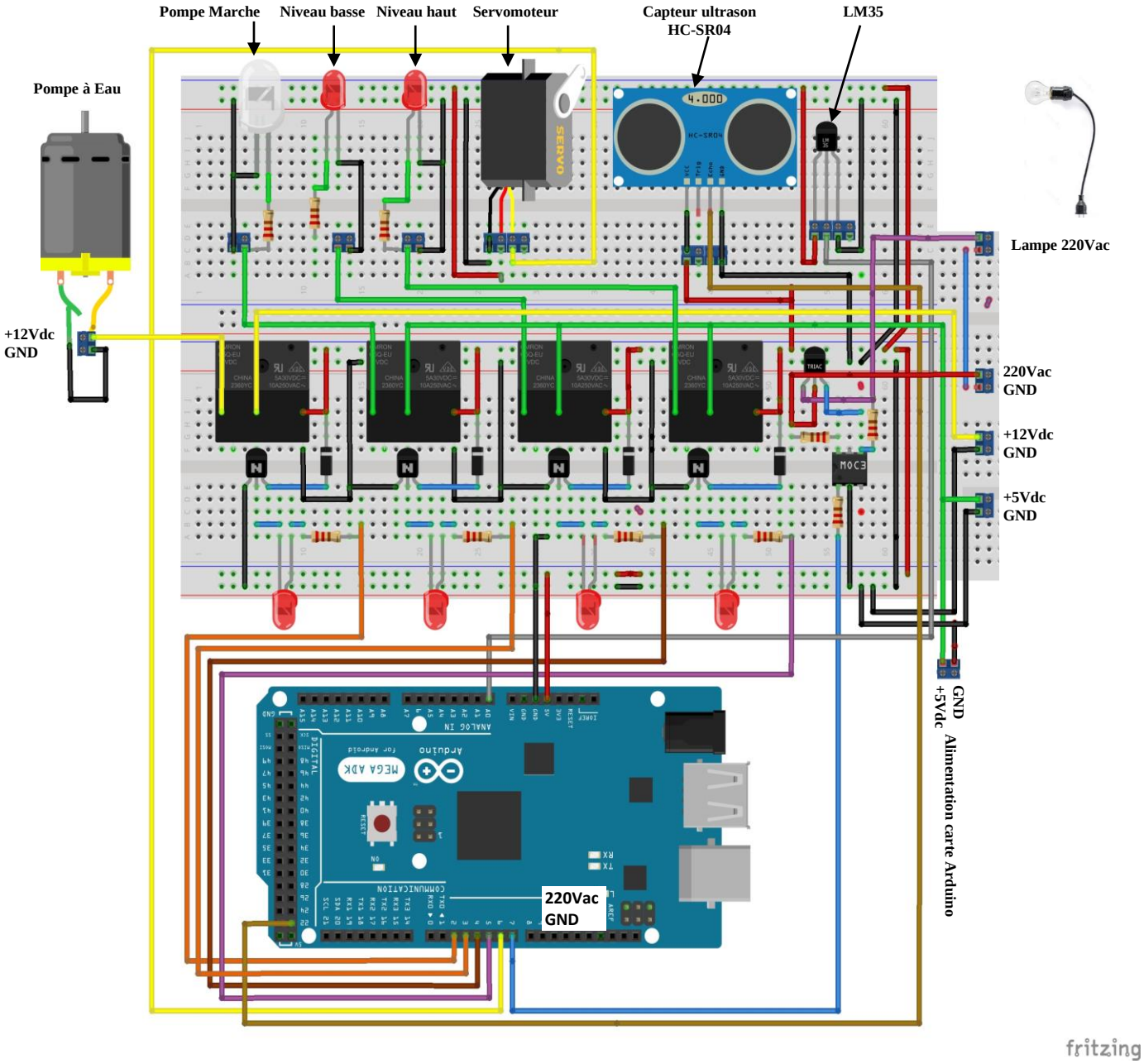


Figure IV.8 Schéma générale de la carte du module électronique

IV.6 Développement du programme et conception HMI sous Labview

Pour cette partie soft nous allons divisées le travaille en deux grandes parties, une pour la programmation et conception HMI de la régulation PID de niveau d'eau, et la deuxième partie sert à la contrôle et la régulation PID de la température.

IV.6.1 Programme de l'application PID de régulation de niveau d'eau

Pour la programmation nous avons procédé à l'installation de la boîte à outils nécessaire à la communication de l'Arduino avec Labview, comme l'interface LINX Labview pour Arduino qui contient tous les bibliothèques nécessaires de programmation tel que :

- ✓ Control Design & simulation qui contient l'algorithme PID.
- ✓ La bibliothèque pour contrôler le capteur à ultrasons HC-SR04 et Servomoteur MG996R.
- ✓ Entrés et sortie analogique.
- ✓ Entrées Sortie digitales TOR et PWM.

Ensuite, nous avons procédé aux tests de chacun des composants électroniques du système (capteur à ultrason, servomoteur et relais pour la pompe) séparément pour vérifier leur bon fonctionnement et leur étalonnage via la bibliothèque Labview LINX.

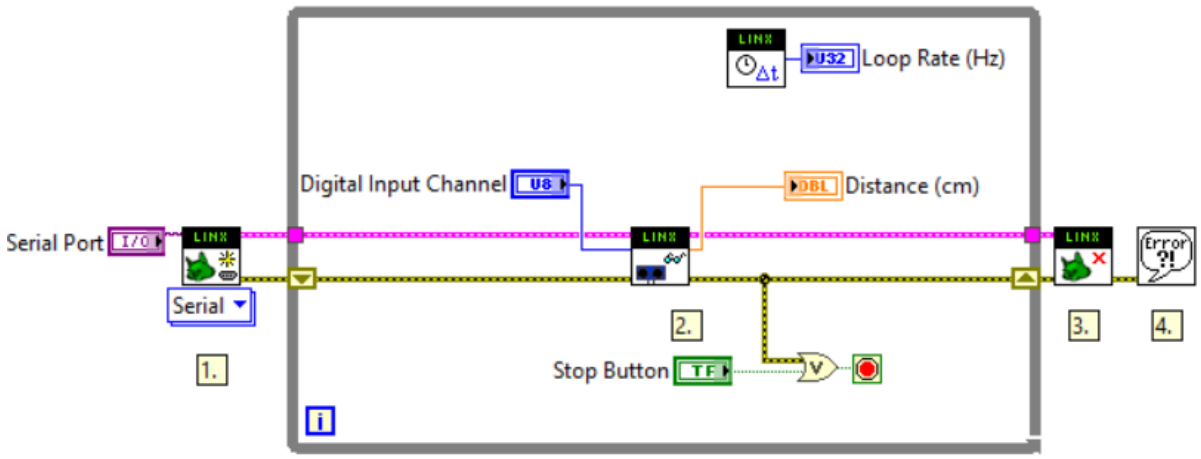


Figure IV.9 Programme de mesure et étalonnage ultrasonique HC-SR04.

1. Ouvrez une connexion à l'appareil LINX.
2. Lire la valeur du capteur à ultrasons connecté au canal d'entrée numérique spécifié.
3. Fermé la connexion à l'appareil LINX.
4. Gérer l'erreur.

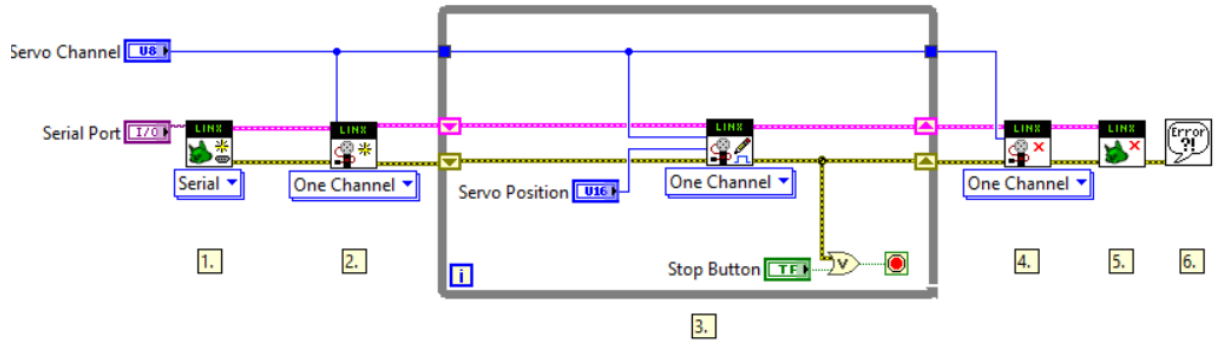


Figure IV.10 Programme de pilotage et commande du servomoteur

1. Ouvrez une connexion à l'appareil LINX.
2. Ouvrir le canal servomoteur spécifie. Notes bien que le canal ne peut pas être utilisé comme DIO tant que le servo n'pas été fermé.
3. Écrivez une largeur d'impulsion sur le servo pour définir le positon.
4. Fermer le canal Servomoteur.
5. Fermé la connexion à l'appareil LINX.
6. Gérer l'erreur.

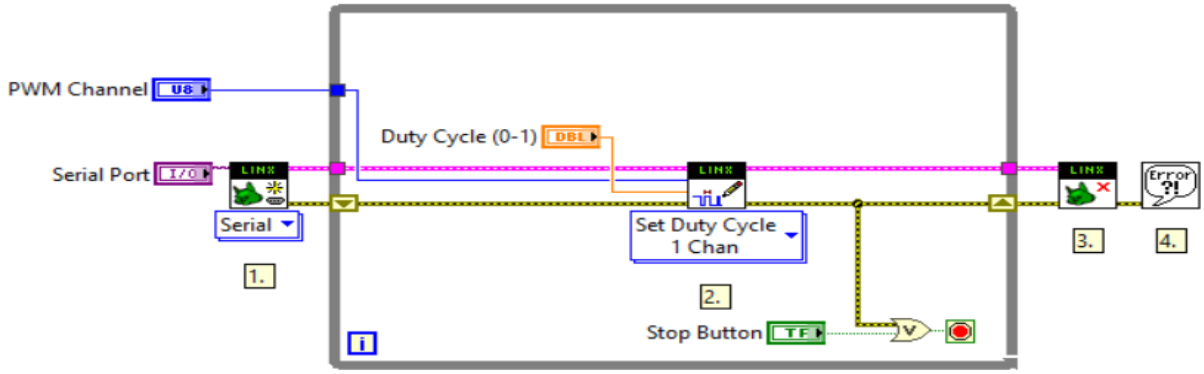


Figure IV.11 Programme de teste marche / arrêt pompe d'eau

1. Ouvrez une connexion à l'appareil LINX.
2. Définition le rapport cyclique du canal PWM spécifié (pin PWM Arduino).
3. Fermé la connexion à l'appareil LINX.
4. Gérer l'erreur.

Ensuite, après avoir vérifié le bon fonctionnement de chacun des éléments du système, nous avons procédé à l'intégration de tous les codes dans un seul VI, nous avons à nouveau testé la connexion du système et effectué un contrôle manuel de la vanne.

Après avoir pu contrôler manuellement, nous avons programmé un VI complet intégrant l'algorithme de contrôle PID, qui nous permettrait d'enregistrer le comportement naturel de ce type de système. Pour cela, nous avons implémenté le code d'une boucle de régulation suivant:

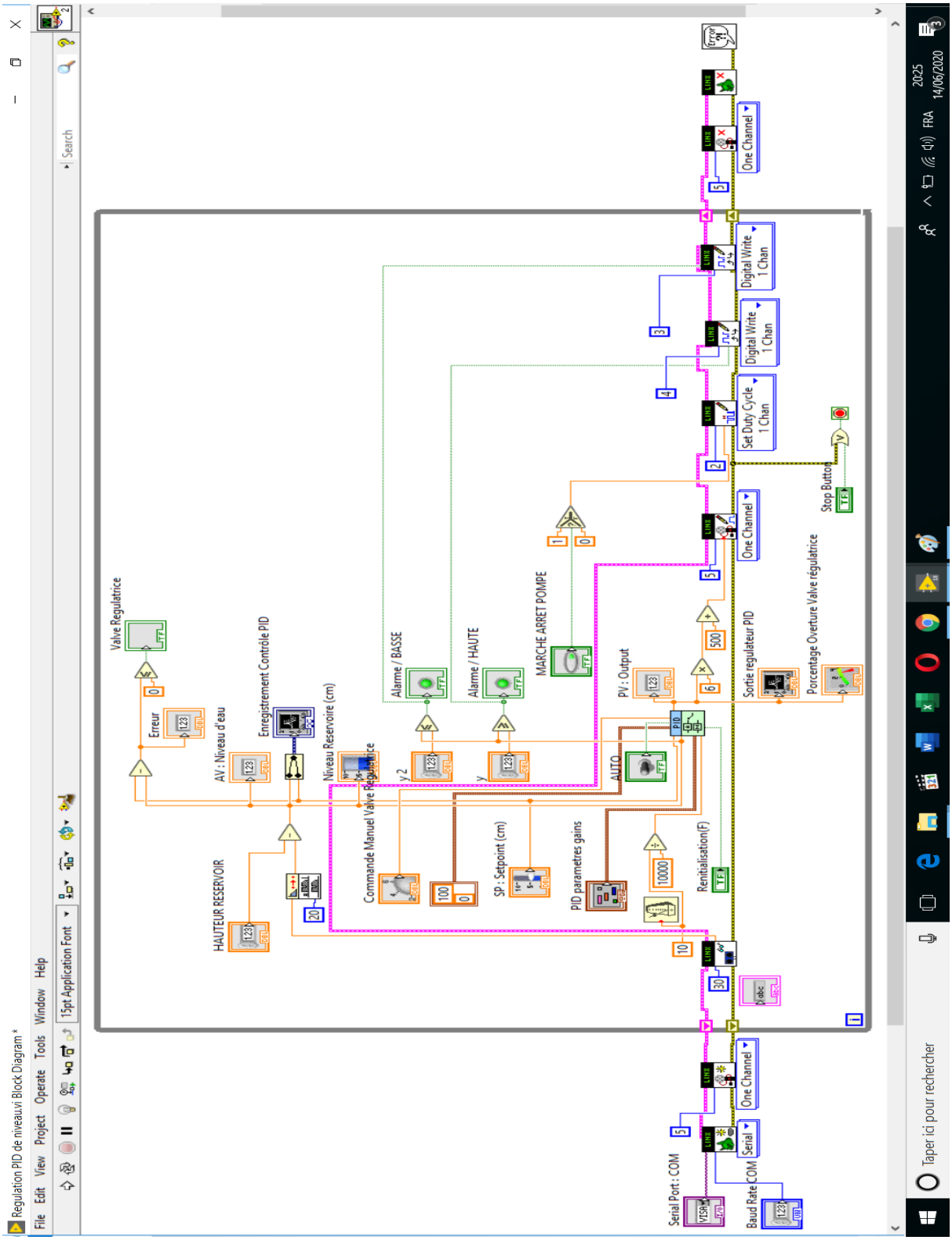


Figure IV.12 Programme de la boucle de régulation de niveau

Le résultat final de notre HMI de supervision et contrôle PID de niveau d'eau est la suivant :

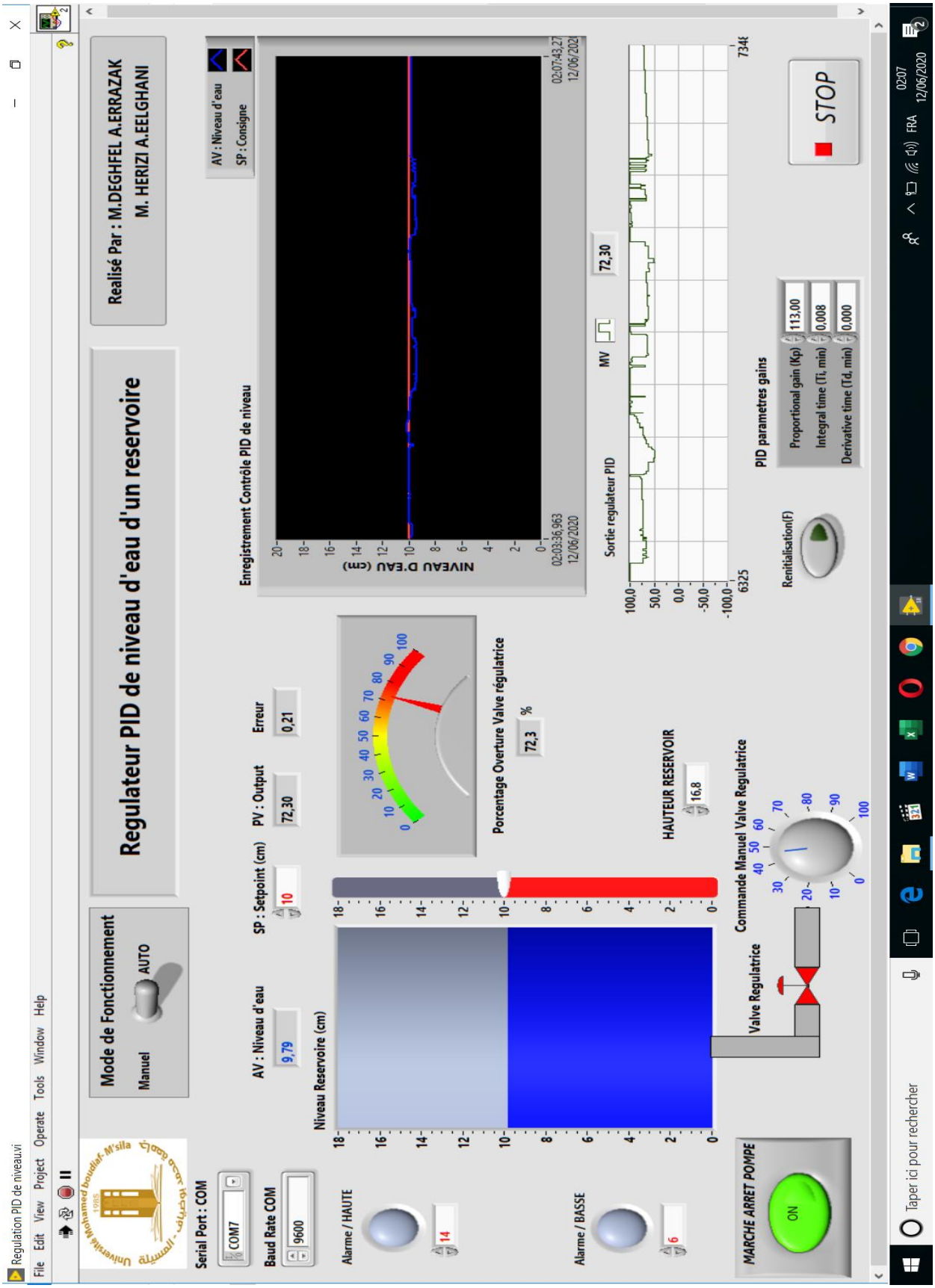


Figure IV.13 Synoptique HMI de commande et régulation de niveau

IV.6.1.1 Teste, réglage des paramètres

Après avoir préparé le programme et le modèle, nous procéderons à la connexion de tous les appareilles (PC, module électronique et ces périphériques), et suivre les instructions du mode opératoire de paramétrage de la partie communication logicielle et ajustement des paramètres d'exploitation de l'utilisateur comme la suivante :

- L'utilisateur doit sélectionner le port **COM** du PC et sa vitesse figurent sur l'HMI.
- Introduit la consigne **SP** désiré en cm.
- Fixer les valeurs des alarmes de bas et haut niveau selon besoin.
- Sélection au première lieu le mode de fonctionnement on manuel.
- Saisir les paramètres PID de départ tel que :

  Kp=20, **Ti=0.08 min, Td=0.001 min**,

La détermination de ces paramètres est faite grâce soit

- ❖ à la méthode des approximations successives. Cette méthode consiste à faire des testes sur chacune des actions du régulateur Indépendamment des autres actions.
- ❖ Soit par autoréglage automatique qui implanté dans l'algorithme PID basé sur le principe de calcule du Zeigler-nicols.

Après le démarrage de l'HMI en mode manuel, nous observons comment le servo se déplace, comment le capteur HC-SR04 communique et mesure la distance, nous observons également le bon fonctionnement de la pompe d'eau.

Pour faire des testes de notre régulateur PID nous pouvons procédure au démarrage en mode manuel en ferment complètement la vanne régulatrice afin de commence à remplir le réservoir de mesure, et basculant en mode automatique d'Eque le système commence à cherche et soit proche du point de référence (consigne fixé).

IV.6.1.2 Résultats expérimentaux

- **1ére cas (SP=11cm)** pour KP=113, Ki=1.36 et Kd=0.001

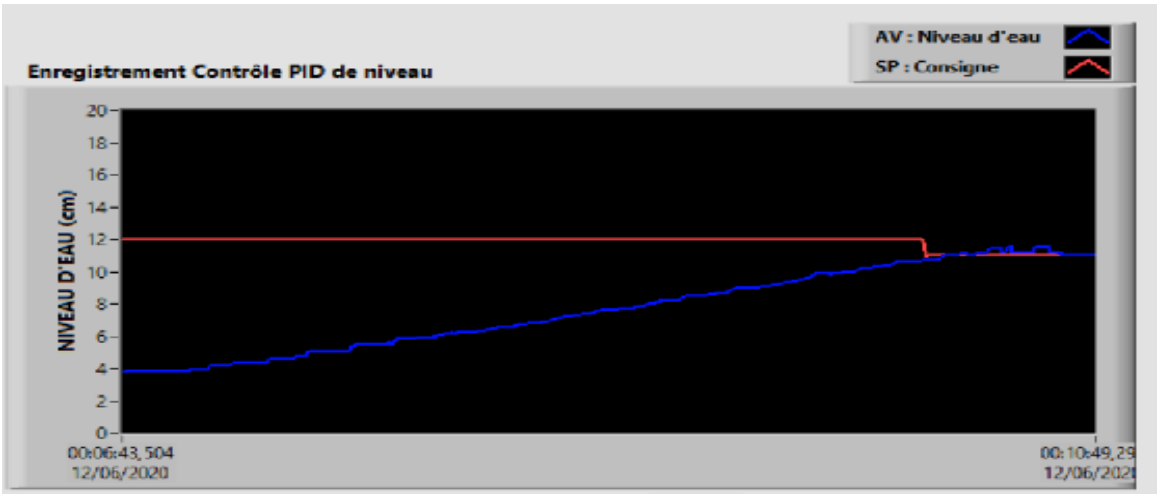


Figure IV.14 Réponse du système à une consigne de 11 cm

- **2éme cas (SP=9cm)** pour KP=113, Ki=1.36 et Kd=0.001

  Dans ce cas nous allons perturber de la vanne de régulation par changement de consigne brusque de valeur haut **9cm** à une valeur basse **6cm** par exemple.

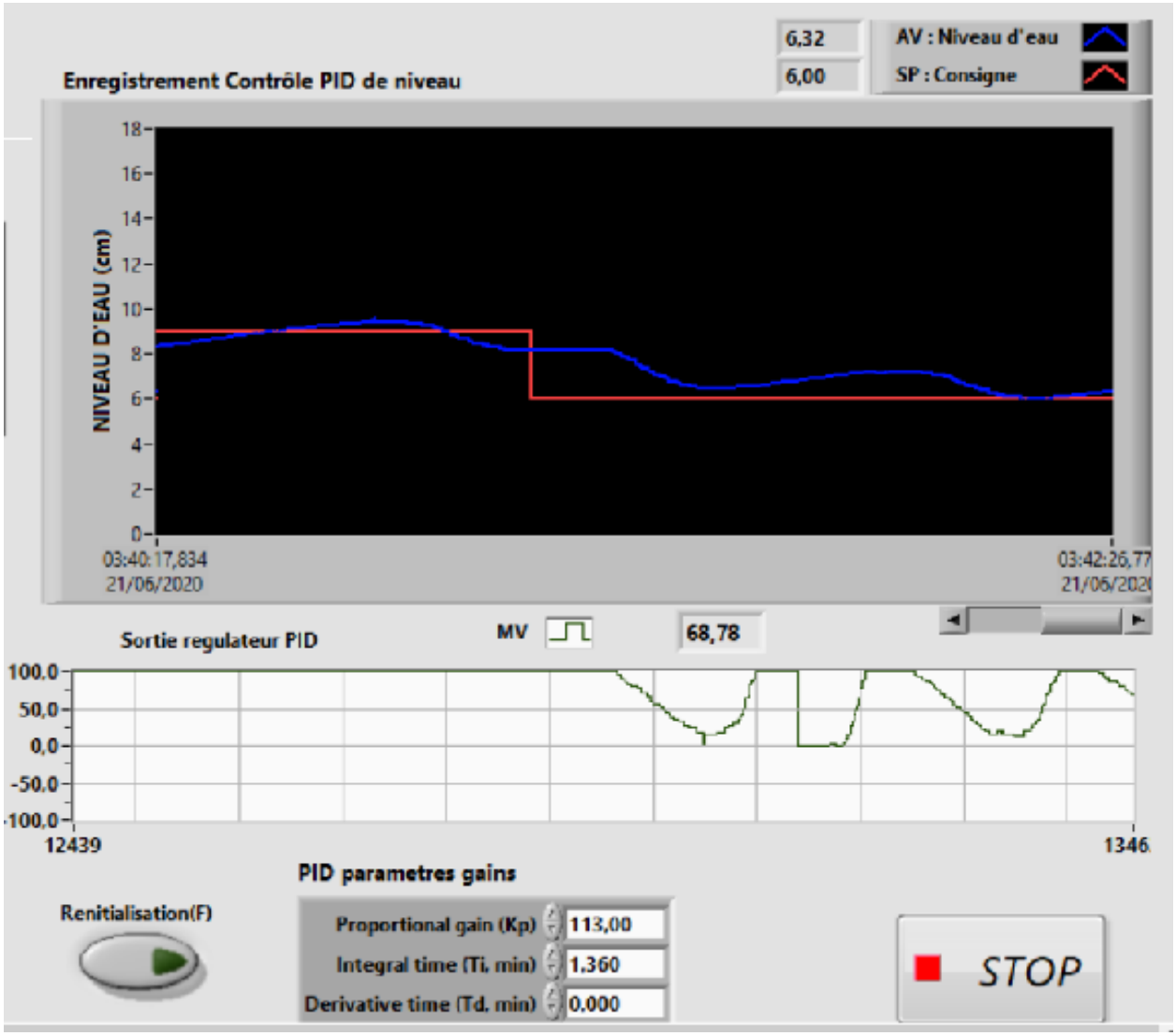


Figure IV.16 Réponse du système de changement de consigne (de 9 à 6) cm

Nous constatons dans cette cas notre régulateur réagie et suite le changement de consigne de l'utilisateur automatiquement par une sortie proportionnelle à la consigne de référence.

IV.6.2 Programme et résultats de l'application PID de régulation de température

Dans cette partie d'application nous avons utilisé le capteur LM35 pour la mesure de la température ambiante, l'acquisition, le traitement et affichage en temps réel des valeurs et des données reçus de ce capteur font via une bibliothèque de l'entrée analogique du LINX de Labview, chaque valeur reçu (Tension de 0-5V) par le module est multiplié par 100 pour avoir la valeur exacte de la température mesurée. Ensuite toutes ses valeurs seront affichées dans l'interface graphique pour faciliter à l'utilisateur l'exploitation, supervision des données et intervention rapide au cas des problèmes.

Pour faire la régulation manuel et automatique l'algorithme PID (Bibliothèque de LINX dans Labview), acquérir la mesure, la consigne et paramètres de réglage PID pour faire la régulation et maintenir la valeur désiré, ce dernière envoi des signaux de commande et sort des valeurs proportionnelle au l'actionneur de source de chaleur pour attendre la consigne.

Le programme de la boucle de régulation résulte est la suivant :

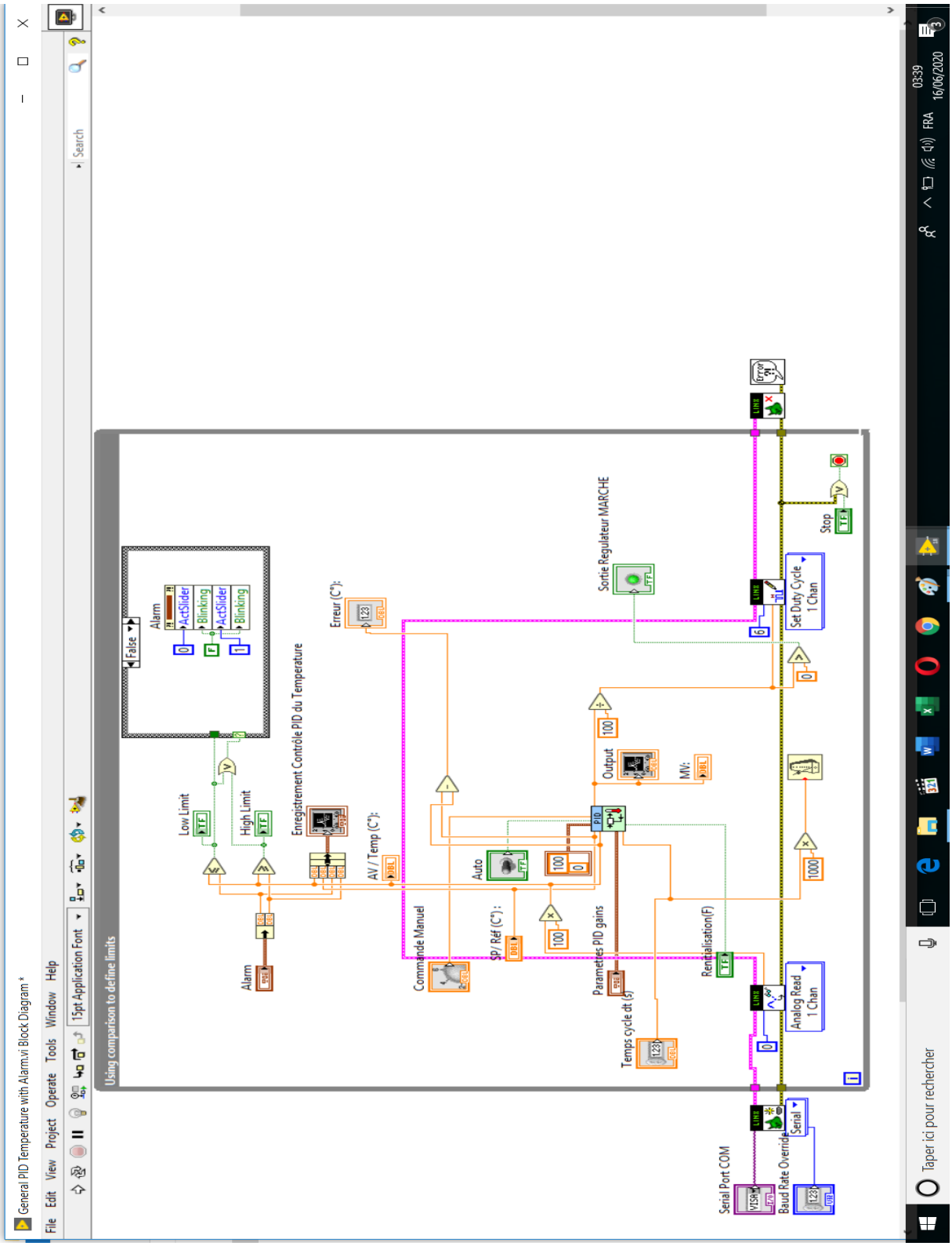


Figure IV.16 Programme de la boucle de régulation de Température

Après avoir l'HMI de supervision et contrôle PID de la temperature, nous avons procédure à faire des testes en mode automatique et voici les résultats pour une consigne SP=60C°, KP=20, Ti= 0.008 min, Td= 0.001min :

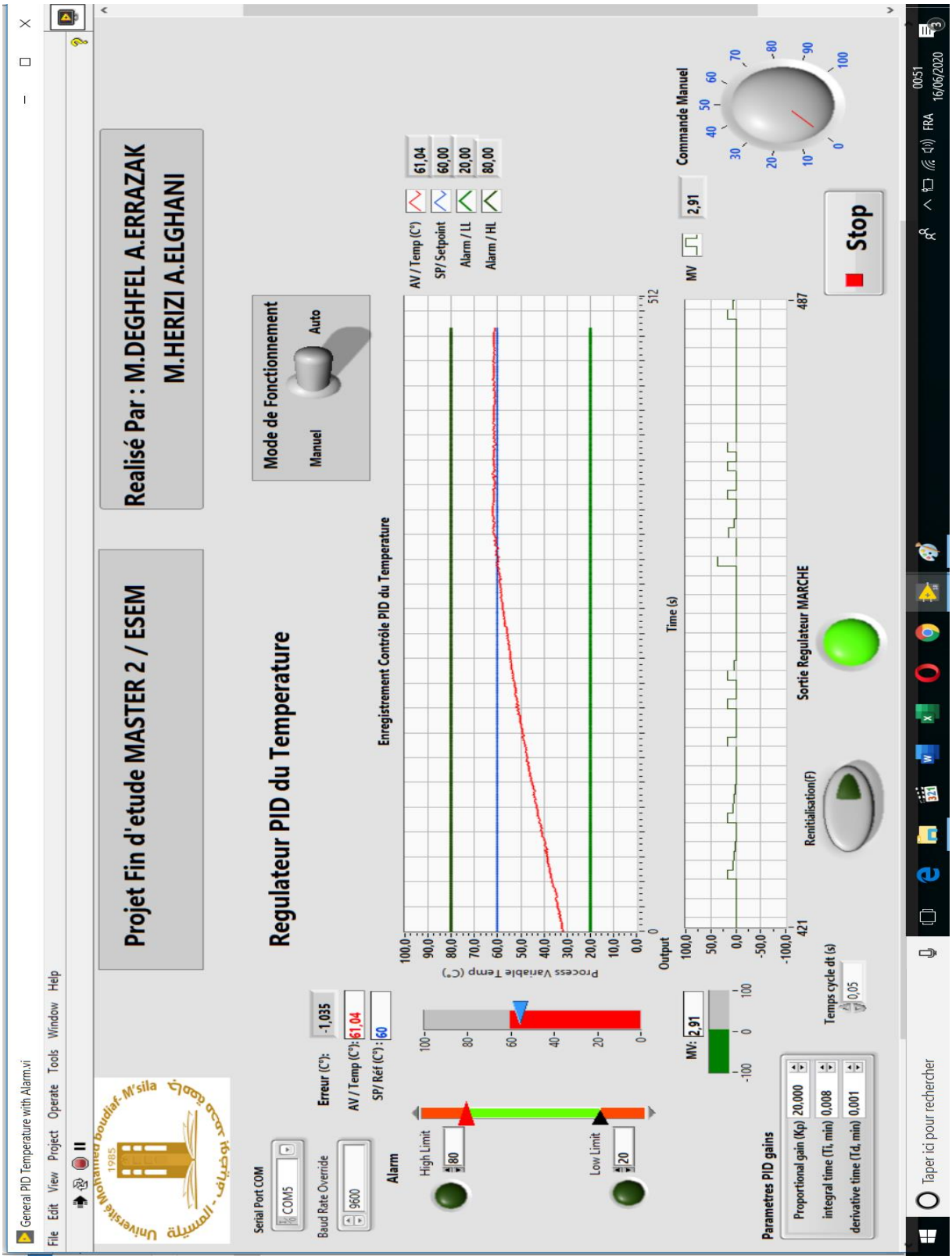


Figure IV.17 Synoptique HMI et résultats de régulation de la température

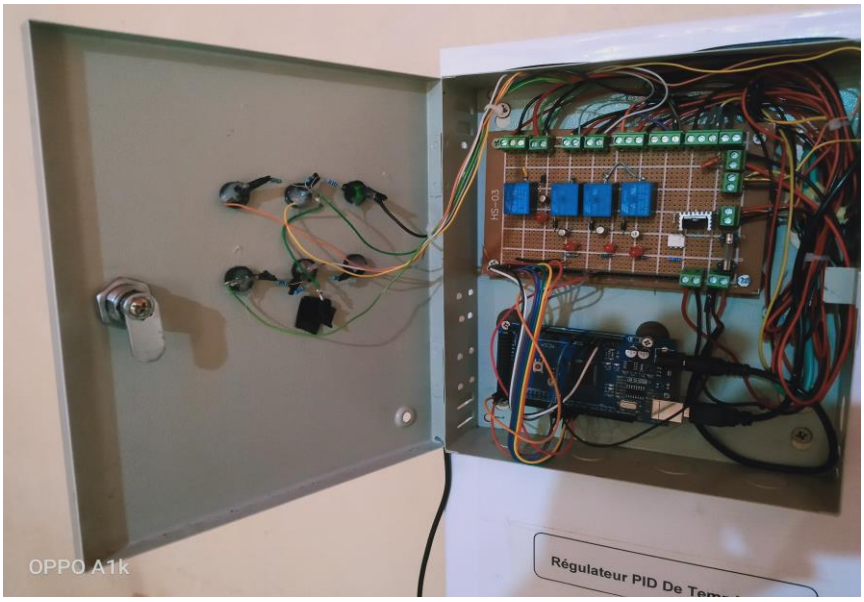


Figure IV.18 Le module électronique réalisé

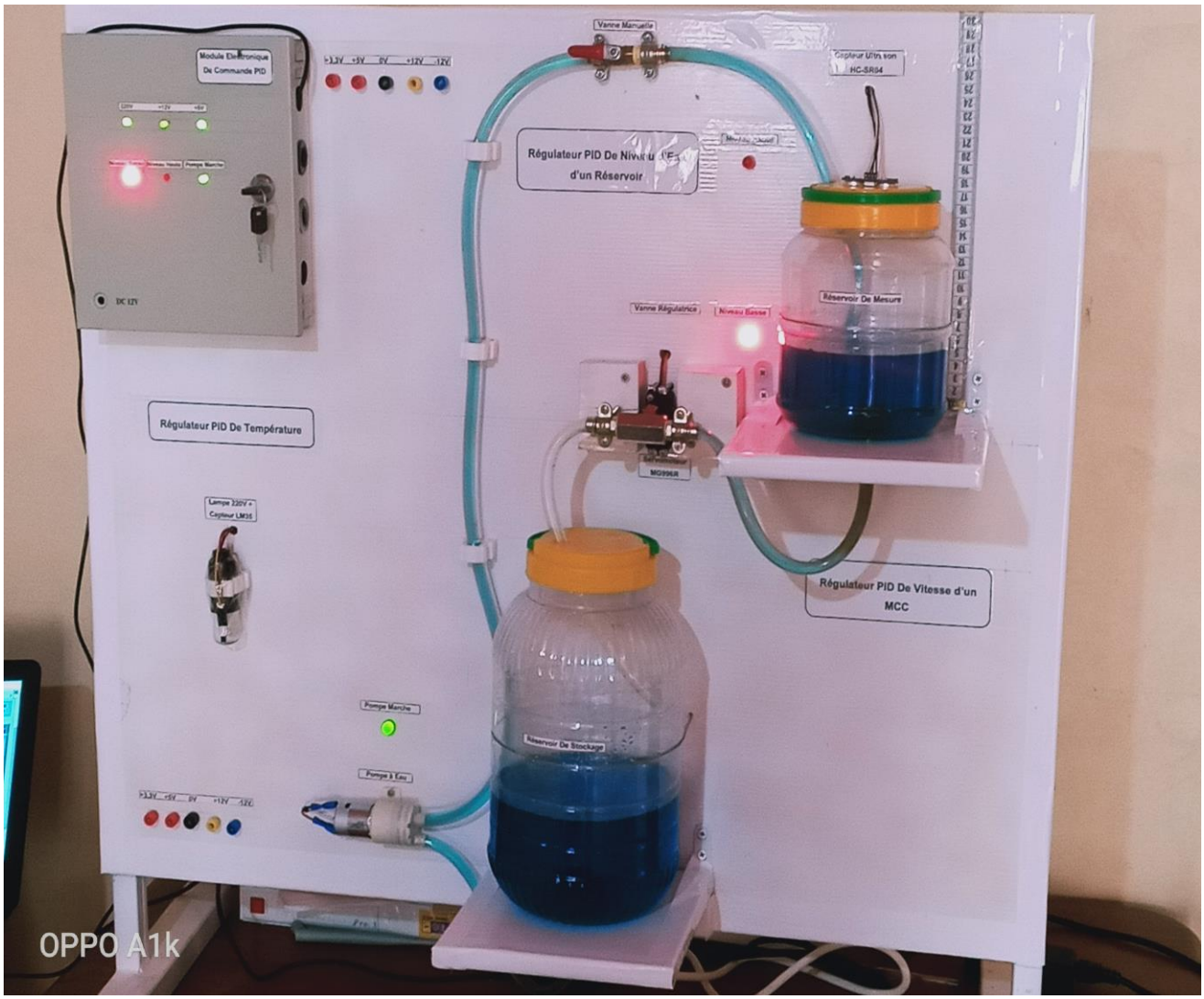


Figure IV.19 maquette et pont d'essai final

IV.7 Conclusion

Dans ce chapitre nous avons étudié et réalisé la partie soft et hard du module électronique utilisée pour le contrôle et régulation PID de la température et niveau d'eau. Ce module électronique est constitué d'une carte Arduino Mega 2560 qui permet d'assurer l'acquisition, la conversion analogique numérique, et une carte d'interfaçage E/S pour le pilotage des périphériques (capteurs, actionneurs, etc.…). Pour piloter notre système, nous avons réalisé une interface graphique HMI à l'aide du logiciel Labview qui nous a permet de supervisé et contrôle le déroulement du processus industrielle en temps réel.

Cependant, au préalable nous avons testé et vérifier l'ensemble de circuits intégrés constituant cette carte d'interface. Cet essai nous a permis de confirmer le bon fonctionnement et la fiabilité de la carte électronique avant son couplage au port de PC.

Deux types de régulateurs sont utilisés, un régulateur de type PI pour la régulation de niveau, et un régulateur PID complet pour la régulation de la température.

Après l'analyse des courbes de réponse du système en boucle ouvert et fermée pour les deux types de régulateurs nous concluons que notre système de commande fonctionne et réagisse correctement.

CONCLUSION GENERALE

L'objectif de ce travail est consacré vers la mise en œuvre d'une stratégie de la commande et régulation PID le plus utilisé au monde industriel, Ce travaille présente une étude et réalisation expérimentale de cette commande numérique qui permet de contrôler dans notre cas la température ambiante et le niveau d'eau d'un réservoir, cette commande est une combinaison d'une commande par station PC doté par logicielle Labview et par carte d'acquisition à base Arduino (microcontrôleur).

Dans notre travaille nous avons bien voulu au première chapitre rappeler les définitions et notions de basses sur la régulation PID, ainsi que les méthodes classique de calcules des régulateurs telles que la méthode de Ziegler-Nicole.

Le deuxième et troisième chapitre consiste à présenter en détaille les moyennes matériels et logicielle de programmation utilisées dans cette pratique.

La dernière partie constitue en premier lieu à l'étude et la conception d'un modèle prototype purement mécanique (Domain des fluides), ainsi que la réalisation d'un module électronique de commande (carte d'acquisition+ carte d'interface), à base d'une carte Arduino Mega 2560. Ce module sera une jonction entre la station PC-HMI et le processus via un port COM série.

En deuxième lieu l'implémentation et l'élaboration du software sous forme d'interface HMI (Homme Machine Interface) sous Labview installer dans un PC, qui permet le pilotage et supervision en temps réel de l'ensemble des paramètres du processus de manière visuelle.

L'ensemble du hard et soft à été mise en service et testée étape par étape dans deux mode de fonctionnement manuel et automatique, afin de s'assurer le bon fonctionnement et de sa fiabilité avant de mise à la disposition d'un utilisateur.

En fin nous avons mis en place une nouvelle façon de programmer un Arduino en tant que carte d'acquisition de données et sa bonne communication avec le logiciel Labview, en plus des avantages que ces deux technologies nous permettent de développer et d'optimiser tout processus industriel ou mécatronique.

Pour des développements futurs plusieurs propositions peuvent être envisagées On peut suggérer de développer les points:

- Le développement et réalisation de la commande PID flou pour la régulation de niveau.
- Création d'un fichier Excel de stockage des données et un tampon des alarmes pour faciliter au technicien le diagnostique des problèmes.
- Mette le module sous réseau via un protocole de communication Ethernet TCP/IP, pour éliminer le problème de distance ente le PC de commande et le module, cela va nous permettre de mette ce module avec un API sous le système OPC server.

BIBLIOGRAPHIE

[1] : Régulation industrielle –Notion de Base Article PDF.

[2] : Cours d'Automatique 2005-2006 Bernard Bayle Ecole Nationale supérieur de Physique De Strasbourg.

[3] ALINA BESACON- VODA et SYLVIANNE GENTIL Régulateurs PID analogique et Numérique Technique de l'ingénieur : R7416 (03/1999).

[4] : Performances système asservi –Article PDF.

[5] : Henri Bourles, Hervé Guillard, 2012, commande des systèmes, Performance et Robustesse, ellipses.

[6] : Prouvost, P. (2005). Automatique : contrôle et régulation, Dunod.

[7] : https://fr.wikipedia.org/wiki/Régulateur_PID.

[8] 'cours commande numérique des systèmes', Mr K.KARA, Université SAAD DAHLEB Blida Faculté de Technologie, Septembre 2015.

[9] Régulation automatique : Chapitre 4: Régulateur PID, Michel ETIQUE, Haute Ecole D'Ingénieurs et de Gestion du canton de Vaud (HEIG- *´Vd*).

[10] Mémoire fin d'étude commande d'un moteur à courant continu octobre 2011, université Saad Dahleb BLIDA.

[11] Document sur la Régulation de la température dans une serre agricole par commande PID (chapitre.pdf).

[12] Instrumentation et régulation en 30fiches (*prof patrick prouvost)* agrégé en génie Mécanique au Lycée Val de Murigny à Reims.

[13] http://blog.ac-versailles.fr/technopeguy/public/Programmation/Arduino.pdf mai 2017

[14] https://www.flossmanualsfr.net/_booki/arduino/arduino.pdf Mai 2017

[15] ERIK Bartmann. « Le grand liver d' Arduino ». 2eme Edition. 2015. Liver.

[16] http://fr.farnell.com/atmel/atmega2560-16au/micro-8-bits-256k-flash-5vcms /dp/ 1288330# techDocsHook Mai 2017.

[17] www.cours-gratuit.com--Course LabView-id5006.pdf.

[18] www.cours-gratuit.com--Course LabView-id4491.pdf.

[19] MARTAJ, Nadia et MOKHTARI, Mohand. Apprendre et maîtriser Labview par ses Applications. Springer Science & Business, 2014

[20] www.cours-gratuit.com--Course LabView-id5009.pdf.

[21] www.cours-gratuit.com--Course LabView-id5007.pdf.

[22] Guide d'utilisateur du logiciel Labview.

[23] https://www.labviewmakerhub.com/doku.php?id=get_labview.

[24] https://www.gotronic.fr/cat-composants-electroniques-5.htm.

Annexe (A)

Triacs sensitive gate — BT139 series E

GENERAL DESCRIPTION

Passivated, sensitive gate triacs in a plastic envelope, intended for use in general purpose bidirectional switching and phase control applications, where high sensitivity is required in all four quadrants.

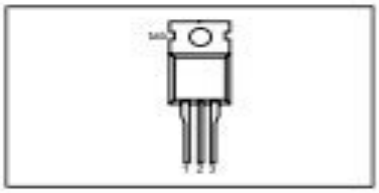
QUICK REFERENCE DATA

SYMBOL	PARAMETER	MAX.	MAX.	UNIT
	BT139-	600E	800E	
V_{DRM}	Repetitive peak off-state voltages	600	800	V
$I_{T(RMS)}$	RMS on-state current	16	16	A
I_{TSM}	Non-repetitive peak on-state current	140	140	A

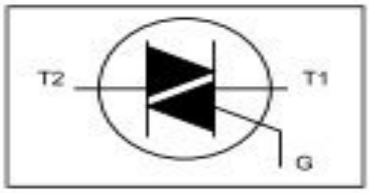
PINNING - TO220AB

PIN	DESCRIPTION
1	main terminal 1
2	main terminal 2
3	gate
tab	main terminal 2

PIN CONFIGURATION



SYMBOL



LIMITING VALUES

Limiting values in accordance with the Absolute Maximum System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.		UNIT
V_{DRM}	Repetitive peak off-state voltages		-	-600 600[1]	-800 800	V
$I_{T(RMS)}$	RMS on-state current	full sine wave; $T_{mb} \leq 99$ °C	-	16		A
I_{TSM}	Non-repetitive peak on-state current	full sine wave; $T_j = 25$ °C prior to surge				
		t = 20 ms	-	140		A
		t = 16.7 ms	-	150		A
I^2t	I^2t for fusing	t = 10 ms	-	98		A^2s
dI_T/dt	Repetitive rate of rise of on-state current after triggering	$I_{TM} = 20$ A; $I_G = 0.2$ A; $dI_G/dt = 0.2$ A/µs				
		T2+ G+	-	50		A/µs
		T2+ G-	-	50		A/µs
		T2- G-	-	50		A/µs
		T2- G+	-	10		A/µs
I_{GM}	Peak gate current		-	2		A
V_{GM}	Peak gate voltage		-	5		V
P_{GM}	Peak gate power		-	5		W
$P_{G(AV)}$	Average gate power	over any 20 ms period	-	0.5		W
T_{stg}	Storage temperature		-40	150		°C
T_j	Operating junction temperature		-	125		°C

1 Although not recommended, off-state voltages up to 800V may be applied without damage, but the triac may switch to the on-state. The rate of rise of current should not exceed 15 A/µs.

Annexe (A)

FAIRCHILD SEMICONDUCTOR®

September 2009

MOC3010M, MOC3011M, MOC3012M, MOC3020M, MOC3021M, MOC3022M, MOC3023M 6-Pin DIP Random-Phase Optoisolators Triac Driver Output (250/400 Volt Peak)

Features

- Excellent I_{FT} stability—IR emitting diode has low degradation
- High isolation voltage—minimum 5300 VAC RMS
- Underwriters Laboratory (UL) recognized—File #E90700
- Peak blocking voltage
  - 250V-MOC301XM
  - 400V-MOC302XM
- VDE recognized (File #94766)
  - Ordering option V (e.g. MOC3023VM)

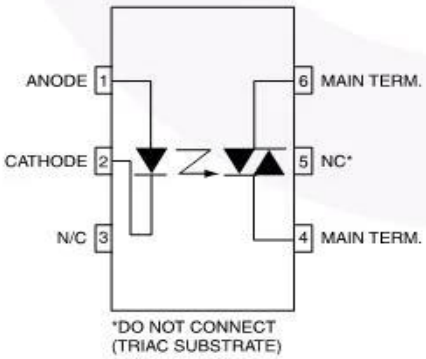
Applications

- Industrial controls
- Solenoid/valve controls
- Traffic lights
- Static AC power switch
- Vending machines
- Incandescent lamp dimmers
- Solid state relay
- Motor control
- Lamp ballasts

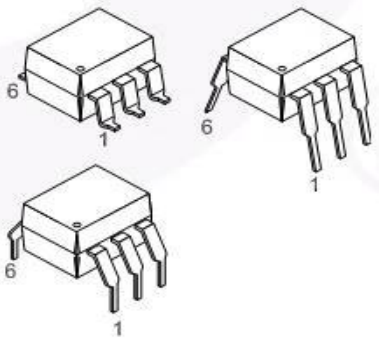
Description

The MOC301XM and MOC302XM series are optically isolated triac driver devices. These devices contain a GaAs infrared emitting diode and a light activated silicon bilateral switch, which functions like a triac. They are designed for interfacing between electronic controls and power triacs to control resistive and inductive loads for 115 VAC operations.

Schematic



Package Outlines



Annexe (A)

Principe de la mesure par ultrason