الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة

République Algérienne Démocratique et Populaire

وزارة التعليـم العالـي والبحـث العلمـي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

---

UNIVERSITE YAHIA FARES DE MEDEA

Faculté de Technologie

Département de Génie Electrique

Mémoire de fin d'études de Master

Filière : Electronique

Spécialité : Électronique des systèmes embarqués

Commande PID Optimisée par l'Algorithme Métaheuristique Bald Eagle Search pour le Contrôle du Niveau de Liquide dans un Réservoir

Réalisé par :

M. IDDER Yahya
M. BELFACI Samir

Proposé et Dirigé par :

Pr. BOUHEDDA Mounir

Dr. ANIBA Yahia

M. HAMRANI Abdelmoumin

Promotion 2024/2025

Remerciements

À l'issue de notre parcours de recherche, nous exprimons notre profonde gratitude à Dieu Tout-Puissant, qui nous a accordé la force et l'aide nécessaires pour mener à bien ce travail.

Nous adressons nos remerciements les plus sincères à :

Monsieur Pr. BOUHEDDA Mounir et Dr. ANIBA Yahia et Mr. HAMRANI Abdelmoumin pour leurs conseils précieux et le suivi qu'ils nous ont assurés tout au long de la réalisation de ce travail.

.

Nous apportons aussi nos vifs remerciements aux personnels enseignants de l'Université Yahia Farés de Médéa.

Nous remercions chaleureusement les membres du jury pour leur regard critique et plus que pertinent sur l'ensemble de notre travail.

À l'ensemble de la promotion 2024/2025

Dédicace

À nos chères mères,

À nos chers parents,

Ceux qui ne se sont jamais arrêtés, ont prié pour nous et nous ont soutenus afin que nous puissions atteindre nos objectifs.

À nos frères,

Pour leur soutien moral et matériel ainsi que leurs précieux conseils,

À notre famille,

À tous nos amis,

À tous ceux qui m'ont aidé de près ou de loin,

Dans l'accomplissement de ce modeste travail,

À tout le monde sans exception.

تنظيم PID محسَّن باستخدام الخوارزمية الميتاهيوريستية Bald Eagle Search للتحكم في مستوى السائل داخل خزان

ملخص:

تهدف هذه المذكرة إلى تصميم ومحاكاة نظام للتحكم في مستوى السائل داخل خزان، من خلال دمج تقنيتين مختلفتين في التحكم :التحكم التبدلي (TOR) والتحكم التناسبي التكاملي التفاضلي (PID) ، مع تحسين هذا الأخير باستخدام خوارزمية ذكية تدعى Bald Eagle Search (BES) . يستخدم النظام لوحة Arduino UNO لقيادة المضخة والصمام، بالإضافة إلى حساس فوق صوتي HC-SR04 لقياس مستوى السائل.

تم تنفيذ النظام ضمن بيئتين للمحاكاة:

- Proteus لمحاكاة التحكم الثنائي(TOR) ؛
- MATLAB/Simulink لمحاكاة وتنظيم النظام بواسطة PID.

وللحصول على أفضل أداء ممكن، تم توليف معاملات PID (Kp, Ki, Kd) باستخدام خوارزمية BES ، التي أظهرت كفاءتها في تقليل التجاوز، تسريع الاستجابة، وتقليل الخطأ في الحالة المستقرة.

أثبتت نتائج المحاكاة فعالية النظام المقترح، سواء من حيث البنية التقنية أو الأداء الديناميكي. كما أظهرت أهمية الدمج بين التحكم الكلاسيكي وخوارزميات الذكاء الاصطناعي في تطوير أنظمة أكثر دقة وموثوقية، مما يفتح آفاقًا واسعة للتطبيق في مجالات الزراعة، معالجة المياه، والصناعات المختلفة.

الكلمات المفتاحية: PID، أردوينو ، بروتيوس ، ماتلاب , TOR , خوارزمية BES ,مستوى السائل داخل خزان .

PID Control Optimized by the Bald Eagle Search Metaheuristic Algorithm for Liquid Level Control in a Tank

Abstract:

This thesis aims to design and simulate a liquid level control system in a tank by combining two different control strategies: On/Off control (AON) and Proportional–Integral–Derivative (PID) control, with the latter being optimized using an intelligent algorithm called Bald Eagle Search (BES). The system is based on an Arduino UNO board to drive a pump and a valve, and uses an HC-SR04 ultrasonic **sensor** to measure the liquid level accurately.

The system was implemented in two simulation environments:

- Proteus for AON control;
- MATLAB/Simulink for PID regulation.

PID parameters (Kp, Ki, Kd) were automatically tuned using the BES algorithm, which significantly improved the system's dynamic performance by reducing overshoot, response time, and steady-state error.

Simulation results validated the efficiency of the proposed approach in both technical and performance aspects. This work highlights the importance of integrating classical control with intelligent algorithms to develop more precise, stable, and adaptive systems, applicable to fields such as agriculture, water treatment, and various industrial processes.

Keywords: PID **,** Arduino , Proteus **,** MATLAB **,** AON **,** the BES algorithm, Liquid level in a tank

Commande PID Optimisée par l'Algorithme Métaheuristique Bald Eagle Search pour le Contrôle du Niveau de Liquide dans un Réservoir

Résumé :

Ce mémoire a pour objectif la conception et la simulation d'un système de régulation du niveau de liquide dans un réservoir, en combinant deux techniques de commande différentes : la commande Tout ou Rien (TOR) et la commande PID (Proportionnelle–Intégrale–Dérivée), cette dernière étant optimisée à l'aide d'un algorithme intelligent appelé Bald Eagle Search (BES). Le système repose sur une carte Arduino UNO pour piloter la pompe et la vanne, ainsi qu'un capteur à ultrasons HC-SR04 pour mesurer avec précision le niveau du liquide.

La mise en œuvre a été réalisée dans deux environnements de simulation :

- Proteus pour la commande TOR .
- MATLAB/Simulink pour la régulation PID.

Les paramètres PID (Kp, Ki, Kd) ont été ajustés automatiquement grâce à l'algorithme BES, permettant d'améliorer significativement la réponse dynamique du système en réduisant le dépassement, le temps de réponse et l'erreur en régime permanent.

Les résultats de simulation ont confirmé l'efficacité du système proposé, aussi bien sur le plan technique que sur le plan de la performance. Ce travail met en lumière l'importance de l'intégration entre le contrôle classique et les algorithmes d'intelligence artificielle pour développer des systèmes plus précis, stables et adaptables, applicables notamment dans les domaines de l'agriculture, du traitement de l'eau et des industries diverses.

Mots clés: PID , Arduino , Proteus **,** MATLAB **,** TOR **,** l'algorithme BES **,** niveau de liquide dans un réservoir

Table des matières

Introduction générale 2

Chapitre I | Théories de base des régulations 4

I.1 Introduction 5

I.2 Régulation Tout ou Rien (TOR) 5

I.2.1 Domaine d'utilisation 6

I.2.2 Principe de fonctionnement 6

I.2.3 Avantages 7

I.2.4 Inconvénients 7

I.2.5 Les différents types de régulateur tout ou rien 7

I.3 Régulation PID 8

I.4 Application de la régulation PID.....9

I.5 Les actions élémentaires (proportionnelle, intégrale, dérivée) 10

I.5.1 Action proportionnelle 10

I.5.2 Action intégrale (I).....11

I.5.3 Action dérivée 12

I.6 Type de régulateurs 13

I.6.1 Régulateur à action proportionnelle (P) 13

I.6.2 Régulateurs à action proportionnelle-intégrale (PI)..... 14

I.6.3 Régulateur à action proportionnelle-dérivée (PD) 15

I.6.4 Régulateur PID 16

I.6.5 Régulateur PID numérique (PID digital) 17

I.6.6 Régulateur PID adaptatif 17

I.7 Méthodes de réglage des paramètres PID 17

I.7.1 Méthode de Ziegler–Nichols 17

I.7.2 Méthode de l'essai-erreur (Trial and Error) 19
I.7.3 Méthode analytique (modèle mathématique) 19
I.8 Effet des gains PID sur la réponse temporelle d'un système 20
I.9 Conclusion 22
Chapitre II | Algorithme Bald Eagle Search (BES) : Concepts et Fondements Théoriques 23
II.1 Introduction 24
II.2 L'algorithme d'optimisation Bald Eagle Search (BES)..... 24
II.3 conclusion 30
Chapitre III | Simulation et interprétation des résultats 31
III.1 Introduction 32
III.2 Outils Hardware 32
III.2.1 Arduino UNO 32
III.2.2 Capteur ultrason (HC-SR04) 35
III.2.3 L'écran d'affichage LCD 2x16..... 35
III.2.4 Pompe à eau 37
III.3 Les logiciels (Software) 38
III.3.1 IDE Arduino..... 38
III.3.2 Logiciel PROTEUS 39
III.4 La simulation de système TOR 41
III.5 Matlab Simulink 45
III.5.1 Composants principaux de Simulink 46
III.6 Description du système à commander 47
III.6.1 Modélisation du système……..... 48
III.6.2 Discussion des resultats 51
III.7 Conclusion 60
Conclusion générale 62

Liste des figures

Chapitre I

Figure I. 1 : Régulation Tout ou Rien.[2] 6
Figure I. 2 : schéma typique de contrôle PID.[5] 9
Figure I. 3: Schéma fonctionnel du régulateur P.[8] 14
Figure I. 4: Réponse indicielle du régulateur P.[8] 14
Figure I. 5: Schéma fonctionnel du régulateur PI.[8] 15
Figure I. 6: Réponse indicielle du régulateur PI.[8] 15
Figure I. 7: Schéma fonctionnel du régulateur PD.[8] 15
Figure I. 8: Réponse indicielle du régulateur PD.[8] 16
Figure I. 9 : Réglage d'un régulateur PID par rapport à PD et P.[9] 16

Chapitre II

Figure II. 1 : Étapes de chasse de l'aigle à tête blanche.[16] 26
Figure II. 2 : Organigramme de l'algorithme BES.[16] 28

Chapitre III

Figure III. 1: Constitution de la carte Arduino UNO.[18]. 33
Figure III. 2: Capteur Ultrason HC-SR04.[21] 35
Figure III. 3: L'afficheur LCD 2x16.[22]..... 36
Figure III. 4: Pompe à Eau 5V. [24] 38
Figure III. 5: formule d'Arduino IDE.[27]. 39
Figure III. 6: logiciel PROTEUS.[29]. 40
Figure III. 7: Architecture générale du système TOR. 40
Figure III. 8: Organigramme de système TOR. 41
Figure III. 9: Importer la bibliothèque du capteur. 42
Figure III. 10: Récupérer le code Arduino. 43
Figure III. 11: Premier cas de simulation 44

Figure III. 12: Second cas de simulation 45
Figure III. 13: Interface de matlab Simulink.[31]. 46
Figure III. 14: Représentation du système de régulation de niveau de liquide. 48
Figure III. 15: Schéma bloc de modélisation du niveau h(t)..... 50
Figure III. 16: Schéma bloc global du système avec régulateur PID. 50
Figure III. 17 : Processus d'optimisation des paramètres PID via l'algorithme BES. 50
Figure III. 18 : Réponse temporelle du système sans optimisation. 52
Figure III. 19 : Réponse temporelle du système avec optimisation par BES 53
Figure III. 20: Comparaison des réponses : méthode empirique vs BES. 53
Figure III. 21 : Effet de l'augmentation du gain proportionnel Kp. 55
Figure III. 22 : Effet de l'augmentation du gain intégral Ki. 56
Figure III. 23 : Effet de l'augmentation du gain dérivé Kd. 57
Figure III. 24 : Asservissement global du système avec gains optimisés. 58
Figure III. 25: Réponse du système à une perturbation : variation de la section de l'orifice a.. 60

Liste des tableaux

Tableau I. 1 : Les formules de Ziegler-Nichols.[11] 18
Tableau I. 2: effets des gains PID.[14] 20
Tableau I. 3 : Comparatif régulation (PID) vs Tout ou Rien (TOR).[15] 21
Tableau III. 1: Caractéristiques techniques de l'Arduino UNO [19]. 34
Tableau III. 2: caractéristiques de capteur Ultrason HC-SR04.[20] 35
Tableau III. 3: Les pins de LCD 2x16.[23] 37
Tableau III. 4: Différence entre MATLAB et Simulink.[32] 47
Tableau III. 5: les paramètres, le nombre d'itérations T et la taille de la population N de l'algorithme BES. 51
Tableau III. 6: Gains du PID et ITAE: Essais & erreurs vs BES. 51
Tableau III. 7: Performance dynamique selon la méthode de réglage des gains PID. 54

Liste des abréviations

Abréviation	Signification
API	Interface de Programmation d'Application *(Application Programming Interface)*
D	Terme dérivé (dans PID)
D0 – D7	Broches de données pour l'afficheur LCD
GND	Masse (Ground)
GSM	Système mondial de communications mobiles *(Global System for Mobile Communications)*
HC-SR04	Capteur à ultrasons utilisé pour la mesure de distance
I	Terme intégral (dans PID)
ICSP	Programmation série en circuit *(In-Circuit Serial Programming)*
IDE	Environnement de Développement Intégré *(Integrated Development Environment)*
IoT	Internet des Objets *(Internet of Things)*
Kp, Ki, Kd	Gains proportionnel, intégral et dérivé du contrôleur PID
LCD	Afficheur à cristaux liquides *(Liquid Crystal Display)*
MCU	Unité de microcontrôle *(Microcontroller Unit)*
MIMD	Instructions Multiples, Données Multiples *(Multiple Instruction, Multiple Data)*
MQTT	Protocole de Télétransmission avec Mise en File d'Attente *(Message Queuing Telemetry Transport)*
P	Terme proportionnel (dans PID)
PCB	Carte de circuit imprimé *(Printed Circuit Board)*
PID	Régulateur Proportionnel–Intégral–Dérivé
PLC	Automate programmable *(Programmable Logic Controller)*
PWM	Modulation de largeur d'impulsion *(Pulse Width Modulation)*
R/W	Lecture / Écriture *(Read/Write)*

RS232	Standard de communication série recommandé
SRAM	Mémoire statique à accès aléatoire *(Static Random-Access Memory)*
TOR	Tout Ou Rien (Commande binaire)
TCP	Protocole de Contrôle de Transmission *(Transmission Control Protocol)*
Vss, Vdd, Vo, RS, E	Pins de l'écran LCD (Alimentation, Masse, Contraste, etc.)

Liste des symboles

Symbole	Signification
a	Surface de l'orifice de sortie du réservoir [m²]
Bp	Bande proportionnelle
C(p)	Fonction de transfert
E(p)	Erreur dans le domaine de Laplace
e(t)	Erreur instantanée entre consigne et mesure
Ɛ(t)	Écart entre la consigne et la mesure
Gc(p)	Fonction de transfert du contrôleur
h(t)	Hauteur du liquide dans le réservoir au temps t [m]
p (Laplace)	Variable dans le domaine de Laplace
qout(t)	Débit de sortie du réservoir [m³/s]
Qpompe	Débit de la pompe [m³/s]
Td	Constante de temps dérivée
Ti	Constante de temps intégrale
U(p)	Signal de commande dans le domaine de Laplace
u(t)	Signal de commande dans le domaine temporel
u_0	Valeur initiale du signal de commande

Introduction générale

Introduction générale

Dans un contexte de progrès technologique rapide et de besoin croissant en systèmes de contrôle automatisés, les systèmes de régulation intelligents occupent une place centrale dans de nombreuses applications industrielles et domestiques, notamment dans le domaine du contrôle du niveau des liquides dans les réservoirs. Le maintien précis du niveau de liquide est essentiel dans plusieurs secteurs clés tels que l'agriculture, le traitement de l'eau, les industries chimiques et les systèmes de refroidissement .

Ce projet a pour objectif de concevoir et de simuler un système de régulation du niveau de liquide dans un réservoir, en combinant deux techniques de régulation distinctes : la commande tout ou rien (TOR – Tout ou Rien), et la commande PID (Proportionnelle–Intégrale–Dérivée). Le système s'appuie sur une carte Arduino pour contrôler une pompe et une vanne, et utilise un capteur ultrasonique HC-SR04 pour mesurer avec précision le niveau du liquide.

La mise en œuvre du système a été réalisée dans deux environnements de simulation différents selon la nature de chaque régulation :

1) Le logiciel Proteus a été utilisé pour simuler la régulation TOR, en raison de sa simplicité et de son adéquation avec les systèmes de type marche/arrêt.
2) L'environnement MATLAB/Simulink a été choisi pour modéliser et analyser la régulation PID, grâce à ses capacités avancées en simulation dynamique.

L'objectif principal est d'améliorer la réponse du système en réduisant les oscillations, le temps de stabilisation et l'erreur en régime permanent. Pour atteindre cet objectif, les paramètres du régulateur PID (Kp, Ki, Kd) ont été optimisés à l'aide de l'algorithme méta heuristique Bald Eagle Search (BES), inspiré du comportement de chasse de l'aigle chauve, permettant ainsi de trouver un compromis optimal entre rapidité, stabilité et précision.

Ce mémoire est structuré en trois chapitres. Le premier chapitre expose les bases théoriques des régulations TOR et PID. Le deuxième chapitre est consacré à la présentation de l'algorithme BES et de son fonctionnement. Le troisième chapitre est dédié à la simulation, à la mise en œuvre pratique du système et à l'analyse des résultats. À travers cette approche intégrée, cette étude vise à démontrer l'intérêt de l'utilisation des algorithmes d'intelligence artificielle dans l'optimisation des systèmes de régulation classiques, ouvrant ainsi la voie à des applications industrielles plus performantes et flexibles.

Chapitre I

Théories de base des régulations

I.1 Introduction

Parmi les techniques utilisées pour atteindre cet objectif, on distingue principalement deux approches : la régulation tout ou rien (TOR), basée sur le principe de mise en marche et arrêt, et la régulation proportionnelle-intégrale-dérivée (PID), qui repose sur le calcul de l'erreur et son ajustement progressif pour assurer une meilleure stabilité.

Dans ce chapitre, nous présentons les fondements théoriques de ces deux méthodes de régulation et mettons en lumière les caractéristiques propres à chacune, en termes de structure, de réponse et de précision. Une comparaison analytique entre TOR et PID est également proposée, en examinant les avantages, les limitations et leur adéquation à différents scénarios d'utilisation.

I.2 Régulation Tout ou Rien (TOR)

Un régulateur tout ou rien est un dispositif de commande simple mais efficace, largement utilisé dans les systèmes de régulation où la complexité et la précision ne sont pas primordiales. Ce type de régulateur produit un signal de sortie discontinu, qui ne peut prendre que deux états distincts : soit l'état bas (0) correspondant à l'arrêt du système, soit l'état haut (1) représentant la mise en marche du système. Cette approche binaire, souvent qualifiée de « marche/arrêt » ou On-Off, ne permet aucune modulation intermédiaire de la sortie, contrairement aux régulateurs analogiques ou proportionnels.

Le régulateur tout ou rien est également connu sous les appellations de régulateur à deux positions (Two-step controller) ou commande à deux états, car il alterne uniquement entre ces deux niveaux de commande en fonction de l'écart entre la valeur mesurée (grandeur réglée) et la consigne (valeur souhaitée).

Cette simplicité de fonctionnement en fait une solution idéale dans de nombreuses applications pratiques, notamment lorsque la précision de la régulation est moins critique, mais où la robustesse, la fiabilité et le faible coût sont des priorités, il est couramment intégré dans des systèmes domestiques (chauffe-eau, thermostats, compresseurs d'air), ainsi que dans certaines installations industrielles, notamment pour des procédés thermiques ou hydrauliques à forte inertie.

Cependant, cette méthode présente certaines limites, notamment la possibilité d'oscillations autour de la consigne, ce qui peut provoquer un cycle fréquent de mise en marche et d'arrêt (appelé cycle de basculement ou "cycling") si aucun dispositif d'hystérésis n'est ajouté. Pour cette raison, bien que simple, le régulateur tout ou rien doit être soigneusement dimensionné et adapté au système dans lequel il est intégré.[1]

La figure I.1 donne en exemple de Régulation Tout ou Rien :

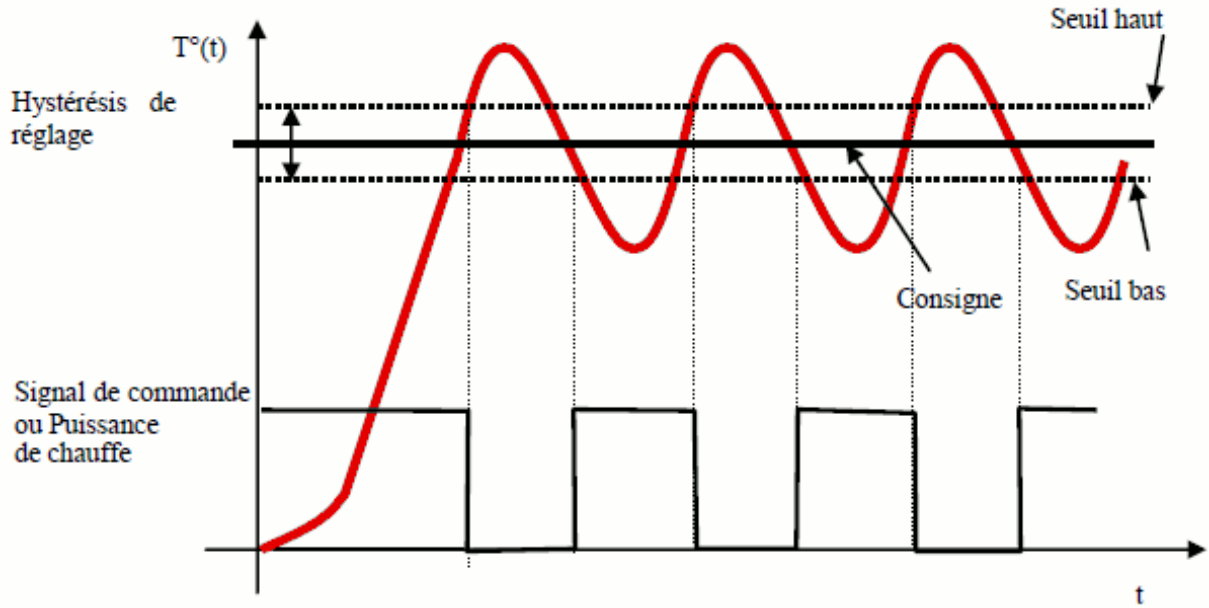


Figure I. 1 : Régulation Tout ou Rien.[2]

I.2.1 Domaine d'utilisation

Les régulateurs tout ou rien sont principalement utilisés dans les systèmes où la précision n'est pas primordiale et où l'inertie du système est élevée. Un exemple typique est la régulation du niveau d'eau dans un réservoir, où de petites variations ne posent pas de problème majeur étant donné la simplicité du système et la faible exigence de précision.

I.2.2 Principe de fonctionnement

Dans ce type de régulation, on utilise généralement un relais électromécanique comme dispositif de commutation. Lorsque le signal de commande est à l'état 1, une bobine est excitée, fermant le contact du relais et alimentant, par exemple, une résistance chauffante. Lorsque le signal passe à l'état 0, la bobine est désactivée et le circuit s'ouvre, arrêtant ainsi le fonctionnement. Ce type de régulation est couramment rencontré dans les appareils

domestiques comme les thermostats, ou encore dans les dispositifs de sécurité tels que les pressostats.

I.2.3 Avantages

- Simple et facile à mettre en œuvre.
- Ne nécessite pas de calculs complexes.
- Coût faible.
- Entretien facile.
- Adapté aux systèmes à forte inertie.

I.2.4 Inconvénients

- Ne permet pas une régulation de haute précision.
- Peut entraîner des oscillations dans le système.
- Le fonctionnement répétitif peut provoquer une usure mécanique des composants comme les relais.
- Inadapté aux systèmes rapides ou sensibles aux variations.[3]

I.2.5 Les différents types de régulateur tout ou rien

I.2.5.1 Régulateur tout ou rien sans hystérésis (simple)

- Le plus basique.
- Le changement d'état (marche/arrêt) se produit exactement lorsque la valeur mesurée atteint la consigne.
- Risque élevé d'oscillations rapides (mise en marche et arrêt fréquents).

Exemple : Un thermostat qui allume un chauffage dès que la température descend en dessous de 20°C et l'éteint immédiatement lorsqu'elle atteint 20°C.

I.2.5.2 Régulateur tout ou rien avec hystérésis

- Introduit une zone de tolérance (hystérésis) autour de la consigne.
- Le régulateur ne change d'état que si la valeur dépasse un certain seuil au-dessus ou en dessous de la consigne.
- Réduit le cyclage et prolonge la durée de vie des composants.

Exemple : Chauffage qui s'allume à 19°C et ne s'éteint qu'à 21°C.

I.2.5.3 Régulateur à cycle fixe (ou à modulation de durée d'impulsion - PWM)

- La sortie est activée/désactivée de manière cyclique avec une durée fixe du cycle, mais la durée de mise en marche (duty cycle) varie selon l'erreur.
- Permet un contrôle plus doux que le TOR classique.
- Transition vers un comportement proche de la régulation proportionnelle.

I.2.5.4 Régulateur TOR différentiel (à deux seuils)

- Utilise deux consignes : une de mise en marche et une d'arrêt.
- Plus stable que le simple TOR.
- Similaire au régulateur avec hystérésis, mais les seuils sont définis explicitement.[3]

I.3 Régulation PID

L'algorithme de contrôle PID (contrôle proportionnel-intégral-dérivé) est devenu l'une des technologies les plus utilisées dans le contrôle industriel, en particulier dans les scénarios qui nécessitent une précision et une stabilité élevées. Le contrôleur PID ajuste dynamiquement le système à l'aide de trois paramètres : proportionnel (P), intégral (I) et dérivé (D), pour garantir la correspondance précise des variables de processus (telles que la température, la pression, la vitesse, etc.) avec les valeurs cibles définies. Il ajuste dynamiquement l'entrée de contrôle en fonction de l'erreur afin d'éliminer rapidement et précisément l'erreur, stabilisant ainsi finalement la sortie du système à proximité de la cible souhaitée.

Le principe de fonctionnement du contrôle PID est basé sur un mécanisme de rétroaction en boucle fermée : en surveillant en permanence l'état du système, le contrôleur compare la valeur mesurée réelle avec le point de consigne, calcule l'erreur et ajuste la sortie du système à l'aide d'actions de contrôle proportionnelles, intégrales et dérivées. Il envoie le signal de réglage à l'actionneur (par exemple, un régulateur de température ou un pilote de moteur) pour ajuster l'état de l'équipement et rapprocher la variable de processus du point de consigne. Ce contrôle de rétroaction garantit que l'environnement de production pour la fabrication de cellules de charge reste précis, préservant la cohérence du produit et une qualité élevée [4].

La figure I.2 donne schéma typique de contrôle PID :

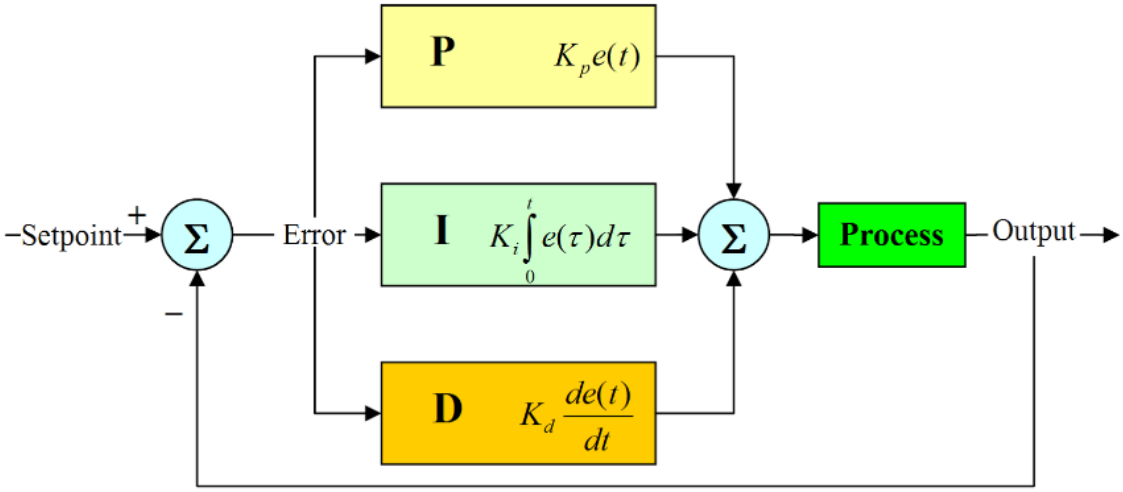


Figure I. 2 : schéma typique de contrôle PID.[5]

I.4 Application de la régulation PID

Applicable à diverses industries :

❖ Laboratoire :

Le contrôle PID est largement utilisé pour réguler avec précision des paramètres tels que la température, la pression et le débit, garantissant un fonctionnement stable des équipements de production, améliorant ainsi la qualité du produit et l'efficacité de la production.

❖ L'industrie chimique :

Dans les réactions chimiques, le contrôle PID ajuste la température, la pression et le débit pour empêcher les réactions de progresser trop rapidement ou de devenir instables, garantissant ainsi la sécurité et la cohérence de la qualité du produit.

❖ Contrôle de l'automatisation :

Les algorithmes PID sont utilisés dans les robots industriels et les lignes de production automatisées, garantissant que l'équipement accomplit les tâches à la vitesse et avec la précision souhaitée, améliorant ainsi à la fois l'efficacité et la précision de la production.

❖ Industrie aérospatiale :

Dans les systèmes de contrôle de vol, les algorithmes PID régulent l'attitude, la vitesse et l'altitude de l'avion, garantissant ainsi la stabilité et la sécurité pendant le vol.

❖ Puissance de l'industrie :

Le contrôle PID est utilisé dans les systèmes électriques pour l'équilibrage de la charge, le contrôle de la tension et la stabilité de la fréquence, garantissant un fonctionnement efficace et stable des systèmes électriques.

❖ Industrie automobile :

Dans les systèmes d'alimentation automobile, les systèmes de suspension et d'autres domaines, les contrôleurs PID garantissent des expériences de conduite fluides tout en optimisant l'efficacité énergétique et le contrôle des émissions[4].

I.5 Les actions élémentaires (proportionnelle, intégrale, dérivée)

I.5.1 Action proportionnelle

- Equation temporelle[6]:

$$u(t) = k_p.\varepsilon(t) + u_0 \quad \textbf{(I.1)}$$

K_p: Gain ou amplification. Cette constante est exprimée sans unité.

u_0: Lorsque l'écart ε(t) est nul, u0 permet de commander l'actionneur avec une v c une valeur non nulle.

La fonction de transfert se déduit de l'équation temporelle .En effet, pour une variation ε(t) on a une variation

$$u(t) = k_p.\varepsilon(t) \quad \textbf{(I.2)}$$

- La fonction de transfert est :

$$C(p) = \frac{u(p)}{\varepsilon(p)} = k_p = \frac{1}{Bp} \quad \textbf{(I.3)}$$

I.5.1.1 Rôle et domaine d'utilisation de l'action proportionnelle:

Le rôle de l'action P est de réduire l'erreur de réglage qui est inversement proportionnelle au gain, mais rend la réponse plus au moins oscillatoire. On choisit un gain qui permet d'avoir un bon taux d'amortissement. On utilise un régulateur P lorsque la précision n'est pas importante. Par exemple, le réglage du niveau d'eau dans un réservoir de stockage. L'action P est souvent suffisante pour régler plusieurs systèmes dans l'industrie. Elle est simple à réaliser (simple amplificateur) d'où son grand avantage[7].

I.5.2 Action intégrale (I)

- Equation temporelle[6]:

$$u(t) = \int_0^t \varepsilon(t)d(t) + u_0 = ki \int_0^t \varepsilon(t)dt + u_0 \qquad \textbf{(I.4)}$$

- T_i: constante de temps de l'action intégrale. Cette constante est souvent exprimée en minutes.
- K_i: coefficient d'action intégrale ou taux d'action intégrale. Ki est exprimé en min^{-1}
- u_0: valeur initiale de u. Cette valeur n'est pas réglable directement; c'est une mise en mémoire de l'évolution de u(t) avant T=0.

- Fonction de transfert:

$$C(p) = \frac{u(p)}{\varepsilon(p)} = \frac{1}{T_i p} = \frac{k_i}{p} \qquad \textbf{(I.5)}$$

I.5.2.1 Rôle et domaine d'utilisation de l'action intégrale:

Dans les régulateurs industriels on affiche 1/T i, alors T i est d'autant plus grand que l'action intégrale est faible. Le rôle principal de l'action intégrale est d'éliminer l'erreur statique. Toute fois l'action intégrale est un élément à retard de phase, donc l'augmentation de l'action intégrale (diminué Ti) produit une instabilité car elle déplace le lieu de Nyquist vers la gauche. La valeur optimale est choisie pour satisfaire un compromis stabilité-rapidité. Si le système possède lui-même un intégrateur (exemple niveau), l'action I est quand même nécessaire pour annuler l'écart de perturbation car, suite aux variations de la consigne l'intérêt de l'action I est moindre car l'écart s'annule naturellement. Dans l'industrie, on utilisera

l'action I chaque fois que nous avons besoin, pour des raisons technologiques, d'avoir une précision parfaite, exemple: la régulation de la pression ou température dans un réacteur nucléaire. De plus, il faut souligner que l'action I est un filtre donc il est intéressant de l'utiliser pour le réglage des paramètres très dynamiques tels que la pression [7]. On peut conclure que l'action intégrale:

- ajoute dans la chaîne de commande une intégration afin d'annuler l'erreur statique en régime permanent.
- améliore la précision.
- mais introduit un déphasage de -90° et risque de rendre le système instable.

I.5.3 Action dérivée

- Equation temporelle[6]:

$$u(t) = T_d \frac{d\varepsilon(t)}{dt} \qquad \textbf{(I.6)}$$

- T_d: constante de temps de l'action dérivée. Cette constante est, le plus souvent, exprimée en secondes.

- Fonction de transfert:

$$C(p) = \frac{u(p)}{\varepsilon(p)} = T_d p \qquad \textbf{(I.7)}$$

I.5.3.1 Rôle et domaine d'utilisation de l'action dérivée :

L'action dérivée compense les effets du temps mort du processus. Elle a un effet stabilisateur mais une valeur excessive peut entraîner une instabilité. Sur le plan de Nyquist l'action D permet de déplacer le lieu de transfert vers la droite car elle possède une avance de phase (de +90 degré). La présence de l'action dérivée permet donc d'augmenter la rapidité du système en augmentant le gain sans être inquiété par la stabilité. Dans l'industrie, l'action D n'est jamais utilisée seule mais en général avec l'action intégrale. On recommande de l'utiliser pour le réglage des paramètres lents tels que la température. Par contre en présence des paramètres bruités, l'action dérivée est déconseillée[7] .

Donc :

P → L'action P s'apparente à la vitesse et à la rapidité. Elle répond de manière instantanée à l'écart constaté sur la grandeur à régler; par son gain, elle permet de vaincre les inerties du système. Le régulateur P est appréciable lorsque la précision n'est pas essentielle. Afin de diminuer l'écart de réglage et rendre le système plus rapide, on augmente le gain (on diminue la bande proportionnelle) mais, on est limité par la stabilité du système. Le régulateur P est utilisé lorsqu'on désire régler un paramètre dont la précision n'est pas importante, exemple : régler le niveau dans un bac de stockage.

I → L'action I s'apparente à la position et à la précision Elle complète et améliore la précision de l'action P en atténuant l'écart de statisme. Elle permet d'éliminer l'erreur résiduelle en régime permanent. Afin de rendre le système plus dynamique (diminuer le temps de réponse), on diminue l'action intégrale mais, ceci provoque l'augmentation du déphasage ce qui provoque l'instabilité en état fermé. L'action intégrale est utilisée lorsqu'on désire avoir en régime permanent, une précision parfaite, en outre, elle permet de filtrer la variable à régler d'où l'utilité pour le réglage des variables bruitées telles que la pression.

D → L'action D s'apparente à l'accélération de la réponse et à la stabilité. Pour le réglage des variables lentes, elle compense les temps morts et les inerties en permettant notamment un amortissement rapide des oscillations dues à l'apparition d'une perturbation ou à une variation subite de la consigne. Dans la pratique, l'action dérivée est appliquée aux variations de la grandeur à régler seule et non de l'écart mesure-consigne afin d'éviter les à-coups dus à une variation subite de la consigne. L'action D est utilisée dans l'industrie pour le réglage des variables lentes telles que la température, elle n'est pas recommandée pour le réglage d'une variable bruitée ou trop dynamique (la pression). En dérivant un bruit, son amplitude risque de devenir plus importante que celle du signal utile[8]

I.6 Type de régulateurs

I.6.1 Régulateur à action proportionnelle (P)

Le régulateur à action proportionnelle, ou régulateur P est une action simple et puisqu'il construit une commande u(t) proportionnelle à l'erreur e(t).

Cette action s'apparente à un effet de ressort (ressort de rappel) [8].

- Loi de commande du régulateur P:

$$u(t) = k_p.e(t) \quad \textbf{(I.8)}$$

- Fonction de transfert du régulateur P:

$$G_c(p) = \frac{U(P)}{E(P)} = k_p \quad \textbf{(I.9)}$$

- Schéma fonctionnel du régulateur P:

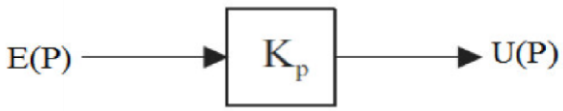


Figure I. 3: Schéma fonctionnel du régulateur P.[8]

- Réponse indicielle du régulateur P:

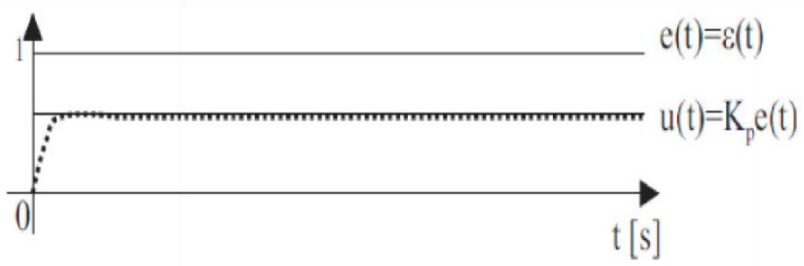


Figure I. 4: Réponse indicielle du régulateur P.[8]

I.6.2 Régulateurs à action proportionnelle-intégrale (PI)

- Fonction de transfert du régulateur PI:

$$C(p) = k_p.\frac{1+T_i p}{T_i p} \quad \textbf{(I.10)}$$

- Equation temporelle:

$$u(t) = k_p(\varepsilon(t) + \frac{1}{T_i}\int_0^t \varepsilon(t)dt \quad \textbf{(I.11)}$$

- Schéma fonctionnel du régulateur PI:

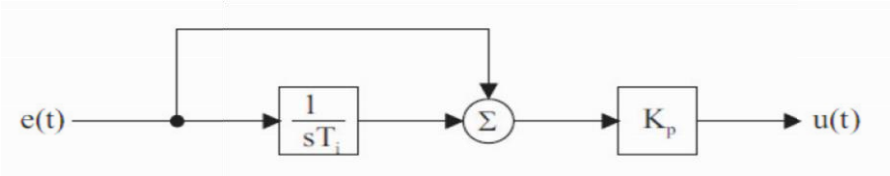


Figure I. 5: Schéma fonctionnel du régulateur PI.[8]

- Réponse indicielle du régulateur PI:

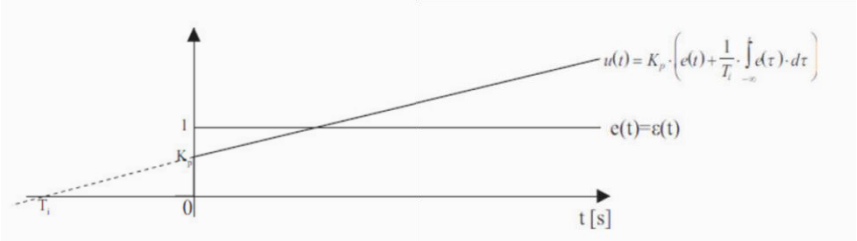


Figure I. 6: Réponse indicielle du régulateur PI.[8]

I.6.3 Régulateur à action proportionnelle-dérivée (PD)

- Fonction de transfert du régulateur PD:

$$c(p) = \frac{U(P)}{E(P)} = k_p(1 + T_d p) \qquad \textbf{(I.12)}$$

- Equation temporelle :

$$u(t) = k_p(e(t) + T_d.\frac{de(t)}{dt} \qquad \textbf{(I.13)}$$

- Schéma fonctionnel du régulateur PD:

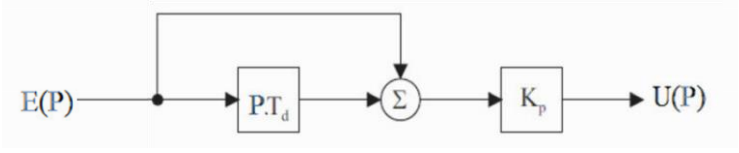


Figure I. 7: Schéma fonctionnel du régulateur PD.[8]

- Réponse indicielle du régulateur PD :

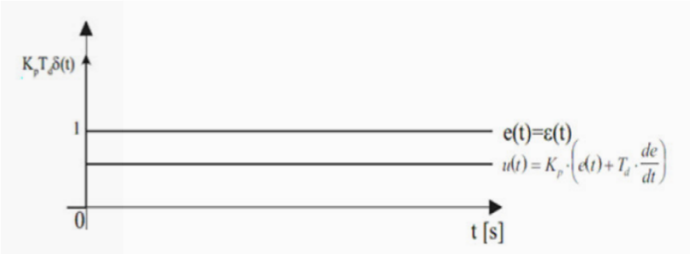


Figure I. 8: Réponse indicielle du régulateur PD.[8]

I.6.4 Régulateur PID

Le régulateur PID est le plus complet et le plus largement utilisé dans les systèmes de régulation automatique. Il combine trois actions : l'action proportionnelle (P) qui corrige l'erreur actuelle, l'action intégrale (I) qui élimine l'erreur cumulée dans le temps, et l'action dérivée (D) qui anticipe les variations futures en fonction du taux de changement de l'erreur. Grâce à cette combinaison, le régulateur PID offre une réponse rapide, stable et précise, ce qui le rend idéal pour les systèmes complexes où la performance et la stabilité sont essentielles.

Domaines d'application : contrôle de la température, de la vitesse, de la pression, du niveau de liquide dans les procédés industriels (figure I.9).

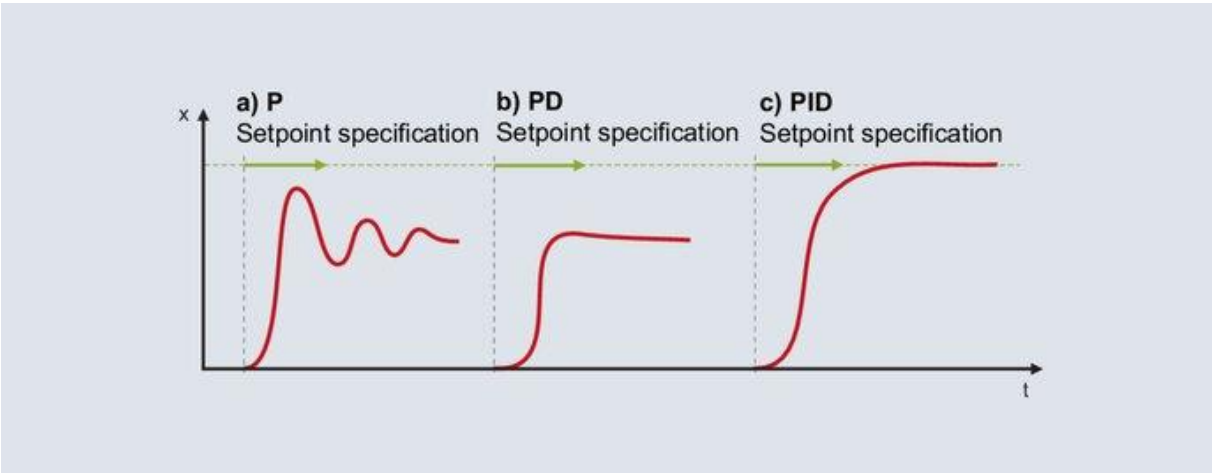


Figure I. 9 : Réglage d'un régulateur PID par rapport à PD et P.[9]

I.6.5 Régulateur PID numérique (PID digital)

Le PID numérique est une version implémentée dans des microcontrôleurs ou automates programmables (Arduino, PLC, etc.). Contrairement aux systèmes analogiques, le PID numérique utilise une logique algorithmique appliquée sur des signaux échantillonnés à intervalles réguliers. Cela permet une grande flexibilité dans le réglage des paramètres (Kp, Ki, Kd), une meilleure adaptabilité, et une facilité d'intégration dans les simulations logicielles ou les interfaces homme-machine.

Domaines d'utilisation **:** systèmes embarqués, robotique, simulations dans Proteus ou MATLAB/Simulink.

I.6.6 Régulateur PID adaptatif

Le PID adaptatif est un régulateur intelligent capable d'ajuster automatiquement ses coefficients (Kp, Ki, Kd) en fonction du comportement du système et des variations de son environnement. Il est souvent basé sur des algorithmes d'intelligence artificielle comme les réseaux neuronaux, la logique floue (fuzzy logic) ou d'autres méthodes d'apprentissage automatique. Ce type de régulation est particulièrement adapté aux systèmes non linéaires, instables ou en évolution dynamique, où les paramètres du système changent au cours du temps.

Domaines d'application **:** procédés complexes, systèmes instables, ou conditions de fonctionnement variables.

I.7 Méthodes de réglage des paramètres PID

Le réglage d'un contrôleur PID consiste à déterminer les valeurs optimales de Kp (proportionnel)**,** Ki (intégral) et Kd (dérivatif) pour obtenir une réponse stable, rapide et sans dépassement excessif**.**

I.7.1 Méthode de Ziegler–Nichols

Proposée par Ziegler et Nichols en 1942, cette méthode est l'une des plus utilisées dans l'industrie. Elle permet un réglage initial rapide sans avoir besoin d'un modèle mathématique du système.[10]

Étapes :

1. Mettre Ki = 0 et Kd = 0**.**
2. Augmenter progressivement Kp jusqu'à obtenir des oscillations soutenues (ni croissantes ni décroissantes).
3. Enregistrer :
   - K_n (*ultimate gain*).
   - T_n (période d'oscillation en secondes).
4. Appliquer les formules de Ziegler-Nichols(tableau I.1)[10] :

Tableau I. 1 : Les formules de Ziegler-Nichols.[11]

Type de régulateur	Kp	Ki	Kd
P	$0.5 \times K_n$	0	0
PI	$0.45 \times K_n$	$1.2 \times Kp / T_n$	0
PID	$0.6 \times K_n$	$2 \times Kp / T_n$	$Kp \times T_n / 8$

Exemple :
Si :

- $K_n = 6.5$
- $T_n = 4.0$ s

Alors :

- $Kp = 0.6 \times 6.5 = 3.9$
- $Ki = 2 \times 3.9 / 4 = 1.95$
- $Kd = 3.9 \times 4 / 8 = 1.95$

➢ Avantages :

- Rapide et simple à mettre en œuvre.
- Pas besoin de modélisation.

➢ Inconvénients :

- Moins précis pour les systèmes non linéaires ou très sensibles.
- Peut produire un dépassement important.[12]

I.7.2 Méthode de l'essai-erreur (Trial and Error)

Description :

Consiste à ajuster les paramètres PID manuellement en observant la réponse du système, que ce soit sur une maquette réelle ou en simulation (ex : Proteus).

Stratégie :

- Commencer par Kp faible, puis l'augmenter jusqu'à obtenir une réponse rapide et stable.
- Ajouter Ki pour éliminer l'erreur statique.
- Ajouter Kd pour atténuer les oscillations.

Indices de performance :

- Réponse lente → augmenter Kp.
- Oscillations → réduire Kp ou Ki.
- Retard important → augmenter Ki.
- Oscillations rapides → augmenter Kd.

Avantages :

- Très utilisé en simulation et en projets éducatifs.
- Ne nécessite aucune connaissance mathématique poussée.

Inconvénients :

- Chronophage.
- Résultats dépendants de l'expérience de l'opérateur.[13]

I.7.3 Méthode analytique (modèle mathématique)

Description :

Cette méthode repose sur l'obtention du **modèle mathématique** du système à réguler (équation différentielle ou fonction de transfert), ce qui permet de concevoir un PID à partir d'outils de la théorie du contrôle.

Étapes :

1. Identifier le modèle du système, par exemple

$$G(s) = \frac{K}{\tau s + 1}$$

2. Appliquer des techniques comme :
    - Placement des pôles
    - Réponse fréquentielle
    - Optimisation basée sur les performances souhaitées

➢ Avantages :

- Très précis et rigoureux.
- Permet de maîtriser précisément le comportement dynamique du système.

➢ Inconvénients :

- Nécessite des connaissances approfondies en mathématiques et automatique.
- Moins adaptée si le système est inconnu ou complexe à modéliser.

I.8 Effet des gains PID sur la réponse temporelle d'un système

Dans l'analyse des systèmes dynamiques, les performances de la réponse à une entrée en échelon sont évaluées à l'aide de trois critères principaux :

- Le dépassement (Overshoot) : l'excès de la sortie au-delà de la valeur souhaitée.
- Le temps de monté et de stabilisation (Rise & Settling Time) : durée nécessaire pour que la sortie atteigne et reste proche de sa valeur finale.

- L'erreur de régime permanent (Steady-State Error) : la différence entre la sortie finale et la valeur désirée à long terme.

Les gains PID (K_p, K_d et K_i) jouent un rôle essentiel dans l'amélioration de ces performances.

Tableau I.2 récapitule les effets des gains du correcteur PID :

Tableau I. 2: effets des gains PID.[14]

Gain	Dépassement	Temps de monté /réponse	Erreur de régime permanent
K_p	↗ Augmente le dépassement	↘ Accélère la réponse	↘ Réduit l'erreur de régime
K_d	↘ Réduit le dépassement	↘ Accélère la réponse	N'a pas d'effet
K_i	↗ Augmente le dépassement	↗ Ralentit la réponse	↘ Réduit fortement l'erreur de régime

- **Remarques :**

- K_p permet d'accélérer le système mais peut provoquer des dépassements importants.
- K_d agit comme un amortisseur, réduisant les oscillations sans affecter l'erreur finale.
- K_i agit sur l'erreur persistante, mais à un coût de stabilité et de lenteur potentielle.

Tableau I.3 récapitule comparaison entre régulation (PID) et Tout ou Rien :

Tableau I. 3 : Comparatif régulation (PID) vs Tout ou Rien (TOR).[15]

Caractéristique	**PID (Proportionnel – Intégral – Dérivé)**	**TOR (Tout ou Rien / Marche-Arrêt)**
Type de signal	Signal de commande **continu** (analogique ou PWM)	Signal **binaire** (0 ou 1 uniquement)
Précision	Élevée, grâce à la correction de l'erreur actuelle, cumulée et anticipée	Faible, avec des oscillations autour de la consigne
Réponse dynamique	Rapide et stable (si bien réglé)	Relativement lente, dépend de l'inertie du système
Stabilité	Très bonne en cas de réglage approprié	Moyenne, acceptable pour des systèmes lents
Complexité de mise en œuvre	Complexe (algorithmes, réglage des gains)	Très simple à implémenter
Coût	Plus élevé (besoin d'un microcontrôleur et de programmation)	Très faible
Usure mécanique	Faible (commande douce et progressive)	Élevée (du fait des commutations fréquentes)
Applications idéales	Processus nécessitant de la précision : température, pression, niveau, vitesse	Applications simples : chauffage, niveau de cuve avec forte inertie
Réglage / ajustabilité	Réglable dynamiquement en temps réel	Peu ou pas ajustable une fois installé

I.9 Conclusion

Ce chapitre a présenté le cadre théorique fondamental sur lequel repose ce projet, en mettant en lumière deux techniques principales de régulation appliquées aux systèmes dynamiques : la régulation tout ou rien (TOR) et la régulation proportionnelle-intégrale-dérivée (PID). Les avantages et les limites de chacune ont été clarifiés, accompagnés d'une comparaison analytique illustrant l'efficacité relative de chaque méthode en fonction de la nature du système et du niveau de complexité requis pour le contrôle.

Cependant, malgré l'efficacité reconnue du régulateur PID, ses performances dépendent fortement du réglage précis de ses trois paramètres (Kp, Ki, Kd). Le choix d'une stratégie d'optimisation adaptée pour ajuster ces paramètres représente donc une étape cruciale afin d'améliorer la réponse du système, de réduire l'erreur et de garantir sa stabilité.

Dans cette optique, le deuxième chapitre est consacré à la présentation de l'algorithme BES (Bald Eagle Search), une méthode d'intelligence artificielle inspirée du comportement naturel de l'aigle chauve, et utilisée dans ce projet pour optimiser les paramètres du PID afin d'obtenir les meilleures performances possibles en termes de régulation.

Chapitre II

Algorithme Bald Eagle Search (BES) : Concepts et Fondements Théoriques

II.1 Introduction

Dans le contexte croissant de la recherche en intelligence computationnelle et des méthodes d'optimisation bio-inspirées, de nouvelles approches se développent pour résoudre des problèmes complexes non linéaires que les techniques classiques peinent à traiter efficacement. Parmi ces approches, l'algorithme Bald Eagle Search (BES) représente une méthode innovante qui s'inspire du comportement naturel de chasse de l'aigle à tête blanche (bald eagle), un oiseau de proie symbolisant la précision, l'agilité et l'efficacité dans la nature.

L'algorithme BES se distingue par une structure simple mais puissante, articulée autour de trois phases fondamentales imitant les stratégies de l'aigle dans sa quête de nourriture : la sélection d'un espace de chasse, la recherche locale en spirale, et enfin l'attaque ciblée vers la proie. Ces phases sont traduites mathématiquement en étapes d'exploration globale, d'exploitation locale, et de raffinement de la solution, ce qui confère à l'algorithme un équilibre remarquable entre diversification et intensification deux qualités essentielles en optimisation.

Grâce à son faible nombre de paramètres, sa capacité à éviter les minima locaux, et sa flexibilité d'implémentation, le BES a suscité l'intérêt de nombreux chercheurs pour une variété d'applications, notamment en optimisation de systèmes de contrôle, en apprentissage automatique, en ingénierie et en bio-informatique. Dans ce chapitre, nous nous intéresserons en particulier à son potentiel dans l'optimisation des régulateurs PID utilisés pour la commande de systèmes dynamiques tels que la régulation du niveau d'un réservoir.

II.2 L'algorithme d'optimisation Bald Eagle Search (BES)

L'algorithme d'optimisation Bald Eagle Search (BES) est une méthode méta heuristique innovante qui s'inspire du comportement naturel de chasse de l'aigle à tête blanche (bald eagle), un oiseau de proie connu pour son efficacité et sa stratégie lors de la recherche de nourriture. Cette technique d'optimisation cherche à reproduire les différentes étapes qu'adopte cet oiseau dans son processus de chasse, depuis la localisation de zones prometteuses jusqu'à l'attaque finale sur la proie (Figure II.1).

Le BES se structure autour de trois phases fondamentales :

1. **La phase de sélection (Selection)**, durant laquelle l'aigle explore de vastes régions afin de repérer les zones potentiellement riches en proies. Cela représente l'étape d'exploration globale dans le contexte de l'optimisation, permettant à l'algorithme de ne pas se limiter à une seule région du domaine de recherche.

Phase d'initialisation :

$$P_i^j = l_b + rand \times (u_b - l_b), \quad \begin{cases} i = 1, 2, \ldots, N. \\ j = 1, 2, \ldots, dim. \end{cases} \qquad \textbf{(II.1)}$$

- P^j_i : position du j^{ème} paramètre de l'aigle i
- l_b, , u_b : bornes inférieure et supérieure de la variable
- rand : nombre aléatoire dans [0,1]

Phase de sélection de l'espace :

$$P_{i_new} = P_b + \alpha \cdot R_{rand} \cdot (P_m - P_i) \qquad \textbf{(II.2)}$$

- P_b : meilleure position actuelle
- P_m : moyenne de la population
- P_i : position de l'aigle i
- α : coefficient de mouvement
- R_{rand} : nombre aléatoire uniforme dans [0,1]

2. **La phase de recherche (Search)**, où l'aigle restreint son champ d'action autour d'une zone spécifique et y effectue une recherche en spirale. Cette étape est cruciale pour améliorer la précision de la solution en se focalisant sur une zone prometteuse, favorisant l'exploitation locale.

$$P_{i_new} = P_i + x(I).(P_i - P_m) + y(I).(P_i - P_{i+1}) \qquad \textbf{(II.3)}$$

$$x_i = xr_i / max(|\, xr \,|), \qquad y_i = yr_i / max(|yr|)$$
$$xr_i = r_i.sin(\theta_i), \qquad yr_i = r_i.cos(\theta_i)$$
$$r_i = \theta_i + R.R_{rand}, \qquad \theta_i = a.\pi.R_{rand}$$

- R ∈ [0.5, 2] , a ∈ [5,10]

3. **La phase d'attaque (Swooping)**, qui correspond à la descente rapide de l'aigle vers sa proie une fois qu'il a identifié l'endroit optimal. Cela se traduit algorithmique par un raffinement du meilleur candidat trouvé jusque-là.

$$P_{i_new} = R_{rand}P_b + x_{1i}.(P_i - c_1P_m) + y_{1i}.(P_i - c_2P_b) \quad \textbf{(II.4)}$$

$$x_{1i} = xr_i/max(|xr|), \quad y_{1i} = yr_i/max(|yr|)$$
$$xr_i = r_i.sinh(\theta_i), \quad yr_i = r_i.cosh(\theta_i)$$
$$r_i = \theta_i, \quad \theta_i = a.\pi.R_{rand}$$

- c1, c2 ∈ [1,2]: coefficients d'attraction vers la moyenne ou la meilleure solution.

Ce qui distingue l'algorithme BES, c'est sa simplicité de mise en œuvre, son nombre réduit de paramètres à ajuster, et sa forte capacité à échapper aux optima locaux grâce à un bon équilibre entre exploration (global search) et exploitation (local search). Ces caractéristiques le rendent particulièrement adapté pour traiter des problèmes d'optimisation complexes, non linéaires, et à variables multiples, que ce soit dans les domaines de l'ingénierie, de l'intelligence artificielle, de la bio-informatique ou encore de l'optimisation des systèmes de contrôle tels que les régulateurs PID.

Ainsi, l'algorithme BES s'inscrit dans la catégorie des algorithmes bio-inspirés récents et représente une alternative intéressante aux méthodes.[16]

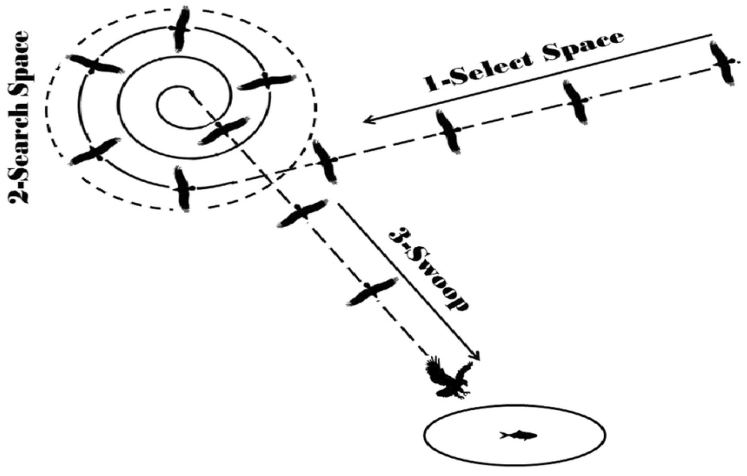


Figure II. 1 : Étapes de chasse de l'aigle à tête blanche**.**[16]

Ce paragraphe décrit les étapes générales de l'optimisation dans l'algorithme BES. L'algorithme BES comprend sept étapes principales**,** détaillées ci-dessous. En outre, le diagramme de flux illustrant l'algorithme BES est présenté graphiquement dans la figure II.2.

Étape 1 : Les paramètres de l'algorithme BES et du problème d'optimisation sont configurés : des valeurs sont attribuées au nombre maximal d'itérations T, à la dimension du problème dim, ainsi qu'à la taille de la population N.

Étape 2 : La position initiale de chaque membre de la population est établie, avec une matrice aléatoire de dimensions N×dim , générée pour représenter la population initiale à l'intérieur des bornes de recherche.

Étape 3 : La valeur de la fonction d'aptitude (fitness**)** de chaque individu est déterminée à l'aide de la fonction d'évaluation, puis les individus sont classés par ordre croissant de valeur, ce qui permet d'identifier la meilleure solution actuelle P_b

Étape 4 : Lors de la phase de sélection, les positions des individus sont ajustées selon l'équation **(II.2) ,** puis la valeur globale de l'aptitude est recalculée une fois la mise à jour terminée. Seuls les individus ayant une meilleure aptitude que leur état précédent sont conservés.

Étape 5 : Lors de la phase de recherche, les positions sont modifiées selon l'équation **(II.3),** puis l'évaluation globale de la population est mise à jour. Encore une fois, seules les solutions améliorées sont retenues.

Étape 6 : Lors de la phase d'attaque (swooping)**,** les positions sont mises à jour conformément à l'équation **(II.4),** et la valeur de fitness est recalculée. Seules les améliorations par rapport à l'état précédent sont acceptées.

Étape 7 : Le processus itératif, englobant les étapes 4 à 6, est répété jusqu'à ce que la condition d'arrêt soit remplie. À ce moment-là, l'algorithme prend fin et la meilleure solution globale trouvée est renvoyée en sortie**.**[16]

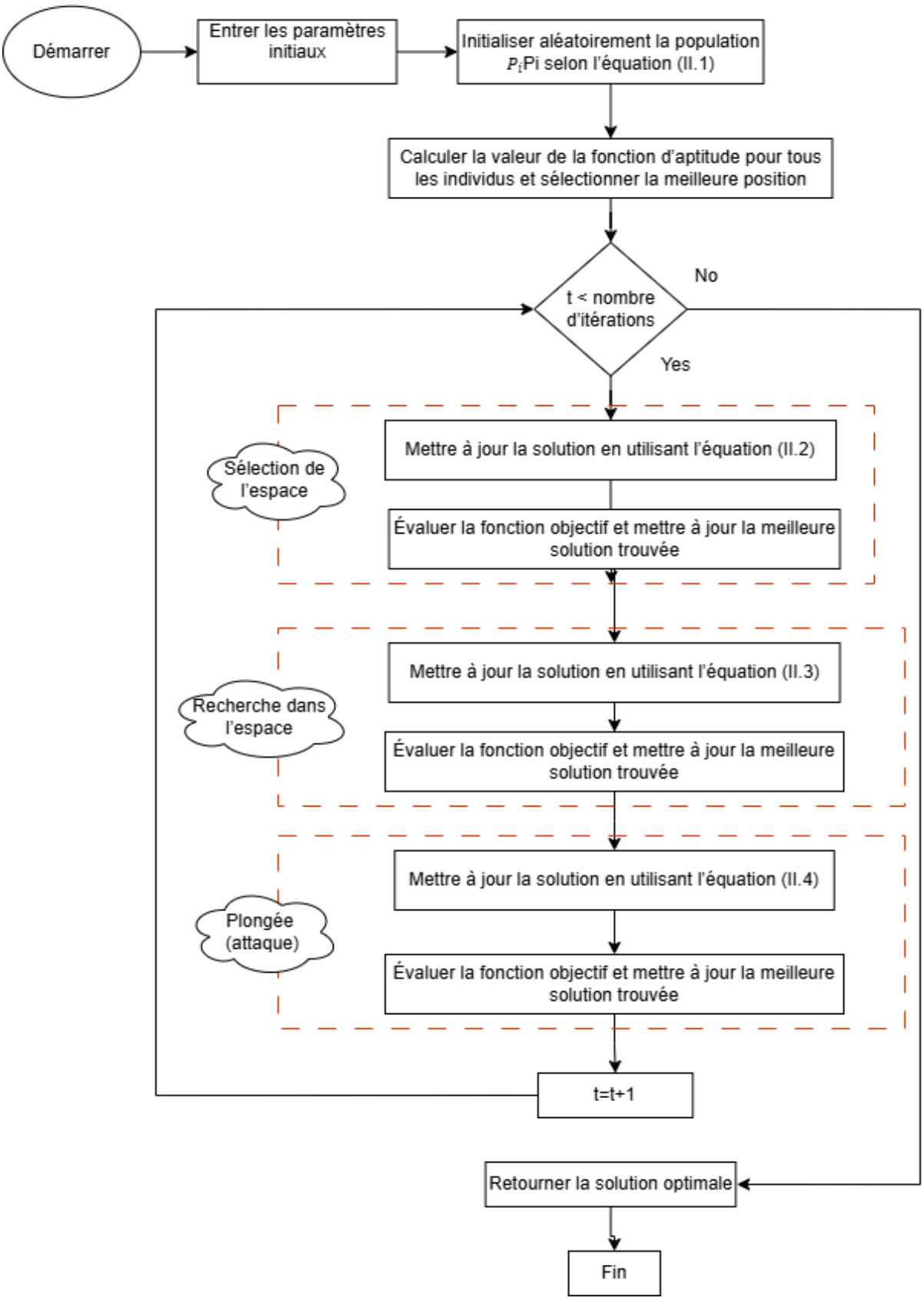


Figure II. 2 : Organigramme de l'algorithme BES.[16]

L'algorithme 1 présente le pseudo-code exposant les étapes générales de l'algorithme Bald Eagle Search (BES). Implémenté sous MATLAB**,** BES est disponible en code source libre. Ce code permet de traiter efficacement des problèmes de référence standards, ce qui contribue à la popularité croissante de l'algorithme.[16]

- **Algorithme 1 : Pseudo-code de l'algorithme BES**

1. **Entrée** : Initialisation des paramètres :
   - bornes supérieures et inférieures des variables
   - taille du problème (dim)
   - taille de la population (N)
   - nombre maximal d'itérations (iter)
2. Initialisation de la population P_i (pour i=1,2,..., N) selon l'équation (II.1)
3. Calcul de la valeur de la fonction de fitness pour chaque individu
4. Initialisation du compteur d'itération t=1
5. **Tant que** t<iter **faire**
6. **Pour** i=1 à N **faire** /* Phase de sélection */
7. Mettre à jour la position actuelle selon l'équation (II.2)
8. Évaluer la valeur de fitness fit_i de l'individu iii
9. **Fin pour**
10. **Pour** i=1 à N **faire** /* Phase de recherche */
11. Mettre à jour la position actuelle selon l'équation (II.3)
12. Évaluer la valeur de fitness fit_i de l'individu iii
13. **Fin pour**
14. **Pour** i=1 à N **faire** /* Phase d'attaque (swooping) */
15. Mettre à jour la position actuelle selon l'équation (II.4)
16. **Fin pour**
17. Mettre à jour la meilleure position trouvée jusqu'à présent P_b
18. t=t+1
19. Fin du bloc **while**
20. **Sortie** : La meilleure solution trouvée P_b

II.3 conclusion

L'algorithme d'optimisation Bald Eagle Search (BES) s'inscrit dans la lignée des méthodes bio-inspirées récentes, offrant une solution efficace et robuste pour l'exploration d'espaces de recherche complexes. À travers son analogie comportementale avec la stratégie de chasse de l'aigle à tête blanche, il réussit à combiner judicieusement les phases d'exploration globale et d'exploitation locale, permettant ainsi d'éviter les pièges des optima locaux tout en assurant une convergence rapide vers des solutions optimales.

Les différentes phases de l'algorithme sélection, recherche et attaque ont été décrites en détail, accompagnées de leurs formulations mathématiques, de leur représentation algorithmique ainsi que de leur intégration dans un schéma de résolution itératif. Sa simplicité de mise en œuvre, son efficacité dans les problèmes de haute dimension, et sa capacité d'adaptation à divers domaines d'application en font un candidat de choix pour l'optimisation de systèmes complexes.

Dans le cadre de ce projet, l'algorithme BES sera utilisé comme un outil d'optimisation pour ajuster les paramètres du régulateur PID, dans le but d'améliorer la performance du système de régulation du niveau de liquide dans le réservoir. Le chapitre suivant présentera la simulation ainsi que les évaluations pratiques qui en résultent.

Chapitre III

Simulation et interprétation des résultats

III.1 Introduction

Ce chapitre est consacré à la mise en œuvre pratique et à la simulation du système de régulation du niveau de liquide. Il présente dans un premier temps les composants matériels et logiciels utilisés, notamment la carte Arduino UNO, le capteur ultrasonique HC-SR04, l'afficheur LCD, ainsi que le logiciel de simulation Proteus.

La simulation du système TOR (Tout ou Rien) est d'abord abordée, permettant une validation fonctionnelle simple du comportement du système dans un environnement virtuel. Ensuite, une modélisation plus avancée est introduite avec l'environnement MATLAB/Simulink, dans lequel un régulateur PID est implémenté pour assurer une régulation plus fine et dynamique.

Les performances du contrôleur PID sont ensuite optimisées à l'aide de l'algorithme métaheuristique Bald Eagle Search (BES), et comparées à celles obtenues par une méthode empirique. Cette démarche permet non seulement d'illustrer l'efficacité de l'approche intelligente, mais aussi de démontrer l'impact des différents paramètres du PID sur le comportement du système simulé.

III.2 Outils Hardware

III.2.1 Arduino UNO

L'Arduino UNO (Figure III.1) est la meilleure carte pour débuter en électronique et en programmation. Si c'est votre première expérience avec cette plateforme, l'UNO est la carte la plus robuste avec laquelle vous pouvez commencer à expérimenter. L'UNO est la carte la plus utilisée et la mieux documentée de toute la famille Arduino.

L'Arduino UNO est une carte à microcontrôleur basée sur l'ATmega328P. Elle dispose de 14 broches numériques d'entrée/sortie (dont 6 peuvent être utilisées comme sorties PWM), de 6 entrées analogiques, d'un résonateur en céramique de 16 MHz, d'une connexion USB, d'une prise d'alimentation, d'un en-tête ICSP et d'un bouton de réinitialisation. Elle contient tout ce qu'il faut pour faire fonctionner le microcontrôleur ; il suffit de la connecter à un ordinateur avec un câble USB ou de l'alimenter avec un adaptateur secteur ou une batterie pour commencer. Vous pouvez expérimenter avec votre UNO sans trop vous soucier de faire une

erreur ; dans le pire des cas, vous pouvez remplacer la puce pour quelques dollars et recommencer [17].

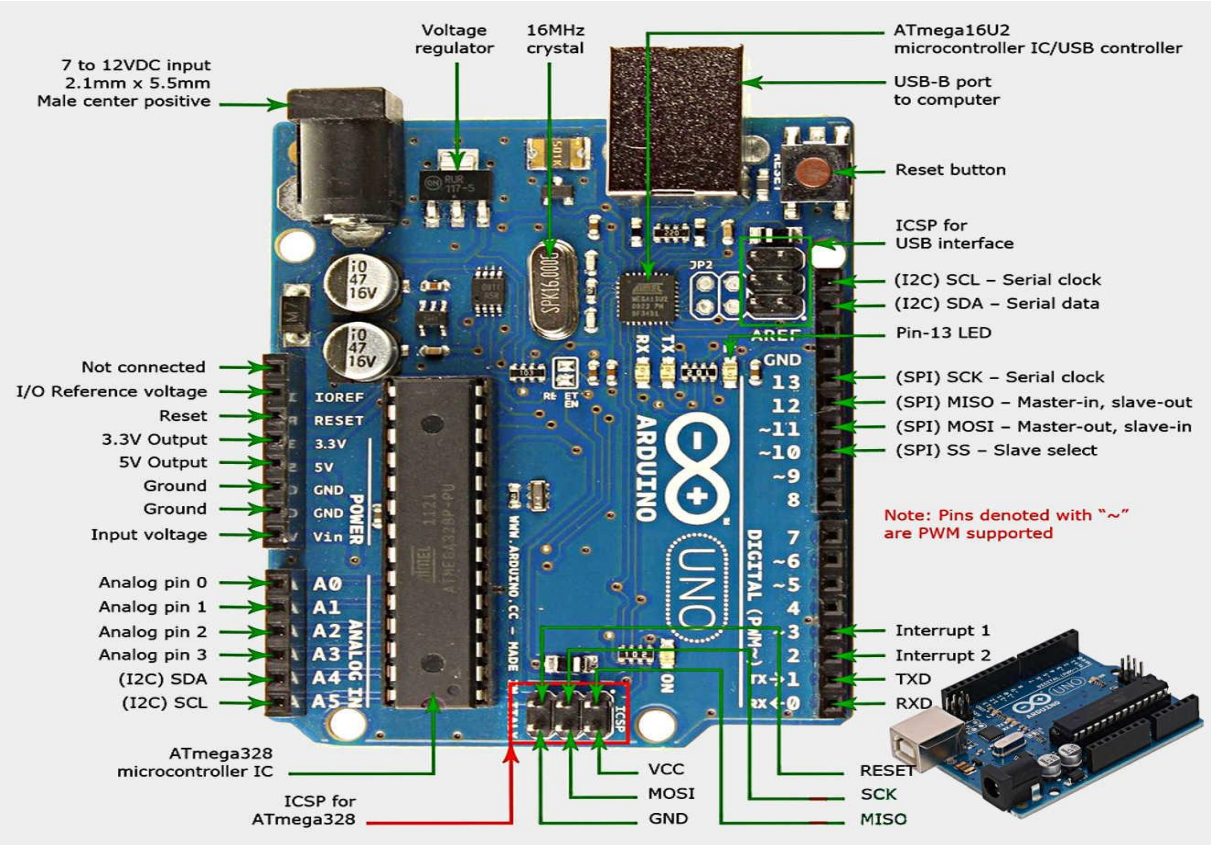


Figure III. 1: Constitution de la carte Arduino UNO.[18].

III.2.1.1 Pourquoi Arduino UNO

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

- **Multi plateforme** : le logiciel Arduino, écrit en JAVA, tourne sous les systèmes d'exploitation Windows, Macintosh et Linux. La plupart des systèmes à microcontrôleurs sont limités à Windows.
- **Un environnement de programmation clair et simple** : l'environnement de programmation Arduino (le logiciel Arduino IDE) est facile à utiliser pour les débutants, tout en étant assez flexible pour que les utilisateurs avancés puissent en tirer profit également.

- **Logiciel Open Source et extensible** : le logiciel Arduino et le langage Arduino sont publiés sous licence open source, disponible pour être complété par des programmateurs expérimentés. Le logiciel de programmation des modules Arduino est une application JAVA multi plateformes (fonctionnant sur tout système d'exploitation), servant d'éditeur de code et de compilateur, et qui peut transférer le programme au travers de la liaison série (RS232, Bluetooth ou USB selon le module).
- **Matériel Open source et extensible** : les cartes Arduino sont basées sur les Microcontrôleurs ATMEL ATMEGA8, ATMEGA168, ATMEGA 328, les schémas des modules sont publiés sous une licence créative Commons, et les concepteurs des circuits expérimentés peuvent réaliser leur propre version des cartes Arduino, en les complétant et en les améliorant. Même les utilisateurs relativement inexpérimentés peuvent fabriquer la version sur plaque d'essai de la carte Arduino, dont le but est de comprendre comment elle fonctionne pour économiser le coût3.
- **Le prix (réduits)** : les cartes Arduino sont relativement peu coûteuses comparativement aux autres plates-formes. La moins chère des versions du module Arduino peut être assemblée à la main, (les cartes Arduino pré-assemblées coûtent moins de 3500 Dinars) [19].

III.2.1.2 Caractéristiques techniques de l'Arduino UNO

Le tableau III.1 donne les **c**aractéristiques techniques de l'Arduino UNO :

Tableau III. 1: Caractéristiques techniques de l'Arduino UNO [19].

Microcontrôleur	ATmega328
Tension de fonctionnement	5V
Tension d'alimentation (recommandé)	7-12 V
Tension d'alimentation (limite)	6-20 V
Broche E/S numériques	14 (dont 6 disposent d'une sortie PWM)
Broche d'entrée analogique	6 (utilisable en broches E/S numérique)
Intensité Maxi disponible par broche E/S (5V)	40 mA
Intensité Maxi disponible pour la sortie 3,3V	50 Ma
Mémoire programme flash	32 KB (ATmega328) dont 0,5 sont utilisés par le boot loader
Mémoire SRAM (mémoire volatile)	2 KB (ATmega 328)
Mémoire EEPROM (mémoire non volatil)	1KB (ATmega 328)
Vitesse d'horloge	16 MHz

III.2.2 Capteur ultrason (HC-SR04)

Le HC-SR04 (Figure III.2) est un capteur qui utilise les ultrasons pour déterminer la distance d'un objet de 2cm à 400cm. La mesure s'effectue "sans contact" et dispose d'une précision de 3mm dans les conditions optimales. Le module inclus un émetteur et un récepteur ultrason ainsi qu'un système de contrôle[20].



Figure III. 2: Capteur Ultrason HC-SR04.[21]

III.2.2.1 Caractéristique

Le tableau III.2 donne les caractéristiques de capteur Ultrason HC-SR04 :

Tableau III. 2: caractéristiques de capteur Ultrason HC-SR04.[20]

Nombre de pattes	4
Dimension	45mm x 20mm x 15mm
Alimentation max	5.5 V DC
Alimentation min	4.5 V DC
Plage de mesure	2cm à 400cm
Courant de fonctionnement max	10 mA
Courant de fonctionnement min	20 mA
Résolution de la mesure	0.3 cm
Angle de mesure	15°
Largeur d'impulsion sur l'entrée de déclenchement	10 µs
La fréquence des ultrasons	40 KHZ
TRIG	Entrée de déclenchement de la mesure (trigger IN PUT)
ECHO	Sortie de mesure donnée en écho (ECHO OUT PUT)
GND	Masse de l'alimentation

III.2.3 L'écran d'affichage LCD 2x16

Un écran LCD (Figure III.3) possède des cristaux liquides. Si vous regardez de très près sur l'‟écran, vous pouvez voir une grille de carré. Ces carrés sont appelés des pixels (de l'anglais " Picture Element "), Chaque pixel est un cristal liquide. Lorsque aucun courant ne le traverse,

ses molécules sont orientées dans un sens (admettons, 0°). En revanche lorsqu'un courant le traverse, ses molécules vont se tourner dans la même direction (90°). Pour qu'il y ait inscription sur l'écran de l'afficheur, il faut qu'il reçoive, sur son bus de D0 à D7 des codes binaires qui correspondent à des caractères alphanumériques (ou autres). Ce codage est appelé : code ASCII.

Pour pouvoir afficher des caractères sur l'écran il nous faudrait activer individuellement chaque pixel de l'écran. Un caractère est représenté par un bloc de 7*5 pixels. Ce qui fait qu'un écran de 16 colonnes et 2 lignes représente un total de 16*2*7*5 = 1120 pixels. Heureusement pour nous, des ingénieurs sont passés par là et nous ont simplifié la tâche.

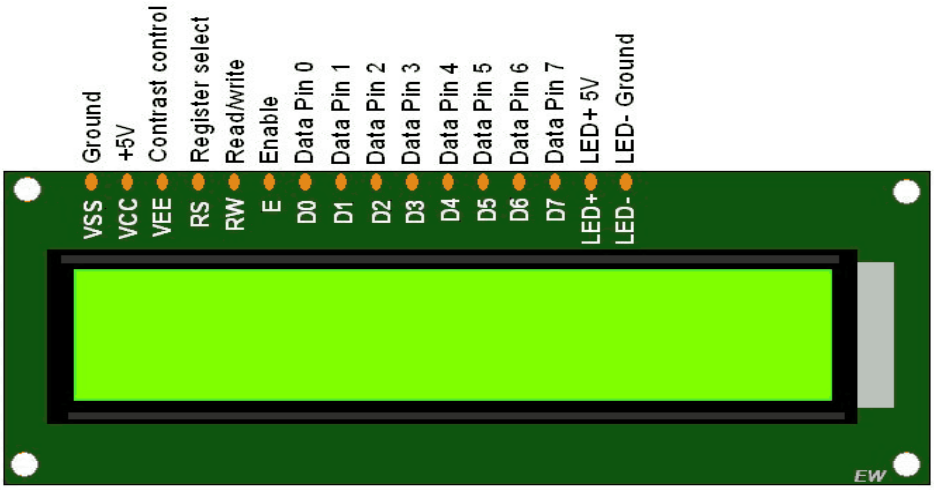


Figure III. 3: L'afficheur LCD 2x16.[22]

III.2.3.1 Caractéristiques de l'afficheur

Tout comme il existe un driver vidéo pour la carte graphique d'ordinateur, il existe un driver "LCD" pour l'afficheur. Ce composant va servir à décoder un ensemble "simple" de bits pour afficher un caractère à une position précise ou exécuter des commandes comme déplacer le curseur par exemple. Ce composant est fabriqué principalement par Hitachi et se nomme le HC44780. Il sert de décodeur de caractères. Ainsi, plutôt que de devoir multiplier les signaux pour commander les pixels un à un, il nous suffira d'envoyer des octets de commandes pour afficher, à partir de la colonne 3 sur la ligne 1.

Le tableau III.3 donne Les pins de LCD 2x16 **:**

Tableau III. 3: Les pins de LCD 2x16.[23]

Broche	Nom	Fonction
1	Vss	Masse
2	Vdd	Alimentation (+5V)
3	Vo	Cette tension permet, en la faisant varier entre 0 et +5V, le réglage du contraste de l'afficheur.
4	RS	Sélection du registre (Register Select) Grâce à cette broche, l'afficheur est capable de faire la différence entre une commande et une donnée. Un niveau bas indique une commande et un niveau haut indique une donnée.
5	R/W	Lecture ou écriture (Read/Write) L : Écriture
		H : Lecture
6	E	Entrée de validation (Enable) active sur front descendant. Le niveau haut doit être maintenue pendant au moins 450 ns à l'état haut.
7	D0	Bus de données bidirectionnel 3 états (haute impédance lorsque E=0)
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	
15	A	Anode rétro-éclairage (+5V)
16	K	Cathode rétro-éclairage (masse)

III.2.4 Pompe à eau

Une pompe à eau est un dispositif utilisé pour transférer de l'eau ou d'autres liquides d'un endroit à un autre. Elle contient généralement un moteur électrique qui génère l'énergie nécessaire pour faire tourner la partie rotative de la pompe. Ce mouvement rotatif est ensuite transformé en un flux de liquide par un mécanisme de déplacement, souvent basé sur l'effet centrifuge. La pompe est composée d'une entrée et d'une sortie pour le liquide, ainsi que d'un corps qui assure la conversion du mouvement en pression pour acheminer l'eau.

Les pompes à eau existent avec différentes capacités et se distinguent par leur débit, leur pression et leur consommation énergétique. Ces caractéristiques sont importantes à prendre en compte lors du choix d'une pompe adaptée à des besoins spécifiques.

L'Arduino peut être utilisé pour contrôler une pompe à eau directement via une sortie numérique (Digital Output), en utilisant un transistor comme interrupteur électronique pour activer ou désactiver automatiquement la pompe.

Il est également possible d'intégrer l'Arduino avec d'autres microcontrôleurs (MCUs) pour un contrôle plus précis du fonctionnement de la pompe. De plus, des capteurs (sensors) peuvent être utilisés avec l'Arduino pour mesurer le niveau de liquide, la température du liquide ou d'autres paramètres nécessaires pour un contrôle intelligent de la pompe.

Grâce à l'Arduino et à d'autres composants électroniques disponibles, il est possible de créer un système de contrôle automatique pour la pompe à eau (Fiigure III.4), qui peut être commandée à distance via Internet ou à l'aide d'applications mobiles ou de tablettes[25].



Figure III. 4: Pompe à Eau 5V. [24]

III.3 Les logiciels (Software)

III.3.1 IDE Arduino

Le code du programme écrit pour la carte Arduino est également appelé un croquis. Le logiciel utilisé pour développer de tels croquis pour un Arduino est l'IDE Arduino (Figure III.5).

L'EDI contient les parties suivantes :

- **Éditeur de texte :** C'est l'endroit où l'on peut écrire du code en langage de programmation C/C++.
- **Zone de messages :** Elle affiche les erreurs et fournit un retour lors de l'enregistrement et de l'exportation du code.

- **Texte :** La console affiche le texte généré par l'environnement Arduino, y compris les messages d'erreur complets et d'autres informations.
- **Barre d'outils de la console :** Cette barre comprend des boutons comme Compiler, Téléverser, Nouveau, Ouvrir, Enregistrer et Moniteur Série [26].

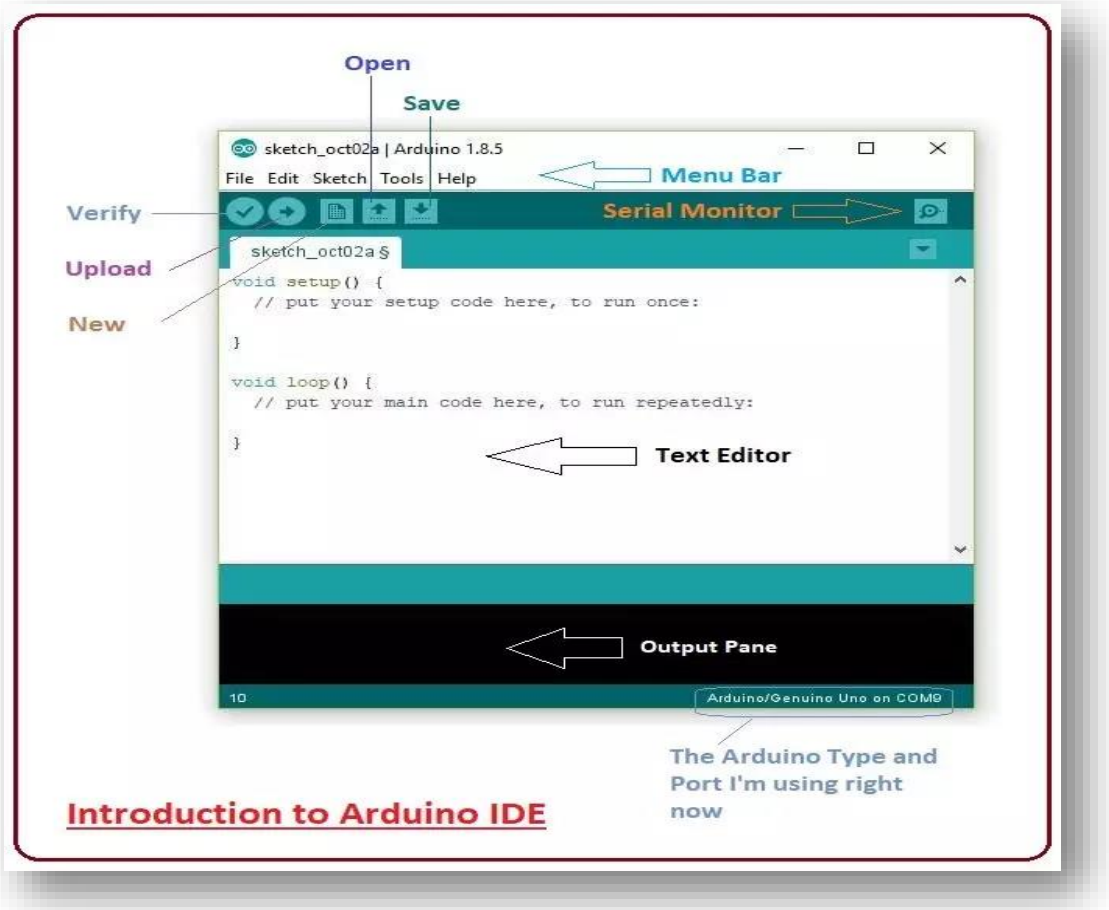


Figure III. 5: formule d'Arduino IDE.[27].

III.3.2 Logiciel PROTEUS

PROTEUS (Figure III.6) est un simulateur basé sur l'exécution pour les machines MIMD, il exécute directement la plupart des instructions afin d'atteindre des performances très élevées. En dépit de sa rapidité exceptionnelle, PROTEUS produit des simulations précises et a reproduit plusieurs résultats publiés. Sa structure modulaire permet de simuler un large éventail d'architectures et facilite la personnalisation.

PROTEUS est un outil complet pour les systèmes parallèles : il offre un support intégré pour la collecte et l'affichage de données, ainsi que pour la surveillance et le débogage non intrusifs.

PROTEUS permet aux utilisateurs de faire un compromis entre précision et performance afin d'atteindre les meilleures performances selon leurs exigences en matière de précision. PROTEUS étend l'utilisation de l'augmentation de code pour fournir une détection de débordement de pile, du profilage, et un contrôle explicite des coûts du code utilisateur. L'augmentation empêche également les threads de progresser excessivement dans le temps simulé, ce qui améliore la précision de la simulation[28].

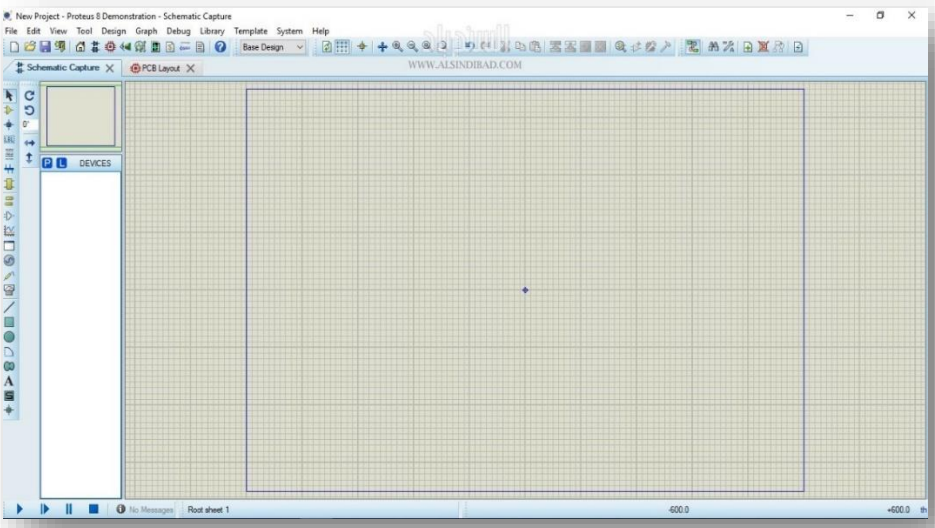


Figure III. 6: logiciel PROTEUS.[29].

La figure III.7 donne l'architecture générale du système TOR.

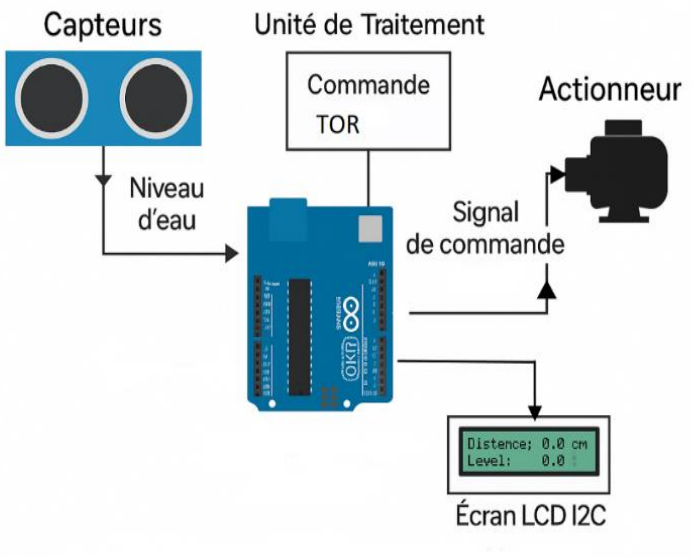


Figure III. 7: Architecture générale du système TOR.

La figure III.8 donne l'organigramme de système TOR.

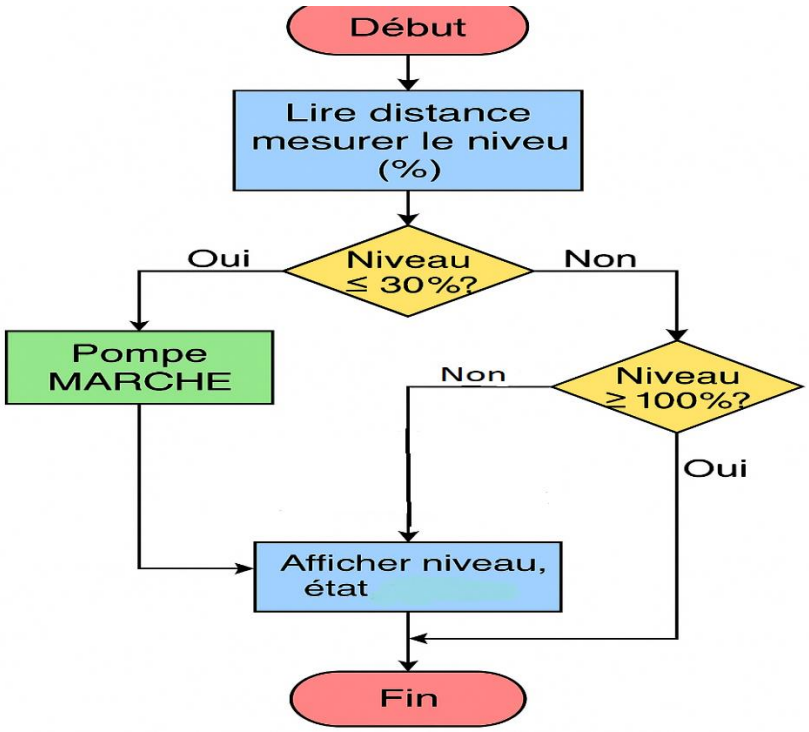


Figure III. 8: Organigramme de système TOR.

III.4 La simulation de système TOR

- **Importer la bibliothèque du capteur**

Après avoir importé le fichier HEX, double-cliquez sur le composant HC-SR04 dans Proteus. Dans le champ "Program File", sélectionnez le chemin du fichier HEX correspondant au code. Ensuite, cliquez sur "OK" pour valider l'intégration, et le code s'exécutera pendant la simulation Figure III.9.

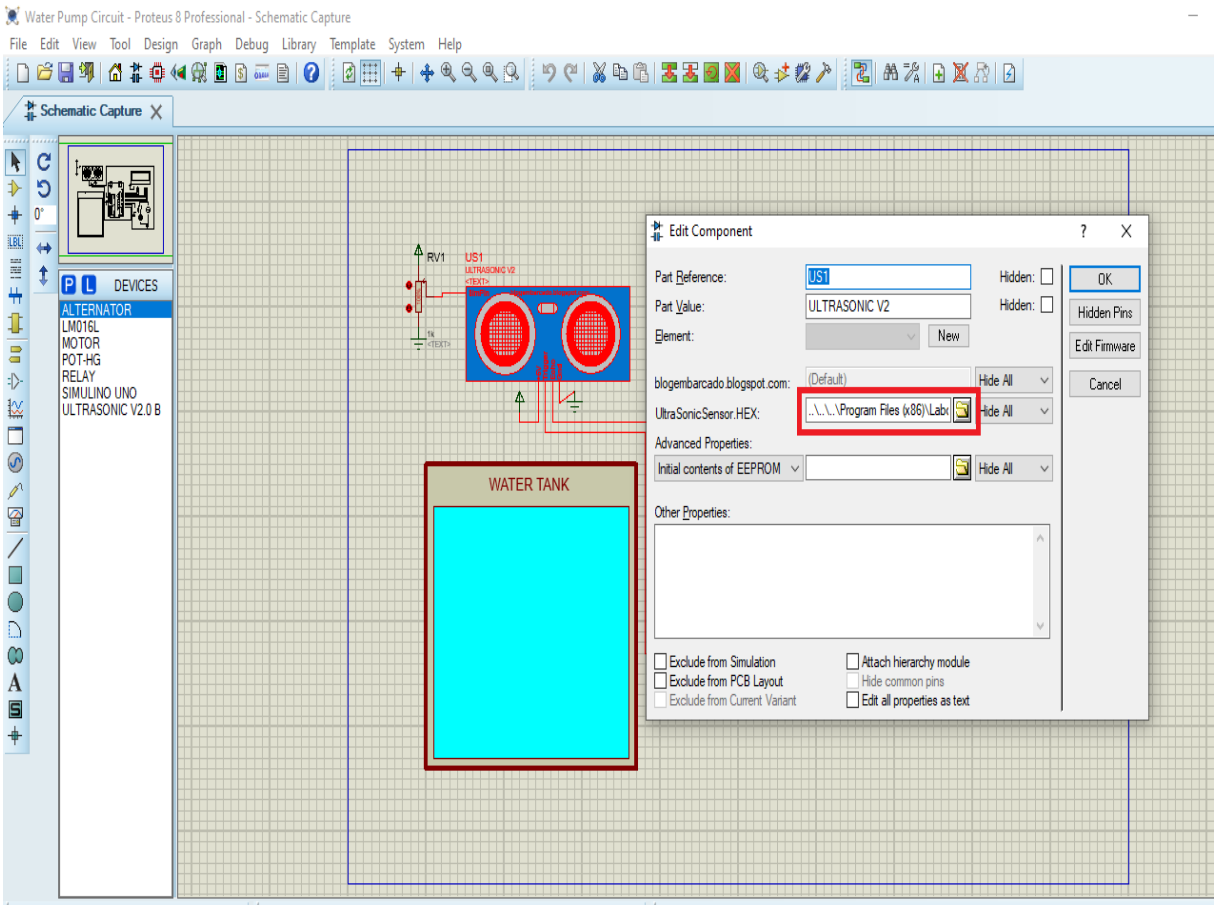


Figure III. 9: Importer la bibliothèque du capteur.

- **Récupérer le code Arduino**

Après avoir écrit le code dans le logiciel Arduino IDE, on choisit le type de carte (par exemple Arduino Uno), puis on clique sur "Exporter le fichier compilé au format HEX". Ensuite, on ouvre Proteus, on ajoute la carte Arduino, puis on charge le fichier HEX dans le champ "Program File" correspondant (Figure III.10) .

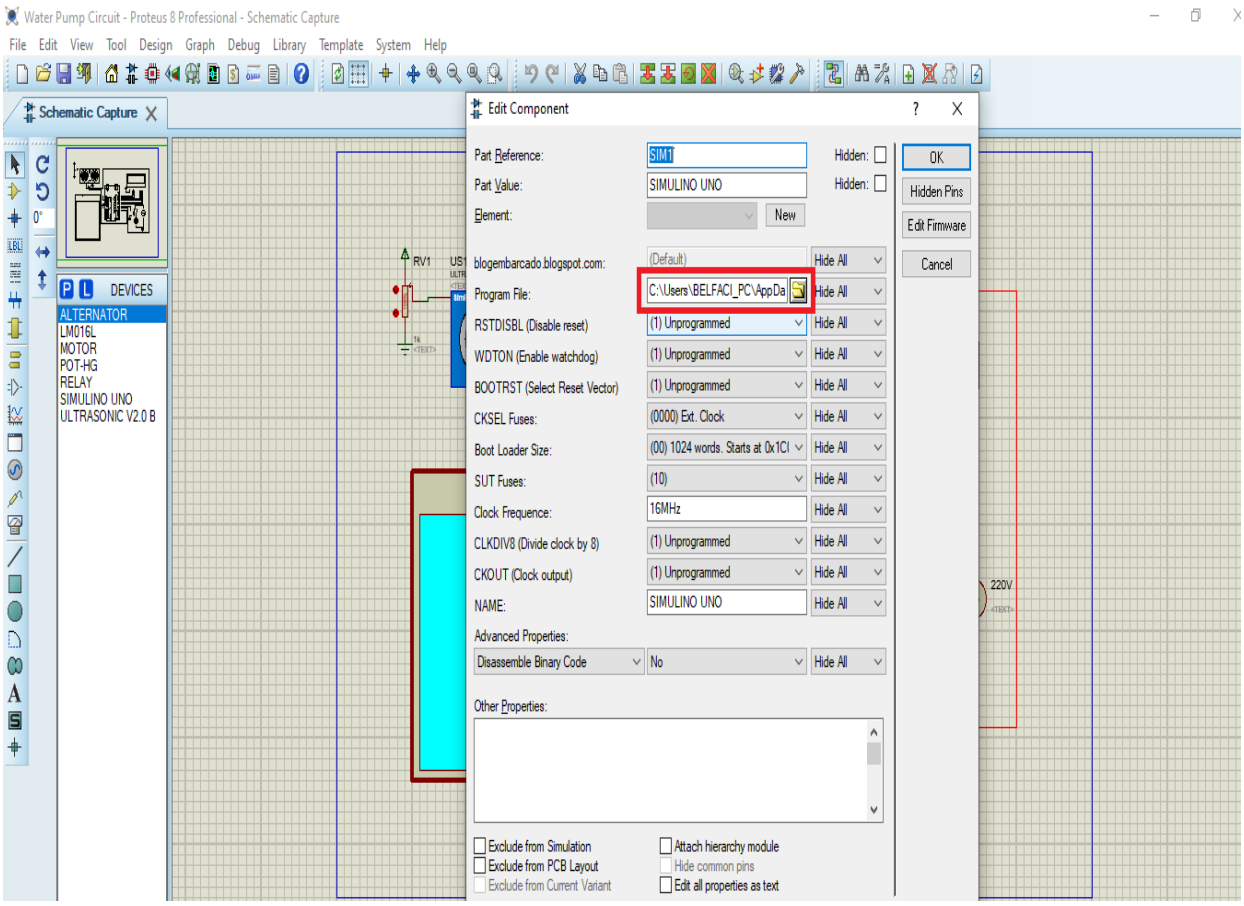


Figure III. 10: Récupérer le code Arduino.

➢ **Composants utilisés dans le projet**

Ce projet repose sur un ensemble de composants électroniques et logiciels intégrés permettant de réaliser un système de régulation du niveau de liquide dans un réservoir à l'aide d'un algorithme PID. Les principaux composants utilisés sont les suivants :

- **Carte Arduino UNO** : constitue l'unité de contrôle principale du système, où le code est exécuté pour traiter les données et faire fonctionner l'algorithme PID.
- **Capteur ultrasonique HC-SR04** : utilisé pour mesurer la distance entre la surface de de liquide et le capteur, afin de déterminer le niveau de liquide dans le réservoir.
- **Écran LCD 16x2** : permet d'afficher les informations relatives au niveau de liquide et à l'état du système de contrôle.
- **Relais (Relay)** : utilisé pour activer ou désactiver la pompe en fonction des signaux de la carte Arduino.

- **Pompe (symbolique)** : représente l'élément d'action dans le système, qui se déclenche lorsque le niveau d'un liquide devient insuffisant.
- **Source de courant alternatif (220V)** : sert à alimenter la pompe via le relais.
- **Réservoir d'un liquide (virtuel)** : élément graphique dans Proteus utilisé pour visualiser le niveau de liquide.
- **Potentiomètre** : utilisé pour tester la réponse du système dans différentes conditions lors de la simulation.

Sur le plan logiciel, l'environnement Arduino IDE a été utilisé pour rédiger le code et générer le fichier au format HEX. Des bibliothèques spécifiques comme NewPing pour le capteur ultrasonique et LiquidCrystal pour l'écran LCD ont été intégrées. L'ensemble du système a été simulé dans le logiciel Proteus 8 Professional, qui offre un environnement visuel pour connecter les composants et exécuter le modèle virtuellement.

La figure III.11 présente une situation où le niveau d'un liquide (l'eau) est de 45 %, et l'on observe que la pompe est à l'arrêt, car le seuil permettant son activation, fixée à 30 %, n'a pas été atteint. Cela indique que le système fonctionne correctement, en respectant la logique de contrôle prévue, et qu'il maintient une gestion stable du niveau de liquide dans le réservoir.

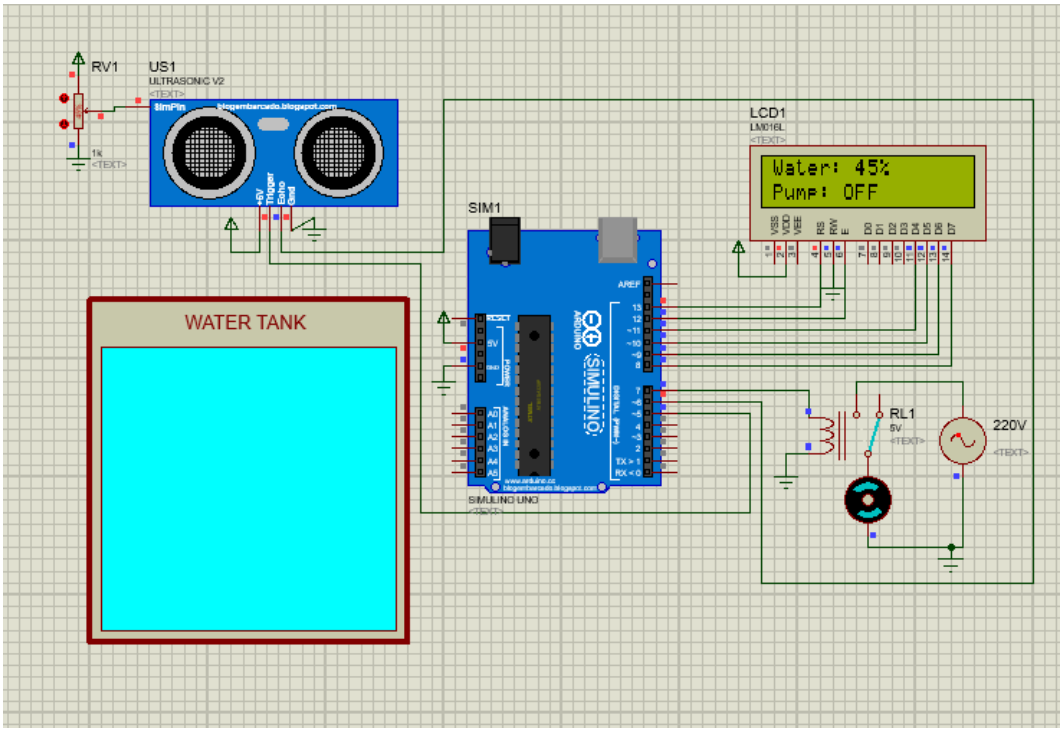


Figure III. 11: Premier cas de simulation

La Figure III.12 illustre une situation où le niveau de liquide est de 29 %, et l'on constate que la pompe est en marche, car ce niveau a atteint le seuil autorisant son activation, fixé à 30 %. Cela démontre que le système répond de manière efficace aux variations, en adaptant son fonctionnement aux conditions mesurées avec une certaine précision.

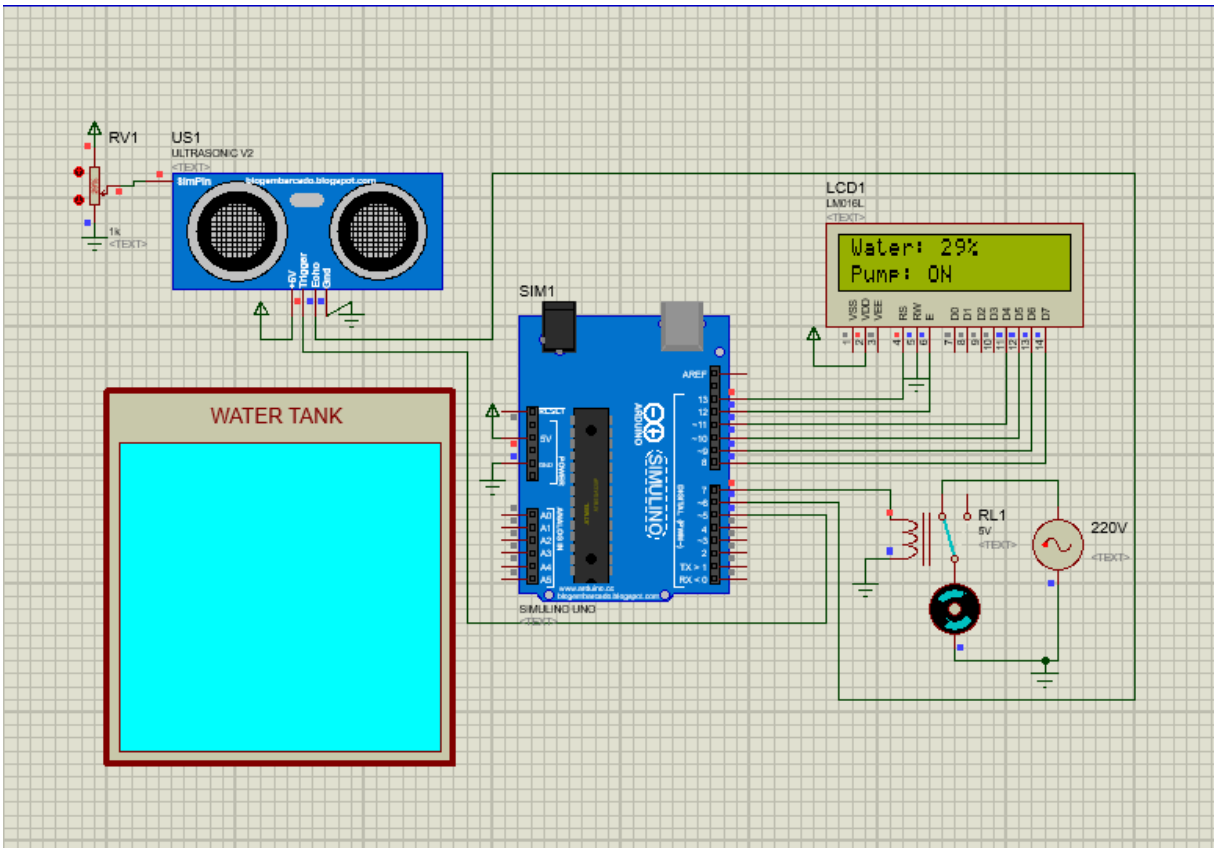


Figure III. 12: Deuxième cas de simulation

III.5 Matlab Simulink

Simulink figure III.13 est un environnement de simulation graphique intégré à MATLAB, développé par MathWorks, il permet de modéliser, simuler et analyser des systèmes dynamiques continus, discrets ou mixtes, dans de nombreux domaines :

- Automatique (systèmes de contrôle)
- Électronique embarquée
- Télécommunications
- Énergies renouvelables
- Traitement du signal et de l'image

- Robotique et mécatronique.[30]

III.5.1 Composants principaux de Simulink

1. **Éditeur Simulink (Simulink Editor)**
   Une interface graphique où l'on construit des modèles à l'aide de blocs connectés par des lignes de signal.
2. **Navigateur de bibliothèques (Library Browser)**
   Contient des milliers de blocs : sources, fonctions de transfert, systèmes de contrôle, capteurs, afficheurs, etc.
3. **Contrôle de simulation**
   Pour démarrer, mettre en pause, ou arrêter une simulation, et configurer la durée.
4. **Entrées / Sorties**
   Utilisés pour simuler les interactions avec l'environnement externe ou MATLAB.[30]

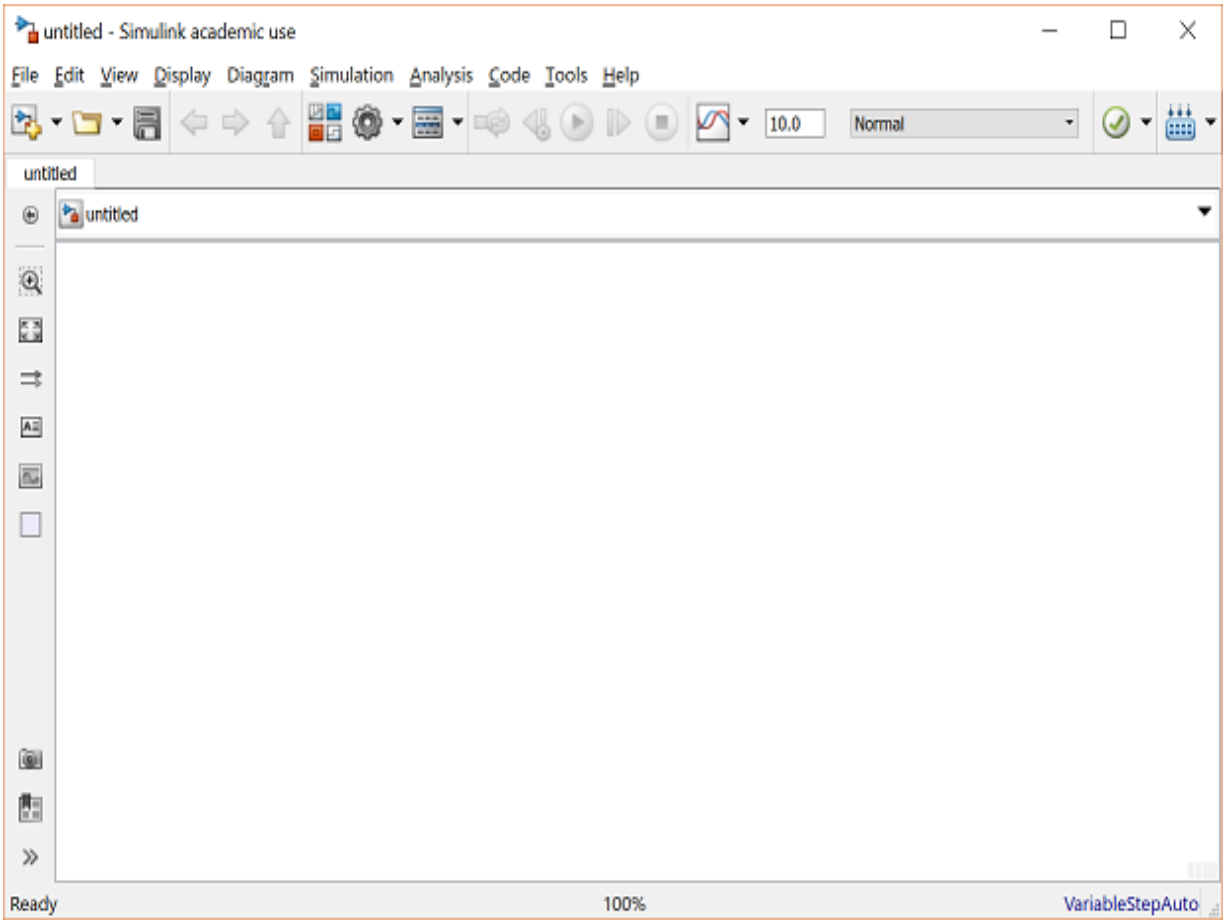


Figure III. 13: Interface de matlab Simulink.[31].

Le tableau III.4 donne la Différence entre MATLAB et Simulink :

Tableau III. 4: Différence entre MATLAB et Simulink.[32]

Élément	MATLAB	Simulink
Type de programmation	Textuelle (basée sur des scripts)	Graphique (blocs et diagrammes)
Usage	Calculs numériques, scripts	Simulation de systèmes dynamiques
Interaction	Ligne de commande	Glisser-déposer

- **Avantages de Simulink**

- Interface intuitive sans codage nécessaire.
- Intégration complète avec MATLAB.
- Simulation temps réel et hors ligne.
- Support du Hardware-In-The-Loop (HIL)**.**
- Compatible avec les cartes de prototypage (Arduino, etc.).

- **Inconvénients**

- Logiciel **propriétaire et payant**.
- Courbe d'apprentissage pour les débutants.
- Consommation élevée de ressources pour des simulations complexes.[30]

III.6 Description du système à commander :

La figure III.14 présente le système physique étudié : un réservoir dont le niveau de liquide est à réguler à l'aide d'une vanne proportionnelle. La vanne reçoit une commande en tension et ajuste le débit d'entrée. Cette modélisation constitue la base du système dynamique à asservir.

Description générale du schéma :

- **Source**: Représente la source de liquide qui sera pompée vers le réservoir.
- **Pompe**: Sert à acheminer le liquide depuis la source jusqu'au réservoir. Dans le cadre de cette étude, on considère une pompe à débit constant et maximal fixé à $Q_{pompe} = 0.5\ m^3/s$ Ainsi, le débit effectif d'entrée $q_e(t)$ dans le réservoir est contrôlé à l'aide d'une vanne proportionnelle.

- **Vanne proportionnelle (vanne à ouverture variable)** : Une vanne dont l'ouverture varie progressivement de 0 à 1 selon le signal de commande reçu du régulateur PID. Elle permet de réguler de manière proportionnelle la quantité de liquide entrant dans le réservoir, contrairement à une commande tout ou rien, elle permet de contrôler de manière continue et précise le débit de liquide entrant.
- **Réservoir**: C'est le contenant principal qui reçoit le liquide venant de la vanne, Il a une base de surface $A = 1m^2$ et une hauteur maximale $h_{max} = 2m$, ce qui correspond à un volume maximal de $V_{max} = 2m^3$. Le niveau du liquide dans le réservoir est noté $h(t)$, est variable régulée que l'on cherche à maintenir stable autour d'une consigne.
- **Orifice de sortie** : Il s'agit d'une ouverture ou d'un tuyau de section $a = 0.05m^2$ permettant à une partie du liquide de s'écouler hors du réservoir avec un débit de sortie noté $q_s(t)$, ce qui entraîne une diminution du niveau $h(t)$ au fil du temps. Cela représente la charge ou les pertes naturelles du système.

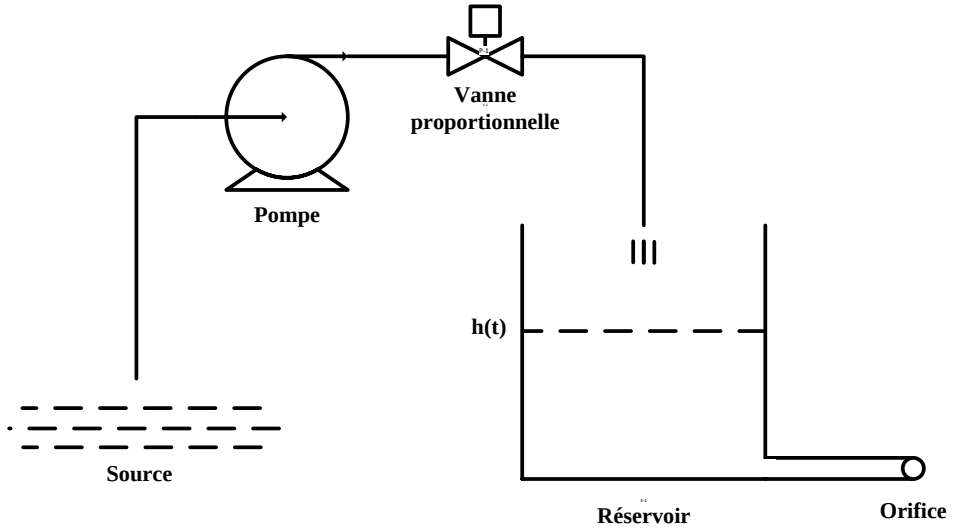


Figure III. 14: Représentation du système de régulation de niveau de liquide.

III.6.1 Modélisation du système :

Pour arriver à trouver l'expression de h(t) qui représente la sortie du système, on commence par calculer le débit net du réservoir qui donnée par l'équation suivante :

$$q_{net}(t) = q_e(t) - q_s(t)$$

Avec:

$$q_e(t) = Q_{pompe} \cdot vanne$$

Et:

$$q_s(t) = a \cdot v(t)$$

Pour déterminer l'expression de la vitesse d'écoulement $v(t)$ du liquide à la sortie du réservoir via l'orifice, on applique le principe de conservation de l'énergie mécanique totale, l'énergie potentielle de pression due à la hauteur du liquide est entièrement convertie en énergie cinétique à l'orifice, soit :

$$E_{pp} = E_c$$

$$p(t)V(t) = \frac{1}{2}mv^2(t)$$

$$\rho gh(t)V(t) = \frac{1}{2}mv^2(t)$$

où ρ est la masse volumique du liquide, g l'accélération gravitationnelle [9,81 m/s²], et $h(t)$ la hauteur du liquide dans le réservoir.

$$gh(t) = \frac{1}{2}v^2(t)$$

En isolant $v(t)$, on obtient :

$$v(t) = \sqrt{2gh(t)}$$

Donc:

$$q_s(t) = a\sqrt{2gh(t)}$$

Finalement on obtient l'expression du débit net $q_{net}(t)$:

$$q_{net}(t) = Q_{pompe} \cdot vanne - a\sqrt{2gh(t)}$$

Cette relation est utilisée pour établir le schéma bloc du système figure III.15 traduisant mathématiquement sa dynamique, afin d'obtenir l'évolution de la variable de sortie $h(t)$ dans le temps.

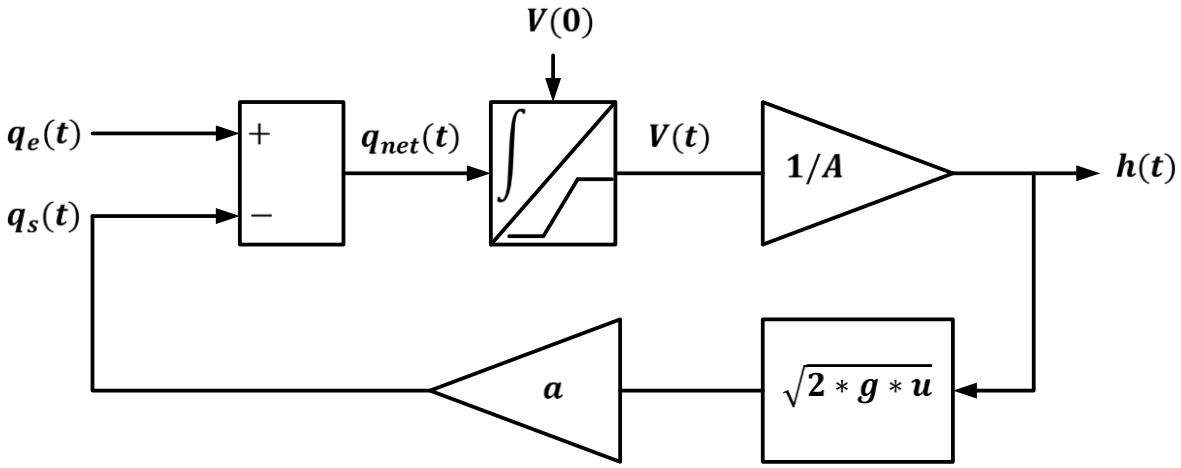


Figure III. 15: Schéma bloc de modélisation du niveau h(t).

La figure III.16 montre l'intégration du régulateur PID dans la boucle de rétroaction complète. Le PID compare la consigne avec la mesure du niveau, calcule l'erreur, et applique une commande appropriée à la vanne pour corriger le système.

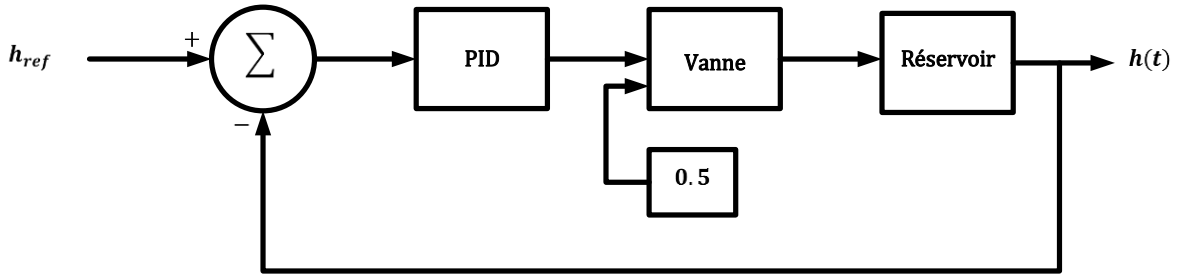


Figure III. 16: Schéma bloc global du système avec régulateur PID.

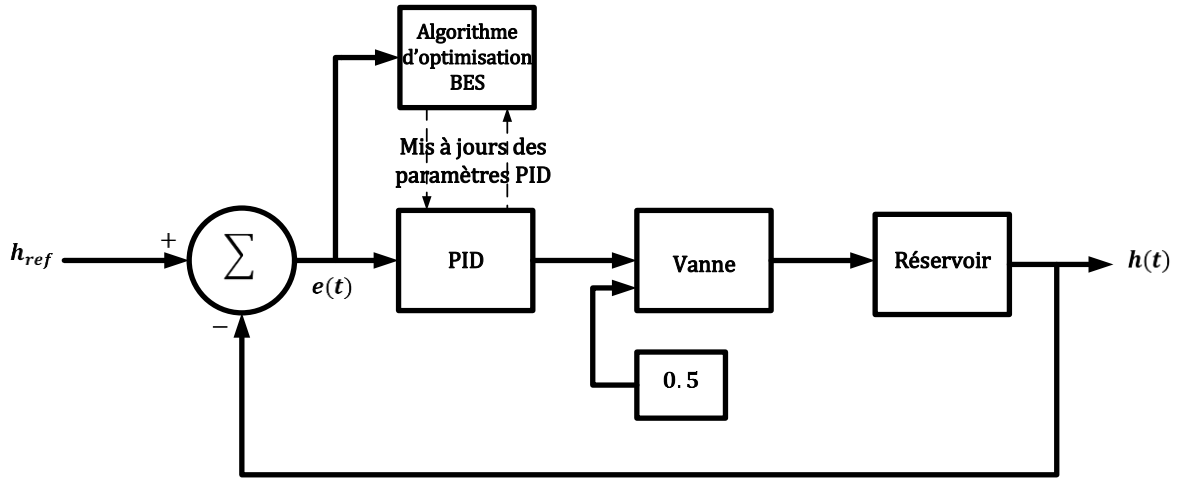


Figure III. 17 : Processus d'optimisation des paramètres PID via l'algorithme BES.

III.6.2 Discussion des resultats :

III.6.2.1 Comparaison des performances dynamiques des deux méthodes :

Dans cette section, l'efficacité de l'algorithme d'optimisation BES est évaluée et ses performances sont démontrées par des analyses de simulation, comparées à la méthode essais et erreurs. Le tableau III.5 présente les paramètres, le nombre d'itérations T et la taille de la population N de l'algorithme BES.

Tableau III. 5: les paramètres, le nombre d'itérations T et la taille de la population N de l'algorithme BES.

Paramètres	Valeurs
Taille de la population	20
Nombre d'itérations maximales	10
a	10
R	1.5

L'équation ci-dessous de l'Indice de Performance ITAE (Integral of Time-weighted Absulate Error) est utilisée comme fonction objective par l'algorithme BES et il est également utilisé pour évaluer la qualité des stratégies de réglage en mesurant l'erreur au fil du temps :

$$ITAE = \int_0^T t.|e(t)|dt$$

Tableau III. 6: Gains du PID et ITAE: Essais & erreurs vs BES.

Paramètres	Essais & erreurs	BES
K_p	2	2.5401
K_i	0	0
K_d	1	3.1186
ITAE	8.758	5.476

D'après le tableau III.6, on remarque que l'algorithme d'optimisation BES atteint un ITAE inférieur, indiquant son efficacité à minimiser l'erreur au fil du temps.

La figure III.18 représente la réponse temporelle du système réglé par méthode manuelle (trial & error), et sert de base comparative pour évaluer l'apport de l'optimisation.

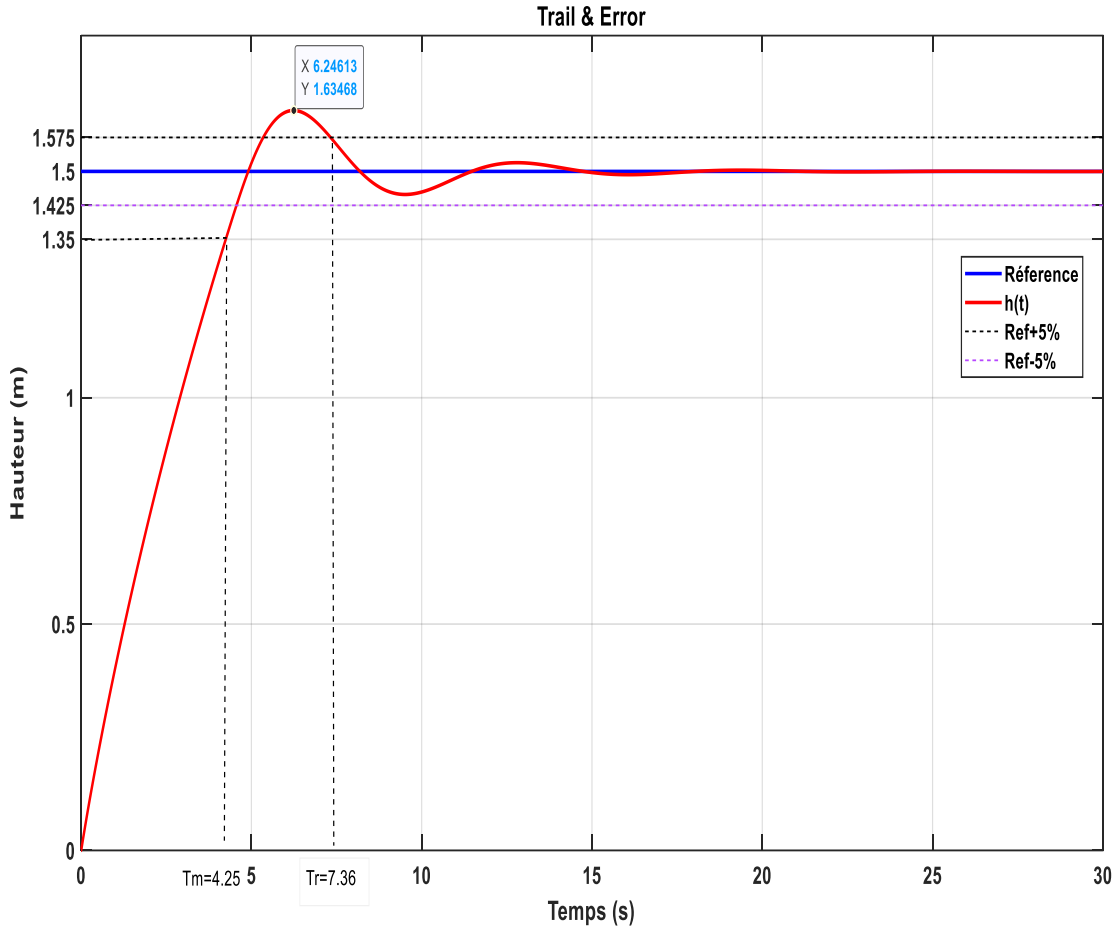


Figure III. 18 : Réponse temporelle du système sans optimisation.

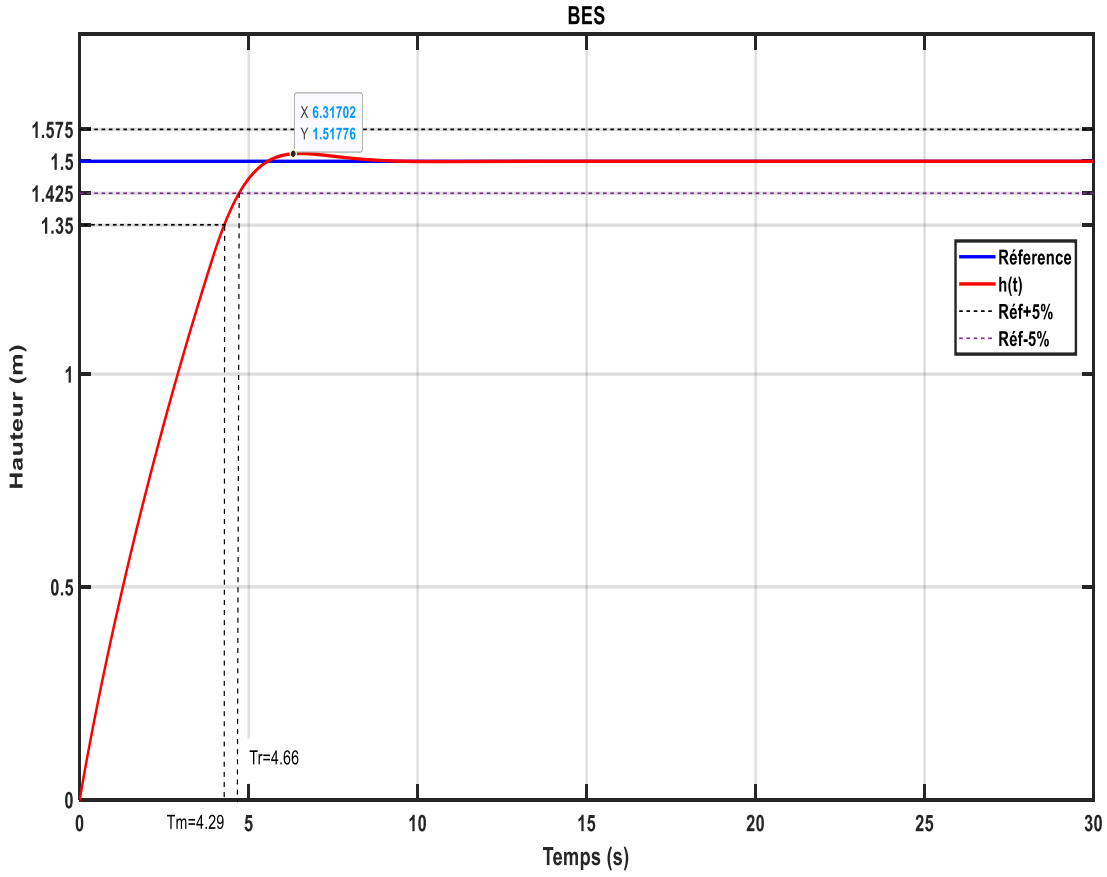


Figure III. 19 : Réponse temporelle du système avec optimisation par BES

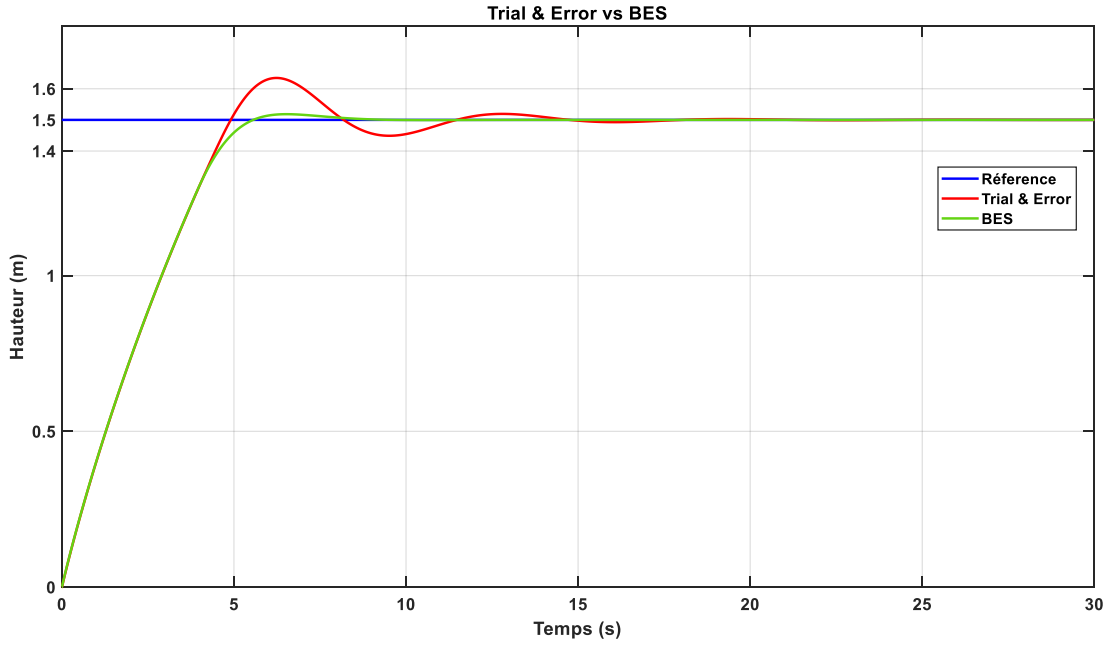


Figure III. 20: Comparaison des réponses : méthode empirique vs BES.

D'après les figures III.18-20 la méthode d'essais et d'erreurs entraîne une réponse du système plus lente, avec un temps de stabilisation de 7,36 secondes et un dépassement notable de 0,134 mètre. En revanche, l'algorithme d'optimisation BES montre une nette amélioration en offrant une réponse transitoire plus rapide et plus fluide, avec un temps de stabilisation de 4,66 secondes et un dépassement nettement réduit de seulement 0,017 mètre, démontrant ainsi sa performance supérieure en termes de précision de régulation et de justesse du système. Ces résultats, résumés dans le tableau III.7, mettent en évidence la dynamique avancée de l'optimisation BES.

Tableau III. 7: Performance dynamique selon la méthode de réglage des gains PID.

Méthode de réglage	Temps de stabilisation (s)	Dépassement (m)
Essais & erreurs	7,36	0,134
Algorithme BES	4,66	0,017

III.6.2.2 Effets d'augmentation des gains sur la réponse du système :

À présent, on procède à l'augmentation individuelle des gains K_p, K_i et K_d, en conservant les deux autres constants par rapport aux valeurs optimales obtenues via l'algorithme d'optimisation BES, afin d'observer leur impact sur la réponse du système.

- **Effet de k_p:**

La figureIII.21 montre que l'augmentation de K_p accélère la réponse mais peut aussi provoquer un dépassement important. Elle met en lumière l'effet direct de ce paramètre proportionnel sur la rapidité et la stabilité du système.

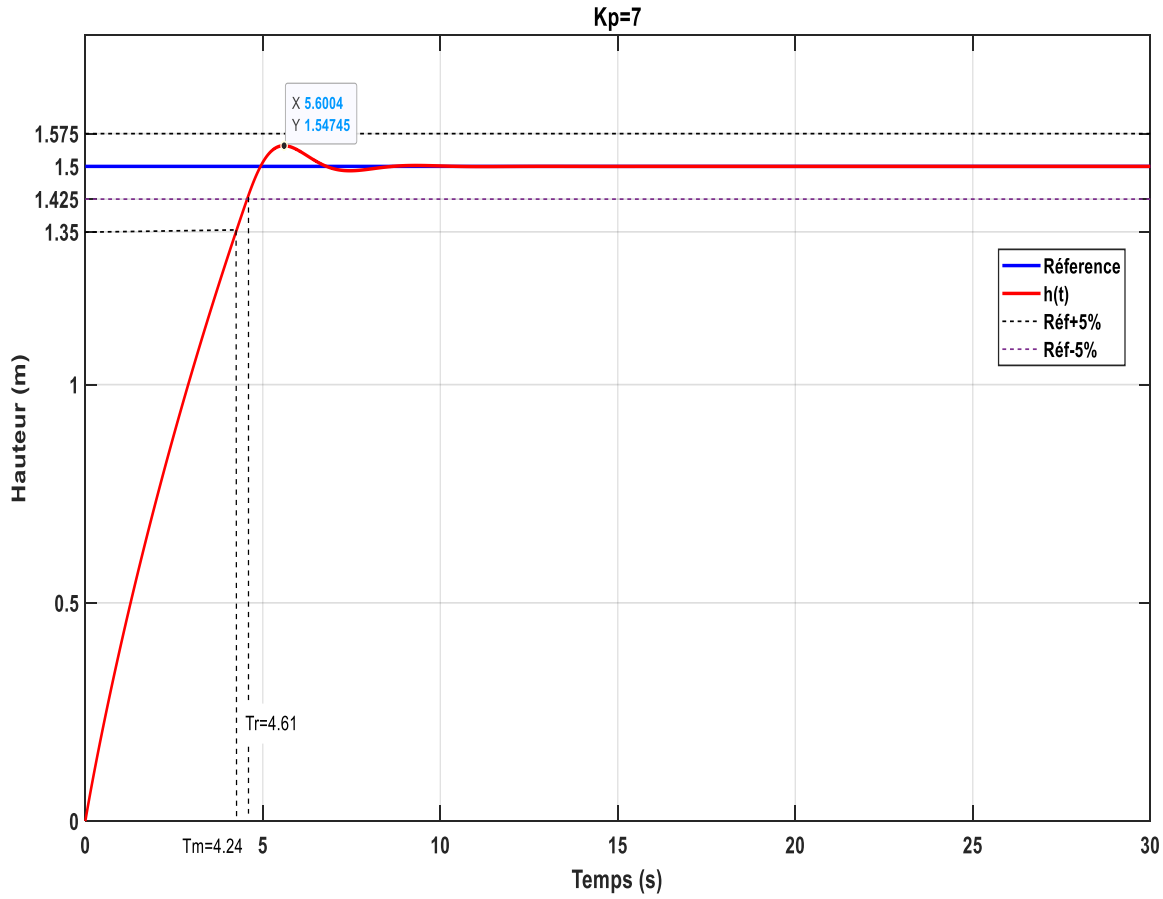


Figure III. 21 : Effet de l'augmentation du gain proportionnel $\boldsymbol{K_p}$.

- **Effet de $\boldsymbol{K_i}$:**

D'après la figureIII.22, on observe que l'augmentation de K_i peut entraîner un dépassement important, induire des oscillations, ou nuire à la stabilité si mal ajustée. Cela montre l'impact du terme intégral sur la précision du système.

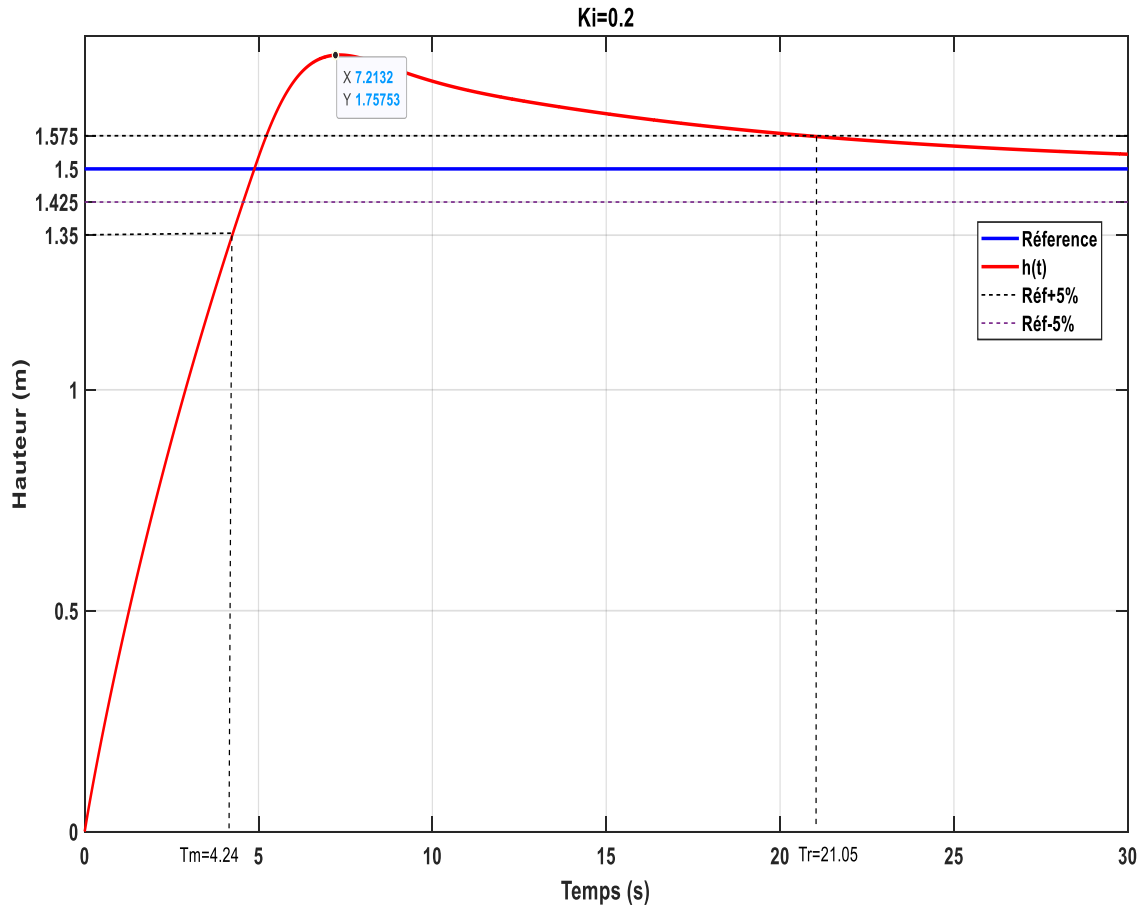


Figure III. 22 : Effet de l'augmentation du gain intégral K_i.

- **Effet de K_d:**

L'augmentation du gain K_d a permis d'améliorer l'amortissement du système en réduisant le dépassement et les oscillations, tout en maintenant une erreur statique nulle, comme il est montré dans la figure III.23.

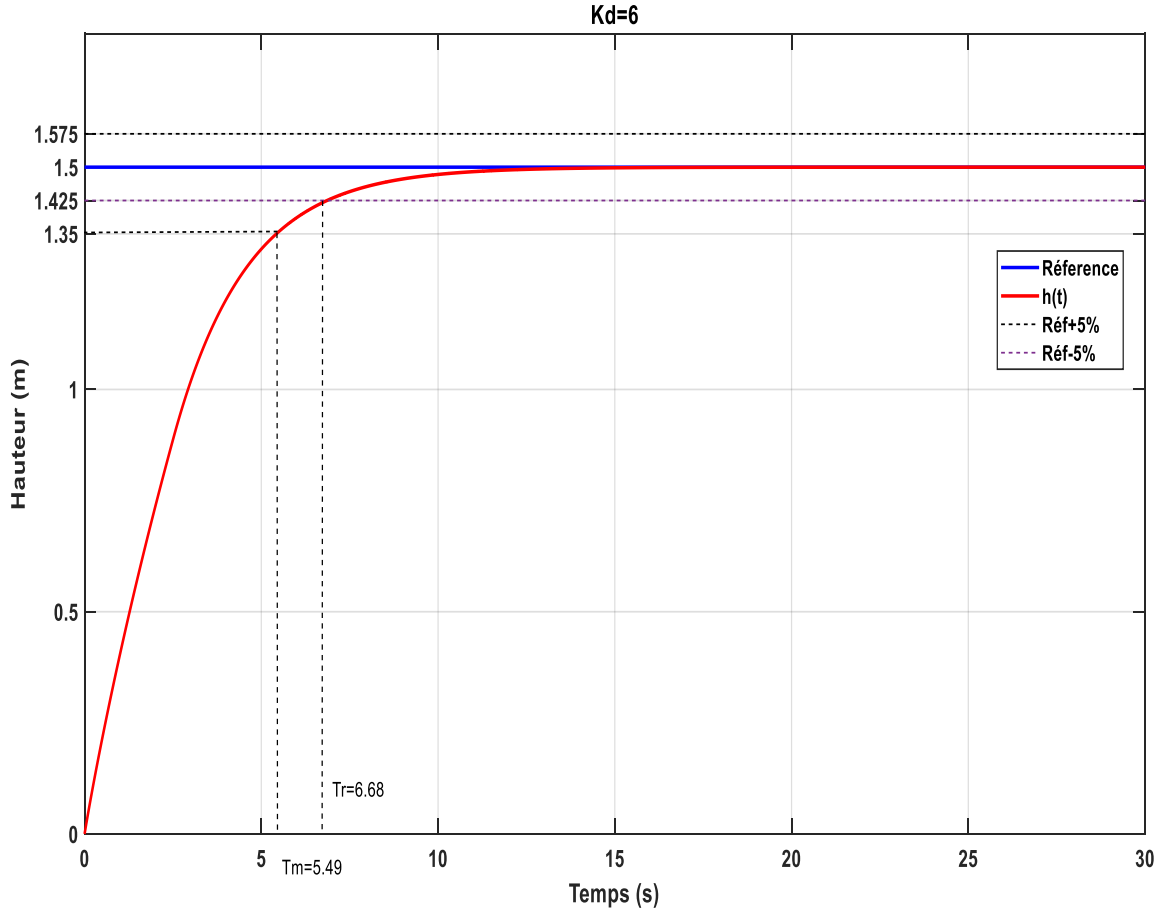


Figure III. 23 : Effet de l'augmentation du gain dérivé K_d.

III.6.2.3 Asservissement du sytème:

La réponse du système suit fidèlement la référence en escalier grâce à l'algorithme d'optimisation BES, qui permet de trouver les valeurs optimales du régulateur PID. Cela se traduit par une bonne rapidité, une erreur statique quasi nulle et un léger dépassement bien amorti, ce qui démontre la robustesse et l'efficacité du contrôleur face à des variations brusques de la consigne.

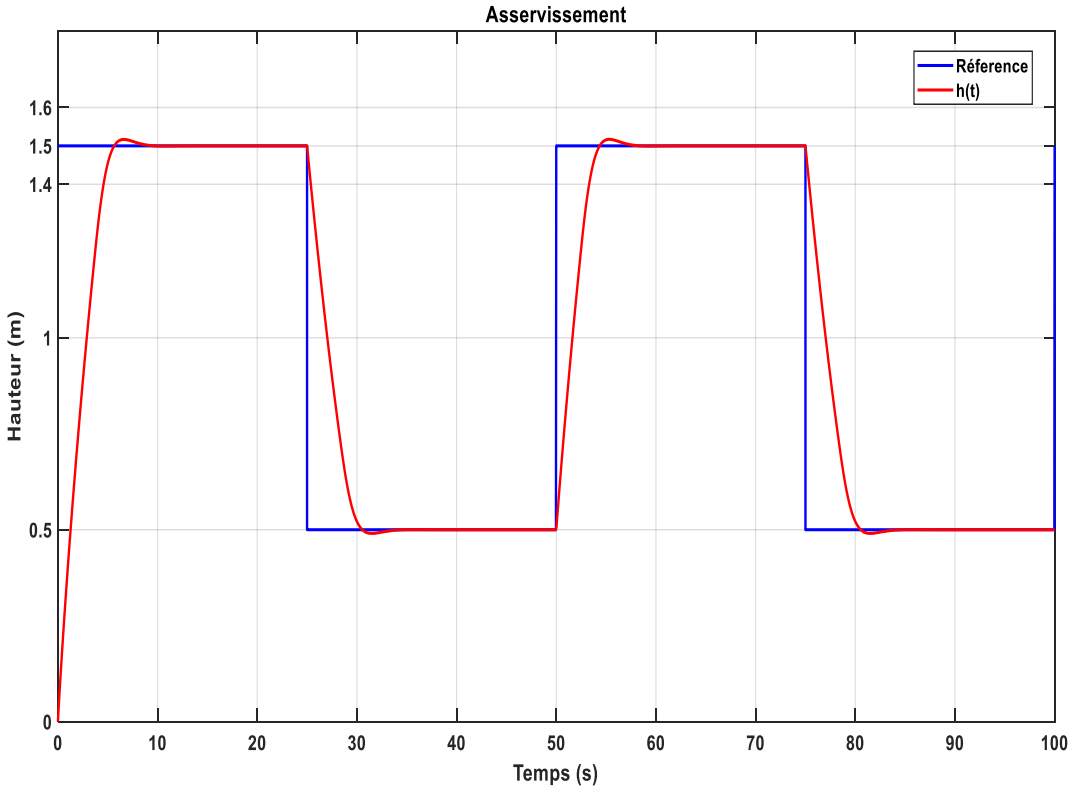


Figure III. 24 : Asservissement global du système avec gains optimisés.

III.6.2.4 Test de robustesse du régulateur PID

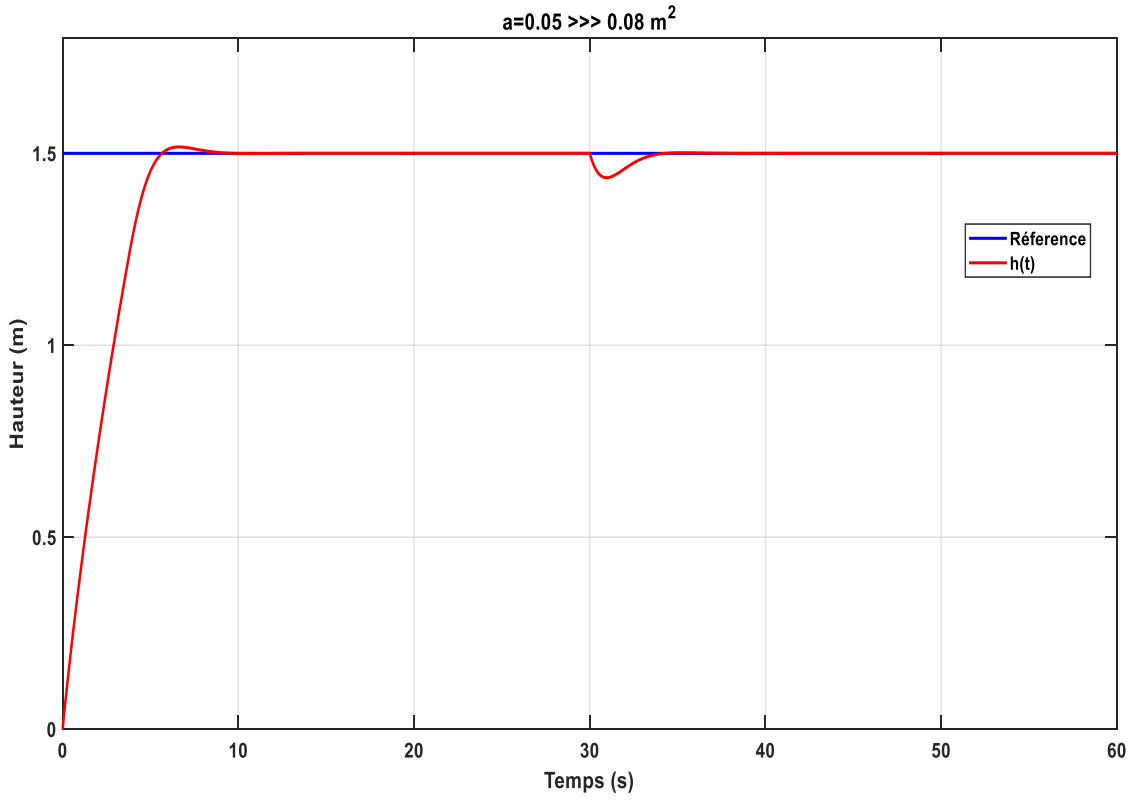
Afin d'évaluer la robustesse du régulateur PID optimisé par l'algorithme Bald Eagle Search (BES), deux tests de perturbation ont été réalisés en modifiant la section de l'orifice de sortie pendant le fonctionnement du système, tout en maintenant une consigne constante de hauteur $h = 1.5\, m$.

- Premier scénario (figure III.25a) : à $t = 30$ s, la section passe de $0.05\, m^2$ à $0.08\, m^2$, ce qui augmente le débit de sortie. Cette perturbation engendre une baisse temporaire du niveau d'eau. Le régulateur détecte rapidement l'écart et ajuste la commande de la vanne proportionnelle, permettant au niveau $h(t)$ de revenir à sa consigne en quelques secondes, sans provoquer d'oscillations majeures. Cela illustre une bonne réactivité en cas de perte accrue.
- Deuxième scénario (figure III.25b) : à $t = 30$ s, la section est réduite à $0.02\, m^2$ diminuant fortement le débit de sortie. Ce changement provoque une accumulation de liquide dans le réservoir, d'où un dépassement transitoire de la consigne. Néanmoins, le

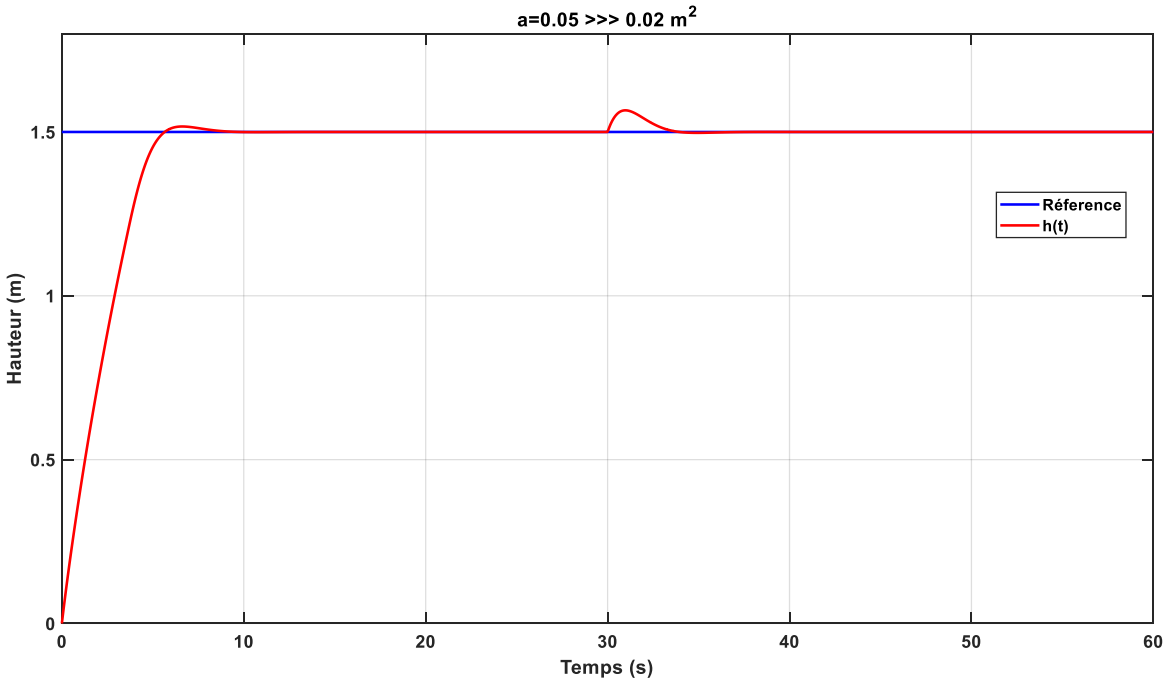
régulateur corrige rapidement la commande pour réduire le débit d'entrée, ramenant ainsi le système à l'équilibre avec un temps de stabilisation court et sans instabilité persistante.

Ces deux scénarios montrent que le régulateur PID, réglé automatiquement via l'algorithme BES, possède une robustesse satisfaisante face à des perturbations significatives du système. Il est capable de :

- détecter rapidement les écarts,
- corriger efficacement la commande,
- maintenir la stabilité du système sans oscillations prolongées,
- garantir un retour rapide à la consigne.



(a) : a = 0.08m^2



(b) : $\mathbf{a = 0.02\ m^2}$

Figure III. 25 : Réponse du système à une perturbation : variation de la section de l'orifice a

III.7 Conclusion

Les simulations réalisées dans ce chapitre ont permis de valider la conception du système de régulation proposé, tant dans sa version TOR que dans sa version optimisée via PID. L'implémentation sur Proteus a démontré la cohérence du comportement attendu en environnement discret, tandis que les modèles sous Simulink ont permis une analyse plus fine de la dynamique du système.

L'utilisation de l'algorithme BES pour l'optimisation des gains PID a clairement amélioré les performances du système, en réduisant le dépassement, le temps de stabilisation et l'erreur en régime permanent. Les tests réalisés ont également mis en évidence l'influence individuelle des paramètres Kp, Ki et Kd sur la réponse du système.

Ces résultats confirment l'intérêt de l'optimisation pour concevoir des systèmes de contrôle robustes et performants, ouvrant ainsi la voie à des implémentations physiques plus fiables et intelligentes.

Conclusion Générale

Conclusion générale

Au terme de ce travail, l'objectif principal a été atteint : concevoir et simuler un système de régulation du niveau de liquide dans un réservoir, alliant techniques de contrôle classiques (TOR et PID) et méthodes modernes d'intelligence artificielle. Les résultats de simulation ont montré que le système développé est capable de maintenir efficacement le niveau du liquide dans une plage définie, tout en améliorant des critères clés tels que le dépassement, le temps de stabilisation et l'erreur permanente.

Le choix d'utiliser deux environnements de simulation distincts s'est révélé pertinent : Proteus a permis de valider la commande TOR de manière simple et efficace, tandis que MATLAB/Simulink a offert un cadre puissant pour modéliser le comportement dynamique et appliquer un régulateur PID. Ce dernier a été optimisé à l'aide de l'algorithme Bald Eagle Search (BES), démontrant sa capacité à ajuster automatiquement les paramètres PID pour une performance optimale.

Ce travail ouvre plusieurs perspectives de recherche et d'application futures visant à améliorer davantage le système proposé. Tout d'abord, la mise en œuvre pratique du système sur une plateforme réelle constitue une étape essentielle pour valider expérimentalement les résultats obtenus par simulation, et pour identifier les éventuelles contraintes techniques liées à l'installation et au fonctionnement dans un environnement réel.

Par ailleurs, il serait pertinent d'étendre l'étude en comparant l'algorithme Bald Eagle Search (BES) à d'autres algorithmes métaheuristiques avancés tels que l'algorithme des abeilles artificielles (ABC), celui du loup gris (GWO) ou encore l'algorithme évolutionnaire différentiel (DE), afin d'identifier les points forts et les limites de chaque méthode en fonction du type de système à réguler.

D'un point de vue technologique, l'intégration d'un module de communication sans fil (comme le Wi-Fi ou le GSM) permettrait une surveillance et un contrôle à distance, ouvrant ainsi la voie vers un système intelligent basé sur les technologies de l'Internet des Objets (IoT). Ce système pourrait également être connecté à une plateforme cloud afin de stocker les données

et d'analyser les performances en continu grâce à des outils d'apprentissage automatique, dans le but d'anticiper les pannes et d'optimiser le comportement du système.

En outre, l'amélioration de l'interface homme-machine (IHM) pourrait faciliter l'utilisation du système en milieu industriel, en permettant une visualisation plus intuitive et efficace de l'état du processus.

Enfin, l'intégration de techniques de commande prédictive (MPC) ou d'apprentissage par renforcement (Reinforcement Learning) représenterait une alternative prometteuse au PID classique, notamment pour les systèmes complexes, multivariables ou opérants dans des environnements dynamiques et incertains.

Bibliographie

[1] B. ABDENNOUR, “Etude et simulation d’un régulateur de contrôle de niveau d’eau,” 2022, *university center of abdalhafid boussouf-MILA*.

[2] “Labview et Arduino.” Accessed: Jun. 23, 2025. [Online]. Available: http://bts2m.free.fr/TP_Labview/Labview_Arduino.html

[3] “Cours : Régulation industrielle.” Accessed: Jun. 22, 2025. [Online]. Available: http://elearning.univ-dbkm.dz/course/view.php?id=10788#section-3

[4] “Algorithme de contrôle PID et son application dans la production de cellules de charge - Microtess.” Accessed: May 12, 2025. [Online]. Available: https://www.microtess.com/fr/pid-control-algorithm/

[5] “Contrôleur PID expliqué- Wattco.” Accessed: Jun. 19, 2025. [Online]. Available: https://www.wattco.com/fr/2024/07/controleur-pid-explique/

[6] *Patrick Prouvost-Automatique - Contrôle Et Régulation - Cours Et Exercices Corrigés*. Accessed: May 11, 2025. [Online]. Available: https://www.scribd.com/document/574081544/Patrick-Prouvost-Automatique-Controle-Et-Regulation-Cours-Et-Exercices-Corriges

[7] E. Bonabeau, E. Bonabeau, and G. Theraulaz, *Intelligence collective*. in Systèmes Complexes. Paris: Hermès, 1994.

[8] A. Mohammed Belhadj and A. Keddar, “Régulation PID, étude et applications sur un système de température,” 2016.

[9] “Régulateur PID : les informations les plus importantes, vidéo comprise.” Accessed: Jun. 23, 2025. [Online]. Available: https://www.jumo.fr/web/services/faq/controller/pid-controller

[10] “Advanced Matlab/Octave Tutorial.” Accessed: Jun. 19, 2025. [Online]. Available: https://romi-lab.github.io/simulation/advanced/Control.html

[11] T. Nagy, F. Enyedi, E. Haaz, D. Fozer, A. J. Toth, and P. Mizsey, “Flexible and efficient solution for control problems of chemical laboratories,” *Comput. Aided Chem. Eng.*, vol. 46, pp. 1819–1824, Jan. 2019, doi: 10.1016/B978-0-12-818634-3.50304-0.

[12] “What Is The Ziegler-Nichols Method For PID Tuning? - IndMALL Automation.” Accessed: Jun. 20, 2025. [Online]. Available: https://www.indmall.in/faq/what-is-the-ziegler-nichols-method-for-pid-tuning/

[13] K. Ogata, “Modern conrol engineering,” p. 978, 2010.

[14] “Chapitre 3 Les Régulateurs Standards P, I, D, PI, PD Et PID | PDF | Électrotechnique | Ingénierie informatique.” Accessed: Jun. 24, 2025. [Online]. Available: https://www.scribd.com/document/607304006/Chapitre-3-Les-Regulateurs-Standards-P-I-D-PI-PD-Et-PID

[15] “Interface, AZprocede simulation de procédés temps réel sur PC.” Accessed: Jun. 24, 2025. [Online]. Available: https://www.azprocede.fr/aidehtml/regulateur.html

[16] M. A. El-Shorbagy, A. Bouaouda, H. A. Nabwey, L. Abualigah, and F. A. Hashim, "Bald eagle search algorithm: a comprehensive review with its variants and applications," *Syst. Sci. Control Eng.*, vol. 12, no. 1, p. 2385310, 2024, doi: 10.1080/21642583.2024.2385310/ASSET/7040441C-277B-4D7D-BEED-1598148E47B2/ASSETS/GRAPHIC/TSSC_A_2385310_F0017_OC.JPG.

[17] "UNO R3 | Arduino Documentation." Accessed: May 13, 2025. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3/

[18] "Arduino UNO Pinout | Explore Components | Jameco Electronics." Accessed: Jun. 19, 2025. [Online]. Available: https://www.jameco.com/Jameco/workshop/CircuitNotes/CN-Arduino-uno.html

[19] B. Katia and C. Siham, "Conception et réalisation d'un suiveur de soleil à base d'une carte Arduino UNO," 2017, *Université Mouloud Mammeri*.

[20] I. Malha, "Conception et réalisation d'un système domotique par GSM," 2017, *Université Mouloud Mammeri*.

[21] "Ultrasonic Sensor [HC-SR04] basics ⋆ EmbeTronicX." Accessed: Jun. 19, 2025. [Online]. Available: https://embetronicx.com/tutorials/tech_devices/ultrasonic_sensor/

[22] "Interface a 16x2 LCD Display with evive - STEMpedia Education." Accessed: Jun. 19, 2025. [Online]. Available: https://ai.thestempedia.com/docs/evive/evive-tutorials/how-to-interface-16-x-2-lcd-with-evive/

[23] E. DE La Recherche Scientifique, R. par, and B. Ahmed Benadjina Brahim Soutenu le Jury, "Réalisation d'un système de contrôle automatique de l'éclairage.," 2015, Accessed: Jun. 16, 2025. [Online]. Available: http://dspace.univ-tiaret.dz:80/handle/123456789/3145

[24] "Pompe à Eau 5V pour Arduino." Accessed: Jun. 19, 2025. [Online]. Available: https://micromaroc.com/products/pompe-5v

[25] "Les pompes à eau." Accessed: May 21, 2025. [Online]. Available: https://www.robotique.site/tutoriel/les-pompes-a-eau/

[26] A. S. Ismailov and Z. B. Jo'Rayev, "Study of arduino microcontroller board," *Sci. Educ.*, vol. 3, no. 3, pp. 172–179, 2022.

[27] "Introduction to Arduino IDE - The Engineering Projects." Accessed: Jun. 19, 2025. [Online]. Available: https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html

[28] E. A. Brewer, C. N. Dellarocas, A. Colbrook, and W. E. Weihl, "Proteus: A high-performance parallel-architecture simulator," *Acm Sigmetrics Perform. Eval. Rev.*, vol. 20, no. 1, pp. 247–248, 1992.

[29] "Proteus 8.17 - Télécharger pour PC Gratuit." Accessed: Jun. 19, 2025. [Online]. Available: https://proteus.fr.malavida.com/windows/

[30] "Simulink - Simulation and Model-Based Design - MATLAB." Accessed: Jun. 20, 2025. [Online]. Available: https://www.mathworks.com/products/simulink.html

[31] "Control Tutorials for MATLAB and Simulink - Simulink Basics Tutorial." Accessed: Jun. 20, 2025. [Online]. Available: https://ctms.engin.umich.edu/CTMS/index.php?aux=Basics_Simulink

[32] “What Are the Differences Between Simulink and Matlab | Skill-Lync Blogs.” Accessed: Jun. 24, 2025. [Online]. Available: https://skill-lync.com/blogs/what-are-the-difference-between-simulink-and-matlab