الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي والبحث العلمي

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

جامعة 20 اوث 1955-سكيكدة

UNIVERSITE 20 AOUT 1955-SKIKDA

Faculté des Sciences

Département d'informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme de Master

Filière : Informatique

Spécialité : Réseaux et systèmes distribués

Thème

Utilisation de l'Apprentissage par Renforcement pour l'équilibrage d'une plateforme robotique

Présenté par :

- **Hamaili Ahmed-Imad**

Devant le jury composé de :

…….…….. Président

…….. Encadrant

…….. Examinateur

Année universitaire : 2022/2023

Remerciements

Je remercie, du plus profond de mon cœur, Dieu le tout puissant de m'avoir donné le courage et la volonté d'achever ce travail.

A mes chers parents, pour leur confiance, leurs encouragements et pour leurs sacrifices durant toute la vie, je souhaite que ce travail soit le fruit de leurs efforts…

Au Professeur Boutine rachid, enseignant au département d'informatique, pour avoir accepté de diriger ce travail, en me faisant bénéficier de son expérience, ses conseils et son aide.

J'ai eu la chance et le plaisir d'effectuer ce travail de recherche. Je tiens à vous exprimer mon profond respect et ma gratitude.

Aux membres du jury, pour le grand honneur qu'il nous a fait en acceptant de juger ce travail, je souhaite que cette dissertation soit le témoignage de ma reconnaissance et de mon profond respect

Je remercie aussi toutes les personnes qui ont contribué pour me transmettre le savoir scientifique durant toute la durée de nos études universitaires.

À ***tous ceux qui, de près ou de loin, ont contribué à la finalisation de ce travail et tous ceux qui ont souhaité me voir arriver à ce stade.***

Dédicace

Je tiens à témoigner toute ma reconnaissance aux personnes suivantes :

Mes parents : pour les sacrifices déployés à mes égards ;

A ma famille ;

A mes amis ;

A mon encadreur : Mr Boutine rachid pour m'avoir aidé dans mon travail ;

A tous mes chers collègues de la promotion pour les années qu'on a passées ensemble.

A tous ceux et celles qui m'ont aidé de près ou de loin.

Liste de figure

Chapitre 01: Etat de l'art

Liste de figure

Liste de figure

AG : Algorithme génétique

AI : Artificial intelligence (Intelligence artificielle)

ANN : Artificial Neural Network

CC : Courant Continu

DL : Deep Learning (Apprentissage profond)

DNN : Deep Neural Network

DQN : Deep-q-Network

I2C : Inter Integrated Circuit Bus

HIGH : Haut

IDE : Integrated Development Environment

LQR : Linear Quadratic Regulator

LOW : Bas

MDP : Markov Decision Process

ML : Machine Learning (Apprentissage automatique)

OUTPUT : Sortie.

PID : Proportionnel-Intégral-Dérivé

PG : Policy Gradient

SCL : Serial Clock Line

SDA : Serial Data Line

PWM : Pulse Width Modulation

TD : Temporal difference

UART : *Universal Asynchronous Receiver Transmitter*

Sommaire

Table des matières

Sommaire

Sommaire

Sommaire

Introduction générale

Dans le domaine de la robotique, il est fondamental de trouver une solution adéquate pour l'équilibre d'un robot. Les robots auto-balancés sont des systèmes qui peuvent maintenir leur équilibre en utilisant des capteurs de mouvement et des algorithmes de contrôle. Néanmoins la mise au point de ces algorithmes peut s'avérer très complexes et nécessite souvent l'utilisation de l'apprentissage par renforcement, une technique de Machine Learning (Apprentissage automatique) qui pourrait se révéler très utile par sa compétence à s'adapter aux différentes situations.

Le travail présenté concerne la conception et l'implémentation d'un robot qui utilise l'apprentissage par renforcement, en particulier l'algorithme Q-learning, pour maintenir son équilibre. L'objectif est de démontrer l'efficacité de cet algorithme dans l'entraînement de tels robots et de mettre en évidence son impact sur l'amélioration de leurs performances et leur flexibilité.

La rédaction du document, est divisée en quatre principaux chapitres :

Dans le premier chapitre, nous présenterons les définitions des méthodes d'apprentissage automatique. Ensuite, nous définirons le concept d'équilibre. Enfin, nous aborderons quatre techniques spécifiques qui permettraient d'assurer l'équilibre d'un robot.

Dans le deuxième chapitre, nous accorderons une attention particulière au Markov Decision Process (MDP), car il joue un rôle essentiel dans la compréhension des algorithmes d'apprentissage par renforcement. Nous présenterons ensuite en détail les approches du Q-learning, du Deep Q-Network (DQN) et du SARSA. Enfin, nous aborderons le sujet du pendule inversé.

Dans le troisième chapitre, nous nous concentrerons sur la conception d'un robot équilibré en mettant l'accent sur les aspects matériels et logiciels. Nous examinerons en détail l'architecture du robot, à la fois sur le plan logiciel et matériel, en utilisant des organigrammes pour fournir une représentation visuelle de son fonctionnement.

Dans le quatrième chapitre, nous aborderons les différentes composantes du robot, le langage, l'environnement de simulation ainsi que l'IDE qu'on a utilisé. Ensuite, nous présenterons les résultats obtenus à partir de notre expérience, en mettant en évidence les améliorations qu'on a pu apportées.

Introduction générale

Problématique :

Bien que différentes techniques d'équilibrage aient été développées pour l'équilibrage des robots à deux roues, elles peuvent présenter des limites lorsqu'elles sont soumises à des perturbations externes. Les perturbations telles que des surfaces inégales, des obstacles imprévus ou des mouvements brusques peuvent compromettre la stabilité des robots, limitant ainsi leur capacité à fonctionner de manière fiable et efficace.

Afin de relever ce défi, il est nécessaire d'explorer une approche novatrice qui combine un algorithme d'apprentissage capable d'apprendre et d'ajuster le comportement du robot en fonction de son environnement, avec l'utilisation de techniques d'équilibrage existantes. L'idée est de permettre au robot d'acquérir une stabilité robuste et adaptative, en s'appuyant à la fois sur les connaissances préexistantes en matière d'équilibrage et sur les capacités d'apprentissage et d'adaptation de l'algorithme d'apprentissage automatique.

Par conséquent, la contribution de ce mémoire est de combiner un algorithme d'apprentissage par renforcement avec l'une des techniques d'équilibrage existantes, afin de garantir une stabilité robuste dans toutes les circonstances.

Chapitre 01 :
Etat de l’art

1.1 Introduction :

Dans ce chapitre, nous allons commencer par définir l'apprentissage automatique ainsi que ses différentes méthodes. Ensuite, nous aborderons la notion d'équilibre et expliquerons comment un robot peut s'équilibrer. Enfin, nous ferons l'état de l'art des quatre techniques qui pourraient permettre la stabilité d'un robot : le PID, la logique floue, les algorithmes génétiques et l'apprentissage par renforcement. Pour chacune de ces techniques, nous décrirons leur fonctionnement, donner leur résultats ainsi que leur avantages et inconvénients.

1.2 L'apprentissage automatique :

L'apprentissage automatique ou Machine Learning en anglais : Est une branche de l'intelligence artificielle où « l'intelligence », elle a été définie, par Arthur Samuel (un pionnier en matière d'intelligence artificielle) comme étant un domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmé [1] , et été définie par Tom Mitchell comme étant un programme informatique qui apprend à partir de l'expérience E par rapport à une classe de tâches T et une mesure de performance P, si sa performance dans les tâches de T, mesurée par P, s'améliore avec l'expérience E. [2]

1.3 Type d'apprentissage automatique :

Il existe différents types de modèles d'apprentissage automatique, chacun utilisant des techniques algorithmiques spécifiques. Ils sont présentés dans la **Figure 1.1**. En fonction des données traitées et des objectifs recherchés, il est possible d'utiliser l'un des quatre principaux modèles d'apprentissage :

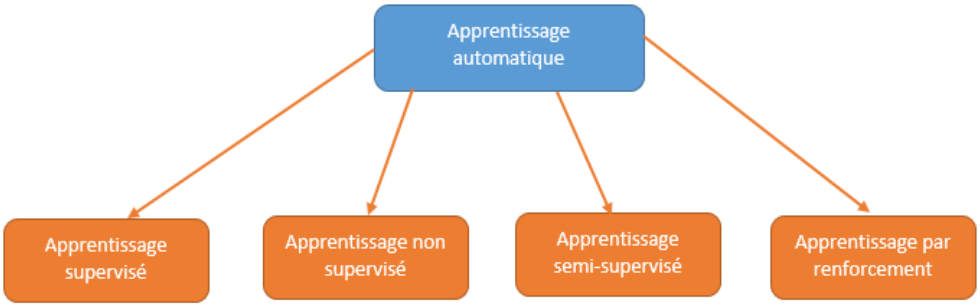


Figure 1.1 Types d'apprentissage automatique.

1.3.1 Apprentissage supervisé :

L'apprentissage supervisé est une technique d'apprentissage automatique ou un modèle est entrainé sur un ensemble de données étiquetées. Ces étiquettes permettent au modèle de comprendre quelle sortie est attendue pour une entrée donnée. Le modèle est entrainé en ajustant ses paramètres pour minimiser l'erreur entre les sorties prédites et les étiquettes réelles.

Plus formellement, étant donné un ensemble de données D, décrit par un ensemble de caractéristiques X, un algorithme d'apprentissage supervisé va trouver une **fonction de mapping entre les variables prédictives en entrée X et la variable à prédire Y**. la fonction de mapping décrivant la relation entre X et Y s'appelle un **modèle de prédiction**. **F(x) -> Y.** [3]

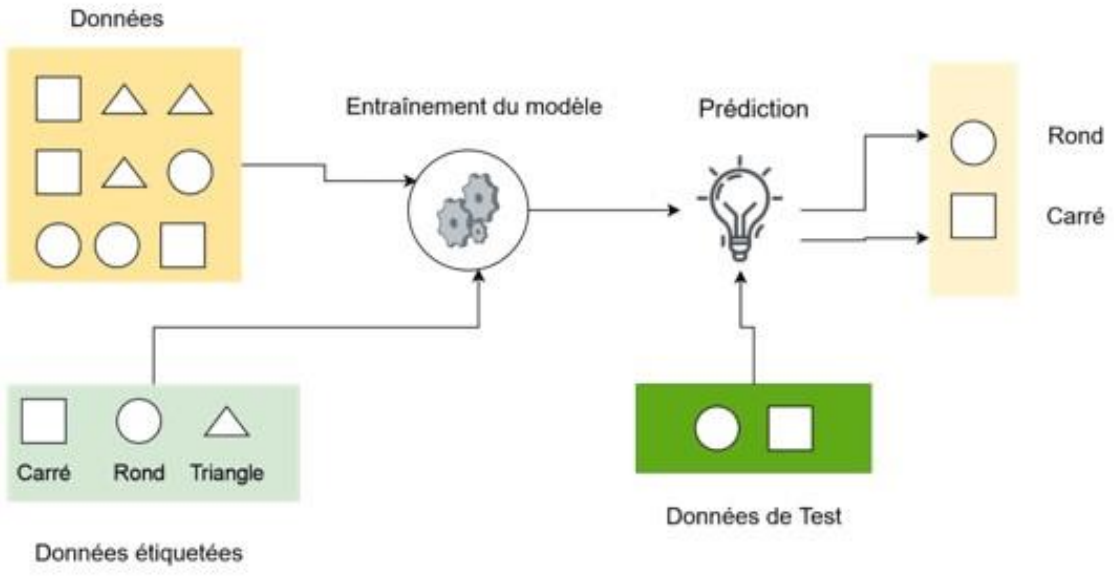


Figure 2.2 concept apprentissage supervisé [4].

1.3.2 Apprentissage non-supervisé :

C'est un type d'apprentissage automatique qui recherche des motifs non détectés dans un ensemble de données sans étiquettes préexistantes et avec un minimum de supervision humaine [5]. Contrairement à l'apprentissage supervisé, où le modèle est entraîné à prédire des étiquettes ou des réponses spécifiques, l'apprentissage non-supervisé cherche à découvrir des structures et des relations cachées dans les données. Ces algorithmes la sont souvent utilisés pour la segmentation de données, la réduction de dimensionnalité, la détection d'anomalies,

la classification automatique, la recommandation de produits, et bien d'autres applications.

L'apprentissage non supervisé prend uniquement des données sans label (pas de variable à prédire **Y**) **F(X) -> X'** (X représente la donnée et X' la donnée modifié).Un algorithme de ce type d'apprentissage, va essayer de trouver des patterns ou une structuration dans les données.[6]

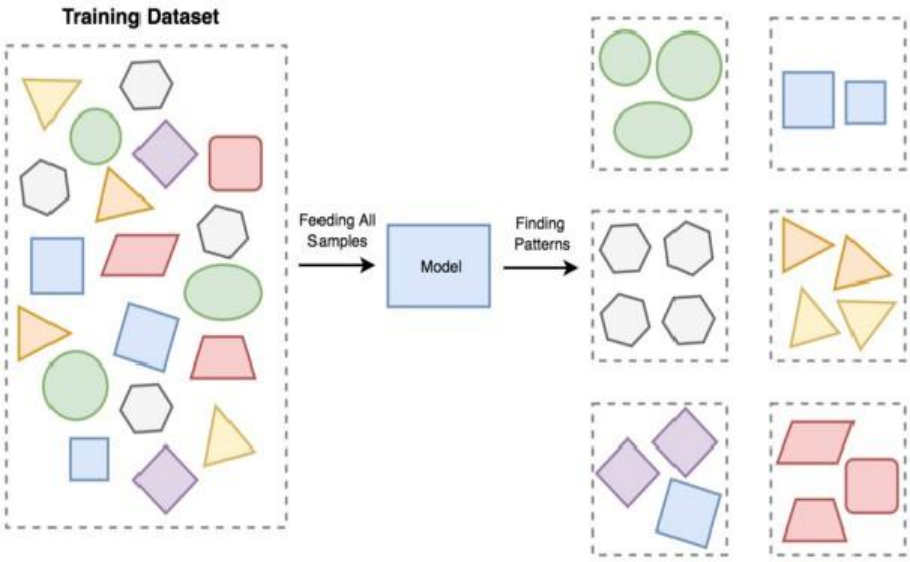


Figure 3.3 concept apprentissage supervisé [7].

1.3.3 Apprentissage semi-supervisé :

L'algorithme combine simultanément des données étiquetées et non-étiquetées au cours de la phase d'apprentissage et les différentes classes sont connues à l'avance. Dans ces conditions, la combinaison de deux catégories de données permet d'améliorer significativement la qualité de l'apprentissage notamment lorsque les jeux de données sont très grands où l'étiquetage manuel peut s'avérer fastidieux. Parmi les méthodes de ce mode d'apprentissage, on peut distinguer le "clustering" semi-supervisé et la classification semi-supervisée. Le but de "clustering " est de regrouper les instances les plus similaires entre elles en exploitant l'information apportée par les données étiquetées. Par contre, la classification utilise les données étiquetées pour séparer les instances des différentes classes. Elle affine la fonction de classification à l'aide des données non étiquetées.[8]

1.3.4 Apprentissage par renforcement :

L'apprentissage par renforcement est l'une des méthodes de l'apprentissage automatique. Il sert à décrire et résoudre des problèmes dans lesquels un agent apprend des stratégies pour maximiser les récompenses ou atteindre des objectifs spécifiques lors de son interaction avec l'environnement. L'apprentissage par renforcement peut être considéré comme un processus d'essais et d'erreurs, et il est composé de six éléments principaux : l'agent, l'environnement, l'état, l'action, la récompense et la politique.[9]

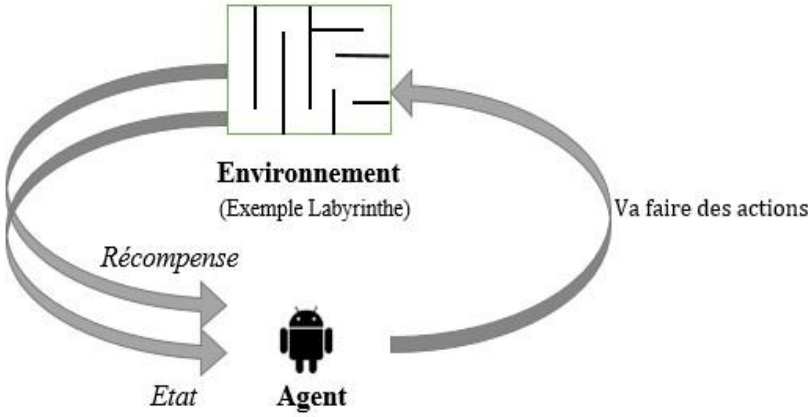


Figure 1.4 concept d'apprentissage par renforcement

1.4 Apprentissage profond (Deep Learning) :

Est un type d'apprentissage automatique (machine Learning) qui vise à former et à enseigner à l'ordinateur afin d'exercer des fonctions humaines telles que la distinction des objets visuels et l'identification des sons et des images. Au lieu d'organiser des données, l'apprentissage en profondeur définit ses propres paramètres de base, ce qui permet à la machine d'apprendre par elle-même, et l'intérêt actuel pour l'apprentissage en profondeur est en partie dû au bourdonnement qui entoure l'intelligence artificielle. Les techniques d'apprentissage en profondeur ont amélioré la capacité de classer, reconnaître, détecter, etc [10].

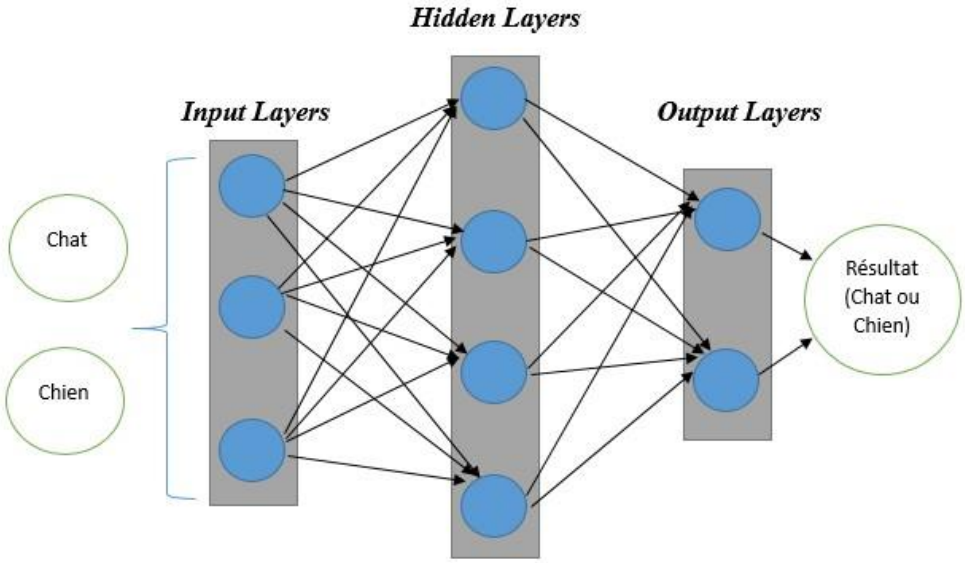


Figure 1.5 Illustration de l'apprentissage profond (Deep learning)

1.5 Définition de l'équilibre :

État de repos, position stable d'un système obtenus par l'égalité de deux forces, de deux poids qui s'opposent : Mettre les plateaux d'une balance en équilibre.[11] L'équilibre peut être statique, lorsque les forces sont équilibrées mais l'objet est immobile, ou dynamique, lorsque les forces sont équilibrées mais l'objet est en mouvement constant.

1.6 Principe de fonctionnement du robot auto équilibré :

Le robot auto-équilibré à deux roues fonctionne grâce à un système de contrôle sophistiqué qui permet de maintenir une position stable en tout temps. Pour y parvenir, il est nécessaire de bien comprendre les paramètres du système et de les intégrer dans un modèle de contrôle avancé. Ce modèle permet de surveiller en temps réel les mouvements du robot et de corriger automatiquement sa position pour éviter toute chute.

Son principe de fonctionnement est fondé sur le même concept que le pendule inversé. Dans les deux situations, l'objectif est de maintenir un objet en équilibre. Pour mieux comprendre, le principe du pendule inversé implique de garder les roues du robot situées sous le centre de masse de son châssis afin d'assurer sa stabilité. Si le robot commence à pencher en avant, la roue doit avancer pour ramener le centre de masse au-dessus d'elle. Si cette condition n'est pas respectée, le robot tombera.

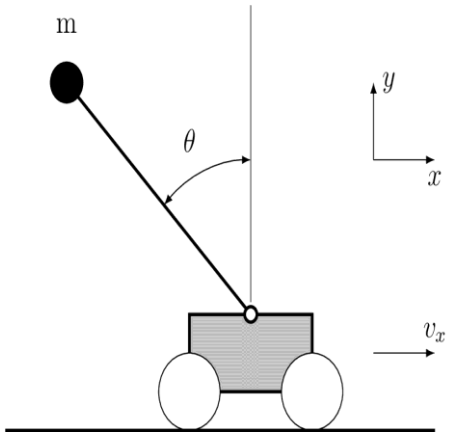


Figure 1.7 Principe de la pendule inversé [12]

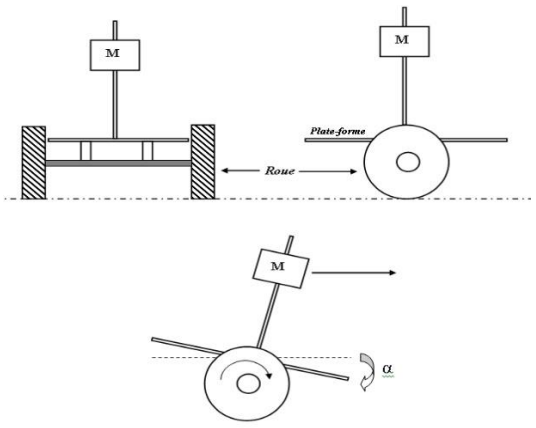


Figure 1.6 Principe d'un robot à deux roues [13]

1.7 Les techniques permettant l'équilibre d'une plateforme :

Il existe différentes techniques permettant la stabilité (l'équilibre) d'une plateforme à deux roues, on peut en citer :

1.7.1 Equilibrage d'un robot avec le PID :

1.7.1.1 Définition :

La commande PID (proportionnel-intégral-dérivé) est un mécanisme de régulation qui améliore les performances d'un système de contrôle en boucle fermée, également connu sous le nom d'asservissement. Ils sont utilisés depuis plusieurs décennies dans l'industrie pour des applications de contrôle de processus [14]. La raison de leur popularité généralisée réside dans la simplicité de conception et les bonnes performances, notamment un faible dépassement en pourcentage et un temps de stabilisation réduit pour les installations de processus lents [14].

1.7.1.2 Principe de fonctionnement :

Son principe de fonctionnement (PID) repose sur le calcul d'une commande en fonction des trois termes qui sont : (**P**) pour *proportionnel*, (**I**) pour *intégral* et (**D**) pour *dérivé*.

Le terme (**P**) a pour but de prendre en compte l'erreur actuelle entre la valeur mesurée et la valeur de consigne (la valeur mesurée c'est la valeur de référence à laquelle on souhaite maintenir le système dans notre cas de robot auto-équilibré, la valeur de consigne correspond à l'angle d'inclinaison souhaité pour le robot) multiplié par un coefficient proportionnel. Ce terme nous permet donc d'ajuster la commande en fonction de l'erreur actuelle.

Le terme (**I**) prend en compte l'erreur cumulée entre la valeur mesurée et la valeur consigne, multipliée par un coefficient intégral. Ce terme nous permet de corriger les erreurs de long terme, en accumulant les erreurs passées et en ajustant la commande en conséquence.

Le terme (**D**) prend en compte la variation de l'erreur entre la valeur mesurée et la valeur de consigne, multipliée par un coefficient dérivé. Ce terme permet de corriger les erreurs de court terme, en ajustant la commande en fonction de la vitesse à laquelle l'erreur change

En combinant ces trois termes, le PID calcule une commande de contrôle qui permet de maintenir le système à une valeur de consigne. Dans notre cas de robot auto-équilibré, le PID est utilisé pour ajuster la vitesse des roues et cela en fonction de l'angle d'inclinaison du robot.

Le régulateur PID est une simple implémentation de retour d'information. Il a la capacité d'éliminer la compensation de l'état d'équilibre grâce à l'action intégrale, et il peut anticiper le futur grâce à une action dérivée L'équation d'un régulateur PID est donnée comme suit :

$$R(t) = K_p.e(t) + K_i.\int_0^t e(t)dt + K_d.\frac{de(t)}{dt}$$

1.7.1.3 Schéma de régulateur PID :

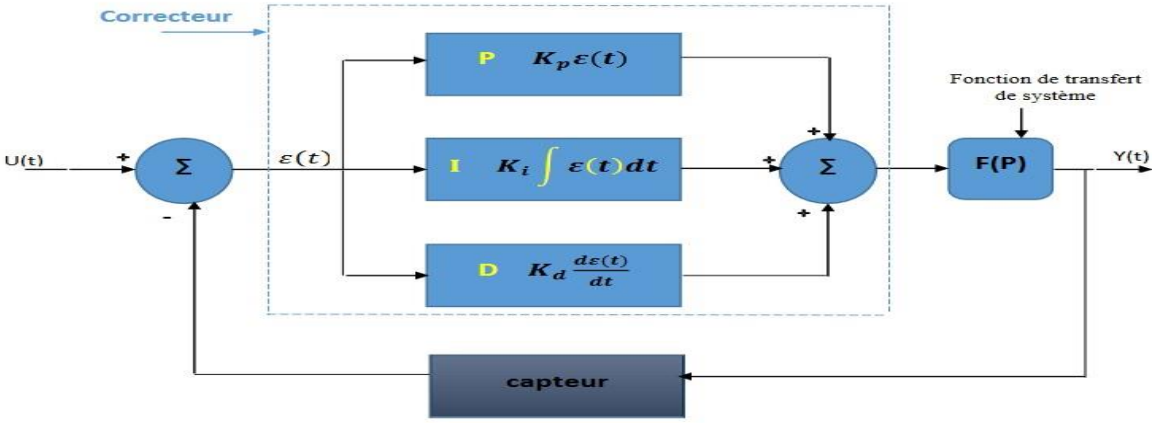


Figure 1.8 Schéma de régulateur PID

Kp : il s'agit du gain proportionnel ; **u(t) :** s'agit de la consigne

Ki : il s'agit du gain intégral ; **ε(t)** : **u(t) – y(t) l'erreur**

Kd : il s'agit du gain dérivé.

1.7.1.4 Régulation du PID :

Dans la pratique, chaque catégorie de systèmes à asservir est associée à un type spécifique de correcteur. Afin de faire un choix éclairé, il est important de comprendre les effets des différentes actions du régulateur PID : la proportionnelle, l'intégrale et la dérivée. Ces actions sont illustrées sur le schéma suivant :

1.7.1.4.1 Le régulateur propotionnel P :

Cette commande, elle est proportionnelle à l'erreur. Commande = Kp * erreur

Kp : c'est le coefficient de proportionnalité de l'erreur à régler de façons manuelle.

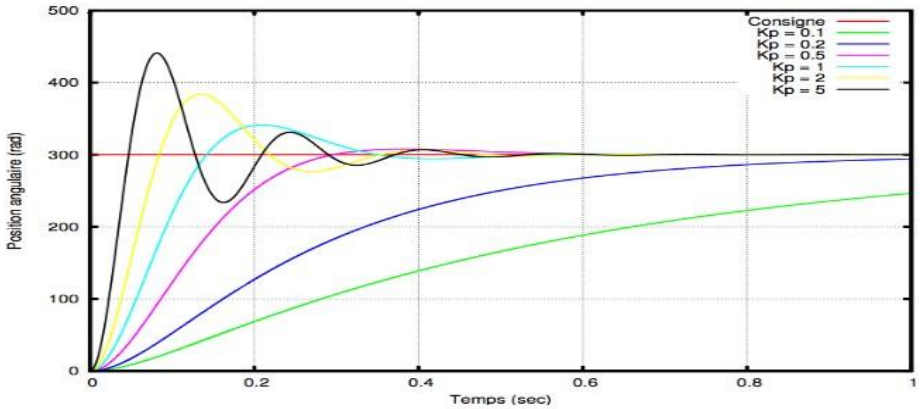


Figure 1.9 Schéma d'échelon d'un régulateur proportionnel. [15]

1.7.1.4.2 Le régulateur dérivé D :

Le régulateur dérivé détermine une valeur régulée en se basant sur la vitesse de variation de l'erreur plutôt que sur son amplitude, contrairement au régulateur P. C'est pourquoi il réagit de manière beaucoup plus rapide qu'un régulateur P. Même en présence d'une petite erreur, il générera une valeur régulée importante dès qu'il détecte une variation d'amplitude de l'erreur.

Kd : est le coefficient de proportionnalité de la variation de l'erreur.

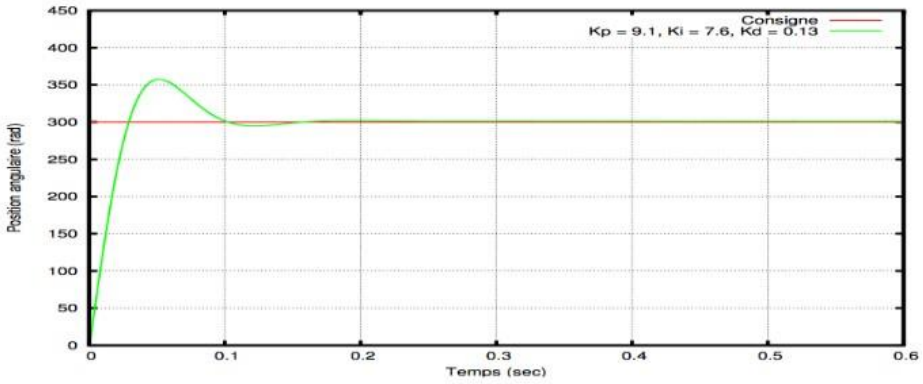


Figure 1.11 Schéma d'échelon d'un régulateur dérivé [15].

1.7.1.4.3 Le régulateur proportionnel intégral dérivé (PID) :

La commande englobe à la fois la proportionnalité de l'erreur, la somme des erreurs cumulées et la variation de l'erreur. Elle s'exprime mathématiquement par l'équation : Commande = Kp * erreur + Ki * la_somme_des_erreurs + Kd * (erreur - erreur_précédente). Pour appliquer le contrôle PID, il est nécessaire de mesurer régulièrement l'état du système à une fréquence d'échantillonnage donné afin de calculer l'erreur et de l'utiliser dans la formule.

On peut résumer tout ça selon cet algorithme PID (Piette, 2011)[16] :

Tous les x millisecondes, faire :

.Erreur = consigne - mesure;

.La_somme_des_erreurs += Erreur;

.Variation_erreur = Erreur – Erreur_précédente;

.Commande = Kp * erreur + Ki * la_somme_des_erreurs + Kd * variation_erreur;

.Erreur_précédente = erreur.

1.7.1.5 réglages des coefficients du PID :

Le réglage des coefficients Kp, Ki et Kd se font par une approche d'essais/erreurs. Cependant il est déconseillé d'essayer de régler les trois coefficients

en même temps, il y'a trop de combinaisons possibles et trouver un triplet performant relèverait de l'exploit. Et pour cela il faudrait y procéder étape par étape comme suit :

- En premier lieu, il est recommandé de configurer un régulateur proportionnel simple avec des coefficients Ki et Kd nuls. Par essais/erreurs, on essaye d'améliorer le temps de réponse du système en ajustant le coefficient Kp (on doit trouver la valeur qui permettra à notre système de se rapprocher très vite de la consigne et ce, en maintenant sa stabilité, en évitant les oscillations excessives) ;
- Une fois la première étape satisfaite c'est-à-dire le Kp est réglé, on peut passer au coefficient Ki. Ce dernier nous permettra d'annuler l'erreur finale du système et obtenir une réponse précise en peu de temps, tout en minimisant les oscillations apportées par l'intégrateur.
- Enfin, la dernière étape consiste à régler le coefficient Kd pour améliorer la stabilité du système et réduire les oscillations. Ce paramètre est important pour garantir un fonctionnement optimal et une réponse rapide du système.

La réponse type d'un procédé stable est la suivante :

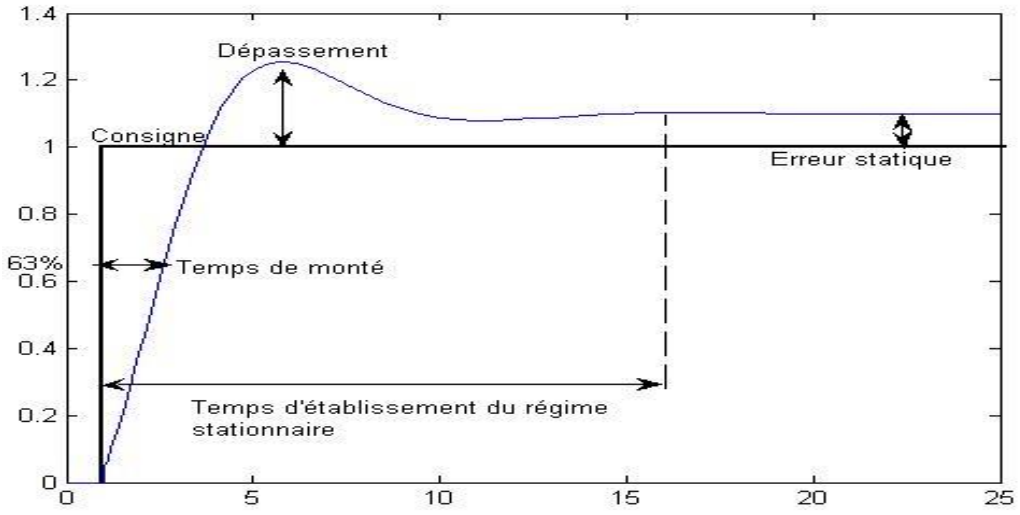


Figure 1.12 Réponse échelon PID[17]

1.7.1.6 L'utilisation du PID pour équilibrer un robot :

L'utilisation du PID pour équilibrer un robot à deux roues est une méthode couramment utilisé, pour se faire, on commence d'abord par mesuré l'angle d'inclinaison du robot à l'aide d'un gyroscope, Ensuite on calcule l'erreur en soustrayant l'angle mesuré de l'angle cible (c'est l'angle ou le robot tient en équilibre), après on utilise le PID pour calculer la commande de vitesse des roues en fonction de l'erreur, une fois les calculs terminé il ne reste plus qu'à appliquer cette commande de vitesse aux moteurs de roues pour ajuster la vitesse et maintenir le robot en équilibre.

1.7.1.7 Exemple d'équilibrage d'un robot en utilisant le PID:

Dans la **Figure 1.13**, la réponse impulsionnelle de la première itération du système est affichée, lors de l'utilisation d'une méthode de lecture de gyroscope à blocage. De toute évidence, le système est instable et ne se comporte pas conformément à la théorie. La raison de la divergence était liée au retard causé par le blocage mise en œuvre de la lecture gyroscopique[18].

Le blocage devient un problème pour la direction du robot. Afin d'augmenter la vitesse de pointe des moteurs pas à pas, le temps moyen de la boucle principale doit être augmenté, ce qui n'est pas compatible avec une lecture à chaque cycle de boucle. Une façon théorique de résoudre ce problème est de ne pas lire le capteur à chaque

itération de la boucle principale, mais plutôt d'attendre un certain temps spécifié avant de lire. Cependant, cela conduit à un autre problème, un retard accru du système qui conduit également à l'instabilité du système.[18]

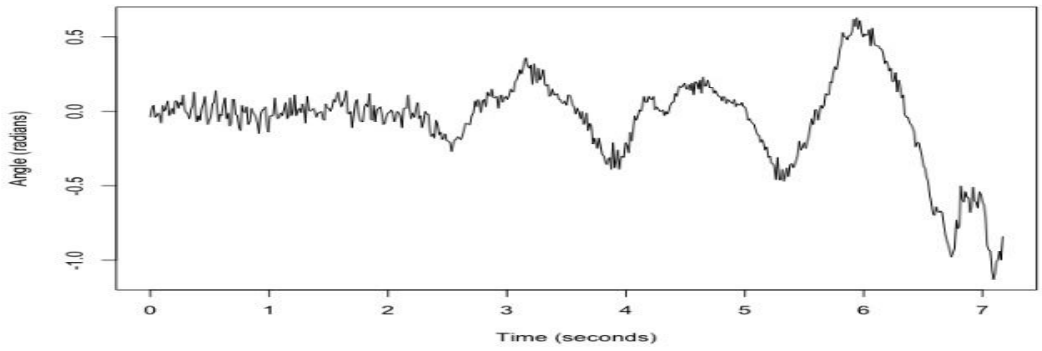


Figure 1.13 Parcelle de développement de système instable [18]

En mettant en œuvre une méthode de lecture de capteur non bloquante, le système a été stabilisé. Avec le PID-paramètres : Kp = 2,3, KI = 0 et KD = 1,4 que le système a pu gérer impulsions synthétiques – comme le montre la **Figure 1.14**.

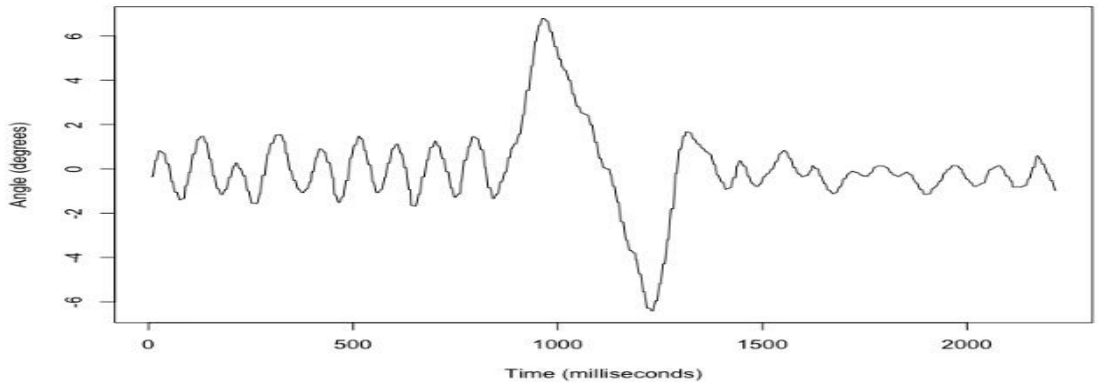


Figure 1.14 Tracé d'une réponse impulsionnelle synthétique pour le système stable|18]

1.7.1.8 Avantages du PID :

Voici quelques avantages de l'utilisation du PID pour l'équilibrage d'un robot à deux roues :

- Contrôle précis ;
- Stabilité ;
- Réponse rapide ;
- Simple à mettre en œuvre ;
- Faible coût.

1.7.1.9 Inconvénient du PID :

Bien que le PID soit un algorithme de contrôle populaire et largement utilisé en raison de sa simplicité et de son efficacité, il présente quelques inconvénients potentiels :

- Sa sensibilité aux perturbations ;
- Paramétrage complexe ;
- Peut nécessiter une mise à jour constante des paramètres pour maintenir une performance optimale.

1.7.2 Equilibrage d'un robot avec la logique floue :

1.7.2.1 Définition :

- Est une approche de l'informatique basée sur des « degrés de vérité » plutôt que sur la logique booléenne habituelle sur laquelle repose l'informatique moderne. Elle a été formulée pour la première fois par le Dr Lotfi Zadeh, de l'université de Californie à Berkeley. [19]

1.7.2.2 Différence entre la logique floue et la logique classique :

La logique classique se base sur le principe de non-contradiction et qui attribue une valeur de vérité précise à chaque proposition (soit vrai, soit faux : 1 ou 0) alors que la logique floue permet d'exprimer des nuances et des degrés de vérité. Pour cela la logique floue se propose de remplacer les variables booléennes par des variables flous.

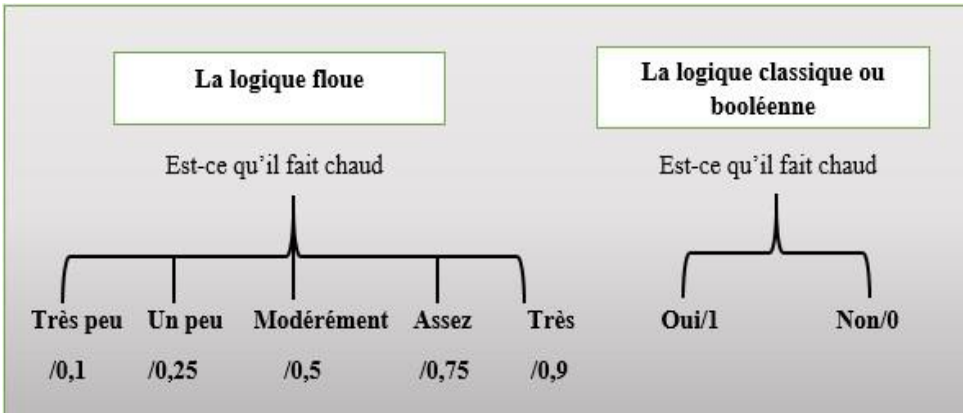


Figure 1.15 Différence entre la logique floue et la logique classique.

1.7.2.3 Principe de la logique floue :

Afin de mettre en évidence le principe de la logique floue, on présente deux exemples de représentation de la température, une en logique classique, et l'autre en logique floue la figure (1.16 et 1.17). Selon cette figure, en logique classique, une température de 22.5◦ est considérée comme "élevée".

En logique floue, une température de 22.5◦ appartient au groupe "moyenne" avec un degré d'appartenance de 0.167, et appartient au groupe "élevée" avec un degré d'appartenance de 0.75, (et au groupe "faible" avec un degré d'appartenance de 0) [20].

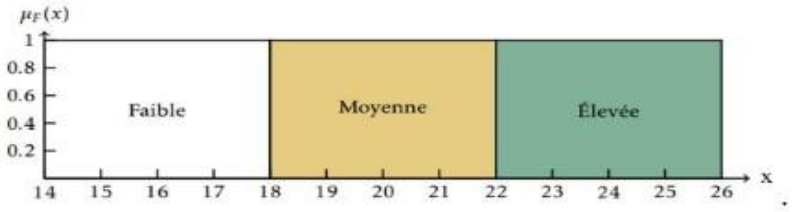


Figure 1.16 Représentation classique [20]

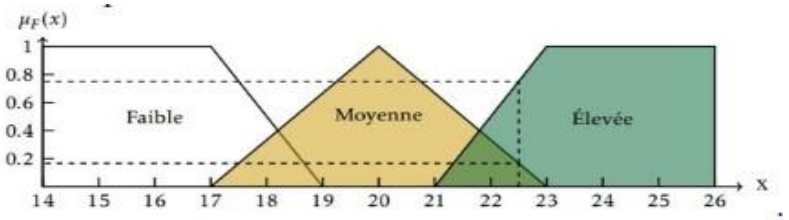


Figure 1.17 Représentation floue [20]

1.7.2.4 L'ensemble flou :

Est un type d'ensemble mathématique permettant la modélisation des phénomènes complexes qui ne peuvent pas être facilement décrits à l'aide de concepts binaires, elle permet alors de traiter ces données en utilisant des opérations de logique floue. Cette théorie permet d'exprimer l'idée d'un élément en tant que membre partiel d'un ensemble

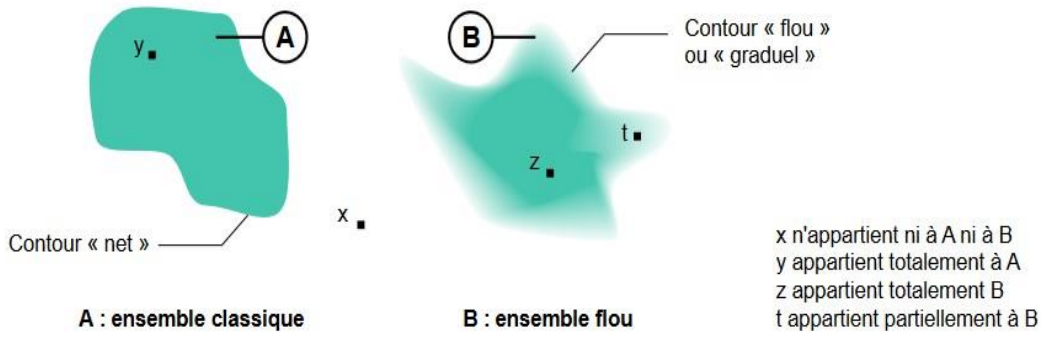


Figure 1.18 Comparaison entre la logique floue et la logique classique [21]

1.7.2.5 Opérateurs flous :

Ils permettent d'écrire des combinaisons logiques entre notions floues, c'est-à-dire de faire des calculs sur des degrés de vérité. Comme pour la logique classique, on peut définir des opérateurs ET, OU, négation. Exemple : Appartement Intéressant = Loyer Raisonnable ET Surface Suffisante. [21]

1.7.2.6 La fuzzification et defuzzification :

La fuzzification consiste a transformer une valeur numérique en degré d'appartenance flou par évaluation d'une fonction d'appartenance.[21]

La defuzzification, quant à elle, est la Transformation, après inférence, d'un ensemble flou d'une variable linguistique de sortie en valeur numérique.[21]

1.7.2.7 Exemple d'équilibrage d'un robot en utilisant la logique floue

Les données qui sont mesuré par le capteur sont entrées dans un système de contrôle qui utilise des règles floues pour déterminer les actions à entreprendre pour maintenir l'équilibre du robot, ces règles la sont des déclarations type : If-then qui associent des conditions floues à des actions floues. Par exemple, si l'inclinaison du robot est grande, alors une action floue appropriée serait d'augmenter la vitesse des roues[21].

Les valeurs floues sont des nombres qui représentent l'intensité de l'action à entreprendre. Par exemple si l'inclinaison du robot est «forte», alors l'action de « vitesse des roues » peut être représentée par une valeur floue élevée, ces valeurs floues sont ensuite agrégées et transformées en valeurs numériques précises qui

peuvent être utilisées pour contrôler les mouvements du robot. Voici une représentation de la table du robot [21]:

IF	AND	THEN
If the robot is leaning forward by a large degree	its speed is fast	slow down the robot to maintain balance
If the robot is leaning forward by a large degree	its speed is medium	slow down the robot slightly to maintain balance
If the robot is leaning forward by a small degree	its speed is fast	maintain the current speed to maintain balance
If the robot is leaning forward by a small degree	its speed is medium	maintain the current speed to maintain balance
If the robot is leaning forward by a small degree	its speed is slow	speed up the robot slightly to maintain balance
If the robot is leaning backward by a large degree	its speed is fast	slow down the robot significantly to maintain balance
If the robot is leaning backward by a large degree	its speed is medium	slow down the robot moderately to maintain balance
If the robot is leaning backward by a small degree	its speed is fast	speed up the robot slightly to maintain balance
If the robot is leaning backward by a small degree	its speed is medium	maintain the current speed to maintain balance
If the robot is leaning backward by a small degree	its speed is slow	speed up the robot slightly to maintain balance

Figure 1.19 Règles floues du robot[21]

Ces règles floues tiennent compte de l'angle et de la vitesse du robot, et déterminent la mesure appropriées à prendre pour maintenir l'équilibre. En combinant ces règles avec une logique flou contrôleur, le robot peut être contrôlé d'une manière qui s'adapte aux conditions changeantes et maintient l'équilibre même en présence d'incertitude. La méthode du centre de gravité est utilisée pour résoudre la valeur de sortie nette floue, comme indiqué dans l'équation [21] :

$$\eta = \frac{\sum \mu_i \cdot f(\mu_i)}{\sum f(\mu_i)}$$

Le modèle d'espace d'états implémenté dans l'environnement LabVIW est illustré à la **Figure 1.21** sous la forme d'un bloc "State-Space" de la palette "Control Design and Simulation", avec les matrices d'espace d'états A, B, C et D câblées à ses entrées. [21]

La matrice A représenté la dynamique du système en l'absence de toute entrée. Sa première ligne représente le taux de changement de la position angulaire du pendule, qui est lié à sa vitesse angulaire [21].

La matrice B représente la façon dont le système répond aux entrées. [21]

La matrice C représente la façon dont le système produit des sorties [21].

La matrice D représente le gain en régime permanent du système, qui est toujours nul pour un système à pendule inversé [21].

Ces matrices sont la notation standard utilisée pour représenter un système linéaire invariant dans le temps sous forme d'espace d'état et peuvent être utilisées pour simuler et contrôler le comportement d'un système de pendule inversé en utilisant une variété de techniques, telles que la rétroaction d'état, la conception d'observateur et contrôle flou [21].

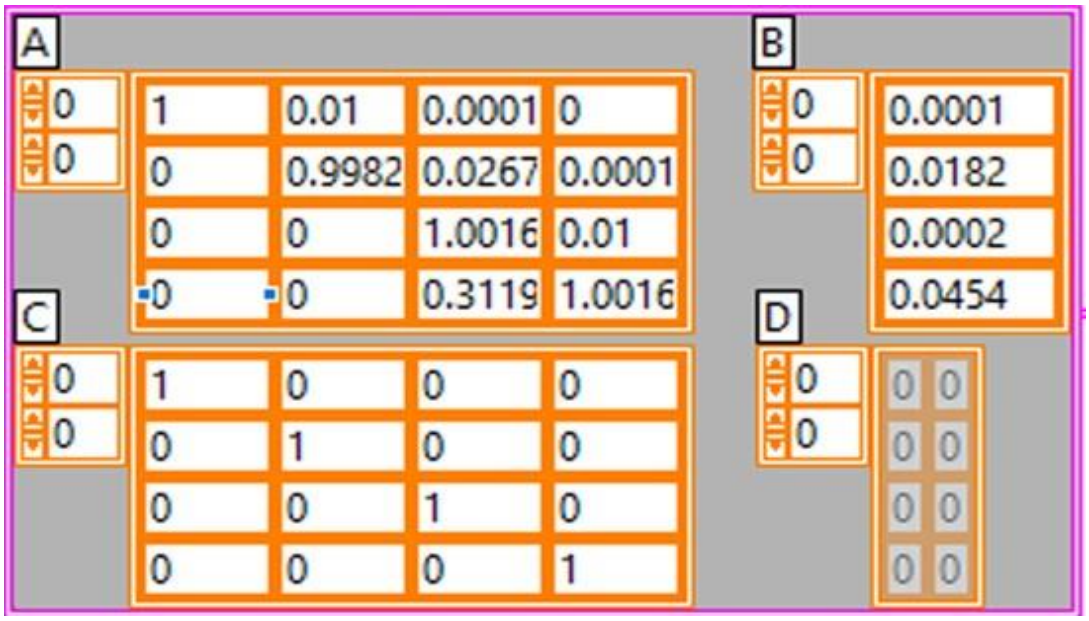


Figure 1.20 Modèle d'espace d'état du robot mobile auto-équilibré à deux roues [21]

Le panneau ci-dessous (**Figure 1.21**) montre le mouvement en temps réel du robot mobile à deux roues le long avec les paramètres de contrôle et l'angle de vue qui peuvent être modifies :

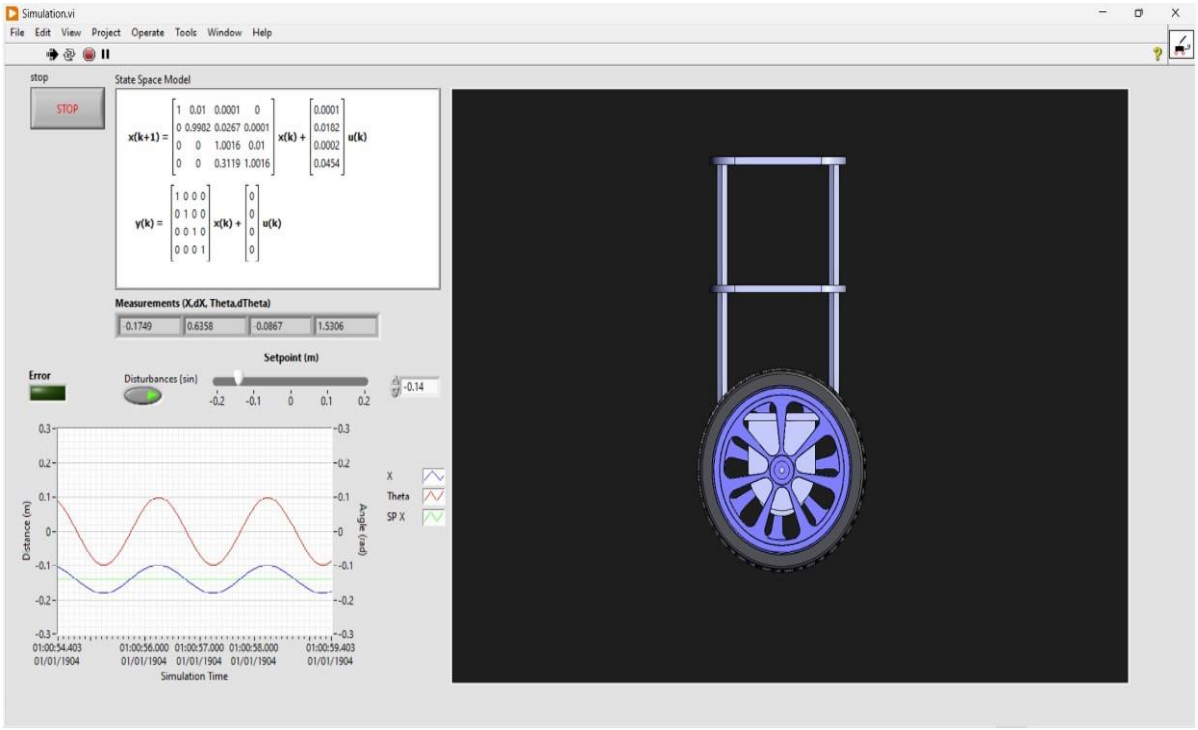


Figure 1.21 Modèle d'espace d'état du robot mobile auto-équilibré à deux roues dans un bloc LabVIEW[21]

1.7.2.8 Avantage de la logique floue pour équilibrer un robot :

- Gestion de l'incertitude ;
- Ne nécessite pas une modélisation ;
- Optimisation des performances.

1.7.2.9 Inconvénient de la logique floue pour équilibrer un robot :

- Complexité ;
- Sujette aux erreurs ;
- Manque d'adaptabilité et de flexibilité.

1.7.3 Equilibrage d'un robot avec Les algorithmes génétiques :

1.7.3.1 Définition :

L'algorithme génétique est un algorithme d'optimisation intelligent important qui opère sur une population spécifique en simulant le processus d'évolution naturelle et en utilisant l'évolution artificielle pour optimiser continuellement la

population afin de rechercher la solution optimale.[22] Il utilise des principes d'évolution pour trouver des solutions à des problèmes complexes en utilisant des techniques de sélection, de reproduction et de mutation.

1.7.3.2 Principe de fonctionnement des algorithmes génétiques :

Le schéma suivant résume le fonctionnement de l'algorithme génétique [23] :

1. L'algorithme commence par créer une population initiale aléatoire.
2. L'algorithme crée alors une séquence de nouvelles populations. A chaque étape, l'algorithme utilise les individus de la génération actuelle pour créer la population suivante. Pour créer la nouvelle population, l'algorithme effectue les étapes suivantes :

    a) Évalue chaque membre de la population actuelle en calculant sa valeur de fitness. Ces valeurs sont appelées les scores de fitness bruts.

    b) Met à l'échelle les scores de fitness bruts pour les convertir en une plage de valeurs plus utilisables. Ces valeurs mises à l'échelle sont appelées les valeurs d'attente.

    c) Sélectionne les membres, appelés parents, en fonction de leur valeur d'attente.

    d) Certains des individus de la population actuelle qui ont une faible valeur de fitness sont choisis comme élite. Ces individus d'élite sont transmis à la prochaine population.

    e) Produit des enfants à partir des parents. Les enfants sont produits soit en apportant des modifications aléatoires à un seul parent (mutation), soit en combinant les entrées vectorielles d'une paire de parents (croisement).

    f) Remplace la population actuelle par les enfants pour former la prochaine génération.

3. L'algorithme s'arrête lorsque l'un des critères d'arrêt est atteint. L'algorithme effectue des étapes modifiées pour les contraintes linéaires et entières.

L'algorithme est ensuite modifié pour les contraintes non linéaires.

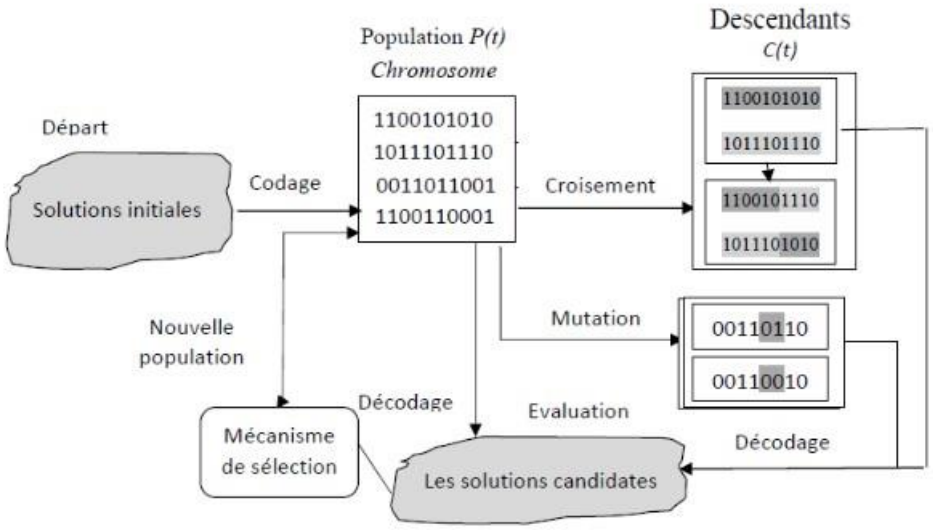


Figure 1.22 Fonctionnement d'un algorithme génétique. [24]

1.7.3.3 Paradigme :

Ce paradigme, associé avec la terminologie de la génétique, nous permet d'exploiter les algorithmes génétiques : Nous retrouvons les notions de Population, d'Individu, de Chromosome et de Gène.[25]

La population : c'est l'ensemble des solutions envisageables. (C'est un ensemble d'individus.

L'individu : représente une solution. (C'est une proposition au problème défini).

Le chromosome : est une composante de la solution.

Le gène : est une caractéristique, une particularité.

Espace de recherche : Définit l'ensemble des configurations possibles des paramètres de la fonction à optimiser

1.7.3.4 Codage :

Le codage des algorithmes génétiques est la manière dont les solutions potentielles d'un problème sont représentées sous forme de données compréhensibles par l'algorithme. Il existe plusieurs structures de données adaptées au problème, telles que des chaînes de bits, des nombres réels ou des vecteurs. Le choix du codage est crucial car il détermine comment les opérations génétiques, comme le croisement et la mutation, sont appliquées aux solutions pour générer de nouvelles solutions. Un bon codage permet une exploration efficace de l'espace de recherche et facilite la recherche de solutions optimales.

1.7.3.5 Opérateurs d'évolution

Il y'a exactement 3 opérateur d'évolution dans les algorithmes génétiques qui sont : La sélection, le croisement et la mutation.

1. **La sélection :** La sélection consiste à choisir les individus les mieux adaptés afin d'avoir une population de solution la plus proche de converger vers l'optimum global.[25]

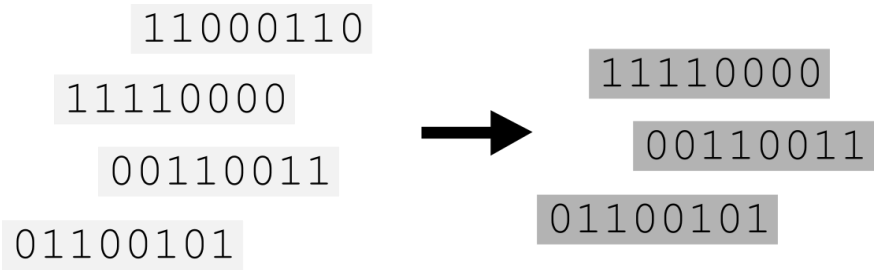


Figure 1.23 Opérateur de sélection [25].

2. **Le croisement :** Cette opération combine les caractéristiques de deux individus pour créer un nouvel individu. Cette opération est similaire à la reproduction sexuée dans la nature.

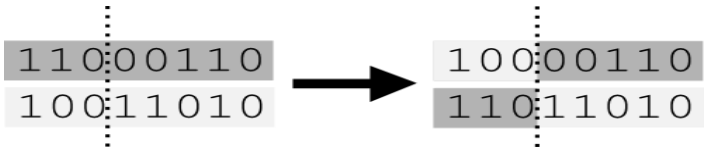


Figure 1.24 Opérateur de croisement (Simple enjambement) [25].

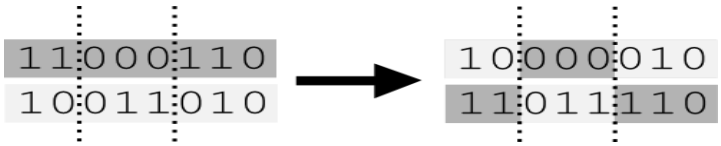


Figure 1.25 Opérateur de croisement (Double enjambement) [25].

3. **La mutation :** Cette opération consiste à modifier aléatoirement les caractéristiques d'un individu. Elle permet d'introduire de la diversité dans la population et d'explorer de nouvelles solutions potentielles.

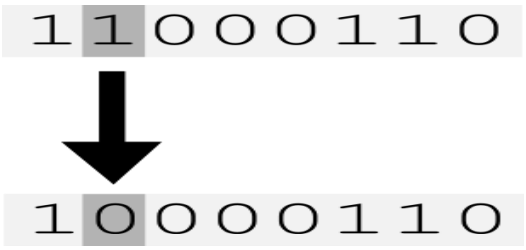


Figure 1.26 Opérateur de mutation.[25]

1.7.3.6 Fitness function :

La fitness function, ou bien, la fonction d'évaluation en français, est une fonction mathématique qui va calculer le score d'un individu. Cette fonction nous permet donc d'évaluer la qualité d'une solution candidate, Elle attribue une note à chaque individu de la population en fonction de sa capacité à résoudre le problème posé.

1.7.3.7 L'utilisation d'un algorithme génétique pour équilibrer un robot :

Théoriquement, voici comment pourrait-on faire pour équilibrer un robot en utilisant un algorithme génétique :

1. On définit les paramètres du robot tels que : la taille, le poids du robot etc. pour pouvoir créer un modèle de simulation du robot qui va nous servir à évaluer les performances de notre algorithme génétique.
2. On définit les gènes et les chromosomes : les gènes peuvent représenter les caractéristiques du robot, telles que la vitesse des roues, et les chromosomes sont des combinaisons de gènes.
3. On définit la fonction d'évaluation (la fitnesse function) pour évaluer la capacité du robot à rester en équilibre.
4. On génère une population initiale de façons aléatoire pour générer une population initiale de chromosomes.
5. En utilisant la fonction qu'on a définie on va évaluer les performances de chaque chromosome.
6. On sélectionne ensuite, les meilleurs chromosomes ceux qui donnent les meilleurs résultats.
7. On applique les opérateurs d'évolution : la mutation et le croisement pour créer une nouvelle génération de chromosomes encore plus performantes..
8. On réévalue encore une fois la performance de chaque chromosome de la nouvelle population.
9. On répète les étapes 7 à 9 jusqu'à obtenir la meilleur performance possible, selon la fonction d'évaluation qu'on a définie.

1.7.3.8 Exemple d'équilibrage d'un robot en utilisant les algorithmes génétiques:

Il s'agit d'une simple simulation d'un robot auto-équilibrant qui apprend ses paramètres LQR à l'aide d'un algorithme génétique. (Un LQR est une méthode bien connue qui fournit des gains de rétroaction de commande optimale permettant la conception en boucle fermée stable et à haute performance des systèmes.[26]).

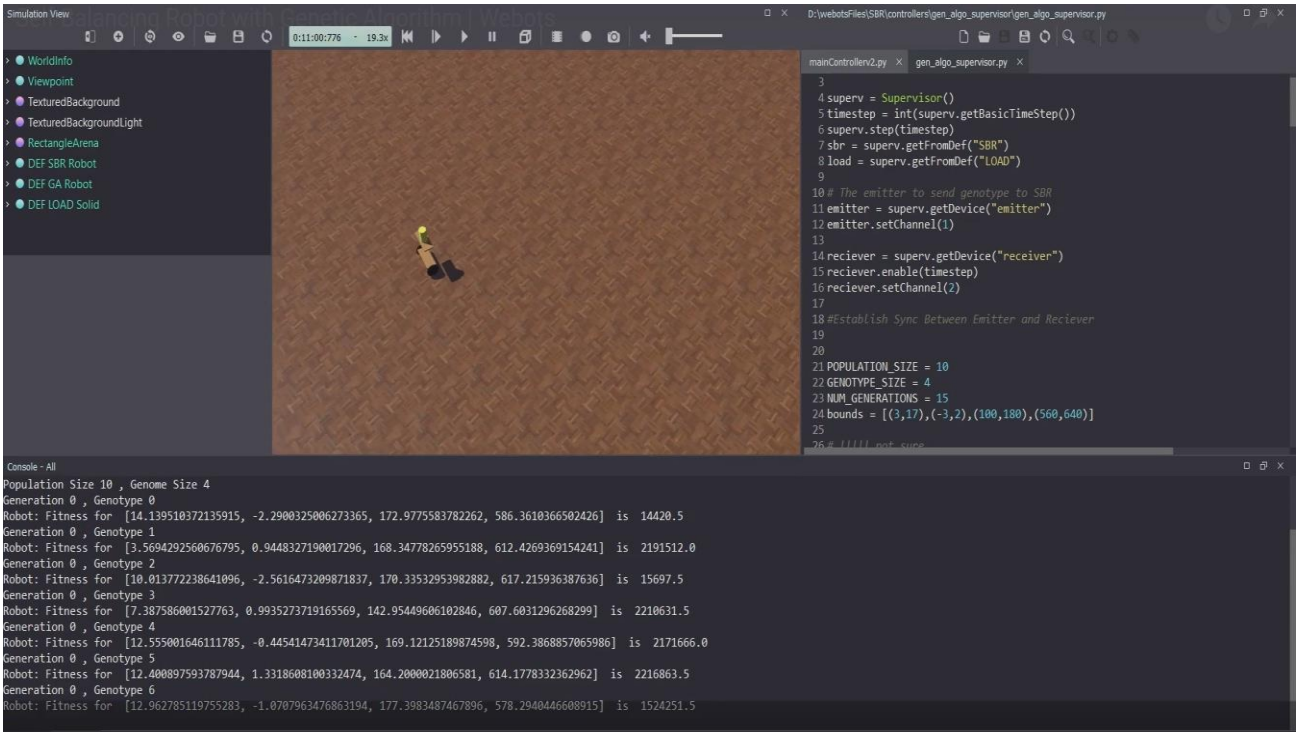


Figure 1.27 L'utilisation de l'algorithme génétique + LQR pour équilibrer un robot à deux roues [27].

1.7.3.9 Avantage d'un algorithme génétique pour l'équilibre :

- Capacité à traiter simultanément plusieurs solutions en parallèle ;

- Optimisation des performances en sélectionnant les meilleures solutions ;

- Possibilité de trouver des solutions optimales dans un grand espace de recherche de solutions possibles.

1.7.3.10 Inconvénient d'un algorithme génétique pour l'équilibre :

- Nécessite souvent un temps de calcul important pour trouver des solutions optimales ;

- L'algorithme génétique peut produire des résultats difficiles à interpréter, car il optimise une combinaison complexe de paramètres ;

- Il peut être difficile de définir les paramètres de l'algorithme génétique pour qu'il fonctionne correctement.

1.7.4 Equilibrage d'un robot avec L'apprentissage par renforcement :

1.7.4.1 Principe de fonctionnement :

Depuis un état S de l'environnement, l'agent sélectionne une action à l'aide d'une politique π.Il existe différents algorithmes d'apprentissage par renforcement pour la prise de décisions on peut en citer : Q-learning, Sarsa, Policy Gradient, Actor-critic, Deep-q-Network.

L'apprentissage par renforcement a pour but d'optimiser la politique d'action π de l'agent en utilisant un ensemble de récompenses positives et négatives.

L'apprentissage demandant un grand nombre d'expérimentation, il est très utile voir indispensable de disposer d'un environnement simulé. Les jeux vidéo étant un environnement simulé avec des récompenses (le score), ils sont un support classique pour l'apprentissage par renforcement. [28]

1.7.4.2 Agent et environnement :

Agent : L'ontologie de l'apprentissage par renforcement, on pourrait la considérer comme le cerveau humain. L'objectif de l'agent est d'obtenir la récompense cumulative maximale en interagissant avec l'environnement en suivant une politique donnée. [29]

L'environnement : est le contexte dans lequel l'agent opère. Cela peut être un monde physique réel ou bien, un simulateur virtuel. Il définit les règles et les contraintes avec lesquelles l'agent doit faire face. [9]

1.7.4.3 Politique π :

Est représentée par π. La politique est le cœur de l'algorithme et elle est suffisante en elle-même pour déterminer les comportements de l'agent. [30]

1.7.4.4 Exploration et Exploitation :

L'exploration et l'exploitation sont des concepts clés en apprentissage par renforcement. Dans ce contexte, l'exploration fait référence à la recherche d'actions qui pourraient être bénéfiques mais qui n'ont pas encore été essayées, tandis que l'exploitation consiste à utiliser les actions qui ont été jugées efficaces par le passé.

L'équilibre entre exploration et exploitation est crucial dans l'apprentissage par renforcement. Si un agent ne fait que choisir les actions qu'il sait être efficaces, il risque de manquer de nouvelles opportunités qui pourraient conduire à de meilleures récompenses à long terme. D'un autre côté, si l'agent se concentre trop sur l'exploration, il risque de gaspiller du temps et des ressources à essayer des actions qui ne sont pas utiles.

1.7.4.5 Value fuction :

Est une estimation de l'utilité ou de la qualité d'un état ou d'une action dans un système. Elle est généralement définie en termes de la récompense cumulative attendue, ou du retour (return), à partir de cet état ou de cette action en suivant une politique donnée, Plus précisément, la fonction de valeur d'état (state Value Function) est définie comme l'espérance du retour, notée V(s), étant donné un état s et en suivant une politique pi. Elle représente la valeur d'un état

en termes de la récompense cumulative attendue à partir de cet état sous la politique pi :

$$V(s) = E [G | S_t = s, \pi]$$

Où **G** est le retour (return) total, défini comme la somme des récompenses futures actualisées :

$$G = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...$$

Avec R_t+1, R_t+2, ... les récompenses futures et γ (gamma) le facteur d'escompte qui représente l'importance relative des récompenses futures par rapport aux récompenses immédiates.

1.7.4.6 Equation de Bellman :

Est une équation fondamentale en théorie de la décision et en apprentissage par renforcement. Elle décrit la relation entre la valeur d'un état ou d'une action et la valeur des états ou actions suivants. La formule de l'équation de Bellman est la suivante :

$$V(s) = \max_a [R(s,a) + \gamma * \Sigma_{s'} [P(s'|s,a) * V(s')]]$$

où :

- **V(s) :** est la valeur de l'état s
- **max_a :** est l'opérateur de maximisation sur toutes les actions possibles a dans l'état s
- **R(s,a) :** est la récompense obtenue en prenant l'action a dans l'état (s)
- **γ :** est le facteur d'actualisation, qui représente l'importance accordée aux récompenses futures par rapport aux récompenses immédiates
- **Σ_s' :** est la somme sur tous les états (s') atteignables à partir de l'état (s) en prenant l'action a
- **P (s'|s,a) :** est la probabilité de passer de l'état (s) à l'état (s') en prenant l'action a

1.7.4.7 Utilisation de l'apprentissage par renforcement pour équilibrer un robot :

La méthode consiste à ce qu'un 'agent, qui est le robot lui-même, interagisse avec l'environnement et ajuste ses actions (commandes moteur) en fonction de son inclinaison. En explorant différentes actions et en observant les récompenses ou pénalités associées (récompense pour maintenir l'équilibre, pénalité en cas de chute), l'agent apprend à optimiser sa politique pour maximiser les récompenses obtenues. Grâce à cette approche, le robot acquiert progressivement la capacité d'équilibrer de manière autonome, s'adaptant aux perturbations et maximisant sa récompense globale.

1.7.4.8 Algorithme de l'apprentissage par renforcement pour équilibrer un robot :

Cet exemple illustre l'utilisation d'un algorithme de l'apprentissage par renforcement plus précisément l'algorithme Q-learning pour équilibrer un robot à deux roues :

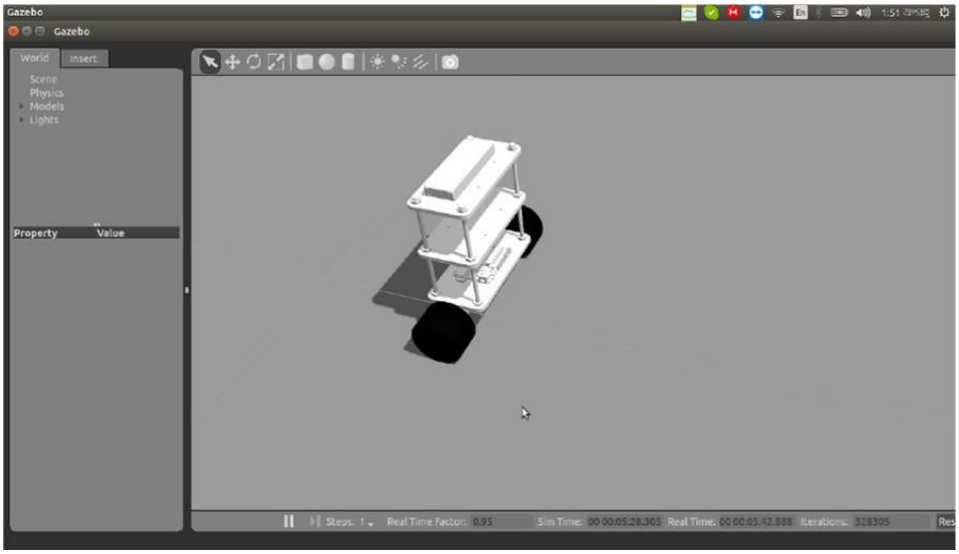


Figure 1.28 Plateforme robotique dans l'environnement Gazebo [31].

La simulation a été exécutée pour trois valeurs α différentes (0,7, 0,65, 0,8), avec une valeur γ de (0,999). La **Figure 1.29** montre les récompenses vs épisodes pour ces αs. C'est évident que le robot n'a pas pu gagner le montant ciblé de récompenses dans la période de formation pour ces taux d'apprentissage. Nous voir que, pour les valeurs α de 0,7 et 0,8, le robot a atteint au maximum de récompenses accumulées possibles, 2000, dans 400 épisodes. La courbe avec la valeur α de 0,7 est moins stable que celle de 0,8. Cependant, la courbe avec la valeur α de 0,65 n'a jamais atteint la précision maximale récompense mulée[31].

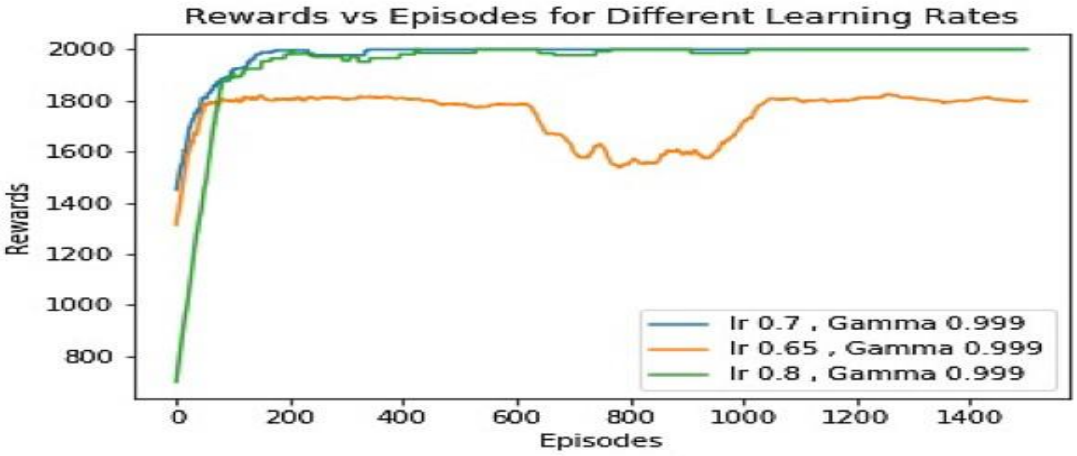


Figure 1.29 Récompense pour différents α [31].

1.7.4.9 Avantage de l’apprentissage par renforcement pour équilibrer un robot :

- Adaptabilité ;

- Optimisation ;

- Réduction du coût de développement.

1.7.4.10 Inconvénient de l’apprentissage par renforcement pour équilibrer un robot :

- Apprentissage lent ;

- Sensibilité aux récompenses ;

- Paramétrages des hyperparamètres complexe.

1.8 Conclusion

En conclusion, l'équilibrage d'un robot est un défi complexe qui nécessite l'utilisation de techniques appropriées. Nous avons exploré différentes approches, telles que le contrôleur PID, la logique floue, les algorithmes génétiques et l'apprentissage par renforcement. Chacune de ces méthodes offre ses propres avantages et inconvénients, et le choix de la technique dépendra des spécificités du problème, des contraintes et des objectifs. Il est crucial de prendre en compte la complexité de l'environnement, la flexibilité requise, les capacités de calcul et les ressources disponibles lors du choix de la méthode d'équilibrage.

Chapitre 02 :

Les algorithmes d’apprentissage par renforcement

2.1 Introduction

Dans ce chapitre, nous aborderons les principaux algorithmes d'apprentissage par renforcement tels que le **Q-learning**, **Deep-Q-Network** et **SARSA**. Pour une compréhension approfondie de ces algorithmes, il est essentiel de mettre en évidence le processus de décision markovien (MDP) qui constitue un élément essentiel dans la prise de décision de ces algorithmes. Ensuite, nous explorerons en détail chaque algorithme, en exposant leur principe de fonctionnement, leurs avantages et inconvénients respectifs et finalement, nous mettrons l'accent sur le pendule inversé, car il offre une illustration pertinente du principe d'équilibre utilisé par les robots auto-équilibrés à deux roues. En examinant le fonctionnement du pendule inversé, nous pourrons mieux appréhender les fondements et les mécanismes qui permettent à ces robots de maintenir leur stabilité.

2.2 Le processus de décision markovien

L'apprentissage par renforcement se place dans le cadre des processus de décision markovien (Markov Decision Process MDP). L'évolution de l'environnement doit respecter la propriété de Markov : la fonction de transition T(**s**,**a**,**s'**) doit être une fonction stochastique attribuant, pour une action **a** exécutée dans un état **s**, les probabilités de se retrouver dans chaque état **s'** et cette distribution de probabilité doit être indépendante des actions et état précédents. Ainsi, le processus est modélisé par le quadruplet {S, A, T, R}[32] :

- un ensemble d'états S. ensemble (continu ou discret) des états de l'environnement que perçoit l'agent ;
- un ensemble d'actions A, discret ou continu ;
- une fonction de transition T(**s**,**a**,**s'**) représentant la probabilité de se retrouver dans l'état **s'** en effectuant l'action a depuis l'état s ;
- une fonction de récompense R(**s**,**a**,**s'**), indiquant la récompense obtenue lors du passage de s à **s'** en effectuant l'action a.

Notons que dans un environnement déterministe, la fonction de transition est simplifiée : elle indique directement l'état **s'** dans lequel se retrouve l'environnement après l'exécution de l'action a depuis l'état **s** [32]**.**

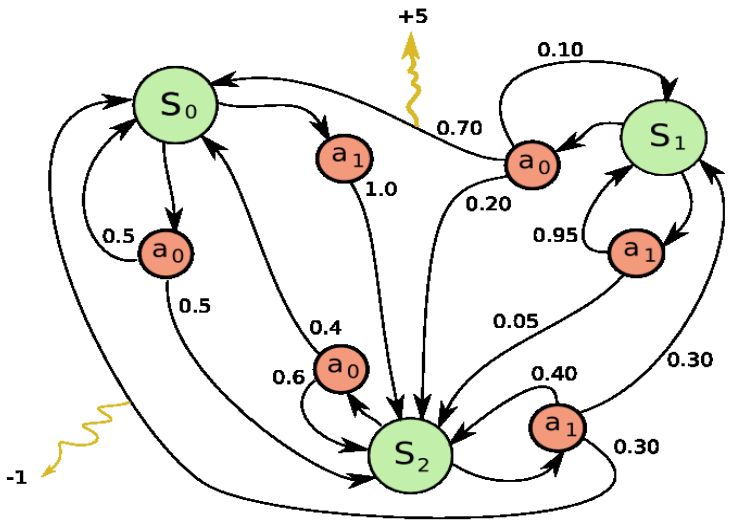


Figure 2.1 Exemple d'une représentation d'un MDP.

2.3 L'algorithme Q-learning :

2.3.1 Définition

Le Q-learning est largement reconnu comme l'un des algorithmes les plus populaires et utilisés dans le domaine de l'apprentissage par renforcement. Il repose sur l'apprentissage par l'expérience et a été initialement introduit par CJ Watkins en 1989 [33]. Bien qu'il soit considéré comme une forme d'apprentissage primitive, il constitue une base solide pour des systèmes plus avancés.

L'algorithme consiste à ce qu'un agent explore son environnement, représenté par différents états. L'agent interagit avec l'environnement en effectuant des actions et en observant les conséquences de ces actions, généralement sous la forme de récompenses. La récompense est une somme pondérée des valeurs attendues de toutes les futures récompenses à venir. L'objectif principal est de maximiser la récompense cumulée au fil du temps en choisissant l'action qui offre la récompense la plus élevée pour un état donné. Cette action, qui génère la plus grande récompense à long terme, est considérée comme l'action optimale. À mesure que l'agent explore et exécute suffisamment d'actions dans tous les états, il apprend progressivement un modèle des actions optimales.

Figure 2.2 Principe du Q-learning

2.3.2 Propriétés Q-learning:

Q-learning est un algorithme de contrôle de différence temporelle (TD) hors politique, et sans modèle. Examinons maintenant la signification de ces propriétés :

2.3.2.1. Apprentissage par renforcement sans modèle

Q-learning est un algorithme sans modèle. Nous pouvons considérer les algorithmes sans modèle comme des méthodes d'essais et d'erreurs. L'agent explore l'environnement et apprend directement des résultats des actions, sans construire un modèle interne ou un processus décisionnel de Markov. Au début, l'agent connaît les états et les actions possibles dans un environnement. Ensuite, l'agent découvre les transitions d'état et les récompenses par l'exploration [34].

2.3.2.2 Différence temporelle

Il s'agit d'un algorithme TD, où les prédictions sont réévaluées après avoir franchi une étape. Même les épisodes incomplets génèrent des entrées pour un algorithme TD. Dans l'ensemble, nous mesurons en quoi la dernière action est différente de ce que nous avions estimé initialement, sans attendre un résultat final. Dans Q-learning, les valeurs Q stockées dans la table Q sont partiellement mises à jour à l'aide d'une estimation. Par conséquent, il n'est pas nécessaire d'attendre la récompense finale et de mettre à jour les valeurs de paire état-action précédentes dans Q-learning [34].

2.3.2.3 Apprentissage hors politique

Q-learning est un algorithme hors politique. Il estime la récompense pour les paires état-action en fonction de la politique optimale (gourmande), indépendante des

actions de l'agent. Un algorithme hors stratégie se rapproche de la fonction de valeur d'action optimale, indépendamment de la stratégie. En outre, les algorithmes hors

stratégie peuvent mettre à jour les valeurs estimées à l'aide d'actions inventées. Dans ce cas, l'algorithme Q-learning peut explorer et bénéficier d'actions qui ne se sont pas produites pendant la phase d'apprentissage [34].

2.3.3 Q-Table

Est une approche pratique qui consiste à stocker les Q-values dans une table dont les axes représentent respectivement les états et actions comme illustré en Figure 2.1. Chaque valeur de la table représente une Q-value pour une paire (état, action) dont la valeur est mises à jour à l'aide de la Q-function [35].

La taille de la Q-table dépend du nombre d'états et d'actions inclus dans le problème à résoudre. Par exemple si vous avez 20 états et 3 actions, vous aurez une table de taille 20x3

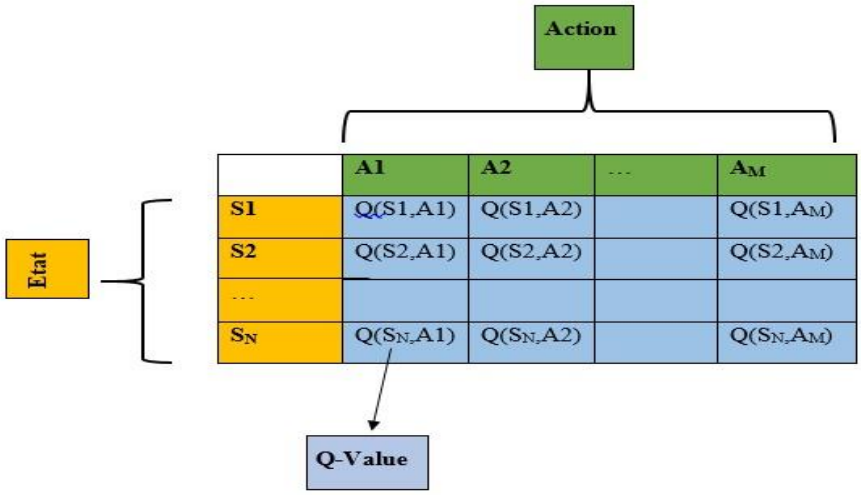


Figure 2.3 Représentation de la Q-table

2.3.4 Q-Value

La Q-value représente l'estimation de la valeur à long terme d'une action dans un état donné, en prenant en compte les récompenses futures potentielles.

2.3.5 Q-function

La **Q-function** est une fonction mathématique qui est utilisé par un agent pour évaluer la valeur d'une action dans un état spécifique

Ainsi, lorsque un agent acquiert une nouvelle expérience, il ajoute une nouvelle donnée à sa Q-table notée comme suit : **Q(s,a)**, où **s** représente l'état actuel et **a** représente l'action effectuée. Cette valeur Q estime la valeur optimale pour cette action dans cet état. L'agent utilise ensuite ces informations pour apprendre un modèle optimal d'actions.

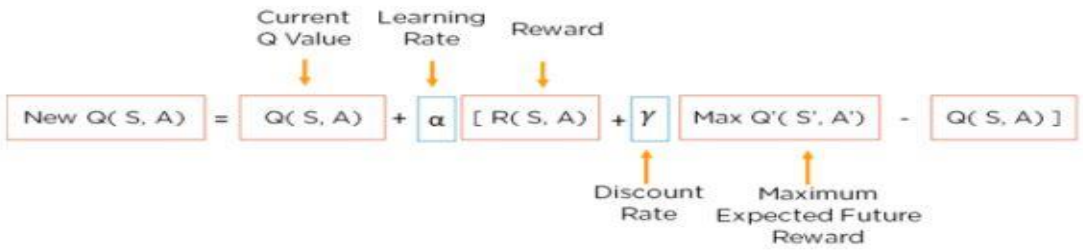


Figure 2.4 Q-Function [37]

S : Représente l'état actuel.

A : Représente l'action.

Learning rate (Alpha) : lr, souvent appelé alpha, peut être défini comme le **degré d'acceptation** de la nouvelle valeur par rapport à l'ancienne. Ci-dessus, nous prenons la différence entre la nouvelle et l'ancienne valeur, puis nous multiplions cette valeur par le taux d'apprentissage. Cette valeur est ensuite ajoutée à notre valeur q précédente, ce qui la fait évoluer dans la direction de notre dernière mise à jour. [36]

Reward : La récompense (reward) est la **valeur reçue** après avoir effectué une certaine action à un état donné. Une récompense peut survenir à n'importe quel pas de temps donné ou seulement au pas de temps terminal. [36]

Discount rate (Gamma) : La récompense (reward) est la **valeur reçue** après avoir effectué une certaine action à un état donné. Une récompense peut survenir à n'importe quel pas de temps donné ou seulement au pas de temps terminal. [36]

S' : Représente l'état à t+1.

max Q(St+1,at+1)-Q(St,At) old : cette fonction elle fait la différence entre le prochain état ou je sélectionne l'action qui maximiserai ma q-function et l'état ou je me trouve actuellement c'est en quelques sorte un gradiant.

2.3.6 La sélection d'action :

Un agent interagit avec l'environnement de deux manières. La première consiste à utiliser la table Q comme référence et à visualiser toutes les actions possibles pour un état donné. L'agent sélectionne alors l'action basée sur la valeur maximale de ces actions. C'est ce qu'on appelle l'**exploitation** puisque nous utilisons les informations dont nous disposons pour prendre une décision.[36]

La deuxième manière consiste à agir de façon aléatoire. C'est ce qu'on appelle l'**exploration**. Au lieu de sélectionner des actions en fonction de la récompense future maximale, nous choisissons une action au hasard. Agir au hasard est important car cela permet à l'agent d'explorer et de découvrir de nouveaux états qui, autrement, pourraient ne pas être sélectionnés pendant le processus d'exploitation. [36]

2.3.7 Algorithme d'epsilon greedy :

Algorithm 2: Epsilon-Greedy Action Selection

```
Data: Q: Q-table generated so far, : a small number, S: current
      state
Result: Selected action
Function SELECT-ACTION(Q, S, ϵ) is
    n ← uniform random number between 0 and 1;
    if n < ϵ then
        A ← random action from the action space;
    else
        A ← maxQ(S,.);
    end
    return selected action A;
end
```

Figure 2.6 algorithme epsilon-greedy pour selectionner une action[34]

Ici la fonction **SELECT-Action (Q,S,ε)** consiste à initialiser une variable n qui va prendre une valeur aléatoire entre 1 et 0 et mettre deux conditions :

1 - si cette valeur est inferieur a epsilon alors on va choisir une action au hasard selon la valeur **n** qu'on a généré et on retourne cette valeur la donc on va faire de l'exploration.

- 2 - sinon on va tout simplement faire de l'exploitation qui va retourner l'action qui va maximiser la récompense dans un état donné.

2.3.8 Algorithme Epsilon greedy Q-learning :

Algorithm 1: Epsilon-Greedy Q-Learning Algorithm

```
Data: α: learning rate, γ: discount factor, ϵ: a small number
Result: A Q-table containing Q(S,A) pairs defining estimated
        optimal policy π*
/* Initialization                                          */
Initialize Q(s,a) arbitrarily, except Q(terminal,.);
Q(terminal,.) ← 0;
/* For each step in each episode, we calculate the
   Q-value and update the Q-table                          */
for each episode do
    /* Initialize state S, usually by resetting the
       environment                                         */
    Initialize state S;
    for each step in episode do
        do
            /* Choose action A from S using epsilon-greedy
               policy derived from Q                       */
            A ← SELECT-ACTION(Q, S, ϵ);
            Take action A, then observe reward R and next state S';
            Q(S, A) ← Q(S, A) + α [ R + γ max_a Q(S', a) - Q(S, A)];
            S ← S';
        while S is not terminal;
    end
end
```

Figure 2.7 L'algorithme Epsilon-greedy Q-learning [34].

2.3.9 Avantage de l'utilisation du q-learning:

- Il peut fonctionner sans avoir connaissance préalable du modèle du système ;
- C'est un algorithme simple et facile à implémenter ;
- il sait gérer les problèmes de transitions et de récompenses stochastiques sans nécessiter d'adaptations.

2.3.10 Inconvénient de l'utilisation du q-learning:

- Ne peut fonctionner qu'avec des environnements qui ont des états finis et discrets ;

- Peut rencontrer des difficultés lorsque l'espace d'état est trop grand ;
- Peut converger vers politiques sous-optimales si les paramètres sont mal définis.

2.4 L'algorithme Deep Q-Network

2.4.1 Définition :

Est une variation de l'algorithme classique de Q-Learning avec 3 contributions principales : (1) une architecture de réseau neuronal convolutionnel profond pour l'approximation de la fonction Q ; (2) l'utilisation de mini-lots de données d'entraînement aléatoires plutôt que des mises à jour d'une seule étape sur la dernière expérience ; et (3) l'utilisation de paramètres de réseau plus anciens pour estimer les valeurs Q de l'état suivant. [38]

Il a démontré une grande capacité à apprendre des stratégies gagnantes dans des jeux vidéo complexes tels que Atari 2600. Il a également été utilisé avec succès dans d'autres domaines, tels que la robotique par exemple pour équilibrer un robot, lui apprendre à marcher et la planification de trajectoire etc.

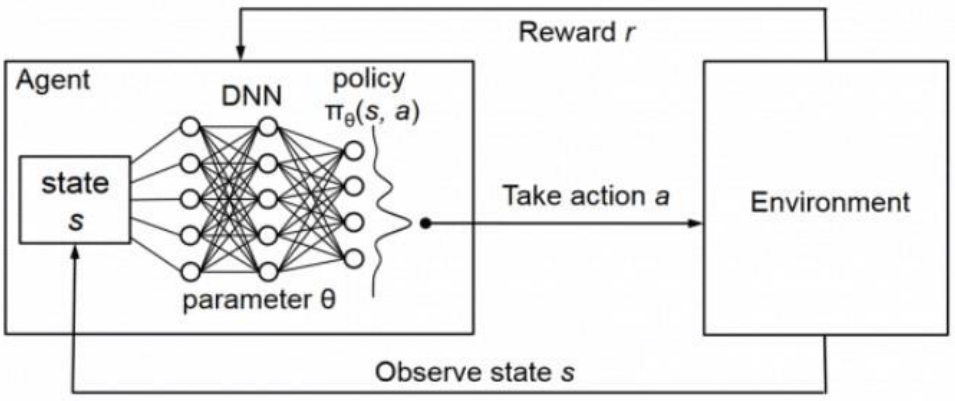


Figure 2.8 illustration du DQN [39].

2.4.2 Réseaux de neurones profonds :

Un réseau de neurones profond est un réseau de neurones multicouches qui peut comporter des millions de neurones, répartis en plusieurs dizaines de couches. [40]

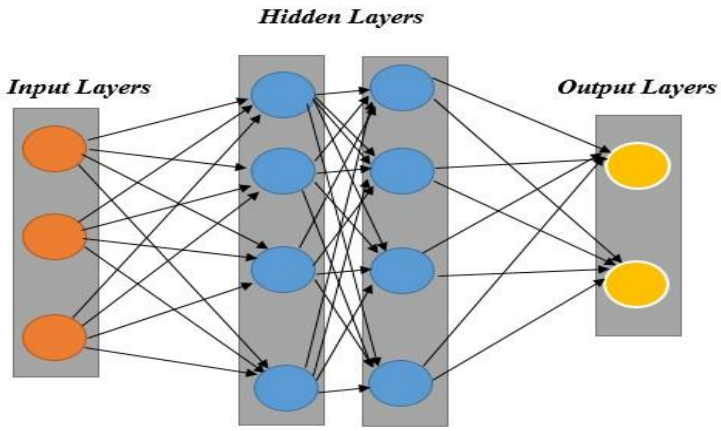


Figure 2.9 illustration d'un réseau de neurones profonds

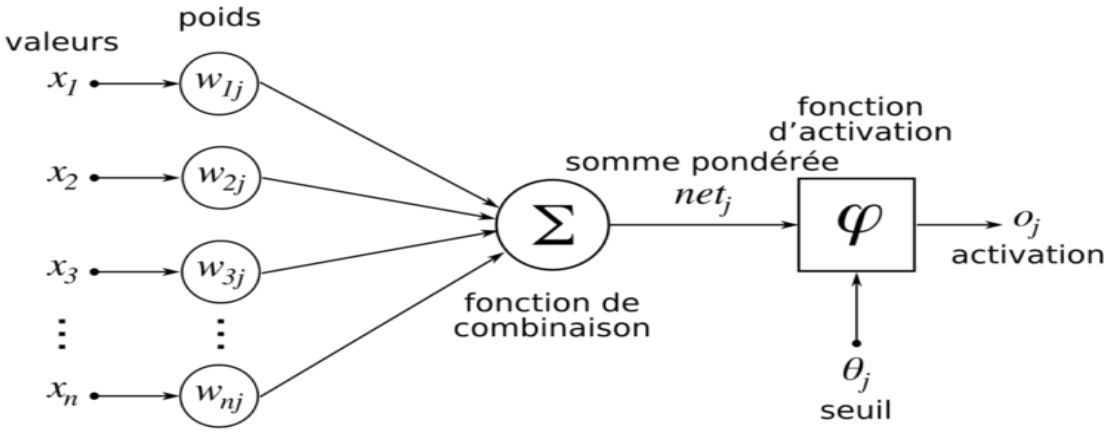


Figure 2.10 figure montrant le fonctionnement d'un réseau de neurone [41].

2.4.3 Paramètres d'un réseau de neurone :

2.4.3.1 Les poids :

Sont des coefficients qui sont appliqués aux entrées du réseau de neurones pour calculer la sortie. Chaque neurone dans une couche est connecté à tous les neurones de la couche précédente, et chaque connexion est associée à un poids. Les poids déterminent l'importance relative de chaque entrée pour le calcul de la sortie.

2.4.3.2 Le biais

Est une constante qui est ajoutée à la somme pondérée des entrées d'un neurone avant d'être passée à travers la fonction d'activation. Son but est d'introduire un décalage dans la sortie du neurone.

2.4.4 La fonction d'activation :

Une fonction d'activation est une fonction mathématique utilisé sur un signal. Elle va reproduire le potentiel d'activation que l'on retrouve dans le domaine de la biologie du cerveau humain. Elle va permettre le passage d'information ou non de l'information si le seuil de stimulation est atteint. Concrètement, elle va avoir pour rôle de décider si on active ou non une réponse du neurone [42]. Un neurone ne va faire qu'appliquer la fonction suivante :

$$X = \sum (\text{entrée} * \text{poids}) + \text{biais}$$

C'est sur cette sortie que la fonction d'activation va s'appliquer.

2.4.5 Processus d'entrainement d'un réseau de neurone :

Le processus d'entrainement d'un réseau de neurone passe par 4 étapes essentielles :

2.4.5.1 Forward Propagation

Est le processus par lequel les données d'entrée sont transmises à travers un réseau de neurones, dans une direction avant, pour générer une sortie. Les données sont acceptées par les couches cachées et traitées, selon la fonction d'activation, et se déplacent vers la couche suivante.[44]

2.4.5.2 Calculer l'erreur de la sortie (Cost function)

Est une mesure mathématique qui évalue à quel point les prédictions du réseau correspondent aux valeurs réelles des données d'apprentissage. Cette étape quantifie l'écart entre les prédictions du réseau et les valeurs cibles, fournissant ainsi une mesure de l'erreur de prédiction.

2.4.5.3 Backward Propagation

Est utilisée en apprentissage automatique pour améliorer la précision des prédictions grâce à la propagation en arrière des dérivées calculées. La propagation en arrière est le processus qui consiste à se déplacer de droite (couche de sortie) à gauche (couche d'entrée)..[44]

2.4.5.4 Correction des paramètres avec l'algorithme Gradient Descent

Cette dernière étape, consiste à corriger chaque paramètre du modèle grâce à l'algorithme de la descente de grandient (est un algorithme d'optimisation couramment utilisé pour entraîner des modèles d'apprentissage automatique et

des réseaux de neurones [45]. Il permet d’ajuster les poids du réseau de neurones afin de minimiser la fonction de coût) avant de reboucler vers la première étape, celle de la Forward propagation, pour recommencer un cycle d’entrainement.

2.4.6 Replay memory :

Est une technique clé à l'origine de nombreuses avancées récentes dans l'apprentissage par renforcement profond. Il permet à l'agent d'apprendre à partir de souvenirs antérieurs peut accélérer l'apprentissage et rompre les corrélations temporelles indésirables. Malgré son application généralisée, on comprend très peu les propriétés de la relecture d'expérience.[46]

2.4.7 Mini-batch :

Le mini-batch est une technique d'apprentissage par renforcement qui consiste à entraîner un modèle de manière incrémentale en utilisant des échantillons de données stockés dans la mémoire de relecture (replay memory). Elle a été démontrée comme offrant de meilleures performances de généralisation et permettant une empreinte mémoire considérablement plus petite, ce qui pourrait également être exploité pour améliorer le débit de la machine. [47]

2.4.8 Algorithme DQN:

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode = 1, M **do**

Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

For $t = 1$,T **do**

With probability ε select a random action a_t

otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

Execute action a_t in emulator and observe reward r_t and image x_{t+1}

Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$

Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ

Every C steps reset $\hat{Q} = Q$

End For

End For

Figure 2.11 l'algorithme deep-q-network [43]

2.4.9 Avantage du DQN:

- Il peut apprendre à partir de données brutes, telles que des images, sans nécessiter d'ingénierie de fonctionnalités manuelle
- Peut être utilisé pour résoudre des problèmes d'apprentissage par renforcement à grande échelle
- Peut être utilisé pour apprendre des politiques stochastiques, ce qui permet de prendre en compte l'incertitude dans l'environnement.

2.4.10 Inconvénient du DQN:

- Peut être instable et difficile à entraîner, en particulier lorsque les récompenses sont rares ou lorsque l'environnement est très complexe.
- Peut être sujet au surapprentissage, ce qui peut entraîner une performance médiocre sur des données de test.

- Peut nécessiter beaucoup de données pour être efficace, ce qui peut rendre l'entraînement coûteux en temps et en ressources.

2.5 Sarsa :

2.5.1 Définition

L'algorithme SARSA (STATE-ACTION-REWARD-STATE- est un algorithme "on-policy" utilisé pour apprendre une politique de processus de décision de Markov dans l'apprentissage par renforcement.[50]. il est similaire au Q-learning, sauf qu'il se concentre sur l'apprentissage en suivant une politique déterministe

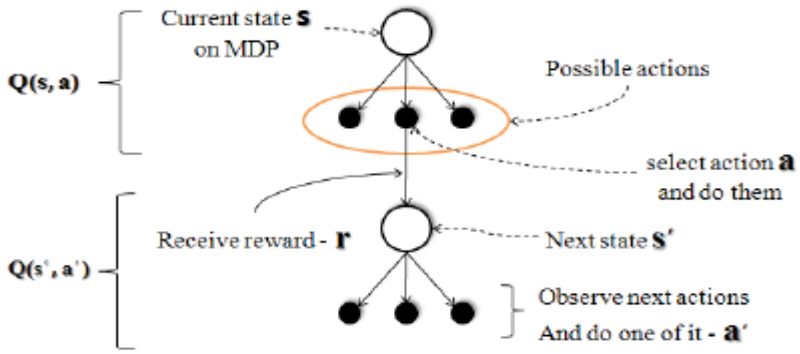


Figure 2.12 principe de l'algorithme Sarsa

2.5.2 L'algorithme Sarsa

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

Figure 2.13 L'algorithme Sarsa [48].

2.5.3 Avantage de l'algorithme Sarsa

- il prend en compte la politique réelle du système, plutôt que de supposer simplement qu'il fait le bon choix ; [49]
- L'avantage d'utiliser SARSA, en suivant l'action qui est réellement prise à l'étape suivante, est que la politique qu'il suit sera plus optimale et l'apprentissage sera plus rapide ; [49]
- C'est un algorithme sans modèle, ce qui signifie qu'il ne nécessite pas de modèle de l'environnement pour apprendre ; [49]
- Il peut apprendre des entrées sensorielles brutes et mapper directement les entrées aux actions ; [49]
- Il converge vers une politique optimale, même dans des environnements complexes. [49]

2.5.4 Inconvénient de l'algorithme Sarsa

- Il peut être lent à converger, en particulier dans les grands environnements ; [49]
- Il est sensible au choix des hyperparamètres, tels que le taux d'exploration et le taux d'apprentissage ; [49]
- Il peut ne pas trouver la stratégie optimale dans certains environnements. [49]

2.6 Définition de la pendule inversé :

Le système du pendule inversé représente un problème classique d'une importance considérable dans les domaines de la dynamique et de la théorie du contrôle. Il comprend un pendule inversé fixé à un chariot motorisé, où la masse est positionnée au-dessus du point de pivotement. Ce système sert de référence importante pour l'évaluation expérimentale des algorithmes de contrôle dans des applications en temps réel.[51]

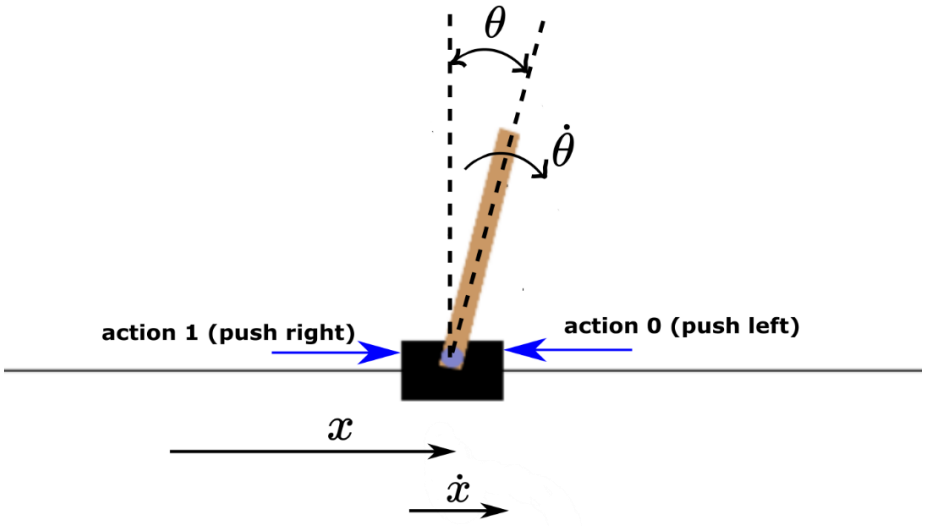


Figure 2.14 Pendule inversée [52].

- Position du chariot, notée **x** sur la figure 1.
- Vitesse du chariot, notée **\dot{x}**
- Angle de rotation polaire (mesuré en radians), noté **\theta.**
- Vitesse angulaire du pôle, notée **\dot{\theta}.**

2.7 Conclusion

En conclusion, les algorithmes d'apprentissage par renforcement tels que le Q-Learning, SARSA et le Deep Q-Network (DQN) ont révolutionné le domaine de l'intelligence artificielle en permettant à un agent d'apprendre à prendre des

- décisions optimales dans des environnements complexes. Le Q-Learning offre une approche basée sur les valeurs Q, SARSA se concentre sur l'apprentissage basé sur la politique, et le DQN utilise des réseaux de neurones profonds pour estimer les valeurs Q. Chacun de ces algorithmes présente ses propres forces et limitations, mais ils ont tous contribué à des avancées significatives dans des domaines.

Chapitre 03 :
Conception

3.1 Introduction

Pour pouvoir maintenir un robot en équilibre, nous aurons besoin d'une plateforme agissant en tant qu'interface entre le matériel et le logiciel. C'est pourquoi nous utiliserons un système embarqué. Un système embarqué est composé essentiellement de deux parties qui sont : le logiciel, également appelé « Software », et le matériel, appelé « Hardware ». Dans ce chapitre, nous nous pencherons sur ces deux parties, pour cela nous allons pour chaque partie établir un organigramme qui va illustrer le schéma utilisé pour mieux comprendre leur fonctionnement.

Partie Software, nous aborderons l'algorithme qu'on a utilisé, qui est un mélange entre un contrôleur PID et un algorithme Q-learning afin d'optimiser efficacement les actions qu'entreprend le robot.

Partie Hardware, nous aborderons les différents composants matériels qu'on a utilisé pour concevoir le robot à deux roues.

3.2 Partie logiciel :

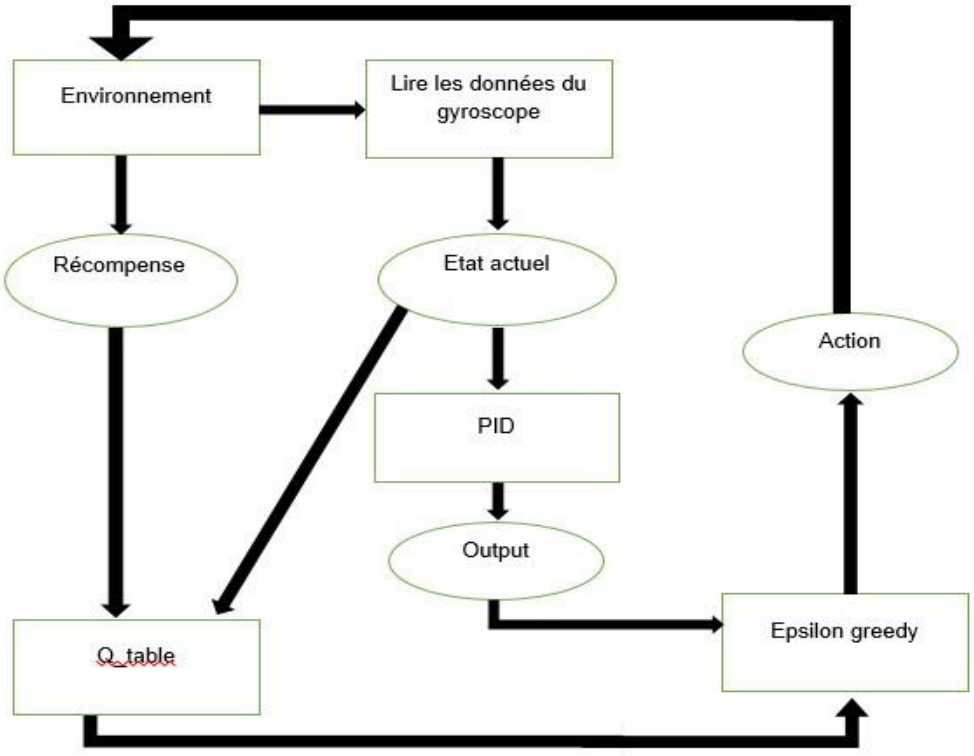


Figure 3.1 Organigramme partie Software du robot

Explication :

Afin de mieux comprendre le fonctionnement de cet organigramme et mettre en lumière les processus internes du logiciel, nous allons l'expliquer de manière détaillée :

Nous avons un environnement dans lequel notre agent va interagir avec, dans notre cas, il s'agit du robot. Cet agent la, aura un état initial qui est représenté par la valeur mesurée du gyroscope selon cet organigramme

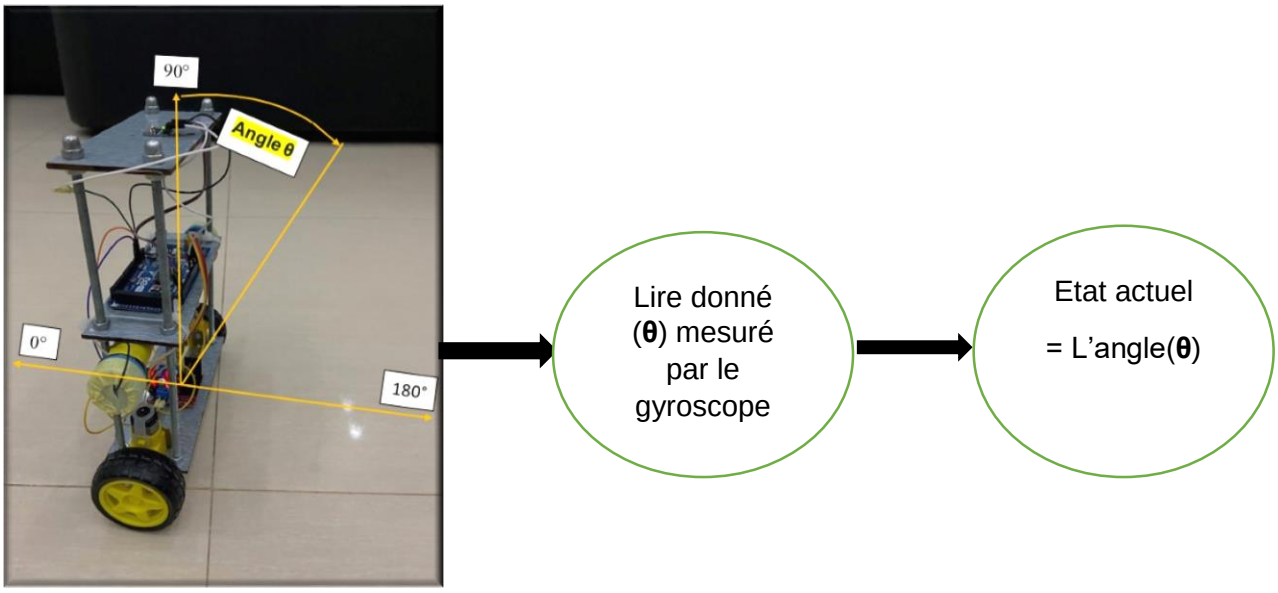


Figure 3.2 Organigramme représentant l'environnement

Ensuite, nous allons passer cette valeur au PID pour qu'il puisse faire du traitement dessus en utilisant les coefficients proportionnel (KP), intégral(KI) et dérivé (KD). Une fois les calculs terminé le PID va nous générer une valeur de sortie (Output) qui représente l'action à effectuer. On passe cette action à un algorithme d'epsilon greedy pour soit choisir l'action du PID ou bien trouver une action plus optimale selon cet organigramme :

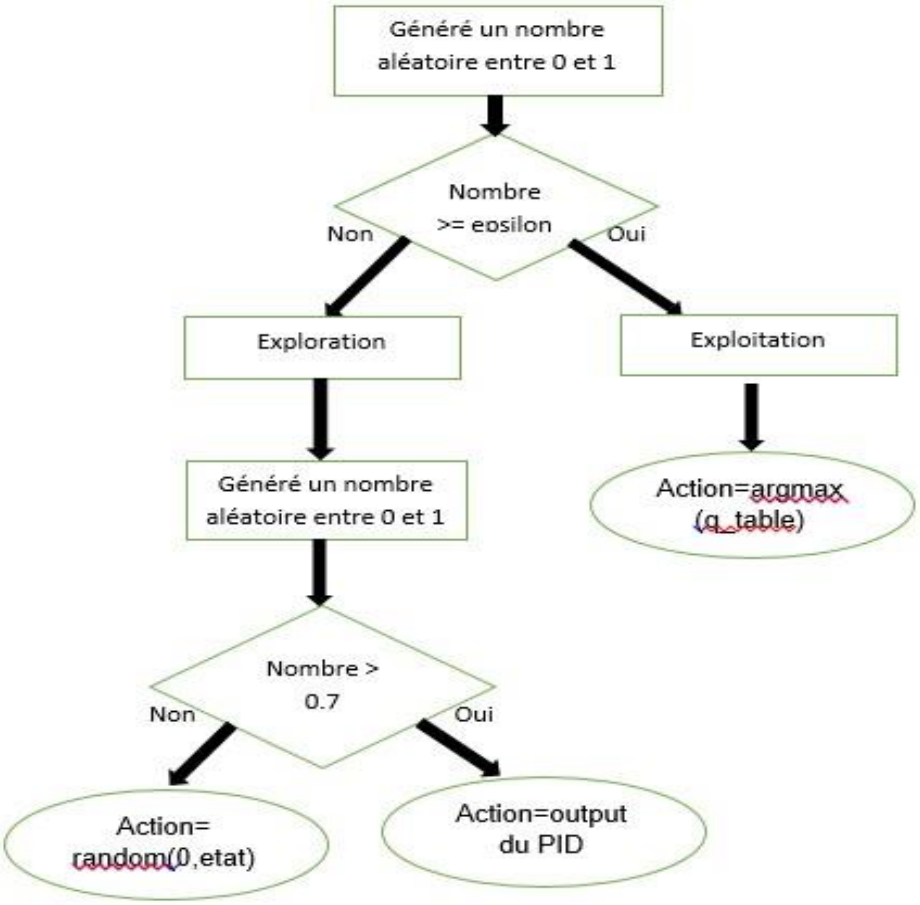


Figure 3.3 Organigramme epsilon-greedy

L'argmx (q_table) consiste à choisir l'action la plus haute selon cet organigramme (c'est ce qu'on appelle la greedy policy) :

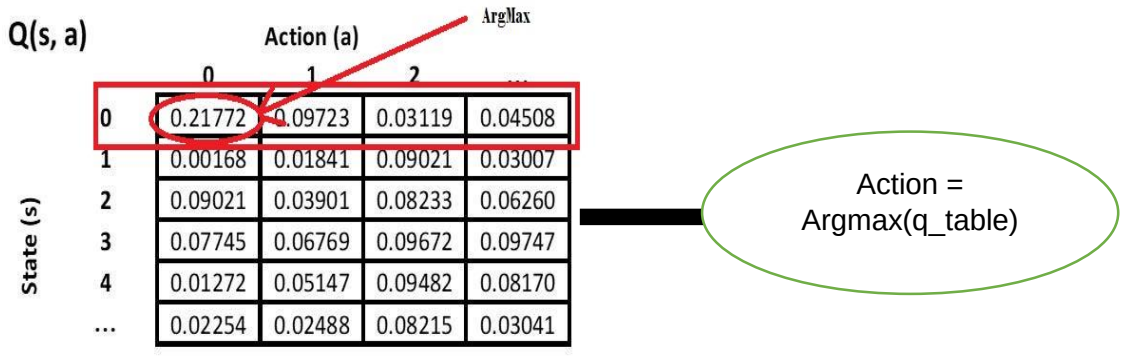


Figure 3.4 Organigramme Argmax de la qtable

L'action choisie sera par la suite exécuter dans notre environnement, qui correspondra à la vitesse des moteurs (Elle varie entre -255 et 255) les valeurs entre

0 et 255 seront utilisé pour faire avancer le robot, tandis que les valeurs compris entre -255 et 0 c'est pour faire reculer le robot.

L'environnement nous renverra par la suite 2 valeurs qui sont : le nouvel état et la récompense qui est obtenue, si après avoir effectué une action A notre robot se trouve dans un intervalle propice à l'équilibre sinon il recevra une pénalité, selon cet organigramme :

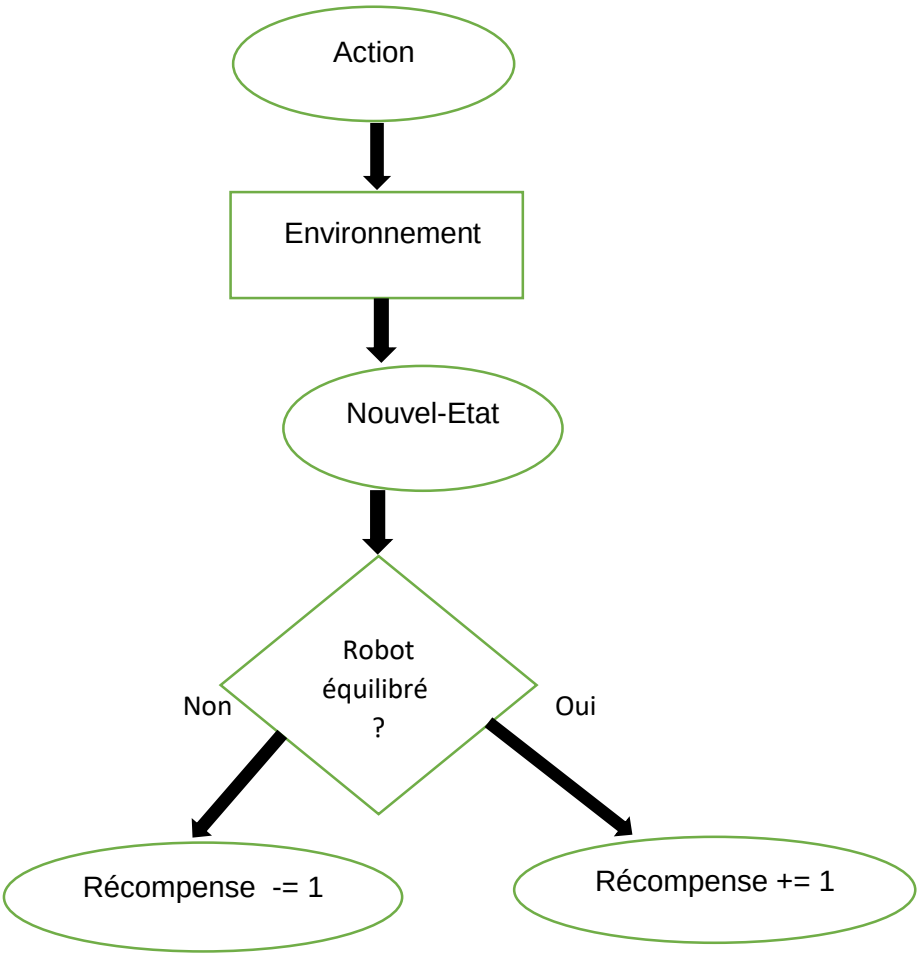


Figure 3.5 Organigramme pour les récompenses

Une fois la récompense attribué, il ne reste plus qu'à mettre à jour notre q_table en utilisant la q_function : **Q(s,a) = Q(s,a) + alpha * [R(s,a) + gamma * max Q'(s',a') - Q(s,a)]** selon cet Exemple :

On commence par définir les hyperparamètres :

- **gamma = 0,9**
- **alpha = 0,1**

Q(s,a)		Action			
		0	**1**	**2**	**….**
State	**0**	0.04214	0.02314	0.0542	0.1234
	1	0.4255	0.5258	-0.425	-0.01
	2	0.098	0.5742	0.8542	0.4924
	….	0.7548	-0.9582	0.1234	0.05732

Si on prend comme paramètres :

- **l'état S = 1 :**

0	**1**	**2**	**3**
0.4255	0.5258	-0.425	-0.01

- **l'action = 1**
- **reward (s, a) = 5**

- **L'état S' = 2 :**

0	**1**	**2**	**3**
0.098	0.5742	0.8542	0.4924

max Q'(s',a') = max Q'(2) = 0.8542

Q(1,1) = Q(1,1) + 0,1 * [5 + 0,9 * 0.8542 - Q(1,1)]

Q(1,1) = 0.5258 + 0,1 * [5 + 0,9 * 0.8542 - 0.5258]

Q(1,1) = 1,050098

Q(s,a)	Action			
	0	1	2	….
0	0.04214	0.02314	0.0542	0.1234
State 1	0.4255	1,050098	-0.425	-0.01
2	0.098	0.5742	0.8542	0.4924
….	0.7548	-0.9582	0.1234	0.05732

Et on répète ce processus jusqu'à ce que notre q_table ait une policy optimale.

3.3 Partie matériel :

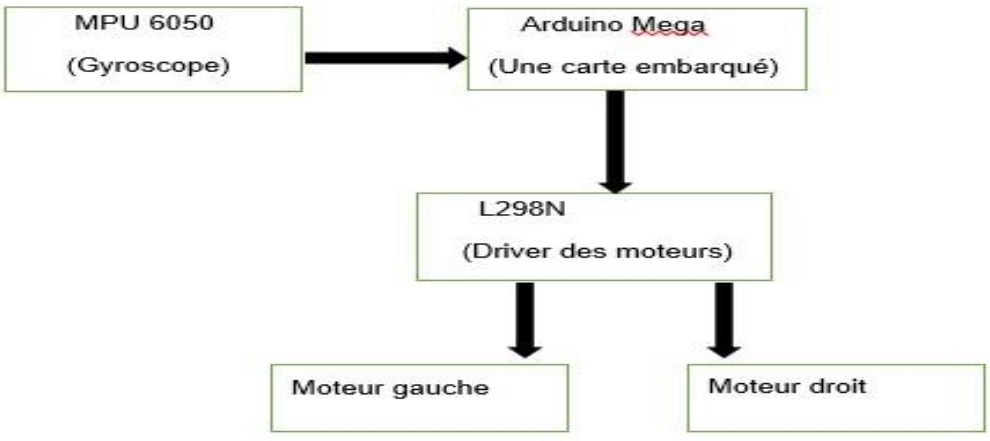
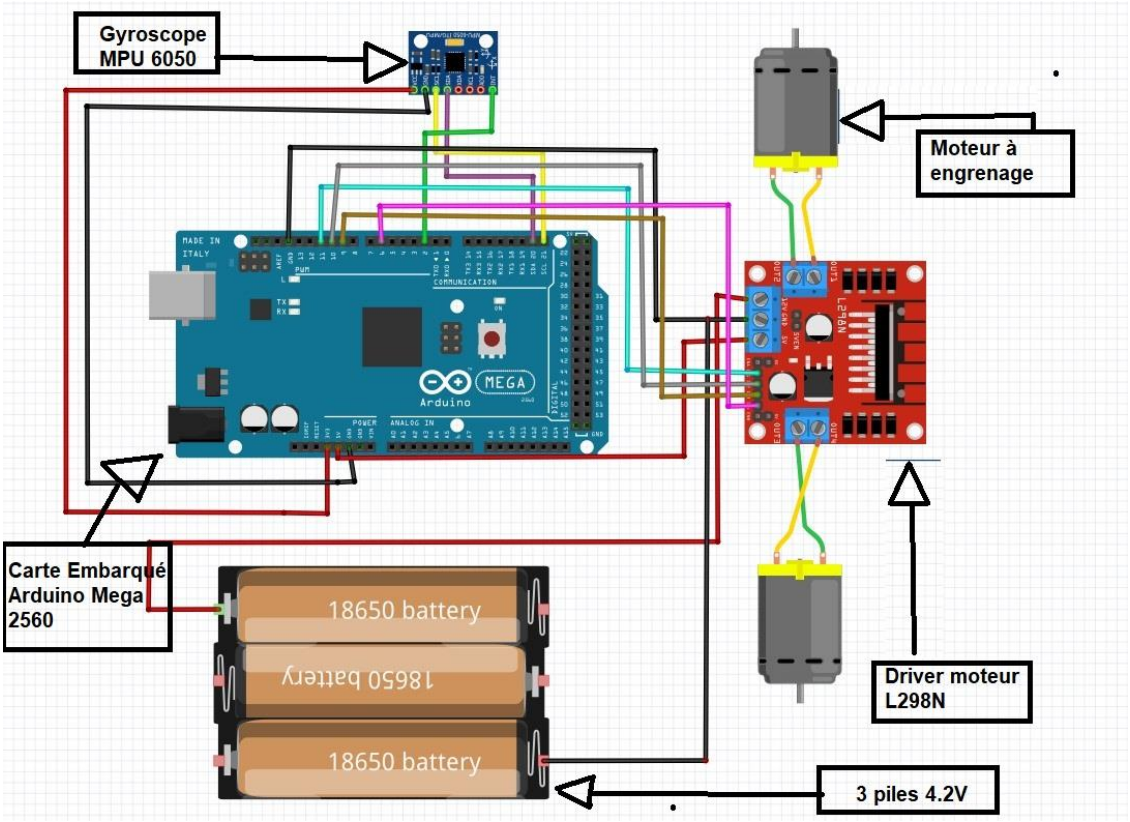


Figure 3.6 Organigramme pour le schéma matériel

3.3.1 Schéma matériel de la plateforme



3.4 Conclusion

En conclusion, dans ce chapitre, nous avons apporté une clarté essentielle à notre projet en utilisant des organigrammes détaillés pour représenter et décrire chaque aspect. Ces organigrammes ont permis d'illuminer notre compréhension globale de la solution, en mettant en évidence les différentes étapes et les composants utilisés. Grâce à cette approche, nous avons pu visualiser de manière précise la structure et les interactions de notre projet, ce qui nous a aidés à prendre des décisions éclairées et à anticiper d'éventuels problèmes.

Chapitre 04 :
Implémentation et Résultats

4.1 Introduction

Dans ce chapitre nous allons détailler la partie hardware (les étapes pour réaliser le robot), on va présenter chaque composant matériel qu'on a utilisé, et de donner les résultats de notre projet.

4.2. Présentation des composants

Nous allons présenter chaque composant qu'on a utilisé pour notre robot :

4.2.1 L'arduino MEGA :

Est une carte de développement open-source basée sur le microcontrôleur ATmega2560. Elle est conçue pour faciliter la création de projets électroniques interactifs et programmables. Il s'agit d'une version avancée de l'Arduino Uno, offrant une plus grande capacité de mémoire (256 KO), elle dispose de 54 broches d'entrée/sortie numériques, de 16 entrées analogiques, de 4 ports UART (Universal Asynchronous Receiver/Transmitter), d'un oscillateur à quartz de 16 Mhz, d'une connexion USB, d'un connecteur d'alimentation, et bien plus encore. L'arduino Mega 2560 est compatible.

Elle assume un rôle fondamental en tant que centre névralgique de notre robot, étant responsable à la fois de la régulation des tensions, de la réception et du traitement des données, ainsi que de l'exécution des instructions préalablement programmées dans notre environnement de développement intégré (IDE), en coordination avec les autres cartes impliquées dans le système, ça sera le cerveau de notre robot.

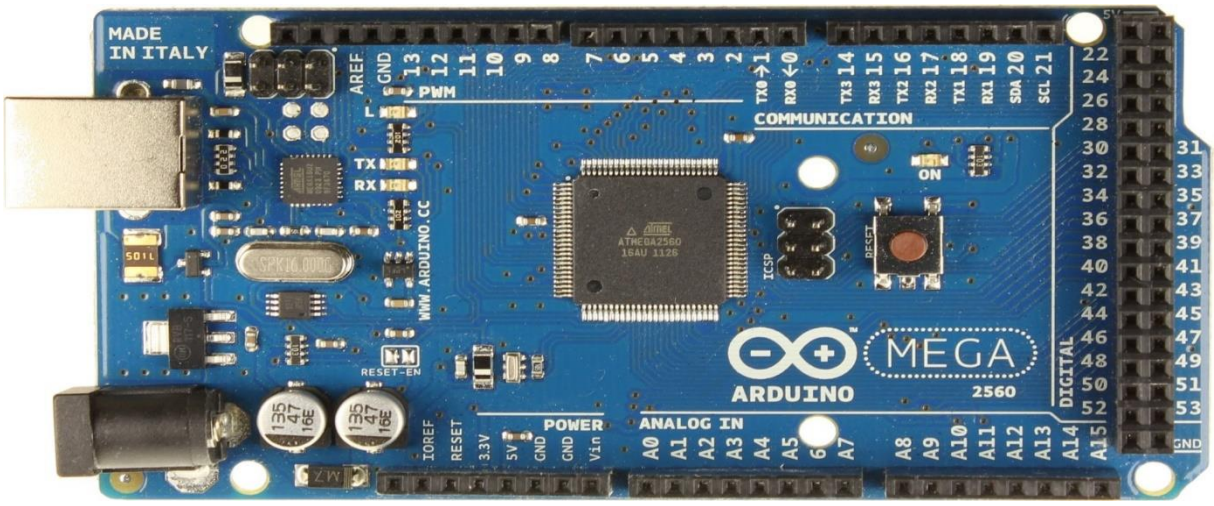


Figure 4.2 Figure de l'arduino mega 2560

4.2.2 MPU6050 :

Le MPU6050 est un capteur d'inertie à six axes qui mesure l'accélération et la vitesse angulaire dans les trois axes de l'espace : **Yaw**, **Pitch** et **Roll**. Il est souvent utilisé dans les applications de contrôle de mouvement, de navigation et de stabilisation. Le capteur est capable de mesurer des mouvements très précis et peut être utilisé pour détecter des changements de position, d'orientation et de mouvement. Le MPU6050 est compatible avec plusieurs interfaces de communication, y compris I2C et SPI, et peut être programmé en utilisant différents langages de programmation tels que C++, Python, C et bien d'autres.

.Le module dispose également de deux broches auxiliaires pouvant être utilisées pour interconnecter des modules I2C externes tels qu'un magnétomètre, cependant, cela est facultatif. Étant donné que l'adresse I2C du module est configurable, il est possible d'interconnecter plusieurs capteurs MPU6050 à un microcontrôleur en utilisant la broche AD0. Ce module dispose également de bibliothèques bien documentées et révisées, ce qui le rend très facile à utiliser avec des plateformes populaires telles que Arduino. Ainsi, si vous recherchez un capteur pour contrôler les mouvements de votre voiture RC, drone, robot auto-équilibré, humanoïde, bipède ou quelque chose du même genre, alors ce capteur pourrait être le bon choix pour vous.

VCC : broche d'alimentation pour le capteur, généralement connectée à une source d'alimentation de 3,3 V ou 5 V.

GND : broche de mise à la terre pour le capteur, généralement connectée à la masse du circuit.

SDA : broche de données série bidirectionnelle pour la communication I2C.

SCL : broche d'horloge série pour la communication I2C.

XDA : broche de données série bidirectionnelle pour la communication I2C supplémentaire.

XCL : broche d'horloge série pour la communication I2C supplémentaire.

AD0 : broche d'adresse qui permet de sélectionner l'adresse I2C du MPU6050.

INT : broche d'interruption qui peut être utilisée pour signaler une condition d'interruption, telle qu'une mesure de mouvement ou une détection de choc.

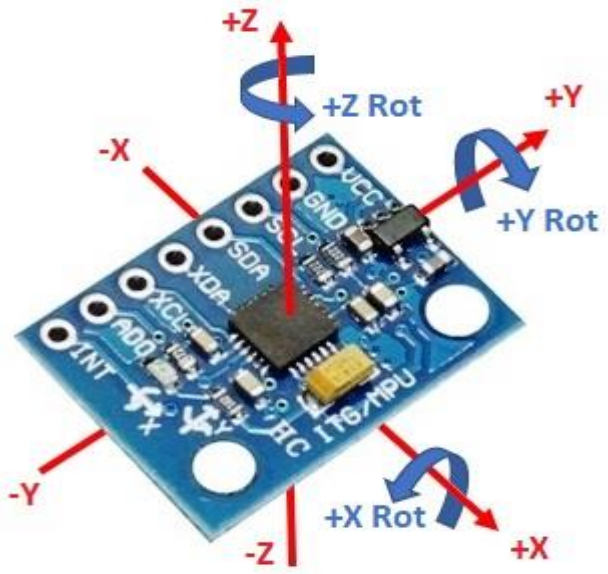


Figure 4.3 Figure montrant le gyroscope MPU6050 avec les 6 axes

4.2.3 Moteur à courant continu :

Est un type de moteur électrique qui convertit l'énergie électrique en mouvement mécanique. Il fonctionne en utilisant un champ magnétique créé par des aimants permanents ou des électroaimants pour faire tourner un rotor qui est alimenté en courant continu. Le courant électrique est fourni au rotor par des brosses qui établissent un contact électrique avec les collecteurs du rotor. Les moteurs à courant continu sont largement utilisés dans les applications industrielles, automobiles, aéronautiques et de robotique en raison de leur faible coût, de leur efficacité énergétique élevée et de leur capacité à fournir un couple élevé à des vitesses de rotation variables.

Figure 4.4 Figure montrant un moteur à courant continu

4.2.4 Driver L298N :

Le driver L298N est un circuit intégré couramment utilisé pour contrôler les moteurs à courant continu (DC). Il permet une commande bidirectionnelle des moteurs DC, ce qui signifie qu'il peut contrôler leur vitesse et leur direction de rotation. Le L298N est équipé de ponts en H intégrés, qui sont des circuits électroniques conçus pour inverser la polarité de l'alimentation fournie aux moteurs. Cela permet de faire tourner les moteurs dans les deux sens et de contrôler leur vitesse en modulant la tension d'alimentation. Il est capable de piloter des moteurs de puissance moyenne à élevée grâce à sa grande capacité de courant.

OUT1, **OUT2**, **OUT3**, **OUT4** : ces branches sont les sorties pour le contrôle d'un moteur à courant continu. Elles sont utilisées pour fournir la tension et le courant nécessaires aux deux moteurs.

IN1, **IN2**, **IN3**, **IN4** : ce sont les broches d'entrée pour contrôler la direction de rotation du moteur connecté aux sorties OUT1 et OUT2 (pour IN1 et IN2) et OUT3 et OUT4(pour IN3 et IN4). En fournissant des signaux logiques HIGH (1) ou LOW (0) à ces broches, il est possible d'inverser la direction de rotation.

IN1	**IN2**	**La direction**
LOW	LOW	Moteur éteint
HIGH	LOW	Avancer
LOW	HIGH	Reculer
HIGH	HIGH	Moteur éteint

ENA,**ENB** : Ces broches d'entrée sont utilisées pour contrôler la vitesse du moteur. En fournissant un signal de modulation de largeur d'impulsion(PWM) à ces broches. ENA est généralement utilisée pour le contrôle du moteur connecté aux broches IN1 et IN2, tandis que ENB est utilisée pour le contrôle du moteur connecté aux broches IN3 et IN4.

L'entrée 12v: C'est pour alimenter le driver L298N.

GND : C'est pour la mise à la terre.

L'entrée 5v(sortie) : Sert soit à alimenter le driver(entrée), soit à alimenter une autre carte(Sortie).

Voici la figure montrant le driver L298N :

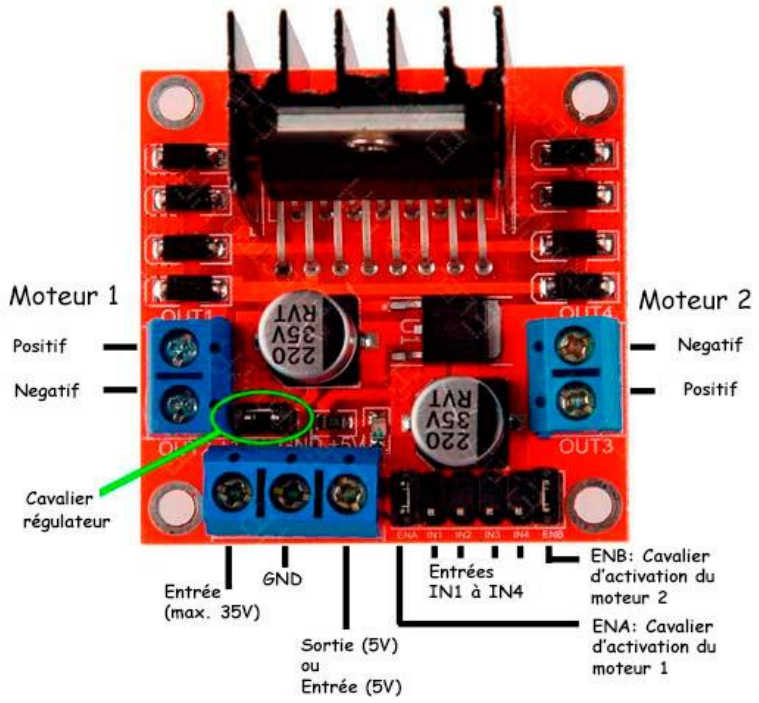


Figure 4.5 Figure montrant le driver L298N

4.2.5 Roue :

Dans notre cas nous avons pris des roues en plastique.

Figure 4.6 Figure montrant une roue en plastique

4.2.6 Piles 4.2V :

On a utilisé 3 piles de types GT LINE rechargeable 18650 4.2V pour alimenter le driver L298N et qui va lui-même alimenter l'arduino.

Figure 4.7 Figure montrant une pile 4.2V GT LINE

4.3 Le langage utilisé :

Le langage de programmation de l'arduino est basé sur les langages de programmation C-arduino/C++ arduino, mais dans notre cas on a utilisé le langage C-arduino. Ce langage est une variante du langage C qui a été spécialement conçue pour être utilisée avec les cartes arduino, la seule différence entre ces deux langages c'est que le C-arduino contient des fonctions supplémentaires qui facilitent la programmation des microcontrôleurs.

4.4 L'ide utiliser :

L'IDE Arduino (Integrated Développent Environment) est un logiciel open-source utilisé pour programmer les cartes Arduino. Il fournit un environnement de développement convivial pour écrire, téléverser et déboguer du code sur les cartes Arduino. L'IDE Arduino est basé sur le langage de programmation Processing et utilise une version modifiée du compilateur GCC. Il est disponible pour Windows, Mac OS X et Linux. L'IDE Arduino est conçu pour être facile à utiliser pour les débutants, mais il offre également des fonctionnalités avancées pour les utilisateurs expérimentés.

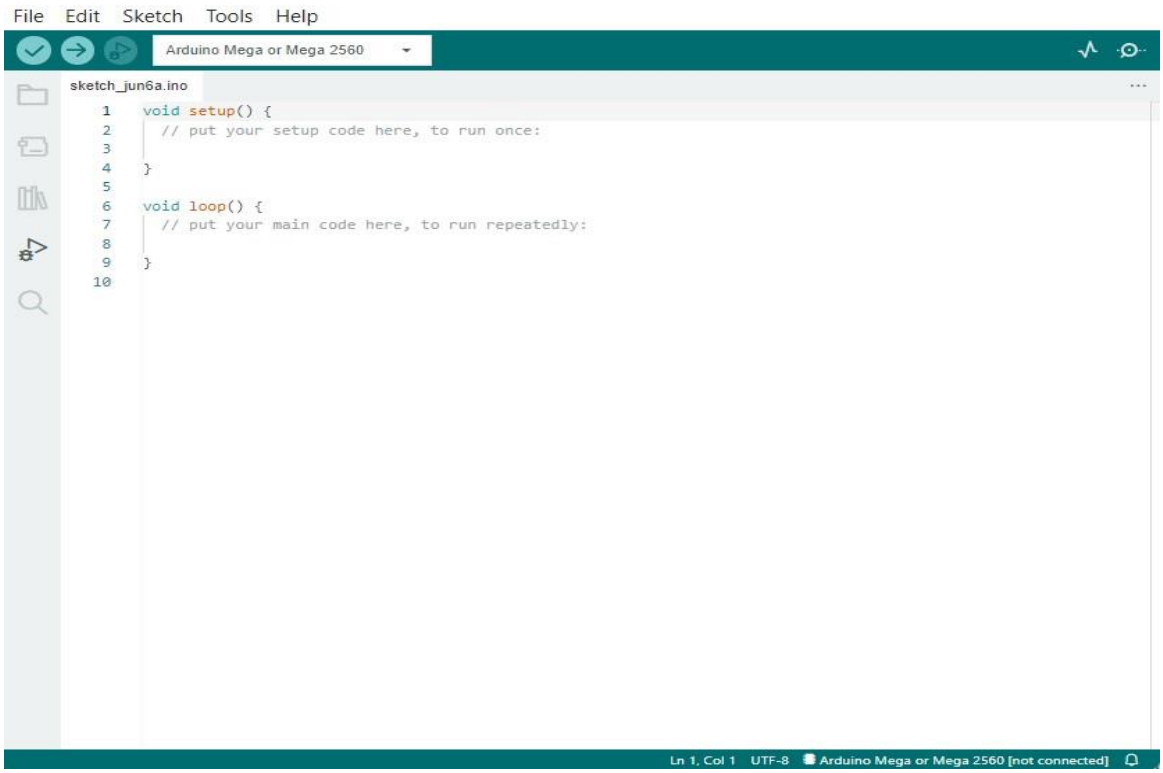


Figure 4.8 L'IDE arduino

4.5 Simulation :

Pour la simulation, nous avons utilisé l'environnement CartPole-v1 gym d'open AI, Gym est une boîte à outils pour développer et comparer les algorithmes d'apprentissage par renforcement. Il permet d'enseigner tout aux agents, de la marche à des jeux comme le pong ou le flipper[23]

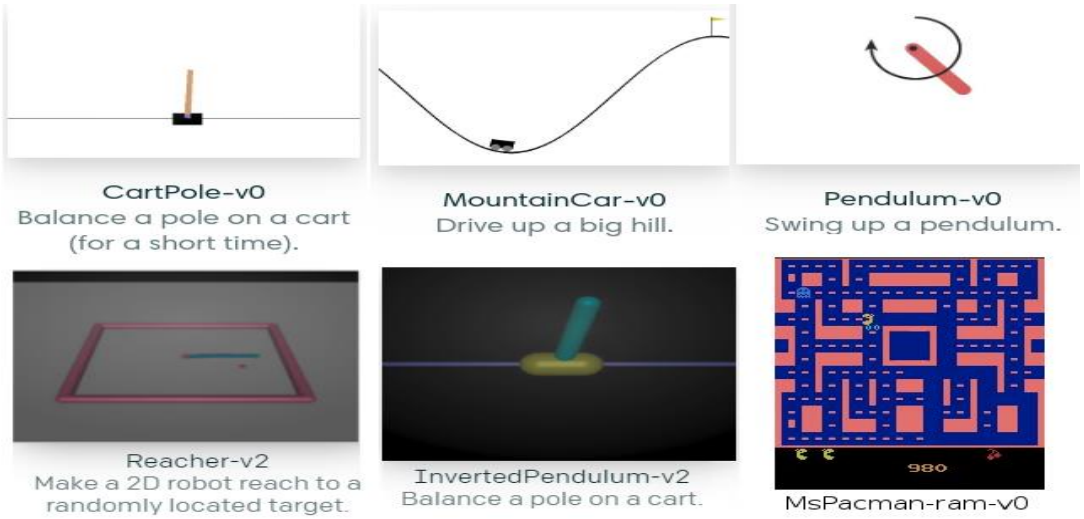


Figure 4.9 Environnement OpenAI Gym

4.6 Les caractéristiques du robot :

Hauteur = 27 cm.

Largeur = 22.7 cm.

Epaisseur des roues = 2-3 millimètres.

Diamètres des roues = 6.5 cm .

Poids = 0,883kg

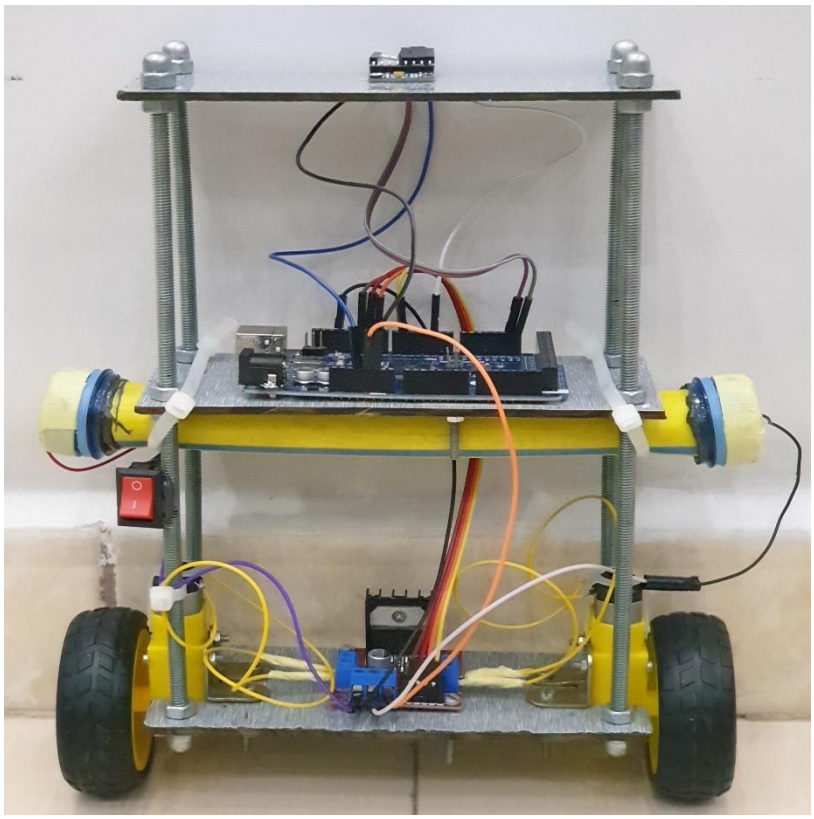


Figure 4.10 Figure montrant notre plateforme

4.7 Résultat

4.7.1 Simulation d'un pendule inversé sur gym

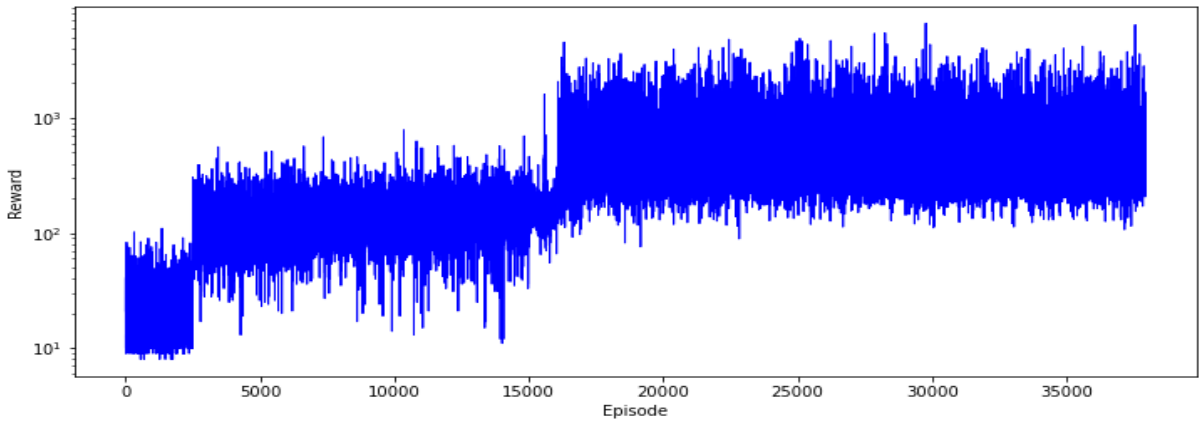


Figure 4.11 Entrainement CartPole.

Explication

Nous avons fait plusieurs simulations avec différents paramètres et nous avons retenue l'expérience qui nous a donné le meilleur résultat, dans la **Figure 4.11** nous avons mis : alpha =0.1, gamma= 1, epsilon = 0.2, nombre d'épisodes = 40000.

Comme nous pouvons le constater, au début de l'apprentissage, notre agent effectue des actions de manière aléatoire, ce qui se traduit par un nombre limité de récompenses. Cependant, au fur et à mesure que le temps passe et que les épisodes se)d succèdent, notre algorithme commence à converger et à produire de meilleurs résultats. En d'autres termes, notre agent obtient des performances de plus en plus satisfaisantes jusqu'à atteindre une convergence optimale.

4.7.2 Résultat du pid pour équilibrer notre robot

Nous avons choisi comme paramètres :

Kp = 20

KD = 0.9

KI = 140

La **Figure 4.12** ci-dessous représente le résultat de notre expérience pour essayer d'équilibrer un robot à deux en utilisant un contrôleur PID :

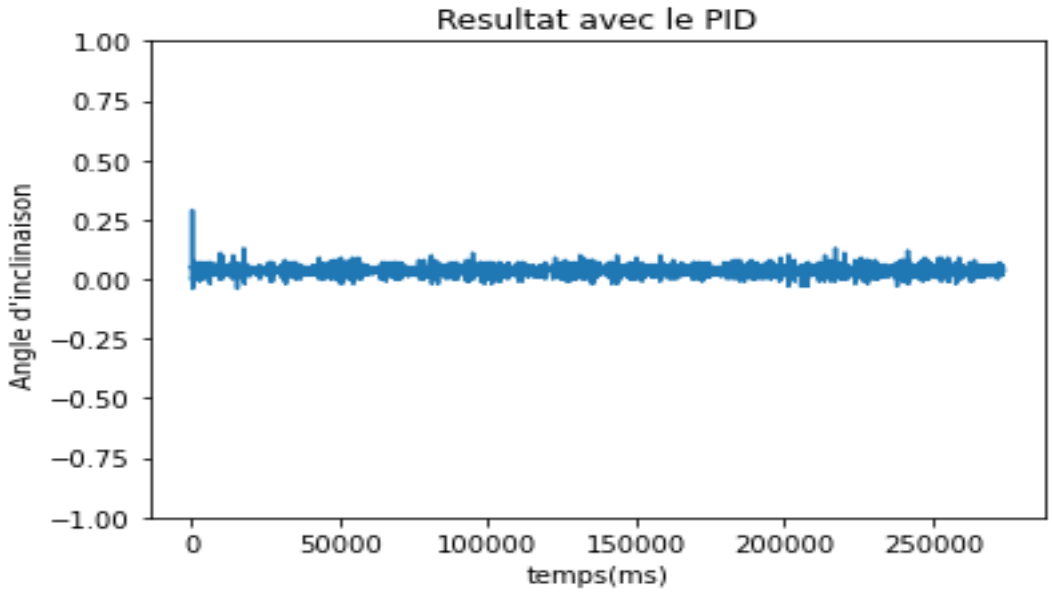


Figure 4.12 Entrainement CartPole.

Explication

La figure représente l'angle d'inclinaison de notre robot au fil du temps, nous avons laissé notre robot en marche pendant quasiment 5 minutes (4 minutes et 54 secondes plus exactement), et comme on peut le voir notre robot a réussi à s'équilibrer car l'angle varie entre 0,06 et -0.04 qui sont proche de 0.

4.8 Conclusion

En conclusion, après avoir exploré différentes solutions, nous avons constaté que l'utilisation du Q-learning sur Arduino n'était pas optimale en raison des contraintes matérielles. La nécessité de discrétiser les vitesses et les états avec des valeurs réduites pour le traitement de l'Arduino a entraîné une détérioration des performances de notre robot, l'empêchant ainsi de maintenir son équilibre.

Cependant, nous sommes convaincus que les performances du contrôleur PID peuvent être améliorées en exploitant les avantages de l'adaptabilité offerte par l'algorithme Q-learning.

Pour des perspectives futures, il serait judicieux d'envisager l'utilisation d'une carte Raspberry Pi. Avec sa puissance de calcul supérieure et sa capacité de stockage suffisante, elle permettrait de supporter des algorithmes d'apprentissage plus avancés. En optant pour un algorithme tel que le Deep Q-Network, spécialement conçu pour gérer des valeurs continues, nous pourrions considérablement améliorer les performances du contrôleur

Conclusion générale

Dans le cadre de notre projet de recherche, nous avons employé une combinaison de deux approches, le contrôleur PID ainsi qu'un algorithme d'apprentissage par renforcement qui est le Q-learning, dans le but de stabiliser une plateforme robotique à deux roues. L'objectif principal consistait à améliorer les performances du contrôleur en lui conférant une adaptabilité accrue face à des perturbations externes et à des situations imprévues.

Malheureusement, malgré nos efforts pour exploiter pleinement le potentiel de l'apprentissage par renforcement, nous avons dû faire face à des contraintes matérielles qui ont limité nos possibilités. Les exigences en termes de puissance de calcul et de mémoire pour obtenir des améliorations significatives étaient tout simplement incompatibles avec les ressources disponibles sur notre plateforme Arduino.

En conséquence, nous avons été contraints de nous concentrer sur l'utilisation du contrôleur PID et de mettre de côté l'apprentissage par renforcement dans ce contexte spécifique. Bien que le PID ait permis d'obtenir une certaine stabilité, il est important de noter que l'apprentissage par renforcement aurait pu offrir des performances supérieures et une adaptation plus dynamique aux changements de conditions.

Pour des travaux futurs, il serait nécessaire de passer à une plateforme plus puissante, telle qu'un Raspberry Pi, capable de répondre aux exigences en termes de calcul et de mémoire pour exécuter efficacement des algorithmes d'apprentissage par renforcement plus avancés, tels que le Deep Q-Network (DQN). L'utilisation du DQN permettrait d'explorer davantage les avantages de l'apprentissage par renforcement et d'améliorer les performances de stabilisation de la plateforme robotique.

Bibliographie

Références

[1] : Terrier, L. (2015, avril 21). Récupéré sur Toiledefond: https://toiledefond.net/le-machine-learning-quand-les-ordinateurs-predisent-lavenir/

[2] : Tom M. Mitchell (1997). Récupéré sur http://www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf

[3] : Benzaki , Y. (2017, Février 19). Récupéré sur mrmint: https://mrmint.fr/apprentissage-supervise-machine-learning

[4] : Juvénal JVC. (s.d.). Récupéré sur https://www.data-transitionnumerique.com/machine-learning-python/

[5] : Saman Siadati (2018, Aout) Récupéré sur https://www.researchgate.net/publication/342121950_What_is_unsupervised_Learning

[6] : Benzaki, Y. (2017, Mars 01). Récupéré sur mrmint: https://mrmint.fr/lapprentissage-non-supervise-machine-learning

[7] : Romeyssa, M. (2017). Récupéré sur theses-algerie: https://www.theses-algerie.com/3762970674582005/memoire-de-licence/universite-kasdi-merbah-ouergla/modelisation-floue-et-commande-pdc-dun-systeme-non-lineaire

[8] : Houda Maâmatou (2018, Juin 06). Récupéré sur : https://theses.hal.science/tel-01809530/

[9] : Ziyu Zhang (2023). Récupéré sur : https://www.researchgate.net/publication/372213808_Basic_things_about_reinforcement_learning

[10] : Manseur Ali (2019, Juin). Récupéré sur : http://dspace.univ-tebessa.dz:8080/jspui/bitstream/123456789/1831/1/memoire%20finale%202019.pdf

[11] : Récupéré sur : https://www.larousse.fr/dictionnaires/francais/%C3%A9quilibre/30674#:~:text=1.,d%27une%20balance%20en%20%C3%A9quilibre.&text=2.,peine%20%C3%A0%20garder%20son%20%C3%A9quilibre.

[12] : Récupéré sur : Erwan Renaudo. (s.d). Récupéré sur : https://www.researchgate.net/figure/Probleme-du-pendule-inverse-ou-cartpole-problem-decrit-notamment-dans-Sutton-et-Barto_fig2_317162679

[13] : (s.d.). Récupéré sur iutenligne : https://public.iutenligne.net/etudes-et realisations/nardi/Segway/Principe/index.html

[14] : Pritesh Shah, Sudhir Agashe (2016). Récupéré sur : https://www.sciencedirect.com/science/article/abs/pii/S095741581630068X

[15] : Récupéré sur : Ouenzar, B. (2013). Récupéré sur wikimemoires: https://wikimemoires.net/2022/03/regulation-pid-qualites-attendues-et-type-de-regulateurs/

Bibliographie

[16] :Piette, F. (2011, Août 5). implementer-un-pid-sans-faire-de-calculs. Récupéré sur ferdinandpiette: http://www.ferdinandpiette.com/blog/2011/08/implementer-un-pid-sans-faire-de-calculs/

[17] : Wikipedia. (s.d.). Récupéré sur https://fr.wikipedia.org/wiki/R%C3%A9gulateur_PID.

[18] : Ludvig, L., & David , A. (2022). Récupéré sur http://kth.diva-portal.org/smash/get/diva2:1737768/FULLTEXT01.pdf

[19] : TechTarget, L. R. (2018, Avril). Récupéré sur https://www.lemagit.fr/definition/logique-floue

[20] : CHEVRIE, F., & GUÉLY, F. (1998, Mars). Récupéré sur epfi: https://www.epfi.fr/Files/Other/newsletters-juin-2017-ct191.pdf

[21] : Simon, J. (2023, Avril 10). Récupéré sur researchgate: https://www.researchgate.net/publication/369936909_Fuzzy_Control_of_Self-Balancing_Two-Wheel-Driven_SLAM-Based_Unmanned_System_for_Agriculture_40_Applications

[22] : Ziru Lun, Zhanyu Ye (2023, Juillet). Récupéré sur : https://www.researchgate.net/publication/372206619_A_Bilevel_Genetic_Algorithm_for_Global_Optimization_Problems

[23] : (s.d.). Récupéré sur mathworks: https://www.mathworks.com/help/gads/how-the-genetic-algorithm-works.html

[24] : ALI, L. (2017, Juin). Récupéré sur oraprdnt.uqtr.uquebec: https://oraprdnt.uqtr.uquebec.ca/pls/public/docs/FWG/GSC/Publication/1645/34/1918/1/170011/8/O0000330598_M_moire__d_pot_final_.pdf

[25] : (s.d.). Récupéré sur igm.univ-mlv: http://igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/fonctionnement.html

[26] : Triantafyllou, M. (s.d.). Récupéré sur kostasalexis: http://www.kostasalexis.com/lqr-control.html

[27] : Naim, A. (s.d.). Récupéré sur github: https://github.com/aaftabnaim/genetic_self_balancing_robot_webots

[28] : L, B. (2021, Juin 30). Récupéré sur lebigdata: https://www.lebigdata.fr/reinforcement-learning-definition

[29] : Max Fischer (2019). Récupéré sur : http://www.diva-portal.org/smash/get/diva2:1334684/FULLTEXT01.pdf

[30] : Richard S. Sutton, Andrew G. (2018) Barto https://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

Bibliographie

[31] : MD Muhaimin, R., S. M. Hasanur , R., & M. M. , H. (2018, Decembre 21). Récupéré sur jrobio.springeropen: https://jrobio.springeropen.com/articles/10.1186/s40638-018-0091-9

[32] : JUTON, A., NOËL, V., & LALI, R. (2022, Juillet 20). Récupéré sur eduscol.education.fr: https://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/pedagogiques/14756/14756-introduction-lapprentissage-par-renforcement-ensps.pdf

[33] : Christopher J. C. H. Watkins & Peter Dayan. (1992) Récupéré sur https://link.springer.com/article/10.1007/BF00992698

[34] : baeldung. (2023, Mars 24). Récupéré sur baeldung: https://www.baeldung.com/cs/epsilon-greedy-q-learning#:~:text=The%20epsilon%2Dgreedy%20approach%20selects,what%20we%20have%20already%20learned.

[35] : Paul Gautier (2021, Juillet 02). Récupéré sur : https://theses.hal.science/tel-03276733

[36] : (s.d.). Récupéré sur : https://datascientest.com/q-learning-le-machine-learning-avec-apprentissage-par-renforcement

[37] : Mayank Banoula. (s.d.). Récupéré sur : https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning

[38] : Melrose Roderick, James MacGlashan, Stefanie Tellex. (2017, Novembre 20) Récupéré sur : https://arxiv.org/pdf/1711.07478.pdf

[39] : Hongzi Mao, Mohammad Alizadeh, Ishai Menache, Srikanth Kandula. (s.d.). Récupéré sur : http://people.csail.mit.edu/hongzi/content/publications/DeepRM-HotNets16.pdf

[40] : (s.d.). Récupéré sur : https://dataanalyticspost.com/Lexique/reseau-de-neurones-profonds/

[41] : Jérémy Lecoeur. (s.d.). Récupéré sur : https://www.researchgate.net/figure/Structure-dun-neurone-artificiel-Le-neurone-calcule-la-somme-de-ses-entrees-puis-cette_fig14_29640044

[42] : Bastien Maurice. (2018, Septembre 26) Récupéré sur : https://deeplylearning.fr/cours-theoriques-deep-learning/fonction-dactivation/

[43] : Ovanes Petrosian. (s.d.). Récupéré sur : https://www.researchgate.net/figure/algorithm-of-deep-Q-learning_fig3_344238597

[44] : (s.d.). Récupéré sur : https://h2o.ai/wiki/forward-propagation/

[45] : (s.d.). Récupéré sur : https://www.ibm.com/topics/gradient-descent

[46] : Ruishan Liu, James Zou. (2018, Octobre 02) Récupéré sur : https://proceedings.allerton.csl.illinois.edu/2018/media/files/0091.pdf

[47] : Dominic Masters, Carlo Luschi. (2018, Avril 20) Récupéré sur : https://arxiv.org/pdf/1804.07612.pdf

[48] : (s.d.). (s.d.). Récupéré sur : https://www-igm.univ-mlv.fr/~dr/XPOSE2014/Machin_Learning/A_Sarsa.html

Bibliographie

[49] : nova. (2023, Avril 24). Récupéré sur aitechtrend: https://aitechtrend.com/all-you-need-to-know-about-sarsa-in-reinforcement-learning/

[50] : Récupéré sur : https://proceedings.neurips.cc/paper_files/paper/2019/hash/9f9e8cba3700df6a947a8cf91035ab84-Abstract.html

[51] : Caglar Uyulan. (2023, Mai) Récupéré sur : https://www.researchgate.net/publication/371077257_Control_Methods_for_Inverted_Pendulum_on_a_Cart

[52] : Aleksandar Haber.(2023, Janvier 31) Récupéré sur : https://aleksandarhaber.com/cart-pole-control-environment-in-openai-gym-gymnasium-introduction-to-openai-gym/

Annexe

Vidéo YouTube de la conception de notre robot :

Equilibrer un robot à deux roues avec un contrôleur PID - YouTube

Résumé :

Le projet visait à étudier et à appliquer une approche combinant un régulateur PID (Proportionnel, Intégral, Dérivé) et l'apprentissage par renforcement pour équilibrer une plateforme robotique à deux roues. L'objectif était de renforcer les capacités du PID et de concevoir un robot capable de s'adapter aux perturbations externes.

Les résultats obtenus pour l'équilibrage du robot à l'aide du régulateur PID ont été satisfaisants, le robot parvenait à maintenir son équilibre sur les deux roues sans difficulté. Cependant, lors de l'intégration de l'apprentissage par renforcement sur l'Arduino, des contraintes matérielles ont limité les performances. Les résultats obtenus avec cette combinaison n'ont pas été satisfaisants

Abstract:

The project aimed to study and implement an approach combining a PID controller (Proportional, Integral, Derivative) with reinforcement learning to balance a two-wheeled robotic platform. The objective was to enhance the capabilities of the PID controller and design a robot capable of adapting to external disturbances.

The results obtained for balancing the robot using the PID controller were satisfactory, as the robot was able to maintain its balance on the two wheels without difficulty. However, during the integration of reinforcement learning on the Arduino platform, hardware constraints limited the performance. The results obtained with this combination were not satisfactory.