

ÉCOLE CENTRALE CASABLANCA



Cahier des Charges : Projet : Robot Mobile Autonome pour la Collecte et le Stockage Intelligent

Encadré par :
Mr Abdelkader EL KAMEL

Réalisé par :
Bouzidi Safa
Cheikh Ayette
Khalfa Youssef
Trabelsi Mohamed Yassine

26 avril 2025

Table des matières

1	Introduction	3
1.1	Bref historique des robots d'entrepôt	3
1.2	Notre cas d'Application : Automatisation en Environnement Pharmaceutique	3
2	Objectif	4
3	Problématique	4
4	Structure et organisation fonctionnelle du robot	5
4.1	Module mobile (châssis roulant)	5
4.2	Modules fonctionnels additionnels	7
4.2.1	Capteurs infrarouges (IR)	7
4.2.2	Module driver moteur (pont en H)	9
4.3	Module supérieur (unité de manipulation)	10
4.4	Description du schéma de déplacement	12
4.5	Scénario de déplacement et de manipulation	13
5	Affichage de l'état via un afficheur 7 segments	15
5.1	Intérêt de l'afficheur 7 segments	15
5.2	Problème rencontré : manque de broches disponibles	16
5.3	Solution adoptée : utilisation d'un registre à décalage	16
6	Algorithme embarqué sur Arduino Uno	17
6.1	Présentation de la carte Arduino Uno	17
6.2	Explication des fonctions définies	17
6.2.1	Boucle principale	17
6.2.2	Fonction touch5()	18
6.3	Stratégie de réutilisation : simplification du code	19
6.4	Fonctions de Roulement	19
6.5	Suivi de ligne et détection d'arrêt	20
6.6	Enchaînement général du programme	20
7	Simulation MATLAB : Déplacement et Animation du Bras	20
7.1	Objectifs de la simulation	21
7.2	Trajectoire et Contrôle	21
7.3	Modélisation du bras articulé	22
7.4	Résultats visuels de la simulation	22
7.5	Extrait de code représentatif	22
7.6	Conclusion	23

8 Matériel Utilisé	23
9 Livrables	23
10 Valeur Ajoutée par Rapport au Cours	24
11 Perspectives d'Évolution	25
12 Conclusion	25
13 Bibliographie	25

1 Introduction

Dans le contexte de l'Industrie 4.0, les entrepôts intelligents cherchent à automatiser les tâches de transport, de tri et de placement des objets dans des casiers de stockage. Le défi est de concevoir un robot autonome capable de naviguer sur un trajet défini, de collecter un objet à un point précis, puis de le déposer dans un emplacement déterminé selon une logique programmable.

1.1 Bref historique des robots d'entrepôt

L'automatisation des entrepôts débute dans les **années 1960** avec l'introduction des premiers *Automated Storage and Retrieval Systems* (AS/RS), destinés à optimiser le stockage vertical. Une avancée majeure est survenue dans les **années 2000** avec l'apparition de robots mobiles autonomes, notamment ceux développés par **Kiva Systems**. En **2012**, l'entreprise est acquise par **Amazon**, donnant naissance à *Amazon Robotics*, pionnière dans le déploiement à grande échelle de flottes de robots logistiques. Aujourd'hui, ces systèmes constituent une brique essentielle des entrepôts intelligents, intégrés aux architectures cyber-physiques de l'Industrie 4.0.



FIGURE 1 – Amazon warehouse robot



FIGURE 2 – amazon robotics

1.2 Notre cas d'Application : Automatisation en Environnement Pharmaceutique

Les environnements pharmaceutiques, hautement régulés et sensibles, exigent une logistique rigoureuse, traçable et sécurisée. Dans ce contexte, un robot mobile autonome équipé d'un système de suivi de ligne et d'un bras de manipulation peut jouer un rôle crucial.

Par exemple, dans un laboratoire ou une unité de production stérile, ce type de robot peut être mobilisé pour :

- transporter des flacons ou échantillons entre des zones de préparation et de stockage à accès restreint .
- réduire les déplacements humains dans les zones à atmosphère contrôlée (ZAC) .
- limiter les risques de contamination croisée .
- automatiser le processus de rangement dans des casiers ou enceintes réfrigérées.



FIGURE 2 – Robot employé par TD Conception

Un cas particulier concerne la **manipulation de produits biologiques ou thermosensibles** nécessitant une conservation à **très basse température** (inférieure à -20°C), condition indispensable à la préservation de leurs propriétés thérapeutiques. Ces environnements sont souvent **insoutenables pour le personnel humain** sur de longues durées. L'intégration de robots autonomes permet alors d'assurer ces opérations critiques de manière fiable, tout en garantissant la sécurité des opérateurs et la conformité aux exigences sanitaires strictes.

2 Objectif

Développer un robot autonome, capable de :

- Réaliser le **transport sécurisé de contenants médicaux** (flacons, échantillons, sachets) entre différentes zones de stockage à température contrôlée.
- Suivre de manière autonome des circuits prédefinis dans des environnements stériles ou sensibles.
- Effectuer la **prise précise d'un objet** dans une zone de collecte, puis son **dépôt rigoureux** dans un emplacement de stockage désigné.
- Réduire l'exposition du personnel à des **conditions extrêmes**, notamment les basses températures ou les environnements à haute exigence sanitaire ;
- Garantir une exécution régulière, fiable et conforme aux normes en vigueur dans le secteur pharmaceutique.

3 Problématique

Problématique

Dans un environnement de type entrepôt intelligent, comment concevoir un robot mobile capable de suivre un trajet fixe, collecter un objet dans un casier spécifique (Mur 1), le transporter et le déposer dans un autre casier (Mur 2), tout en automatisant le processus pour une logistique efficace ?

4 Structure et organisation fonctionnelle du robot

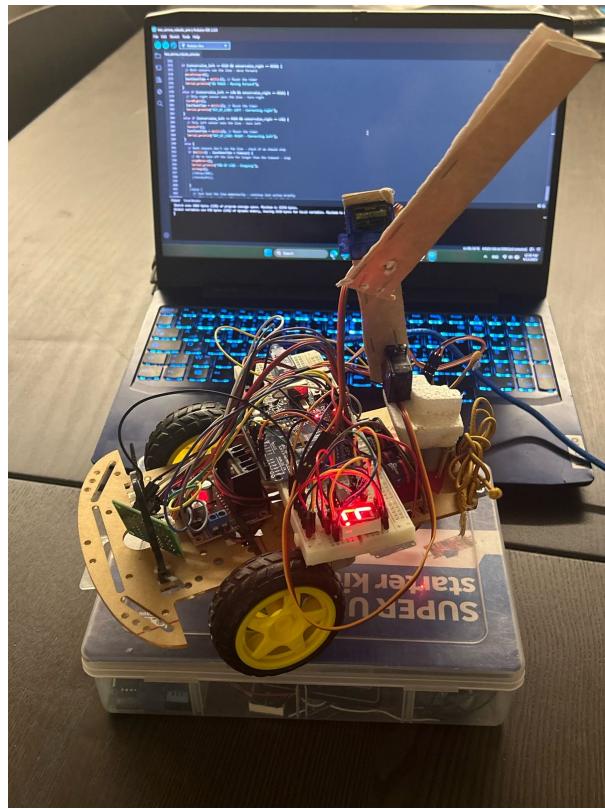


FIGURE 3 – Robot final

Le robot est conçu autour d'une architecture simple, compacte et fonctionnelle, pensée pour naviguer dans un environnement prédéfini et y effectuer des actions ciblées de collecte et de dépôt. Il se compose de deux modules principaux, qui interagissent pour exécuter des cycles logistiques répétés.

4.1 Module mobile (châssis roulant)



FIGURE 4 – Composante mobile

Ce module constitue la base du robot. Il permet le déplacement autonome le long d'un trajet fixe matérialisé au sol (par exemple, une ligne ou un marquage contrasté). Sa

structure stable assure un mouvement fluide, régulier et contrôlé, y compris lors des arrêts pour manipulation.

- Il est optimisé pour circuler dans des espaces restreints (couloirs, travées d'armoires, zones de passage entre casiers).
- Il est capable d'effectuer des arrêts précis à des points stratégiques, correspondant aux emplacements des casiers de collecte ou de dépôt.

Choix du matériel et des composantes électroniques

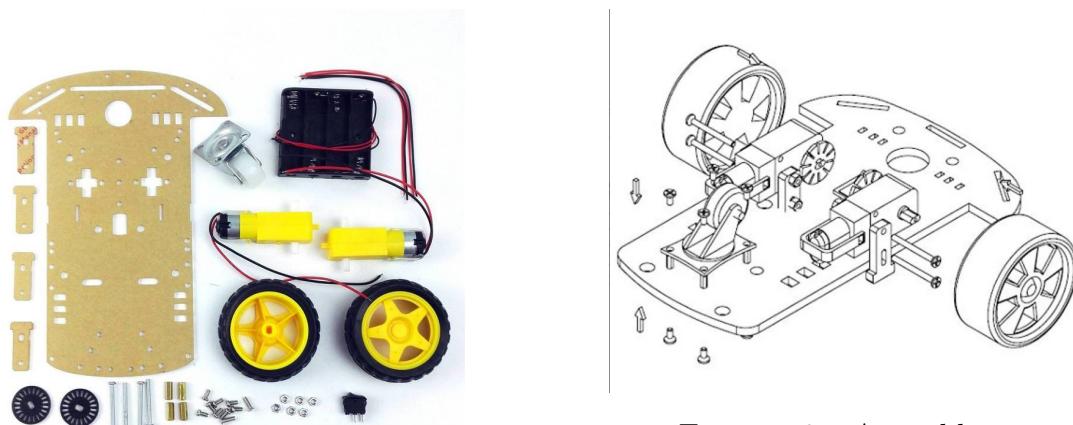


FIGURE 6 – Assemblage

FIGURE 5 – vue générale des composantes principales

Composant	Fonction
Plaque de châssis (en MDF ou acrylique)	Support structurel de l'ensemble du robot, accueille les différents modules mécaniques et de commande
Deux roues motrices	Permettent le déplacement du robot par propulsion
Deux moteurs avec réducteur	Fournissent l'entraînement aux roues avec un couple adapté
Roue folle pivotante	Maintient l'équilibre tout en autorisant des mouvements dans toutes les directions
Support de piles (x4 AA)	Permet l'alimentation mobile du système
Entretoises, vis, écrous	Assurent l'assemblage mécanique de la structure
Disques encodeurs optiques (optionnels)	Utilisables pour le comptage des rotations et la mesure de vitesse
Plaques de fixation latérales	Support mécanique pour les moteurs et éventuellement des capteurs de navigation

TABLE 1 – Composantes principales du châssis robotique

Ce modèle de châssis a été retenu car il répond pleinement aux recommandations

formulées dans le document *AEK_Intro_Smart-Robotics_Jan25*, notamment aux pages 59 et 60, dans le cadre du développement d'un robot mobile intelligent.

- **Simplicité et accessibilité :** Ce châssis à deux roues motrices et roue folle représente une solution éprouvée pour initier des fonctions de navigation autonome sur circuit prédéfini. Sa mécanique minimaliste en fait une plateforme idéale pour des applications logistiques simples, tout en restant extensible.
- **Maniabilité dans des espaces restreints :** Grâce à sa structure différentielle, le robot peut tourner sur place, ce qui est particulièrement adapté à des environnements comme des travées de casiers ou de petites allées. Cela permet d'optimiser les mouvements dans des parcours fixes.
- **Adapté aux missions logistiques répétitives :** Ce châssis est conçu pour exécuter des cycles simples et répétés (ex. : collecte depuis le casier 1, dépôt dans le casier 2), ce qui le rend conforme aux besoins d'automatisation de tâches de type *pick-and-place* dans des environnements structurés comme les laboratoires pharmaceutiques ou les entrepôts.
- **Compatibilité avec l'ajout d'un bras supérieur :** Sa plaque plane offre une base stable et dégagée, parfaitement adaptée pour accueillir un module de manipulation supplémentaire (système de dépôt d'objet).
- **Conformité aux principes de l'Industrie 4.0 :** En tant que plateforme physique simple, modulaire et reprogrammable, ce robot répond aux critères de flexibilité, traçabilité et automatisation progressive définis dans le contexte de la robotique intelligente pour les systèmes logistiques.

4.2 Modules fonctionnels additionnels

Au-delà de la base roulante, certains composants fonctionnels ont été ajoutés afin de permettre au robot d'assurer ses missions de manière autonome, fiable et répétitive. Chaque élément joue un rôle précis dans l'architecture globale.

4.2.1 Capteurs infrarouges (IR)

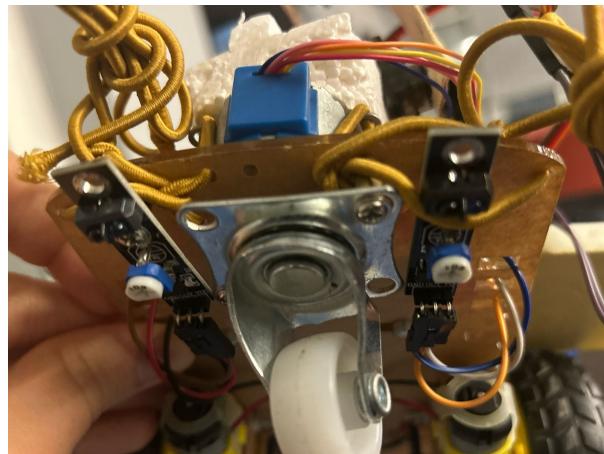


FIGURE 7 – Emplacement des capteurs IR

Les capteurs infrarouges permettent de détecter les contrastes de couleur au sol, notamment entre une ligne noire et un fond clair. Ils sont utilisés pour le suivi de

trajectoire, en lisant en temps réel la position du robot par rapport au tracé défini.

Contribution au système : Ces capteurs sont essentiels à la navigation. Ils permettent

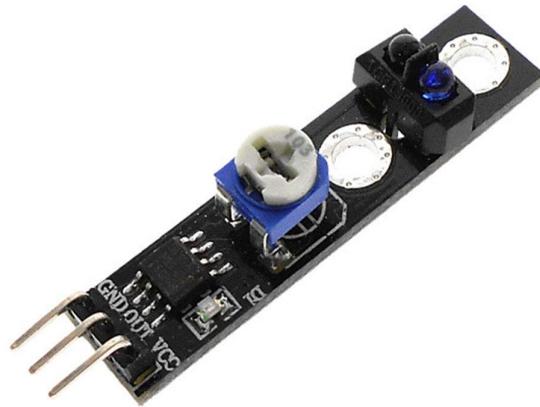


FIGURE 8 – Capteur infra-rouge

au robot de se déplacer de manière autonome sur un parcours fixe en corrigeant sa direction lorsque la ligne est déviée. Cela garantit une exécution précise des cycles de collecte et de dépôt.

fonctionnement des capteurs infra-rouges

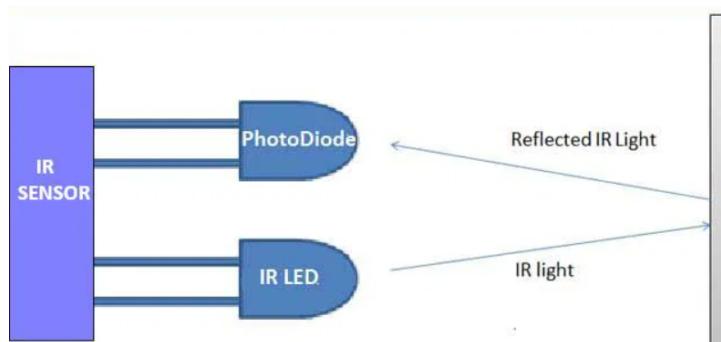


FIGURE 9 – Mécanisme des capteurs IR

4.2.2 Module driver moteur (pont en H)



FIGURE 10 – Emplacement de Driver

Le driver moteur (ou module de commande de moteurs) est un composant électronique qui permet de piloter la rotation des moteurs de manière indépendante. Il gère la mise en marche, l'arrêt, la direction (avant/arrière), et éventuellement la vitesse.

fonctionnement du Driver

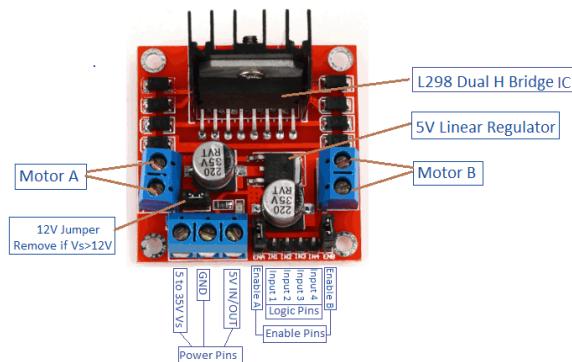


FIGURE 11 – Driver modèle L298N

Contribution au système : Il permet d'appliquer des consignes de déplacement précises et différencierées à chaque moteur. Grâce à ce module, le robot peut effectuer des virages, des rotations sur place, ou des mouvements rectilignes, rendant sa navigation fluide et contrôlée.

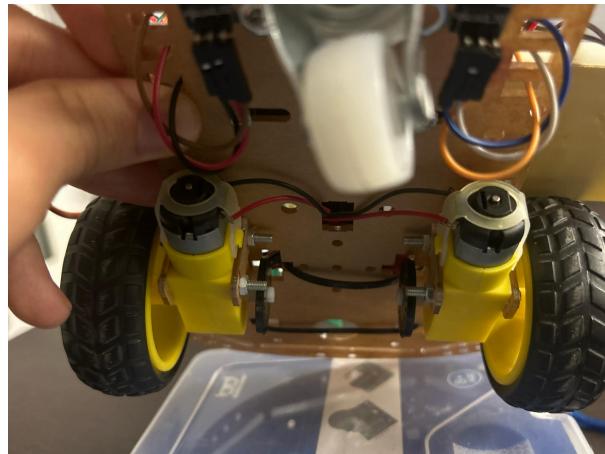


FIGURE 12 – connexion des moteurs (roues)

4.3 Module supérieur (unité de manipulation)

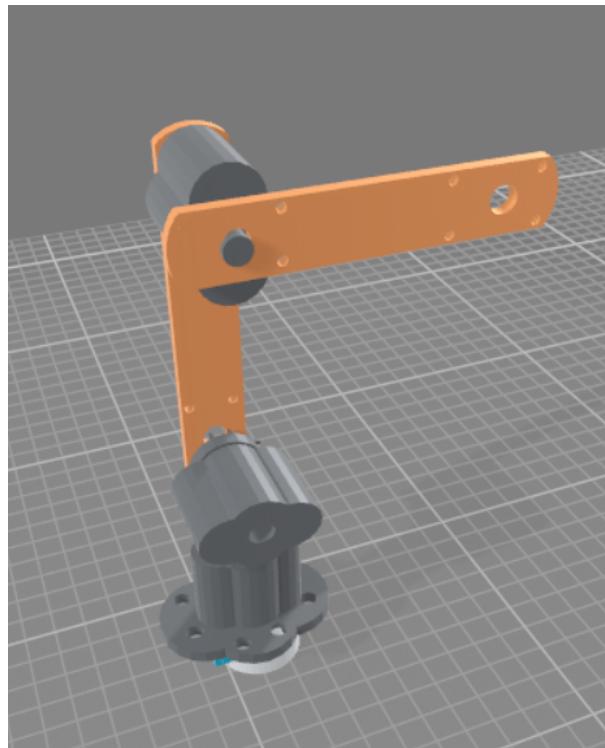


FIGURE 13 – Modèle 3D pour bras automatique

Fixé sur le module mobile, ce sous-ensemble est dédié à la prise et au dépôt des objets. Il est composé d'un système articulé qui permet au robot d'atteindre une zone cible (comme une case d'un casier), de saisir ou relâcher un élément, et d'adapter son mouvement à la hauteur ou la position souhaitée.

- Ce bras assure une interaction fiable avec les casiers de collecte (Mur 1) et de stockage (Mur 2 ou armoire).
- Sa course est pensée pour s'aligner précisément sur les emplacements standards d'un espace de rangement (étagère, bac, tiroir).

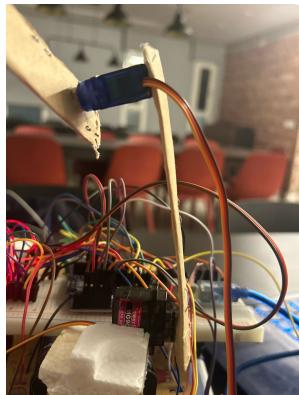


FIGURE 14 – bras dans le robot

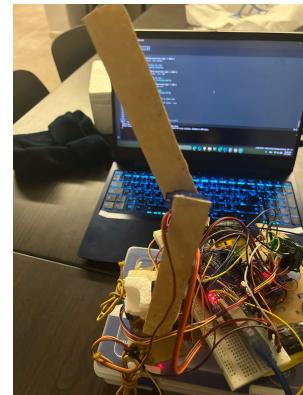


FIGURE 14 – bras avec une position différente

Le bras robotisé installé sur la base mobile permet d'assurer les opérations de prise et de dépôt d'objet dans les casiers définis. Il est constitué d'éléments simples mais fonctionnels, articulés autour de deux degrés de liberté et d'un socle rotatif.

Les composantes principales utilisées dans ce module sont :

Composant	Fonction dans le bras robotisé
Servomoteur MG90/MG90S (180°)	Utilisé pour l'articulation de l'« épaule » du bras. Permet le mouvement vertical du bras principal
Servomoteur SG90 9g (180°)	Placé au niveau du « coude » pour contrôler l'extension ou la rétraction de l'avant-bras
Deux bras découpés en carton rigide	Élément structurel léger pour la réalisation des segments du bras articulé (épaule et avant-bras)
Moteur pas-à-pas 4 phases (type 28BYJ-48)	Assure la rotation du socle du bras, permettant un balayage horizontal du bras autour de la base
Visserie légère / colle renforcée	Permet l'assemblage des éléments carton/servo de manière fiable et réutilisable

TABLE 2 – Composantes physiques du bras robotisé



FIGURE 15 – servo épaule



FIGURE 16 – servo bras



FIGURE 17 – stepper

Contribution au système : Ce bras robotisé permet au robot de réaliser une tâche de manipulation simple dans un environnement structuré : atteindre un objet, le saisir, le déplacer latéralement puis le déposer avec précision dans une zone cible. Il est directement impliqué dans la logique de tri ou de transfert d'un casier source vers un casier de destination, en parfaite adéquation avec les contraintes de logistique intelligente.

4.4 Description du schéma de déplacement

Le trajet suivi par le robot est défini à l'aide d'un schéma en **forme de triangle équilatéral**, matérialisé au sol par une ligne noire épaisse. Cette ligne représente le chemin de navigation que le robot doit suivre de manière autonome, à l'aide de ses capteurs de détection de contraste.

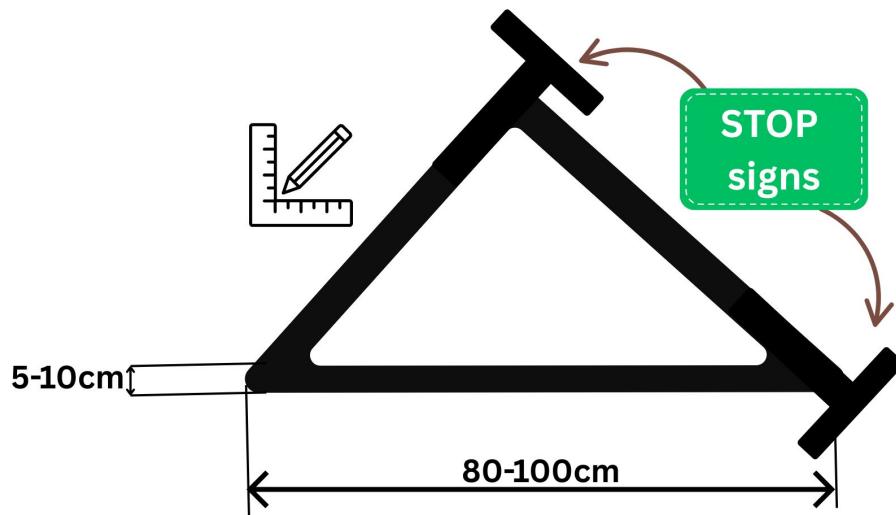


FIGURE 18 – Dimensions du schéma

Forme

Le parcours adopte une **forme triangulaire fermée**, composée de trois segments droits reliés par trois sommets, chacun jouant un rôle spécifique dans le scénario :

- **Sommet 1** : point de départ du robot (station de lancement) ;
- **Sommet 2** : zone d'arrêt devant une armoire de stockage (emplacement de dépôt ou de collecte) ;
- **Sommet 3** : point de pivot avant retour vers le point de départ.

Cette forme triangulaire permet de simuler un *cycle logistique simple et répétitif*, idéal pour tester des scénarios de collecte, de tri ou de distribution.

Dimensions

Les dimensions du triangle sont adaptées à un espace de démonstration sur table ou au sol, selon les besoins pédagogiques :

- **Longueur de chaque côté** : environ **80 cm à 100 cm** ;
- **Largeur de la ligne noire** : entre **7 cm et 10 cm**, assurant une bonne détection par les capteurs infrarouges ;
- **Rayon de rotation aux angles** : dégagement minimal de ~10 cm, suffisant pour un robot compact à base différentielle.

Support et matière

La ligne noire est appliquée sur un support rigide et lisse, garantissant une navigation fluide et une adhérence correcte des roues. Plusieurs méthodes de réalisation sont possibles :

- **Support** : panneau en MDF peint, carton plume ou plaque de PVC ;
- **Matérialisation de la ligne** : ruban adhésif vinyle noir mat, peinture noire, ou impression découpée sur vinyle autocollant ;
- **Finition du fond** : surface claire (blanche ou beige) pour assurer un contraste optimal avec la ligne noire.

Justification du choix

- La forme triangulaire offre une **structure fermée logique**, simple à suivre par un robot autonome ;
- Elle permet de tester à la fois les **virages**, les **arrêts précis**, et les **manceuvres de rotation** sans complexité excessive ;
- La largeur de la ligne garantit une détection fiable, **réduisant les risques de perte de trajectoire** ;
- L'utilisation de matériaux rigides et plans assure une **surface stable**, indispensable au bon déroulement des déplacements et à la précision du positionnement.

4.5 Scénario de déplacement et de manipulation

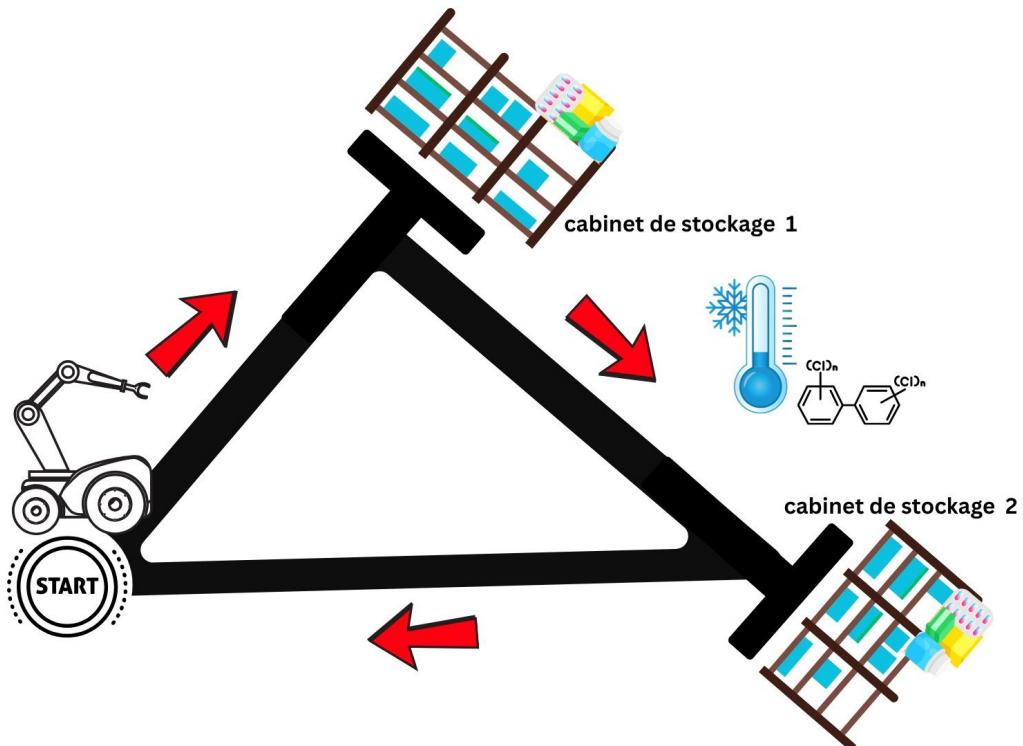


FIGURE 19 – Trajet simplifié du robot

Le robot mobile autonome évolue sur un trajet en forme de triangle équilatéral prédéfini, matérialisé au sol par une ligne noire. Le parcours débute au niveau du **point de départ**

(correspondant à l'*angle 1* du triangle). Le robot suit la ligne noire jusqu'à atteindre le second sommet (*angle 2*), où la trajectoire linéaire s'interrompt. À ce point précis, situé devant un **casier de collecte**, le robot marque un arrêt.

À l'arrêt, le bras robotisé entre en action pour réaliser l'interaction avec l'environnement : il effectue une manœuvre de dépose (ou éventuellement de prise) dans le casier, simulant une action logistique telle que le dépôt d'un contenant médical.

Une fois cette opération terminée, le robot redémarre et poursuit son chemin en effectuant une **rotation contrôlée**, à droite ou à gauche selon l'algorithme de navigation implémenté. Il poursuit ainsi la trajectoire triangulaire jusqu'à atteindre le sommet suivant, où il répète la même séquence : arrêt, interaction avec le casier, rotation et reprise du parcours.

Ce cycle se poursuit de manière autonome jusqu'à ce que le robot boucle le triangle et **revienne à son point de départ**, complétant ainsi un cycle complet de collecte et de dépôt. Cette logique simple mais structurée permet d'illustrer un scénario de distribution ou de tri automatisé dans un environnement logistique ou pharmaceutique.

Manipulation du bras et mécanisme

Lorsque le robot atteint la fin d'un segment de trajet, il rencontre une situation de **sorte de ligne**, détectée par l'ensemble des deux capteurs infrarouges. Cette condition correspond à un arrêt net du robot, les deux capteurs ne percevant plus la ligne noire simultanément. Cette position d'arrêt est volontairement alignée avec un **casier de stockage**, représenté par une matrice fixe de type 3×3 comportant neuf emplacements, numérotés de 1 à 9.

Une fois cette situation atteinte, le robot déclenche la séquence de manipulation via son bras articulé. Le mécanisme repose sur deux articulations principales :

- **L'articulation de l'épaule** : permet le mouvement vertical du bras ;
- **L'articulation du bras (avant-bras)** : contrôle l'extension horizontale vers l'avant.

En fonction du **numéro de case** sélectionné (allant de 1 à 9), une correspondance interne détermine les *angles précis de rotation* à appliquer à chacun des deux servos. Le bras adopte alors une **position spécifique et calibrée** lui permettant d'atteindre physiquement la case visée dans la matrice.

Exemple : mouvement vers la case 4

Le déplacement vers la case numéro 4 illustre une séquence de mouvement combinée entre la base rotative (gérée par un moteur pas-à-pas) et le bras articulé contrôlé par deux servomoteurs.

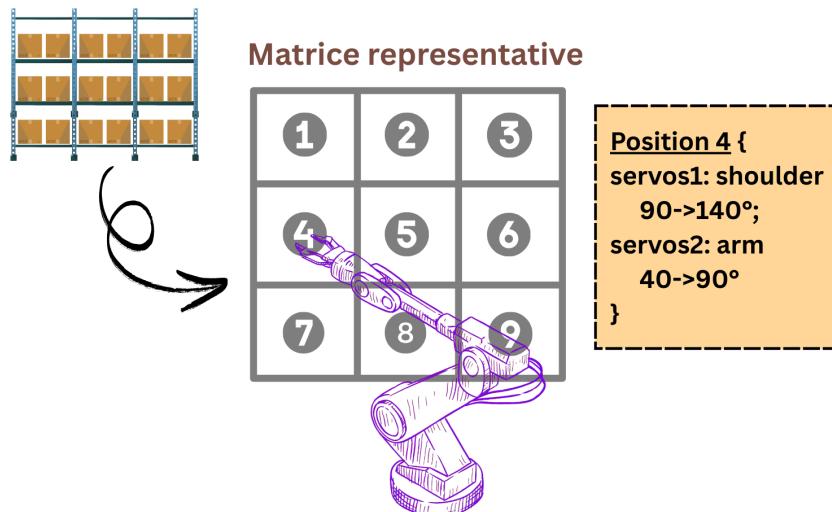


FIGURE 20 – Séquence de mouvement du bras pour atteindre la case 4

- **Orientation horizontale** : le moteur pas-à-pas effectue une rotation vers la gauche équivalente à un tour complet (-360° ou $-stepsPerRevolution$), permettant d’aligner la base avec la colonne de gauche (cases 1, 4, 7).
- **Positionnement vertical** : la fonction `touch5()` est appelée pour déployer le bras vers la ligne centrale (case 5), simulant un accès en hauteur moyen.
- **Retour à l’état initial** : une rotation inverse est exécutée pour repositionner le bras vers sa position d’origine, prêt pour un nouveau mouvement.

Après avoir touché (ou simulé l’action de dépôt dans) la case ciblée, le bras effectue une **retraction automatique** pour revenir à sa position initiale d’attente, prêt à redémarrer le cycle de navigation vers le prochain point.

Cette séquence de manipulation assure une interaction coordonnée, précise et répétable avec un meuble de rangement structuré, en accord avec les scénarios de tri ou de stockage intelligent.

5 Affichage de l’état via un afficheur 7 segments

5.1 Intérêt de l’afficheur 7 segments

L’afficheur 7 segments est intégré au système dans le but d’assurer une **visualisation claire et immédiate** de l’état du robot, en particulier lors de la sélection aléatoire d’une case cible dans la matrice 3×3 . Lorsqu’un arrêt est détecté (perte de ligne), un nombre aléatoire entre 1 et 9 est généré pour désigner la case à atteindre. Ce nombre est affiché en temps réel sur le 7 segments, permettant ainsi une interprétation facile du comportement du robot durant les phases de test, de démonstration ou de diagnostic.

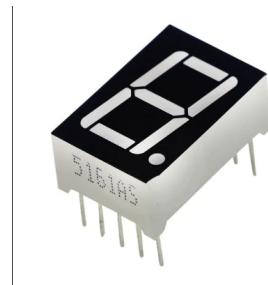


FIGURE 21 – Affichage du numéro de case cible sur le 7 segments

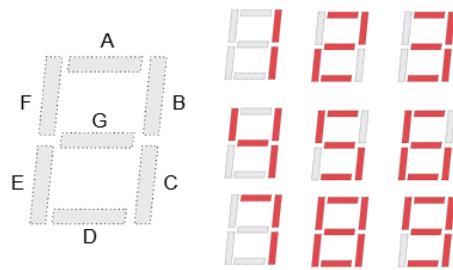


FIGURE 22 – Câblage de l'afficheur 7 segments avec registre à décalage

L'utilisation de cet affichage visuel apporte une réelle plus-value pédagogique et fonctionnelle, car elle permet de suivre les **décisions internes** du programme sans connexion externe (ni ordinateur, ni écran série).

5.2 Problème rencontré : manque de broches disponibles

La carte Arduino Uno dispose d'un nombre limité de broches numériques utilisables. Dans notre cas, de nombreuses broches sont déjà allouées à d'autres composants : deux capteurs IR, deux servomoteurs, un moteur pas-à-pas et des LED indicatrices. Il est donc devenu difficile de connecter les 7 (ou 8) broches nécessaires au contrôle direct d'un afficheur 7 segments standard.

5.3 Solution adoptée : utilisation d'un registre à décalage

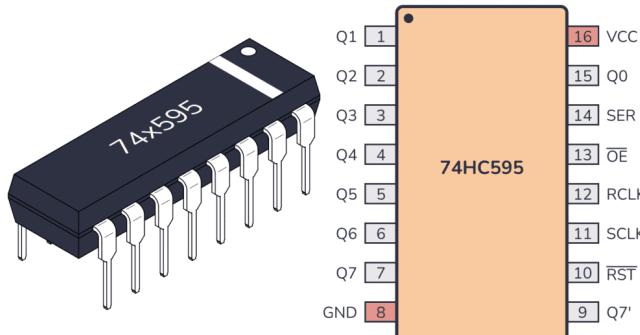


FIGURE 23 – Shift register 74hc595

Pour surmonter cette contrainte matérielle, un **registre à décalage** (type **74HC595**) a été intégré au circuit. Ce composant permet de **piloter jusqu'à 8 sorties numériques** à partir de seulement **3 broches de contrôle** sur l'Arduino (Data, Clock et Latch).

Grâce à cette technique, l'afficheur 7 segments peut être contrôlé efficacement sans saturer les broches restantes, et le câblage reste simple et modulaire. Le registre reçoit un octet représentant le motif binaire correspondant au chiffre à afficher, puis met à jour les segments via un processus de décalage série (bit-shifting).

Cette solution optimise l'utilisation des ressources disponibles sur la carte et permet d'ajouter une interface de communication visuelle sans compromettre les autres fonctionnalités du robot.

6 Algorithme embarqué sur Arduino Uno

6.1 Présentation de la carte Arduino Uno



FIGURE 24 – Carte Arduino Uno utilisée pour le contrôle embarqué

L'**Arduino Uno** est une carte à microcontrôleur basée sur l'ATmega328P. Elle permet de programmer et de contrôler des systèmes embarqués en temps réel. Dans notre projet, elle est utilisée comme *unité centrale de traitement*, orchestrant à la fois la navigation du robot, la détection d'environnement, et l'action des moteurs (servo et pas-à-pas).

La programmation s'effectue via l'environnement Arduino IDE, en utilisant le langage C++ simplifié fourni par les bibliothèques intégrées.

6.2 Explication des fonctions définies

6.2.1 Boucle principale

Listing 1 – Logique de suivi optimisée

```

1 void loop() {
2     // Lecture des capteurs
3     int left = digitalRead(trackingsensor_left);
4     int right = digitalRead(trackingsensor_right);
5
6     // Cas 1: Ligne centr e
7     if (left == HIGH && right == HIGH) {
8         moveForward();

```

```

9     lastSeenTime = millis(); // R initialisation du timeout
10    }
11    // Cas 2: Déviation gauche
12    else if (left == LOW && right == HIGH) {
13        turnRight(); // Correction progressive
14        lastSeenTime = millis();
15    }
16    // Cas 3: Déviation droite
17    else if (left == HIGH && right == LOW) {
18        turnLeft(); // Correction progressive
19        lastSeenTime = millis();
20    }
21    // Cas 4: Ligne perdue
22    else {
23        if (millis() - lastSeenTime > timeout) {
24            stopMotors();
25            arrange(); // Déclenchement des actions
26        }
27    }
28    delay(50); // Anti-rebond
29}

```

- **Stratégie PID simplifiée** : Correction proportionnelle par différence de vitesse
- **Gestion du timeout** : 300ms sans détection déclenche l'arrêt
- **Anti-rebond** : Delay de 50ms pour stabiliser la lecture

6.2.2 Fonction touch5()

Fonction touch5() : exemple de mouvement vertical

La fonction touch5() positionne le bras robotisé pour atteindre la case centrale (5) de la matrice 3×3 . Elle règle directement les deux servomoteurs à des angles précis afin d'amener la pince du bras sur la cible, puis elle revient à la position d'attente.

Listing 2 – Mouvement vers la case 5

```

1 void touch5(){
2     // Phase d'extension (mouvement simultané paule / coude)
3     for (int pos = 40; pos <= 90; pos++) {
4         servo1.write(pos+50); // paule : 90          140      (+50)
5         servo2.write(pos);   // Coude: 40           90
6         delay(35); // Vitesse moyenne
7     }
8
9     // Phase de repli de l'paule
10    for (int pos = 150; pos >= 90; pos--) {
11        servo1.write(pos); // paule revient      90
12        delay(15); // Vitesse rapide
13    }
14
15    // Phase de repli du coude
16    for (int pos = 90; pos >= 40; pos--) {

```

```

17   servo2.write(pos);      // Coude revient      40
18   delay(15);
19 }
20 delay(1000); // Pause pour stabilisation
21 }
```

Cette fonction implémente un mouvement en 3 temps : 1. **Extension coordonnée** : Les deux servos avancent simultanément pour atteindre la position avant 2. **Retour séquentiel** : L'épaule revient d'abord, puis le coude 3. **Vitesses différenciées** : Mouvement avant lent (35ms), retour rapide (15ms)

6.3 Stratégie de réutilisation : simplification du code

Pour éviter la répétition de neuf fonctions différentes (`touch1()`, `touch2()`, ..., `touch9()`), seules les fonctions `touch2()`, `touch5()` et `touch8()` ont été programmées pour gérer les trois lignes verticales principales de la matrice.

Les autres positions sont obtenues par ajout de commandes de rotation du moteur pas-à-pas (base rotative), avant ou après l'appel d'une des fonctions de base :

- `touch1()` = rotation gauche + `touch2()`
- `touch3()` = rotation droite + `touch2()`
- `touch4()` = rotation gauche + `touch5()`
- `touch6()` = rotation droite + `touch5()`
- `touch7()` = rotation gauche + `touch8()`
- `touch9()` = rotation droite + `touch8()`

6.4 Fonctions de Roulement

Listing 3 – Contrôle moteur différentiel

```

1 // Vitesses prédéfinies
2 const int motorSpeed = 150;          // 60% puissance
3 const int correctionSpeed = 200; // 80% puissance
4
5 void moveForward() {
6   analogWrite(motorLeft_A, motorSpeed); // PWM gauche
7   analogWrite(motorRight_A, motorSpeed); // PWM droit
8   // Broches B    0 pour sens avant
9 }
10
11 void turnRight() {
12   analogWrite(motorLeft_A, correctionSpeed);
13   analogWrite(motorRight_B, correctionSpeed/2); // Rotation douce
14   // Compensation asymétrique
15 }
```

Paramètre	Effet
motorSpeed = 150	Équilibre vitesse/précision
correctionSpeed/2	Adoucit la rotation droite
asymétrie LEFT/RIGHT	Compense le désalignement mécanique

Optimisations :

- Minimisation des oscillations (delay 15ms)
- Angles limites prédéfinis (40° - 150°)
- Séquence garantissant le retour à la position initiale

6.5 Suivi de ligne et détection d'arrêt

Le robot utilise deux capteurs infrarouges (IR) pour suivre une ligne noire. L'algorithme compare l'état des capteurs :

- Gauche = noir, Droite = blanc \rightarrow virage léger à gauche
- Gauche = blanc, Droite = noir \rightarrow virage léger à droite
- Les deux = noir \rightarrow avancer droit
- Les deux = blanc \rightarrow fin de trajectoire \rightarrow déclenchement de la séquence de dépôt

6.6 Enchaînement général du programme

L'algorithme embarqué implémente un cycle autonome complet de navigation et de manipulation. Le robot commence par suivre une ligne noire à l'aide de ses deux capteurs infrarouges. Tant qu'au moins l'un des capteurs détecte la ligne, le robot ajuste sa direction pour rester aligné : en avançant droit, ou en effectuant de légers virages à gauche ou à droite selon l'état des capteurs. Lorsqu'aucun des capteurs ne détecte plus la ligne (c'est-à-dire que les deux capteurs perçoivent un fond clair), le robot interprète cette situation comme une **fin de segment** et s'arrête devant une armoire de stockage.

À ce moment-là, le programme génère un nombre aléatoire entre 1 et 9, correspondant à une case de la matrice 3×3 simulant les emplacements de dépôt. Ce nombre est simultanément affiché sur un *afficheur 7 segments* pour visualisation. Le robot appelle alors la fonction **arrange(pos)**, qui déclenche la séquence de mouvements nécessaires pour orienter le bras et atteindre la case cible, en combinant les mouvements de rotation de la base et les articulations verticales du bras. Une fois la tâche exécutée (prise ou dépôt), le bras revient à sa position initiale, et le robot effectue une rotation (droite ou gauche) pour reprendre son circuit triangulaire vers le prochain point de collecte ou de dépôt.

7 Simulation MATLAB : Déplacement et Animation du Bras

Dans le but de simuler le comportement du robot autonome et valider la logique du cycle de collecte, nous avons développé un script en MATLAB utilisant le contrôleur

`controllerPurePursuit`, couplé à un modèle de bras articulé. Le robot évolue sur un circuit triangulaire et s'arrête automatiquement aux sommets pour effectuer une action via le bras.

OneDrive ▶ Bureau ▶ github_repo ▶

) Editor - C:\Users\dodi\OneDrive\Bureau\github_repo\test2.m

1 **% Define Vehicle**
2 R = 0.1;
3 L = 0.5;
4 dd = DifferentialDrive(R, L);
5
6 **% Simulation parameters**
7 sampleTime = 0.1;
8 tVec = 0:sampleTime:50;
9 initPose = [0; 0; 0];
10 pose = zeros(3, numel(tVec));
11 pose(:,1) = initPose;
12
13 **% Waypoints : triangle fermé**
14 waypoints = [
15 0, 0;
16 4, 0;
17 2, 3.5;
18 0, 0;

Command Window

```
>> startMobileRoboticsSimulationToolbox
>> test2
● Arrêt au sommet B (4, 0)
● Arrêt au sommet C (2, 3.5)
✓ Retour au point de départ (0,0). Fin de simulation.
```

FIGURE 25 – Simulation MATLAB du robot autonome avec bras

7.1 Objectifs de la simulation

- Suivre une trajectoire fermée : triangle ($A \rightarrow B \rightarrow C \rightarrow A$)
 - Représenter la position du robot en temps réel
 - Arrêt automatique à chaque sommet
 - Animation visuelle du bras (prise/dépôt)

7.2 Trajectoire et Contrôle

Nous avons défini les points de passage suivants :

$$\text{Waypoints} = \begin{bmatrix} 0 & 0 \\ 4 & 0 \\ 2 & 3.5 \\ 0 & 0 \end{bmatrix}$$

Le contrôleur de poursuite pure guide le robot à chaque instant vers un point situé à une distance d'anticipation (Lookahead).

7.3 Modélisation du bras articulé

Le bras robotique est modélisé comme un système plan à deux degrés de liberté :

- Segment 1 : l'épaule (longueur $L_1 = 0.5 m$)
- Segment 2 : l'avant-bras (longueur $L_2 = 0.4 m$)

L'angle d'orientation du robot est noté θ , celui du bras secondaire est animé dynamiquement par une fonction sinusoïdale lors de l'arrêt :

$$\phi(t) = \phi_0 + A \cdot \sin(\omega t)$$

Les positions du coude (joint) et de la main (pince) sont calculées comme suit :

$$\text{Joint} = \text{base} + L_1 \cdot \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}, \quad \text{Main} = \text{Joint} + L_2 \cdot \begin{bmatrix} \cos(\theta + \phi(t)) \\ \sin(\theta + \phi(t)) \end{bmatrix}$$

7.4 Résultats visuels de la simulation

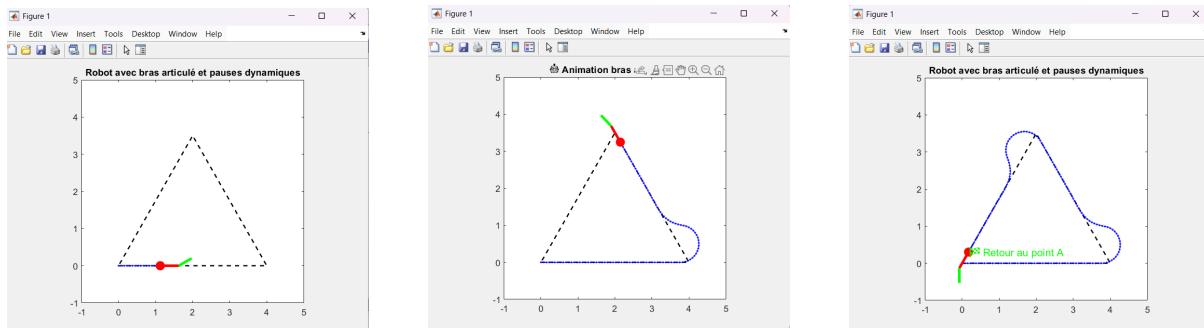


Figure : Phases de simulation du robot avec bras (suivi, animation bras B, manipulation C)

7.5 Extrait de code représentatif

```

1 % Détection de point B
2 if ~isPausedAtB && norm(currentPos - [4; 0]) < 0.3
3     for k = 1:20
4         dynamic_angle = deg2rad(30 * sin(2 * pi * 0.25 * k *
5             sampleTime));
6         joint = pos + L1*[cos(theta); sin(theta)];
7         hand = joint + L2*[cos(theta + dynamic_angle); sin(theta +
8             dynamic_angle)];
9         % Affichage du bras
10    end
11 end

```

Cette animation reproduit fidèlement le comportement de prise et de dépôt d'un bras mécanique lors du passage aux sommets.

Observation sur les déviations de trajectoire

Il est important de noter que les légères déviations observées dans les courbes ne sont pas des erreurs de navigation, mais bien une illustration du comportement correcteur du robot.

En effet, à chaque fois que le robot s'écarte légèrement de la ligne noire (en raison de la dynamique ou du virage), le contrôleur ajuste sa trajectoire pour le ramener progressivement vers le chemin défini. Ces micro-déviations traduisent donc la réactivité et l'efficacité du système de correction intégré dans la simulation.

7.6 Conclusion

Pourquoi la simulation MATLAB est-elle cruciale ?

Avant toute implémentation physique, la simulation MATLAB permet de :

- **Vérifier la cohérence du modèle** : Le comportement du robot peut être testé dans un environnement maîtrisé, sans risque matériel.
- **Valider les lois de commande** : Le contrôleur PurePursuit et la logique d'arrêt/animation du bras sont analysés en temps réel.
- **Optimiser les trajectoires et les paramètres** : Longueurs des segments du bras, amplitude de mouvement, seuils d'arrêt... tout peut être ajusté rapidement.
- **Préparer l'intégration physique** : Les comportements simulés servent de base pour programmer les microcontrôleurs (Arduino) avec une logique validée.

La simulation constitue donc une étape indispensable pour éviter les erreurs coûteuses lors de la phase de prototypage.

La simulation MATLAB valide le cycle complet de déplacement, arrêt, prise, dépôt, et retour. Elle démontre la synchronisation entre la navigation du robot et l'animation de son bras articulé. Cette modélisation est utile à la fois pour le test des lois de commande et la visualisation du comportement global du système.

8 Matériel Utilisé

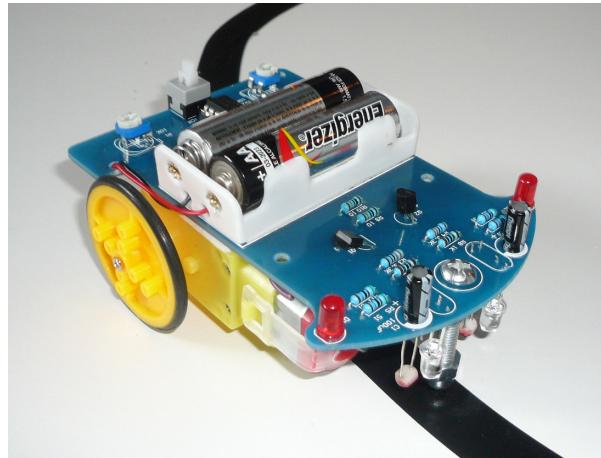
Composant	Utilisation
Arduino Uno	Unité de traitement et de commande
2 capteurs infrarouges	Suivi de ligne
2 servomoteurs	Manipulation d'objet pour prise et dépôt
1 moteur pas-à-pas	Positionnement angulaire ou translation précise
Châssis robot 2 roues	Support et locomotion du robot
3 LED	Indicateurs d'état (prise, transport, dépôt)
Matrice de casiers (Mur 1 et Mur 2)	Collecte et dépôt des éléments

TABLE 3 – Liste des composants et leur utilisation dans le robot

9 Livrables

- Cahier des charges

- Schéma électrique
- Programme Arduino commenté
- Simulation sous Simulink (optionnelle)
- Démonstration fonctionnelle du robot



Semaine	Tâches
S1	Choix du matériel, définition du parcours et Assemblage du châssis et des moteurs
S2	Programmation du suivi de ligne et Intégration des servos et logique de collecte
S3	Intégration des servos et logique de collecte et Tests finaux et ajustements
S4	Rapport final et démonstration

TABLE 4 – Planning prévisionnel du projet

10 Valeur Ajoutée par Rapport au Cours

Ce projet applique concrètement plusieurs notions abordées dans le cours de robotique intelligente pour l'Industrie 4.0 :

- **Robot mobile autonome** : Notre système est un exemple de *Wheeled Mobile Robot (WMR)* doté d'un déplacement autonome, illustrant les principes de locomotion et de navigation vus en cours.
- **Capteurs et actionneurs intelligents** : L'usage de capteurs infrarouges (IR) pour la détection de ligne et de servomoteurs pour la manipulation d'objets met en œuvre l'architecture robotique : *Perception → Décision → Action*.
- **Automatisation et contrôle** : Le projet exploite une boucle de contrôle simple sur Arduino, basée sur des conditions détectées en temps réel, incarnant les principes d'**automation intelligente**.

- **Manipulation et pick-and-place :** L'action de saisir un objet et le déposer s'inscrit dans le modèle de *Pick and Place*, typique des bras industriels, et adapté ici à un robot mobile.
- **Cycle intelligent et reprogrammable :** Le robot peut être reprogrammé pour modifier son trajet ou la logique de dépôt, en lien direct avec la définition d'un **robot intelligent reprogrammable** selon le cours.
- **Connexion avec l'Industrie 4.0 :** Ce projet représente une miniaturisation des systèmes logistiques automatisés utilisés dans les usines connectées et entrepôts intelligents modernes.

11 Perspectives d'Évolution

- Intégration de modules Bluetooth/WiFi pour pilotage à distance
- Ajout de capteurs de distance pour l'évitement d'obstacles
- Extension vers plusieurs casiers avec algorithme de tri intelligent

12 Conclusion

Ce projet nous a donné l'opportunité de créer un système robotique autonome, conçu de A à Z, pour répondre à un défi concret d'automatisation dans un environnement exigeant. Grâce à la conception modulaire du robot, nous avons intégré des capteurs pour la navigation, des actionneurs pour la manipulation, et un microcontrôleur pour coordonner les tâches, tout en appliquant les principes fondamentaux que nous avons appris durant notre formation.

Chaque décision technique, que ce soit le choix d'un châssis différentiel, les algorithmes de suivi de ligne, la manipulation à deux degrés de liberté, ou encore la simulation MATLAB, a été guidée par une compréhension approfondie des architectures robotiques, des modèles cinématiques, et des systèmes embarqués. Ce processus a vraiment renforcé notre capacité à transformer des concepts théoriques en solutions concrètes et fonctionnelles.

En reproduisant un scénario logistique réaliste, ce projet met en lumière les enjeux essentiels des systèmes robotiques modernes : autonomie, précision, adaptabilité, et sécurité. Il s'inscrit parfaitement dans l'approche d'apprentissage axée sur la robotique intelligente et son rôle de plus en plus important dans les environnements industriels complexes.

13 Bibliographie

Robots in Pharmaceutical Cold Chain Logistics