INSEA

Royaume du Maroc

Ministère de l'Education Nationale,
du Préscolaire et des Sports

TRAVAUX D'INITIATIVE
PERSONNELLE ENCADRÉS
T.I.P.E. 2026

LIBRIASSIST

Robot organisateur des livres

Réalisé par :
ZINOUN MOHAMED AMINE

Encadré par:
PR A.R. JADID
PR M. ESSADKI

PLAN

- **Introduction**
- **Problématique**
- **Cahier des charges**
- **Partie 1:** Etude de Robot
- **Partie 2:** Assurer une navigation optimal
- **Partie 3:** Réaliser un prototype
- **Conclusion**

LibriAsssit : quelle valeur ajoutée ?

Robot Librarian - Aurora | Senserbot

problématique

Afin de répondre au besoin de remettre les livres à leurs places dans les rangées correspondantes. Comment peut-on concevoir et contrôler LIBRIASSIST dans ses phases de fonctionnement (réception des livres, déplacement vers la rangée, détection d'emplacement approprié, et dépôt d'un livre) ?

Diagramme de cas d'utilisation :

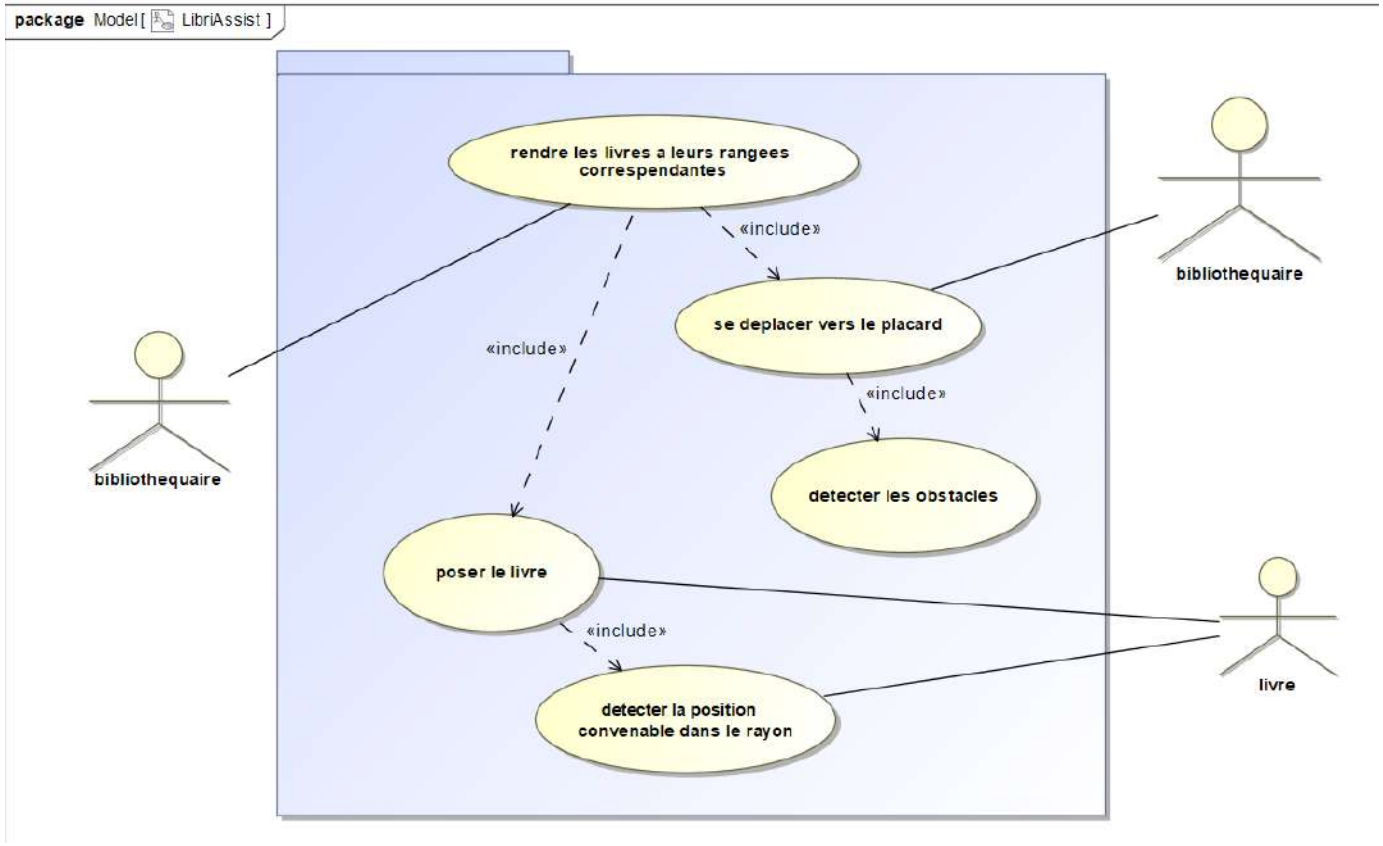


Diagramme de définition des blocs

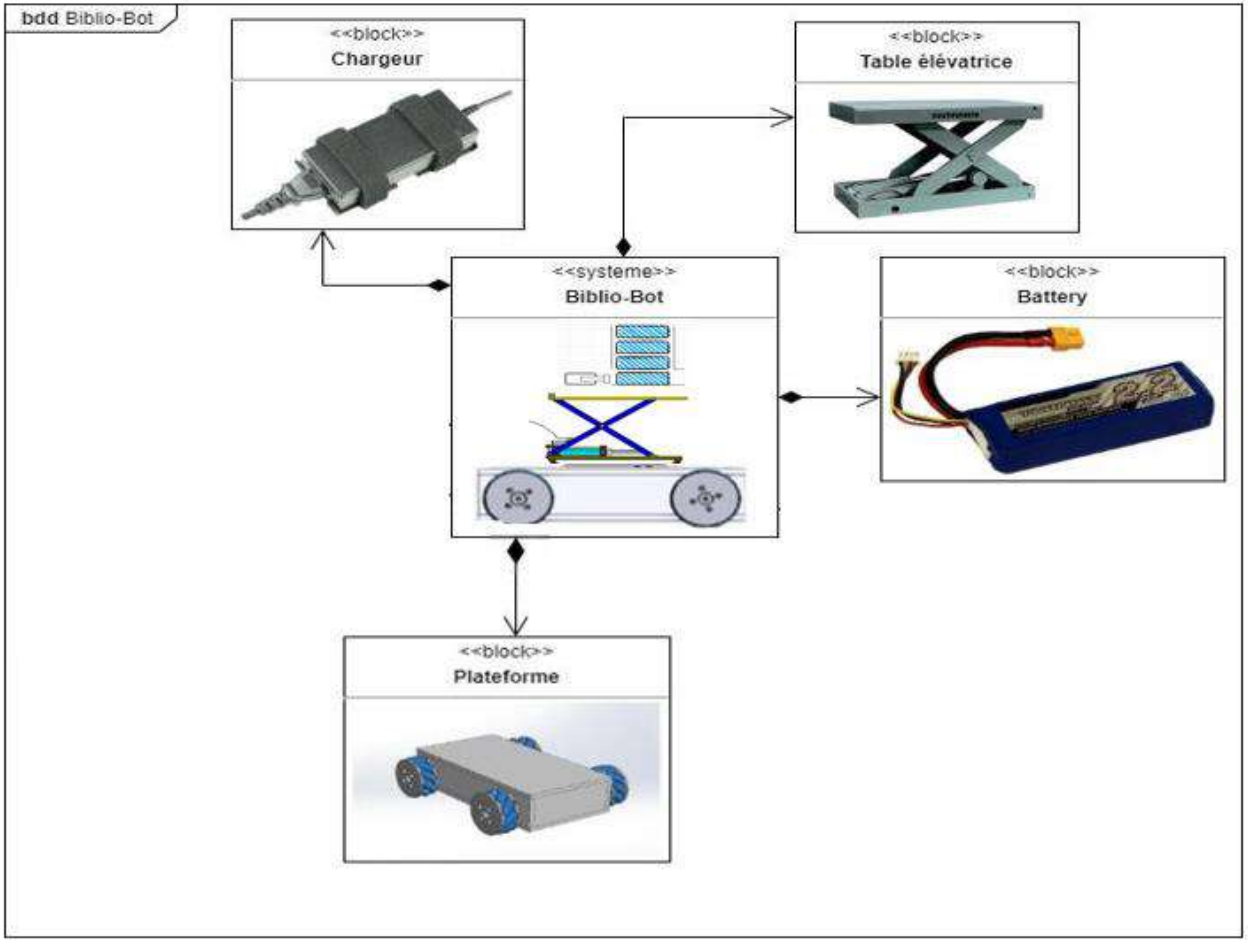


Diagramme de définition des blocs

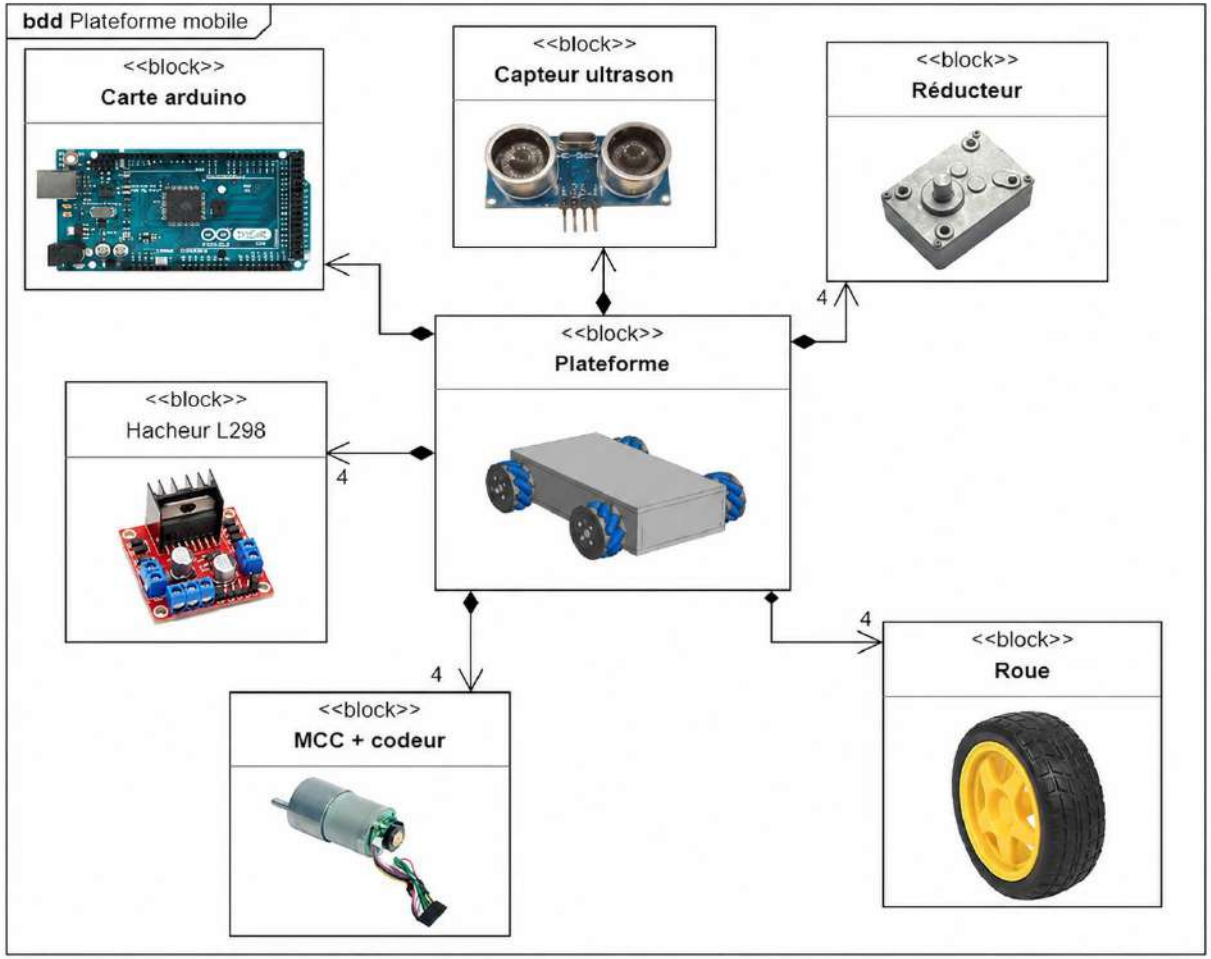


Diagramme de définition des blocs

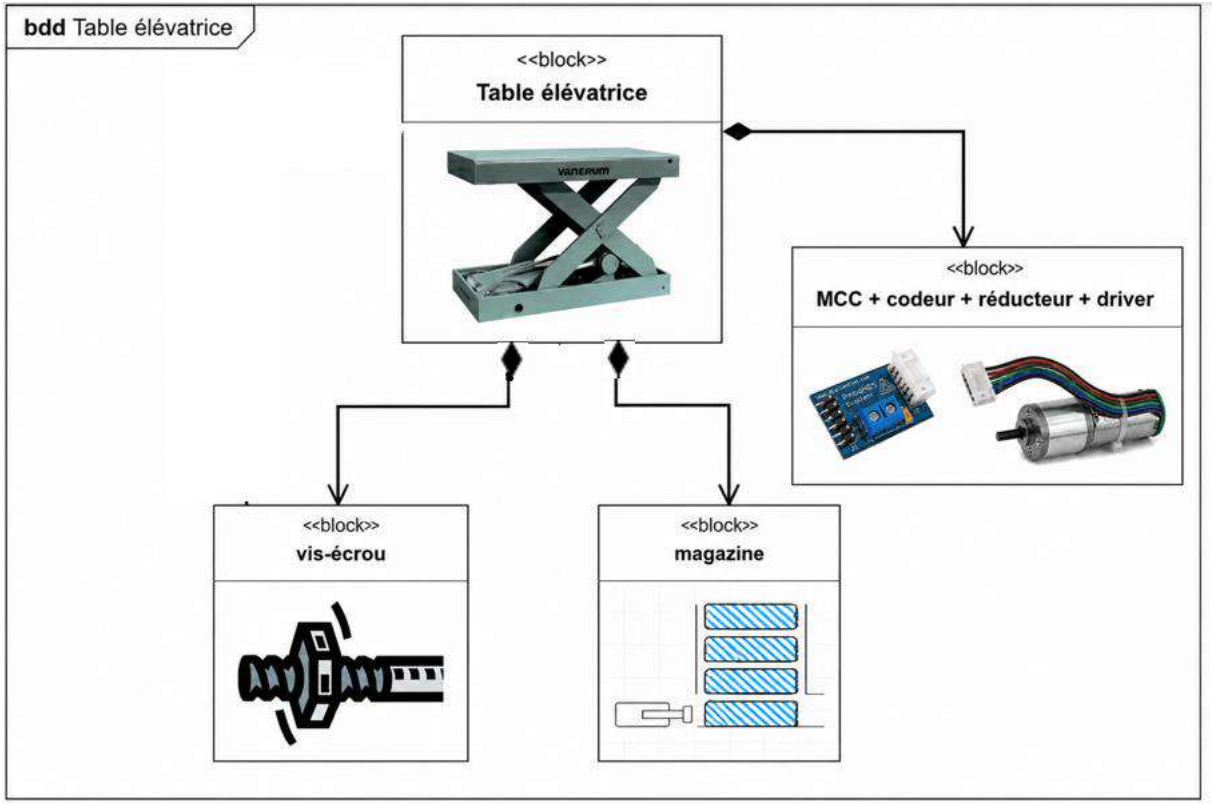


Diagramme des exigences

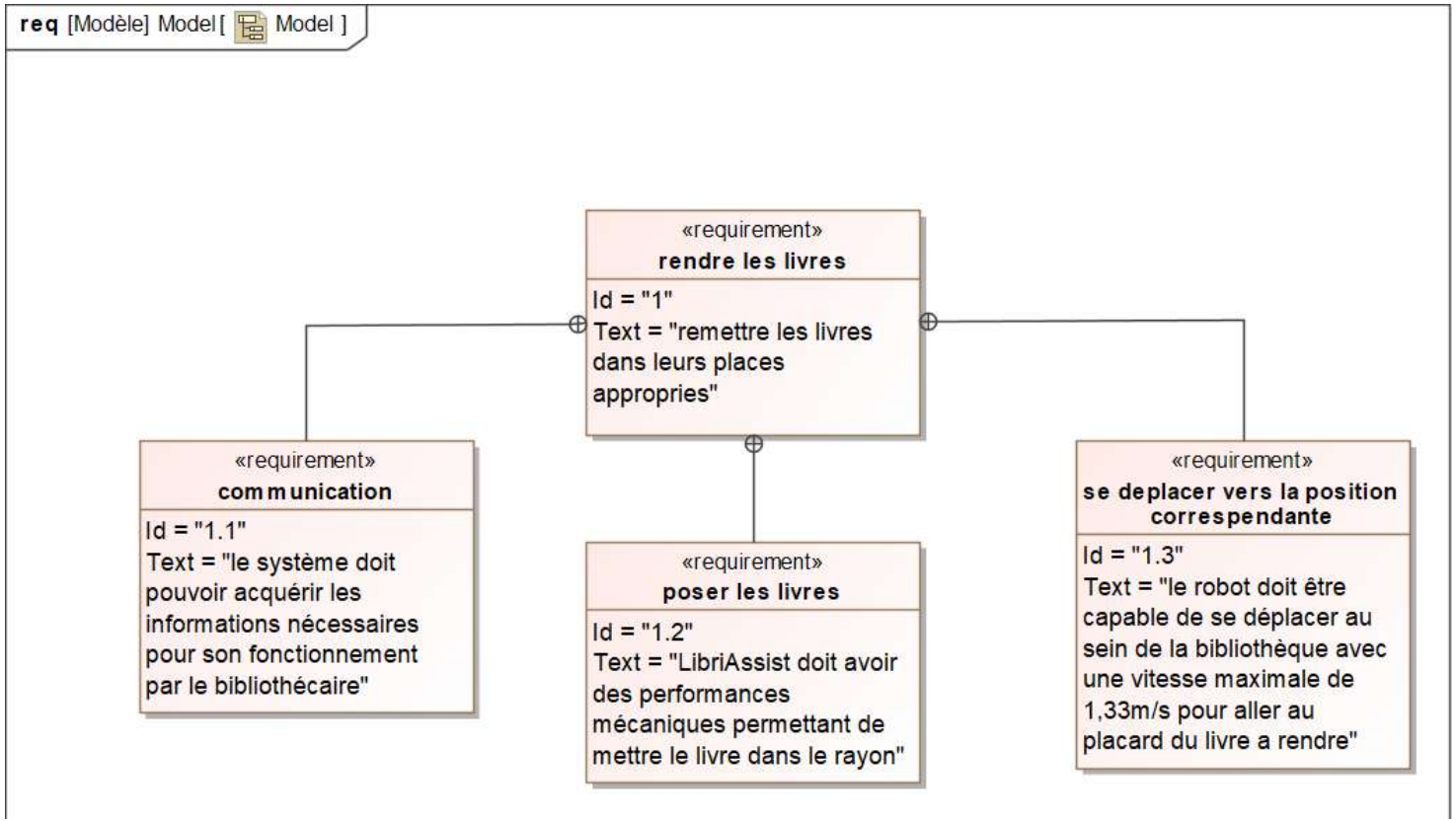


Diagramme des exigences

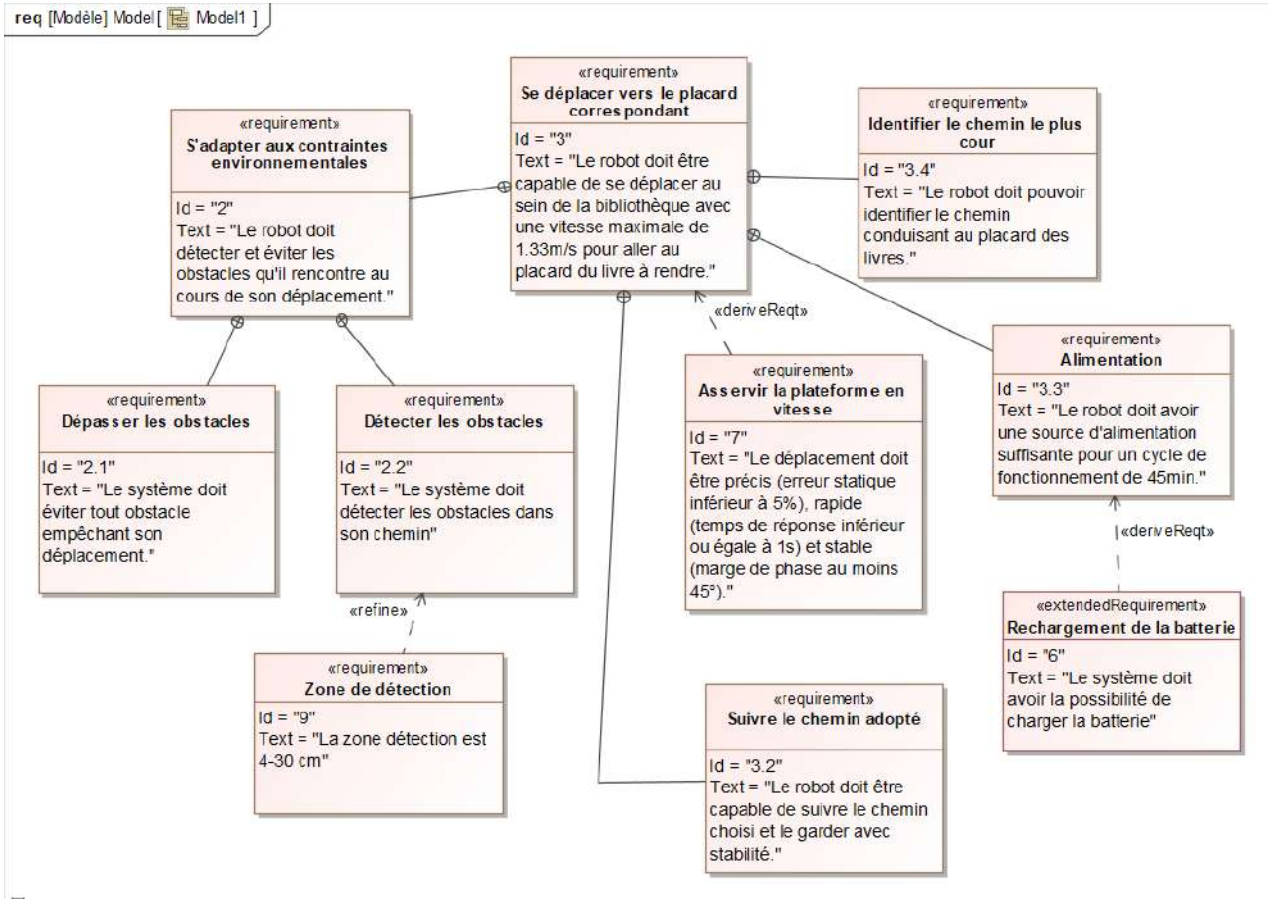
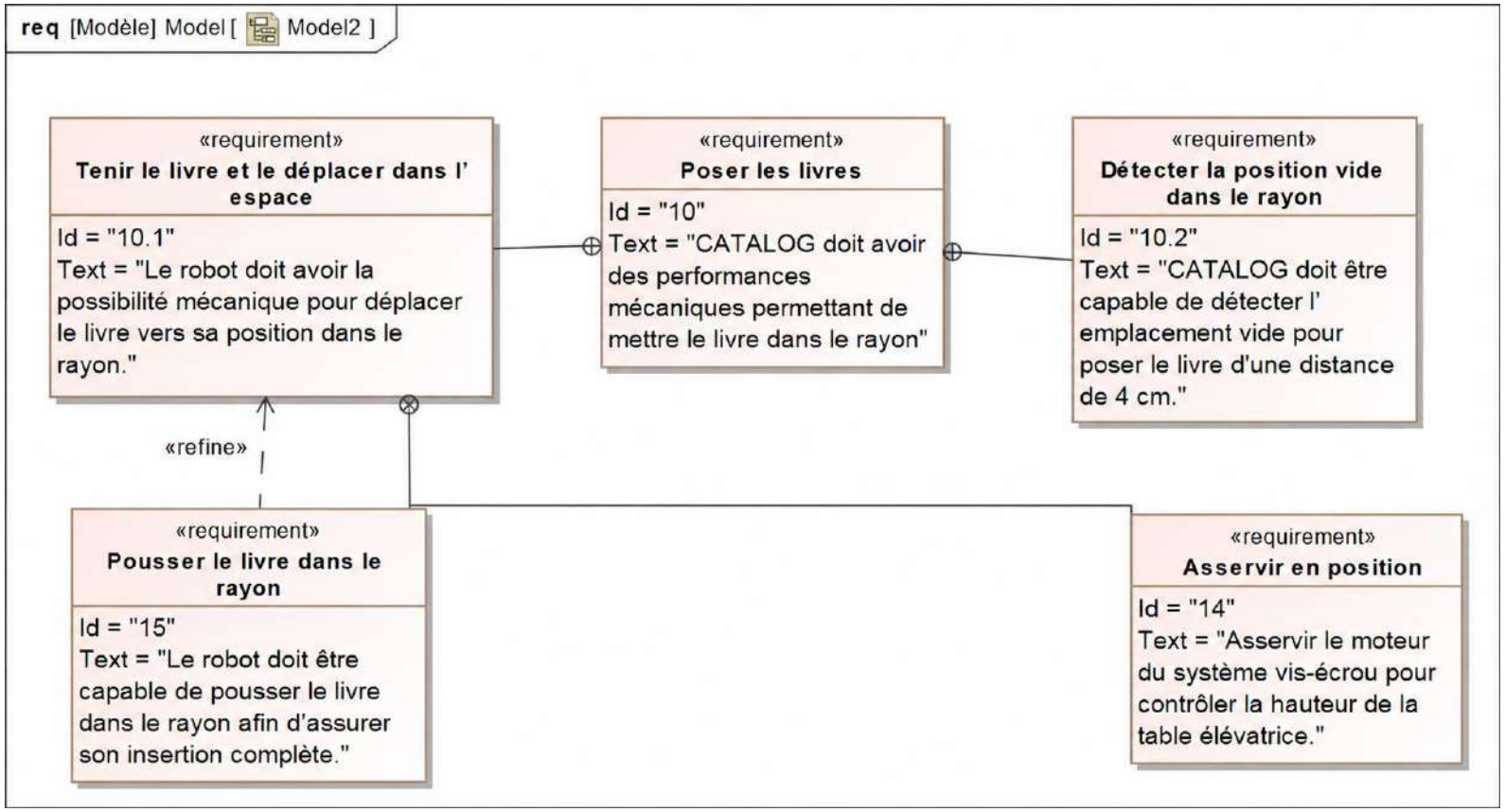
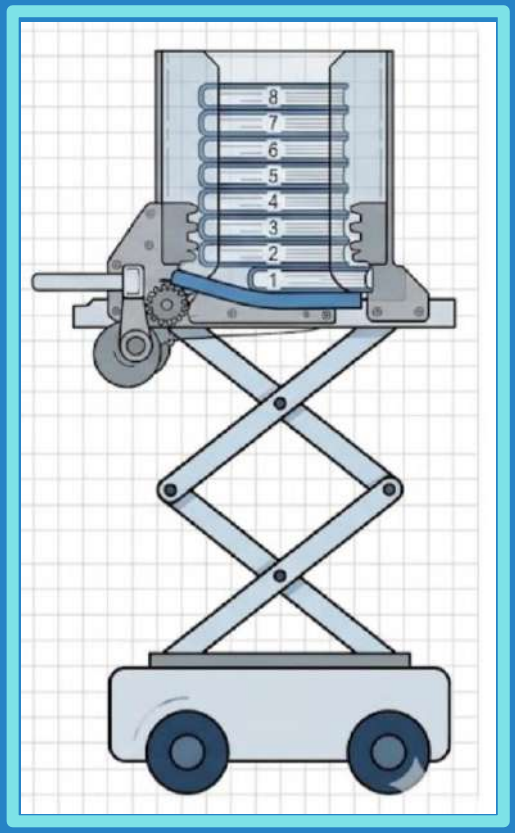


Diagramme des exigences



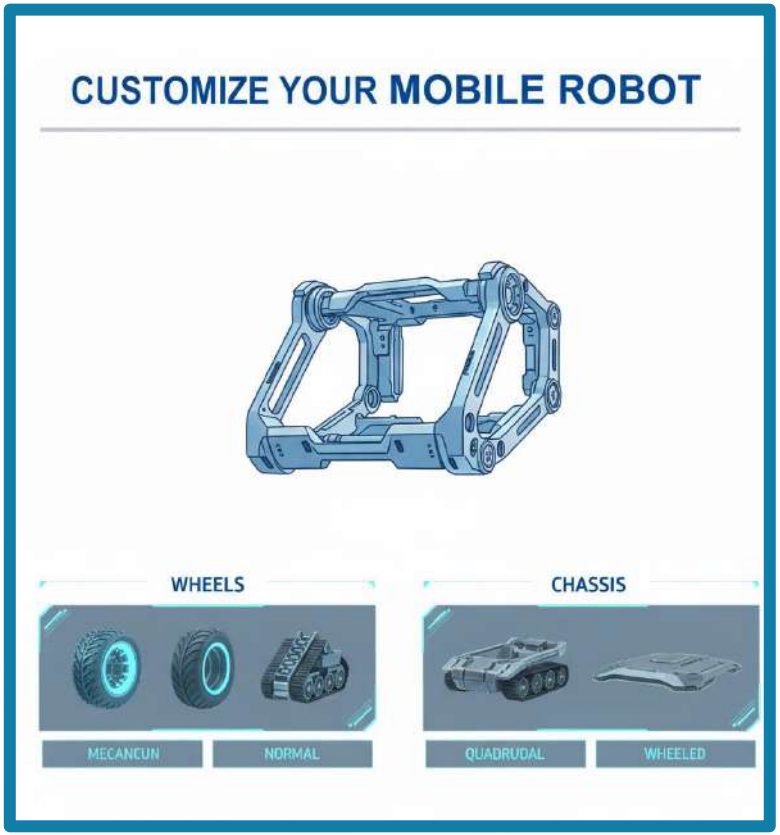
PARTIE 1

ETUDE DU ROBOT



Objectif 1

Choix du robot



I. Justifier le choix du type de robot :

Analyse des besoins de déplacement :

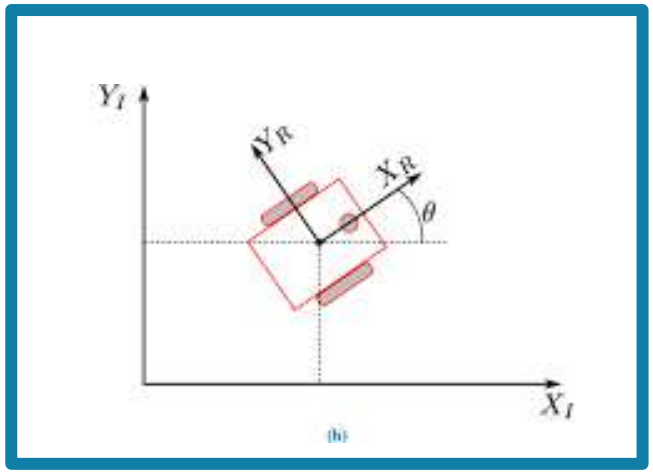
L'environnement d'une bibliothèque est très structuré, avec des couloirs étroits, des trajectoires majoritairement rectilignes, des virages serrés et des arrêts précis devant les étagères. Les déplacements latéraux sont peu nécessaires, tandis que la fiabilité de la localisation est essentielle.

I. Justifier le choix du type de robot :

Comparaison du mouvement des architectures possibles :

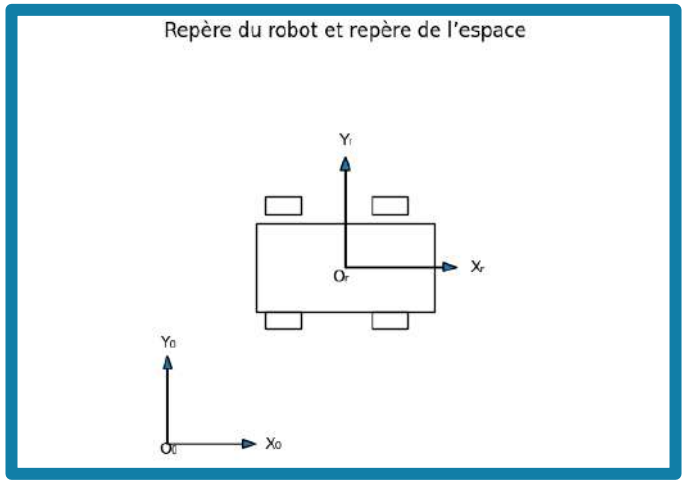
ROBOT DIFFERENTIEL

- Avance / recule (**translation x**)
- Tourne sur place (**rotation θ**)
- Le changement de direction se fait par **rotation**



ROBOT OMNIDIRECTIONNEL

- se déplace **en avant / arrière (x)**
- se déplace **latéralement (y)**
- garder la **même orientation** pendant le déplacement



I. Justifier le choix du type de robot :

critères déterminantes :

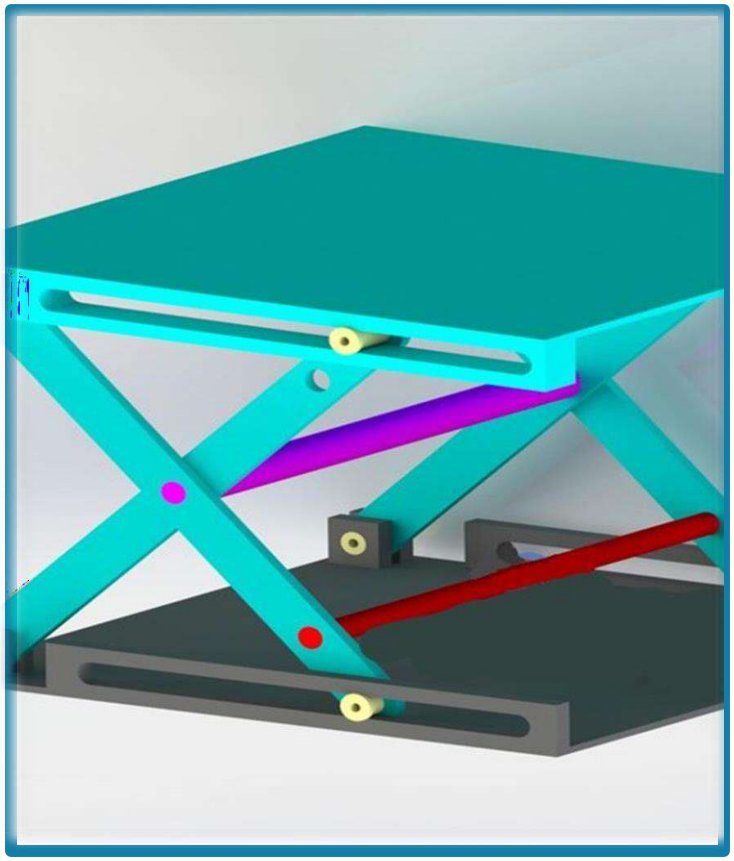
Critère	Robot différentiel	Robot omnidirectionnel
Adaptation aux couloirs étroits	Très adapté (déplacements rectilignes + rotation sur place)	Adapté mais capacité latérale inutile dans ce contexte
Complexité mécanique	Simple (2 roues motrices + roue libre)	Complexe (roues omni/mecanum, plus de moteurs)
Commande et modélisation	Modèle cinématique simple	Modélisation plus complexe
Odométrie / Localisation	Plus fiable et facile à calculer	Plus complexe à estimer
Précision d'arrêt devant étagère	Très bonne	Bonne mais sans avantage décisif
Pertinence pour Biblio-Bot	Solution optimale	Solution surdimensionnée

Type de robot choisi :

→ **ROBOT DIFFERENTIEL**

Objectif 2

- Motorisation et asservissement de position de la table élévatrice



I. choix du moteur :

Fermeture géométrique :

D'après le théorème de Pythagore appliqué au triangle rectangle :

$$L^2 = Z^2 + (a+\lambda)^2$$

$$Z = \sqrt{L^2 - a^2 - 2a\lambda - \lambda^2}$$

$$Z = \sqrt{L^2 - (a+\lambda)^2}$$

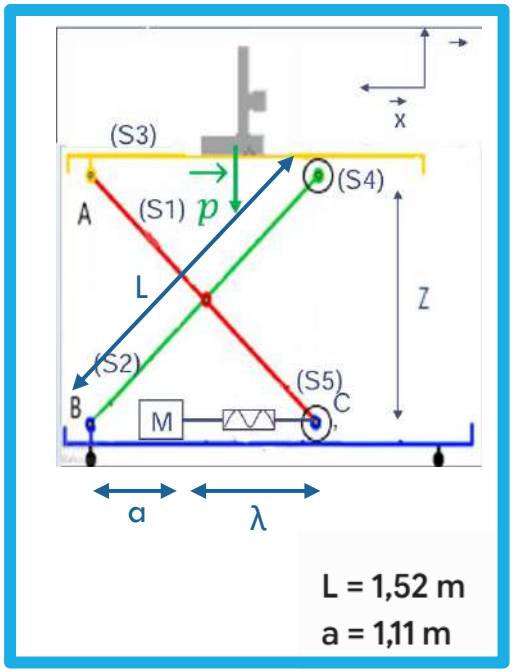
On néglige λ^2 devant la constante $K = \sqrt{L^2 - a^2}$:

$$Z = \sqrt{L^2 - a^2} \cdot \sqrt{1 - \frac{2a\lambda}{L^2 - a^2}}$$

On effectue le développement limité suivant $\sqrt{1-x} \approx 1 - \frac{x}{2}$

$$Z \approx \sqrt{L^2 - a^2} \cdot \left(1 - \frac{1}{2} \cdot \frac{2a\lambda}{L^2 - a^2}\right)$$

$$Z \approx \sqrt{L^2 - a^2} - \frac{a}{\sqrt{L^2 - a^2}} \cdot \lambda$$



$$Z = -1{,}07\lambda + 1{,}04$$

I. choix du moteur :

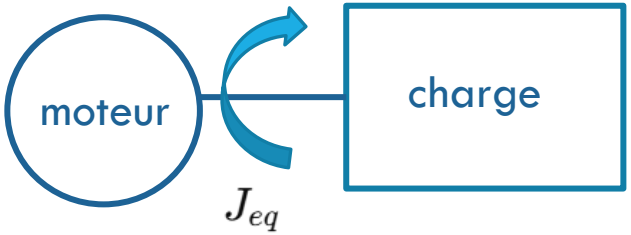
L'effort développé sur l'axe moteur ;

z(t) : déplacement vertical de la charge

$$z(t) = K_t \cdot \frac{p}{2\pi} \cdot \theta_m(t) \quad \dot{z} = K_t \cdot \frac{p}{2\pi} \cdot \Omega_m \quad \ddot{z} = K_t \cdot \frac{p}{2\pi} \cdot \dot{\Omega}_m$$

$$E_c^{\text{charge}} = \frac{1}{2} m \left(K_t \cdot \frac{p}{2\pi} \cdot \Omega_m \right)^2 + E_c^{\text{moteur}} = \frac{1}{2} J_m \Omega_m^2$$

$$E_c = \frac{1}{2} \left[J_m + m \left(K_t \cdot \frac{p}{2\pi} \right)^2 \right] \Omega_m^2$$



Puissance moteur

$$P_m = C_m \cdot \Omega_m$$

Puissance du poids

$$P_p = -mg \cdot K_t \cdot \frac{p}{2\pi} \cdot \Omega_m$$

Théorème de l'Énergie Cinétique (TEC)

$$\frac{d}{dt}\left(\frac{1}{2} J_{eq} \Omega_m^2\right) = C_m \Omega_m - mgK_t \frac{p}{2\pi} \Omega_m$$

$$C_m = \left[J_m + mK_t^2 \left(\frac{p}{2\pi} \right)^2 \right] \dot{\Omega}_m + mgK_t \frac{p}{2\pi}$$

$$J_{eq} = J_m + m \cdot K_t^2 \cdot \left(\frac{p}{2\pi} \right)^2$$

Application numérique:

$$Cm = 51{,}3\ mN.m$$

$$m = 10{,}25\ \text{kg}$$
$$g = 9{,}81\ \text{m/s}^2$$
$$p = 3 \times 10^{-3}\ \text{m}$$
$$K_t = 1{,}07$$
$$\dot{\Omega}_{max} = 1000\ \text{rad/s}^2$$

I. choix du moteur :

MOTEUR CHOISI

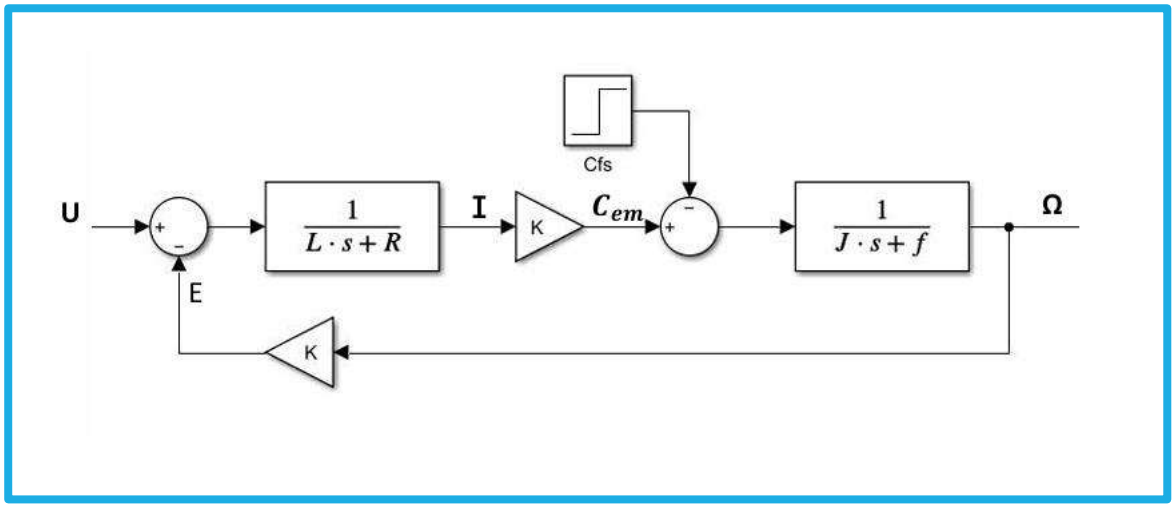
le moteur Polulo#4822

Rated Voltage	Motor Type	Stall Current	No-Load Current	Gear Ratio	No-Load Speed (RPM)	Extrapolated Stall Torque (kg · cm)	Extrapolated Stall Torque (oz · in)	Max Power (W)	Without Encoder	With Encoder
6 V	Low-Power (LP)	2.0 A	100 mA w/o encoder	1:1 (no gearbox)	6200	0.15	2.1	2.1	–	item #4820
				4.4:1	1300	0.63	8.7	2.1	item #1581	item #4821
				9.7:1	630	1.3	18	1.9	item #1582	item #4822
				20.4:1	290	2.5	35	1.9	item #1583	item #4823
				34:1	180	3.9	54	1.7	item #1584	item #4824
				47:1	130	4.8	67	1.5	item #1585	item #4825
			120 mA with encoder	75:1	80	7.5	100	1.5	item #1586	item #4826
				99:1	61	9.1	130	1.4	item #1587	item #4827
				172:1	35	14	190	1.2	item #1588	item #4828
				227:1	27	17	240	1.1	item #1589	item #4829
				378:1	16	25	350	–	item #1590	item #4830
				499:1	12	31	430	–	item #1591	item #4831

$$C_m = 1.3\ Kg.cm = 0.127\ Nm$$
$$n_{red} = 630\ tr.min^{-1}$$

II. Modèle du moteur :

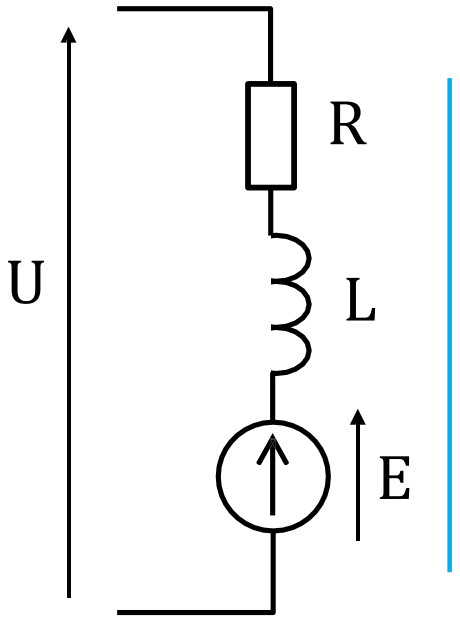
Modèle de connaissance du moteur à courant continu:



II. Modèle du moteur :

Détermination de la résistance interne:

U (V)	1.2	1.9	3
I (mA)	223	360	525

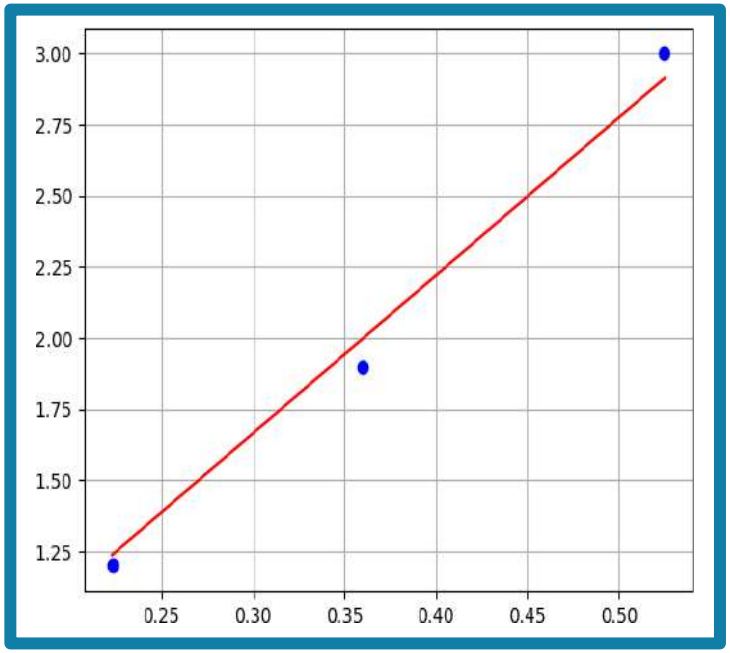


Rotor bloqué :

$\Omega = 0 \text{ rad.s}^{-1} \Rightarrow E = 0$ V

Régime permanent :

$$L.\frac{dI}{dt} = 0 \text{ V}$$



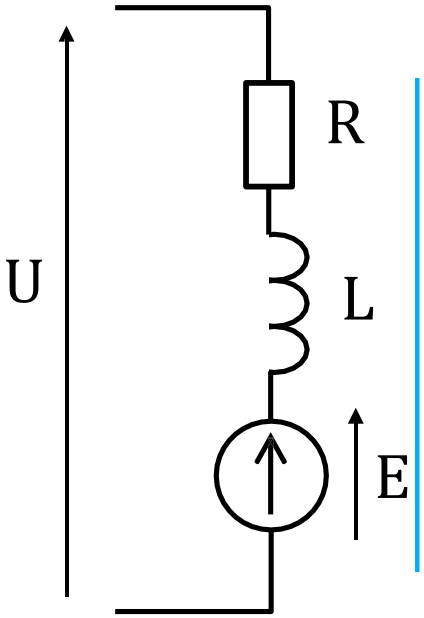
Ainsi : $U = R.I$

$$R = 5.55 \pm 0.61\ \Omega$$

II. Modèle du moteur :

Détermination de l'inductance interne:

U (V)	1.2	1.9	3
I (mA)	223	360	525

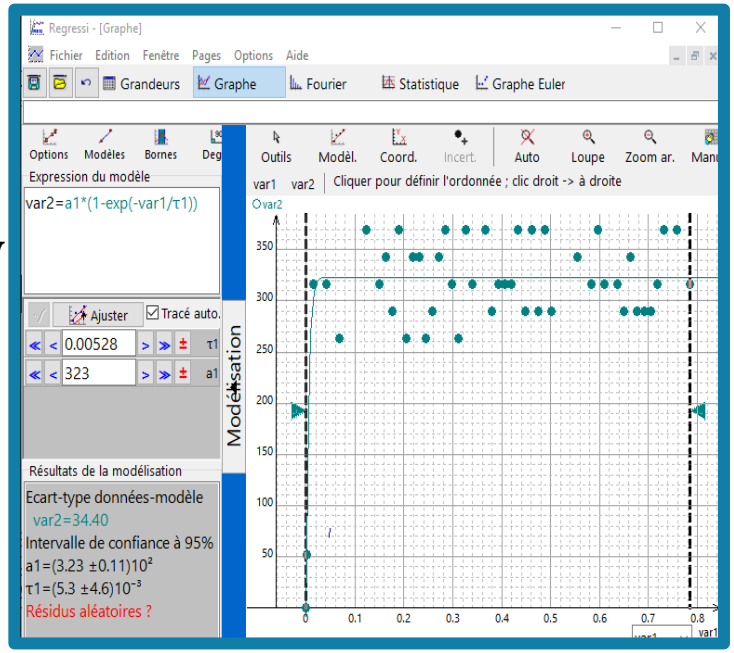


Rotor bloqué :

$\Omega = 0 \text{ rad.s}^{-1} \Rightarrow E = 0 \text{ V}$

Régime transitoire :

$$U = R.I + L.\frac{dI}{dt}$$



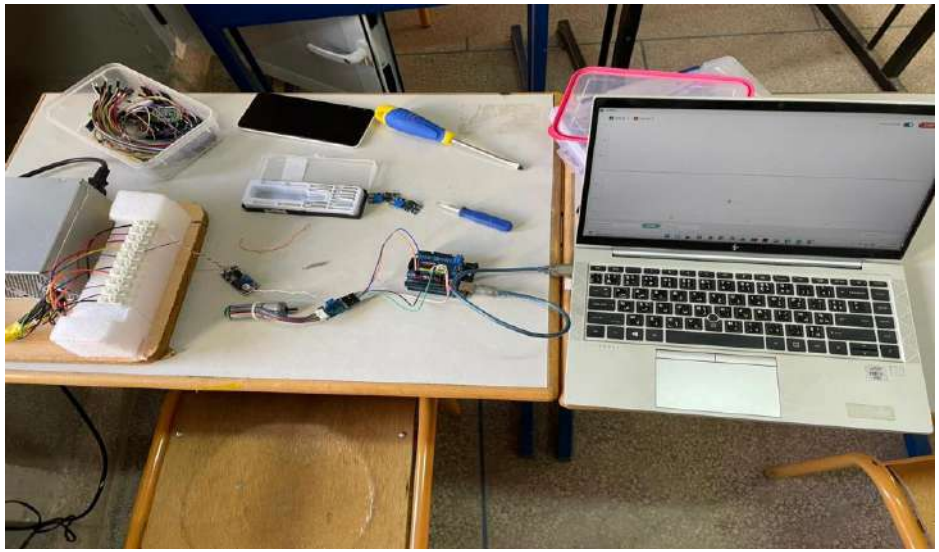
$$\tau = \frac{L}{R} = 5.28\text{ms}$$

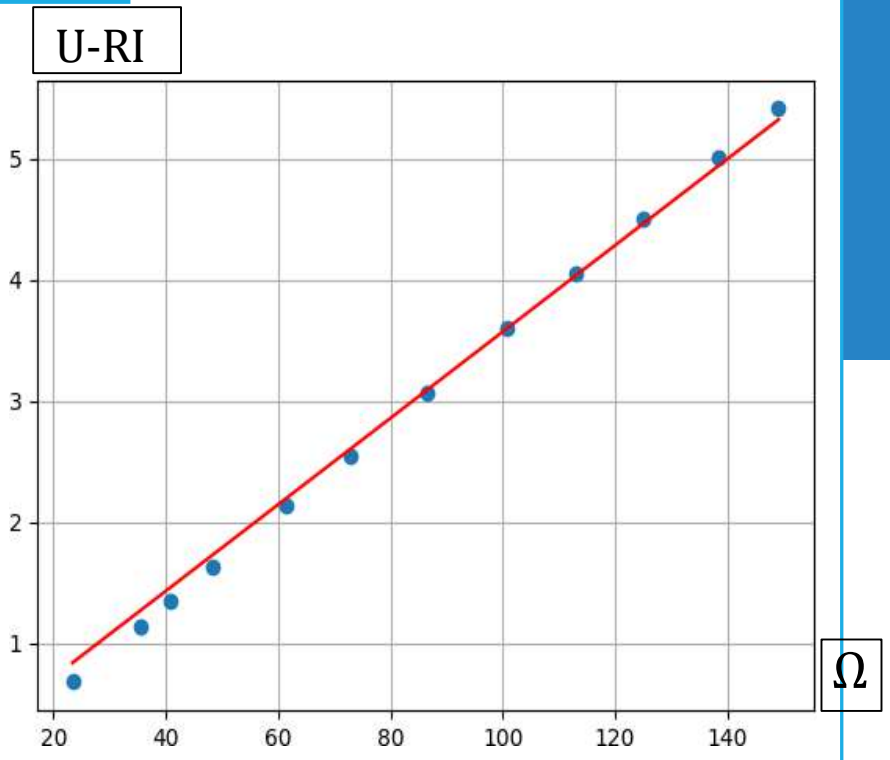
Ainsi : $U = R.I$

$$L = 29 \text{ mH}$$

II. Modèle du moteur :

Détermination du coefficient de la f.é.m. :





Régime permanent :

$$U = R.I + K.\Omega$$

Ainsi : $(U - R.I) = K.\Omega$

$K = 0.0376 \pm 0.0015\ Vs$

II. Modèle du moteur :

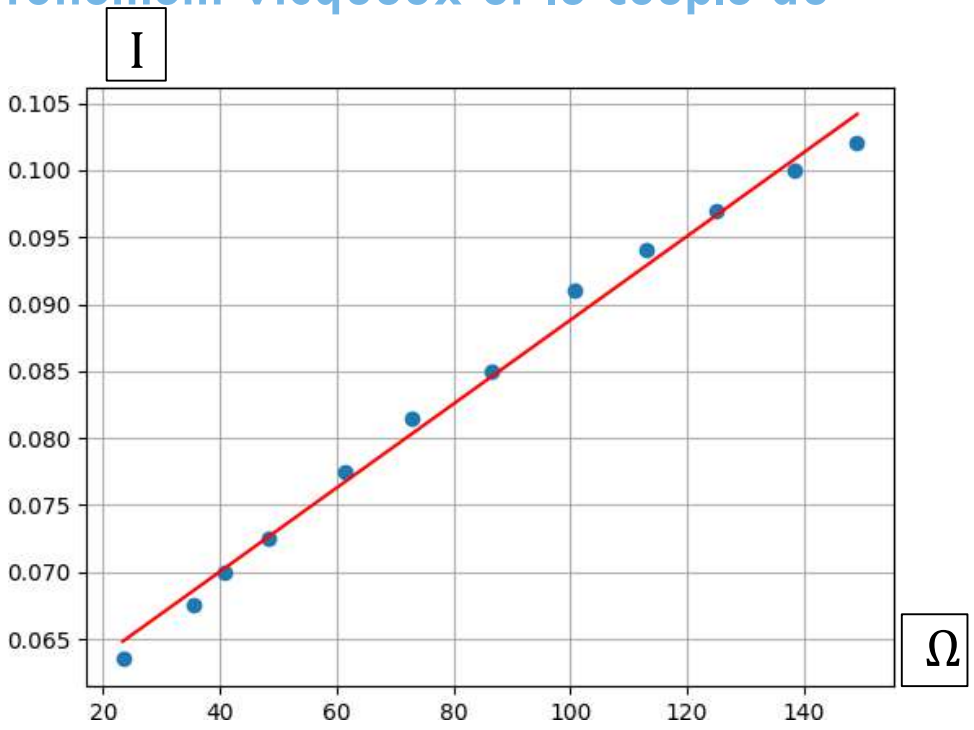
Détermination du coefficient du frottement visqueux et le couple du frottement sec:

Théorème du moment dynamique :

$$J\frac{d\Omega}{dt} = C_{em} - C_{fs} - f.\Omega$$

Régime permanent :

$$C_{em} - C_{fs} - f.\Omega = 0$$
$$\Rightarrow KI = f.\Omega + C_{fs}$$



Ainsi : $$I = \frac{f}{K}.\Omega + \frac{C_{Fs}}{K}$$

$$f = (1.174 \pm 0.052)10^{-5}\ \text{Nms}$$

$$C_{fs} = (1.906 \pm 0.037)10^{-3}\ \text{Nm}$$

II. Modèle du moteur :

Détermination du moment d'inertie (moteur+réducteur+codeur):

Essai à vide et à induit ouvert au ralentissement

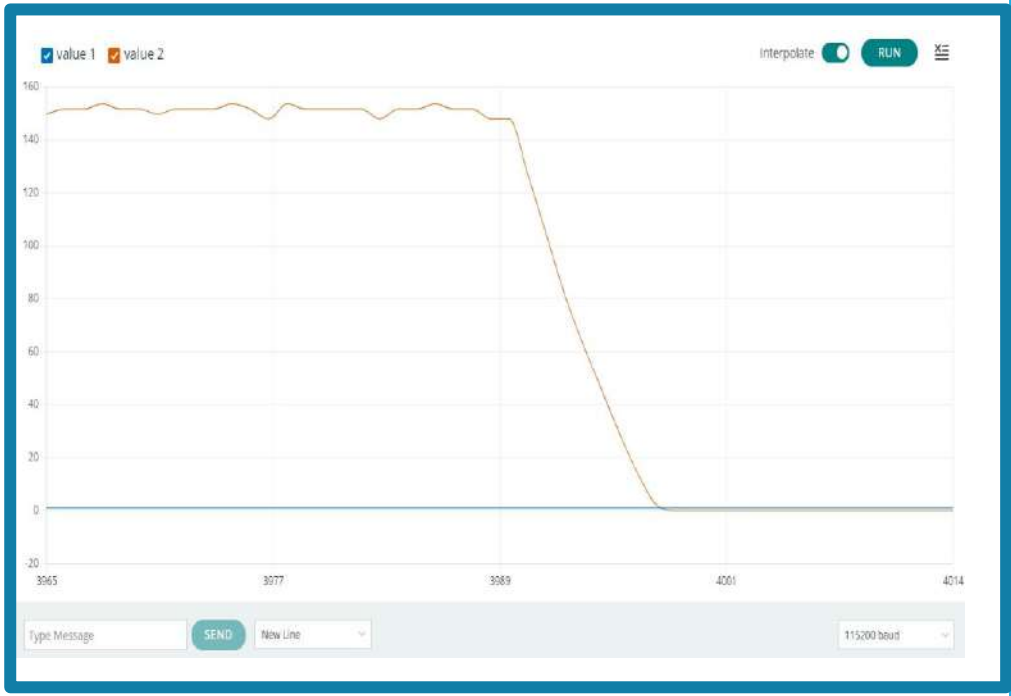
Induit ouvert:

$$I = 0 \quad \Rightarrow \quad C\_em = KI = 0$$

Théorème du moment dynamique :

$$J.\frac{d\Omega}{dt} = -C_{fs} - f.\Omega$$

$$\Omega(t) = \left(\Omega_0 + \frac{C_{Fs}}{f}\right).e^{-\frac{F}{J}t} - \frac{C_{Fs}}{f}$$



Ainsi : $$J = \frac{f.\Delta t}{\ln\left(\frac{C_{Fs}+F.\Omega_0}{C_{Fs}}\right)}$$

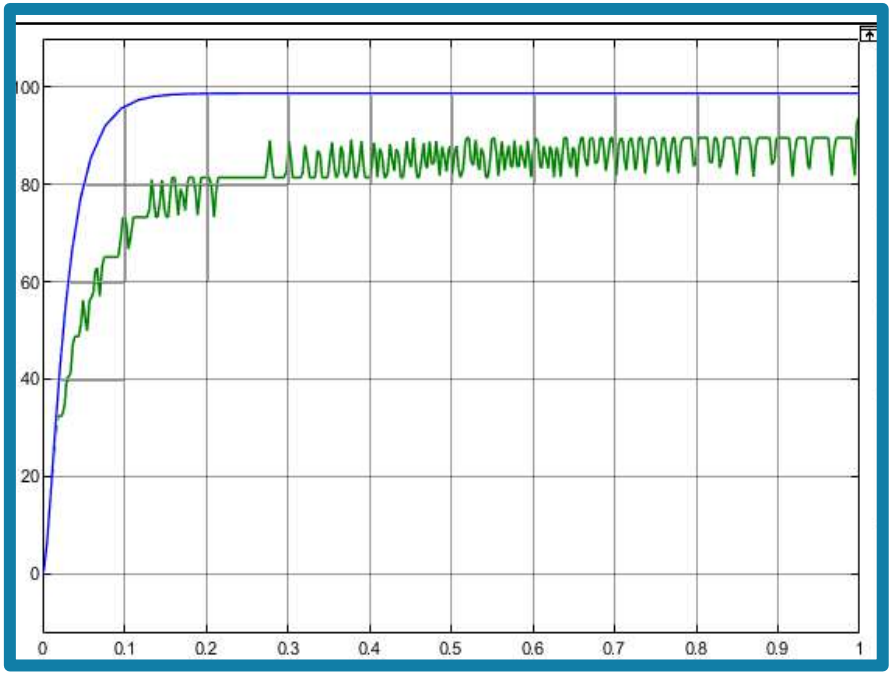
$$\Delta t = 0.35\ s\ \&\ \Omega_0 = 150\ rads^{-1}$$

$$J = 8.13 \times 10^{-6}\ Kgm^2$$

II. Modèle du moteur :

bilan:

R	$5.55 \pm 0.61\ \Omega$
L	29 mH
K	0.0376 ± 0.0015 Vs
f	$(1.174 \pm 0.052)10^{-5}$ Nms
c_{fs}	$(1.906 \pm 0.037)10^{-3}$ Nm
J	8.13×10^{-6} Kgm^2



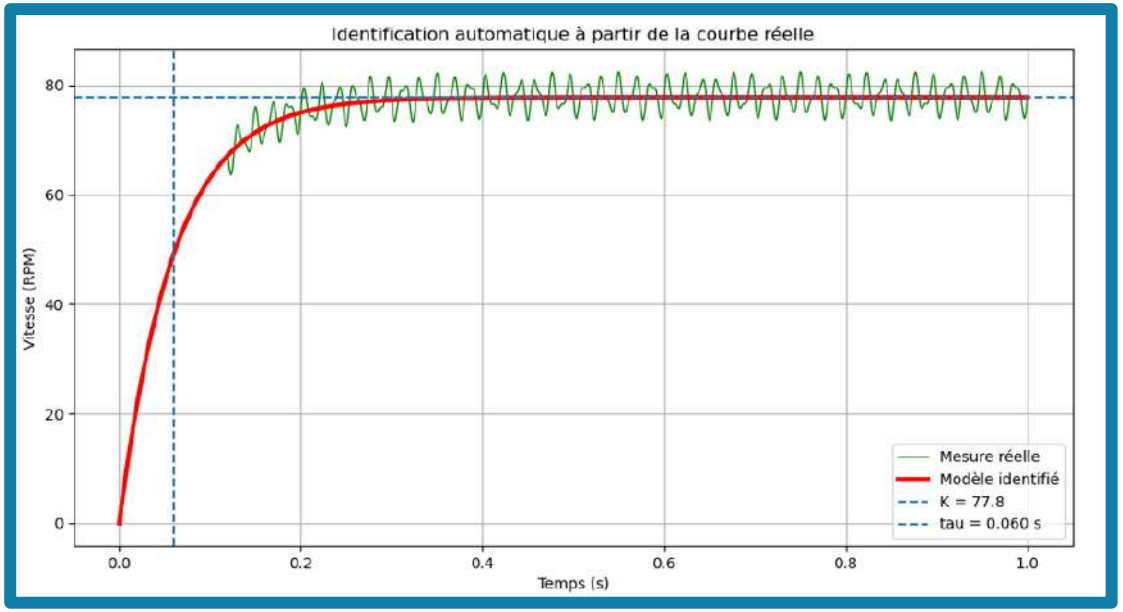
MODELE NON VALIDE

II. Modèle du moteur :

bilan:

On adopte un modèle simplifié du premier ordre pour faciliter l'étude de l'asservissement :

$$H(p) = \frac{K}{1 + \tau p}$$



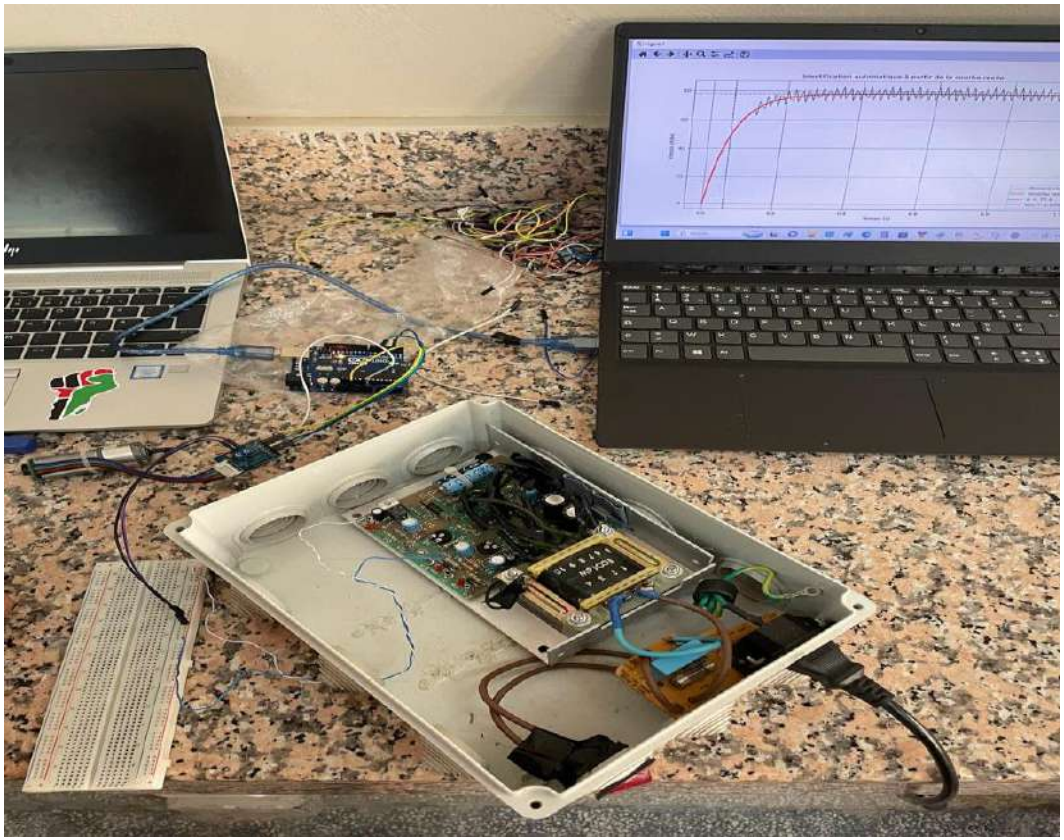
➡ MODELE VALIDE

K = 77.8
tau = 0.060 s

II. Modèle du moteur :

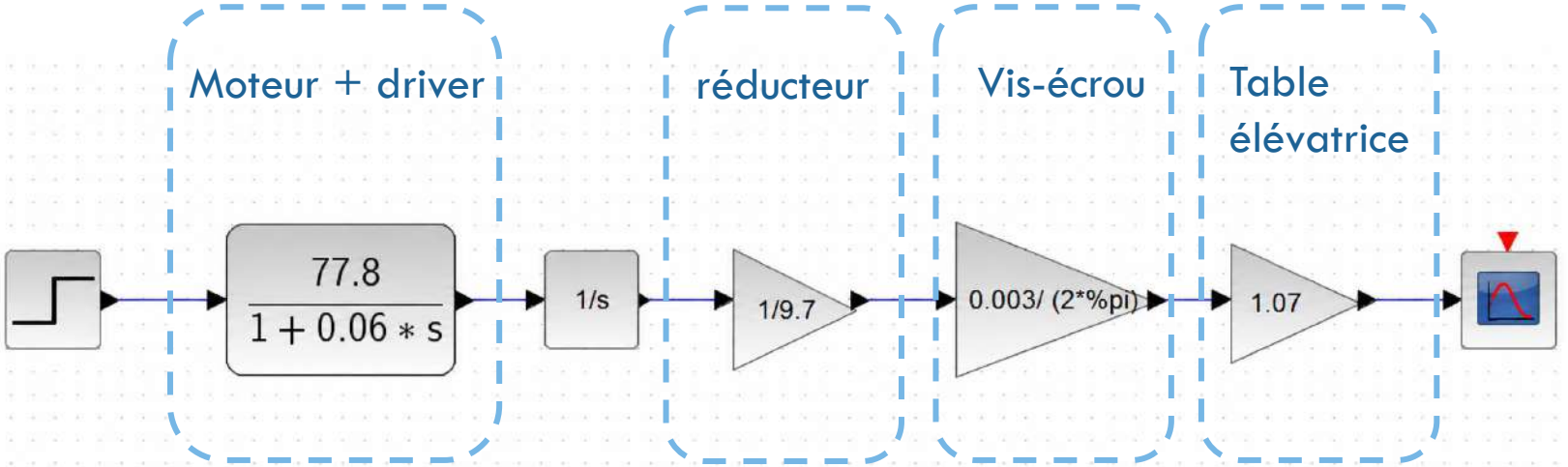
bilan:

BLOC= Hacheur + Moteur + Réducteur + Codeur



II. Modèle du moteur :

Modèle final :



Simulation avec Scilab

II. Modèle du moteur :

Boucle d'asservissement sans correction :

La masse transportée :

$$m = 10.25 \text{ kg}$$

alors :

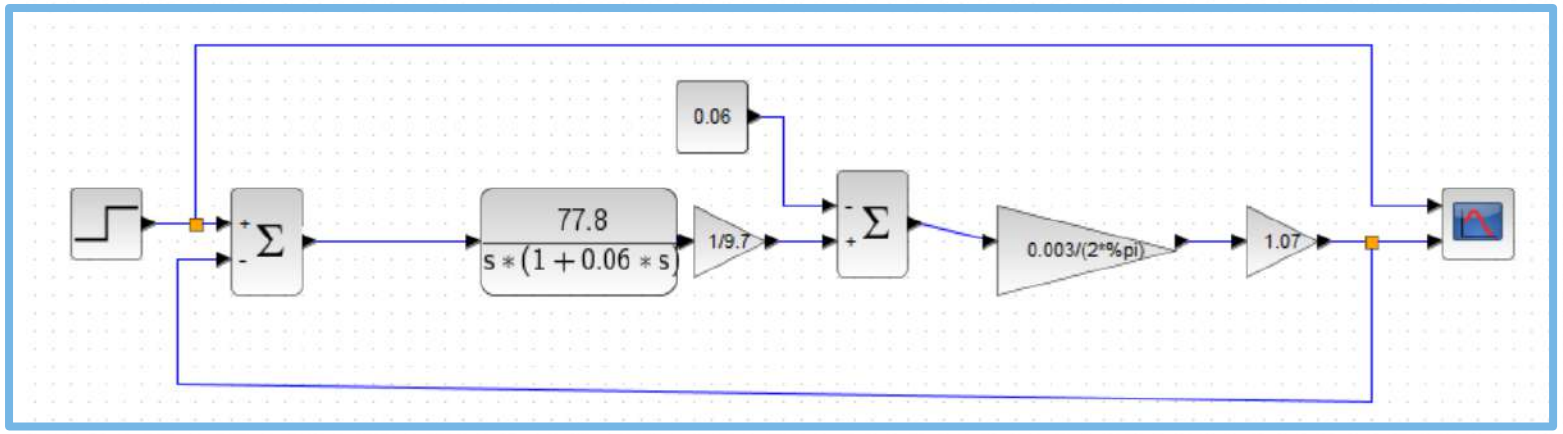
$$F = mg$$

$$F = 10.25 \times 9.81 \approx 100.6\ N$$

On a : $$C_r = \frac{F\,p}{2\pi\,\eta}$$

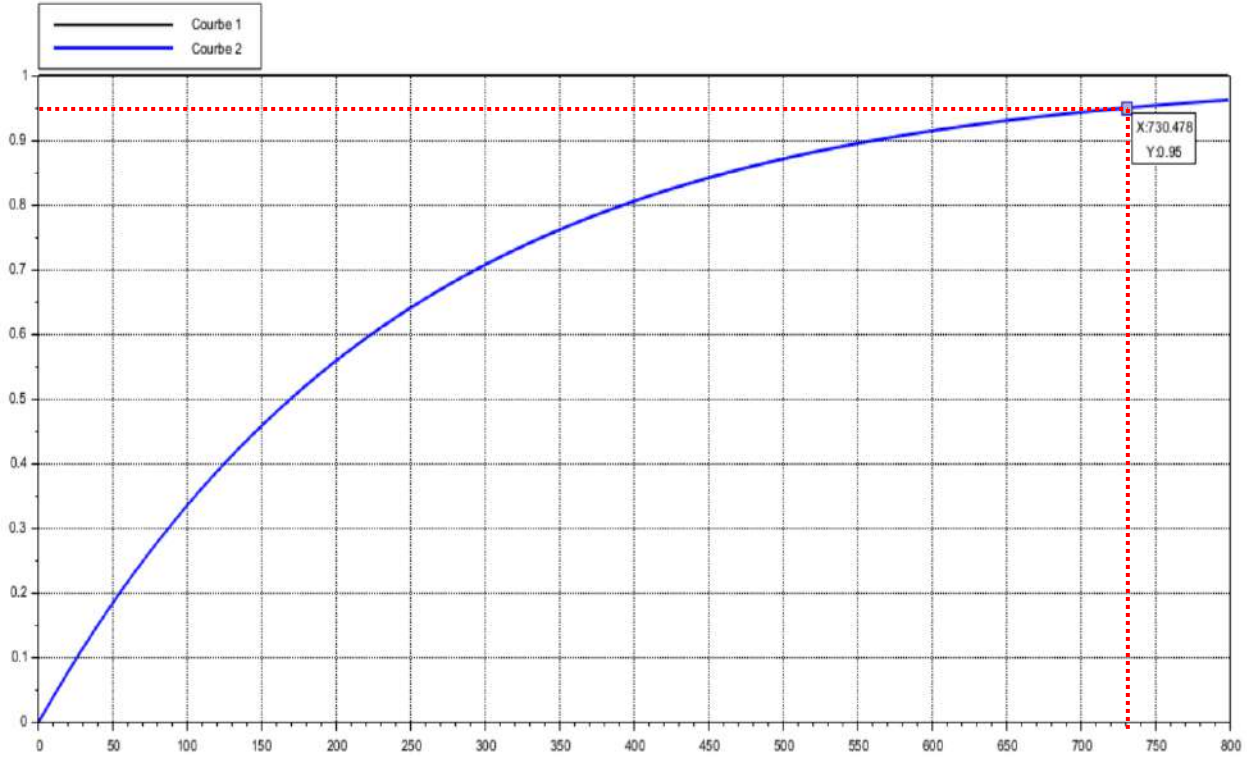
Avec un rendement de 0,8 on obtient :

$$C_r \approx 0.06\ N \cdot m$$



II. Modèle du moteur :

Réponse a la consigne :



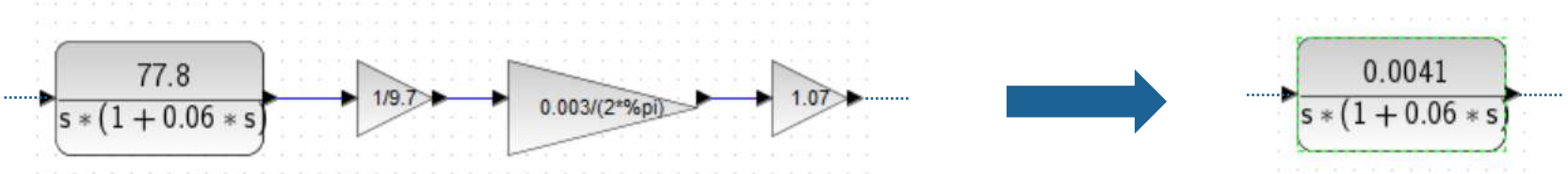
$Précision: \ \varepsilon_s = 0\%$ $(satisfait)$

$Rapidité: t_{5\%} = 730s$ $(non\ satisfait)$

II. Modèle du moteur :

Choix du correcteur :

regroupe les gains du moteur, du réducteur, de la vis et du capteur :



FTBO : $$G(p) = \frac{4,1 \times 10^{-3}}{p(1 + 0,06p)}$$

On choisit un correcteur proportionnel de gain Kp : $$C(p) = K_p$$

FTBF : $$H_{BF}(p) = \frac{4,1 \times 10^{-3} K_p}{0,06p^2 + p + 4,1 \times 10^{-3} K_p}$$

Sous la forme canonique : $$H(p) = \frac{\omega_n^2}{p^2 + 2m\omega_n p + \omega_n^2}$$

II. Modèle du moteur :

Choix du correcteur :

En divisant le dénominateur par 0,06 :

$$H_{BF}(p) = \frac{\frac{4,1 \times 10^{-3} K_p}{0,06}}{p^2 + \frac{1}{0,06}p + \frac{4,1 \times 10^{-3} K_p}{0,06}}$$

Par identification :

$$\omega_n^2 = \frac{4,1 \times 10^{-3} K_p}{0,06} \quad \text{et} \quad 2m\omega_n = \frac{1}{0,06}$$

On impose un temps de réponse à 5 % :

$$t_r = 10\,s$$

Pour un système du second ordre :

$$t_r \approx \frac{3}{\omega_n}$$

d'où :

$$\omega_n = \frac{3}{10} = 0,3\ rad/s$$

on obtient :

$$K_p \approx 52,7$$

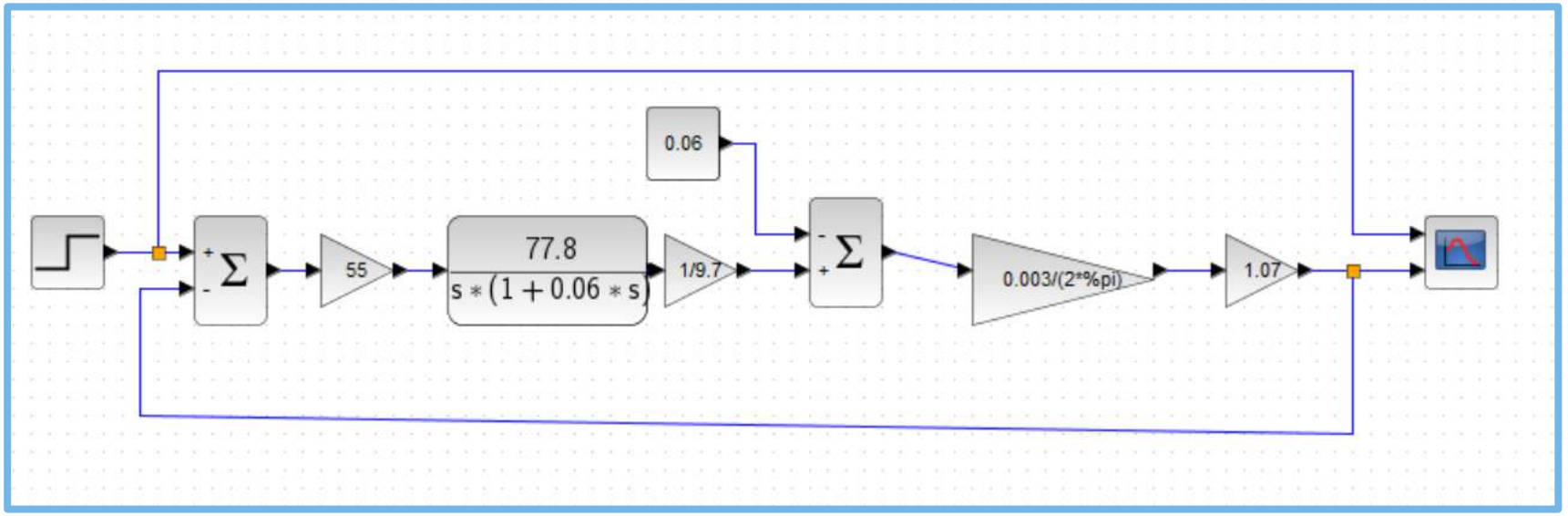
On choisit donc :

$$\boxed{K_p = 55}$$

II. Modèle du moteur :

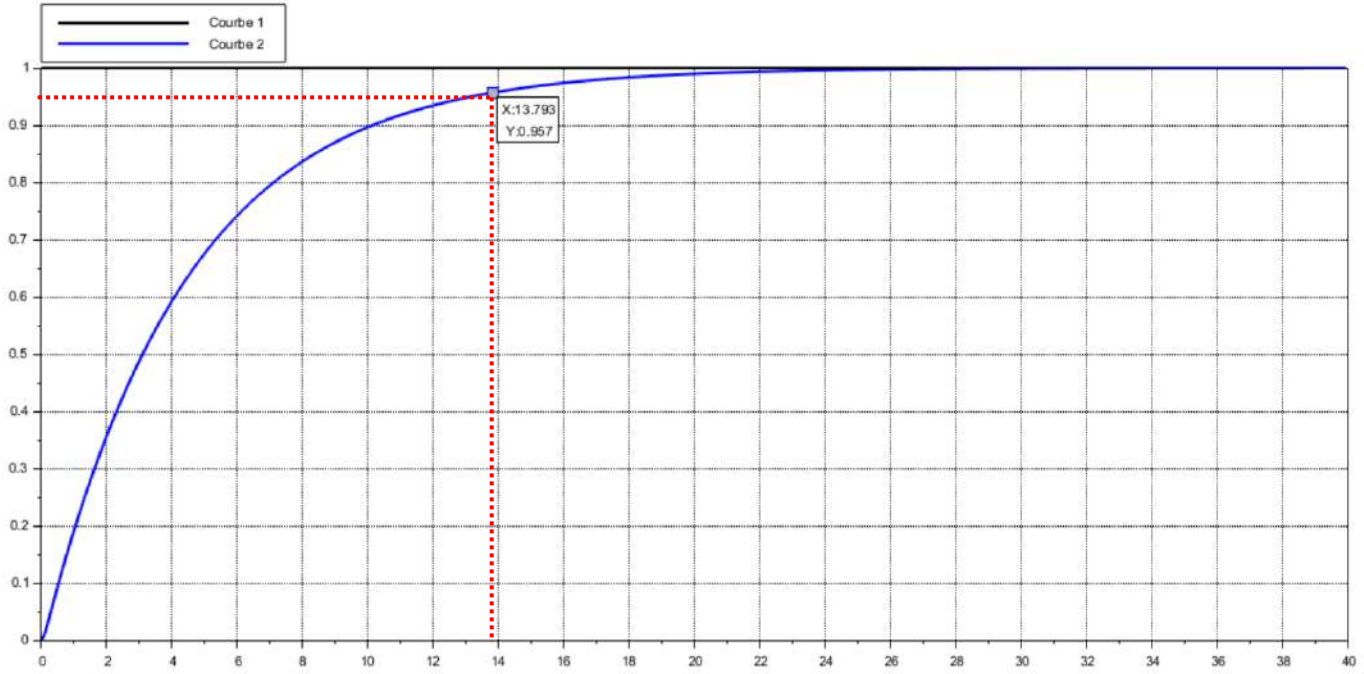
Choix du correcteur :

Boucle d'asservissement avec le correcteur proportionnel :



II. Modèle du moteur :

Modèle final en charge:



$Précision: \ \varepsilon_s = 0\% \quad (satisfait)$

$Rapidité: t_{5\%} = 13{,}7\ s \quad (satisfait)$

Objectif 4

- Choix du capteur

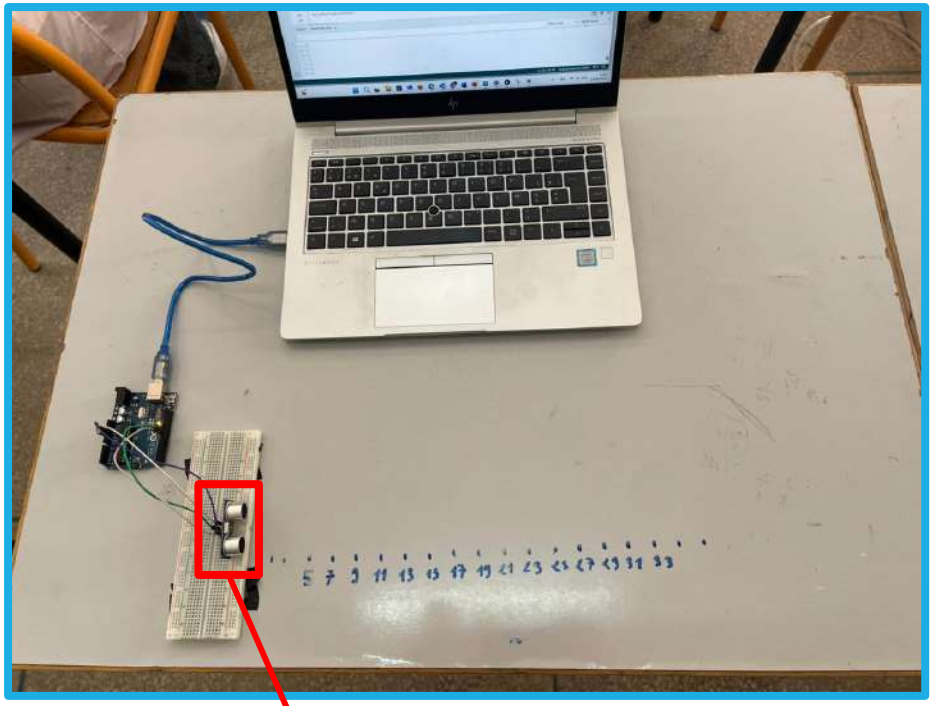
I. Justifier le choix du type de robot :

Cahier de charge :

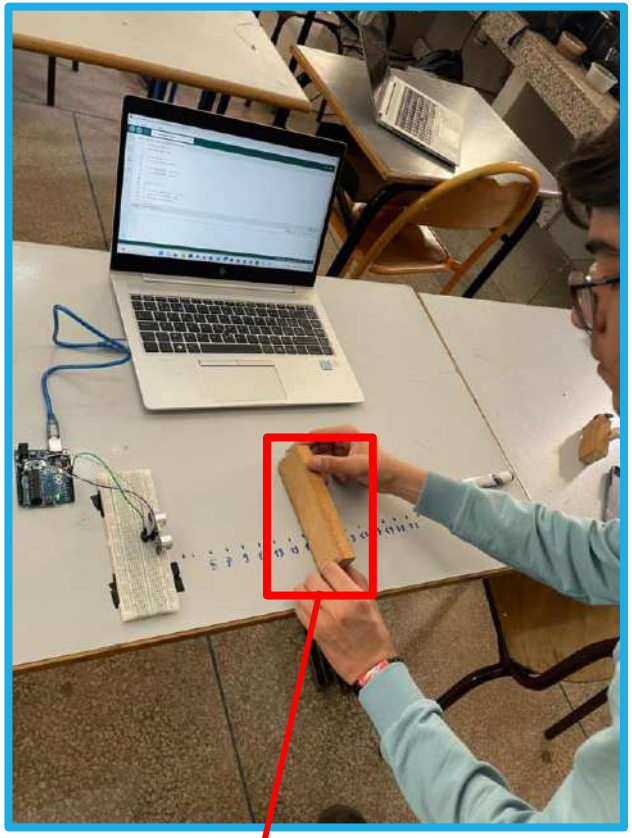
Exigence	Critère	Valeur
Détecter l'emplacement vide	Lineaire	A partir de 4 cm
Précision	Un pourcentage d'erreur faible	$\varepsilon \leq 5\%$

I. Justifier le choix du type de robot :

Expérience avec le capteur ultrason HC- SR04



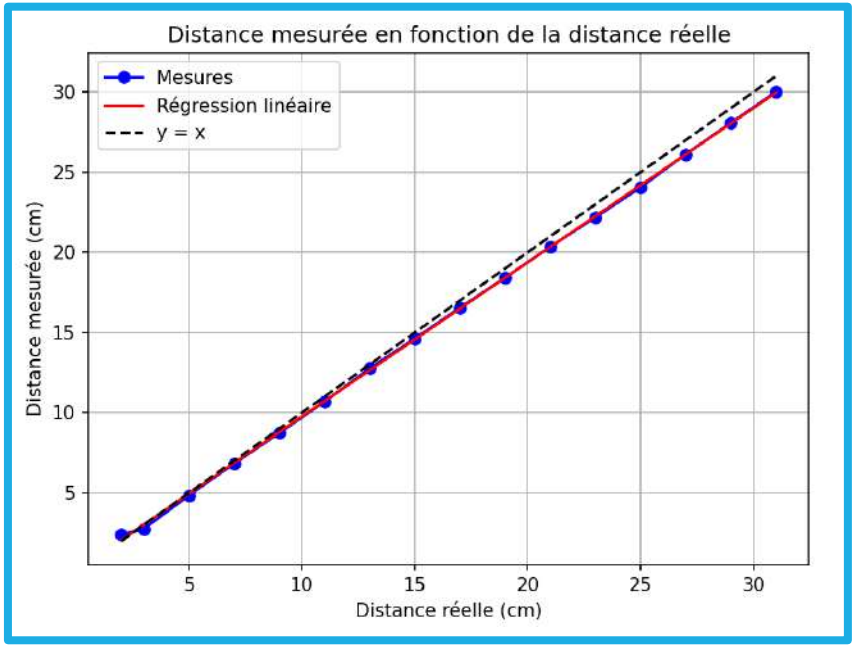
HC-SR04



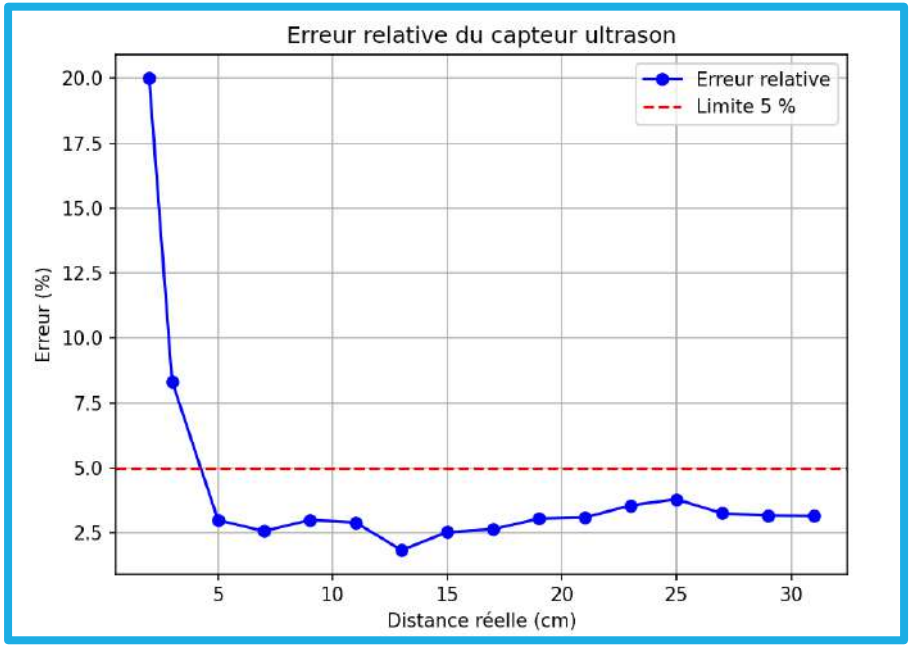
Obstacle

I. Justifier le choix du type de robot :

Expérience avec le capteur ultrason HC- SR04



Courbe des valeurs mesurables en fonction de celles réelles

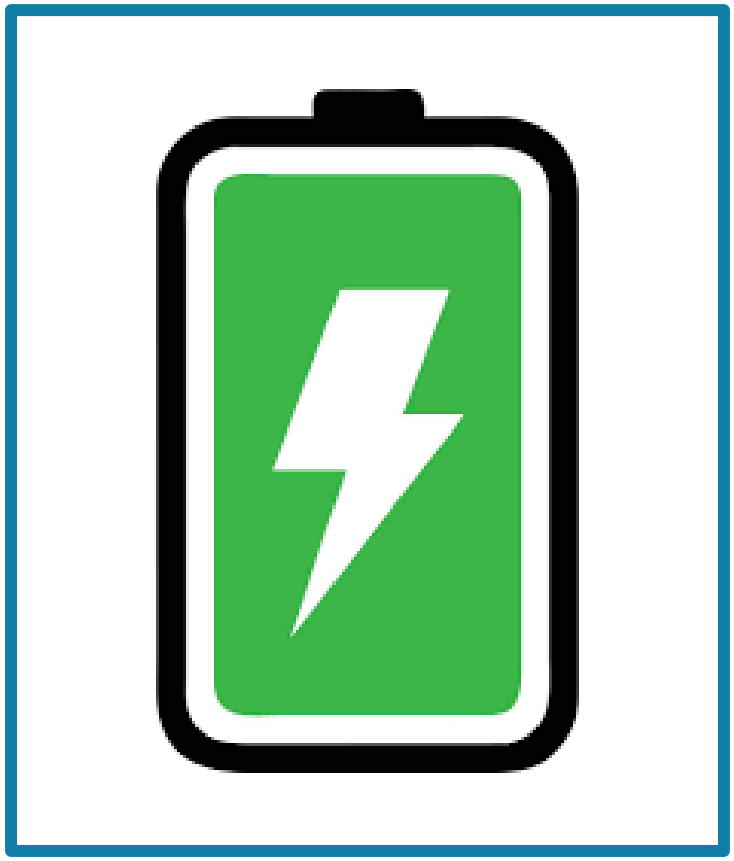


Courbe d'erreur commise à chaque distance

➡ Le capteur présente un comportement linéaire à partir de 4 cm. L'erreur relative reste inférieure à 5 % à partir de 5 cm, ce qui permet son utilisation pour la mesure de distance dans notre système avec une précision satisfaisante.

Objectif 4

- Dimensionner la batterie



I. Dimensionnement de la batterie :

Choix de la batterie

Capacité : $C = \frac{P.\Delta t}{U_{batt}} \Rightarrow C = 6.43Ah$

Composant		Puissance (W)
Plateforme	2×Hacheur L298N	50
	4×Codeur incrémental	0.4
	1×Ultrason HC-SR04	0.075
	1×Arduino Mega	1.68
Table élévatrice	2×Hacheur L298N	50
	1×Codeur incrémental	0.1
	1×Ultrason HC-SR04	0.075
Communication	1×Lecteur RFID	0.2
	1×Afficheur LCD	0.2
Puissance totale		102.73

Batterie choisie:

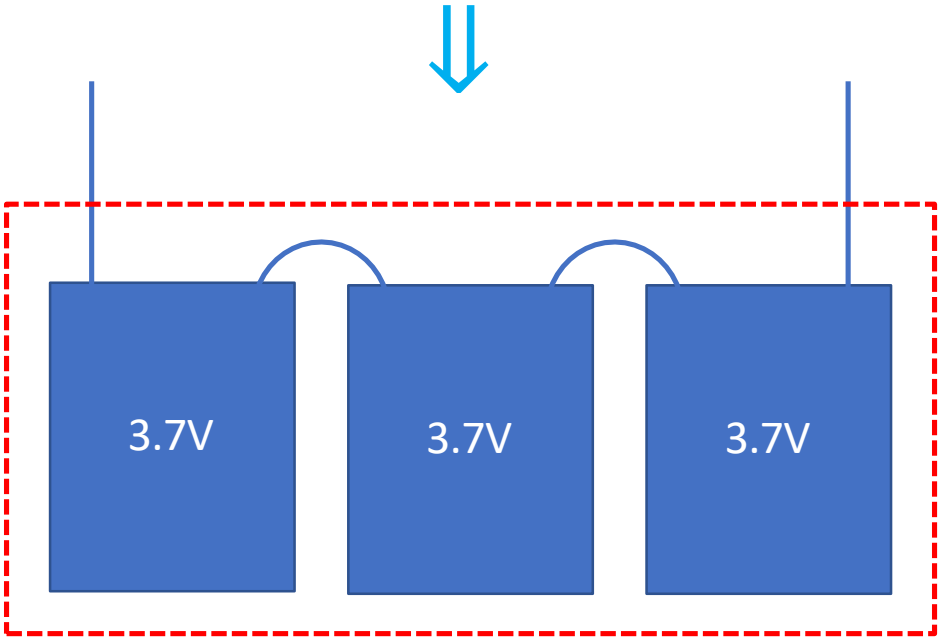
Model: LP3SR
Configuration: 3S1P
Voltage: 11.1V
Capacity: 10000mAh

Durée d'un cycle de fonctionnement : $\Delta t = 45min$

I. Dimensionnement de la batterie :

Conception du chargeur

La configuration de la batterie:
3S1P



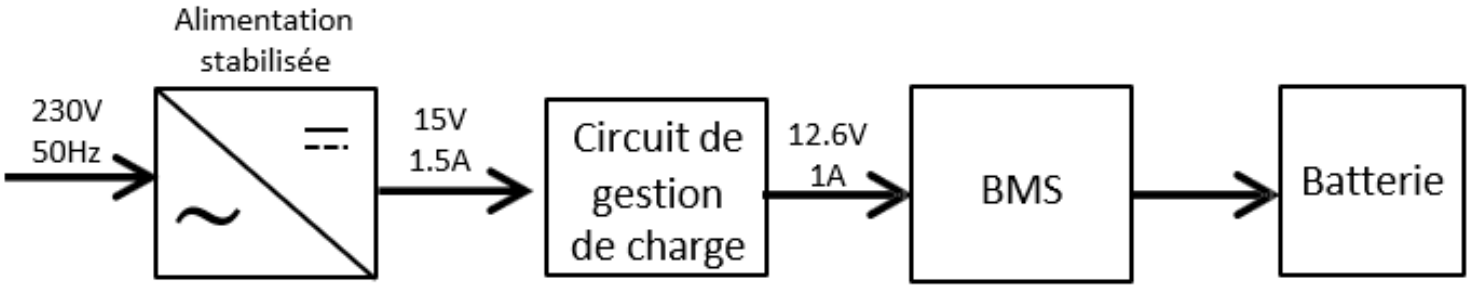
Tension d'une cellule: 3.7V
Courant de charge d'une cellule: 1A
Tension de charge nominal: 4.2V

I. Dimensionnement de la batterie :

Conception du chargeur :

Caractéristiques du chargeur:

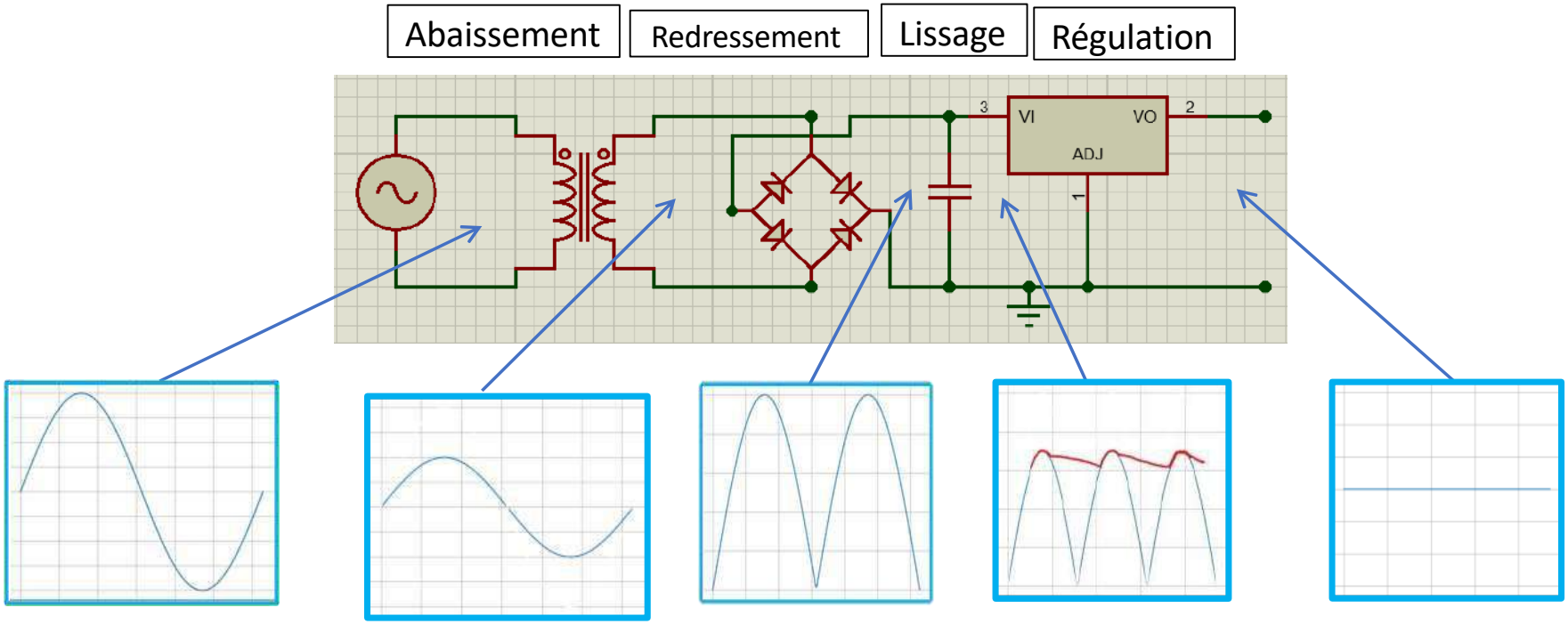
- ✓ Tension de sortie: U = 12.6V
- ✓ Courant de sortie: I = 1A



Structure interne du chargeur

I. Dimensionnement de la batterie :

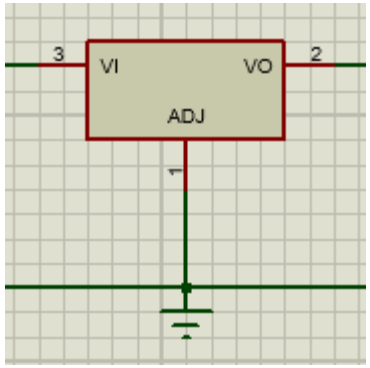
Alimentation stabilisée :



I. Dimensionnement de la batterie :

Alimentation stabilisée :

Régulateur

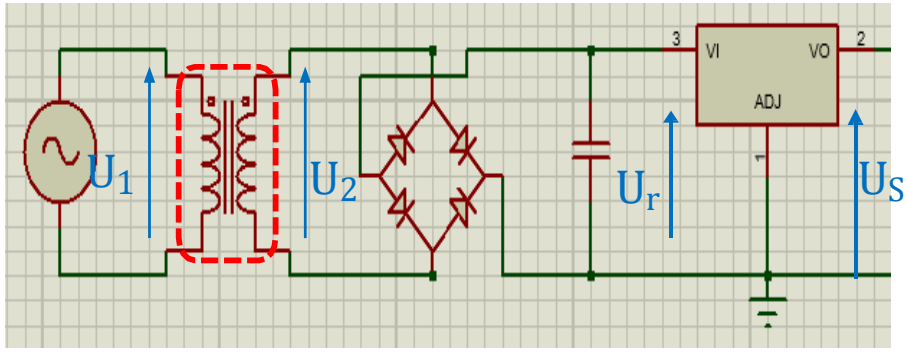


$U = 15V$

$I = 1.5A$

Régulateur 15V/1.5A
7815, boitier TO220

Transformateur



$U_2 = U_r + 2 \times U_{seuil,diode} = 18.7\text{V}$

$m = \frac{U_2}{U_1} = 0.083$

$P = U.I = 28.5VA$

$U_2 \approx 19V$

$m \approx 0.083$

$P = 28.5VA$

I. Dimensionnement de la batterie :

Alimentation stabilisée :

Condensateur:

$$U = U_r \quad \Rightarrow \quad U = 17.5$$

Capacité nécessaire:

$$I = C\frac{dU}{dt} \quad \Rightarrow \quad I.T = C.\Delta U$$

$$\Rightarrow \quad C = \frac{I}{f.\Delta U} = 0.001714F$$

$$\boxed{U = 17.5}$$

$$\boxed{C = 1.71mF}$$

Diode

$$V_{RRM} = 18.7 \times \sqrt{2}$$
$$I_{moy} = 1.5$$
$$I_{RMS} = 1.5/\sqrt{2}$$

$$\boxed{V_{RRM} = 26.4V}$$

$$\boxed{I_{moy} = 1.5A}$$

$$\boxed{I_{RMS} = 1A}$$

$$\boxed{1N5822}$$

$$\boxed{V_{RRM} = 40V}$$

$$\boxed{I_{moy} = 3A}$$

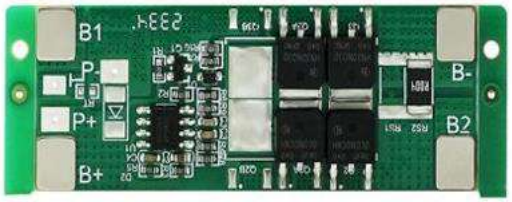
$$\boxed{I_{RMS} = 2.5A}$$

I. Dimensionnement de la batterie :

Alimentation stabilisée :

Rôle du BMS

- Protection contre la surcharge.
- Protection contre la décharge profonde.
- Protection contre les courts-circuits.
- Équilibrage des cellules.
- Coupure automatique lorsque la batterie est pleine



BMS

Objectif 2

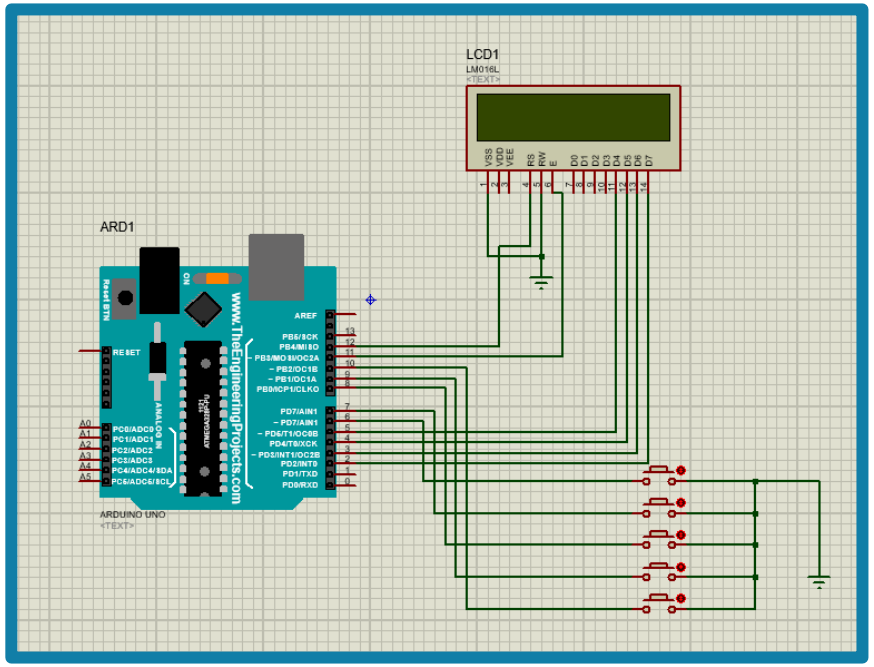
réaliser une interface homme machine

I. Dimensionnement de la batterie :

Montage réalisé par Isis porteus :

Les composant utilises :

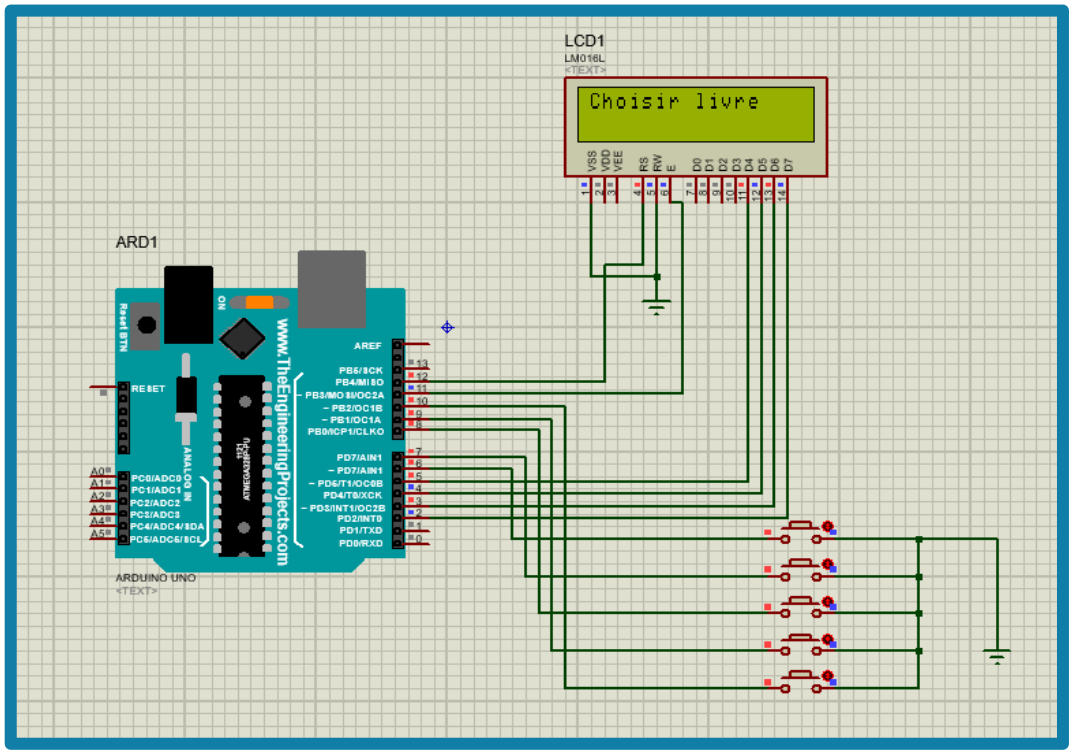
- Arduino Uno : unité de traitement
- LCD 16×2 : affichage des informations et du trajet optimal
- Bouton Livre 1 : sélection du livre 1
- Bouton Livre 2 : sélection du livre 2
- Bouton Livre 3 : sélection du livre 3
- Bouton Livre 4 : sélection du livre 4
- Bouton START : validation de la sélection et affichage du trajet optimal



I. Dimensionnement de la batterie :

Montage réalisé par Isis porteus :

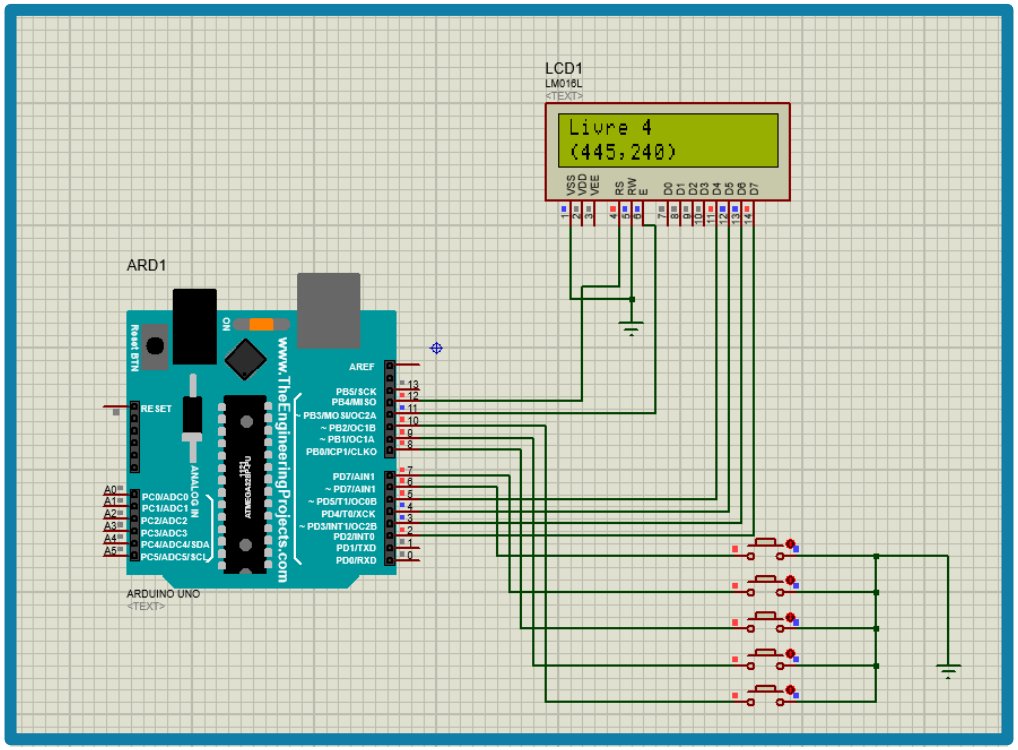
Le montage demande de saisir les livres :



I. Dimensionnement de la batterie :

Montage réalisé par Isis porteus :

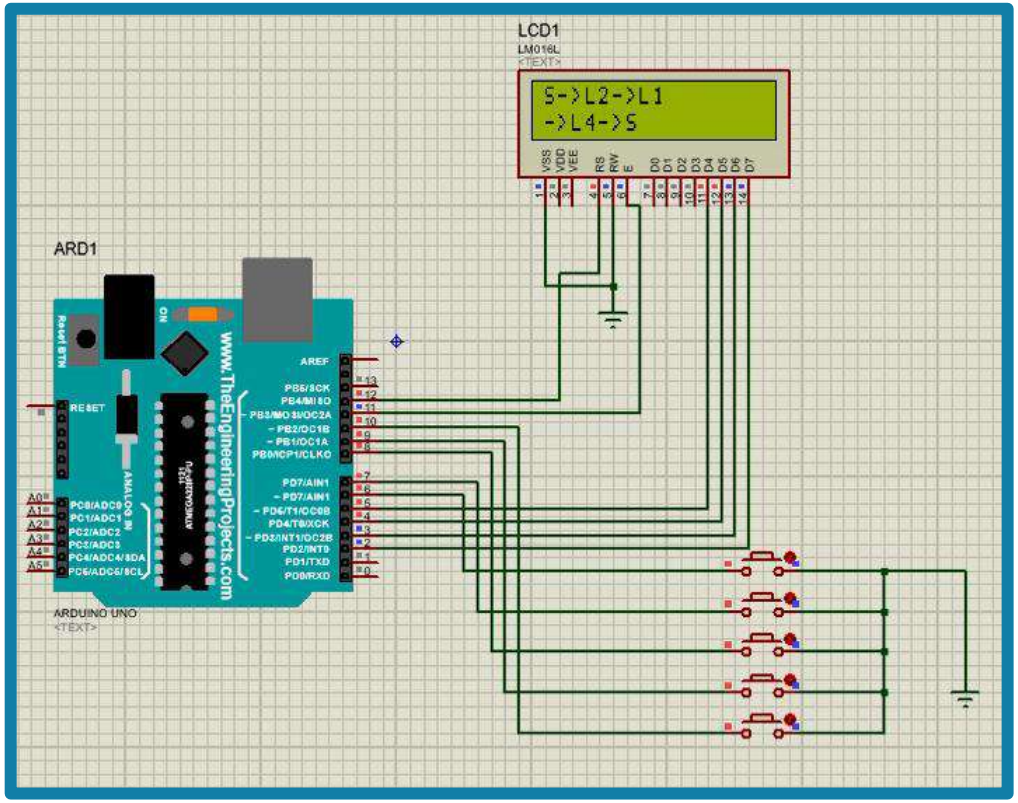
L'afficheur montre le nom et les coordonnées du livre choisis:



I. Dimensionnement de la batterie :

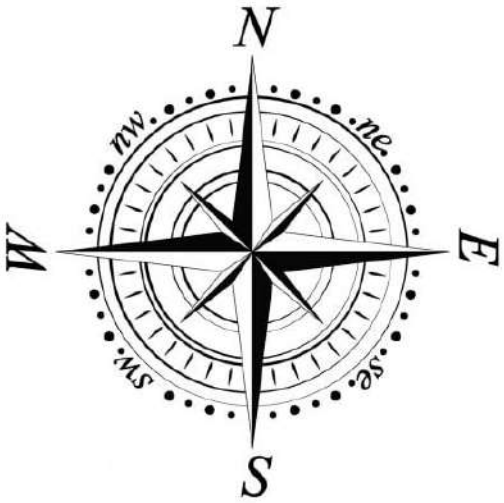
Montage réalisé par Isis porteus :

Apres la sélection des livres , la LCD affiche le trajet optimal .



PARTIE 2

ETUDE DE L'ALGORITHME

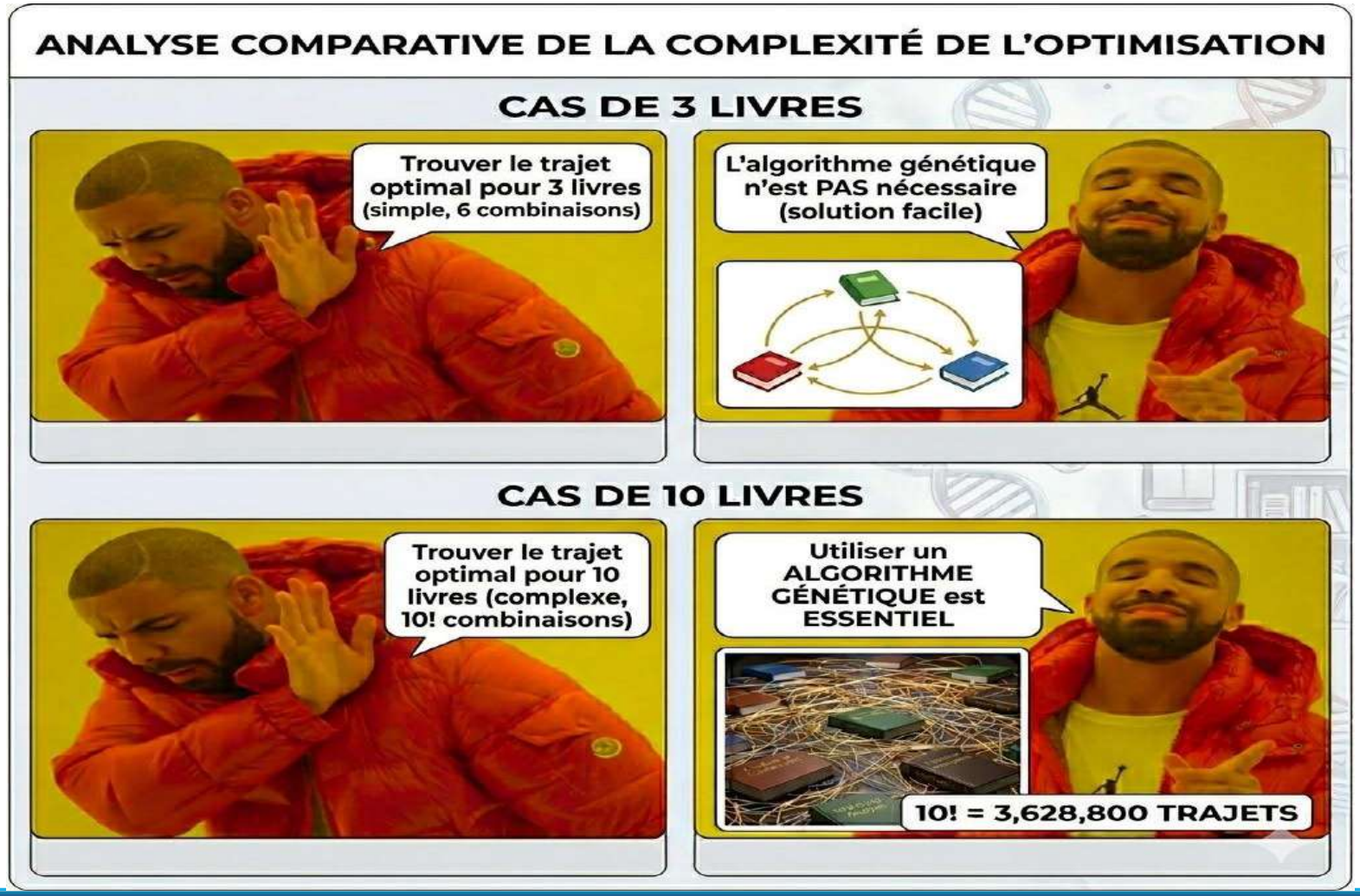


Objectif 2

- Assurer une navigation pour avoir un cycle optimal

I. Etude de l'algorithme :

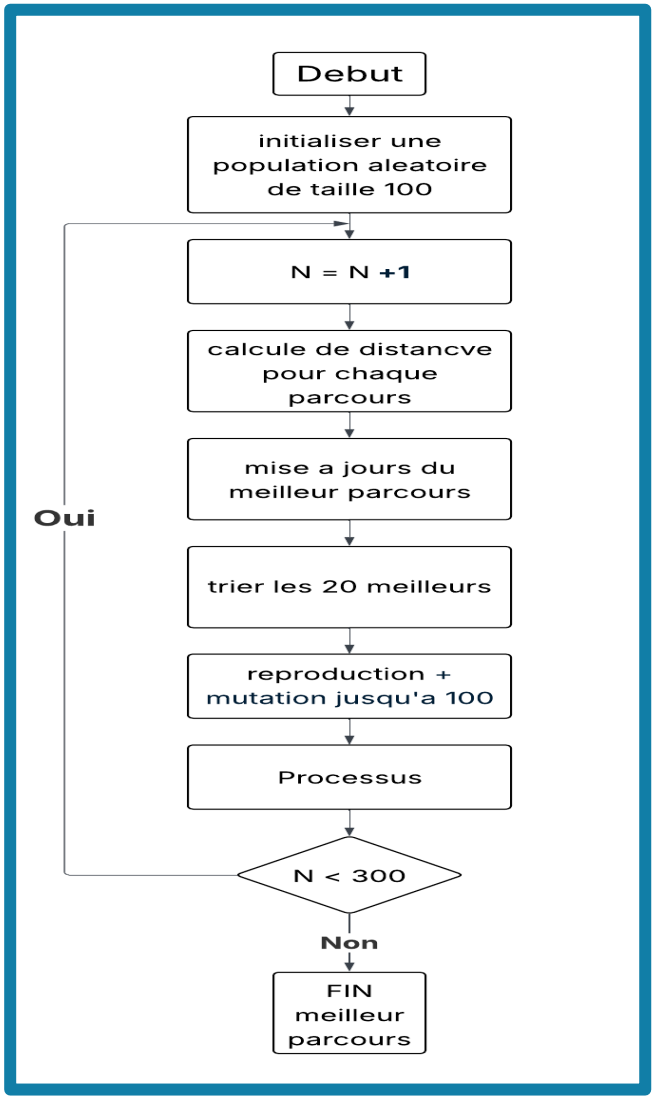
Problématique :



I. Etude de l'algorithme :

Algorithme génétique :

Un **algorithme génétique** est une méthode d'optimisation inspirée de l'évolution naturelle. Il génère plusieurs solutions possibles, puis sélectionne les meilleures. Ces solutions sont ensuite modifiées (mutation ou combinaison) pour créer de nouvelles solutions et améliorer progressivement le résultat.



I. Etude de l'algorithme :

Création de la population

Trajet 1

Trajet 2

Trajet 3

Trajet 100

100 population

- On crée 100 parcours aléatoires sans répétitions.
- Tous commencent par 12
- Tous finissent par 12

I. Etude de l'algorithme :

Calcule de la distance de chaque trajets :

On a : $D = \sum_{i=1}^{11} d(P_i, P_{i+1})$ avec :

- D = distance totale du trajet,
- P_i = point actuel,
- P_{i+1} = point suivant,
- $d(P_i, P_{i+1})$ = distance entre deux points successifs.

Si les deux points sont sur la même colonne :

déplacement vertical direct

$$d = |y_2 - y_1|$$

Si un couloire intervient entre eux :

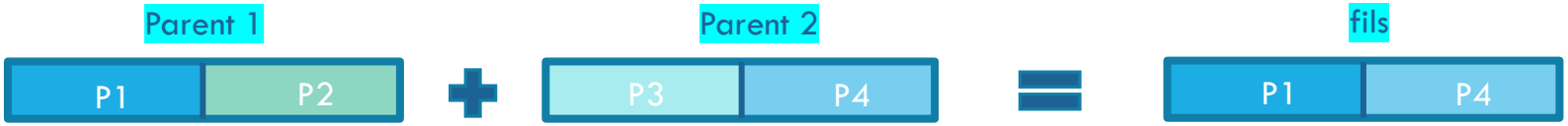
- monte/descend jusqu'au couloir,
- avance horizontalement,
- redescend/remonte vers le point final

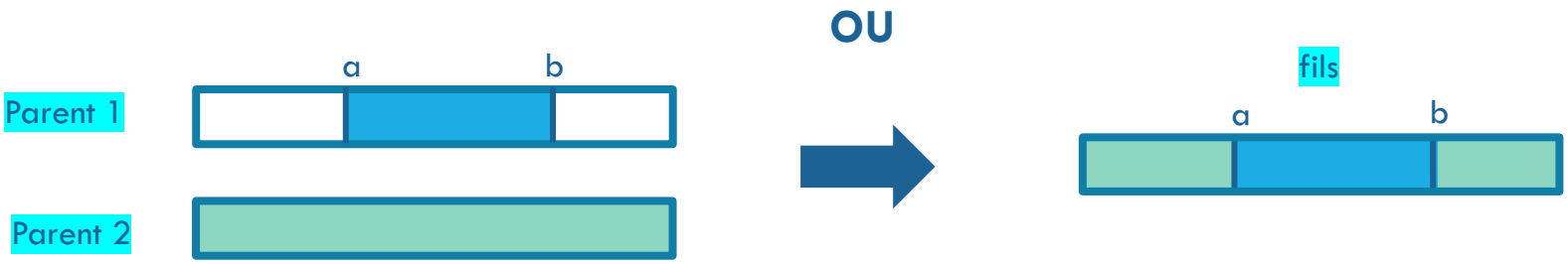
$$d = |y_1 - y_c| + |x_1 - x_2| + |y_2 - y_c|$$

I. Etude de l'algorithme :

Croisement (crossover) :

- **Croisement (Crossover)** : c'est le mélange de deux solutions parentes pour créer une nouvelle solution.
- On trouve souvent deux méthodes :





- On choisi la deuxième méthode , ou prend l' intervalle [a,b] du parent 1 comme il est et on complet le fils par le parent 2 sans répétition des points

I. Etude de l'algorithme :

Mutation :

- **Mutation** : c'est modification aléatoire d'une petite partie d'une solution pour ajouter de la diversité.
- Dans notre cas , deux points du trajet sont échangés avec une de probabilité 30% .

```
if random.random() < 0.3:
    i, j = random.sample(range(1, 13), 2)
    enfant[i], enfant[j] = enfant[j], enfant[i]
```

Avant l'exécution

Apres l'exécution

II. simulation :

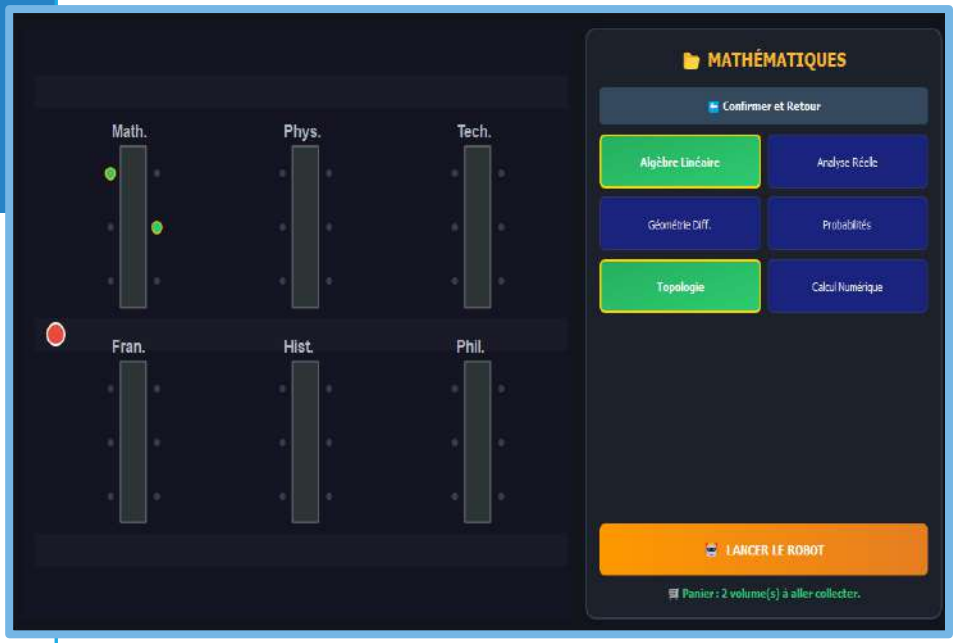
Modélisation de la bibliothèque :

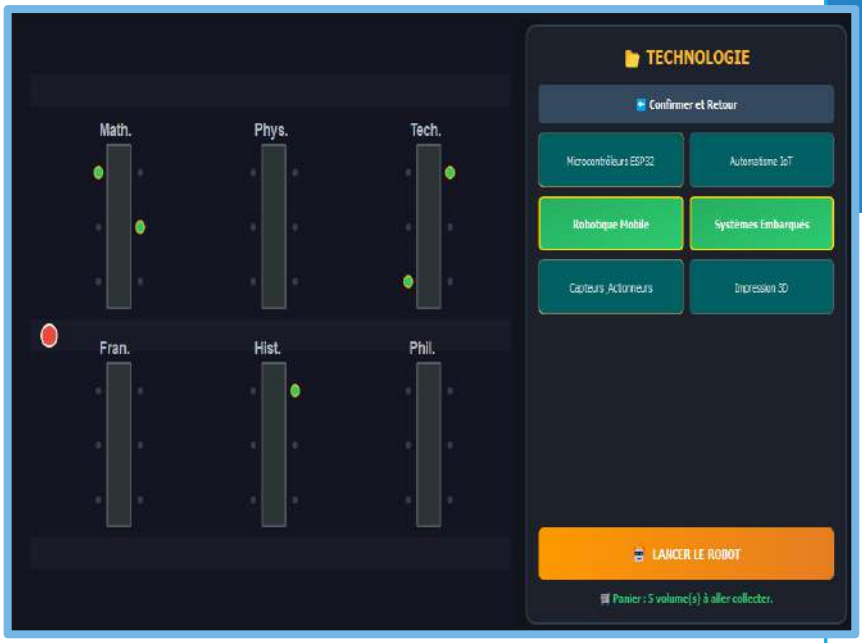
Nous disposons de :

- 6 rayonnages dans la bibliothèque

II. simulation :

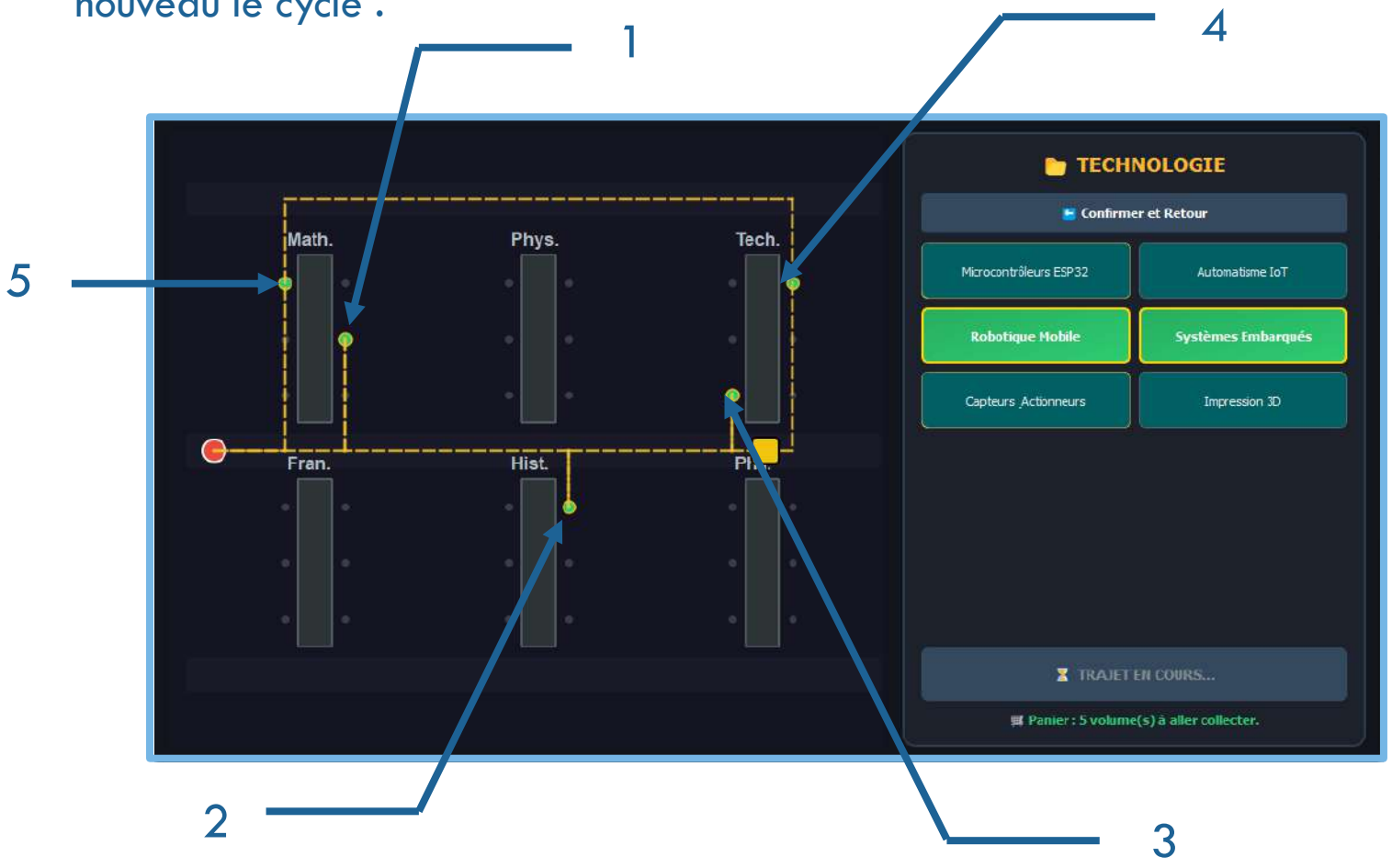
- On veut remettre 5 de différents matières a leurs emplacements convenables





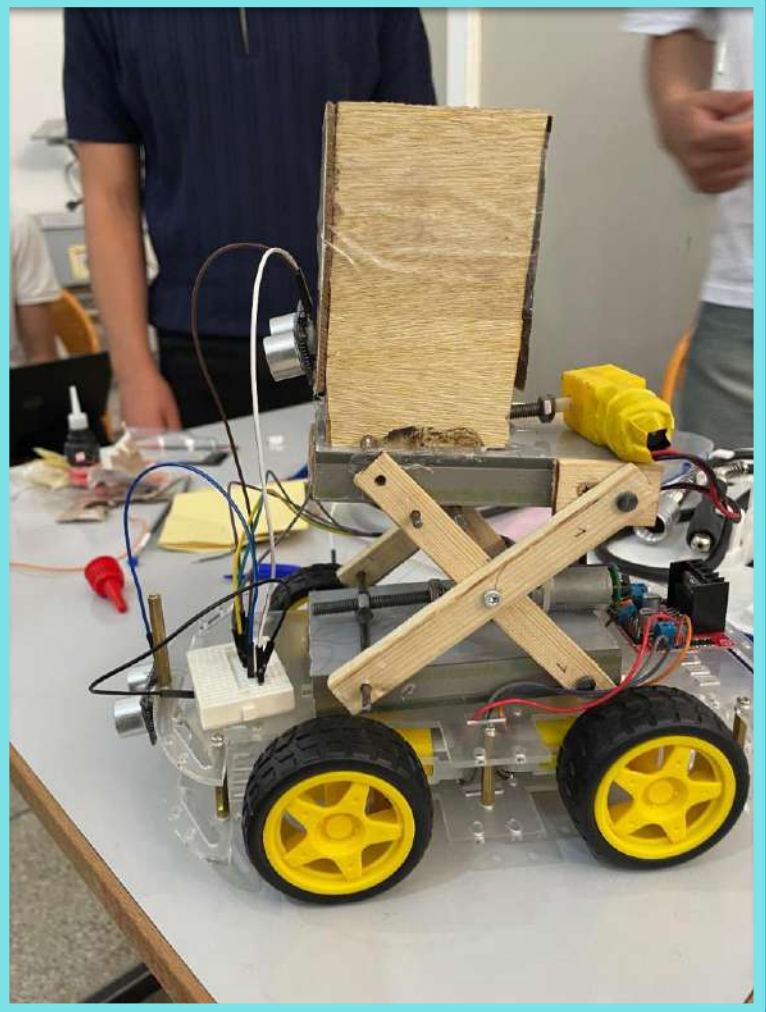
II. simulation :

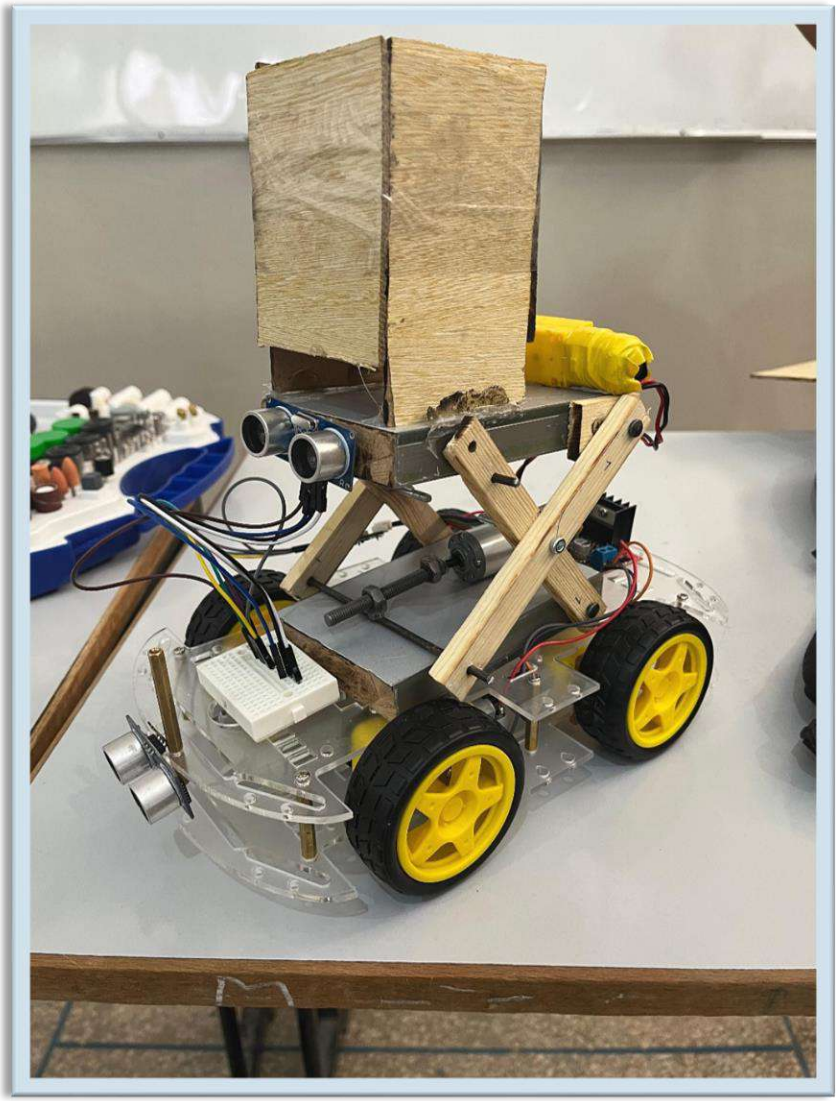
- On lance la simulation et le robot commence le cycle dans l'ordre optimale .puis , il revient a ca position initiale pour recommencer a nouveau le cycle .



PARTIE 3

REALISATION DU PROTOTYPE





conclusion

MERCI POUR VOTRE ATTENTION

Algorithme génétique

<tmp 1>

```python
from PyQt5.QtWidgets import (QWidget, QApplication, QHBoxLayout, QVBoxLayout,
                             QPushButton, QLabel, QFrame, QGridLayout, QSizePolicy, QScrollArea)
from PyQt5.QtCore import QTimer, Qt, QPointF
from PyQt5.QtGui import (QPainter, QColor, QPen, QFont, QLinearGradient,
                         QBrush, QPainterPath)
import sys
import random

# ==========================================
# PARTIE 1 : BIBLIOTHÈQUE ET LIVRES INDIVIDUELS
# ==========================================
class Bibliotheque:
    def __init__(self):
        # Élargissement léger de la zone pour accueillir confortablement le 6ème rayon
        self.largeur = 620
        self.hauteur = 500
        self.y_passages = [50, 275, 475]

        # Structure des 6 rayonnages (3 en haut, 3 en bas)
        self.murs_specs = [
            # Rangée du HAUT (y = 100)
            {"matiere": "Mathématiques", "rect": (100, 100, 30, 150)},
            {"matiere": "Physique",      "rect": (300, 100, 30, 150)},
            {"matiere": "Technologie",   "rect": (500, 100, 30, 150)},

            # Rangée du BAS (y = 300) -> Mêmes coordonnées X (100, 300, 500)
            {"matiere": "Français",      "rect": (100, 300, 30, 150)},
            {"matiere": "Histoire",      "rect": (300, 300, 30, 150)},
            {"matiere": "Philosophie",   "rect": (500, 300, 30, 150)},
        ]

        # Point de départ fixe (Dépôt)
        self.point_depot = (25, 275)
        self.points = [self.point_depot]

        # Catalogue enrichi à 6 matières (6 volumes chacune)
        self.catalogue = {
            "Mathématiques": ["Algèbre Linéaire", "Analyse Réelle", "Géométrie Diff.", "Probabilités", "Topologie", "Calcul Numérique"],
            "Physique":      ["Mécanique Quantique", "Thermodynamique", "Optique Ondulatoire", "Électromagnétisme", "Relativité", "Physique Nucléaire"],
            "Technologie":   ["Microcontrôleurs ESP32", "Automatisme IoT", "Robotique Mobile", "Systèmes Embarqués",
```

Algorithme génétique

```
"Capteurs & Actionneurs", "Impression 3D"],
            "Français":       ["Molière - Œuvres", "Victor Hugo - Poésie", "Le Surréalisme", "Grammaire Structurale",
"Littérature Classique", "Analyse de Textes"],
            "Histoire":       ["Antiquité Romaine", "Histoire du Maroc", "Moyen-Âge Central", "Révolution Française",
"Première Guerre Mondiale", "Guerre Froide"],
            "Philosophie":    ["Platon & Socrate", "Métaphysique", "Éthique & Morale", "Philo Politique",
"Épistémologie", "Existentialisme"]
        }

        self.coords_livres_physiques = {}
        self._generer_placement_livres()

    def _generer_placement_livres(self):
        """Calcule l'emplacement exact de chaque livre (3 à gauche, 3 à droite du rayon)."""
        for rayon in self.murs_specs:
            mat = rayon["matiere"]
            mx, my, mw, mh = rayon["rect"]
            titres = self.catalogue[mat]
            y_positions = [my + 25, my + 75, my + 125]

            for i, titre in enumerate(titres):
                if i < 3:
                    lx = mx - 12  # Côté gauche
                    ly = y_positions[i]
                else:
                    lx = mx + mw + 12  # Côté droit
                    ly = y_positions[i - 3]
                self.coords_livres_physiques[titre] = (lx, ly)

    def réinitialiser_points(self):
        self.points = [self.point_depot]

    def ajouter_objectif_livre(self, coord_livre):
        if coord_livre not in self.points:
            self.points.append(coord_livre)

    def calculer_meilleur_passage(self, p1_idx, p2_idx):
        x1, y1 = self.points[p1_idx]
        x2, y2 = self.points[p2_idx]

        if abs(x1 - x2) < 5:
            if not self._est_bloque_vertical(x1, y1, y2):
```

Algorithme génétique

```
77          if abs(x1 - x2) < 5:
78              if not self._est_bloque_vertical(x1, y1, y2):
79                  return abs(y1 - y2), None
80
81          meilleure_dist = float('inf')
82          meilleur_y = self.y_passages[1]
83          for y_couloir in self.y_passages:
84              d = abs(y1 - y_couloir) + abs(x1 - x2) + abs(y2 - y_couloir)
85              if d < meilleure_dist:
86                  meilleure_dist = d
87                  meilleur_y = y_couloir
88          return meilleure_dist, meilleur_y
89
90      def _est_bloque_vertical(self, x, y1, y2):
91          for rayon in self.murs_specs:
92              mx, my, mw, mh = rayon["rect"]
93              if mx <= x <= mx + mw:
94                  if min(y1, y2) < my + mh and max(y1, y2) > my:
95                      return True
96          return False
97
98  # ==========================================
99  # PARTIE 2 : ALGORITHME GÉNÉTIQUE
100 # ==========================================
101 class AlgoGenetiqueOptimise:
102     def __init__(self, biblio):
103         self.biblio = biblio
104         self.taille_pop = 100
105         self.generations = 250  # Légèrement augmenté pour gérer le point additionnel potentiel
106         n = len(self.biblio.points)
107
108         if n > 2:
109             self.population = [
110                 [0] + random.sample(range(1, n), n - 1) + [0]
111                 for _ in range(self.taille_pop)
112             ]
113             self.meilleur_parcours = None
114             self.meilleure_dist = float('inf')
115             self._evoluer()
116         else:
117             self.meilleur_parcours = list(range(n)) + [0] if n == 2 else [0, 0]
118             self.meilleure_dist = sum(self.biblio.calculer_meilleur_passage(self.meilleur_parcours[i],
```

Algorithme génétique

```
118 self.meilleur_parcours[i+1])[0] for i in range(len(self.meilleur_parcours)-1))
119
120     def calculer_distance(self, parcours):
121         return sum(self.biblio.calculer_meilleur_passage(parcours[i], parcours[i+1])[0]
122                    for i in range(len(parcours)-1))
123
124     def _crossover(self, p1, p2):
125         taille = len(p1) - 2
126         if taille < 2: return p1.copy()
127         a, b = sorted(random.sample(range(1, taille + 1), 2))
128         enfant = [None] * len(p1)
129         enfant[0], enfant[-1] = 0, 0
130         enfant[a:b] = p1[a:b]
131         p2_genes = [gene for gene in p2 if gene not in enfant]
132         idx_p2 = 0
133         for i in range(len(enfant)):
134             if enfant[i] is None:
135                 enfant[i] = p2_genes[idx_p2]
136                 idx_p2 += 1
137         return enfant
138
139     def _evoluer(self):
140         for _ in range(self.generations):
141             self.population.sort(key=self.calculer_distance)
142             d_min = self.calculer_distance(self.population[0])
143             if d_min < self.meilleure_dist:
144                 self.meilleure_dist = d_min
145                 self.meilleur_parcours = self.population[0].copy()
146             elites = self.population[:20]
147             nouvelle_pop = elites.copy()
148             while len(nouvelle_pop) < self.taille_pop:
149                 parent1, parent2 = random.sample(elites, 2)
150                 enfant = self._crossover(parent1, parent2)
151                 if len(enfant) > 3 and random.random() < 0.3:
152                     i, j = random.sample(range(1, len(enfant) - 1), 2)
153                     enfant[i], enfant[j] = enfant[j], enfant[i]
154                 nouvelle_pop.append(enfant)
155             self.population = nouvelle_pop
156
157 # ==========================================
158 # PARTIE 3 : ROBOT MOBILISÉ
159 # ==========================================
```

Algorithme génétique

```python
class RobotOptimise:
    def __init__(self, biblio):
        self.biblio = biblio
        self.algo = AlgoGenetiqueOptimise(biblio)
        self.parcours = self.algo.meilleur_parcours
        self.chemin_complet = self._generer_points()
        self.idx = 0
        self.x, self.y = self.chemin_complet[0]
        self.vitesse = 6
        self.termine = False

    def _generer_points(self):
        chemin = []
        for i in range(len(self.parcours)-1):
            p1, p2 = self.parcours[i], self.parcours[i+1]
            x1, y1 = self.biblio.points[p1]
            x2, y2 = self.biblio.points[p2]
            dist, y_transit = self.biblio.calculer_meilleur_passage(p1, p2)
            if i == 0: chemin.append((x1, y1))
            if y_transit is not None:
                chemin.append((x1, y_transit))
                chemin.append((x2, y_transit))
            chemin.append((x2, y2))
        return chemin

    def avancer(self):
        if self.idx >= len(self.chemin_complet) - 1:
            self.termine = True
            return True
        tx, ty = self.chemin_complet[self.idx + 1]
        dx = tx - self.x
        dy = ty - self.y
        if abs(dx) > 0.1:
            pas = min(self.vitesse, abs(dx))
            self.x += pas if dx > 0 else -pas
        elif abs(dy) > 0.1:
            pas = min(self.vitesse, abs(dy))
            self.y += pas if dy > 0 else -pas
        if abs(self.x - tx) < 0.1 and abs(self.y - ty) < 0.1:
            self.x, self.y = tx, ty
            self.idx += 1
        return False
```

Algorithme génétique

```
# ============================================
# PARTIE 4 : COMPOSANTS INTERFACE
# ============================================
class BoutonMenuPrincipal(QPushButton):
    THEMES = {
        "Mathématiques": ("#1a237e", "📐"),
        "Physique":      ("#1b5e20", "⚛️"),
        "Technologie":   ("#006064", "🛠️"),  # Thème couleur pour la Technologie
        "Français":      ("#880e4f", "📖"),
        "Histoire":      ("#bf360c", "🏛️"),
        "Philosophie":   ("#4a148c", "💭"),
    }
    def __init__(self, nom, parent=None):
        super().__init__(parent)
        self.nom = nom
        couleur, emoji = self.THEMES.get(nom, ("#333", "🔹"))
        self.setText(f"{emoji}  {nom}")
        self.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Preferred)
        self.setMinimumHeight(52)
        self.setStyleSheet(f"""
            QPushButton {{
                background-color: {couleur}; color: white; font-size: 13px; font-weight: bold; border-radius: 8px;
                border: 1px solid rgba(255,255,255,0.1);
            }}
            QPushButton:hover {{ background-color: qlineargradient(x1:0, y1:0, x2:0, y2:1, stop:0 {couleur}, stop:1
#555); border: 2px solid white; }}
        """)

class BoutonLivreSelection(QPushButton):
    def __init__(self, nom, couleur_base="#2c3e50", parent=None):
        super().__init__(parent)
        self.nom = nom
        self.setText(nom)
        self.setCheckable(True)
        self.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Preferred)
        self.setMinimumHeight(50)
        self.couleur_base = couleur_base
        self._update_style(False)
        self.toggled.connect(self._update_style)

    def _update_style(self, checked):
```

Algorithme génétique

```
243         if checked:
244             self.setStyleSheet("""
245                 QPushButton { background: qlineargradient(x1:0, y1:0, x2:1, y2:1, stop:0 #27ae60, stop:1 #2ecc71);
246                               color: white; font-weight: bold; font-size: 11px; border: 2px solid #FFD700; border-
    radius: 6px; }
247             """)
248         else:
249             self.setStyleSheet(f"QPushButton {{ background-color: {self.couleur_base}; color: #ecf0f1; font-size:
    11px; border: 1px solid rgba(255,255,255,0.1); border-radius: 6px; }} QPushButton:hover {{ background-color: #34495e;
    border: 1px solid #ffb733; }}")
250
251 # ==========================================
252 # PARTIE 5 : RENDU GRAPHIQUE DE LA CARTE
253 # ==========================================
254 class ZoneCarte(QWidget):
255     def __init__(self, biblio, panier):
256         super().__init__()
257         self.biblio = biblio
258         self.panier = panier
259         self.robot = None
260         self.setMinimumSize(640, 520)
261
262     def paintEvent(self, event):
263         p = QPainter(self)
264         p.setRenderHint(QPainter.Antialiasing)
265         p.fillRect(self.rect(), QBrush(QColor(20, 20, 35)))
266         p.translate(15, 10)
267
268         # Couloirs de passage horizontaux
269         p.setPen(Qt.NoPen)
270         for yc in self.biblio.y_passages:
271             p.setBrush(QColor(255, 255, 255, 6))
272             p.drawRect(0, yc - 15, self.biblio.largeur, 30)
273
274         # Tracé des structures des 6 rayonnages et de leurs livres
275         for rayon in self.biblio.murs_specs:
276             mat = rayon["matiere"]
277             mx, my, mw, mh = rayon["rect"]
278
279             p.setBrush(QColor(45, 52, 54))
280             p.setPen(QPen(QColor(99, 110, 114), 1.5))
281             p.drawRect(mx, my, mw, mh)
```

Algorithme génétique

```
282
283             p.setFont(QFont("Arial", 8, QFont.Bold))
284             p.setPen(QColor(178, 190, 195))
285             p.drawText(mx - 10, my - 8, mat[:4] + ".")
286
287             titres_du_rayon = self.biblio.catalogue[mat]
288             for titre in titres_du_rayon:
289                 lx, ly = self.biblio.coords_livres_physiques[titre]
290
291                 if titre in self.panier[mat]:
292                     p.setBrush(QColor(46, 204, 113)) # Vert si coché
293                     p.setPen(QPen(QColor(241, 196, 15), 1.5))
294                     taille = 11
295                 else:
296                     p.setBrush(QColor(116, 125, 140, 100)) # Gris neutre
297                     p.setPen(Qt.NoPen)
298                     taille = 7
299                 p.drawEllipse(int(lx) - taille//2, int(ly) - taille//2, taille, taille)
300
301         # Base / Dépôt
302         xd, yd = self.biblio.point_depot
303         p.setBrush(QColor(231, 76, 60))
304         p.setPen(QPen(Qt.white, 2))
305         p.drawEllipse(xd-10, yd-10, 20, 20)
306
307         # Affichage de la ligne de guidage du robot
308         if self.robot:
309             p.setPen(QPen(QColor(241, 194, 50, 220), 2.5, Qt.DashLine))
310             pts = self.robot.chemin_complet
311             for i in range(len(pts)-1):
312                 p.drawLine(QPointF(*pts[i]), QPointF(*pts[i+1]))
313
314             # Dessin du robot mobile
315             rx, ry = int(self.robot.x), int(self.robot.y)
316             p.setBrush(QColor(241, 196, 15))
317             p.setPen(QPen(Qt.black, 2))
318             p.drawRoundedRect(rx-12, ry-12, 24, 24, 4, 4)
319
320 # ==========================================
321 # PARTIE 6 : FENÊTRE PRINCIPALE
322 # ==========================================
323 class FenetrePrincipale(QWidget):
324     def __init__(self):
```

Algorithme génétique

```
    def __init__(self):
        super().__init__()
        self.biblio = Bibliotheque()
        self.robot = None
        self.livres_selectionnes = {m: [] for m in self.biblio.catalogue.keys()}

        self.setWindowTitle("Suivi Robot – Collecte Individualisée (6 Rayonnages)")
        self.resize(1100, 580)
        self.setStyleSheet("background-color: #11141a; color: white;")

        layout_global = QHBoxLayout(self)
        layout_global.setContentsMargins(15, 15, 15, 15)
        layout_global.setSpacing(15)

        self.zone_carte = ZoneCarte(self.biblio, self.livres_selectionnes)
        layout_global.addWidget(self.zone_carte, stretch=3)

        self.conteneur_tablette = QFrame()
        self.conteneur_tablette.setStyleSheet("QFrame { background-color: #1c212c; border: 2px solid #2c3e50; border-
radius: 12px; }")
        self.layout_tablette = QVBoxLayout(self.conteneur_tablette)
        self.layout_tablette.setContentsMargins(15, 15, 15, 15)

        self.label_titre = QLabel("TABLETTE TACTILE")
        self.label_titre.setAlignment(Qt.AlignCenter)
        self.label_titre.setStyleSheet("font-size: 18px; font-weight: bold; color: #ffb733; border:none;")
        self.layout_tablette.addWidget(self.label_titre)

        self.scroll_area = QScrollArea()
        self.scroll_area.setWidgetResizable(True)
        self.scroll_area.setStyleSheet("border: none; background: transparent;")
        self.widget_dynamique = QWidget()
        self.layout_dynamique = QVBoxLayout(self.widget_dynamique)
        self.layout_dynamique.setContentsMargins(0, 5, 0, 5)
        self.scroll_area.setWidget(self.widget_dynamique)
        self.layout_tablette.addWidget(self.scroll_area)

        self.btn_valider = QPushButton("🤖  LANCER LE ROBOT")
        self.btn_valider.setMinimumHeight(48)
        self.btn_valider.setStyleSheet("QPushButton { background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop:0
#ff9900, stop:1 #e67e22); color: white; font-weight: bold; font-size: 13px; border-radius: 6px; border:none; }
QPushButton:hover { background: #f39c12; } QPushButton:disabled { background-color: #34495e; color: #7f8c8d; }")
```

Algorithme génétique

```
363             self.btn_valider.clicked.connect(self.declencher_mission_robot)
364             self.layout_tablette.addWidget(self.btn_valider)
365
366             self.label_status = QLabel("Aucun livre sélectionné")
367             self.label_status.setAlignment(Qt.AlignCenter)
368             self.label_status.setStyleSheet("color: #7f8c8d; font-size: 11px; font-style: italic; border:none;")
369             self.layout_tablette.addWidget(self.label_status)
370
371             layout_global.addWidget(self.conteneur_tablette, stretch=2)
372
373             self.timer = QTimer()
374             self.timer.timeout.connect(self.animer_robot)
375             self.afficher_menu_principal()
376
377         def nettoyer_layout_dynamique(self):
378             while self.layout_dynamique.count():
379                 enfant = self.layout_dynamique.takeAt(0)
380                 if enfant.widget(): enfant.widget().deleteLater()
381
382         def afficher_menu_principal(self):
383             self.nettoyer_layout_dynamique()
384             self.label_titre.setText("📚 MENU PRINCIPAL")
385             lbl = QLabel("Choisissez une catégorie de livre :")
386             lbl.setStyleSheet("color: #b2bec3; font-size: 12px; border:none;")
387             self.layout_dynamique.addWidget(lbl)
388
389             for mat in self.biblio.catalogue.keys():
390                 btn = BoutonMenuPrincipal(mat)
391                 if self.livres_selectionnes[mat]:
392                     btn.setText(btn.text() + f"  ({len(self.livres_selectionnes[mat])} vol.)")
393                 btn.clicked.connect(lambda checked, m=mat: self.afficher_sous_menu_matiere(m))
394                 self.layout_dynamique.addWidget(btn)
395
396             self.layout_dynamique.addStretch()
397             self.rafraichir_statut_global()
398             self.zone_carte.update()
399
400         def afficher_sous_menu_matiere(self, matiere):
401             self.nettoyer_layout_dynamique()
402             self.label_titre.setText(f"📂 {matiere.upper()}")
403
404             btn_retour = QPushButton("⬅ Confirmer et Retour")
```

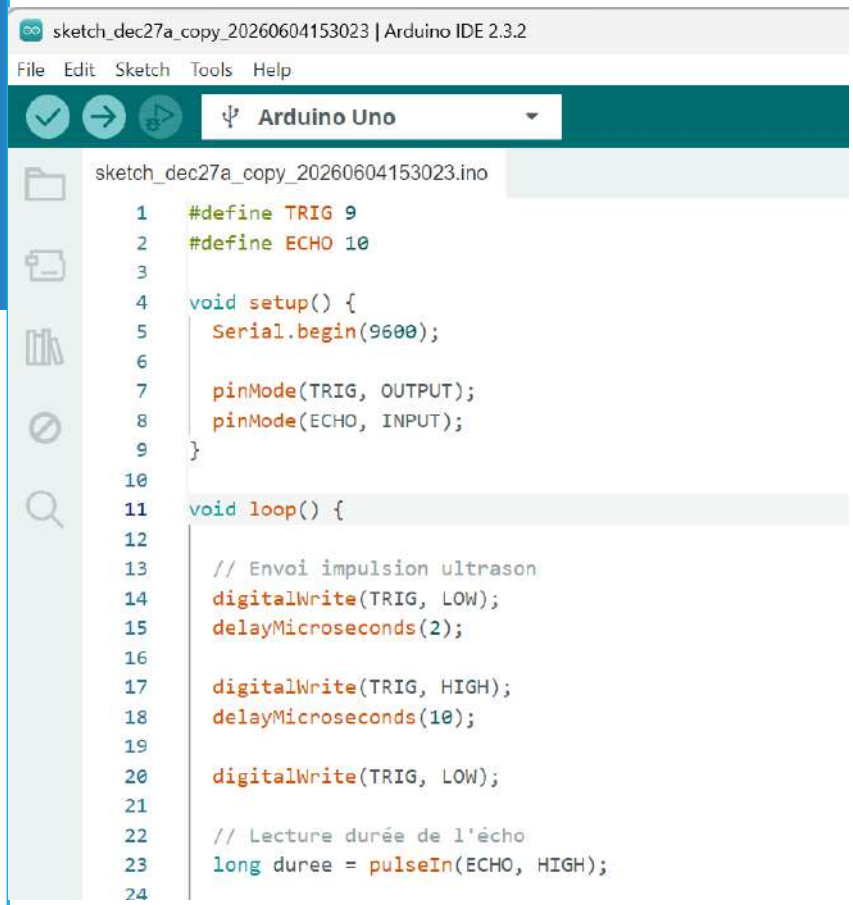
Algorithme génétique

```
405         btn_retour.setMinimumHeight(35)
406         btn_retour.setStyleSheet("QPushButton { background-color: #34495e; color: white; font-weight: bold; border-
    radius: 4px; border: none; font-size: 11px; } QPushButton:hover { background-color: #415b76; }")
407         btn_retour.clicked.connect(self.afficher_menu_principal)
408         self.layout_dynamique.addWidget(btn_retour)
409
410         grille_livres = QGridLayout()
411         grille_livres.setSpacing(8)
412         liste_titres = self.biblio.catalogue[matiere]
413         couleur_bouton = BoutonMenuPrincipal.THEMES.get(matiere, ("#2c3e50", ""))[0]
414
415         for idx, titre in enumerate(liste_titres):
416             btn_livre = BoutonLivreSelection(titre, couleur_base=couleur_bouton)
417             if titre in self.livres_selectionnes[matiere]: btn_livre.setChecked(True)
418             btn_livre.toggled.connect(lambda checked, t=titre, m=matiere: self.gerer_panier_livre(m, t, checked))
419             row, col = divmod(idx, 2)
420             grille_livres.addWidget(btn_livre, row, col)
421
422         self.layout_dynamique.addLayout(grille_livres)
423         self.layout_dynamique.addStretch()
424
425     def gerer_panier_livre(self, matiere, titre, coché):
426         if coché and (titre not in self.livres_selectionnes[matiere]):
427             self.livres_selectionnes[matiere].append(titre)
428         elif not coché and (titre in self.livres_selectionnes[matiere]):
429             self.livres_selectionnes[matiere].remove(titre)
430         self.rafraichir_statut_global()
431         self.zone_carte.update()
432
433     def rafraichir_statut_global(self):
434         total = sum(len(v) for v in self.livres_selectionnes.values())
435         if total > 0:
436             self.label_status.setText(f"🛒 Panier : {total} volume(s) à aller collecter.")
437             self.label_status.setStyleSheet("color: #2ecc71; font-size: 11px; font-weight: bold; border:none;")
438             self.btn_valider.setEnabled(True)
439         else:
440             self.label_status.setText("Aucun livre sélectionné")
441             self.label_status.setStyleSheet("color: #7f8c8d; font-size: 11px; font-style: italic; border:none;")
442             self.btn_valider.setEnabled(False)
443
444     def declencher_mission_robot(self):
445         self.biblio.réinitialiser_points()
```

Algorithme génétique

```
440             self.label_status.setText("Aucun livre sélectionné")
441             self.label_status.setStyleSheet("color: #7f8c8d; font-size: 11px; font-style: italic; border:none;")
442             self.btn_valider.setEnabled(False)
443
444     def declencher_mission_robot(self):
445         self.biblio.réinitialiser_points()
446
447         # Prise en compte de tous les livres de Technologie choisis en plus des autres
448         for matiere, liste_livres in self.livres_selectionnes.items():
449             for titre in liste_livres:
450                 coord_physique_livre = self.biblio.coords_livres_physiques[titre]
451                 self.biblio.ajouter_objective_livre_ou_passage(coord_physique_livre)
452
453         self.robot = RobotOptimise(self.biblio)
454         self.zone_carte.robot = self.robot
455
456         self.conteneur_tablette.setEnabled(False)
457         self.btn_valider.setText("⏳ TRAJET EN COURS...")
458         self.timer.start(35)
459
460     def animer_robot(self):
461         if self.robot and self.robot.avancer():
462             self.timer.stop()
463             self.conteneur_tablette.setEnabled(True)
464             self.btn_valider.setText("🤖  LANCER LE ROBOT")
465             self.livres_selectionnes = {m: [] for m in self.biblio.catalogue.keys()}
466             self.afficher_menu_principal()
467             self.label_status.setText("✅ Mission accomplie ! Tous les ouvrages ciblés sont récupérés.")
468             self.label_status.setStyleSheet("color: #f1c40f; font-size: 12px; font-weight: bold; border:none;")
469         self.zone_carte.update()
470
471 def _secure_inject(cls):
472     cls.ajouter_objective_livre_ou_passage = cls.ajouter_objectif_livre
473 _secure_inject(Bibliotheque)
474
475 if __name__ == "__main__":
476     app = QApplication(sys.argv)
477     app.setStyle("Fusion")
478     fenetre = FenetrePrincipale()
479     fenetre.show()
480     sys.exit(app.exec_())
```

Mesure de distance a un obstacle




```
#define TRIG 9
#define ECHO 10

void setup() {
  Serial.begin(9600);

  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
}

void loop() {

  // Envoi impulsion ultrason
  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);

  digitalWrite(TRIG, LOW);

  // Lecture durée de l'écho
  long duree = pulseIn(ECHO, HIGH);

  // Calcul distance en cm
  float distance = duree * 0.0343 / 2.0;

  Serial.println(distance);

  delay(500);
}
```

Mesure de la vitesse du moteur

```
// ============================
// MCC + encodeur quadrature
// ============================

const int PIN_DIR = 7;
const int PIN_PWM = 9;

const int PIN_SA = 2;
const int PIN_SB = 3;

volatile long compteur = 0;

const float PPR = 12.0;
const float REDUCTION = 52.734;

unsigned long t_precedent = 0;
const unsigned long Te_ms = 50;

void interruptionA()
{
  if (digitalRead(PIN_SA) == digitalRead(PIN_SB))
    compteur++;
  else
    compteur--;
}

void interruptionB()
{
  if (digitalRead(PIN_SA) != digitalRead(PIN_SB))
    compteur++;
  else
    compteur--;
}

void setup()
{
  Serial.begin(115200);

  pinMode(PIN_DIR, OUTPUT);
  pinMode(PIN_PWM, OUTPUT);

  pinMode(PIN_SA, INPUT);
  pinMode(PIN_SB, INPUT);
```

```
45    attachInterrupt(digitalPinToInterrupt(PIN_SA), interruptionA, CHANGE);
46    attachInterrupt(digitalPinToInterrupt(PIN_SB), interruptionB, CHANGE);
47
48    digitalWrite(PIN_DIR, LOW);
49    analogWrite(PIN_PWM, 255);
50
51    t_precedent = millis();
52  }
53
54  void loop()
55  {
56    unsigned long t_now = millis();
57
58    if (t_now - t_precedent >= Te_ms)
59    {
60      noInterrupts();
61      long impulsions = compteur;
62      compteur = 0;
63      interrupts();
64
65      float Te_s = Te_ms / 1000.0;
66
67      float tours_moteur = impulsions / PPR;
68
69      float rpm_moteur = (tours_moteur / Te_s) * 60.0;
70
71      float rpm_sortie = rpm_moteur / REDUCTION;
72
73
74      Serial.print(1);
75      Serial.print(" ");
76      Serial.println(rpm_sortie*52.734);
77
78      t_precedent = t_now;
79    }
80  }
```