Question 4)

Pour implémenter un réseau de neurones pour l'évitement d'obstacles selon la méthode de *Braitenberg* dans le cas de l'Alphabot2, qui possède des capteurs de proximité à réponse booléenne (tout ou rien), on peut suivre les étapes suivantes :

1. Structure du Réseau : Tout d'abord on va créer un réseau de neurones simple avec une couche d'entrée, une couche cachée (si nécessaire pour la complexité), et une couche de sortie. Les neurones d'entrée correspondent aux capteurs de proximité. La couche de sortie aura deux neurones, un pour la vitesse de chaque moteur.

2. Entrées Booléennes : Les entrées du réseau sont les états booléens des capteurs de proximité. Une valeur de 1 indique qu'un obstacle est détecté et 0 qu'il n'y a pas d'obstacle.

3. Poids Synaptiques : On va initialiser les poids synaptiques pour refléter le comportement désiré. Par exemple, un capteur détectant un obstacle pourrait inhiber la vitesse du moteur sur le même côté pour tourner à l'opposé de l'obstacle.

4. Fonction d'Activation : On va utiliser une fonction d'activation appropriée pour la couche de sortie qui permet de traduire l'activation du réseau en vitesse de moteur. Pour un comportement tout ou rien, une fonction de type step peut être appropriée.

5. Logique de Braitenberg : L'avant dernière étape concerne l'implémentation de la logique de Braitenberg où la présence d'un obstacle (entrée à 1) cause une réaction des moteurs qui permet l'évitement. Ceci peut être directement codé dans les poids synaptiques. Par exemple, si le capteur gauche détecte un obstacle, le poids entre ce capteur et le moteur gauche pourrait être négatif (causant un ralentissement ou arrêt du moteur gauche), tandis que le poids vers le moteur droit pourrait être positif (accélérant le moteur droit et causant une rotation à gauche).

6. Apprentissage : Si nécessaire, on peut entrainer le réseau de neurones avec des exemples de comportements d'évitement d'obstacles pour affiner les poids synaptiques.

Voici un pseudo code général qui pourrait représenter cette implémentation :

Cf : **alphaBot_Braitenberg.c**

Question 5)

On peut envisager un ensemble d'états qui représentent les différentes réponses du robot aux conditions de son environnement.

Veuillez trouver pseudo-code détaillé dans le fichier :

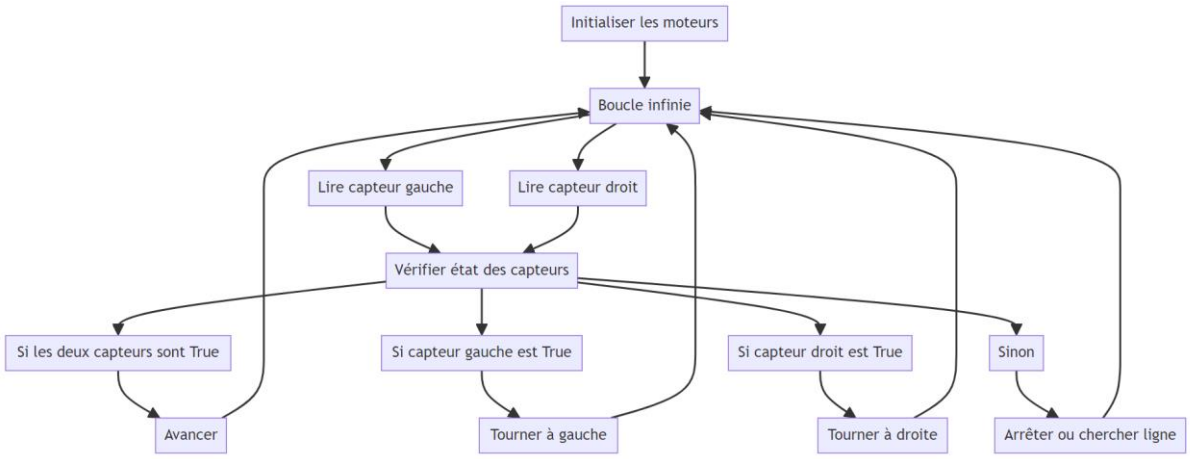
Cf : **pseudo_code.txt**

Dans ce pseudo-code :

- Le robot commence par avancer (AVANCER) et vérifie périodiquement son environnement (VERIFIER).
- La détection d'obstacles déclenche des actions de contournement en tournant à gauche (TOURNER_GAUCHE) ou à droite (TOURNER_DROITE), ou en reculant (RECULER) si nécessaire.
- Les fonctions correspondant à chaque action doivent être définies et intégrées dans le contrôleur du robot.La durée et l'angle de rotation pour TOURNER_GAUCHE et TOURNER_DROITE doivent être déterminés en fonction de la géométrie et de la cinématique du robot.
- La commande d'arrêt (STOP) peut être activée par une condition externe, comme un signal de l'utilisateur ou une condition de sécurité.
- La transition de l'état STOP vers l'état AVANCER nécessite une commande explicite de redémarrage.

Question 7)

Pour implémenter un suivi de ligne à l'aide de capteurs de proximité qui renvoient une information booléenne (détecteur d'obstacle), nous devons créer un algorithme simple qui réagit en fonction de la présence ou de l'absence d'une ligne sous chaque capteur.



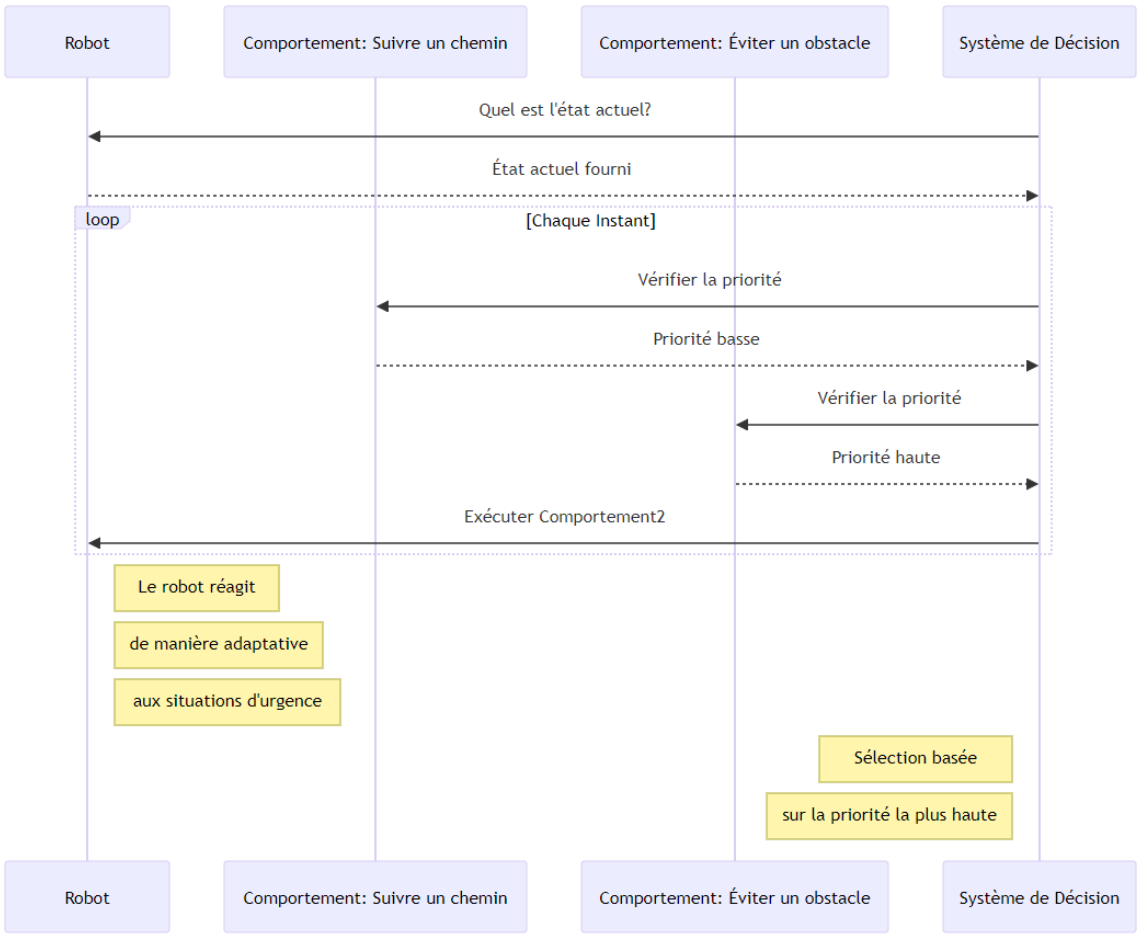
Dans une boucle continue, on lit l'état de chaque capteur : un capteur renvoie vrai (True) s'il détecte une ligne, sinon il renvoie faux (False). La logique de décision commande les moteurs en fonction des lectures des capteurs : si les deux détectent la ligne, l'AlphaBot2 avance droit; si un seul capteur détecte la ligne, l'AlphaBot2 tourne légèrement du côté du capteur activé; et si aucun capteur ne détecte la ligne, l'AlphaBot2 s'arrête ou effectue une action prédéfinie, comme tourner sur place pour chercher la ligne. Ce processus se répète à une fréquence élevée pour assurer un suivi fluide de la ligne.

Pour adopter des techniques de suivi de ligne plus sophistiquées telles que l'utilisation d'un contrôleur PID, qui offre une réponse dynamique et ajustée aux variations de trajectoire, il est souvent nécessaire d'ajouter ou d'utiliser des capteurs plus avancés. Le PID, qui signifie Proportional-Integral-Derivative, est une méthode de contrôle en boucle fermée où l'action est ajustée en fonction de l'erreur actuelle (proportionnelle), de l'historique des erreurs (intégrale) et de la vitesse de changement de l'erreur (dérivée). Pour que le PID fonctionne efficacement dans le suivi de ligne, le système a besoin de données précises sur la position de la ligne. Cela peut être réalisé à l'aide de capteurs analogiques qui mesurent des degrés de déviation plutôt que de simples états binaires, ou par un réseau de capteurs numériques qui fournissent une résolution spatiale plus fine.

Question 8)

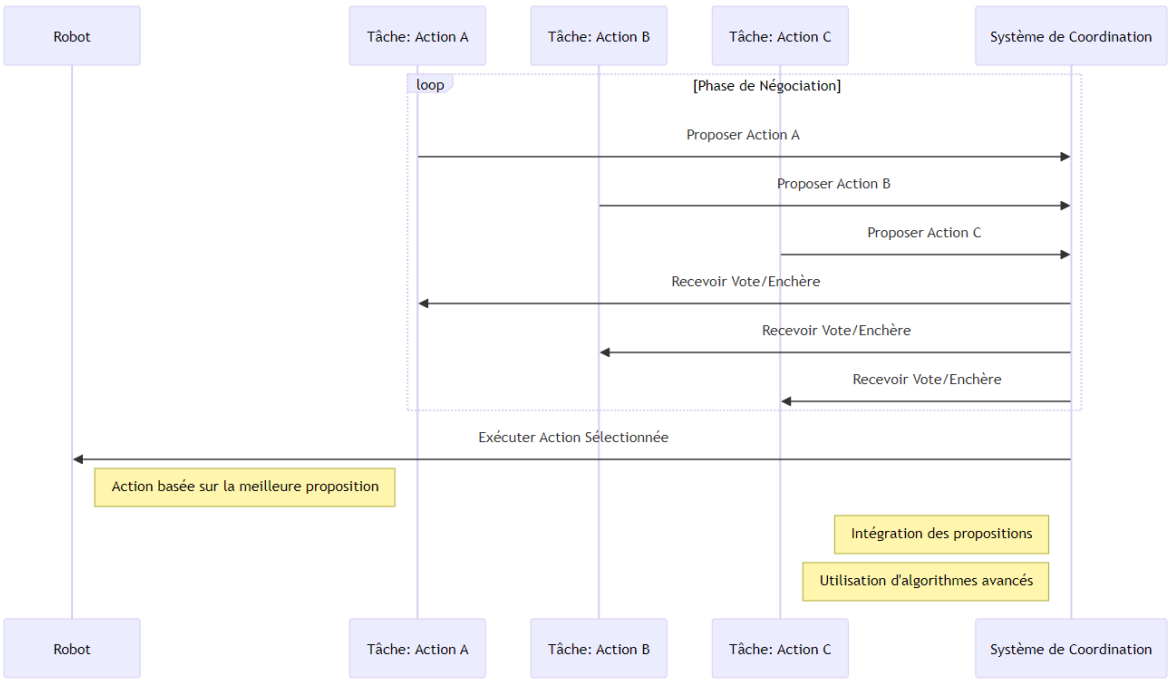
1ère stratégie :

Coordination par priorisation : Chaque comportement que le robot peut exhiber est associé à une priorité. La coordination se fait en sélectionnant à chaque instant le comportement ayant la plus haute priorité pour déterminer l'action à exécuter. Par exemple, éviter un obstacle pourrait avoir une priorité plus élevée que suivre un chemin prédéterminé. Cela garantit que le robot peut réagir de manière adaptative aux situations d'urgence tout en poursuivant ses objectifs généraux.



2eme stratégie :

Coordination par négociation : Dans cette approche, chaque tâche ou comportement proposerait une action et, au lieu de choisir en fonction de priorités fixes, une phase de négociation aurait lieu où les comportements pourraient "voter" ou "enchérir" pour leur action proposée. La coordination déterminerait l'action finale en intégrant les propositions de toutes les tâches, peut-être en utilisant des techniques comme les algorithmes génétiques ou les réseaux bayésiens pour estimer la meilleure action globale qui satisfait le plus grand nombre de comportements.


Robot
Tâche: Action A
Tâche: Action B
Tâche: Action C
Système de Coordination
loop
[Phase de Négociation]
Proposer Action A
Proposer Action B
Proposer Action C
Recevoir Vote/Enchère
Recevoir Vote/Enchère
Recevoir Vote/Enchère
Exécuter Action Sélectionnée
Action basée sur la meilleure proposition
Intégration des propositions
Utilisation d'algorithmes avancés
Robot
Tâche: Action A
Tâche: Action B
Tâche: Action C
Système de Coordination