Projet Robot LEGO : Cahier des charges Cours Modélisation Réaliste et Implantation Multi-tâche

Thao Dang et Pascal Raymond

1 Modélisation réaliste et commande robuste

1. **Problème 1.a** - Modéliser les sources possibles dimprécision et perturbations dans le robot et des contrôleurs en pratique. Par simulation, étudier leur influence sur la stabilité du système global.
2. **Problème 1.b** - Etudier la robustesse des contrôleur d'angle en utilisant des critère de stabilité pour des systèmes incertains et avec retards. Justifier le choix des paramètres : la période T déchantillonnage, ki_θ (de l'action proportionielle) et kp_θ (de l'action intégrale) des contrôleurs (voir TD1).
3. **Problème 1.c** - Le délai τ dans la réaction moteur est introduit dans la fonction transfert. Etudier la stabilité du système en fonction de τ (voir TD2).

2 Commande multi-objective - suiveur de ligne et anti-collision

Problème 2 - Créer en Simulink un système de commande de robots LEGO qui réalise les 2 fonctions suivantes :

- **Régulateur de trajectoires** : Cette fonction assure que le robot suit une trajectoire désirée. Elle envoie périodiquement des valeurs de vitesse des deux roues aux moteurs. Ces valeurs sont calculées par deux contrôleurs (de distance et d'angle) en fonction des informations sur la position et l'orientation x, y, θ courantes du robot et celles de consigne ϵ_d et ϵ_θ.
- **Planification et anti-collision** : pour éviter des obstacles, quand le robot rencontre un obstacle, le planificateur génère une commande de changement de direction et/ou de vitesse (par exemple, arrêt, marche arrière, avancement en vitesse réduite) quand le robot rencontre un obstacle. Le planificateur détermine d'une manière continue les valeurs de consigne ϵ_d et ϵ_θ et puis communique ces valeurs aux contrôleurs de trajectoires ci-dessus.

  Une **stratégie d'anti-collision** possible est la suivante. D'abord, le robot s'arrête (le robot prend un certain temps pour s'arrêter complètement). Ensuite, le robot fait un demi-tour sur place. Supposons que le robot tourne vers le capteur à gauche, pour détecter le moment où le robot termine le demi-tour, on peut détecter une séquence de valeurs "Blanc-Noir-Blanc" dans la sortie du capteur à gauche.

Remarques

- Pour détecter une séquence de valeurs "Blanc-Noir-Blanc", il faut mémoriser, par exemple, l'événement qu'un capteur voit la couleur "Blanc". Pour ceci, on peut ulitiser un bloc "Logical Or" dont une entrée est connectée à la sortie d'un comparateur (voir la Figure 1 où N_b est le seuil de la couleur blanche).
- Après terminer le demi-tour, le contrôleur doir revenir dans l'état avant le mode d'anti-collision, il est utile de pouvoir "réinitialiser" les variables importantes du contrôleur. Pour ceci, voir la Figure 2 pour un exemple de bloc "mémoire avec reset".

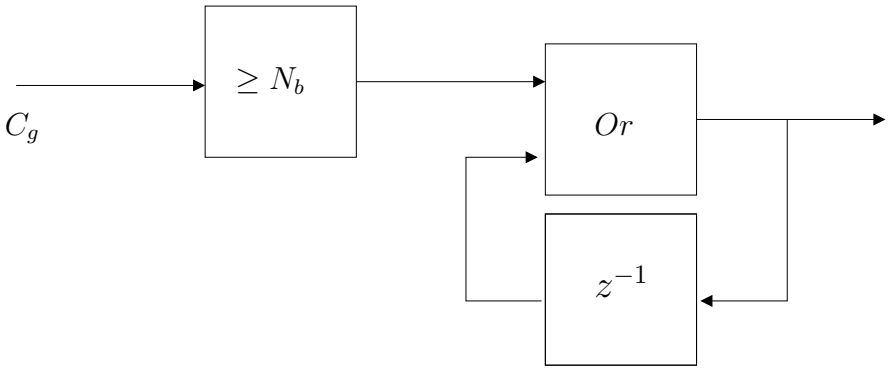

FIGURE 1 – Mémorisation de l'événement de détection de couleur blanche.

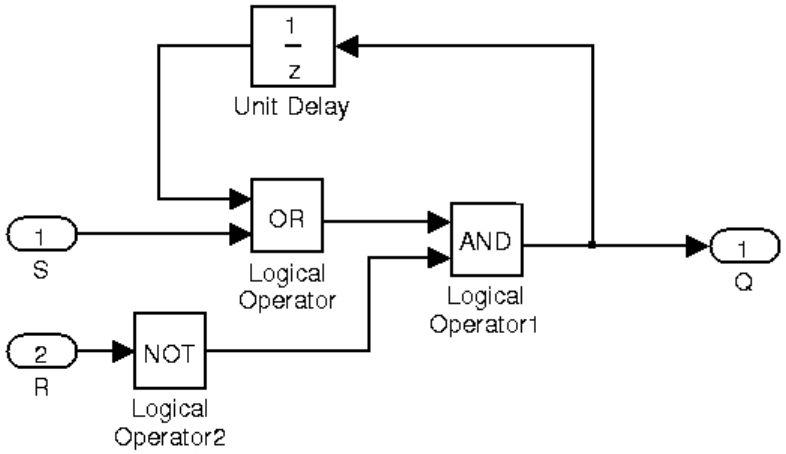

FIGURE 2 – Mémoire avec reset.

3 Implantation multi-tâche

Problème 3 - Utiliser la chaîne **mdl2lus2osek** pour générer le code du contrôleur et léxecuter sur les robots en utilisant deux méthodes : mono-tâche et multi-tâche.

1. Pour l'implantation multi-tâche, il faut extraire 2 modules correspondant à 2 tâches, en créant deux modèles Simulink séparés : par exemple le régulateur de trajectoire et le planificateur, puis générer le code pour chaque modèle. Expliquer la séparation en 2 modules - suivant quels critères (fréquences des tâches, priorités) ?
2. Faire des expériments avec le robot et ajuster les contrôleurs et le planificateur, si besoin en modifiant les modèles SIMULINK. Expliquer la procédure.