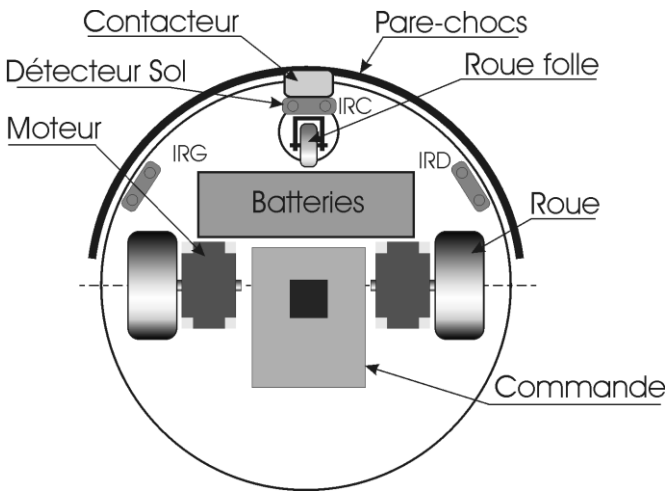
AG43
Examen Final A2013
Durée 1h30
Aucun document autorisé

Robot Aspirateur

Schéma de la partie motrice



Fonctionnement

Le Robot se déplace grâce à deux roues motrices indépendantes. Celles-ci sont placées selon un diamètre de la plateforme circulaire. Cette disposition permet un mouvement de rotation sur place du robot. Les mouvements se font à vitesse constante. Ceci signifie que l'on ne cherche pas à optimiser la vitesse. Il n'y a donc pas de phase d'accélération à gérer. Le moteur s'arrête et se met en mouvement instantanément. Un réducteur diminue les effets d'inertie du robot complet.

Les mouvements possibles du robot sont :

- Avance : le moteur droit tourne dans le sens horaire, le moteur gauche dans le sens trigo
- Recul : le moteur droit tourne dans le sens trigo, et le gauche dans le sens horaire
- Pivotement sur place à gauche, les 2 moteurs tournent dans le sens horaire
- Pivotement sur place à droite, les 2 moteurs tournent dans le sens trigo

L'exploration de la pièce se fait uniquement à l'aide de ces fonctions. On dispose de capteurs détectant un obstacle ou un vide (sommet d'une marche). Les capteurs d'obstacle sont au nombre de 2. On ne prendra pas en compte, dans ce travail, les capteurs de distance infrarouge (détection des seuils de marche). Les capteurs d'obstacle seront donc ContactG et ContactD, qui seront considérés comme des variables entières globales, de valeur 1 si un obstacle est détecté. Lorsque le contact avec l'obstacle est frontal, les deux capteurs seront sollicités.

On décide de faire fonctionner les moteurs à une cadence de 20 pas par seconde. La façon la plus simple de gérer le fonctionnement des moteurs est de le faire dans une **procédure de type**

interruption appelée `InterGestion()`. Cette procédure est appelée automatiquement 20 fois par seconde.

On dispose du tableau `etat[]` de 4 octets, décrivant les 4 états successifs d'alimentation des phases du moteur. Le tableau est indicé par rapport à 0. *Lorsqu'un moteur se voit appliquer successivement les valeurs `etat[i]` du tableau, avec l'indice i croissant, il tourne dans le sens* ***horaire****. En décrivant le tableau avec un indice i décroissant, il tourne dans le sens trigonométrique*.

Mise en mouvement du robot

Le robot est mis en mouvement par la procédure de type interruption appelée `InterGestion()` décrite ci-dessous :

```
Procédure InterGestion() de type interruption

Variables globales :
etat[] : Tableau de 4 octets décrivant les états successifs d'alimentation
des phases des moteurs
moteurG, moteurD : octets, ports moteurs
iG, iD : Entiers, indices de parcours du tableau etat[] pour les moteurs
Gauche et Droite
incG, incD : entiers, incréments, 0,1,-1
Compteur : entier

      Si Compteur > 0 alors
            Compteur = Compteur - 1
      Fin Si
      iG = (iG + incG) % 4
      iD = (iD + incD) % 4
      moteurG = etat[iG]
      moteurD = etat[iD]

Fin procédure
```

Question 1 : Répondre aux questions suivantes, par une seule phrase à chaque fois :

1. Pourquoi n'utilise-t-on pas d'argument ?
2. Quel est l'intérêt de l'usage de variables globales ?
3. Dans quel cas un moteur est-il mis en marche ?
4. Quel est l'intérêt de l'opération modulo dans l'instruction `iG = (iG + incG)% 4` ?
5. Quel est la relation entre l'évolution de compteur lorsqu'il est non nul et la rotation des moteurs ?
6. Quel est le temps nécessaire au compteur, initialisé à 1000, pour arriver à une valeur nulle ?

Question 2 : Procédure faisant reculer le robot de 50 pas

De par sa conception, ce robot ne doit qu'avancer, les capteurs de seuils et d'obstacles étant situés à l'avant. La rencontre d'un obstacle doit provoquer l'arrêt, puis le recul du robot de 50 pas, ce qui est tolérable, puisqu'il est supposé revenir sur ses pas.

On vous propose d'écrire la procédure `Recule()` qui se contente de faire reculer le robot de 50 pas et de l'immobiliser (utiliser la variable compteur). Ne pas oublier que les moteurs sont activés automatiquement par l'interruption `InterGestion()`.

Question 3 : Rotation à gauche du robot

Ecrire la procédure `PivoteG(n)`, réalisant un pivotement du robot de n pas à gauche et arrêtant ensuite la rotation des moteurs. Ne pas oublier de déclarer les variables.

Question 4 : Gestion des contacts fin de course

En cas de contact frontal, les contacts fin de course gauche et droite ne sont pas actionnés simultanément, mais l'un des deux commutera en premier. En cas de contact latéral, un seul commutera. Afin de s'assurer du type de contact, après la détection du changement d'état d'un fin de course, on laisse encore les moteurs effectuer deux pas avant de les arrêter. Noter que les fin de course ne peuvent être sollicités que lorsque le robot avance. Ecrire une fonction appelée Obstacle() qui fait se contente de faire avancer encore de deux pas les moteurs, stoppe les moteurs et retourne les valeurs suivantes :

- 0 si aucun obstacle est détecté
- 1 si ContactG est actionné
- 2 si ContactD est actionné
- 3 si les deux contacts sont actionnés

Remarque : Cette fonction ne sera appelée que si l'on détecte l'activation d'un contact fin de course.

Question 5 : Déplacement avec évitement des obstacles

Ecrire une boucle de type Tant Que dans laquelle on placera l'algorithme de gestion du déplacement. On dispose des procédures et fonctions vues précédemment, notamment PivoteG(), PivoteD(), Recule() et Obstacle().

La gestion du déplacement est prévue comme suit :

- Le robot avance en ligne droite jusqu'à la rencontre d'un obstacle (ContactG et/ou ContactD passe de 0 à 1)
- Le robot recule de 50 pas (procédure `Recule()`)
- Le robot pivote de 30° vers la gauche et repart en marche avant si ContactD est activé, et 30° vers la droite dans le cas de l'activation de ContactG
- Le robot pivote de 90° vers la gauche si les deux contacts sont activés
- On quitte cette boucle après environ 30 mn de fonctionnement. Réfléchir à une manière d'utiliser la variable `compteur`, sachant qu'elle est probablement également utilisée par les procédures PivoteG(n), PivoteD(n) et Recule().

On dispose des procédures étudiées précédemment ainsi que des variables globales et constantes prédéfinies vues précédemment. On utilisera en argument pour les fonctions PivoteG() ou pivoteD() les constantes prédéfinies (inutile de les déclarer) `N5`, `N10`, `N30`, `N45`, `N90` qui sont les valeurs nécessaires pour le pivotement d'un angle de 5, 10, 30, 45, 90°. D'autres valeurs d'angle peuvent être obtenues en additionnant ces constantes.

On demande d'écrire la boucle de gestion, et il n'est pas demandé de déclarer les variables globales utilisées.