Series 3 – Ground Sensor

Introduction

Dans cette série, on s'intéresse aux capteurs de sols présents sur le robot. Ils sont au nombre de trois et donnent chacun un indice de luminosité. Grâce à ces données recueillies, il est possible de développer des contrôleurs capables de faire interagir le robot avec des inscriptions sur le sol. L'objectif de cette série consiste notamment à mettre au point un contrôleur de sorte à ce que le robot suive une ligne noire.

Pour effectuer ces exercices, seuls les capteurs de sols seront utilisés pour fournir des données d'entrée à interpréter. Le comportement en sortie s'effectue par l'ajustement des vitesses des deux moteurs.

Développement

Lors de la série précédente, une petite bibliothèque de fonctions utilitaires a été développée pour servir de couche d'abstraction sur l'API du robot. Elle a été légèrement modifiée et possède désormais également des fonctions pour manipuler les capteurs de sols.

L'ensemble de ces fichiers se trouve toujours dans le dossier `util`, ils sont compilés via le *Makefile* présent dans ce même dossier. Puisque le *Makefile* de chaque contrôleur ne sélectionne pas individuellement les fichiers de cette bibliothèque à compiler, cela implique que chaque contrôleur possèdera l'ensemble des fonctions dans son exécutable. Cela n'est pas vraiment nécessaire car les LEDs et les capteurs de proximité ne sont pas utilisés dans cette série, mais comme cette bibliothèque est très petite, l'impact est suffisamment négligeable.

1 Ground sensor respons

1.1 Mise en place

L'objectif de cet exercice est de produire une représentation visuelle des valeurs des capteurs de sol lorsque le robot traverse une ligne noire sur un sol blanc. Contrairement à la série précédente, il n'y a pas de phase de calibration car on se contente simplement de distinguer le blanc du noir sans état intermédiaire pour le moment.

Le contrôleur correspondant à cet exercice est `S03_Ground_measurement`. Le code source de celui-là se trouve dans le fichier `S03_Ground_measurement.c`. Il s'agit essentiellement du même code source que celui développé lors de la série 2 pour l'exercice équivalent : *Proximity Sensor Values*.

1.2 Exécution

Le robot utilisé pour cette expérience est le modèle n° 3478. Les valeurs des capteurs sont enregistrées sur 50 *steps* à une vitesse de 2 rad/s. L'expérience s'est déroulée une première fois en traversant la ligne perpendiculairement (cf. figure 1), et une seconde fois en traversant la ligne en diagonale avec un angle d'environ 45° (cf. figure 2).

1.3 Observation

Les graphes générés (figures 1 et 2) sont assez fidèles à ce que l'on pouvait s'attendre et ne montrent rien de particulièrement surprenant. Les valeurs des capteurs se stabilisent vers 950 lorsqu'ils détectent du blanc, et vers 300 lorsqu'ils détectent du noir.

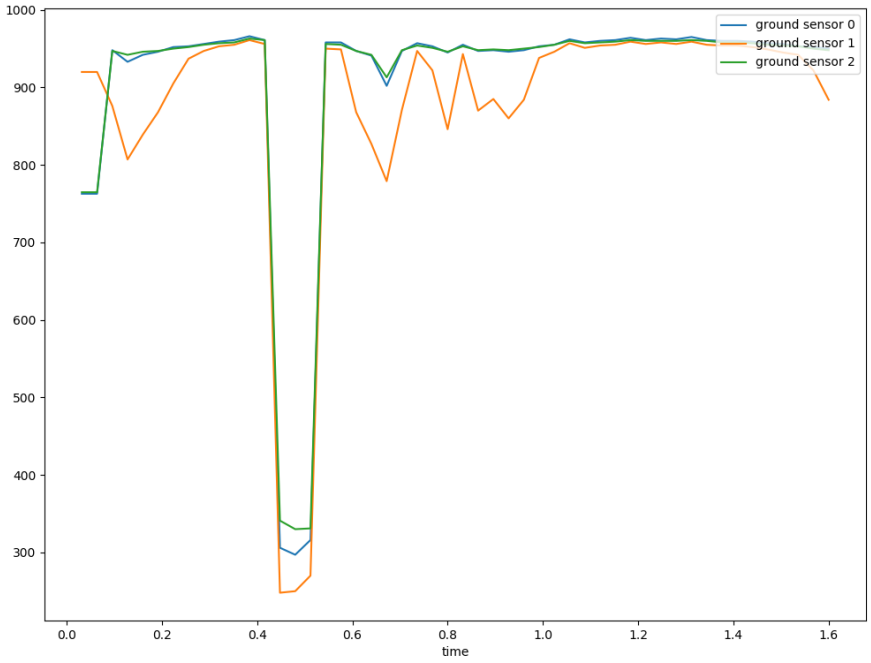

FIGURE 1 – Valeurs des capteurs lorsque le robot traverse une ligne noire perpendiculairement

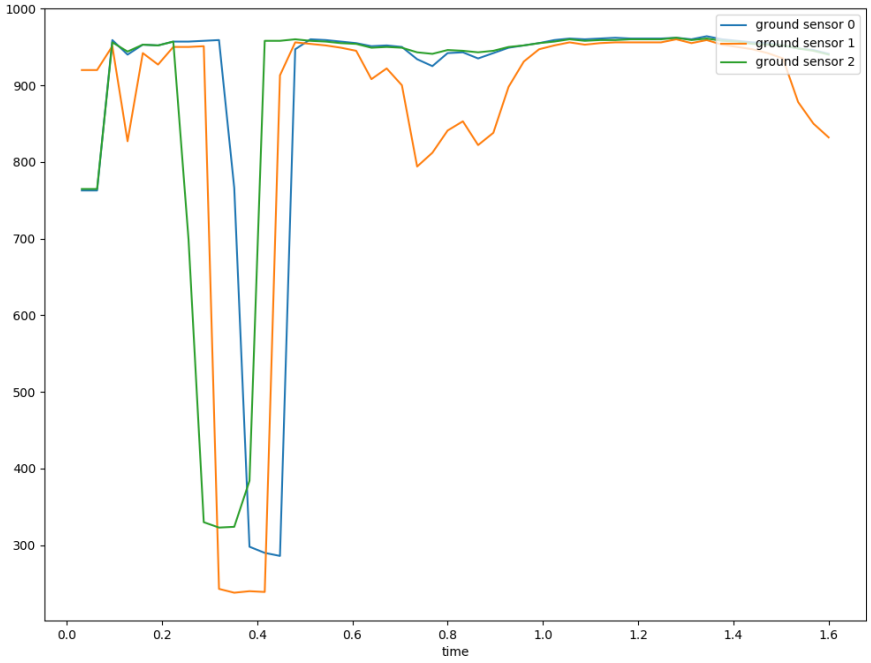

FIGURE 2 – Valeurs des capteurs lorsque le robot traverse une ligne noire en diagonale

On notera que le capteur n° 1 (celui au centre) de ce modèle est plus sensible à un assombrissement que les deux autres, surtout lorsque le robot se trouve sur une surface claire : on voit en effet que lorsque les capteurs n° 0 et n° 3 chutent vers une valeur d'environ 900, le capteur n° 1 chute vers une valeur au-dessous de 800.

2 Line following

2.1 Mise en place

L'objectif de cet exercice est de développer un contrôleur qui permet au robot de se déplacer de manière autonome en suivant une ligne noire sur un sol blanc.

La méthode utilisée pour y parvenir est proche de celles développées pour les comportements « explorer » et « lover » de la série 2 : à chaque *step*, les valeurs des capteurs sont recueillies et influencent la vitesse des moteurs avec un certain poids. Cette fois, chaque capteur est susceptible d'influencer les deux moteurs (plutôt qu'une moitié des capteurs par moteur).

Trois types d'influence des moteurs ont été testés : soit on ralentit le moteur du côté où le capteur détecte du noir, soit on accélère le moteur du côté opposé, soit on combine les deux. Par expérience, la solution la plus robuste semble être la seconde : en effet, celle-ci implique que le robot sera plus lent lorsqu'il capte moins de noir, ce qui semble le coller à la ligne de manière plus efficace que la première solution où le robot est plus rapide lorsqu'il capte moins de noir. Quant à la dernière solution, le robot semble faire des mouvements trop brusques et ne parvient à bien suivre la ligne.

La boucle principale a donc été implémentée comme suit.

Listing 1 – Extrait de la fonction *main*

```
29 // weights { left, center, right }
30 static const double right_weights[GROUND_COUNT] = {0, 0, 1};
31 static const double left_weights[GROUND_COUNT] = {1, 0, 0};
32
33 while(wb_robot_step(TIME_STEP) != -1)
34 {
35     double ground_right = GROUND_MAX - ground_get_value(GROUND_RIGHT);
36     double ground_center = GROUND_MAX - ground_get_value(GROUND_CENTER);
37     double ground_left = GROUND_MAX - ground_get_value(GROUND_LEFT);
38
39     double right_factor =
40             ground_right * right_weights[GROUND_RIGHT] +
41             ground_center * right_weights[GROUND_CENTER] +
42             ground_left * right_weights[GROUND_LEFT];
43     double left_factor =
44             ground_right * left_weights[GROUND_RIGHT] +
45             ground_center * left_weights[GROUND_CENTER] +
46             ground_left * left_weights[GROUND_LEFT];
47
48     right_factor /= array_sum(right_weights, GROUND_COUNT);
49     left_factor /= array_sum(left_weights, GROUND_COUNT);
50
51     double speed_right = SPEED + left_factor / THRESHOLD * SPEED;
52     double speed_left = SPEED + right_factor / THRESHOLD * SPEED;
53
54     motors_set_speed(speed_left, speed_right);
55 }
```

On peut mettre en évidence ici le fait que le poids est toujours nul pour le capteur central (il n'influence rien), et le poids opposé au virage est également nul. Comme il n'y qu'un seul poids non nul par moteur dans cet algorithme, modifier la constante `THRESHOLD` est suffisant pour faire des ajustements.

On notera également qu'un virage à gauche fait accélérer la roue droite (`left_factor` influence `speed_right`) et vice versa. Il s'agit du second type d'influence des moteurs cité audessus.

Pour obtenir le meilleurs résultat, la vitesse `SPEED` a été fixée à 1 rad/s et `THRESHOLD` a été fixé à 250. `GROUND_MAX` correspond à la valeur maximale d'un capteur de sol, soit 1000.

Le contrôleur correspondant à cet exercice est `S03_Line_follower`. Le code source complet de celui-là se trouve dans le fichier `S03_Line_follower.c`.

2.2 Exécution

En simulation, le robot est capable de suivre la ligne de l'octogone de manière assez précise, en roulant très droit. Avec un robot réel (modèle n° 3478), le robot est également capable de suivre l'octogone, mais a tendance à pivoter légèrement vers la gauche et la droite répétitivement afin de continuellement réajuster sa position. Dans les deux cas, le robot semble être capable de pouvoir faire des tours de l'octogone de manière infinie.

Lorsqu'il s'agit de suivre la ligne du rectangle, le robot n'y parvient pas du tout. En effet, les virages de 90° sont beaucoup trop brusques pour l'algorithme élaboré : le robot ne capte le noir du virage que pendant très peu de temps et n'arrive pas à ajuster sa position suffisamment vite. Après plusieurs essais, il n'a pas été possible de déterminer s'il suffisait de modifier les constantes de l'algorithme actuel pour parvenir à suivre le rectangle. À priori, implémenter un algorithme différent serait certainement plus efficace.

2.3 Vidéo

Une vidéo du robot en exécution est disponible à l'adresse suivante.

https://youtu.be/l2RiF0wXHws

On peut voir le robot suivre l'octogone avec succès durant les 40 premières secondes, et échouer à suivre le rectangle durant les 20 dernières secondes.

3 Modular code

Comme le code développé jusqu'à présent est déjà modularisé, aucun contrôleur supplémentaire n'a été élaboré pour cet exercice.

Chaque module correspond à un fichier source qui se trouve dans un dossier `util` commun à tous les contrôleurs. On retrouve `motors.c`, `leds.c`, `prox_sensors.c` et `ground_sensors.c`. Les fichiers *headers* associés contiennent désormais des blocs de documentation selon la syntaxe Doxygen.

La compilation de ces modules se fait simplement en incluant le *Makefile* présent dans le même dossier.