



Projet de Microinformatique
Pucky : Pompier débutant



Shadi NAGUIB

Arthur RÉMONDEAU

Table des matières

1	Introduction	2
1.1	Donnée et cahier des charges	2
1.2	Notre projet	2
2	Fonctionnement général	3
2.1	Gérance de threads	3
2.2	Déroulement d'une simulation	3
2.3	La cartographie	3
2.4	Le déplacement	4
2.5	Reconnaissance des couleurs et réactions	5
3	Analyse et difficultés	6
3.1	Limites et difficultés	6
4	Conclusion	6

1 Introduction

1.1 Donnée et cahier des charges

Ce projet est réalisé dans le cadre du cours de Microinformatique (6ème Semestre de Bachelor Microtechnicien). Nous devons ici utiliser un robot 'e-puck2', déjà introduit lors des TPs de ce cours. Libre à nous de choisir la portée de notre projet et la manière d'y arriver via notre robot.

Quelques contraintes nous sont cependant tout de même imposées :

- Le projet doit être réalisé sur la base de la librairie e-puck2_main-processor vue lors des TPs 4-5;
- Les capteurs suivants doivent impérativement être utilisés (au minimum) :
 - Les deux moteurs pas à pas;
 - Un des deux capteurs de distance;
 - Un capteur parmi : la caméra, les micros et l'IMU.
- Chaque capteur/actuateur doit être géré par un thread dédié;
- Le code doit être rendu sous la forme d'une librairie qui s'intègre avec la librairie ChibiOS.

1.2 Notre projet

Notre projet est relativement simple en terme de conception et d'idée mais jouit d'une grande adaptabilité et précision. En effet, l'idée est que le robot puisse se repérer dans l'espace puis se déplacer vers une série d'objets (après analyse de leur position) et enfin les analyser et réagir, selon un critère de couleur.

Nous avons souhaité privilégier un robot précis (avec des calculs géométriques poussés) et qui puisse réagir positivement aux changements de position, place et forme des objets.

2 Fonctionnement général

2.1 Gérance de threads

Trois threads composent notre programme : les threads CaptureImage et ProcessImage, prises de la correction du TP4 et notre thread principale : DetectAndExplore. Les threads ont tous la même priorité (NORMALPRIO) et une sémaphore permet à la thread ProcessImage d'attendre que la thread CaptureImage prenne l'image afin de pouvoir en extraire la couleur.

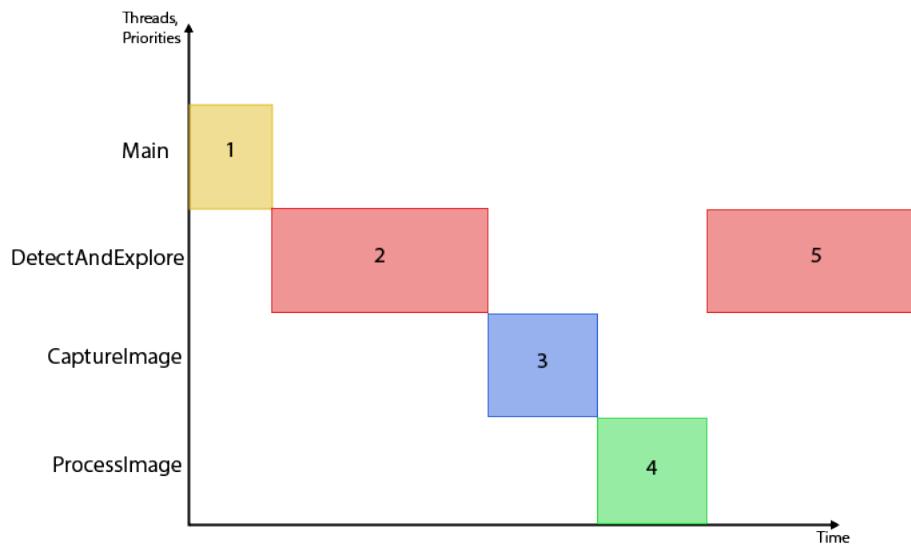


FIGURE 1 – Diagramme de threads de notre projet

2.2 Déroulement d'une simulation

Notre robot se compose d'un seul programme qui se lance en changeant l'état du sélecteur de 0 à n'importe quelle autre valeur.

L'e-puck est placé quelque part dans notre arène circulaire et ensuite allumé. Après avoir changé l'état du sélecteur le programme va directement se lancer. Il commence par faire un premier tour sur lui-même pour repérer la distance qui le sépare des murs de l'arène. Un nombre indéfini d'objets est ensuite disposé autour de lui, ils sont soit de couleur rouge soit verte. L'e-puck effectue alors un nouveau tour sur lui-même pour estimer la position et le centre des objets qui l'entourent. Il va ensuite se diriger vers les objets un par un et procéder à une analyse de couleur à chaque objet. Si l'objet est vert, l'e-puck sera content, allumera ses leds RGB en vert et passera à l'objet suivant. Si l'objet est rouge, on considère qu'il est alors en feu et l'e-puck reproduit les fréquences des pompiers en clignotant rouge et bleu, avec les leds RGB et les LEDS rouges.

2.3 La cartographie

Pendant chaque tour que le robot fait sur lui-même, il prend 324 mesures avec le capteur Time of Flight et stocke toutes ces données dans un tableau dépendant de s'il s'agit du premier ou du deuxième tour. Afin de détecter un objet, on estime qu'il faut que la différence entre les mesures

prises entre les deux tours soit supérieure à un seuil de 60 mm pour dix mesures consécutives. Après avoir fini de comparer les mesures, le robot rempli un tableau de structures correspondant aux paramètres de chaque objet détecté (position du début de l'objet, de sa fin, son centre et de la distance mesurée avec le centre).

2.4 Le déplacement

Maintenant que le robot pompier possède toutes les données sur le nombre d'objets détectés, où se situe le centre de chaque objet et la distance entre le robot et chaque centre, il peut commencer à bouger vers les objets l'un après l'autre et détecter leur couleur.

Il se tourne d'abord dans le sens horaire vers le premier objet, et s'arrête lorsque le nombre de steps de la roue gauche atteint l'angle nécessaire pour être en face du centre du premier objet. Il avance ensuite vers ce centre et s'arrête lorsqu'une distance d'à peu près 6 cm (mesurée avec le capteur TOF) le sépare avec l'objet. Afin d'être sûr que le robot ne continue pas d'avancer s'il est trop proche de l'objet et si le capteur TOF ne fonctionne pas correctement, il s'arrêtera également si les valeurs renvoyées par les deux capteurs IR à l'avant du robot détectent un obstacle. Après avoir étudié la couleur de l'objet et avoir réagit en fonction de cette couleur, le robot fait des calculs trigonométriques dans le but de trouver :

- l'angle qu'il doit atteindre pour faire face au centre de l'objet suivant
- la distance qu'il doit parcourir pour être à une distance de 6 cm de cet objet
- l'angle pour aligner sa caméra avec le centre de l'objet

Sur l'image ci-dessous les valeurs que le robot doit calculer pour arriver au deuxième objet sont les valeurs entourées sur le schéma ci-dessous (les angles α , β , ψ et la distance "c" entre la position du centre du robot lorsqu'il est au premier objet et où son centre devrait être afin de pouvoir lire la couleur du deuxième objet).

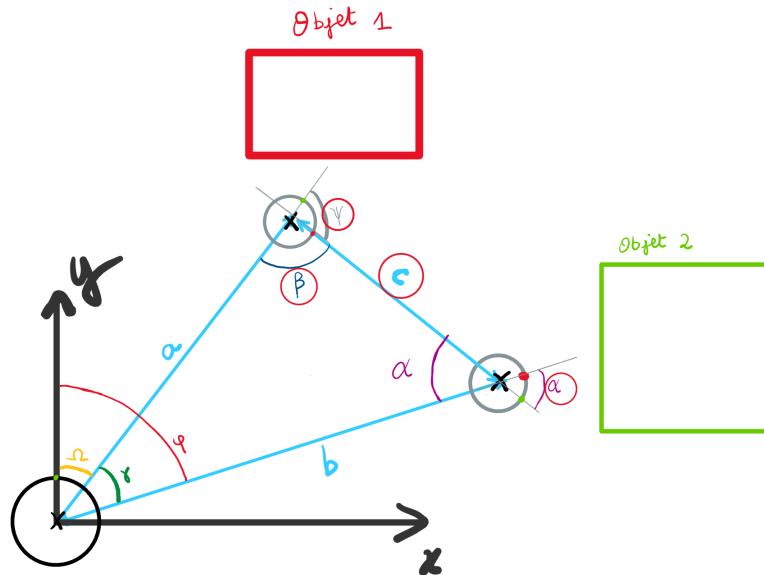


FIGURE 2 – Déplacement du robot du premier au deuxième objet

En effet, nous avons les formules suivantes d'après la loi des cosinus :

$$c = a^2 + b^2 - 2ab \cdot \cos(\gamma)$$

$$\beta = \arccos\left(\frac{a^2 + c^2 - b^2}{2ac}\right)$$

$$\psi = \pi - \beta$$

$$\alpha = \pi - \beta - \gamma$$

En effet, les distances a et b ont été mesurées avec le capteur TOF pendant le deuxième tour du robot, et l'angle γ est connu étant donné qu'il correspond à la différence absolue entre le nombre de steps des roues entre le centre du premier et du deuxième objet.
Le robot doit donc tourner d'un angle ψ , avancer de la distance c, et puis tourner d'un angle α pour avoir sa caméra face au centre de l'objet. Il est maintenant près à voir si la maison est en train de brûler (couleur rouge) ou si elle est hors danger (couleur verte).

2.5 Reconnaissance des couleurs et réactions

Pour reconnaître les couleurs, nous avons opté sur l'alternance de bandes noires et de couleur (Rouge ou Vert, selon l'objet). En effet, nous avons au préalable effectué des tests pour savoir si il serait plus judicieux d'opter pour une alternance des bandes avec du noir ou avec du blanc. Cependant, le blanc présente bien évidemment une intensité bien trop élevée et ne nous permet pas de distinguer les différentes couleurs, raison pour laquelle nous avons opté pour les bandes noirs.

Pour distinguer les couleurs, nous interprétons le gradient d'intensité, présent entre le noir et la couleur. Nous extrayons de chaque pixel les bits qui codent l'intensité de chaque composante de couleur (Rouge, Vert, Bleu) puis nous regardons le gradient le plus important entre le noir et chaque composante et en déduisons ainsi la couleur. En effet, avec les imprimantes utilisées, il est très difficile d'avoir exactement une couleur à niveau de [255 0 0] (Rouge) ou [0 255 0] (Vert), on retrouvera donc ainsi toujours de légères intensités d'autres couleur sur nos impressions. L'imprécision des capteurs renforce une fois de plus cet effet. C'est quelque chose que l'on peut facilement observer sur les figures 2 et 3 ci-dessous : On a mesuré l'intensité des pixels verts sur une surface rouge et une surface verte. On observe que le gradient entre le vert et le noir est bien plus dessiné que celui entre le rouge et le noir, lorsqu'on analyse les pixels verts de chaque image (figures 4 et 5).

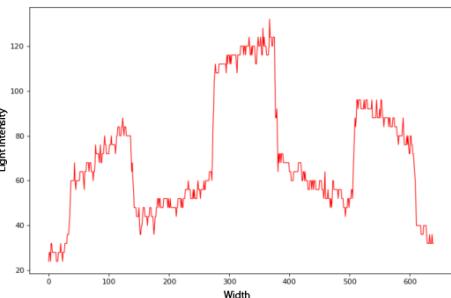


FIGURE 3 – Intensité du vert sur une surface verte

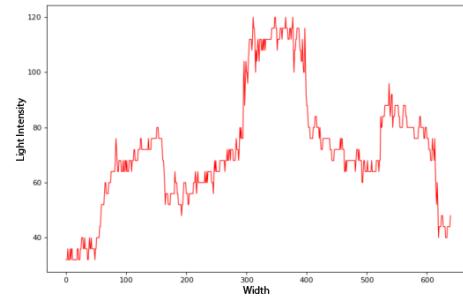


FIGURE 4 – Intensité du vert sur une surface rouge



FIGURE 5 – Objet vert



FIGURE 6 – Objet rouge

Pour contrer ces approximations, une couleur est validée grâce à 20 mesures d'intensité différentes, desquelles on fait la moyenne, pour avoir un résultat qui se rapproche le plus de la réalité.

À cette détection de couleur on associe une réaction via les leds et le buzzer de l'appareil. Un objet rouge et noir entraînera le déclenchement des LEDS rouges en clignotement, des LEDS rgb en alternance bleu-rouge et un buzz bi-fréquentiel imitant la sirène des pompiers.

3 Analyse et difficultés

3.1 Limites et difficultés

Le capteur le plus utilisé dans ce projet est le capteur Time of Flight. Ce dernier nous a posé quelques problèmes surtout lors de la cartographie et de la détection d'objets, dûs en partie à l'utilisation d'une couleur blanche pour les murs et d'une alternance entre le noir et le rouge/vert pour les objets. En effet le noir ne reflète pas bien les ondes arrivant sur lui.

Les informations lues par le capteur n'étant pas toujours les plus précises, nous avons utilisé des objets assez gros et avons fixé un seuil à 10 cm au dessus duquel la différence entre les mesures prises pendant le premier tour (sans les objets) et celles prises pendant le deuxième tour indiquerait qu'un objet existe. Nous avons fait en sorte que notre programme détecte un objet si et seulement si dix mesures consécutives du capteur présentent cette différence de plus de 10 cm afin de ne pas détecter un objet non existant à cause d'une fausse mesure.

L'utilisation de la caméra, comme discuté précédemment, est assez imprécis et requiert malheureusement la mise en place d'approximations et de moyennes. Ainsi, bien que la différence semblerait facile à interpréter, les capteurs de la caméra nous retournent des valeurs d'intensité qui sont bien éloignées de ce à quoi on s'attend.

4 Conclusion

Dans l'ensemble, nous avons produit un travail qui nous satisfait et qui utilise au mieux les capteurs fournis. Notre programme est optimisé et les calculs de déplacements et positionnement de l'e-puck sont précis, dans les limites des capteurs IR et TOF. Il est également modulable, comme nous le désirions initialement : les objets peuvent être ré-arrangés, ajoutés et enlevés au bon vouloir de l'utilisateur, sans compromettre le fonctionnement correct du programme.

Les capteurs et les moteurs ont été limitants dans le développement et la précision de notre projet. Nous avons cependant bien appris à développer et à approfondir à partir de librairies pré-fournies.