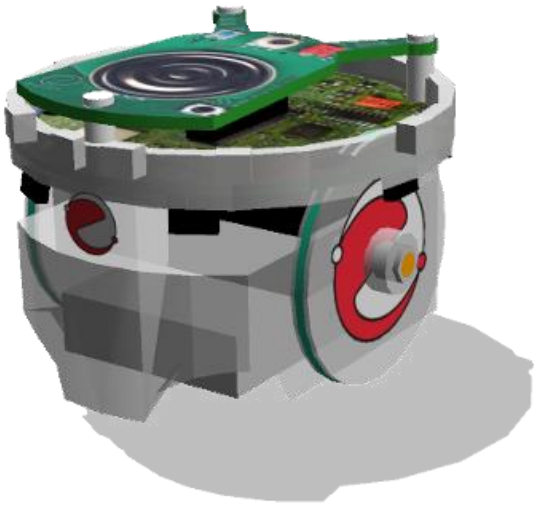
Projet E-Puck : Programmation sous Python

Robotique mobile



Enseignant : **Christophe SABOURIN**
Réalisé par **: HATHARASINGHA Nadeesha, FERREIRA Alicia** ITS2

Année 2021-2022

Table des matières

Table des figures

Introduction

Dans le cadre du cours de robotique mobile dispensé par Mr SABOURIN, nous avons étudié le concept de robot ainsi que son fonctionnement.

Au terme de ce module, il nous a été demandé de réaliser un projet. Le but de celui-ci est de réaliser une simulation automatique d'un robot cherchant une balle de couleur verte et se trouvant dans un environnement encombré d'obstacles. Le robot doit se déplacer dans l'environnement et dès qu'il aperçoit la balle de couleur verte, se diriger vers elle et la suivre. Une fois arrivé devant la balle, le robot doit allumer toutes ses LED et indiquer la position où il se trouve. Ce projet a été réalisé sur le simulateur Webots et a été codé en python.

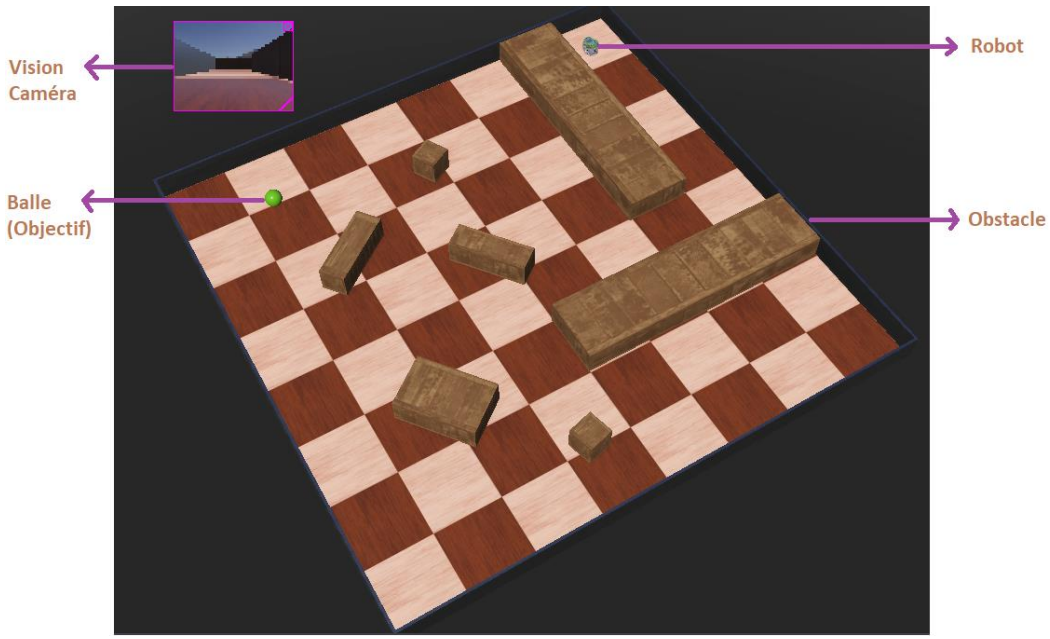


Figure 1 : Robot E-Puck et son environnement (Interface Webots)

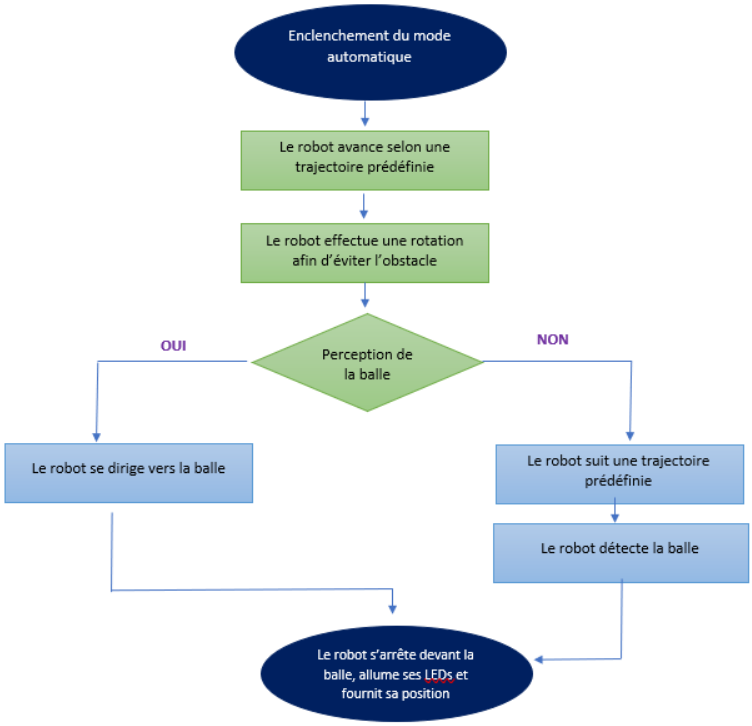


Figure 2 : Logigramme du projet E-Puck

Le robot e-puck possède 8 capteurs disposés autour de lui, qui interviennent pour définir sa position, sa distance, sa luminosité. De plus, une caméra située à l'avant du robot lui permet de reconnaître des objets se trouvant dans son environnement.

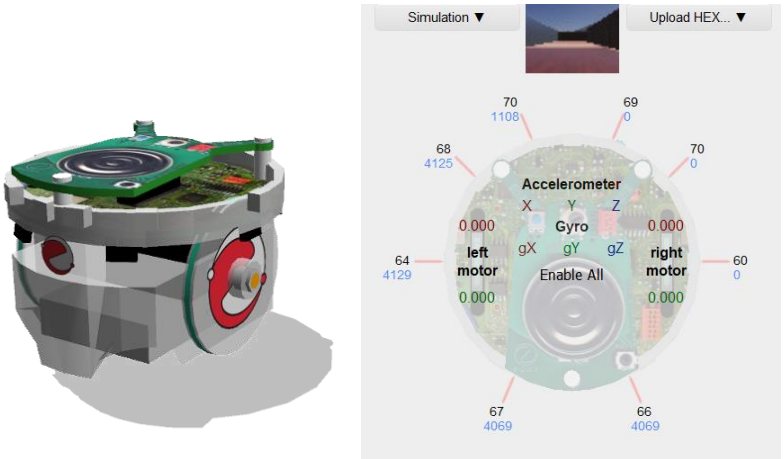


Figure 3 : Robot E-Puck et ses caractéristiques

Au cours de ce module, nous avons pu aborder un type de navigation appelé réactive. Par définition, la navigation réactive est une stratégie basée sur le comportement qui utilise les valeurs courantes de capteurs pour décider de l'action à effectuer. Nous avons choisi ce type de navigation car c'est une mise en œuvre assez simple sur des environnements simples comme proposé dans ce projet.

Pour mener à bien ce projet, nous avons procéder en plusieurs étapes :

1. Evitement d'obstacles
2. Stratégie de contrôle
    a) Stratégie de recherche de la balle
    b) Se diriger et suivre la balle
    c) S'arrêter devant la balle, allumer les LEDs et donner sa position

1. Evitement d'obstacles

L'environnement étant parsemé d'obstacles, nous devons coder le comportement que le robot doit adopter lorsqu'il en rencontre un.
Pour cela, nous avons décidé d'adopter une stratégie d'évitement des obstacles : lorsque le robot s'avance dans une direction (droite ou gauche) où se trouve un obstacle, les capteurs du robot situés dans cette direction préciseront lorsque l'obstacle en question sera atteint, le robot tournera alors dans la direction opposée.
Par exemple, si le robot avance vers un obstacle qui se trouve à sa droite, un de ses capteurs situés à droite lui transmettra l'information et le robot évitera ainsi l'obstacle en tournant à gauche et inversement pour un obstacle situé à sa gauche.

```
# Evitement obstacle
# A compléter ================================================
if right_obstacle:
    leftSpeed = -0.5 * MAX_SPEED
    rightSpeed = 0.5 * MAX_SPEED
if left_obstacle:
    leftSpeed = 0.5 * MAX_SPEED
    rightSpeed = -0.5 * MAX_SPEED
```

Figure 4 : Code python permettant au robot l'évitement d'obstacles

2. Stratégie de contrôle

Afin d'atteindre l'objectif qui est ici la balle verte, nous avons mis en place une stratégie globale comportant trois grandes étapes dans le mode automatique.

```
#Stratégie globale: 3 etapes
    #On clique sur s pour que le robot commence à avancer et effecue sa trajectoire
    if etape_strategie_controle==0 and (currentKey== ord("s") or currentKey== ord("S")):
        etape_strategie_controle=1

   #Le robot a détecté la balle et va avancer en direction de la balle
    if etape_strategie_controle==1 and balle_reperee:
        etape_strategie_controle=2

   #Le robot a trouvé la balle: il allume ses LED et donne sa position
    if etape_strategie_controle==2 and balle_trouvee:
        etape_strategie_controle=3
```

Figure 5 : Code python de notre stratégie de contrôle

a) Stratégie de recherche de la balle

Dans la première étape, on commence par cliquer sur la touche « s » afin que le robot, initialement à l'arrêt, puisse commencer à avancer.
Cette première étape consiste à élaborer, en fixant au robot différentes positions à atteindre ainsi que des angles de rotation permettant au robot d'y parvenir, une trajectoire par défaut permettant au robot d'explorer l'ensemble de son environnement afin de pouvoir repérer la balle à n'importe quel endroit.
Nous avons ainsi établi une stratégie de recherche de la balle composée d'une dizaine d'étapes permettant ainsi d'explorer l'ensemble des recoins de l'environnement.

```python
#Recherche de la balle
if etape_strategie_controle==1:
    #gestion séquentielle des étapes

    if etape_recherche_balle==0:
        leftSpeed  = 0.75 * MAX_SPEED
        rightSpeed = 0.75 * MAX_SPEED
        print("Etape0= OK")
        if gps.getValues()[2]>=-0.18 and gps.getValues()[2]<=-0.15:
            etape_recherche_balle+=1
            print("Go Etape1")

    if etape_recherche_balle==1:
        leftSpeed  = 0.75 * MAX_SPEED
        rightSpeed = -0.75 * MAX_SPEED
        print("Etape1= OK")

        if angle_rotation<=-4.82 and angle_rotation>=-5.02:
            etape_recherche_balle+=1


    if etape_recherche_balle==2:
        leftSpeed  = 0.5 * MAX_SPEED
        rightSpeed = 0.5 * MAX_SPEED
        print("Etape2= OK")

        if gps.getValues()[0]<=-0.30 and gps.getValues()[0]>=-0.35 :
            etape_recherche_balle+=1
```

Figure 6 : Code python permettant la recherche de la balle

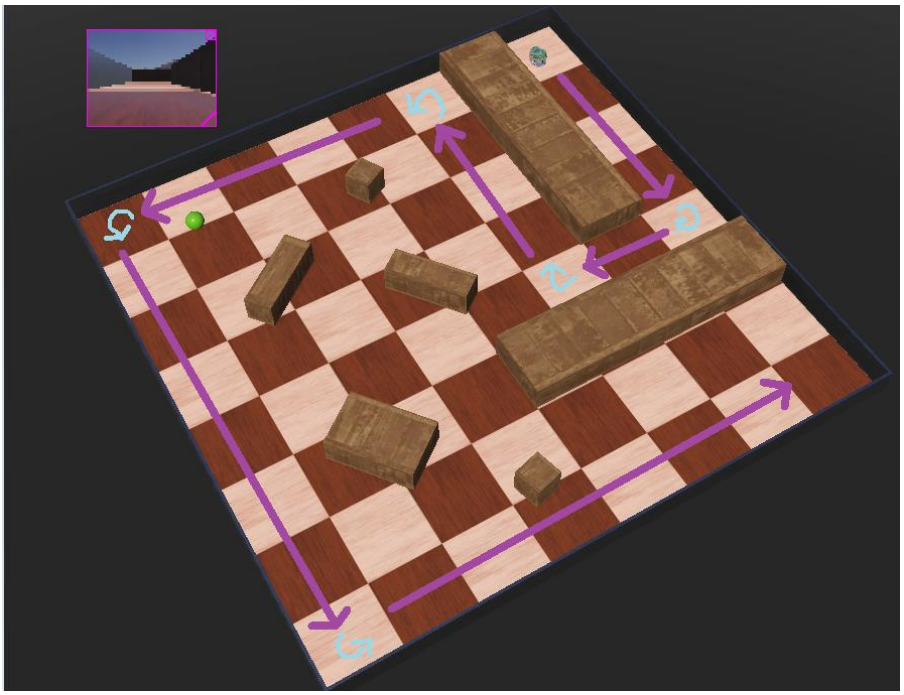


Figure 7 : Etapes de la trajectoire du robot pour la recherche de la balle

b) Se diriger et suivre la balle

La deuxième étape est déclenchée seulement lorsque la balle est repérée par le robot lors de sa trajectoire programmée dans l'étape précédente : la trajectoire du robot dévie et le robot se dirige alors vers la balle.
Pour réaliser cela, nous avons utilisé la variable index_max qui représente le centre de la balle qui permet de s'orienter selon la position du centre de la balle verte. Ainsi, dans notre cas, si l'indice est inférieur à 20 cela signifie qu'il faut tourner à gauche pour se diriger vers le centre de la balle qui a déjà été repérée. De la même manière, si index_max est supérieur à 31, le robot doit alors tourner à droite. Si l'indice se trouve entre 21 et 30, cela signifie que le centre de la balle se trouve en face du robot et celui-ci avancera donc tout droit vers la balle.

Enfin, la variable average représente le nombre de pixels verts détectés à la caméra. Ainsi, lorsque le nombre de pixels verts est supérieur à 20, cela signifie que l'image capturée par la caméra du robot contient un nombre important de pixels verts qui indique ainsi la position très proche de la balle verte. La balle peut alors être considérée comme trouvée et on peut passer à l'étape suivante.

```
if etape_strategie_controle==2:
#gérer posititionnement du robot une fois que la balle est repérée

   #Se diriger et suivre la balle
    if index_max<20:#gauche
        leftSpeed  = -0.5 * MAX_SPEED
        rightSpeed = 0.5 * MAX_SPEED

    elif index_max>31: #droite
        leftSpeed  = 0.5 * MAX_SPEED
        rightSpeed = -0.5 * MAX_SPEED

    else:#tout droit
        leftSpeed  = 0.5 * MAX_SPEED
        rightSpeed = 0.5 * MAX_SPEED

    if average>20:
        balle_trouvee=1
```

Figure 8 : Code permettant au robot de se diriger et de suivre la balle

c) S'arrêter devant la balle, allumer les LEDs et donner sa position

La dernière étape est déclenchée seulement lorsque la balle est trouvée dans l'étape précédente. Le robot s'arrête alors, allume ses LEDs et affiche sa position courante.

```python
if etape_strategie_controle==3:

    #Le robot s'arrête
        leftSpeed  = 0.0 * MAX_SPEED
        rightSpeed = 0.0 * MAX_SPEED

    #On allume les led
        for i in range(10):
            leds[i].set(1)
            print('Les LED sallument')

     #On affiche la position
        print("Le robot se situe aux coordonnées: x={0:0.2f}, y={1:0.2f}, z={2:0.2f} ".format(gps_values[0],gps_values[1],gps_values[2] ))
```

Figure 9 : Code python permettant au robot de s'arrêter devant la balle, d'allumer ses LEDs et de donner sa position dans l'environnement

Conclusion

Notre projet est fonctionnel et l'objectif a été rempli. En effet, le robot se déplace en mode automatique selon une trajectoire prédéfinie, lorsqu'il aperçoit la balle de couleur verte il se dirige vers elle et s'arrête devant celle-ci puis allume ses LED tout en donnant sa position.
Ce projet nous a permis de découvrir le simulateur Webots et de nous exercer dans le codage en python.