MINIPROJET 3 SumotoriQueen

1. Maqueen Sumotori

Objectif :

MVP (ou Minimum Valuable Product)

Créer un robot Maqueen autonome capable de combattre d'autre robots :

- Il se déplace dans un cercle « ligne noire ».
- Il recherche les autres robots dans ce cercles et se déplace pour les pousser.
- Le vainqueur est le dernier restant en piste.

En plus

Utilisation d'outils pour gêner les autres robots.

On utilisera les ressources : https://microbit-micropython.readthedocs.io/fr/latest/

EVALUATION :

APP	• Rechercher l'information utile à l'aide de sources fiables			
	Je copie des solutions dans des sources sans liens apparents entre elles.	Je trie les éléments intéressant dans les sources rencontrées pour les utiliser dans ma solution.	J'identifie et trie les éléments intéressant dans les sources rencontrées pour les utiliser dans ma solution. Je les documente dans mon code.	Je m'inspire de différentes sources pour créer ma solution, en les comparant entre elles pour trouver la plus adaptée. Je cite et document ces sources.
REA	• Mettre en œuvre une solution, par la traduction d'un algorithme ou d'une structure de données dans un langage de programmation.			
	J'écris les grandes étapes du code.	J'écris un code qui répond au problème.	J'écris un code rigoureux qui répond au problème. Je documente et justifie mes choix de langage.	… J'explique clairement le cadre et les limites de la solution. Je propose des améliorations et alternatives possibles.
	• Imaginer et concevoir une solution, décomposer en blocs, se ramener à des sous-problèmes simples et indépendants, adopter une stratégie appropriée			
	J'écris quelques fonctions.	J'écris et utilise des fonctions que je documente.	J'écris et utilise des fonctions, des classes et des modules adaptés que je documente.	J'écris et utilise les fonctions, classes et modules les plus adaptés au problème. Je documente et explique mes choix

Fonctionnalités demandées :

• *Le robot se déplace dans un cercle*	/3	• *Détecter un autre robot*	**/3**
• *Pousse un autre robot*	/2	• *Documentation du projet sur les sources utilisées*	**/2**

Fonctionnalités plus : (+1pt par fct)

• *Carte qui comptabilise et gère le jeu*		• *Instruments pour déstabiliser les adversaires*	
• *Utilisation des classes*		• *...*	

Code :

• *Lisibilité du code*	/2
• *Variables explicites*	/2
• *Pas de répétitions, utilisation de boucles et de fonctions*	/2
• *Commentaires pertinents*	/2

2. Le Robot Maqueen

Le Robot MaQueen, développé par le fabricant chinois Zhiwei Robotics Corp (exporté en europe et aux USA sous le nom DFRobot), est une extension à la carte Micro :Bit, qui vient le piloter.

Pour le manipuler, nous nous servirons ici d'une classe Maqueen développée par la BBC.

Des méthodes spécifiques sont disponibles dans le module `maqueen.py`

- *avance(vitesse)* : avance en ligne droite. vitesse est un nombre entre 0 et 100. Ce paramètre est optionnel. Si non spécifié, c'est la dernière vitesse spécifiée lors de avance() ou setVitesse() qui sera utilisée.
- *recule()* : fait marche arrière.
- *stop()* : stoppe les moteurs
- *moteurDroit(vitesse)* : fait tourner la roue droite.
- *moteurGauche(vitesse)* : fait tourner la roue gauche.
- *getVitesse()* : renvoie la vitesse paramétrée par setVitesse() ou avance()
- *setVitesse()* : change la valeur de la vitesse utilisée par avance, recule, moteur*
- *distance()* : renvoie la distance (en cm) lue par le capteur ultrason
- *son_r2d2()* et *son_bip()* : effets sonores

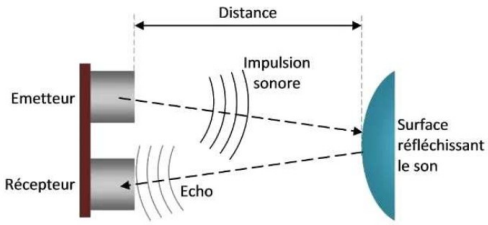
1ers mouvements

```
from microbit import *
from maqueen import Maqueen
mq = Maqueen( )
mq.avance(50)     # 50 : pourcentage de la vitesse maximale
sleep(2000)
mq.stop( )
```

```
from microbit import *
from maqueen import Maqueen
mq = Maqueen ( )
mq.moteurDroit(60)
mq.moteurGauche(30)
sleep(2000)
mq.stop( )
```

Détection d'obstacles

On peut utiliser le capteur à ultrasons pour détecter des obstacles devant le robot.
Pour cela, il suffit d'appeler la méthode distance() de la classe Maqueen, qui renvoie la distance (en cm) à un obstacle à un instant t.




```
from microbit import *
from maqueen import Maqueen
mq = Maqueen( )
d = mq.distance()
display.show(str(d))
sleep(2000)
```

Capteurs de ligne

On utilise les capteurs de ligne pin13 (gauche) et pin14 (droite).

Ex : pin13.read_digital() vaut True si le capteur gauche capte un fond noir.

```
from microbit import *
from maqueen import Maqueen
mq = Maqueen( )
while True:
    if pin14.read_digital() and pin13.read_digital():
        mq.recule()
        time.sleep(500)
        mq.stop()
    if pin14.read_digital() and not pin13.read_digital():
        # tourne à drte
        mq.moteurDroit(60)
        mq.moteurGauche(-60)
        time.sleep(500)
        mq.stop()
    if pin13.read_digital() and not pin14.read_digital():
        # tourne à gche
        mq.moteurDroit(-60)
        mq.moteurGauche(60)
        time.sleep(500)
        mq.stop()
    if not pin14.read_digital() and not pin13.read_digital():
        mq.avance(200)
```

Autres fonctions

Sur le circuit imprimé du robot figurent les adresses des broches pour les LEDs et capteurs de ligne. les voici pour rappel :

- LEDs rouges : 8 (gauche) et 12 (droite). Ex : pin8.write_digital(1)
- Neopixel : pin15
- infrarouge : pin16

```
# Exemple d'éclairage d'ambiance vert avec les neopixels
from microbit import *
from neopixel import NeoPixel
pix=NeoPixel(pin15,4)
for i in range(4):
    pix[i]=(0,255,0)
pix.show()
```

```
# pix.clear() pour eteindre les neopixel
```