TP Robot Maqueen

1 Entrées et sorties de la carte microbit

1.1 Broches de connexion

La carte micorbit comprend des broches de connexion lui permettant d'interagir avec d'autres objets, capteurs et actionneurs. Il est possible d'obtenir des informations sur une broche grâce aux commandes suivantes.

- `pin{num_broche}.get_mode()` pour déterminer si elle est configurée en entrée (*input*) ou en sortie (*output*). Elle renvoie `"read_digital"` si elle en entrée, `"write_digital"` si elle est en sortie, ou `"unused"` si elle n'est pas connectée à un capteur ou un actionneur.
- `pin{num_broche}.read_digital()` renvoie 0 si la broche est à l'état bas et 1 à l'état haut. Certaines entrées sont dites analogiques, elles peuvent recevoir une tension entre 0 et 3,3 volts venant d'un capteur et renvoient une valeur entre 0 et 1023 (et non pas 0 ou 1).
- `pin{num_broche}.read_analog()` lit la valeur renvoyée par la broche.
- `dir(pin{num_broche})` renvoie la liste des fonctions associées à la broche choisie.

1.2 Risques pour la carte

Une mauvaise utilisation des entrées / sorties peut endommager la carte. Celle-ci fonctionne sur 3,3 volts même si elle est alimentée par les 5 volts du câble USB. Il faut veiller à ne pas appliquer de tension supérieure à 3,3 volts sur les entrées / sorties. Afin de protéger la carte des mauvaises manipulations on peut utiliser une carte fille (*shield*). La carte microbit s'emboîte dans la fente (*slot*) prévue à cet effet et c'est le *shield* qui sert d'interface avec les entrées et sorties.

1.3 Bus I2C

En informatique un bus est constitué des liaisons physiques permettant de faire communiquer plusieurs éléments matériels. La carte microbit dispose d'un bus I2C (Inter-Integrated Circuit) sur les broches 19 et 20 permettant d'y relier des capteurs et interfaces (carte fille, robot, etc). La connexion entre la carte et les périphériques est réalisée par l'intermédiaire de la ligne SDA (Serial Data Line) qui transmet les données et la ligne SCL (Serial Clock Line) qui synchronise les échanges. Le nombre maximum d'équipements qu'on peut relier à la carte est limité par le nombre d'adresses disponibles et par la capacité du bus (nombre de lignes et vitesse de transmission des échanges). La carte microbit dispose d'un autre bus plus rapide, le bus SPI (Serial Peripheral Interface), utilisant trois lignes (broches 13, 14, 15).

2 Robot Maqueen

2.1 Équipements

Le robot Maqueen est équipé de :
— deux moteurs contrôlables séparément ;
— deux roues arrière et une avant ;
— deux LED rouges à l'avant
— quatre LED rvb Neopixel (éclairage d'ambiance) sous le châssis ;
— deux capteurs de suivi de ligne (blanc ou noir) sous le châssis ; le capteur envoie 1 lorsqu'il détecte le blanc et 0 quand il détecte le noir ;
— un capteur à ultrason amovible en façade ;
— un capteur infrarouge (télécommande) en façade ;
— une alimentation par piles (trois piles AAA) fixée au-dessus des moteurs ;
— un buzzer sous le boîtier des piles.

Le robot Maqueen est conçu pour que ces équipements soient accessibles par les pin de la carte microbit suivantes.

Entrées		
Équipement	Répère	Pin
Capteur de ligne droit	lineR	14
Capteur de ligne gauche	lineL	13
Capteur infrarouge	ir	16
Capteur Ultrason	trig echo	1 2

Sorties		
Équipement	Répère	Pin
LED rouge droite	ledR	12
LED rouge droite	ledL	8
LED Neopixel	rgb0, rgb1, rgb2, rgb3	15
Buzzer		0

2.2 Utilisation des moteurs

L'utilisation des moteurs se fait grâce au bus I2C à l'aide de la commande suivante :

```
i2c.write(0x10, bytearray([adr. moteur, mode, vitesse]))
```

où
— `0x10` désigne l'adresse du robot Maqueen pour la carte Microbit ;
— `adr. moteur` désigne l'adresse du moteur à actionner :
  — `0x00` pour le moteur gauche ;
  — `0x02` pour le moteur droit ;
— `mode` vaut :
  — `0x0` pour la marche avant ;
  — `0x1` pour la marche arrière ;
  — `0` pour l'arrêt ;
— `vitesse` est un entier compris entre 0 et 255.

Exemples :

```
i2c.write(0x10, bytearray([0x00, 0x0, 100])) # moteur gauche en marche avant
i2c.write(0x10, bytearray([0x02, 0x0, 100])) # moteur droit en marche avant

i2c.write(0x10, bytearray([0x00, 0x1, 100])) # moteur gauche en marche arrière
i2c.write(0x10, bytearray([0x02, 0x1, 100])) # moteur droit en marche arrière

i2c.write(0x10, bytearray([0x00, 0, 0])) # arrêt du moteur gauche
i2c.write(0x10, bytearray([0x02, 0, 0])) # arrêt du moteur droit
```

Exercice 1. Écrire des programmes permettant au robot Maqueen d'effectuer les actions ci-dessous. *Attention :* on veillera à ce que le robot ne percute pas d'obstacles ou ne tombe pas en l'utilisant de préférence au sol et en l'attrapant si nécessaire.

1. Avancer en ligne droite.
2. Reculer en ligne droite.
3. Faire un demi tour.
4. Tourner sur lui-même.
5. Décrire un cercle.

2.3 Utilisation des LED Neopixel

En plus des deux LED rouges à l'avant, le Maqueen possède quatre LED Neopixel sous le châssis. Celles-ci sont contrôlables indépendamment les unes des autres grâce à la bibliothèque Neopixel.

Exemple : le programme suivant attribue des couleurs aléatoires à chaque LED.

```
from microbit import *
import neopixel
from random import randint

# On déclare la pin et le nombre de LED
led = neopixel.NeoPixel(pin15, 4)

while True:
    for i in range(0, 4):
        rouge = randint(0, 255)
        vert = randint(0, 255)
        bleu = randint(0, 255)
        led[i] = (rouge, vert, bleu)

        led.show()
        sleep(100)
```

2.4 Utilisation des capteurs de lignes

Exercice 2. Programmer le robot Maqueen afin qu'il affiche :
- une flèche vers le bas s'il ne détecte aucune ligne ;
- une flèche vers la gauche s'il détecte une ligne à gauche ;
- une flèche vers la droite s'il détecte une ligne à droite ;
- une flèche vers la haut s'il détecte une ligne à gauche et à droite.

En utilisant les capteurs de lignes, on peut programmer le robot pour qu'il circule en suivant une ligne.

Exercice 3. Programmer le robot Maqueen afin qu'il avance le long d'une ligne. Il devra par ailleurs :
- effectuer un-demi tour arrivé au bout de la ligne pour retourner sur celle-ci ;
- allumer ses LED rouges et émettre un bip lors de son demi-tour ;
- afficher un smiley content tant qu'il détecte une ligne ;
- afficher un smiley confus, faire clignoter ses LED rouges et émettre un son d'alerte s'il ne détecte aucune ligne, même après un demi-tour.

2.5 Utilisation du capteur à ultrasons

Les usages du capteur à ultrasons nécessitent régulièrement de mesurer le temps, en microsecondes, pendant où celui-ci détecte (état 1) ou non (état 0) des ultrasons. Cela est possible grâce à la fonction `time_pulse_us` du module `machine` dont la syntaxe est :

```
time_pulse_us(pin, état, timeout_us=1000000)
```

La mesure du temps pendant lequel la pin est dans l'état `état` démarre dès que la pin passe à cet état. Si elle y est déjà lorsque la fonction `time_pulse_us` est appelée, la mesure du temps démarre immédiatement.

Quelque soit l'état dans lequel est la pin, `time_pulse_us` cesse de fonctionner au bout du délai `timeout`. Si ce délai est atteint, deux cas se présentent :

1. l'`état` voulu n'a jamais été mesuré sur la pin, `time_pulse_us` renvoie -2 ;
2. l'`état` voulu était en cours sur la pin et la mesure s'est arrêtée avant que l'`état` prenne fin, `time_pulse_us` renvoie -1.

Exemple : les commandes suivantes permettent de mesurer le temps pendant lequel le capteur détecte des ultrasons.

```
from machine import time_pulse_us

pin2.read_digital() # Activation du capteur ultrason
temps = time_pulse_us(pin2, 1) # Mesure du temps de détection
```

Exercice 4. Programmer la carte Microbit et le robot Maqueen afin qu'il mesure la distance entre lui et les obstacles devant lui et l'affiche sur le pavé LED de la carte.

3 Projets

3.1 Robot piloté

Programmer le robot Maqueen afin qu'il puisse être piloté par une autre carte Microbit. Par ailleurs, il devra :

- afficher un smiley content en marche avant et arrière ;
- afficher une flèche clignotante droite, resp. gauche, lorsqu'il tourne à droite, resp. gauche, et faire clignoter la LED rouge correspondante ;
- allumer ses deux LED rouges et émettre un signal sonore en marche arrière.

On pourra envisager des commandes à l'aide des boutons ou de l'accéléromètre.

3.2 Robot autonome

Programmer le robot Maqueen afin qu'il se déplace seul, en évitant les obstacles. Il devra notamment :

- s'il se retrouve entouré d'obstacles, s'arrêter et signaler qu'il est bloqué en :
  - émettant un signal sonore ;
  - faisant clignoter alternativement les deux LED rouges ;
  - affichant un smiley confus ;
- afficher un smiley content tant qu'il pourra se déplacer.