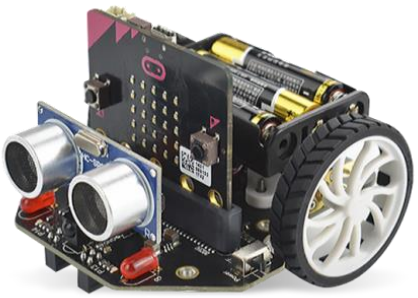
ROBOT MAQUEEN

I –Présentation du robot Maqueen.

Le robot Maqueen micro :bit est un robot très bon marché (autour de 40 à 50€) contrôlé par la carte micro:bit. Il est petit, maniable et facile d'utilisation. Il possède beaucoup de fonctionnalités :

- capteurs de suivi de ligne
- LEDs
- 4 LED RVB neopixel pour éclairage d'ambiance
- capteur de distance ultrason
- buzzer pour effets sonores
- moteurs à engrenage contrôlables séparément par i2c
- alimentation par pack de 3 piles AAA
- capteur infrarouge permettant au robot d'être télécommandé

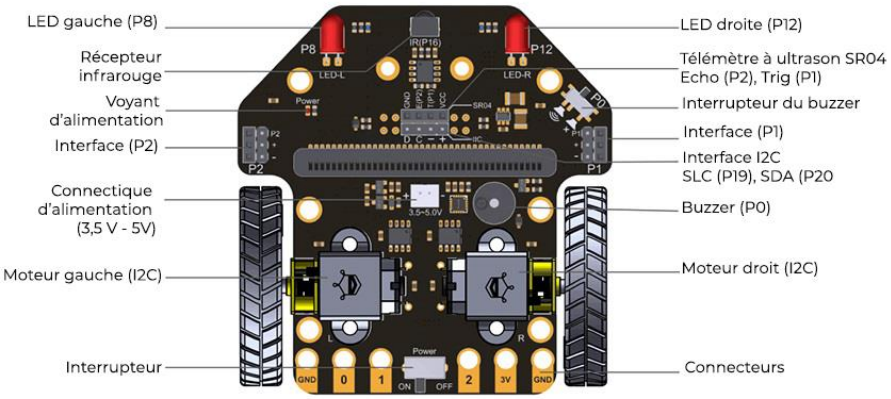


A l'origine, ce robot se programme par blocs. Olivier Lécluse a développé un module python permettant de le programmer facilement sous Python également.

Que trouve-t-on sur le robot Maqueen ?

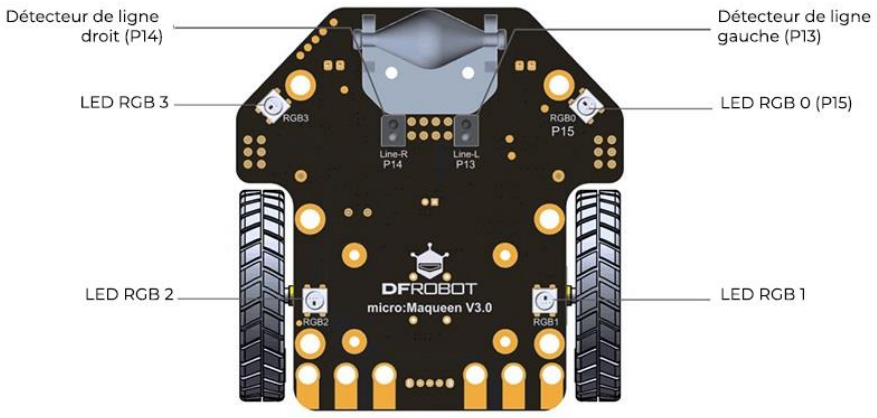
Voici un aperçu du robot avec la vue au-dessus et au-dessous avec les différents modules et composants présents sur le robot Maqueen.

Sur la face supérieure :



Repérer les deux LED rouges, les deux moteurs, et le buzzer sur votre robot Maqueen. Pour les contrôler, il vous faut utiliser les blocs dédiés à ce robot [Maqueen] dans la catégorie "Robot".

Sur la face inférieure :



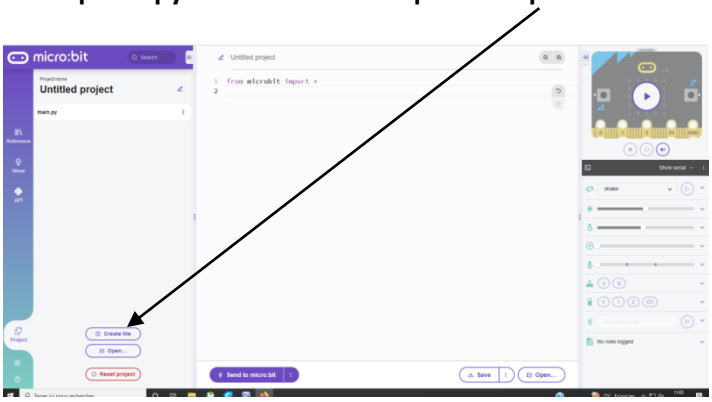
En dessous du robot se trouvent les capteurs de lumière (Line-L et Line-R) qui permettent de repérer une ligne sombre. Repérer également les 4 LED RGB (0, 1, 2 et 3).

II –Prise en mains avec les premiers programmes.

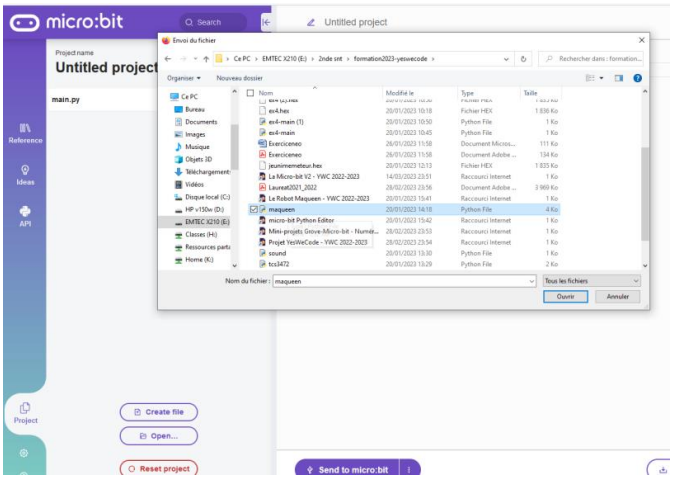
Pour utiliser le module Python d'Olivier Lécluse il faut copier le fichier maqueen.py sur la carte, ce qui se fait facilement avec l'application en ligne https://python.microbit.org/v/3/project

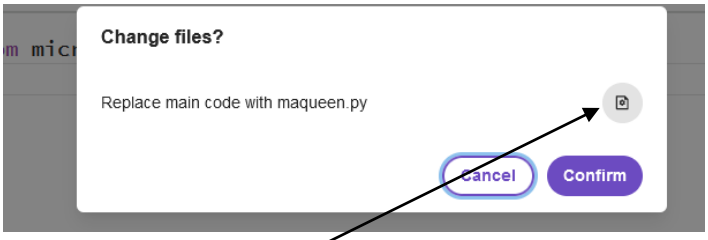
Téléchargement : maqueen.py

Après avoir récupéré le fichier maqueen.py de l'ordinateur cliquer sur open



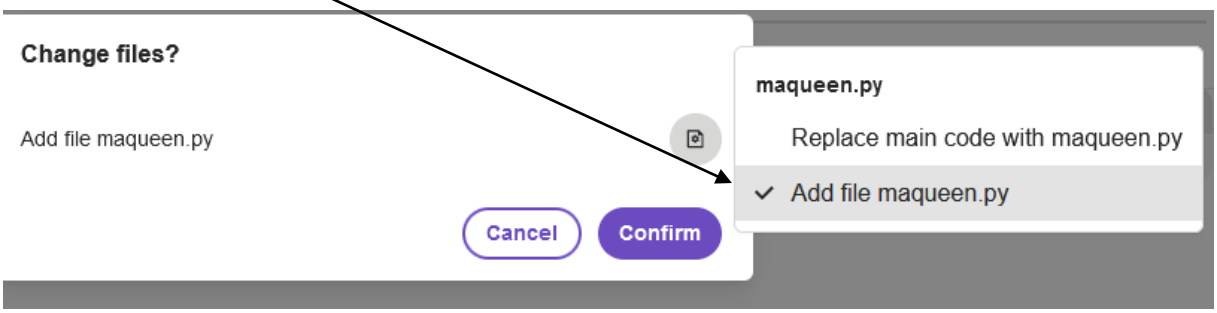
Choisir le fichier maqueen.py





Avant de valider cliquer sur le petit dossier an haut à droite

Choisir Add file maqueen.py et confirmer



Méthodes fournies par le module

Il faut pour utiliser ces méthodes, définir une variable contenant l'objet `Maqueen` de la façon suivante :

```
From maqueen import Maqueen
mq=Maqueen()
```

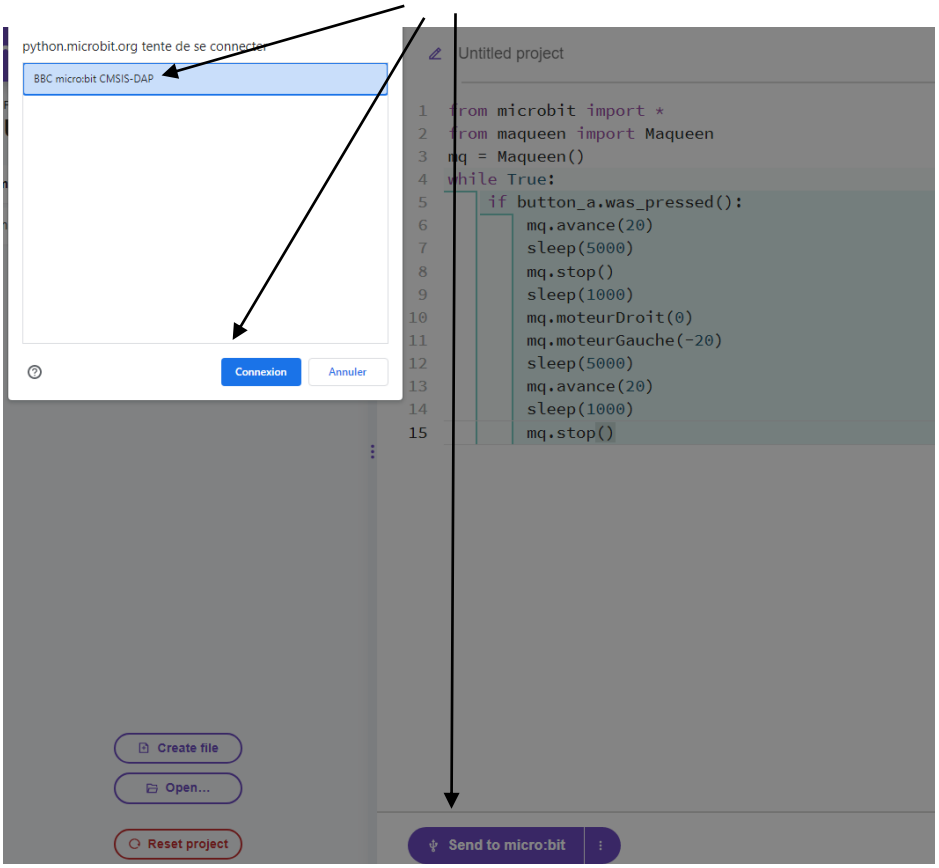
(mq est maintenant notre objet Maqueen auquel on va pouvoir appliquer les methods suivantes)

- `avance(vitesse)` : avance en ligne droite ; `vitesse` est un nombre entre 0 et 100. Ce paramètre est optionnel. S'il est non spécifié, c'est la dernière vitesse spécifiée lors de `avance()` ou `setVitesse()` qui sera utilisée.
  `mq.avance(20)` (Notre objet mq avance à la vitesse 20)
- `recule()` : fait marche arrière en utilisant la valeur de la vitesse indiquée dans `avance`.
  Si on veut indiquer que la vitesse de recul est par exemple 20, il faut écrire l'instruction `mq.setVitesse(20)` avant `mq.recule()`
  `mq.recule()`
- `stop()` : stoppe les moteurs
  `mq.stop()`
- `moteurDroit(vitesse)` : fait tourner la roue droite ; pour faire tourner la roue dans l'autre sens, on indique une vitesse négative.
  `mq.moteurDroit(40)`
- `moteurGauche(vitesse)` : fait tourner la roue gauche ; pour faire tourner la roue dans l'autre sens, on indique une vitesse négative.
  `mq.moteurGauche(20)`
- `getVitesse()` : renvoie la vitesse paramétrée par `setVitesse()` ou `avance()`
  `vitesse_actuelle=mq.getVitesse()`
- `setVitesse()` : change la valeur de la vitesse utilisée par `avance`, `recule`, `moteurDroit` et `moteurGauche`
  `mq.setVitesse(40)`
- `distance()` : renvoie la distance (en cm) lue par le capteur ultrason
  `dist=mq.distance()`

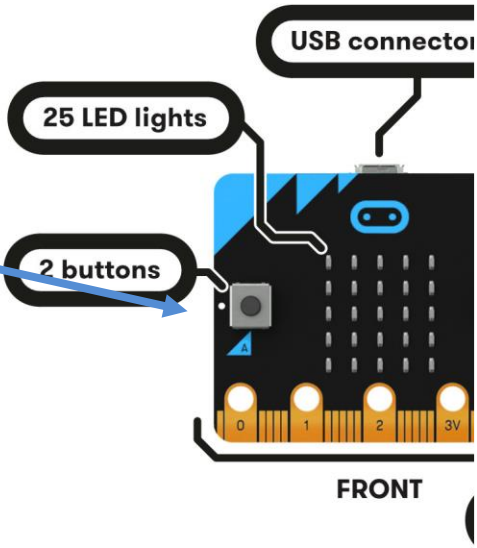
- `son_r2d2()` et `son_bip()` : effets sonores
  `mq.son_r2d2()`
  `mq.son_bip()`

Exercice 1

1. **Charger le programme qui suit dans l'application en ,ligne https://python.microbit.org/v/3/project**
2. **Prévoir l'effet du programme ci-dessous puis le flasher sur la carte.**
3. **Transférer ensuite ce programme sur la carte en la connectant à l'ordinateur :**



```
from microbit import *
from maqueen import Maqueen

mq=Maqueen()

While True:
      if button_a.was_pressed():
            mq.avance(20)
            sleep(5000)
            mq.stop()
            sleep(1000)
            mq.moteurDroit(0)
            mq.moteurGauche(-20)
            sleep(5000)
            mq.avance(20)
            sleep(1000)
            mq.stop()
```



4. **Mettre la carte sur le robot, le tenir hors du sol, mettre le contact puis appuyer sur le bouton a. Vérifier vos prédictions**
5. **recommencer en positionnant le robot par terre.**
6. **Faire quelques essais en modifiant le programme.**
7. **En faisant des essais, remplir le tableau donnant la trajectoire du robot en fonction des vitesses des moteurs droit et gauche. On note vD la vitesse du moteur droit et vG celle du moteur gauche.**

Configuration	Trajectoire
vD = vG avec vd> 0	Trajectoire rectiligne. Marche avant
vD = vG avec vd< 0	
vD = -vG	
vD<vG	
vD>vG	

Leds rouges

La led gauche est sur la broche 8 et la droite est sur la broche 12

Exercice 2

Première activité : Prévoir l'effet du programme ci-dessous puis le flasher sur la carte.

```
from microbit import *

while True:
   pin8.write_digital(1)# allume
   sleep(500)
   pin12.write_digital(1)
   sleep(500)
   pin8.write_digital(0)# éteint
   sleep(500)
   pin12.write_digital(0)
   sleep(500)
```

Mettre la carte sur le robot et mettre le contact (appuyer sur le bouton « reset » pour recommencer).

Deuxième activité : les néopixels : Les 4 neopixels sont sur la broche 15.

Prévoir l'effet du programme ci-dessous puis le flasher sur la carte.

```
from microbit import *
from neopixel import NeoPixel

np=NeoPixel(pin15,4)
for i in range(4):
      np[i]=(255,255,65)
      np.show()
sleep(3000)
np.clear()
```

Mettre la carte sur le robot et mettre le contact, vous pouvez jouer en mettant d'autres couleurs (à relier avec les images et les photosites R,V,B). Blanche : (255,255,255) ; Rouge : (255, 0, 0) ; Vert : (0, 255, 0) ; Bleu : (0, 0, 255) ; Jaune : (255, 255, 65) ; Turquoise: (65 , 255, 255).

Troisième activité : le camion de Pompier :

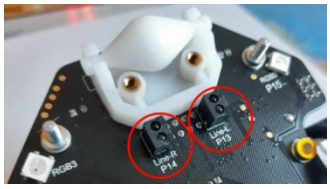
Dans cette partie, nous allons simuler un véhicule de pompier grâce au néopixel (4 LED RGB) et au buzzer intégrés. On définit la couleur en rouge pour le néopixel (R,V,B). La sirène des pompiers répète les 2 fréquences **(435 Hz et 488 Hz)** environ 30 fois en 1 minute. Un cycle dure donc environ 2 secondes (60/30).
On définit alors : **600 ms, la durée de la note et 300 ms, la pause entre les notes**
On fait clignoter en même temps les LED rouges (phares) du véhicule.
Modifier le code suivant puis suivre les étapes décrites dans la partie *Démarrage du robo*t pour transférer le code sur la carte :

```
from microbit import *
import neopixel
import utime
import time

npMaq = neopixel.NeoPixel(pin15, 4)

#fonction permettant de donner sur un pin, la fréquence, la durée d'une note sur le mini HP et la pause avant de la reprduire
def pinpon (pin, noteFrequency, noteDuration, silence_ms = 300):
    microsecondsPerWave = 1e6 / noteFrequency
    millisecondsPerCycle = 1000 / (microsecondsPerWave * 2)
    loopTime = noteDuration * millisecondsPerCycle
    for x in range(loopTime):
        pin.write_digital(1)
        utime.sleep_us(int(microsecondsPerWave))
        pin.write_digital(0)
        utime.sleep_us(int(microsecondsPerWave))
    sleep(silence_ms)
while True:
# mise en place de la couleur rouge sur les neopixels
   for i in range(4):
       npMaq[i] = (…,…, …) # Couleur rouge à choisir
       ….show()
# son du camion de pompier, et leds rouges
   pinpon(pin0, …, …) #(première note du "pin, pon")
   pin8.write_digital(1)
   pin12.write_digital(1)
   time.sleep_ms(1)
   pinpon(pin0, …, …)# (deuxième note du "pin,pon")
   pin8.write_digital(0)
   pin12.write_digital(0)
   time.sleep_ms(1)
```

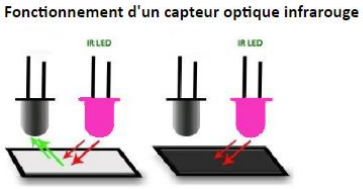
Exercice 3 :

Module suiveur de lignes :

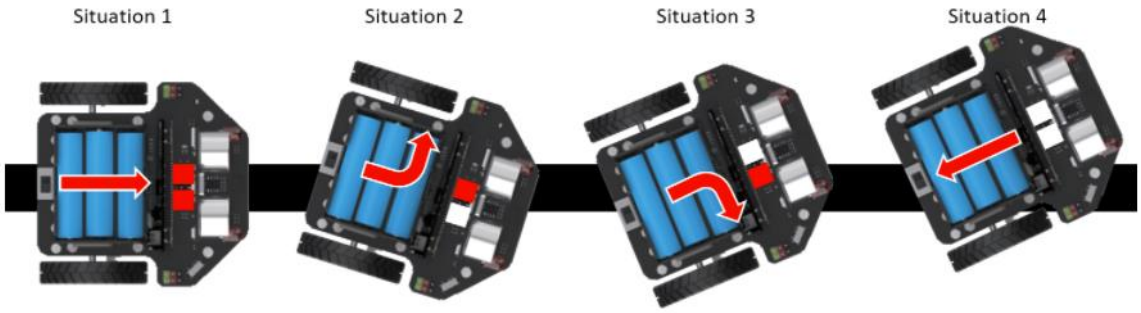
Le robot dispose à l'avant de 2 capteurs optiques infrarouge



Le capteur optique infrarouge est constitué :
d'une LED infrarouge et d'un photo-transistor. Le capteur envoie un signal haut lorsqu'il détecte le blanc et un signal bas quand il détecte le noir.



Plusieurs situations sont possibles :

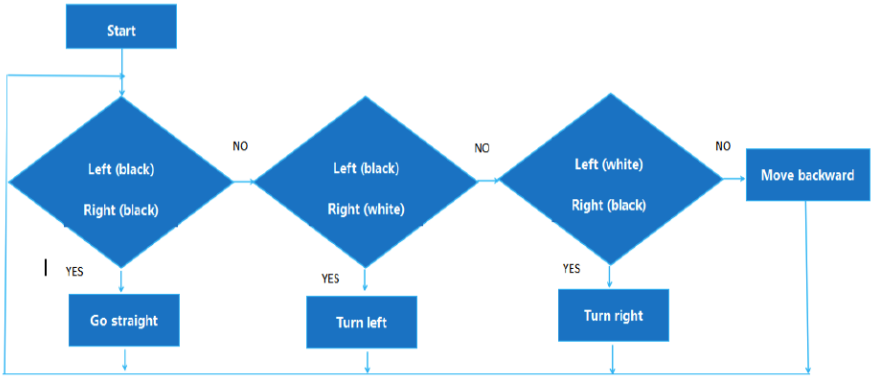


La ligne est détectée par les 2 capteurs --> Avancer
Seul le capteur gauche détecte la ligne --> Tourner à gauche
Seul le capteur droit détecte la ligne --> Tourner à droite
La ligne n'est plus détectée --> Reculer

Prévoir l'effet du programme ci-dessous puis le flasher sur la carte.

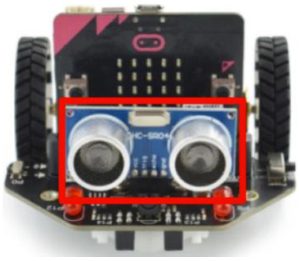
```
from microbit import *

while True:
# Lecture état capteur
   valeur_gauche=pin13.read_digital()
   valeur_droit= pin14.read_digital()
   if valeur_gauche==0:
     pin8.write_digital(0)
   if valeur_gauche==1:
     pin8.write_digital(1)
   if valeur_droit==0:
     pin12.write_digital(0)
   if valeur_droit==1:
     pin12.write_digital(1)
```

Mettre la carte sur le robot, mettre le contact et tester en passant les capteurs sur des lignes noires



A l'aide des fonctions précédentes, programmez le robot pour qu'il suive une ligne noire sur fond blanc.

Module ultrason et évitement d'obstacles :



1-1 La position du module ultrasonique sur la voiture robot maqueen

Le module ultrasonique - cadre rouge.

1-2.En savoir plus sur le principe du module ultrasonique

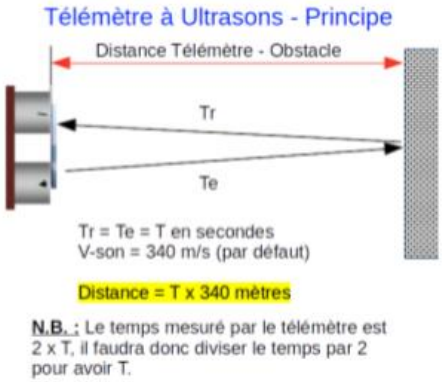
Principe de télémétrie à ultrason:

Le télémètre a deux sondes ultrasonores, qui sont utilisées pour transmettre et recevoir des ondes ultrasonores, respectivement, et la plage de mesure est d'environ 3-450 cm.

Le télémètre mesure le temps écoulé (DURÉE) entre la transmission et le retour de l'onde ultrasonore. Il peut calculer la distance actuelle.

Formule: Distance = Durée * Vitesse du son (340 m/s) / 2

Exemple : Si la Durée (Te+Tr) est de 1s alors Distance = 1*340/2 = 170 m.



2. objectif d'apprentissage : Apprenez à utiliser le télémètre à ultrason. La méthode distance() calcule la distance entre le robot et les éventuels obstacles. La vitesse des ultrasons est celle du son 340m.s^{-1}

Exercice 4

Prévoir l'effet du programme ci-dessous puis le flasher sur la carte.

```
from microbit import *
from maqueen import Maqueen

mq=Maqueen()

while True:
     dist_cm=mq.distance()
     if dist_cm > 10:
             display.show(Image.HAPPY)
     else:
             display.show(Image.SAD)
```

Mettre la carte sur le robot et mettre le contact

Exercice 5

On souhaite rendre autonome le robot. C'est-à-dire que dés qu'il détecte un obstacle à moins de 15cm, il tourne aléatoirement vers la droite ou la gauche en émettant le son R2D2 ou bip.

```
from microbit import *
from machine import *
from music import *
from random import *
from maqueen import Maqueen
```

#Définition de l'objet mq comme étant un robot de la classe Maqueen.

```
mq=Maqueen()
```

#Début du programme

```
while True:
```

 # le robot avance à v=100 en montrant qu'il est content

```
    mq.avance(100)
    display.show(Image.HAPPY)
```

 # si la distance à un obstacle est inférieure à 15 cm, il n'est pas content (sad), ses deux leds rouges s'allument et il s'arrête.

```
    if mq…  < …:
         display.show(…)
         pin8.write_digital(1)
```

```
…
mq….
```

 # tirage au sort entre 2 chiffre 0 et 1

```
r=randint(0,2)
```

 # si le chiffre est 1, alors : mettre le son R2D2, le faire reculer en tournant (roue droite vers l'arrière v =-50) pendant 1s (1000ms) et le faire s'arrêter.

```
if r … 1:
    mq.son_r2d2()
    mq….
    sleep(1000)
    …stop()
```

 # sinon : mettre le son Bip, le faire reculer en tournant (roue gauche vers l'arrière) pendant 1s (1000ms) et le faire s'arrêter.

```
else:
    mq.son_bip()
    …
    …
    …
```

 # le robot reprend sa marche en avant à v=100 en montrant qu'il est content, on éteint les feux rouges

```
mq…….
pin8.write_digital(…)
pin12.write_digital(…)
…(Image.HAPPY)
```

Exercice 6 :

Liaison entre deux cartes :

Deux cartes microbits peuvent échanger des données entre elles par liaison radio. On va donc avoir la carte du Robot Maqueen qui sera la carte esclave et la télécommande qui sera la carte maitre (on la laissera connectée à un port USB afin de l'alimenter)

Prévoir l'effet des programmes ci-dessous puis les flasher sur les cartes.

Chaque groupe choisira une « channel » (entre 0 et 83) différente sous peine de commander le robot du voisin.

Carte émettrice (Maitre)	Carte robot Maqueen (Esclave)

`from microbit import *`
`import radio`
 	
`radio.config(channel=1)`
`radio.on()`
 	
`while True:`
`  if button_a.is_pressed():`
`  radio.send("AVANCE")`
`  if button_b.is_pressed():`
`  radio.send("STOP")`
	`from maqueen import Maqueen`
	`from microbit import *`
	`import radio`
	
	`radio.config(channel=1)`
	`radio.on()`
	
	`while True:`
	`  message_recu = radio.receive()`
	`  if message_recu == "AVANCE":`
	`  display.show(Image.HAPPY)`
	`  mq.avance(50)`
	`  if message_recu == "STOP":`
	`  display.show(Image.PACMAN)`
	`  mq.stop()`

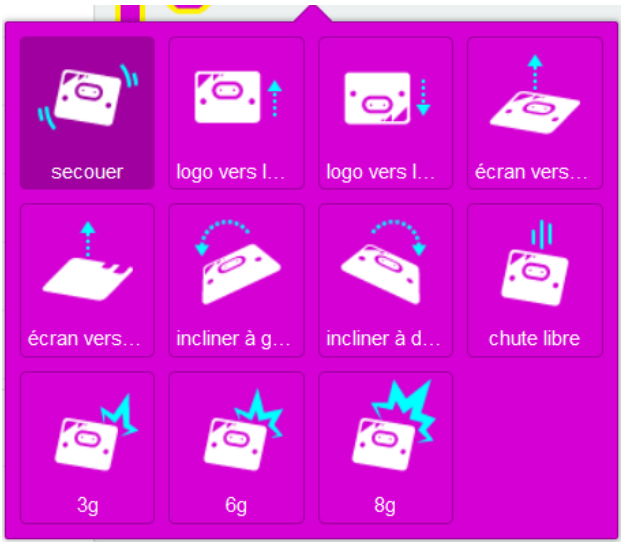
Pilotage sans fil du robot Maqueen (2 possibilités) :

Exercice 7 : avec la carte seule comme Maitre (Utilisation de l'accéléromètre).

Gestures

The really interesting side-effect of having an accelerometer is gesture detection. If you move your BBC micro:bit in a certain way (as a gesture) then micro:bit is able to detect this.

micro:bit is able to recognise the following gestures: `up`, `down`, `left`, `right`, `face up`, `face down`, `free fall`, `3g`, `6g`, `8g`, `shake`. Gestures are always represented as strings. While most of the names should be obvious, the `3g`, `6g` and `8g` gestures apply when the device encounters these levels of g-force.



To get the current gesture use the `accelerometer.current_gesture` method. Its result is going to be one of the named gestures listed above. For example, this program will display a happy emoticon if it's face up:

Compléter les programmes ci-dessous puis les flasher sur les cartes.

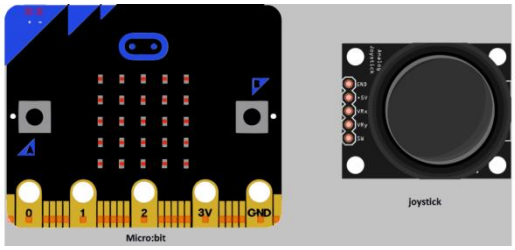
Carte émettrice (Maitre)	Carte robot Maqueen (Esclave)

`from microbit import *`
`import radio`
 	
`radio.config(channel=1) # canal entre 0 et 83 ,`
`radio.on() # pour communication Radio`
 	
`while True:`
 	
`  display.show(1)`
`  if accelerometer.was_gesture('face up'):`
`  …('stop')`
 	
`  if accelerometer.was_gesture('up'):`
`  …('avance')`
 	
`  if accelerometer.was_gesture('down'):`
`  …('…')`
 	
`  if accelerometer.was_gesture('left'):`
`  …('gauche')`
 	
`  if accelerometer.was_gesture('right'):`
`  …('droite')`
	`from microbit import *`
	`from maqueen import Maqueen`
	`import radio`
	
	`radio.config(channel=1) # canal entre 0 et 83 ,`
	`radio.on() # pour communication Radio`
	
	`while True:`
	`  display.show(1)`
	
	`  incoming = …….`
	
	`  while … == 'droite':`
	`  mq.moteurGauche(20)`
	`  incoming = …`
	`  while incoming == 'gauche':`
	`  mq.moteurDroit(20)`
	`  incoming = …`
	`  while incoming == 'avance':`
	`  …(50)`
	`  incoming = …`
	`  while incoming == 'recule':`
	`  mq.recule(20)`
	`  incoming = …`
	`  while incoming == 'stop':`
	`  mq.stop()`
	`  incoming = …`

Exercice 8 : avec la carte et un Joystick.

A l'aide des documents ci-dessous modifier le programme de la carte émettrice de l'exercice 7 pour commander le robot avec le joystick.

FICHE ANNEXE 1 : Commander la carte Micro:bit par la manette joystick



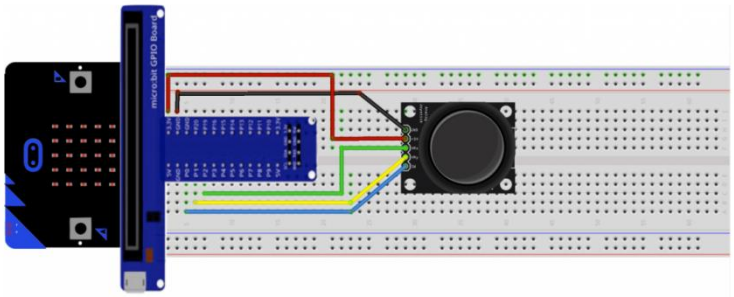
Composants nécessaires

- carte **micro:bit**
- manette joystick
- des fils de connexion
- micro:bit GPIO Expansion Board
- plaque d'essai (Pour le montage 1)

Montage

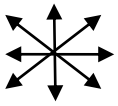
Pour réaliser le montage on connecte:

- La broche +5V à 3.3V du **micro:bit**
- La broche GND au GND du **micro:bit**
- VRx au pin P2 du **micro:bit**
- VRy au pin P1 du **micro:bit**
- SW au pin P0 du **micro:bit**



Programme permettant de repérer les valeurs des entrées du joystick.

```
From microbit import *
# Programme de commande

while True:
    x=pin2.read_analog()
    y=pin1.read_digital()()
    z=pin0.read_analog()
    print('x=',x,'y=',y,"z=",z)
    sleep(1000)
    if button_a.is_pressed():
break
```



Récupération des valeurs des entrées du joystick

x= y= z=	x= y= z=	x= y= z=
x= y= z=	x= y= z=	x= y= z=
x= y= z=	x= y= z=	x= y= z=