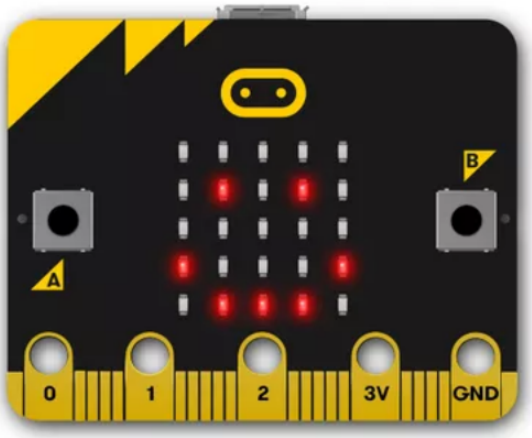
Informatique embarquée, la carte micro :bit



Frédéric Smadja
algoprog.fr

Sommaire

Chapitre 1

Découverte de la carte micro :bit

Plan du T.P.

Retour au sommaire

1 Informatique embarquée

Définition :

- Un **système informatique embarqué** est un **« ordinateur » gravé dans un circuit imprimé**.
- Il possède :
  - un **microprocesseur**,
  - des **mémoires**,
  - un **micro-logiciel**, ou un système d'exploitation (selon les cas),
  - des **capteurs** : permettent de collecter des informations dans le « monde réel ».
  - des **actionneurs** : permettent d'agir sur le « monde réel ».

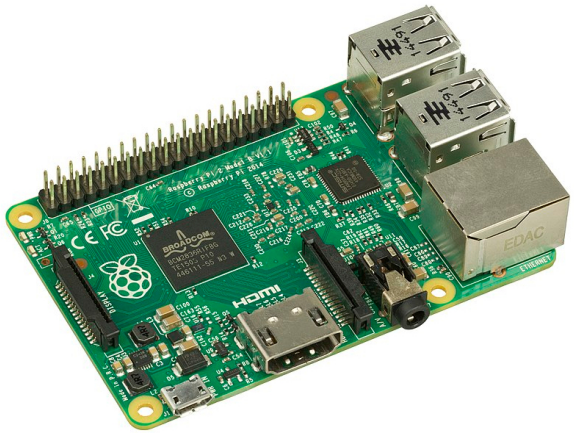


FIGURE 1.1 – Ordinateur monocarte Raspberry Pi

Exemples :

❑ Ouverture d'une voiture à distance :

— appuyons sur le bouton « ouvrir » de nos clés de voiture (**capteur**),
— le **processeur** et son **micro-logiciel** analysent le signal reçu (ouverture ou fermeture ?),
— puis ordonnent à un émetteur radio (**actionneur**), d'émettre un signal pour déverrouiller la voiture.

❑ Robot nettoyeur de piscine :

— les **capteurs** détectent un mur en face,
— le **processeur** et son **micro-logiciel** reçoivent le signal et ordonnent un changement de direction,
— les servomoteurs, **actionneurs**, réalisent le changement de direction.

Remarque :

❑ La définition donnée ci-dessus, un peu large, comprend :

— les **microcontrôleurs** (ou cartes programmables), tels que l'`Arduino` ou la `micro:bit`,

— les **ordinateurs mono-cartes** (ou systèmes sur puce), tels que le `Raspberry Pi` ou les `smartphones`.

2 La carte micro :bit

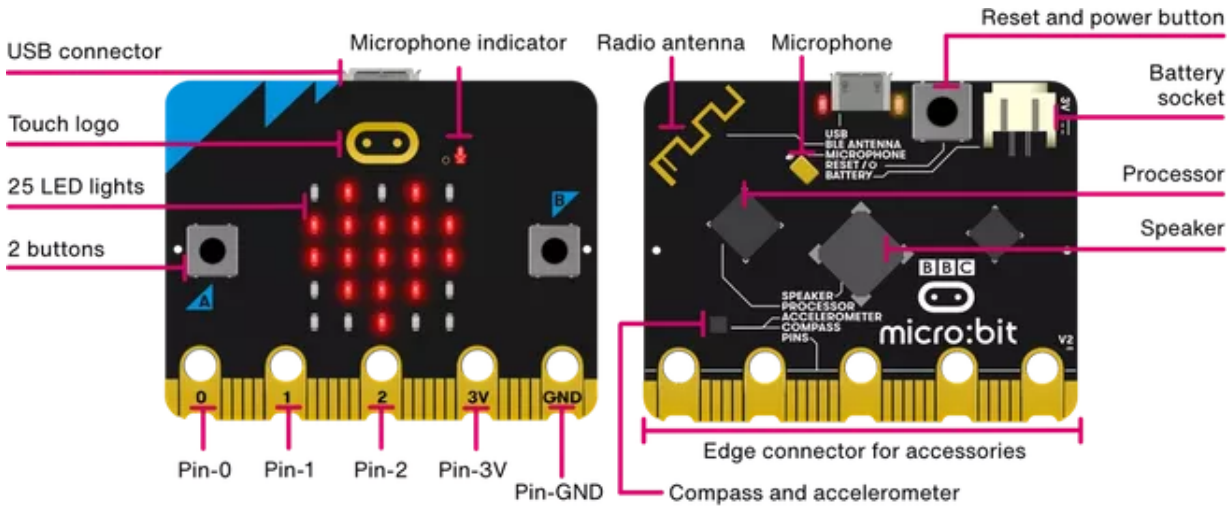
❑ La carte `micro:bit` est un **microcontrôleur** doté d'un **processeur ARM**.

❑ Elle est dotée de nombreux **capteurs** :

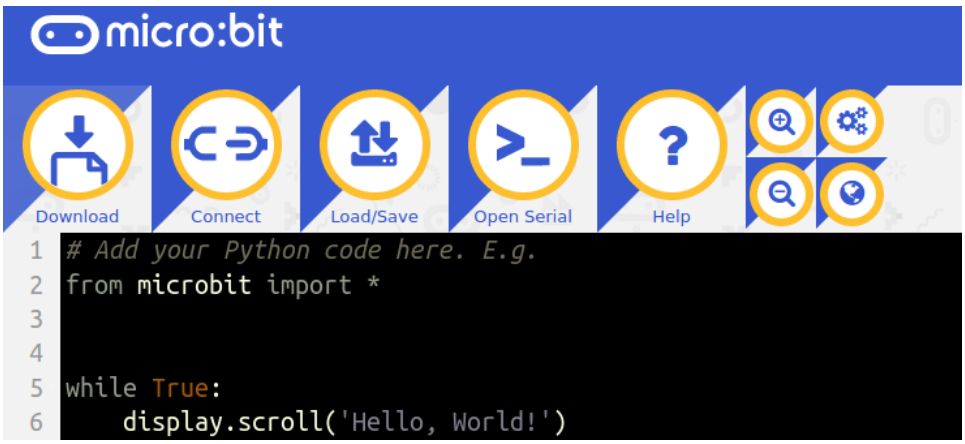
- boutons poussoirs,
- accéléromètre (capteur de mouvement 3D),
- magnétomètre (boussole),
- capteur de lumière,
- capteur de température
- antenne radio pour recevoir des messages...

❑ Elle est dotée aussi d'**actionneurs** :

- 25 LED,
- haut-parleur,
- antenne radio pour envoyer des messages...



❑ Nous la programmerons à l'aide de l'éditeur en ligne fourni sur le site https://python.microbit.org/v/2 :



❑ Programmer la carte, revient à écrire son **micro-logiciel** !

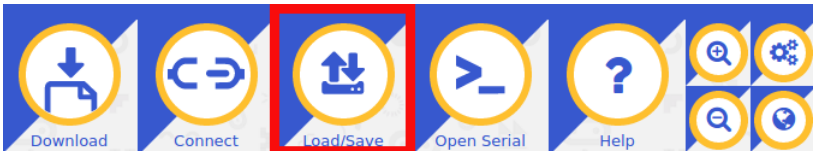
3 LED, boutons, boucles infinies

Exercice 1 : HAPPY !

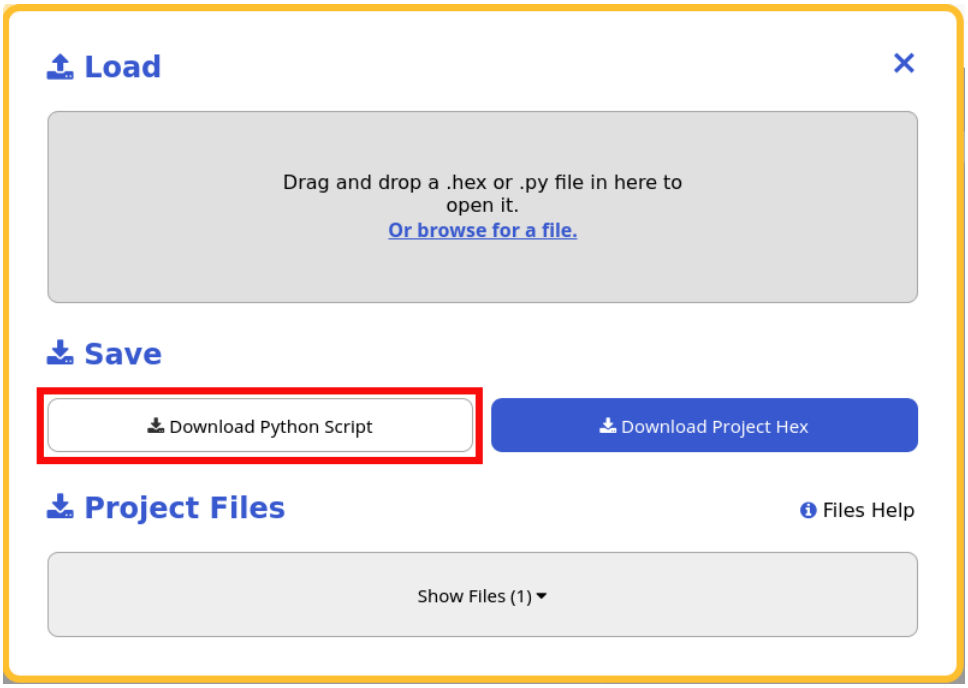
1. Se rendre sur l'éditeur en ligne : https://python.microbit.org/v/2.
2. Effacer le programme proposé et saisir le programme ci-dessous :

```
1 from microbit import *
2
3 display.show(Image.HAPPY)
```

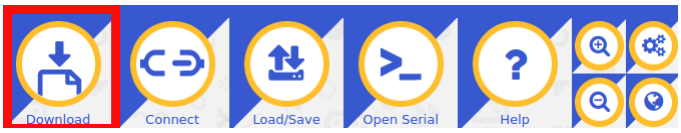
3. Sur votre espace « perso » ou sur votre clé USB, créer un répertoire `Microbit` où vous conserverez vos programmes.
4. Enregistrer votre programme dans ce répertoire :



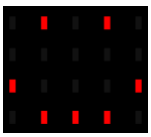
puis :



5. Téléversons maintenant le programme sur la carte :
   - relier la carte et l'ordinateur par le cable USB,
   - téléverser le programme :



   - dans la fenêtre qui s'ouvre, sélectionner la carte `MICROBIT`, puis enregistrer le programme,
   - attendre pendant tout le temps où vous voyez clignoter la LED au dos de la carte (le programme est en train d'être téléversé),
   - vous devriez maintenant voir s'afficher sur la carte notre smiley « HAPPY » :



Exercice 2 : temporisation

Il sera souvent utile d'introduire des temporisations entre les différentes actions.

Ceci peut-être réalisé à l'aide de la fonction `sleep`. Cette dernière prend en argument la durée durant laquelle suspendre l'exécution du programme, exprimée en millisecondes :

```
1 from microbit import *
2
3 display.show(Image.HAPPY)
4 sleep(1000)
5 display.show(Image.SAD)
6 sleep(1000)
7 display.show(Image.RABBIT)
8 sleep(1000)
9 display.show(Image.CLOCK3)
```

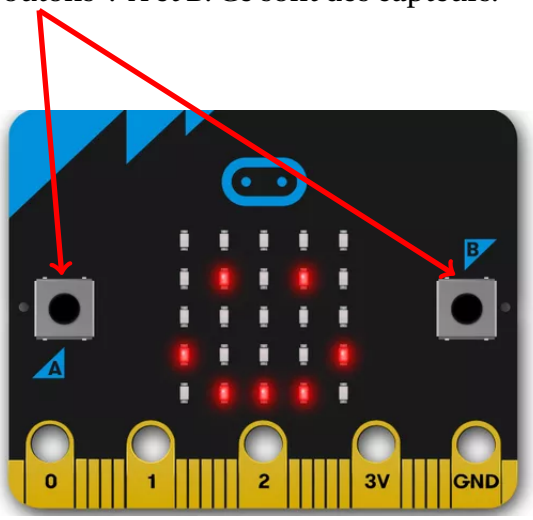
1. Choisir les images de votre choix à l'adresse :

https://microbit-micropython.readthedocs.io/en/latest/tutorials/images.html

2. Écrire un programme affichant une séquence d'images et le téléverser sur la carte.

Exercice 3 : utilisation d'un capteur : bouton

Nous n'avons jusqu'à présent utilisé que des actionneurs (les LED). Passons aux capteurs. La carte dispose sur la face avant de deux boutons : A et B. Ce sont des capteurs.



1. Tester le programme suivant :

```
1 from microbit import *
2
3 if button_a.is_pressed() :
4     display.scroll("A")
5 else :
6     display.scroll("Rien")
```

2. Quel est l'inconvénient de ce programme ?

Exercice 4 : utilisation d'une boucle infinie

Le programme précédent ne s'exécute qu'une seule fois : lors de la mise sous tension de la carte. Nous aimerions obtenir l'affichage du message « A » à tout moment, dès que le bouton « A » est pressé.

Nous aurons besoin pour cela d'une **boucle infinie**, du type du « Répéter indéfiniment » de Scratch :

Scratch

Python

```
1 while True :
2     ...
```

FIGURE 1.2 – Boucles infinies en Scratch et en Python

En effet, la boucle `while`, rencontrée dans le cours de mathématiques, répète le bloc d'instructions qui suit tant qu'une condition est vraie. La condition `True` reste toujours vraie, la boucle va ainsi tourner indéfiniment.

Tester maintenant le programme suivant :

```
1 from microbit import *
2
3 while True :
4     if button_a.is_pressed() :
5         display.show("A")
6     elif button_b.is_pressed() :
7         display.show("B")
8     else :
9         display.show("X")
```

Exercice 5 : SAD or HAPPY ?

Écrire un programme qui affichera :

- l'image « HAPPY » lorsque le bouton A est pressé,
- l'image « SAD » lorsque le bouton B est pressé.

Exercice 6 : un peu plus dynamique

Nous aimerions maintenant que les images de l'exercice précédent ne s'affichent que pendant 500 millisecondes, puis s'effacent.

Reprendre le programme précédent et y ajouter, aux endroits opportuns, les instructions suivantes pour obtenir le résultat souhaité :

- ```
  1 sleep(500)
  ```

- ```
  1 display.clear()
  ```

4 Accéléromètre

Un accéléromètre est un capteur de mouvement. On en retrouve :

- sur les smartphones, pour détecter quand l'affichage doit basculer en mode portrait ou paysage,
- sur les consoles de jeux, pour détecter les mouvements du joueur
- sur les dispositifs d'airbags des voitures, pour détecter un changement brusque de vitesse,
- sur les dispositifs de mesures sismiques ...

La carte `micro:bit` est dotée d'un accéléromètre qui mesure les forces en 3 dimensions, y compris la gravité.

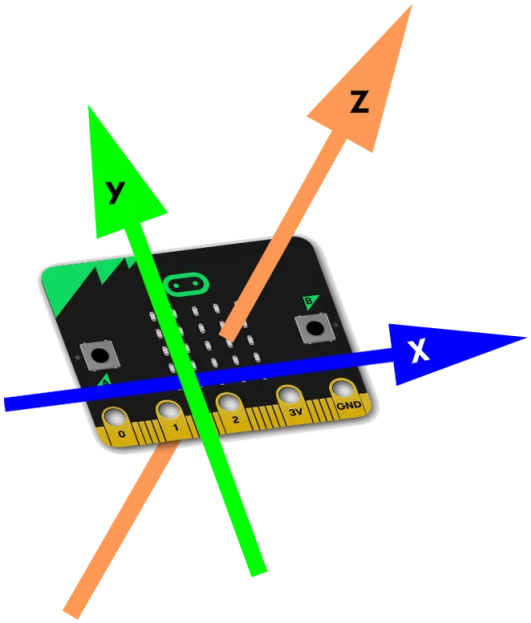


FIGURE 1.3 – Accéléromètre

Exercice 7 : détecteur de secousses

Sur un sismographe, les capteurs enregistrant la vitesse du sol sont appelés des vélocimètres, et ceux enregistrant l'accélération, des accéléromètres. L'accéléromètre est ainsi l'un des composants essentiels du sismographe.

Nous allons ici programmer une version très rudimentaire du sismographe.

1. Écrire puis téléverser sur la carte le programme suivant :

```
1 from microbit import *
2
3 while True :
4     if accelerometer.was_gesture("shake") :
5         display.show(Image.ANGRY)
6         sleep(1000)
7     display.clear()
```

2. Secouer maintenant la carte et observer le résultat.

Exercice 8 : compteur de pas, ou compteur de shakes

L'accéléromètre de la carte peut être utilisé pour créer un compteur de pas. Cela supposerait de brancher la carte sur piles (pour pouvoir marcher), et de l'attacher à notre jambe.

Pour plus de facilité dans son usage en classe, nous allons plutôt l'utiliser comme un compteur de « shakes » (le programme reste le même).

Écrire un programme qui :

- commencera par initialiser une variable `step` à la valeur de départ 0,
- à chaque secousse (`shake`), augmenter de 1 la valeur de `step`, puis afficher sa nouvelle valeur.

❏ Pour un aperçu plus large des fonctionnalités de l'accéléromètre :

https://microbit-micropython.readthedocs.io/en/latest/accelerometer.html

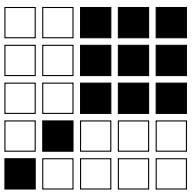
5 Créer ses propres images

Plutôt que d'utiliser les images déjà disponibles, nous pouvons souhaiter créer nos propres images à afficher sur les 25 LED de la `micro:bit`. Il faut alors indiquer, pour chacune des 25 LED une valeur de luminosité entre 0 et 9 :

- 0 : LED éteinte,
- 9 : luminosité maximale.

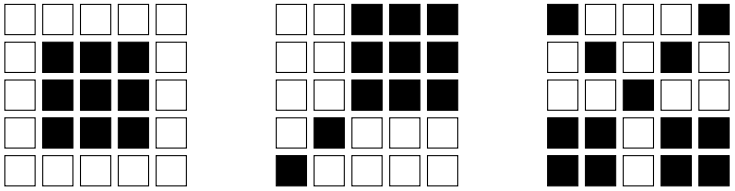
Voici par exemple un programme permettant d'afficher une feuille d'arbre (rudimentaire) :

```
1 from microbit import *
2
3 feuille = Image("00999:"
4                 "00999:"
5                 "00999:"
6                 "09000:"
7                 "90000")
8
9 display.show(feuille)
```



Exercice 9 : images de pierres, feuilles et ciseaux

1. Créer de mêmes des images représentant un pierre et une paire de ciseaux.
2. Écrire un programme affichant alternativement et indéfiniment les trois images.



6 Mini - Projet, niveau initiation

❏ Ce mini-projet consiste à réaliser une « **carte à jouer à pierre - feuille - ciseaux** ».

❏ Il se réalise à deux :
chacun des deux équipiers réalisera le même programme sur sa carte afin de pouvoir jouer ensuite une partie de pierre - feuille - ciseaux avec son équipier.

1. Écrire un programme qui répète indéfiniment le programme suivant :
lorsque la carte sera secouée :

- tirer au hasard un nombre parmi 0, 1 et 2,
- afficher en conséquence pendant 2000 millisecondes pierre, feuille ou ciseaux,
- effacer l'affichage

2. Jouer maintenant une partie de pierre - feuille - ciseaux avec votre équipier.

Indication : tirer un nombre au hasard entre 0, 1 et 2 :

- En début de programme, importer la fonction `randint` depuis la librairie `random` :

```
1 from random import randint
```

- On peut alors tirer un nombre au hasard entre 0, 1, ou 2, à l'aide de l'instruction :

```
1 nombre = randint(0, 2)
```

Chapitre 2

Communication entre 2 cartes par ondes radio

Plan du T.P.

Retour au sommaire

FIGURE 2.1 – Communication par ondes radio

1 Ondes radio

Une onde radioélectrique, communément abrégée en onde radio, est une onde électromagnétique dont la fréquence est inférieure à 300 GHz. Adaptées au transport de signaux issus de la voix et de l'image, les ondes radio permettent les radiocommunications (talkie-walkies, téléphone sans fil, téléphonie mobile, télévision hertzienne . . .).

2 Envoyer et recevoir des messages

La carte `micro:bit` est dotée d'une antenne radio lui permettant de recevoir et d'envoyer des messages par ondes radio. Nous pouvons ainsi concevoir un microcontrôleur faisant intervenir plusieurs composants, chacun doté d'une carte.

C'est le cas par exemple d'un système d'ouverture à distance des voitures : l'**émetteur** situé sur les clés communique avec un **récepteur** dans la voiture qui commande le déverrouillage du véhicule.

Les exercices de ce chapitre se traitent en **binôme**, avec **deux cartes** :

- un carte émettrice,
- un carte réceptrice.

Nous verrons progressivement, que pour plus d'interactivité, les deux cartes peuvent devenir toutes deux émettrices et réceptrices.

Exercice 10 : message binaire

Sur un ordinateur, toute information (nombre, texte, image, son ...) est représenté en BINAIRE, uniquement avec des 0 et des 1.

Nous remplacerons ici les `0` et `1` par `A` et `B`.

1. Commencer par demander à votre enseignant de vous attribuer un canal de communication.
2. Saisir et téléverser les programmes suivants :
   (en remplaçant `group = 5` par le canal qui vous a été attribué).

Carte émettrice

```
1  ########### 1. Importations ###########
2  from microbit import *
3  import radio
4
5  ############ 2. Fonctions #############
6  def envoyer(message) :
7      radio.send(message)
8
9
10
11 ############### 3. Main ###############
12 radio.config(group = 5)
13 radio.on()
14 while True :
15     if button_a.is_pressed() :
16         envoyer("A")
17     elif button_b.is_pressed() :
18         envoyer("B")
```

Carte réceptrice

```
1  ########### 1. Importations ###########
2  from microbit import *
3  import radio
4
5  ############ 2. Fonctions #############
6  def recevoir() :
7      message = radio.receive()
8      if message :
9          display.show(message)
10
11 ############### 3. Main ###############
12 radio.config(group = 5)
13 radio.on()
14 while True :
15     recevoir()
```

3. Tester à l'aide des cartes et préciser la fonction de ces programmes.
4. Afin de pouvoir traiter les exercices suivants, prenons le temps de comprendre certains aspects nouveaux :

❑ **Utilisation des fonctions :**

comme cela a été vu dans le cours de mathématiques, les fonctions permettent de ne pas recopier plusieurs les mêmes portions de code pour réaliser des tâches analogues.

Nous ne souhaitons recopier `radio.send(...)` pour tous les messages possibles, nous avons donc écrit une fonction. Si le gain est pour l'instant faible (3 caractères de moins dans `envoyer("A")` que dans `radio.send("A")`), cela va devenir beaucoup plus intéressant lorsque nos fonctions vont grossir.

❑ **Décomposition du programme en 3 parties :**

nos programmes deviennent plus conséquents. Il est alors essentiel de leurs donner une structure claire, pour éviter d'arriver à un « programme fourre-tout », difficile à relire et debugger.

Nos programmes comporteront désormais **trois parties** :

- **1. Importations** :
  où nous réaliserons les importations des modules nécessaires.
- **2. Fonctions** :
  où nous définirons les fonctions nécessaires au programme.
- **3. Main** :
  c'est le corps principal du programme qui correspond au scénario du programme. C'est d'ici que seront réalisés les appels de fonctions.

❑ **Configuration de la communication radio :**

elle se fait en trois temps :

- **a. Importation du module radio :**

```
1 import radio
```

- **b. Définition du canal de communication :**

```
1 radio.config(group = 5)
```

pour éviter des interférences entre les différentes cartes présentes dans la classe, il faut attribuer à chaque paire de cartes un canal différent des autres binômes.

- **c. Allumer l'antenne radio :**

```
1 radio.on()
```

❑ **Réception des messages :**

- pour récupérer les messages reçus par l'antenne radio, nous utilisons l'instruction :

```
1 message = radio.receive()
```

- Si aucun message n'a été reçu par l'antenne, la variable `message` se voit attribuer la valeur `None`, qui signifie `rien` (ou `vide`) en PYTHON.
- Comment comprendre l'instruction :

```
1 if message :
2     display.show(message)
```

Lorsque la valeur `None` est utilisée dans une condition logique après un `if`, elle est assimilée à `False`.

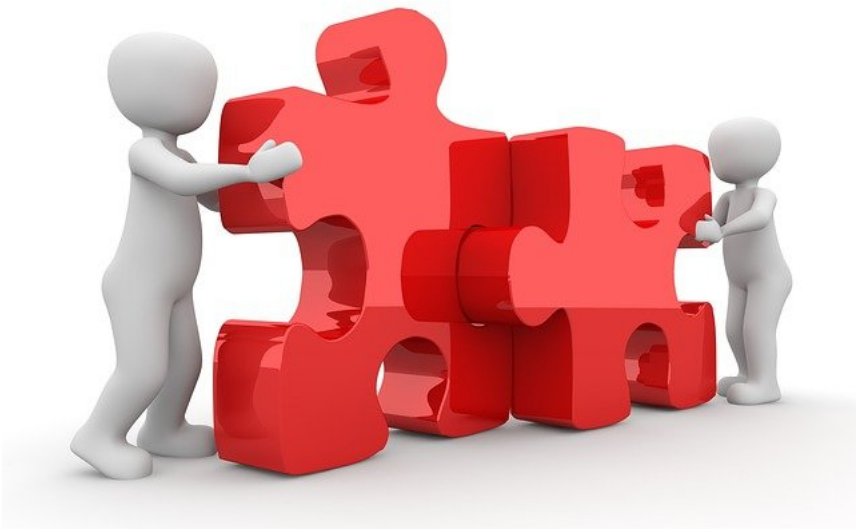
À l'inverse, toute chaîne de caractères non vide est assimilée à `True`.

Vous pouvez vous en convaincre en testant dans une console PYTHON les instructions suivantes :

```
1 >>> if None :
2         print("vu")
3 ...
4 >>>
```

```
1 >>> if "abc" :
2         print("vu")
3 ...
4 vu
5 >>>
```

3 Mini - Projet, niveau entraînement



Ping pong

Nous souhaitons ici pouvoir jouer une partie de ping pong entre les deux cartes :

❏ Commencer par choisir, à l'adresse suivante, une image qui fera office de balle :
https://microbit-micropython.readthedocs.io/en/latest/tutorials/images.html

❏ Lorsqu'une carte est secouée :

- son écran est effacé,
- l'image s'affiche sur l'autre carte.

En téléversant ce même programme sur les deux cartes, nous pourrons alors jouer une partie de ping pong numérique !

Chapitre 3

Le robot Maqueen

Plan du T.P.

Retour au sommaire

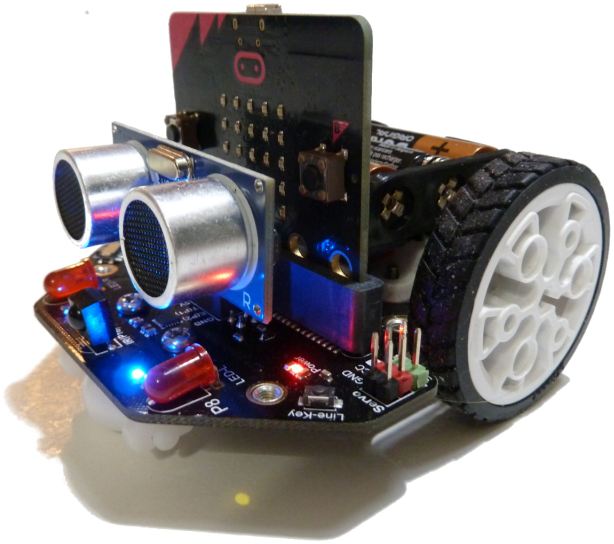


FIGURE 3.1 – Le robot Maqueen

1 Présentation

Le robot `Maqueen` est contrôlé à l'aide d'une carte `micro:bit`.
Parmi ses nombreuses fonctionnalités, on trouve :

- des moteurs à engrenage (pour se déplacer),
- un capteur de distance ultrason (pour détecter les obstacles),
- des capteurs permettant le suivi d'une ligne,
- des LED,
- un haut-parleur,
- un capteur infrarouge permettant de télécommander le robot.

Pour le programmer, nous utiliserons le module à télécharger développé par Olivier Lécluse :

- télécharger le module,
- extraire l'archive,
- placer le fichier `maqueen.py` dans votre répertoire de travail `micro:bit`.

Ce fichier devra être joint à tous vos programmes destinés au robot.

2 Méthodes fournies par le module `maqueen.py`

- **avance(vitesse)** : avancer en ligne droite,
  - — `vitesse` : nombre entre 0 et 100,
  - — ce paramètre est optionnel,
  - — s'il n'est pas spécifié, la dernière vitesse spécifiée par `avance()` ou `setVitesse()` est utilisée.
- **recule()** : faire marche arrière.
- **stop()** : stopper les moteurs.
- **moteurDroit(vitesse)** : faire tourner la roue droite.
- **moteurGauche(vitesse)** : faire tourner la roue gauche.
- **getVitesse()** : renvoyer la vitesse paramétrée par `setVitesse()` ou `avance()`.
- **setVitesse()** : changer la valeur de la vitesse utilisée par `avance`, `recule` ou `moteur`.
- **distance()** : renvoyer la distance en cm lue par le capteur ultrason.
- **son_r2d2()** et **son_bip()** : effets sonores.

3 Premier programme

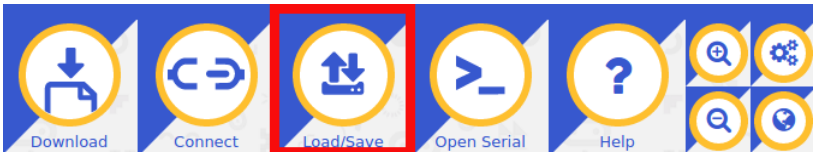
Nous souhaitons ici écrire un programme permettant au robot de déplacer dans un pièce rectangulaire (par exemple) en changeant de direction à chaque fois qu'il détecte un obstacle en face.
C'est, en quelques sortes, une version simplifiée du programme qui dirige un robot aspirateur ou un robot nettoyeur de piscine.

```
1  ############# 1. Importations #############
2  from microbit import *
3  from maqueen import Maqueen
4
5  ############## 2. Fonctions ###############
6  def explore() :
7      mq.avance()
8      d = mq.distance()
9      if d < 20 :
10         mq.son_r2d2()
11         mq.recule()
12         sleep(2000)
13         mq.moteurDroit(0)        # On stoppe l'un des deux moteurs pour tourner
14         sleep(1000)
15
16 ################ 3. Main ##################
17 mq = Maqueen()                   # Creation d'un objet robot Maqueen
18 mq.setVitesse(50)
19 while True :
20     explore()
```

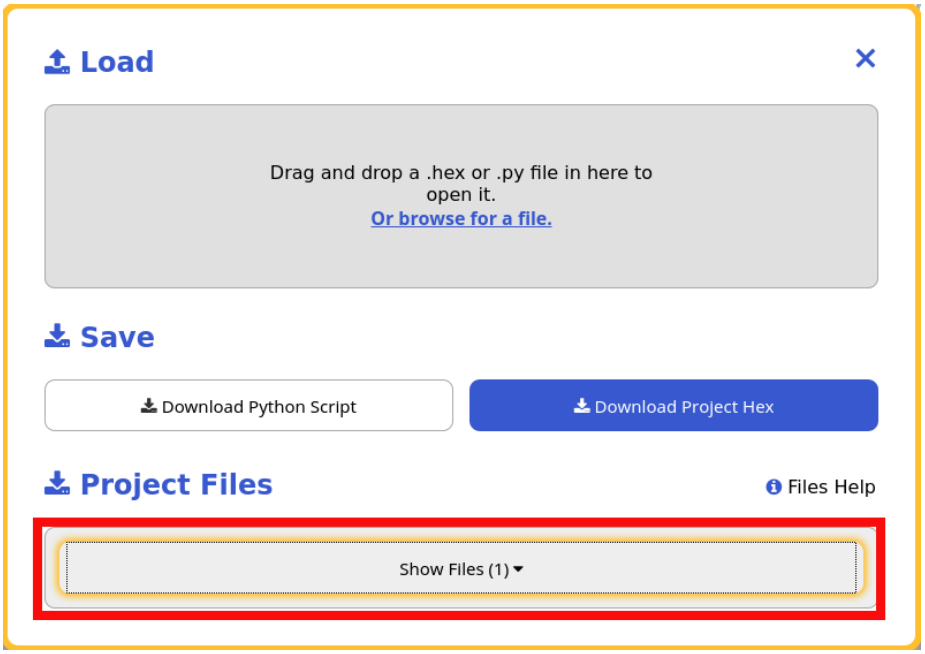
4 Transfert du module et du programme sur la carte

La procédure est différente de celle utilisée dans le chapitre précédent :

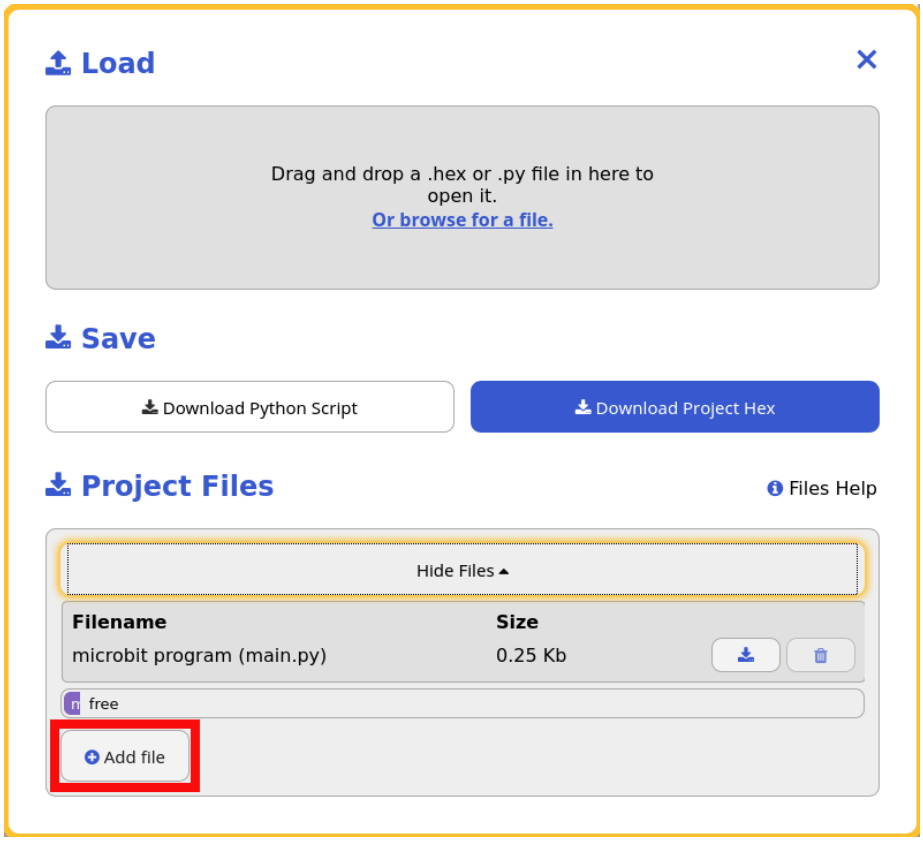
1. Saisir le programme ci-dessus dans l'éditeur en ligne,
2. **Load/Save** :



3. Déplier le menu « **Show Files** » :



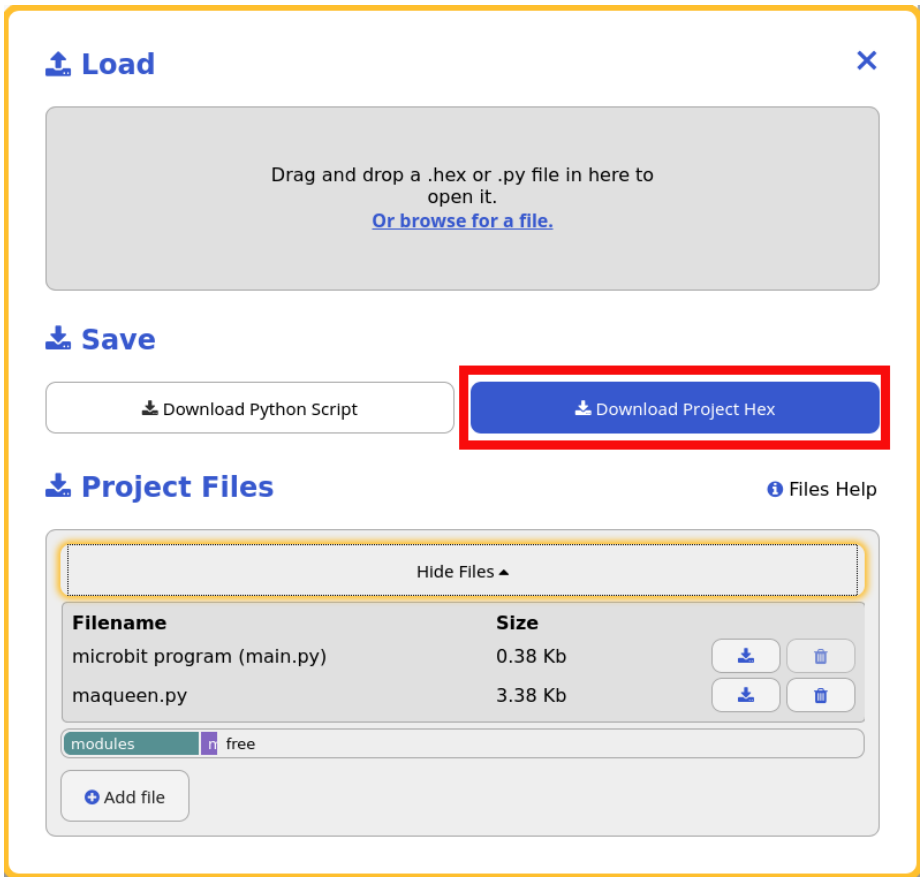
4. **Add File** :



5. Dans la fenêtre de navigation qui s'ouvre, sélectionner le fichier `maqueen.py`, puis ouvrir. Vous devriez maintenant voir apparaître deux fichier PYTHON :

   - `microbit program (main.py)` : le programme que vous avez saisi dans l'éditeur,
   - `maqueen.py` : le module `maqueen` que nous venons de joindre au projet.

6. Nous pouvons maintenant téléverser l'ensemble sur la carte : **Download Project Hex** :



7. Attendre la fin du clignotement de la diode située sur la face arrière de la carte.

8. Assembler le robot `Maquenn` tel que sur l'image ci-dessous.

9. Placer le robot au sol et l'allumer à l'aide l'interrupteur placé à l'arrière entre les deux roues.

5 Applications

Tout ceci peut sembler plaisant, mais futile. Aucunement !
On retrouve des robots de ce type, en versions infiniment plus sophistiquées, dans :

- les robots d'exploration lunaire,
- les voitures autonomes,
- les robots aspirateurs, tondeuses à gazon, nettoyeurs de piscines,
- les robots d'intervention sur sites radioactifs ...

6 Mini - Projet, niveau approfondissement

À vous de définir votre sujet de projet et de vous documenter sur Internet pour le réaliser. Par exemple :

- un robot téléguidé par une seconde carte `micro:bit`,
- un robot faisant le tour de la pièce en longeant les murs,
- un robot suiveur de ligne ...

Chapitre 4

Liens

Retour au sommaire

Quelques liens sur la carte `micro:bit`

- **Éditeur en ligne :**
  https://python.microbit.org/v/2
- **Aperçu des différentes fonctionnalités :**
  https://microbit.org/fr/get-started/user-guide/overview/
- **Programmer la carte en Python :**
  https://microbit-micropython.readthedocs.io/en/latest/index.html
- **Un cours de SNT pour découvrir d'autres fonctionnalités :**
  https://frederic-junier.org/SNT/Theme3_InformatiqueEmbarquee/ressources/Activite1-Microbit-2019V1.p
- **Documentation du module Maqueen :**
  https://lecluseo.scenari-community.org/CircuitPython/co/g_maqueen.html
- **Des idées de projets :**
  https://lecluseo.scenari-community.org/CircuitPython/co/AA_projets.html
- **Tutoriels et idées de projets :**
  https://fr.padlet.com/yeswecode/zhfhpca02kttmfly
- **Fondation CGénial (qui a offert les cartes que nous utilisons) :**
  https://www.cgenial.org/