



# Les objets connectés pour enseigner l’algorithmique en lycée professionnel

Groupe InEFLP



de l’IREM\* de Marseille

3 octobre 2021

---

\*Institut de Recherche sur l’Enseignement des Mathématiques

---

# Table des matières



<b>Mémo : les blocs principaux Micro:bit</b>	<b>5</b>
--	----------



<b>Ton premier programme avec Micro:bit</b>	<b>8</b>
Description .....	8



<b>Commande de base pour Micro:bit (et python)</b>	<b>11</b>
Description .....	11
Commandes essentielles .....	11
Des commandes pour aller plus loin .....	19



<b>Superviser le travail des élèves sur Micro:bit</b>	<b>25</b>
Description .....	25
Mode d'emploi de Classroom .....	26



<b>Borne de satisfaction avec Micro:bit</b>	<b>30</b>
Description .....	30
Niveau initiation - Premier modèle .....	31
Niveau expert - Cumuler les votes .....	33



<b>Les fractions avec Micro:bit</b>	<b>35</b>
Description .....	35
Écrire une fraction .....	36
Afficher une fraction .....	37
Somme de fractions .....	38



<b>Dé à 6 faces avec Micro:bit</b>	<b>39</b>
Description .....	39
Niveau simple .....	40
Niveau intermédiaire .....	41



<b>Normal ou truqué ? avec Micro:bit</b>	<b>43</b>
Description .....	43
Activité .....	44



<b>Pile ou face avec Micro:bit</b>	<b>46</b>
Description . . . . .	46
Niveau simple . . . . .	47
Niveau intermédiaire . . . . .	49
Niveau expert . . . . .	51
	
<b>Fluctuation d'échantillonnage avec Micro:bit</b>	<b>53</b>
Description . . . . .	53
Niveau prise en main - Vérifier le modèle . . . . .	55
Niveau intermédiaire - Générer des échantillons . . . . .	57
Niveau avancée - Produire des données . . . . .	59
	
<b>Chutes et oscillations avec Micro:bit</b>	<b>61</b>
Description . . . . .	61
Niveau simple . . . . .	62
Niveau intermédiaire . . . . .	64
	
<b>Mémo Mu-Editor pour Micro:bit (et python)</b>	<b>66</b>
	
<b>Mémo Micro:bit (et python)</b>	<b>68</b>
	
<b>Simuler un dé avec Micro:bit (et python)</b>	<b>71</b>
Description . . . . .	71
Niveau initiation . . . . .	73
Niveau intermédiaire . . . . .	75
Niveau avancé . . . . .	77
	
<b>Pile ou face communiquant avec Micro:bit (et python)</b>	<b>79</b>
Description . . . . .	79
Niveau simple . . . . .	80
Niveau Expert . . . . .	82
	
<b>Robot Maqueen et capteur de distance</b>	<b>84</b>
Description . . . . .	84
Niveau initiation - Utilisation du capteur de distance à ultrasons . . . . .	87
Niveau intermédiaire - Étalonnage du capteur . . . . .	88
Niveau expert - Robot autonome . . . . .	90
	
<b>Déterminer une vitesse avec MBot</b>	<b>91</b>
Description . . . . .	91
Niveau initiation - Programmer le déplacement de MBot . . . . .	92

---

Niveau intermédiaire - Déterminer la vitesse de MBot	93
--	----



<b>Vitesse de rotation avec MBot</b>	<b>94</b>
Description	94
Niveau initiation	95



<b>Variation de vitesses avec MBot</b>	<b>96</b>
Description	96
Niveau initiation - Nature du mouvement	97
Niveau expert - Mouvement décéléré	98



<b>Normal ou truqué ? avec STM32 Éducation</b>	<b>99</b>
Description	99
Activité	100



<b>Pile ou face avec STM32 Éducation</b>	<b>102</b>
Description	102
Progression proposée	103
Niveau initiation - Premier modèle	104
Niveau intermédiaire - Pile ou Face	105
Notes pour l'enseignant	106
Niveau intermédiaire - 16 tirages	107
Niveau expert - Initialisation	109
Niveau expert - 32 tirages	111
Niveau expert - Enregistrer les effectifs	113



# Mémo : les blocs principaux Micro:bit

## Base

Dans cette catégorie, on trouve les deux blocs essentiels qui sont d'ailleurs présents par défaut à l'ouverture d'un nouveau projet.

- Le bloc "au démarrage" permet d'initialiser un programme, la séquence d'instruction qui y sera placée ne sera donc exécuté qu'une seule fois.
- Le bloc "toujours" contient la séquence d'instruction qui sera perpétuellement exécutée par le processeur.
- Outre ces blocs, vous trouverez dans cette catégorie les éléments permettant d'afficher des figures, du texte, d'effacer l'écran, de faire une pause, etc ...



## Entrées

La catégorie "Entrées" proposent notamment des blocs "Lorsque" qui fonctionnent de la même façon que le bloc "Toujours", à la différence que le Micro:bit est ici en perpétuelle attente d'un évènement, que ce soit l'appui sur un bouton, un geste ou l'activation d'une broche.

On y trouve aussi les blocs permettant de recueillir les données :

- savoir si un bouton est pressé ;
- connaître la mesure de l'accélération ;
- relever la température (du processeur) ;



## Radio

Les fonctions de communication du Micro:bit se trouvent dans cette catégorie.

Les données envoyées (nombre et/ou chaîne de caractères) ne sont pas "routées" : les reçoivent qui peut. Pour filtrer les réceptions il est possible de définir des groupes.

Les blocs de "réception" se comportent comme des blocs d'entrées et attendent perpétuellement un événement.





## C Boucles

Cette catégorie contient les blocs permettant de faire des boucles ils sont aux nombres de 4 :

- "répéter", qui est le plus simple et qui convient dans un grand nombre de cas ;
- "tant que", qui répète la séquence tant que la condition indiquée est vraie ;
- "pour" qui répète une séquence en incrémentant une valeur entière ou un index de liste ;



## X Logique

Dans la catégorie "Logique" nous allons trouver les blocs relatifs aux instructions conditionnelles, aux tests et au booléens

- Les blocs conditionnels permettent d'exécuter une suite d'instruction si le résultat d'un test donné est vrai. On modifie le nombres de conditions en cliquant sur + ou -.
- Les blocs de test permettent de comparer des nombres, des chaînes de caractères, des booléens...
- Les blocs booléens permettent d'effectuer des opérations logiques.



## Variables

Par défaut, cette catégorie est vide tant qu'aucune variable n'a été définie, ou qu'aucun bloc comprenant une variable par défaut n'a été utiliser.

Une variable peut bien sûr contenir un nombre, un booléen, une chaîne de caractères, une liste...

- Le bloc "définir" permet d'affecter une valeur à la variable.
- Le bloc "changer par" consiste à incrémenter la valeur de la variable



## Maths

Vous trouverez dans cette catégorie tous les blocs pour les opérations classiques, mais aussi les blocs permettant de générer de l'aléatoire, ou encore de contraindre ou mapper des valeurs.





## f(x) Fonctions

Pour accéder à cette catégorie, il faut dérouler le menu

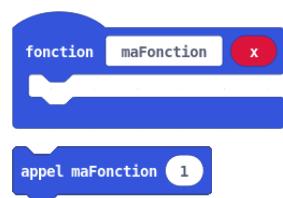
### ▼ Avancé

Tout comme pour la catégorie "Variables", il faut définir au préalable une fonction pour voir apparaître le bloc correspondant.

Une fonction peut être définie avec ou sans paramètres d'entrées.

Une fonction en bloc ne renvoie pas de valeur.

Une fois la fonction définie, on dispose d'un bloc pour l'appeler.





# Ton premier programme avec Micro:bit

## DESCRIPTION

### Objectif

Une minute top chrono !

Il vous faudra un peu plus de temps (mais à peine) pour réaliser ce premier programme, qui vous permettra de simuler un sablier. Ou alors d'égrainer le temps qui passe durant ce cours interminable.

Voilà de bons prétextes pour découvrir les possibilités offertes par Micro:bit !

### Intérêt

découvrir l'**interface de programmation** et élaborer son premier programme ;

découvrir quelques **fonctions** pour avoir envie d'en savoir plus ;

programmer un **objet** très facilement.

### Matériel



- 1 x Micro:bit (mais un accès internet peut suffir)
- 1 x accès internet : IDE programmation par bloc <http://makecode.microbit.org/>

### Pour commencer

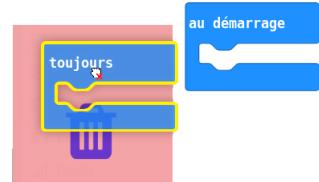
- Se munir d'un Micro:bit et de son cable usb ;
- Brancher un Micro:bit sur l'ordinateur ;
- Ouvrir un navigateur internet récent et aller sur la page citée au-dessus.



- Cliquer sur Nouveau projet

### Prise en main

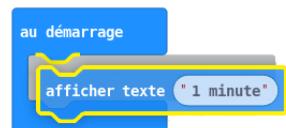
Lors du démarrage d'un nouveau projet, l'interface place par défaut dans l'espace de travail les blocs **au démarrage** et **toujours**; or nous n'avons pas besoin de ce dernier pour notre projet. Pour s'en débarrasser il suffit de le saisir et le lâcher vers la gauche, au dessus du menu des catégories.





## DESCRIPTION

Pour construire le programme, il suffira par la suite de sélectionner les blocs adéquats parmi les catégories et de les enclencher pour créer des séquences d'instructions.



## Définition du programme

Nous voulons programmer un Micro:bit qui :

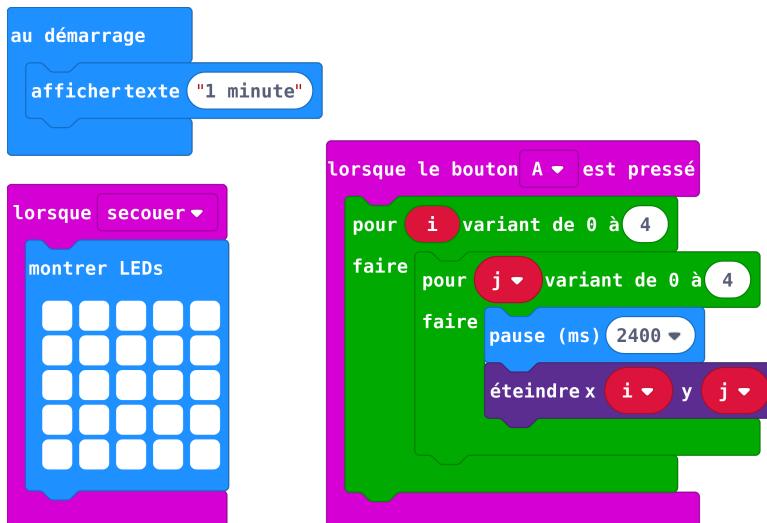
- allume toutes les LED lorsqu'on le secoue;
- éteigne les LED une par une à intervalle régulier;
- éteigne toutes les LED en 1 minute Micro:bit.

Nous allons trouver les blocs nécessaires dans les catégories suivantes :

Base	Pour l'initialisation et l'affichage.
Entrées	Pour l'interaction avec les boutons.
LED	Pour interagir avec chaque LED.
Boucles	Pour répéter des instructions.
Variables	Pour repérer les LED.

## Création du programme

Voilà une proposition de programme pour réaliser notre objet :



Il ne vous reste plus qu'à le télécharger. Le fichier sera compilé sous le nom microbit-XXXXX.hex et se retrouvera sur votre ordinateur dans votre dossier des téléchargements.

Pour flasher le Micro:bit avec, il suffit de déplacer ce fichier vers la carte, qui doit être reconnue comme un lecteur USB par votre gestionnaire de fichiers.

### REMARQUE

Appairer le périphérique

Si vous utilisez le navigateur Chrome ®, il est possible d'appairer le Micro:bit (protocole webUSB) afin de téléverser le code directement sur la carte ce qui évite d'avoir à télécharger le fichier compilé puis à le copier manuellement.

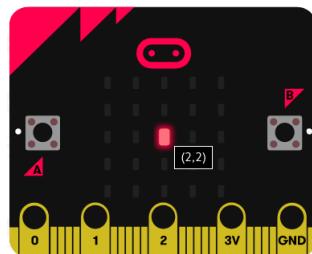


## Explication du programme

Dans ce programme, la principale difficulté consiste à éteindre les LED une par une.

Il faut savoir que les LED sont repérées par leur abscisse (x) et leur ordonnée (y).

La LED de coordonnées (0,0) est située en haut à gauche et la LED de coordonnées (4,4) est située en bas à droite.



Nous allons donc utiliser ces paramètres dans des boucles "pour" imbriquées. Une boucle va parcourir les valeurs de x, et l'autre les valeurs de y.

La boucle "pour" utilise une variable (par exemple "i" ou "index") qui est créée automatiquement lorsqu'on utilise ce bloc.



Nous avons besoin de deux variables : une pour les valeurs de x, l'autre pour les valeurs de y.

Il faudra donc en créer une nouvelle (par exemple "j") pour parcourir les valeurs de y.



Puisque la durée est fixée à 1 minutes, c'est à dire 60 000 millisecondes et qu'il y a 25 LED, avant chaque extinction de LED il faut faire une pause d'une durée de  $\frac{60000}{25} = 2400$  millisecondes.



# Commande de base pour Micro:bit (et python)

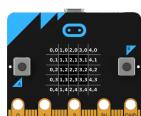
## DESCRIPTION

### Objectif

Il est possible de programmer la carte Micro:bit avec Python. Pour cela, il faut utiliser le module `Micro:bit`.

Dans cette fiche, vous découvrirez les commandes essentielles de ce module.

### Matériel



- 1 x Micro:bit
- 1 x IDE programmation python (Mu) : à télécharger/installer en <https://codewith.mu/>

## COMMANDES ESSENTIELLES



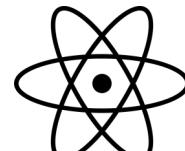
Durée



Public



Maths



Sciences



Algo  
affichage;  
bouton;  
événement.

### Le module `microbit`

En Python, les entrées/sorties de la carte Micro:bit ne sont pas *nativement* accessibles. Afin de pouvoir utiliser les fonctions prévues à cet effet, il faut appeler le module `microbit`.

Pour appeler ce module, il faut utiliser la commande classique `from ... import ...`

#### MÉTHODE

```
from microbit import *
```

### Astuce

La bibliothèque `microbit` est déjà téléchargée et préinstallée avec le logiciel `mu-editor`. Mais il faut tout de même l'appeler pour l'utiliser





## COMMANDES ESSENTIELLES

Le module `micropython` introduit des **classes** (avec `instances`, `méthodes` et `propriétés`) et des **modules** (avec `fonctions` et `constantes`).

Exemple de classes :

**Image** pour créer et manipuler les images. Les `propriétés` sont des images déjà enregistrées (comme les smileys)

**Button** avec deux `instances` `button_a` et `button_b` pour connaître l'état des boutons.

**Pin** avec différentes `instances` en fonction du type de broche (digitale, analogique ou de contact comme `pin0`, `pin1` et `pin2`).

Exemple de modules :

**display** pour gérer l'écran de LED

**accelerometer** pour interroger l'accéléromètre

**compass** pour manipuler et interroger la boussole

**music** pour créer et manipuler de la musique

**speech** pour faire parler le Micro:bit

**radio** pour communiquer entre Micro:bit via un protocole simple

### REMARQUE

Pour aller plus loin, vous pouvez consulter la page [Microbit Module](#) de la documentation officielle.

## Micropython dans la carte Micro:bit

Lorsque la carte Micro:bit est flashée avec le module `microbit`, elle contient un noyau micropython et peut donc exécuter du code python.

L'IDE Mu permet d'accéder au noyau micropython de la carte. Pour accéder au terminal de ce noyau, il suffit de cliquer sur l'icône REPL de l'application.

Il est alors tout à fait possible d'écrire du code dans ce terminal qui sera exécuté par la carte Micro:bit.

### REMARQUE

L'instruction `print()` permet d'écrire dans le terminal REPL.

### MÉTHODE

Pour utiliser le terminal de la carte Micro:bit :

- Flasher la carte Micro:bit avec micropython.
- Ouvrir le terminal de la carte en cliquant sur REPL
- saisir le code ci-dessous :

```
> print("Hello, World!")
```

### MÉTHODE

Le programme ci-dessous affiche 10 fois le même texte dans le terminal :

```
from microbit import *
for i in range(10):
    print("Hello, World!")
```

## Afficher un texte

Le module `display` permet de gérer l'écran de LED de Micro:bit.

La fonction `display.show("string")` permet d'afficher un texte.



## COMMANDES ESSENTIELLES

La fonction `display.scroll("string")` permet de le faire défiler.

### MÉTHODE

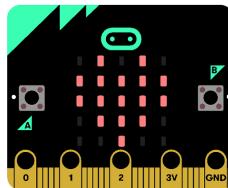
```
from microbit import *
display.scroll("Hello,")
display.show("World!")
```

### REMARQUE

Pour aller plus loin, vous pouvez consulter les pages [Introduction » Hello, World!](#) et [Display](#) de la documentation officielle.

## Des images

La classe `Image` contient comme propriétés de nombreuses images pré-programmées. Par exemple l'image du cœur : `Image.HEART`



Pour afficher une image, il faut utiliser la fonction `display.show()`. La fonction `display.clear()` permet d'effacer l'écran.

Pour afficher successivement des images, il faut mettre le Micro:bit en pause. Sinon l'utilisateur n'aura pas le temps de tout voir. La commande `sleep(nombreEntier)` arrête donc la carte pendant la durée (en milliseconde) indiquée.

### MÉTHODE

Le code Python ci-dessous affiche une image pendant une seconde puis affiche une deuxième image. Une seconde plus tard, l'écran s'efface.

```
from microbit import *
display.show(Image.HAPPY)
sleep(1000)
display.show(Image.ANGRY)
sleep(1000)
display.clear()
```

### REMARQUE

Pour aller plus loin, vous pouvez consulter les pages [Introduction » Images](#) ou [Image](#) de la documentation officielle.

## Des animations

Lorsque plusieurs images sont stockées dans un **tableau**, il est possible de les faire défiler à un rythme donné. Ceci permet de facilement créer une **animation**.

Pour faire défiler des images, utiliser l'option `delay=nombreEntier` de la fonction `display.show()`.



## COMMANDES ESSENTIELLES

Le module `microbit` contient **deux** tableaux d'images :

- `Image.ALL_CLOCKS` qui affiche la grande **aiguille d'une montre** sous différentes orientations
- `Image.ALL_ARROWS` qui affiche des **flèches** dans toutes les directions.

### MÉTHODE

Le code ci-dessous affiche toutes les 0,200 secondes l'aiguille d'une montre qui tourne :

```
from microbit import *
display.show(Image.ALL_CLOCKS, delay=200)
```

### REMARQUE

Pour aller plus loin, vous pouvez consulter la page [Introduction » Images](#) de la documentation officielle.

## Des boutons

Les états des boutons de la carte Micro:bit sont récupérables par le module `micropython`. Il existe pour cela deux classes : `button_a` et `button_b`.

Ces deux classes possèdent 3 méthodes :

- `.get_presses()` retourne le **nombre** de fois que le bouton a été pressé depuis le dernier appel de cette méthode
- `.is_pressed()` retourne un **booléen** qui indique si le bouton est **actuellement** pressé
- `.was_pressed()` retourne un **booléen** qui indique si le bouton **a été** pressé depuis le dernier appel de cette méthode.

### MÉTHODE

Le code ci-dessous affiche le nombre de fois que le bouton A a été pressé depuis le démarrage.

```
from microbit import *
sleep(10000)
display.scroll(str(button_a.get_presses()))
```

Pour programmer un Micro:bit, il est très souvent nécessaire de détecter un **événement** comme par exemple l'appui sur un bouton.

Une méthode simple consiste à créer une **boucle infinie** `while True:` contenant un test qui, à chaque itération de la boucle, vérifie la réalisation de l'événement attendu.

Lorsque l'événement se réalise, l'instruction `break` est appelée pour sortir de la boucle.

### MÉTHODE

Le code ci-dessous tourne en boucle.

- Lorsque aucun événement n'est détecté, c'est l'image triste `Image.SAD` qui s'affiche.
- Pendant que le bouton A est pressé, l'image joyeuse `Image.HAPPY` s'affiche.
- Pendant que la broche `pin1` est touchée (en même temps que la broche `GND`), l'image endormie `Image.ASLEEP` s'affiche.
- Lorsque le bouton B est pressé, l'écran s'efface et le programme quitte la boucle.



```
from microbit import *
while True:
    if button_a.is_pressed():
        display.show(Image.HAPPY)
    elif pin1.is_touched():
        display.show(Image.ASLEEP)
    elif button_b.is_pressed():
        display.clear()
        break
    else:
        display.show(Image.SAD)
```

**REMARQUE**

Les broches instanciées `pin0`, `pin1` et `pin2` peuvent aussi servir de bouton. La méthode `.is_touched()` permet de renvoyer un booléen qui devient vrai lorsqu'une personne en contact avec la masse (broche GND) touche puis relâche la broche en question.

En effet, lorsqu'une instance de ces broches est activée, la carte Micro:bit mesure la résistance entre cette broche et la masse (broche GND). Lorsque cette dernière varie et passe d'une valeur quasi infinie à une valeur faible, le test devient vrai. Cet évènement arrive lorsqu'une personne en contact avec GND touche la broche et la relâche.

**REMARQUE**

Pour aller plus loin, vous pouvez consulter les pages [Introduction » Boutons](#) et [Buttons](#) de la documentation officielle.

## Le hasard

Le module `random` est utilisable avec Micro:bit. Une fois importé, il est très simple de générer des nombres aléatoires avec les fonctions `random.random()` ou `random.randint(a,b)`.

**MÉTHODE**

Le programme ci-dessous affiche très rapidement 50 nombres aléatoires tirés entre 1 et 6. Le dernier nombre tiré est affiché pendant une seconde puis effacé.

```
from microbit import *
import random
for i in range(50):
    display.show(random.randint(1, 6))
    sleep(20)
sleep(1000)
display.clear()
```

**REMARQUE**

Pour aller plus loin, vous pouvez consulter les pages [Introduction » Hasard](#) et [Random Number Generation](#) de la documentation officielle.

## Le mouvement

L'accéléromètre du Micro:bit est accessible par le module `accelerometer`.





## COMMANDES ESSENTIELLES

Il est alors possible de récupérer une des coordonnées du vecteur accélération (avec une fonction du type `accelerometer.get_x()`) .

### MÉTHODE

Le programme ci-dessous affiche une flèche en fonction de l'inclinaison du Micro:bit.  
Le bouton B permet de quitter cette boucle.

```
from microbit import *
button_b.was_pressed()
while True:
    capteur = accelerometer.get_x()
    if capteur > 40:
        display.show(Image.ARROW_E)
    elif capteur < -40:
        display.show(Image.ARROW_W)
    else:
        display.show("-")
    if button_b.was_pressed():
        display.clear()
        break
```

### REMARQUE

Pour aller plus loin, vous pouvez consulter les pages [Introduction](#) » [Mouvement et Accéléromètre](#) de la documentation officielle.

## Les gestes

Le module `accelerometer` peut aussi détecter des mouvements ou des positions pré-programmés : les **gestes** ("up", "down", "left", "right", "face up", "face down", "freefall", "3g", "6g", "8g", "shake").

Il y a alors 3 fonctions qui s'applique sur le module `accelerometer` :

- `accelerometer.get_gestures()` retourne un **tuple** contenant l'historique des gestes. Le dernier élément du tuple est le geste le plus récent. Le tuple est réinitialisé à chaque appel de cette fonction.
- `accelerometer.is_gesture("nom")` retourne un **booléen** qui indique si le geste en cours est "`nom`" .
- `accelerometer.was_pressed("nom")` retourne un **booléen** qui indique si le geste "`nom`" a été pressé depuis le dernier appel de cette fonction.

### MÉTHODE

Le programme ci-dessous affiche le nombre "8"

Lorsque le Micro:bit est secoué, il s'affiche une seconde plus tard de manière aléatoire et équiprobable soit "Oui", soit "Non".

Le jeu recommence sauf si on appui sur le bouton B ce qui fait sortir de la boucle.



```
from microbit import *
import random
button_b.was_pressed()
while True:
    display.show("8")
    if accelerometer.was_gesture("shake"):
        display.clear()
        sleep(1000)
        display.scroll(random.choice(["Oui", "Non"]))
    if button_b.was_pressed():
        break
```

**REMARQUE**

Pour aller plus loin, vous pouvez consulter les pages [Introduction](#) » [Gestes](#) et [Accéléromètre](#) de la documentation officielle.

## La radio

Les cartes Micro:bit peuvent communiquer entre elles au moyen du module `radio`.

La fonction `radio.send("string")` permet d'envoyer par radio le texte `"string"`.

La fonction `radio.receive()` retourne la chaîne de caractère des données reçus par radio. Si rien n'a été reçu, la chaîne vaut `None`.

**MÉTHODE**

Le programme est à téléverser sur 2 cartes Micro:bit. Un appui sur les boutons A ou B de l'un des Micro:bit affiche les texte "A" ou "B" sur l'autre.

```
from microbit import *
import radio
import random

button_a.was_pressed()
button_b.was_pressed()

while True:
    # émetteur
    if button_a.was_pressed():
        radio.send("A")
    if button_b.was_pressed():
        radio.send("B")

    # récepteur
    incomming = radio.receive()
    if incomming == "A":
        display.scroll("A")
    if incomming == "B":
        display.scroll("B")

    # sortir de la boucle
    if pin0.is_touched():
        display.clear()
        break
    sleep(20)
```

**REMARQUE**

Pour aller plus loin, vous pouvez consulter les pages [Introduction » Radio et Radio](#) de la documentation officielle.

## La boussole

Pour interroger la boussole du Micro:bit, il faut utiliser le module `compass`.

Certaines fonctions comme `microbit.compass.get_x()` renvoient une composante du vecteur champ magnétique.

La fonction `microbit.compass.heading()` renvoie l'angle en degré entre l'orientation de la carte Micro:bit et le nord magnétique.

**REMARQUE**

Avant d'utiliser une fonction du module `compass`, il faut obligatoirement [calibrer](#) Micro:bit.

Pour cela, il est automatiquement demandé à l'utilisateur de bouger la carte dans différentes positions. Il faut que le point rouge qui clignote passe par toutes les LED de l'écran.

Pour programmer une calibration de la carte, il est possible d'utiliser la commande `compass.calibrate()`.

## MÉTHODE

Le programme ci-dessous fait office de boussole et indique le nord magnétique.

Attention, le capteur est sensible aux objets tels que téléphones, ordinateurs ou aux lieux tels que ascenseurs ou salle informatique...

```
from microbit import *
compass.calibrate()
button_b.was_pressed()
while True:
    cap = ((15 - compass.heading()) // 30) % 12
    display.show(Image.ALL_CLOCKS[cap])
    if button_b.was_pressed():
        display.clear()
        break
```

**REMARQUE**

Pour aller plus loin, vous pouvez consulter les pages [Introduction » Direction et Compass](#) de la documentation officielle.



## DES COMMANDES POUR ALLER PLUS LOIN



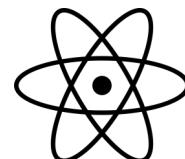
Durée



Public



Maths



Sciences

Algo  
affichage;  
bouton;  
événement;  
son

2 h

2de

### Représentation graphique avec Mu

Il est possible de tracer un **nuage de points** avec l'IDE Mu.

Pour cela, il faut

- utiliser l'outil Graphique de l'IDE Mu,
- faire afficher par le programme en cours d'exécution un tuple de nombres.

#### MÉTHODE

Le programme ci-dessous affiche dans le terminal (ou **dans le Graphique** si l'outil de l'IDE est cliqué) des fréquences d'apparitions lors d'une expérience aléatoire.

```
from microbit import *
import random
while True:
    if button_a.was_pressed():
        nb1 = 0
        total = 0
        for i in range(1000):
            # tirage aléatoire
            tirage = random.randint(0,1)
            total = total + 1
            # calcul d'effectifs
            nb1 = nb1 + tirage
            nb0 = total - nb1
            # affichage dans le terminal
            print((i, nb1/total, nb0/total))
```

### Des images personnalisées

Les images sont des objets qui peuvent être créées grâce au constructeur `Image("string")`.

Le texte `"string"` permet de décrire, **ligne par ligne**, la luminosité de chaque pixel de l'écran de LED. Pour chaque diode, la valeur varie de 0 (éteinte) à 9 (luminosité maximale). Pour séparer les lignes, il faut utiliser le symbole `": "`

#### MÉTHODE

Voici une image représentant un bateau avec la coque plus foncée que les 2 mâts :



```
from microbit import *
bateau = Image("05050: "
               "05050: "
               "05050: "
               "99999: "
               "09990")
display.show(bateau)
sleep(1000)
```

**REMARQUE**

Pour aller plus loin, vous pouvez consulter les pages [Introduction » Images](#) ou [Image](#) de la documentation officielle.

## Des animations personnalisées

En stockant des images personnalisées dans un tableau, il est possible de créer des [animations personnalisées](#).

**MÉTHODE**

Créer un image représentant un tableau et stocker cette image dans la variable `bateau1` :

```
from microbit import *
bateau1 = Image("05050: "
                "05050: "
                "05050: "
                "99999: "
                "09990")
```

Créer d'autres images (donnant l'illusion que le bateau s'enfonce) et les stocker dans de nouvelles variables :

```
bateau2 = Image("00000: "
                  "05050: "
                  "05050: "
                  "05050: "
                  "99999")
```

```
bateau3 = Image("00000: "
                  "00000: "
                  "05050: "
                  "05050: "
                  "05050")
```

```
bateau4 = Image("00000: "
                  "00000: "
                  "00000: "
                  "05050: "
                  "05050")
```

```
bateau5 = Image("00000: "
                  "00000: "
                  "00000: "
                  "00000: "
                  "05050")
```



```
bateau6 = Image("00000:"  
                 "00000:"  
                 "00000:"  
                 "00000:"  
                 "00000")
```

Créer un tableau regroupant toutes les images et afficher l'animation :

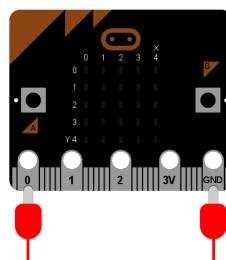
```
tous_bateaux = [bateau1, bateau2, bateau3, bateau4, bateau5, bateau6]  
display.show(tous_bateaux, delay=200)
```

### REMARQUE

Pour aller plus loin, vous pouvez consulter la page [Introduction » Images](#) de la documentation officielle.

## Faire parler Micro:bit

La carte Micro:bit peut aussi **parler**. Avant cela, il faut connecter un casque audio ou un (petit) haut-parleur à la carte.



Le module `speech` contient les fonctions permettant de gérer la **voix** de Micro:bit, dont la fonction `speech.say("string")` qui fera parler.

### MÉTHODE

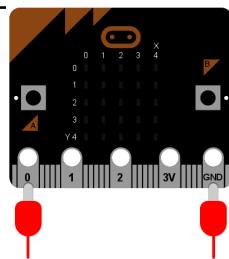
```
from microbit import *  
import speech  
speech.say("Hello, World")
```

### REMARQUE

Pour aller plus loin, vous pouvez consulter les pages [Introduction » Speech](#) ou [Speech](#) de la documentation officielle.

## De la musique

La carte Micro:bit est capable de jouer de la musique. Pour cela, il faut au préalable **connecter un casque audio** ou un petit haut-parleur à la carte.



Le module `music` contient les fonctions permettant de gérer et de contrôler le son. Parmi elles, la fonction `music.play()` joue une mélodie.

Par ailleurs, le module `music` contient aussi un grand nombre de mélodies enregistrées (ie. déjà programmées) comme par exemple :

- `music.DADADADUM`,
- `music.ENTERTAINER` ou encore
- `music.PRELUDE`.

### MÉTHODE

Voici comment jouer la musique `POWER_UP` avec Micro:bit :

```
from microbit import *
import music
music.play(music.POWER_UP)
```

La carte Micro:bit peut aussi générer un son en fonction de sa fréquence. Pour cela, il faut utiliser la fonction `music.pitch(freq, duree)` qui va générer pendant `duree` millisecondes le son de fréquence `freq`.

### MÉTHODE

Le code ci-dessous génère une sirène qui passe d'un son grave à un son aigu puis qui redevient grave.

Chaque fois que le son redevient grave, la boucle teste si le bouton B a été pressé. Si c'est le cas, le programme quitte la boucle.

```
from microbit import *
import music
button_b.was_pressed()
while True:
    for freq in range(880, 1760, 16):
        music.pitch(freq, 6)
    for freq in range(1760, 880, -16):
        music.pitch(freq, 6)
    if button_b.was_pressed():
        break
```

### MÉTHODE

Le programme ci-dessous génère un son dépendant de l'inclinaison du Micro:bit. Le bouton B permet de quitter cette boucle.



```
from microbit import *
import music
button_b.was_pressed()
while True:
    ton = max(0, accelerometer.get_y())
    music.pitch(ton, 10)
    if button_b.was_pressed():
        break
```

**REMARQUE**

Pour aller plus loin, vous pouvez consulter les pages [Introduction » Musique](#) ou [Music](#) de la documentation officielle.

## Les fichiers

La carte Micro:bit peut stocker des fichiers. Ces derniers ne sont pas accessibles par le système classique de fichiers, mais par l'intermédiaire de l'outil Fichier de l'IDE Mu.

Il est tout à fait possible d'utiliser Python pour lire, créer, modifier ou effacer un fichier.

**MÉTHODE**

Pour **lister les fichiers** présents sur la carte Micro:bit, il faut utiliser le module `os`.

La fonction `os.listdir()` renvoie alors une liste de chaîne de caractère. Chaque élément de la liste est un nom de fichier.

```
import os
listeFichiers = os.listdir()
```

**MÉTHODE**

Pour **lire le contenu** d'un fichier (qui doit exister), il faut :

- créer une instance de fichier en mode lecture '`r`'
- utiliser la méthode `.read()` pour copier le contenu du fichier dans une variable.

L'exemple ci-dessous lit le fichier `texte.txt`, copie son contenu dans la variable `txt` et l'afficher dans le terminal :

```
with open('texte.txt', 'r') as monFichier:
    txt = monFichier.read()
print(txt)
```

**MÉTHODE**

Pour **écrire un texte** dans un fichier (qui peut ne pas exister), il faut :

- créer une instance de fichier en mode écriture '`w`'
- utiliser la méthode `.write("string")` pour écrire le texte dans le fichier (attention, si le fichier existait déjà, cela supprimera tout ce que le fichier contenait).



L'exemple ci-dessous ouvre le fichier `texte.txt` en écriture puis écrase son contenu avec le contenu de la variable `txt`.

```
txt = "1,2,3\n4,5,6"
with open('texte.txt', 'w') as monFichier:
    monFichier.write(txt)
```

## MÉTHODE

Le programme ci-dessous effectue 50 tirages aléatoires qu'il enregistre dans le fichier `save.txt`.

- importer tous les modules nécessaires
- sauvegarder les 50 tirages dans la variable `txt`
- établir la liste des fichiers
- si le fichier `save.txt` existe, lire le fichier et sauvegarder ses données dans la variable `contenu`
- ouvrir le fichier en mode écriture et l'écraser avec le contenu mis à jour.

```
# sauvegarder 50 nb aléatoires
from microbit import *
import random
import os

txt = ""
for i in range(50):
    nb = random.randint(1, 6)
    txt = txt + str(nb)
    display.show(nb)
    sleep(20)

# connaître la liste des fichiers sur MB
listeFichiers = os.listdir()

# si le fichier 'save.txt' existe, récupérer le contenu
if ('save.txt' in listeFichiers):
    with open('save.txt', 'r') as monFichier:
        contenu = monFichier.read()
else:
    contenu = ""

with open('save.txt', 'w') as monFichier:
    contenu = contenu + txt + "\n"
    print(contenu)
    monFichier.write(contenu)
```

### REMARQUE

Pour aller plus loin, vous pouvez consulter les pages [Introduction » Storage](#) ou [Local Persistent File System](#) de la documentation officielle.



# Superviser le travail des élèves sur Micro:bit

## DESCRIPTION

### Objectif

Le suivi individuel des élèves est un critère majeur de réussite d'une activité. L'interface **micro:bit | classroom** va permettre de superviser l'avancement de chaque élève, afin de lui proposer un travail en rapport avec ses capacités.

### Intérêt

Micro:bit | classroom est un espace de programmation partagé entre les élèves et leur enseignant. Cet outil en ligne permet de :

- proposer** une situation initiale en bloc ou en python ;
- suivre** en direct le travail des élèves ;
- partager** du code entre élèves de façon globale ou ciblée ;
- sauvegarder** une séance de travail ;
- reprendre** une séance là où elle s'était arrêtée.

### Matériel



- 1 x Micro:bit (facultatif car le simulateur peut suffire)
- 1 x accès internet : <http://classroom.microbit.org/>
- navigateur Chrome®

#### REMARQUE

Le parti pris de l'application est la **simplicité** avec ses avantages et ses inconvénients.

#### Avantages :

- Pas de compte utilisateurs élève/enseignant à créer.
- Fichier unique de sauvegarde/récupération pour reprendre la session plus tard (format .html).
- Compte rendu de la session téléchargeable (format texte .docx).

#### Inconvénients :

- L'identification des élèves repose sur la confiance (possibilité d'usurpation d'identité lors de la reprise d'une activité).
- Pas d'éditeur de texte pour échanger avec les élèves.

#### REMARQUE

Attention la version présentée ici est la version beta de classroom. Les menus encore en anglais sont amenés à évoluer (ou pas...)



# MODE D'EMPLOI DE CLASSROOM

## Créer une Classroom

### MÉTHODE

Ouvrir le navigateur Chrome et rejoindre le site <http://classroom.microbit.org/>. Il suffit de donner le nom de l'activité, le type d'environnement de programmation (bloc ou python), et c'est parti.

### Set up your classroom

Begin your classroom set-up below. Once you've launched your classroom session you will have the option to add your own code to the classroom editor to share with your students.

#### Name your activity

pile\_ou\_face

#### Choose a programming language

MakeCode     Python

#### Select storage setting

Use temporary local storage (?)

Launch classroom

### REMARQUE

Il est possible de choisir si l'on souhaite autoriser l'application à enregistrer des données localement (dans le "local storage" du navigateur), à éviter si l'ordinateur est partagé mais utile si l'on craint de perdre son travail.

## Proposer un programme

### MÉTHODE

Dans le bandeau supérieur on trouve le menu de ce qui correspond aux 4 étapes de la vie d'une session :

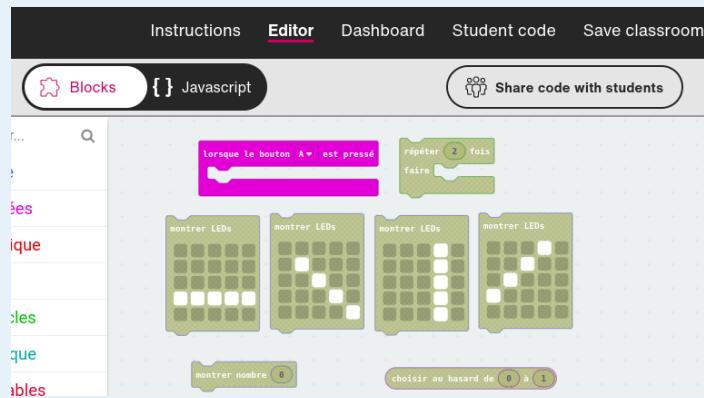
- 1) Création du code dans l'éditeur.
- 2) Partage de la session.
- 3) Supervision des travaux.
- 4) Sauvegarde et clôture d'une session.

Instructions    Editor    Dashboard    Student code    Save classroom





Il faut donc commencer par ouvrir l'éditeur (ici pour la programmation en blocs) et écrire un programme ou un début de programme (par exemple celui proposé par la fiche "pile ou face").



Une fois le code terminé, il ne faut pas oublier de le partager en cliquant sur le bouton "Share code with students".

### REMARQUE

L'éditeur comporte les mêmes fonctionnalités que sur <http://makecode.microbit.org>, il est donc possible de :

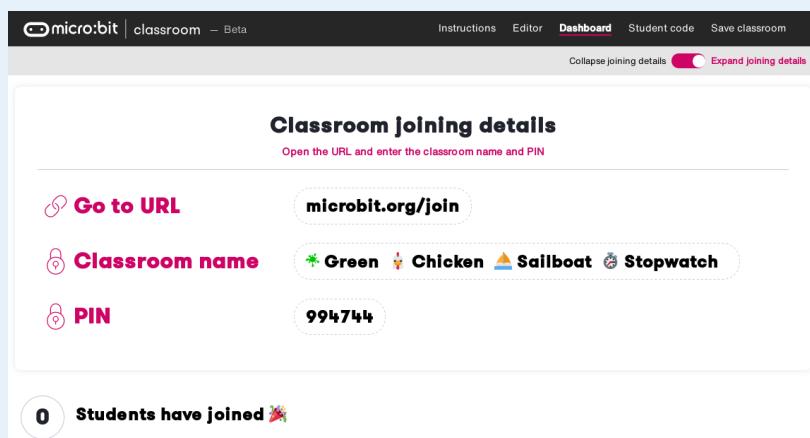
- télécharger le code.
- appairer la carte Micro:bit avec le navigateur.
- ouvrir un programme compilé sous la forme d'un fichier .hex en faisant glisser ce fichier dans la fenêtre de classroom.

Le code peut être modifié ultérieurement pour être distribué à toute la classe ou de façon ciblée, ce qui permet un travail de différenciation.

## Partager la session

### MÉTHODE

Pour se connecter à la session, les élèves doivent aller sur le site : <http://microbit.org/join> renseigner le nom de la classe (l'occasion de réviser son vocabulaire anglais) et rentrer le code PIN.





### REMARQUE

Une fois connectés, les élèves renseigneront leur nom tel qu'il apparaîtra sur votre écran, mais aussi tel qu'il sera enregistré dans le fichier de sauvegarde et de compte-rendu, attention aux noms fantaisistes.

## Superviser les travaux

### MÉTHODE

Dans cette fenêtre, la liste des élèves apparaît à gauche avec l'indication de l'état de son travail : en cours ou terminé (avec en prime une autoévaluation sous la forme de smiley).

Dans la partie droite apparaît le travail de l'élève dont les modifications sont instantanément reportées. Là encore on retrouve le bouton "Share code with students" qui permet d'envoyer à tout ou une partie de la classe le code produit par un élève.

The screenshot shows the 'Student code' section of the micro:bit classroom interface. At the top, there are tabs for 'Instructions', 'Editor', 'Dashboard', 'Student code' (which is selected), and 'Save classroom'. Below this, a heading says 'Student code' with a sub-instruction: 'See each of your students' code live, send code to another student and download a report with all the students' code'. A 'Download report for all students as Word document' button is visible. On the left, a 'Your students' panel lists 'Bob' (In progress) and 'Alice' (Completed). On the right, Bob's code is displayed in a block-based programming language:

```

lorsque le bouton A est pressé
  répéter (2) fois
    faire montrer LEDs
  
```

### REMARQUE

Pour l'éditeur de programmation par bloc, seul les blocs actifs côté élève apparaîtront dans la fenêtre de supervision.



## Sauvegarder une session

### MÉTHODE

Cette dernière étape de la vie d'une session vous permettra de récupérer un fichier html contenant toutes les informations nécessaires pour reprendre le travail ultérieurement. Le fait de clôturer la session, rendra inactive la connexion.

**Save classroom file**

Download classroom file to resume session later

- 1** **Download classroom file**  
Download this file to your computer to save all of your students' work. Use this file to start your next classroom session and resume the activity where your students left off.  
Students' work is not stored online by micro:bit classroom.
- 2** **End session**  
Once you have downloaded the classroom file and saved it for future use, click the end session button to delete any data that has been saved in temporary local storage by micro:bit classroom.
- 3** **Resume classroom**  
Start a new classroom session using the classroom file you have downloaded. Open the file and click the resume classroom session button to launch the resumed activity in your browser.

Pour reprendre le travail, il suffit d'ouvrir fichier .html dans le navigateur et l'on retrouvera le dernier programme proposé ainsi que les travaux des élèves. Ils pourront dès lors de nouveau se connecter, et sélectionner leur nom dans un menu déroulant afin de retrouver leur travail.

### REMARQUE

Attention au paramétrage du navigateur, si celui est trop strict sur la gestion des données personnelles vous pourriez être dans l'impossibilité de réouvrir correctement une session.



# Borne de satisfaction avec Micro:bit

## DESCRIPTION

### Objectif

Le but de ce projet est de fabriquer une borne de vote pour un questionnaire de satisfaction. Une question est posée et la réponse est donnée sur une **échelle de trois valeurs**. Par exemple, la question pourra être

**Journée Portes Ouvertes** : votre avis nous intéresse !  
Que pensez vous de votre visite ?



- 😊 Très satisfaisant
- Satisfaisant
- 😢 Non satisfaisant

## Intérêt

La borne de satisfaction a été présentée et utilisée lors des journées portes ouvertes, dans nos lycées.

**Projet concret** Les élèves visualisent rapidement le but à atteindre. Ils ont tous déjà vu une borne de satisfaction et ils imaginent rapidement son utilité.

**Motivation** La *journée portes ouvertes* est l'occasion de représenter le lycée auprès de personnes extérieures. Les élèves sont fiers de montrer leurs créations.

**Pluridisciplinarité** La création de la borne de satisfaction peut mobiliser de nombreuses disciplines sur le lycée : maths/sciences pour la conduite du projet, mathématiques pour l'exploitation des résultats, mode-vêtement et maroquinerie pour la décoration, arts appliqués pour les visuels ou encore bois, plasturgie et métallurgie pour la boîte.

## Matériel



- 1 x Micro:bit
- 1 x accès internet : IDE programmation par bloc <http://makecode.microbit.org/>
- 1 boite (carton, bois, métal, etc.)
- 4 câbles électriques + 4 pinces crocodiles
- 1 rouleau de papier aluminium



## NIVEAU INITIATION - PREMIER MODÈLE

### Activité élève



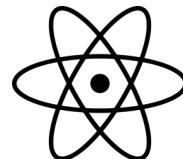
Durée



Public



Maths

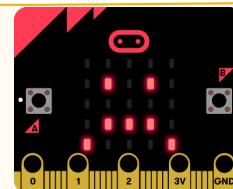


Sciences

Algo  
boucle ;  
événement

#### ACTIVITÉ

La Journée Portes Ouvertes aura lieu dans **1 mois** !  
Nous souhaitons créer une borne de satisfaction.



**TA MISSION :** Programme Micro:bit pour simuler une borne de vote.

Ta mission doit respecter les contraintes suivantes :

**lorsque le bouton A est pressé** une animation de type **non satisfaisant** apparaît

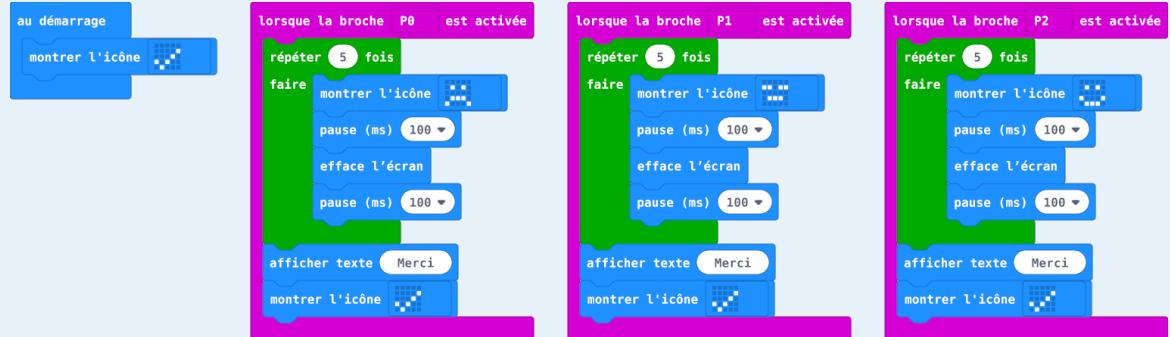
**lorsque le bouton B est pressé** une animation de type **très satisfaisant** apparaît

**après chaque animation** afficher un petit mot de remerciement



## MÉTHODE

Proposition de résolution :



### REMARQUE

Une proposition de code accessible en ligne <http://url.univ-irem.fr/w>.





## NIVEAU EXPERT - CUMULER LES VOTES

### Activité élève



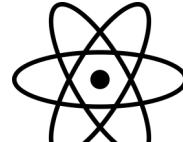
Durée



Public



Maths



Sciences

Algo  
boucle ;  
évènement ;  
variables

#### ACTIVITÉ

Bravo ! Tu as réussi à simuler une borne de vote à **2 choix** : *satisfaisant* ou *non-satisfaisant*.



Notre borne finale sera légèrement différente :

- la borne aura **3 choix** de votes possibles : non-satisfaisant ; satisfaisant ; très satisfaisant
- la borne **enregistrera** les réponses de chaque vote.

TA MISSION : (re)Programme Micro:bit pour simuler la borne de vote finale.

Modifie ton code précédent :

#### **lorsque la broche p0 est pressée**

- afficher une animation de type **non satisfaisant**
- afficher un mot de remerciement
- incrémenter la variable n0

#### **lorsque la broche p1 est pressée**

- afficher une animation de type **satisfaisant**
- afficher un mot de remerciement
- incrémenter la variable n1

#### **lorsque la broche p2 est pressée**

- afficher une animation de type **très satisfaisant**
- afficher un mot de remerciement
- incrémenter la variable n2

**lorsque le bouton A est pressé** afficher les valeurs des variables n0, n1 et n2



## MÉTHODE

Proposition de résolution :

```

toujours
  si bouton A est pressé ou bouton B est pressé alors
    montrer l'icône [pixel icon]
    pause (ms) 500
    efface l'écran
    montrer nombre n0
    montrer l'icône [pixel icon]
    pause (ms) 500
    efface l'écran
    montrer nombre n1
    montrer l'icône [pixel icon]
    pause (ms) 500
    efface l'écran
    montrer nombre n2
    montrer l'icône [pixel icon]

  au démarrage
    définir n0 à 0
    définir n1 à 0
    définir n2 à 0
    montrer l'icône [pixel icon]

lorsque la broche P0 est activée
  répéter (5) fois
    faire
      montrer l'icône [pixel icon]
      pause (ms) 100
      efface l'écran
      pause (ms) 100
      définir n0 à n0 + (1)
      afficher texte Merci
      montrer l'icône [pixel icon]

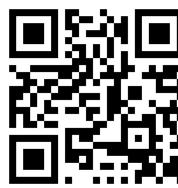
lorsque la broche P1 est activée
  répéter (5) fois
    faire
      montrer l'icône [pixel icon]
      pause (ms) 100
      efface l'écran
      pause (ms) 100
      définir n1 à n1 + (1)
      afficher texte Merci
      montrer l'icône [pixel icon]

lorsque la broche P2 est activée
  répéter (5) fois
    faire
      montrer l'icône [pixel icon]
      pause (ms) 100
      efface l'écran
      pause (ms) 100
      définir n2 à n2 + (1)
      afficher texte Merci
      montrer l'icône [pixel icon]

```

### REMARQUE

Une proposition de code accessible en ligne <http://url.univ-irem.fr/y>





# Les fractions avec Micro:bit

## DESCRIPTION

### Objectif

Le but de ce projet est :

**Cycle 4** Utiliser diverses représentations d'un même nombre ; passer d'une représentation à une autre.

Comparer, ranger, encadrer des nombres rationnels.

**CAP** Comparer, additionner, soustraire, multiplier et diviser les nombres en écriture fractionnaire dans des situations simples.

## Intérêt

Les fractions posent en général des difficultés aux élèves et la majorité font des blocages hérités des années antérieures. Le risque de démotivation est grand et l'usage d'un objet connecté permet de dédramatiser la séance.

**Aspect ludique** L'usage de l'affichage LED du micro :bit rend l'activité ludique. Les élèves sont pour la plupart attirés par le côté simple et sympathique de l'écran, ce qui les poussent à réfléchir et s'investir.

**Proportions** Afficher une image conforme à une contrainte est un problème concret qui permet de mieux matérialiser un calcul de proportion, par rapport à un énoncé classique (type statistiques). En évaluation, il y a plus de réponses justes quand la question repose sur un affichage d'image que sur une situation plus habituelle (point vérifié en classe).

**Plusieurs de méthodes de résolutions** Il y a de nombreuses façons d'arriver aux solutions. Par exemple pour éclairer  $\frac{2}{5}$  des LED, on peut d'abord faire des paquets de 5, en allumer 2 puis recommencer ; soit éclairer 2 colonnes sur les 5 existantes.

## Matériel



- 1 x Micro:bit (facultatif car le simulateur peut suffire, même si on perd un peu sur l'aspect ludique.)
- 1 x accès internet : IDE programmation par bloc <http://makecode.microbit.org/>



# ÉCRIRE UNE FRACTION

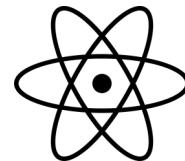
## Activité élève



Durée



Public

Maths  
représentation  
d'un nombre ;  
fraction

Sciences



Algo

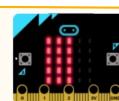
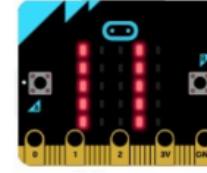
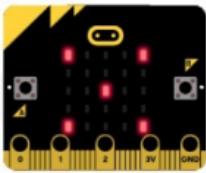
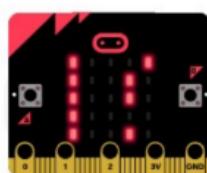
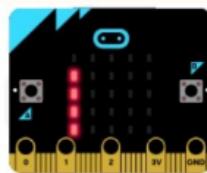
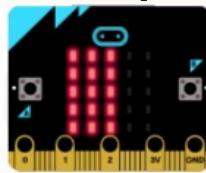
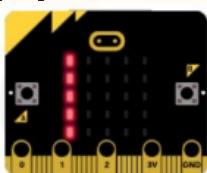
0,5 h

Cycle 4; CAP



### ACTIVITÉ

Écrire la proportion de LED allumée sur chaque micro:bit



## Notes pour l'enseignant

Attendus :

- Les élèves écrivent les fractions correspondant pour chaque appareil.
- Lorsque c'est possible, on écrit la fraction sous différentes formes :  $\frac{1}{5}$  ou  $\frac{5}{25}$ .
- On écrit alors les conditions d'égalité de deux fractions.

### REMARQUE

Prolongement possibles :

- Répondre par une phrase : il y a 5 LED allumées sur un total de 25.
- Pour chaque cas, indiquer la proportion de LED éteintes.



## AFFICHER UNE FRACTION

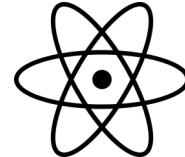
### Activité élève



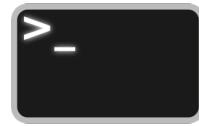
Durée



Public

Maths  
représentation  
d'un nombre ;  
fraction

Sciences



Algo

0,5 h

Cycle 4; CAP

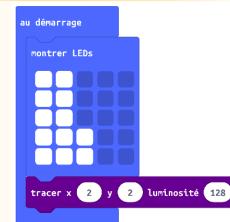
affichage



#### ACTIVITÉ

Représenter les fractions suivantes avec les LED du micro:bit

$$\frac{1}{2} ; \frac{32}{100}$$



### Notes pour l'enseignant

Attendus :

- Les élèves constatent qu'il n'est pas possible d'afficher un demi, on peut alors leur suggérer d'afficher la 13ème led en alternance ou de lui réduire la luminosité.
- Les élèves doivent simplifier la deuxième fraction pour afficher, on peut alors faire le lien avec les pourcentages



## SOMME DE FRACTIONS

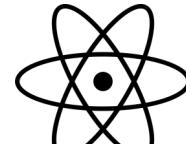
### Activité élève



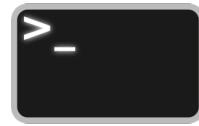
Durée



Public

Maths  
représentation  
d'un nombre ;  
fraction

Sciences



Algo

0,5 h

Cycle 4; CAP

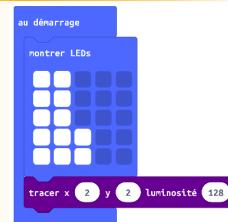
affichage



#### ACTIVITÉ

Représenter les fractions suivantes avec les LED du Micro:bit:

$$\frac{2}{5} + \frac{3}{25} ; \quad \frac{23}{25}$$



### Notes pour l'enseignant

#### REMARQUE

Plusieurs raisonnements sont à construire :

- créer des paquets de 5 et prendre à chaque fois 2 éléments
- diviser le tout en 5 paquets et en prendre 2



# Dé à 6 faces avec Micro:bit

## DESCRIPTION

### Objectif

Le but de ce projet est de simuler une expérience aléatoire de lancer de dé à 6 faces avec une carte Micro:bit.

Toujours à partir d'une situation simple idéale, le programme peut être étoffé au gré des besoins. Il s'agit de prolonger les quelques notions utilisées pour l'activité pile ou face, notamment lorsqu'il faudra tester les différentes issues et proposer une sortie en conséquence.

### Intérêt

L'intérêt de cette activité n'est pas seulement de se départir des bruits des dés qui roulent dans la classe ; l'usage du Micro:bit est ici très pertinent, tant pour les probabilités que pour la programmation.

**Simplicité de la situation.** La situation est très simple à expliquer et les élèves comprennent le but à atteindre. L'absence de difficulté mathématique rend cette situation particulièrement simple à mettre en œuvre.

**Motivation des élèves.** L'envie de programmer un objet connecté est grande pour les élèves. Cette façon de programmer, **utile, concrète et appliquée**, leur correspond parfaitement.

**De nombreuses solutions/améliorations possibles.** Comme pour bien des projets, il y a plusieurs façons d'arriver à la solution ; mais par rapport à l'activité **Pile ou Face** il y aussi plus de possibilité de faire des erreurs de programmation.

**Travail mathématique sur la modélisation.** La modélisation d'un dé non truqué est évidente, et n'apporte pas de difficulté majeure. Par contre l'intérêt majeur du Micro:bit est d'offrir la possibilité de truquer le dé. On a alors une situation beaucoup plus riche, tant dans la modélisation que dans les usages du modèle obtenu.

### Matériel



- 1 x Micro:bit (facultatif car le simulateur peut suffire)
- 1 x accès internet : IDE programmation par bloc <http://makecode.microbit.org/>



## NIVEAU SIMPLE

### Activité élève



Durée

0,25 h



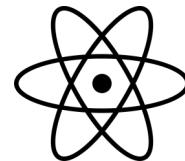
Public

2de



Maths

expérience aléatoire



Sciences



Algo

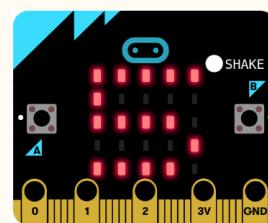
affichage ; événement.



MISSION Utilise Micro:bit pour simuler un **dé à 6 faces** !

En t'aidant des blocs ci-dessous, programme Micro:bit pour :

- 1) programmer un événement lorsque l'appareil est secoué ;
- 2) afficher **un nombre entier aléatoire entre 1 et 6**.



### Notes pour l'enseignant

Ce premier niveau aucun niveau de difficulté, on pourrait dire que son intérêt se limite à poser la problématique.

#### MÉTHODE

Pour résoudre ce problème, il suffit de programmer les instructions de la façon suivante :



#### REMARQUE

Plus d'informations sur la page de l'activité :  
<https://microbit.readthedocs.io/fr/latest/découverte/de6faces-bloc1.html>



## NIVEAU INTERMÉDIAIRE

### Activité élève



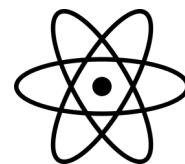
Durée



Public



Maths



Sciences



Algo

affichage ;  
événement ;  
condition

0,5 h

2de

expérience  
aléatoire

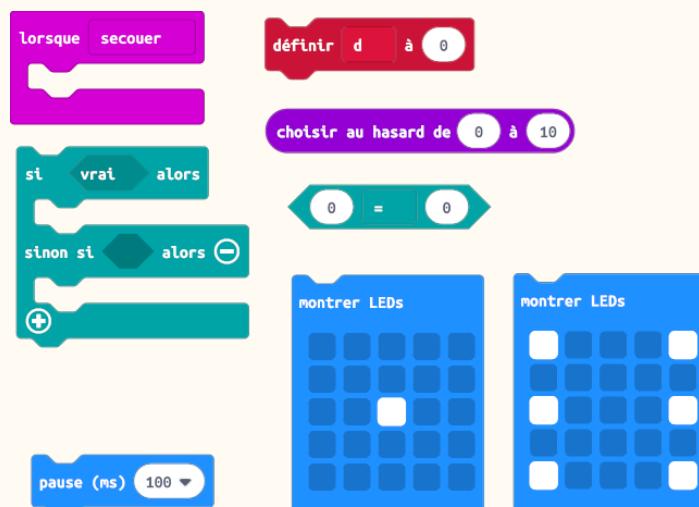
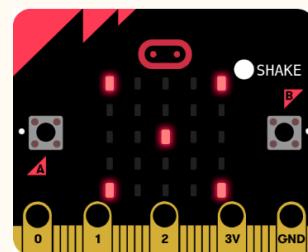
#### ACTIVITÉ

Utilise Micro:bit pour simuler un **dé à 6 faces**!

En t'aidant des blocs ci-dessous, programme Micro:bit pour :

- 1) programmer un événement lorsque l'appareil est secoué ;
- 2) afficher **une face de dé** selon le résultat d'un tirage aléatoire.

Attention tous les blocs ne sont pas représentés et certains blocs doivent être modifiés.





## Notes pour l'enseignant

Pour arriver au résultat escompté il est nécessaire d'utiliser une variable, ainsi qu'une suite d'instruction si/sinon si/sinon. Si l'usage d'une variable n'apparaît pas forcément évident pour les élèves, on peut les laisser programmer sans pour qu'il puisse ainsi constater le bug produit. Il est aussi intéressant de faire remarquer aux élèves que le nombre issue du tirage aléatoire n'a pas forcément à être celui symbolisé par sortie sur l'écran de DEL.

Le programme à assembler est disponible en suivant ce lien :

[https://makecode.microbit.org/\\_XdVPkRXfXLq6](https://makecode.microbit.org/_XdVPkRXfXLq6)

### MÉTHODE

Pour résoudre ce problème, il suffit de programmer les instructions de la façon suivante (seules les sorties correspondant à 1 et à 6 ont été incluses, pour raison de place) :

```

lorsque [secouer]
  définir [d] à [choisir au hasard de 0 à 5]
  si [d = 0] alors
    montrer LEDs
    [montrer LEDs v1]
  fin
  si [d = 1] alors
    montrer LEDs
    [montrer LEDs v2]
  fin
  si [d = 2] alors
    montrer LEDs
    [montrer LEDs v3]
  fin
  si [d = 3] alors
    montrer LEDs
    [montrer LEDs v4]
  fin
  si [d = 4] alors
    montrer LEDs
    [montrer LEDs v5]
  fin
  si [d = 5] alors
    montrer LEDs
    [montrer LEDs v6]
  fin
  pause (ms) [100 v]

```

### REMARQUE

Il est très simple en partant de cette situation d'envoyer les résultats par radio vers un Micro:bit qui centraliserait alors les issues de l'ensemble des tirages. Pour cela il suffit d'ajouter le bloc "envoyer le nombre" qui se trouve dans la catégorie "Radio".

```

lorsque [secouer]
  définir [d] à [choisir au hasard de 0 à 5]
  envoyer [d] par radio

```



# Normal ou truqué ? avec Micro:bit

## DESCRIPTION

### Objectif

Cette activité propose à l’élève de travailler sur la fluctuation d’échantillonnage. Sans accéder au code, seulement en manipulant la carte Micro:bit, l’élève doit déterminer si le jeu téléversé est un jeu normal ou truqué...

### Intérêt

L’intérêt de cette activité est de pouvoir, en toute liberté, créer des situations équiprobales ou non. Même s’il est possible d’utiliser de vrais dés truqués, l’utilisation d’une carte Micro:bit permet un éventail très larges de situations.

Ici le choix a été fait de procéder à un tirage aléatoire (ou non;) d’un nombre entre 0 et 9 (inclus).

### Matériel



- 1 x Micro:bit (facultatif car le simulateur peut suffire)
- 1 x accès internet : IDE programmation par bloc <http://makecode.microbit.org/>

### Progression

L’activité se déroule en 2 temps :

**Analyse du programme** Dans cette partie, l’élève doit étudier les deux programmes possibles (truqué ou non) puis répondre à une série de questions de probabilités

**Expérience aléatoire** Ensuite l’élève manipule la carte Micro:bit et effectue un certain nombre d’expériences aléatoires. Dans cette partie l’analyse des échantillons et la création d’une représentation graphique permettra de conclure sur le type de programme téléversé.



## ACTIVITÉ

### Activité élève



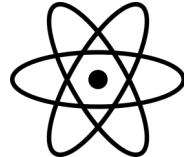
Durée



Public



Maths



Sciences



Algo

2 h

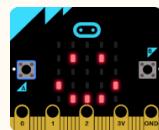
2de ; term

fluctuation  
d'échantillage

### ACTIVITÉ

NORMAL OU truqué ?

Sur votre carte est téléversé un des deux programmes de jeu suivant.  
Quand le joueur gagne, il a un smiley sourire, sinon il a une croix.



au démarrage

toujours

lorsque le bouton A est pressé

```
définir item à choisir au hasard de 0 à 9
si item < 5 alors
    montrer l'icône ☺
    pause (ms) 500
    efface l'écran
sinon
    montrer l'icône ✕
    pause (ms) 500
    efface l'écran
@
```

Jeu normal

au démarrage

toujours

lorsque le bouton A est pressé

```
définir item à choisir au hasard de 0 à 9
si item < 3 alors
    montrer l'icône ☺
    pause (ms) 500
    efface l'écran
sinon
    montrer l'icône ✕
    pause (ms) 500
    efface l'écran
@
```

Jeu truqué

### Analyse du programme

- 1) Si on choisit un nombre entier au hasard entre 0 et 9, combien y-a-t-il d'issues possibles ?
- 2) Si pour gagner il faut obtenir un nombre entre 0 et 4, combien y-a-t-il d'issues favorables ?
- 3) Calculer la probabilité de gagner avec la version normale du programme. Donner le résultat sous forme de fraction, de nombre décimal et de pourcentage.
- 4) Avec le même raisonnement, calculer la probabilité de gagner avec la version truquée du programme. Donner le résultat sous forme de fraction, de nombre décimal et de pourcentage.

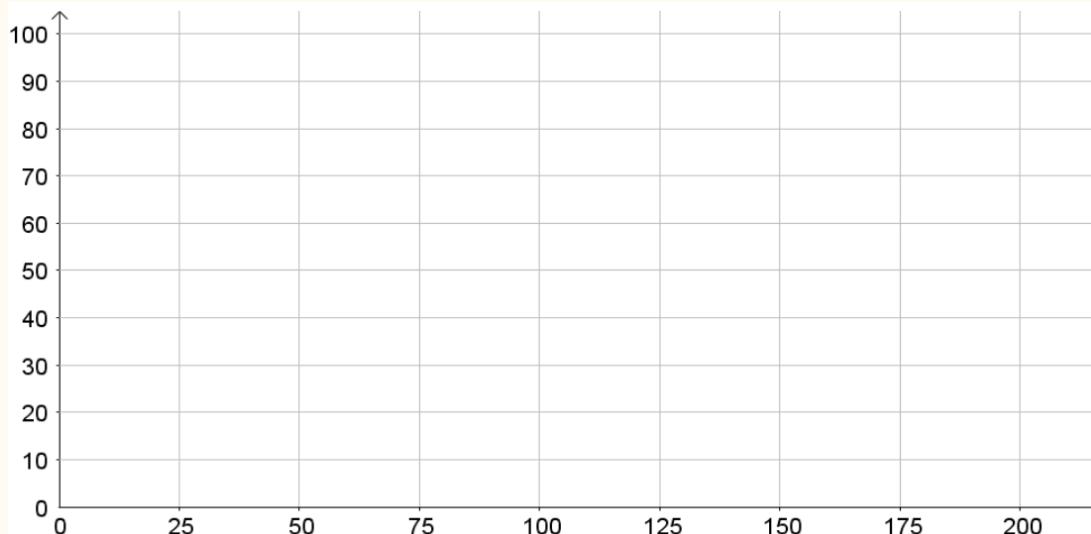


### Analyse du programme

- 5) Jouer des parties par séquence de 25 parties et noter G pour gagner et P pour perdu, dans votre cahier.
- 6) En utilisant vos résultats, compléter le tableau ci-dessous. **Attention!** Bien prendre son temps pour compter.

	25 parties	50 parties	75 parties	100 parties	125 parties	150 parties	175 parties	200 parties
Nombre de victoires								
Proportion de victoires en %	$\frac{\text{Nombre de victoires}}{25}$							

- 7) Compléter le graphique correspondant à la dernière ligne du tableau : placer un point par colonne, relier les points par des segments et légendrer les axes. Donner un **titre** au graphique.



- 8) Analyser le graphique et conclure sur le programme qui est téléchargé dans votre carte. Argumenter avec une ou plusieurs phrases.

### Notes pour l'enseignant

#### REMARQUE

Le code interactif est accessible en ligne :

**Jeu normal** [https://makecode.microbit.org/\\_OE1iAAee46YC](https://makecode.microbit.org/_OE1iAAee46YC)

**Jeu truqué** [https://makecode.microbit.org/\\_4zFAsTUuEgYV](https://makecode.microbit.org/_4zFAsTUuEgYV)



# Pile ou face avec Micro:bit

## DESCRIPTION

### Objectif

Le but de ce projet est de simuler une expérience aléatoire de lancer de pièce avec une carte Micro:bit.

À partir d'une situation simple, idéale pour une prise en main de l'interface de programmation, il s'agit par la suite d'améliorer le programme pas à pas. L'objectif est d'obtenir un programme utilisable dans le cadre d'un cours sur les statistiques et les probabilités.

### Intérêt

Bien évidemment, travailler avec une carte Micro:bit n'exclut pas de réaliser des expériences aléatoires réelles (pièces, dés, etc.). Cependant, il est très intéressant pour l'enseignant d'utiliser des Micro:bit dans cette partie du programme.

**Simplicité de la situation.** La situation est très simple à expliquer et les élèves comprennent le but à atteindre. L'absence de difficulté mathématique rend cette situation particulièrement simple à mettre en œuvre.

**Motivation des élèves.** L'envie de programmer un objet connecté est grande pour les élèves. Cette façon de programmer, **utile, concrète et appliquée**, leur correspond parfaitement.

**De nombreuses solutions/améliorations possibles.** Comme pour bien des projets, il y a plusieurs façons d'arriver à la solution. Par ailleurs, les élèves peuvent apporter ou proposer de nombreuses améliorations. La programmation par bloc, exempte de difficulté syntaxique, est particulièrement adaptée à la créativité.

**Travail mathématique sur la modélisation.** Cette compétence n'est pas facile à mettre en œuvre. Ici, l'absence de difficulté mathématique rend ce travail beaucoup plus accessible à tous.

### Matériel



- 1 x Micro:bit (facultatif car le simulateur peut suffire)
- 1 x accès internet : IDE programmation par bloc <http://makecode.microbit.org/>



## NIVEAU SIMPLE

### Activité élève



Durée

0,5 h



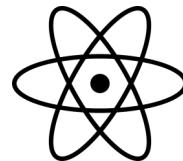
Public

2de



Maths

expérience aléatoire



Sciences



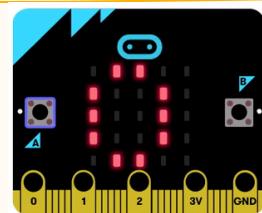
Algo

affichage ;  
boucle ;  
événement.

### ACTIVITÉ

MISSION : UTILISE MICRO:BIT POUR JOUER À **Pile ou Face** !

En t'aidant des blocs ci-dessous, programme Micro:bit pour :



- 1) afficher une courte animation ;
- 2) afficher de façon aléatoire 0 ou 1.

```

when button A pressed
repeat (2)
    show led matrix
    show number [0]
    choose random [0] to [1]
end

```

The Scratch script consists of the following blocks:

- When button A pressed
- Repeat (2)
- Show LED matrix
- Show number [0]
- Choose random [0] to [1]

Four sample LED matrix displays are shown to the right of the script, illustrating different possible outcomes of the random selection:

- Display 1: All red LEDs in the first two columns.
- Display 2: Red LEDs in the first two columns, and one red LED in the third column.
- Display 3: Red LEDs in the first two columns, and two red LEDs in the third column.
- Display 4: Red LEDs in the first two columns, and three red LEDs in the third column.



## Notes pour l'enseignant

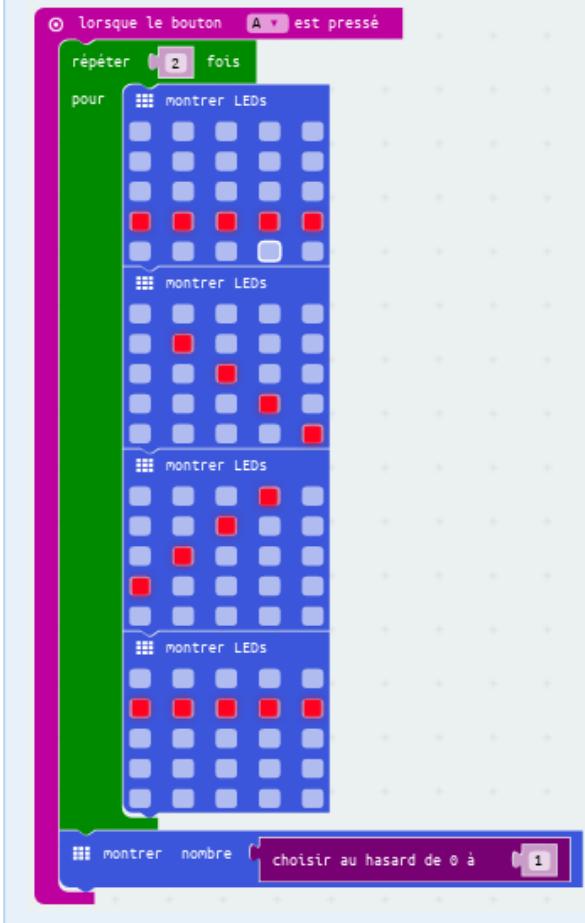
Ce premier niveau permet de se familiariser avec l'interface tout en produisant un premier programme fonctionnel et utile.

Le programme à assembler est disponible en suivant ce lien :

[https://makecode.microbit.org/\\_4JRa0easE67F](https://makecode.microbit.org/_4JRa0easE67F)

### MÉTHODE

Pour résoudre ce problème, il suffit de programmer les instructions de la façon suivante :



### REMARQUE

Plus d'informations sur la page de l'activité :  
<https://microbit.readthedocs.io/fr/latest/découverte/pileface-bloc1.html>



## NIVEAU INTERMÉDIAIRE

### Activité élève



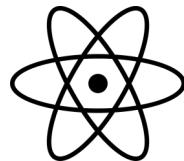
Durée



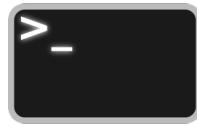
Public



Maths



Sciences



Algo

affichage;  
boucle;  
événement;  
condition;  
fonction.

0,5 h

2de

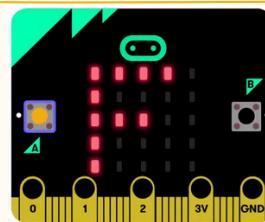
expérience  
aléatoire

#### ACTIVITÉ

Maintenant, améliore le programme de ton Micro:bit.

Voici deux idées :

- au lieu d'afficher 0 ou 1, affiche plutôt P ou F;
- utilise une fonction pour gérer ton animation.



```

when button A pressed
  call function flip
  choose random true or false
  if true then
    say [P]
  else
    say [F]
  end
end

```



## Notes pour l'enseignant

Ce deuxième niveau est l'occasion d'introduire les notions de fonctions et les instructions conditionnelles.

Le programme à assembler est disponible en suivant ce lien :

[https://makecode.microbit.org/\\_FM2Pszge4HoT](https://makecode.microbit.org/_FM2Pszge4HoT)

### MÉTHODE

Voici une proposition qui fonctionne :

```

lorsque le bouton A est pressé
  appeler la fonction flip
    si choisir au hasard vrai ou faux
      alors afficher texte "P"
      sinon afficher texte "F"

```

```

fonction flip
  répéter 2 fois
    pour montrer LEDs
      montrer LEDs
      montrer LEDs

```

### REMARQUE

Plus d'informations sur la page de l'activité :

<https://microbit.readthedocs.io/fr/latest/découverte/pileface-bloc2.html>



## NIVEAU EXPERT

### Activité élève



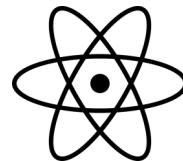
Durée



Public



Maths



Sciences



Algo

affichage ;  
boucle ;  
événement ;  
condition ;  
fonction ;  
variable  
(incément) ;  
texte (concaténation).

0,5 h

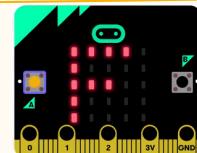
2de

expérience  
aléatoire

### ACTIVITÉ

Est ce que Micro:bit peut afficher les tirages obtenus ?

Pour finir, nous souhaitons ajouter une fonctionnalité supplémentaire à Micro:bit. Il faut maintenant arriver à afficher l'effectif total pour les Piles et pour les Faces.



Regarde ci-dessous les instructions que tu pourrais ajouter.

au démarrage

définir pile à 0

définir face à 0

changer pile par 1

changer face par 1

lorsque le bouton A est pressé

afficher texte "Hello!"

joindre  
"Bonjour"  
"Monde"  
" "  
" "  
" "  
" "  
pile  
face



## Notes pour l'enseignant

Dans ce troisième niveau, nous souhaitons compter les issues obtenues et afficher les totaux. Il faut introduire la notion de variable informatique.

Le programme à assembler est disponible en suivant ce lien :

[https://makecode.microbit.org/\\_PWgKvtHqYHku](https://makecode.microbit.org/_PWgKvtHqYHku)

### MÉTHODE

Le résultat escompté est le suivant :

```

au démarrage
  définir pile à 0
  définir face à 0

lorsque bouton A est pressé
  appeler fonction flip
    si [choisir au hasard vrai ou faux]
      alors
        afficher texte "P"
        changer pile pour 1
      sinon
        afficher texte "F"
        changer face pour 1
    fin si
fin lorsque

lorsque bouton B est pressé
  afficher texte (joinde [P:] (pile) [/] (F:) (face))

fonction flip
  répéter [2] fois
    pour [montrer LEDs]
      montrer LEDs
    fin pour
  fin répéter
fin fonction

```

### REMARQUE

Plus d'informations sur la page de l'activité :

<https://microbit.readthedocs.io/fr/latest/découverte/pileface-bloc3.html>



# Fluctuation d'échantillonnage avec Micro:bit

## DESCRIPTION

### Objectif

Le but de ce projet est d'expérimenter la fluctuation d'échantillonnage à partir d'une situation classique, en établissant tout d'abord un modèle d'expérience aléatoire à partir des données de la situation, puis en proposant de programmer le tirage d'échantillons pour une taille  $n$  fixée.

### Intérêt

L'utilisation de l'interface Micro:bit permet d'obtenir rapidement un programme fonctionnel, mais aussi d'afficher graphiquement la série de données produites. Cela représente un intérêt non-négligeable lorsque l'on souhaite traiter la fluctuation d'échantillonnage.

**Simulation d'une grande série d'expériences aléatoires** Contrairement à l'usage du tableur, où l'élève va devoir manipuler un grand nombre de données en colonnes et en lignes, au risque de se perdre dans leur traitement, l'approche algorithmique de ce problème permet d'aller à l'essentiel.

**Afficher les données** En utilisant le navigateur chrome, il est possible de recueillir les données générées et de les visualiser. L'échelle du graphique se réglant automatiquement sur le maximum et le minimum des données, cela permet une compréhension immédiate du phénomène.

### Matériel



- 1 x Micro:bit (facultatif car le simulateur peut suffire)
- 1 x accès internet : IDE programmation par bloc <http://makecode.microbit.org/>
- lien vers l'activité 1 : [https://makecode.microbit.org/\\_TbPFTK8eaKes](https://makecode.microbit.org/_TbPFTK8eaKes)
- lien vers l'activité 2 : [https://makecode.microbit.org/\\_PW8LCg82z3fh](https://makecode.microbit.org/_PW8LCg82z3fh)
- lien vers l'activité 3 : [https://makecode.microbit.org/\\_11aUTkWzR60J](https://makecode.microbit.org/_11aUTkWzR60J)



## Progression proposée

### MÉTHODE

On propose ici d'aborder la problématique en trois temps :

**1) Prise en main - Vérifier le modèle.**

Pour faciliter la prise en main et gagner du temps on propose aux élèves de vérifier un code déjà prêt. Cela facilite l'appropriation du problème.

**2) Intermédiaire - Du modèle à la génération d'échantillons.**

À partir du modèle, l'élève doit élaborer un algorithme afin de produire des échantillons de taille fixé.

**3) Avancé - Visualisation des données**

Cette étape consiste en une amélioration du programme précédent afin de pouvoir visualiser les données issues de la simulation.



## NIVEAU PRISE EN MAIN - VÉRIFIER LE MODÈLE...

### Activité élève



Durée

0,3 h



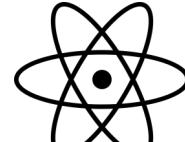
Public

première



Maths

statistiques et probabilités



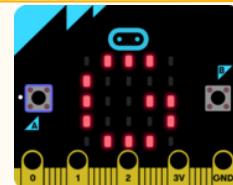
Sciences

Algo  
instruction conditionnelle

#### ACTIVITÉ

TA MISSION : Utiliser Micro:bit pour simuler des **naissances**!

La situation est la suivante : à Ufa, en Russie, 51,2 % des naissances sont des garçons.



Dans cette ville, une usine agrochimique expose ses employés à des pesticides contenant de la dioxine.

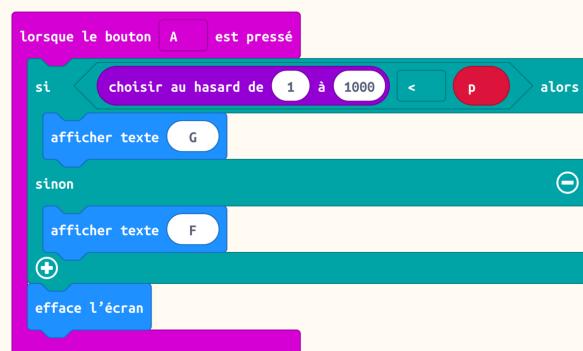
D'après une étude de l'université de Montréal, parmi les 227 enfants nés d'un parent travaillant dans cette usine, 91 sont des garçons.

Cette étude cherche à déterminer si l'usine interfère sur les naissances.

Pour simuler les naissances à Ufa, on propose d'utiliser le programme ci-dessous.

**Expliquer pourquoi** ce programme modélise correctement les naissance à Ufa.

**Proposer** une façon de l'exploiter afin de vérifier l'influence des produits chimiques sur les naissances.



### Notes pour l'enseignant

#### MÉTHODE

Dans cette activité, il s'agit de vérifier que les élèves ont bien compris la problématique et notamment comment est utilisé la fréquence d'apparition du caractère "garçon".

Bien entendu il faut suggérer aux élèves de tester le programme, afin d'en apprécier



les limites.

**REMARQUE**

Ici l'utilisation de la variable `p` n'est pas indispensable pour l'algorithme. Son intérêt est pédagogique : elle permet de mettre en évidence la donnée utilisée ainsi que de faire le lien avec le vocabulaire et les notations utilisées dans le cours.

Avant de passer à l'activité 2, il peut être préférable de lister avec les élèves les éléments manquants qui permettraient de produire et de traiter un échantillon comparable à celui de l'étude :

- une boucle répéter afin de produire un échantillon de taille 227
- des variables pour dénombrer les naissances de garçons (et de filles ?)



## NIVEAU INTERMÉDIAIRE - GÉNÉRER DES ÉCHANTILLONS...

### Activité élève



Durée

0,5 h



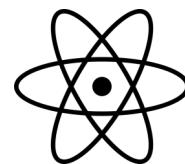
Public

première

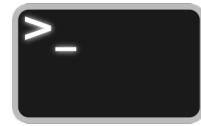


Maths

statistiques et probabilités



Sciences

Algo  
instruction conditionnelle ; boucle

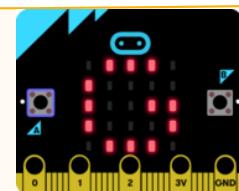
### ACTIVITÉ

TA MISSION : Utiliser Micro:bit pour simuler des naissances!

Utilise les blocs proposés afin de générer des échantillons de taille identique à celui de l'étude.

**Construire** un programme qui affiche le nombre de garçons obtenus dans un échantillon de 227 naissances.

**Utiliser** le programme afin de vérifier si la situation de la problématique est vraisemblable.



```

au démarrage
  définir p à 512
  définir f à 0
  définir g à 0
lorsque bouton A est pressé
  répéter [0] fois
    faire
      si [choisir au hasard de 1 à 1000 < p alors]
        montrer nombre g
        définir g à 0
        définir f à 0
      sinon
        changer g par 1
        changer f par 1
      fin
    fin
fin
  
```



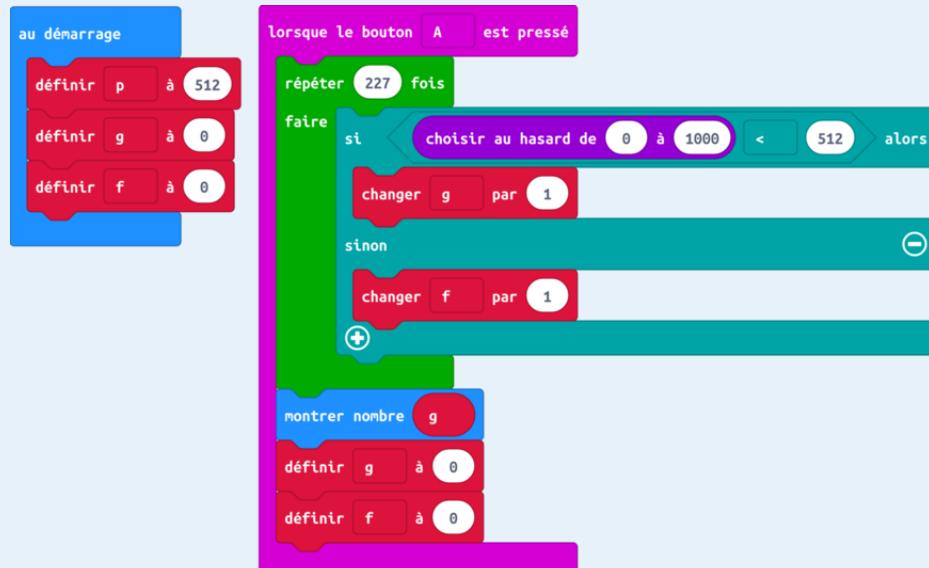
## Notes pour l'enseignant

### MÉTHODE

Dans cette activité, il s'agit de vérifier que les élèves se sont bien appropriés la problématique, notamment par rapport à la taille de l'échantillon.

Bien entendu il faut suggérer aux élèves de tester le programme plusieurs fois.

Le programme attendu peut être celui-ci si l'élève a utilisé tous les blocs proposés (voir Remarque ci-dessous) :



### REMARQUE

Ici l'utilisation de la variable **f** n'est pas indispensable pour l'algorithme. Son intérêt est pédagogique : elle permet de faire le lien avec l'activité précédente en conservant le modèle proposé.

Avant de passer à l'activité 3, il est tout de même préférable de faire s'interroger les élèves sur la nécessité de l'existence de la variable **f**.

Enfin, on interrogera les élèves sur le nombre d'échantillons qu'ils jugent nécessaires afin de valider leurs hypothèses.



## NIVEAU AVANCÉE - PRODUIRE DES DONNÉES...

### Activité élève



Durée

0,5 h



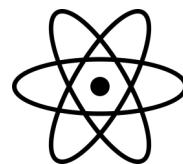
Public

première

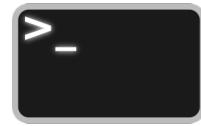


Maths

statistiques et probabilités



Sciences



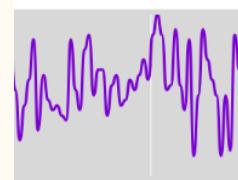
Algo

boucles imbriquées ; communication

#### ACTIVITÉ

TA MISSION : Utiliser Micro:bit pour simuler des **naissances** et produire des **données**!

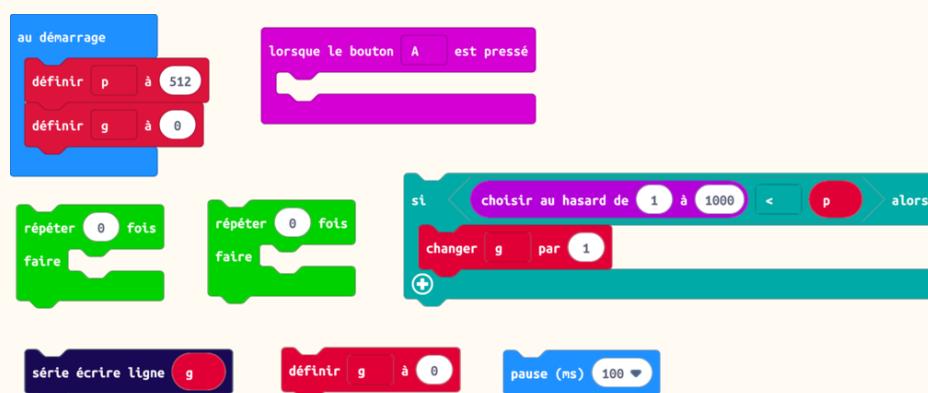
Utilise les blocs proposés afin de générer 100 échantillons de taille identique à celui de l'étude.



**Construire** un programme qui envoie une série de 100 valeurs, chacune correspondant au nombre de garçon dans un échantillon.

**Utiliser** le programme et la fonctionnalité **Afficher la console** du simulateur pour visualiser les données.

**Exploiter** les données produites pour déterminer si l'usine interfère sur les naissances de garçons.





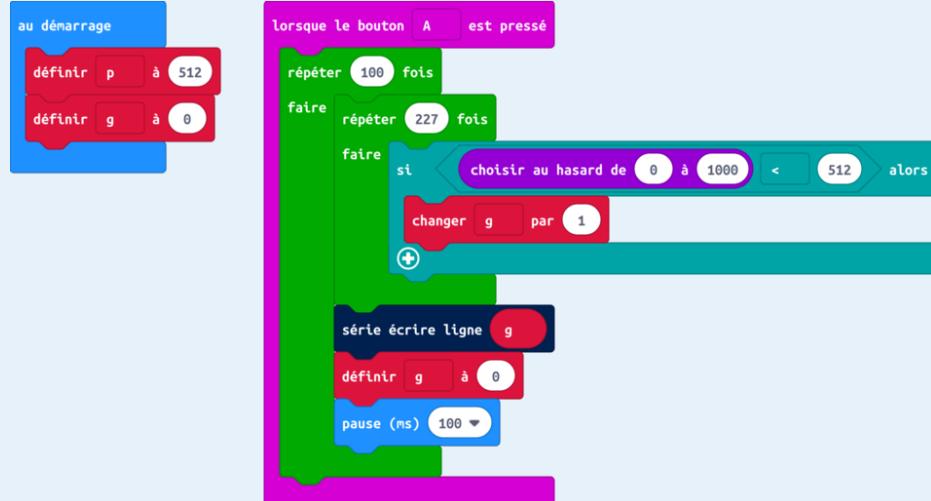
## Notes pour l'enseignant

### MÉTHODE

Dans cette activité, il s'agit de vérifier que les élèves sont bien capable d'interpréter le graphique obtenu.

Bien entendu il faut suggérer aux élèves de tester le programme plusieurs fois, en prenant soin d'attendre la fin d'une série avant d'en lancer une deuxième

Le programme attendu peut être celui-ci :



### REMARQUE

On interrogera les élèves sur les minimum et les maximum observés et sur la fréquence d'apparition d'un effectif inférieur ou égal à 91.

Les données produites sont exportables en .csv et donc exploitables dans un tableur.



# Chutes et oscillations avec Micro:bit

## DESCRIPTION

### Objectif

Le but de ce projet est de programmer un Micro:bit pour récupérer le nombre de chutes subies afin d'étudier les mouvement oscillants verticaux.

En partant d'une situation simple (compter des chutes), le programme va être étoffé afin de pouvoir déterminer la période des oscillations. Il faudra tout de même paramétrer avec soin le système oscillant et vérifier que la fonction du Micro:bit qui permet de détecter une chute libre se déclenche effectivement lors des mouvement du système. Il s'agit notamment de veiller à ce que l'amplitude soit suffisante.

## Intérêt

Cette activité permet d'effectuer des mesures physiques de façon simple et efficace tout en demandant un minimum de programmation.

**Simplicité de la situation.** La situation est très simple à expliquer et les élèves comprennent le but à atteindre. La problématique liée à l'affichage dans le niveau 1 permet d'aborder des problèmes de tris classiques en programmation, c'est l'occasion de voir comment des notions simples de mathématiques (arrondi, modulo) sont utiles dans ce type de situation.

**Motivation des élèves.** Dans cette activité, l'élève est acteur tout au long de la chaîne de l'expérimentation et utilise un composant électronique que l'on retrouve dans de nombreux objets (téléphone, manettes de jeux, hoverboard...).

**De nombreuses solutions/améliorations possibles.** Cette activité propose une solution qui est suffisante pour étudier les oscillations, puisqu'elle ne fait qu'automatiser un traitement qui pouvait être fait à la main. Néanmoins il est envisageable par exemple de coupler un deuxième Micro:bit afin de recueillir et de traiter les données en directes.

**Démarche scientifique.** Il peut être intéressant de faire évaluer la précision du Micro:bit afin de vérifier que la fiabilité des mesures. Par exemple une même expérience pourrait faire l'objet d'une mesure avec le Micro:bit, avec un opérateur manuel et à l'aide de la vidéo. La détection de la chute libre a été évoquée précédemment, là encore il est intéressant d'étudier avec les élèves les limites de l'expérience.

## Matériel



- 1 x Micro:bit
- 1 x accès internet : IDE programmation par bloc <http://makecode.microbit.org/>
- ressorts de différentes raideurs
- 1 support vertical
- masses marquées



## NIVEAU SIMPLE

### Activité élève



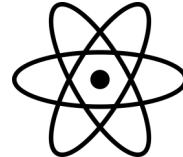
Durée



Public



Maths



Sciences



Algo

affichage ;  
événement ;  
tri ; variable

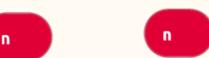
#### ACTIVITÉ

Utilise Micro:bit pour compter et afficher un **nombre de chutes**!

En t'a aidant des blocs ci-dessous, programme Micro:bit pour :



- 1) programmer un événement lorsque l'appareil est en chute libre ;
- 2) itérer une variable correspondant au **nombre de chutes** subies par le Micro:bit.
- 3) afficher ce nombre en allumant une nouvelle LED à chaque chute.





## Notes pour l'enseignant

La difficulté de ce niveau tient dans l'affichage, ce qui pourrait facilement être contournée en affichant simplement un nombre plutôt qu'en allumant une nouvelle LED à chaque nouvelle chute. Néanmoins il est intéressant d'étudier comment ce problème d'affichage peut-être résolu simplement avec les deux opérateurs mathématiques arrondi et modulo.

### MÉTHODE

Pour résoudre ce problème, il suffit de programmer les instructions de la façon suivante :

```

au démarrage
  définir [n] à [-1]

lorsque [chute libre]
  changer [n] par [1]
  allumer x [floor (n ÷ 5) y [reste de (n ÷ 5)]]

```

### REMARQUE

La page vers l'interface de programmation avec le code prêt à télécharger :

[https://makecode.microbit.org/  
xxxxxxxxxx](https://makecode.microbit.org/xxxxxxxxxx)



## NIVEAU INTERMÉDIAIRE

### Activité élève



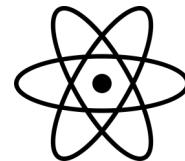
Durée



Public



Maths



Sciences



Algo

1 h

Term

indicateurs statistiques

mécanique

affichage ;  
événement ;  
variable ; liste

### ACTIVITÉ

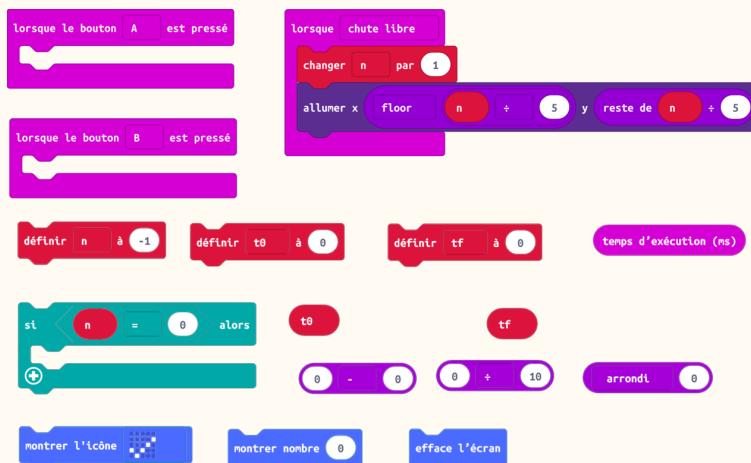


Utilise Micro:bit pour déterminer une période moyenne **une période moyenne d'oscillation !**

En t'a aidant des blocs ci-dessous, programme Micro:bit pour :

- 1) programmer un événement lorsque l'appareil est en chute libre ;
- 2) itérer **une variable** correspondant au **nombre de chutes** subies par le Micro:bit.
- 3) utiliser le bloc **temps d'exécution** pour mesurer la durée d'un nombre d'oscillation déterminé (10 par exemple)
- 4) calculer la période moyenne d'oscillation (en ms et afficher le résultat)

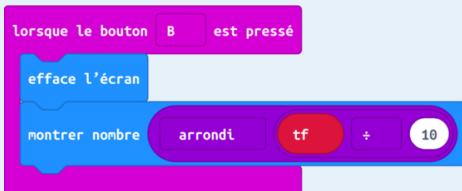
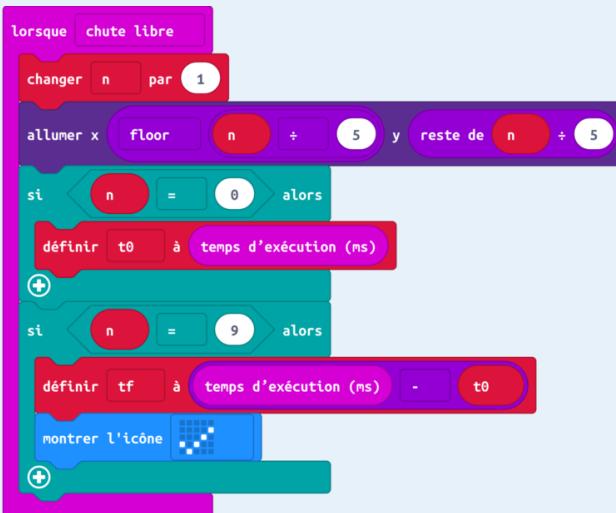
Attention tous les blocs ne sont pas représentés : certains doivent être doublés et/ou modifiés.





## MÉTHODE

Pour résoudre ce problème, il suffit de programmer les instructions de la façon suivante :

```

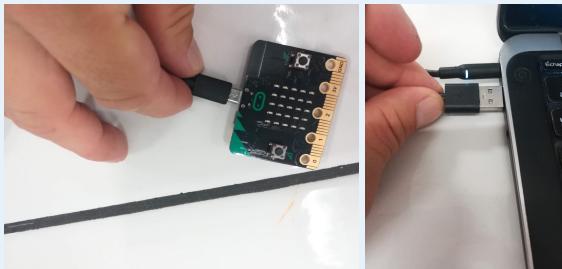
when button A is pressed
  define n as -1
  define t0 as 0
  define tf as 0
when button B is pressed
  clear [all]
  show (number (round (n / 10)))
when free fall
  change n by 1
  set x to (floor (n / 5)) y to (remainder (n / 5))
  if (n = 0) then
    set t0 to (execution time (ms))
  else if (n = 9) then
    set tf to (execution time (ms)) - t0
  end
  show icon [grid icon v1]

```

# Mémo Mu-Editor pour Micro:bit (et python)

## MÉTHODE (1ER PROGRAMME)

Afficher un premier texte sur l'écran Micro:bit.  
**Connecter** la carte à l'ordinateur



**Ouvrir** Mu-editor  
**Copier** le code ci-dessous.

```
from microbit import *
display.scroll("Hello, World!")
```

**Flasher** la carte (envoyer le programme dans la carte)



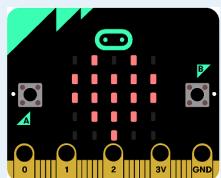
## MÉTHODE (DES IMAGES)

Afficher une image sur l'écran du Micro:bit.

**Copier** le code et **flasher** la carte



```
from microbit import *
for loop in range (10):
    display.show(Image.HEART)
    sleep(100)
    display.show(Image.HEART_SMALL)
    sleep(50)
display.scroll("Hello, World!")
```



## MÉTHODE (LES MOUVEMENTS)

Afficher des enregistrements de l'**accéléromètre**

**Copier** le code et **flasher** la carte



```
from microbit import *
display.show(Image.YES)
while True:
    valeurs= accelerometer.get_values()
    print (valeurs)
    sleep(100)
```

Afficher la vue REPL (**terminal série**)

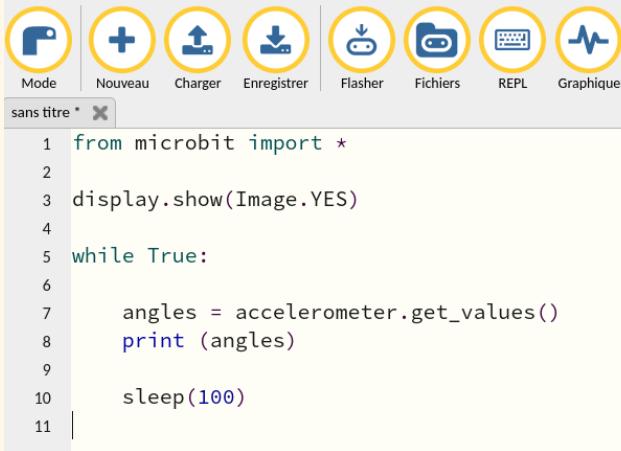


**Réinitialiser** le Micro:bit



## REMARQUE

Vous devriez avoir un affichage du type



```
sans titre * 
1 from microbit import *
2
3 display.show(Image.YES)
4
5 while True:
6
7     angles = accelerometer.get_values()
8     print (angles)
9
10    sleep(100)
11
```

```
BBC micro:bit REPL
MicroPython v1.9.2-34-gd64154c73 on 2017-09-01; m
Type "help()" for more information.
>>>
MicroPython v1.9.2-34-gd64154c73 on 2017-09-01; m
Type "help()" for more information.
>>> (296, 332, -928)
(296, 336, -936)
(292, 292, -940)
(268, 324, -928)
(284, 328, -932)
(296, 312, -940)
(284, 316, -932)
(276, 332, -936)
(288, 332, -928)
```

## MÉTHODE (GRAPHIQUE)

Afficher les enregistrements de l'accéléromètre sous forme de **graphique**

Copier le code et **flasher** la carte

```
from microbit import *
display.show(Image.YES)
while True:
    valeurs= accelerometer.get_values()
    print (valeurs)
    sleep(100)
```



Afficher la vue **Graphique**

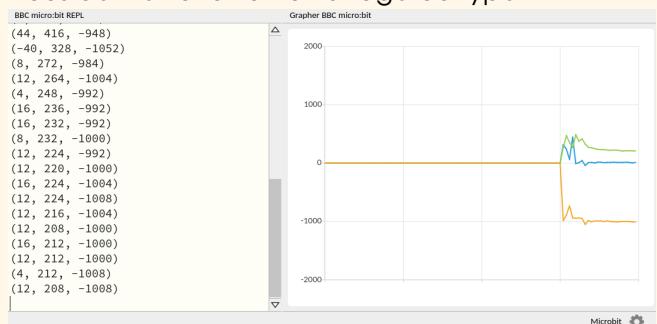


Réinitialiser le Micro:bit

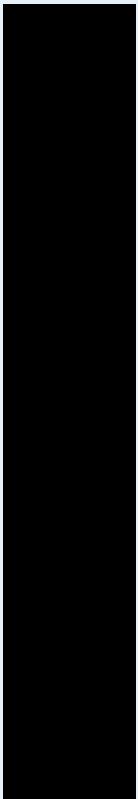


## REMARQUE

Vous devriez avoir un affichage du type



## MÉTHODE (COMMUNICATION RADIO)



## MÉTHODE (ENREGISTRER DES FICHIERS)



# Mémo Micro:bit (et python)

## MÉTHODE (ESSENTIEL)

Importer toutes les fonctions

```
from microbit import *\n\nsleep(...)
```

Faire une pause (en ms)

## MÉTHODE (ÉCRIRE DANS UN TERMINAL)

Écrire du **texte** (string)

```
print()
```

```
ex. print('Bouton A : 5 fois')
```

Aligner avec des **tabulations**

```
'\t'
```

```
ex. print('Bouton \tA \t : \t 5 fois')
```

Sauter des lignes

```
'\n'
```

```
ex. print('Bouton A :\n 5 fois\n ')
```

Convertir en chaînes (string)

```
str()
```

```
ex. print('Bouton A : ' + str(5) + ' fois')
```

## REMARQUE

Activer la **communication série**



## MÉTHODE (LED Micro:bit)

Afficher texte/image

```
display.scroll()
```

```
ex. display.scroll("G")
```

```
ex. display.scroll(Image.HEART)
```

Faire **défiler** texte/image

```
display.show()
```

```
ex. display.show("Gagne !")
```

```
ex. display.show(Image.HAPPY)
```

Créer des **animations**

```
delay=...
```

```
ex. display.show(Image.ALL_CLOCKS, delay=200)
```

Modifier **pixel**

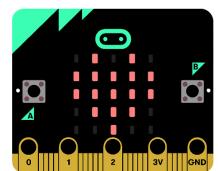
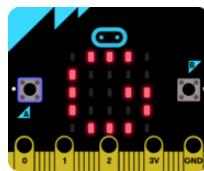
```
display.set_pixel(x,y,lum)
```

(lum = luminosité [0;9])

```
ex. display.set_pixel(0,4,9)
```

Effacer l'écran

```
display.clear()
```



## REMARQUE

Pour créer/utiliser des **animations**,  
⇒ **tableaux d'images**.

```
ex. Image.ALL_CLOCKS (lignes)
```

```
ex. Image.ALL_ARROWS (flèches)
```

Image.HEART

Image.CLOCK12

Image.CHESSBOARD

Image.TSHIRT

Image.HEART\_SMALL

Image.CLOCK11

Image.DIAMOND

Image.ROLLERSKATE

Image.HAPPY

...

Image.DIAMOND\_SMALL

Image.DUCK

Image.SMILE

Image.CLOCK1

Image.SQUARE

Image.HOUSE

Image.SAD

Image.ARROW\_N

Image.SQUARE\_SMALL

Image.TORTOISE

Image.CONFUSED

Image.ARROW\_NE

Image.RABBIT

Image.BUTTERFLY

Image.ANGRY

Image.ARROW\_E

Image.COW

Image.STICKFIGURE

Image.ASLEEP

Image.ARROW\_SE

Image.MUSIC\_CROTCHET

Image.GHOST

Image.SURPRISED

Image.ARROW\_S

Image.MUSIC\_QUAVER

Image.SWORD

Image.SILLY

Image.ARROW\_SW

Image.MUSIC\_QUAVERS

Image.GIRAFFE

Image.FABULOUS

Image.ARROW\_W

Image.PITCHFORK

Image.SKULL

Image.MEH

Image.ARROW\_NW

Image.XMAS

Image.UMBRELLA

Image.YES

Image.TRIANGLE

Image.PACMAN

Image.SNAKE

Image.NO

Image.TRIANGLE\_LEFT

Image.TARGET

## MÉTHODE (LES BOUTONS)

Accéder aux **boutons** A ou B

```
button_a. ...
button_b. ...
```

Nombre d'appuis (**cumul**)

```
button_a.get_presses()
button_b.get_presses()
```

*ex.* `display.scroll(str(button_a.get_presses()))`

Tester si bouton **est** pressé

```
button_a.is_pressed()
button_b.is_pressed()
```

*ex.* `if (button_b.is_pressed()):print("Bouton B")`

Tester si bouton **a été** pressé

```
button_a.was_pressed()
```

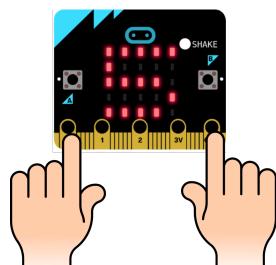
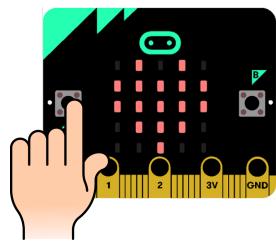
(appeler 2 × la fonction)

```
button_b.was_pressed()
```

*ex.* `if (button_b.was_pressed()):display.show(Image.HAPPY)`

Utiliser les **broches**

```
pin0.is_touched()
pin1.is_touched()
pin2.is_touched()
```



### REMARQUE

DéTECTER un **évenement**, avec une **boucle infinie**.

*ex.* `while True:`

Pour **sortir d'une boucle** lorsque l'évènement est détecté :

## MÉTHODE (ALÉA)

Activer le module **hasard**

```
import random
```

Nombre **aléatoire** sur [0; 1]

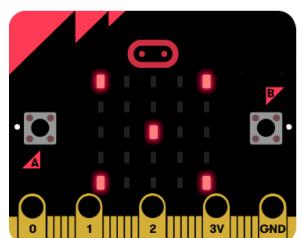
```
random.random()
```

*ex.* `display.show(round(random.random(),3))`

Nombre **entier** aléatoire sur [a; b]

```
random.randint(a,b)
```

*ex.* `display.show(random.randint(1,6))`



## MÉTHODE (MOUVEMENT)

**1 coord.** vect. accélération

```
accelerometer.get_x()
accelerometer.get_y()
accelerometer.get_z()
```

*ex.* `display.show(accelerometer.get_x())`

```
accelerometer.get_values()
```

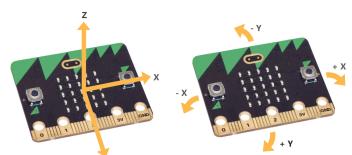
*ex.* `print(accelerometer.get_values())`

```
accelerometer.get_gestures()
```

*ex.* `print(accelerometer.get_gestures())`

```
accelerometer.is_gesture('...')
```

```
accelerometer.was_gesture('...')
```



**Gestes** reconnus

'up'	'freefall'
'down'	'3g'
'left'	'6g'
'right'	'8g'
'face up'	'shake'
'face down'	

**3 coord.** vect. (tuple)

Historique **gestes** (tuple)

(appeler 2 × la fonction)

Tester geste **actuel**

Tester geste **passé**

(appeler 2 × la fonction)



## MÉTHODE (RADIO)

Activer le **module** radio

**Mettre en route** la radio

**Envoyer** un message

**Message** reçu (string) (= *None* si vide)

ex. `if (radio.receive()=='pile'): print('Pile')`

**Canal** perso ( $\leq 100$ )

`radio.config(channel=..)`

ex. `radio.config(channel=21)`

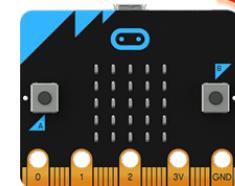
**Puissance** perso ( $\leq 7$ )

`radio.config(power=..)`

ex. `radio.config(power=7)`

```
import radio  
radio.on()  
radio.send('...')  
ex. radio.send('pile')
```

`radio.receive()`



## MÉTHODE (BOUSSOLE)

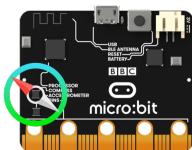
**1 coord.** vect. champ magnétique      `compass.get_x()`  
`compass.get_y()`  
`compass.get_z()`

ex. `display.show(compass.get_x())`

**Angle** (en °) avec le nord      `compass.heading()`

ex. `aiguille=((15-compass.heading())//30)%12`

**Calibrer** la boussole      `compass.calibrate()`



### REMARQUE

**Toujours** calibrer la boussole.

Mesures/valeurs **fluctuantes**  
(présence de métal, d'aimants, etc.)



# Simuler un dé avec Micro:bit (et python)

## DESCRIPTION

### Objectif

Le but de ce projet est de simuler une expérience aléatoire de lancer d'un dé à 6 faces mais en nous appuyant sur les possibilités offertes par le langage Python.

Ainsi, on ne se contentera pas seulement d'afficher l'issue, mais on montrera comment stocker et traiter facilement les effectifs obtenus.

Ici, contrairement à l'activité proposée en programmation par blocs, nous suggérons de commencer par simuler l'affichage tel qu'il apparaît sur un vrai dé. Cela permet de réexploiter les listes d'images introduites avec le projet pile ou face.

Cette activité permettra de couvrir de nombreuses capacités du programme de mathématiques de seconde Bac Pro, tant dans le domaine des statistiques que celui de l'algorithmique.

## Intérêt

**expérimenter** En partant d'une modélisation simple, l'élève va pouvoir observer une expérience aléatoire, dont il va facilement pouvoir visualiser et traiter les résultats.

**programmation événementielle** Pour déclencher les tirages et l'affichage, on peut utiliser les boutons, mais on peut imaginer utiliser la détection de mouvements.

**gestion des listes** Même les listes ne font pas partie explicitement des types de données que les élèves sont sensés maîtrisées, il est difficile de s'en passer en programmation. Cependant, il ne semble pas insurmontable de les initier aux principes de création, modification et parcours de listes dans la mesure où l'on retrouve les notions de variables et de boucle for. La majeure difficulté pourrait éventuellement provenir des index.

**programmation fonctionnelle** le calcul des fréquences peut se faire au travers d'une fonction, ce qui permet de scinder la production des données et leur traitement.

**de multiples déclinaisons possibles** Une fois la situation du tirage d'un dé bien établie, il est facilement envisageable d'étendre à d'autres situations : dés non conventionnels, dés multiples.

## Matériel



- 1 x Micro:bit
- 1 x IDE programmation python (Mu) <https://codewith.mu/> ou interface de programmation ligne <https://python.microbit.org/v/1.1>

## Remarques

### MÉTHODE

Pour afficher aléatoirement un nombre entier entre 1 et 6, on peut se contenter des 3 lignes ci-dessous.



```
while True:  
    if button_a.get_presses():  
        display.show(str(randint(1, 6)))  
        sleep(800)
```

Cette solution peut être envisagée afin de simplifier l'approche, mais la création d'image permet de travailler le repérage et la représentation des nombres, en plus de proposer un affichage plus attrayant.

### 1) Initiation - Mise en place du modèle.

Dans cette activité, il s'agit de proposer aux élèves de créer les images et d'associer l'affichage au résultat d'un tirage aléatoire.

### 2) Intermédiaire - Enregistrement des résultats.

L'étape suivante consistera d'une part à stocker les effectifs des différentes issues dans une listes, et d'autre part à afficher les données recueillies.

### 3) Intermédiaire - Calcul des fréquences

Le dernier niveau consistera à déterminer les fréquences des issues. Pour cela on utilisera une fonction pour calculer et arrondir.



## NIVEAU INITIATION

### Activité élève



Durée

0,5 h



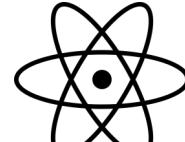
Public

2de

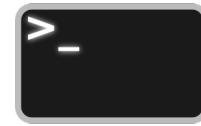


Maths

probabilités



Sciences

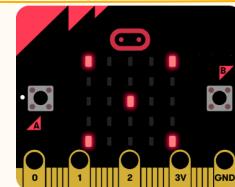


Algo

affichage  
boucles listes

#### ACTIVITÉ

TA MISSION : Utiliser une carte Micro:bit pour simuler un dé, en utilisant le langage Python.



Nous utiliserons la bibliothèque `random` et la fonction `randint()` pour tirer un nombre entier aléatoirement entre deux bornes. Il faut donc importer un module en plus de microbit :

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

Le programme précédent devra être complété avec les éléments suivants :

- 1) Dans un premier temps nous allons créer les images nécessaires pour afficher les nombres comme sur un vrai dé.  
Par exemple le code ci-dessous permet de créer l'image représentant l'issue "cinq".

```
cinq = Image("90009:00000:00900:00000:90009")
```

Chaque nombre correspond à une diode, éclairée de 0 à 9.

- 2) Ces images sont ensuite enregistrée dans une liste :

```
issues = [un, deux, trois, quatre, cinq, six]
```

- 3) Après le démarrage de la boucle `While True:` on déclenchera le tirage avec le bouton A : `button_a.get_presses()` .
- 4) La première image du tableau `issues` est numérotée à 0 et la dernière à 5. On effectue donc un tirage aléatoire d'un entier entre 0 et 5, puis on demande au Micro:bit d'afficher l'image correspondant à ce numéro.

```
i = randint(0, 5)
display.show(issues[i])
```

Vérifie ton code avec et flash-le sur la carte avec



## MÉTHODE

Proposition de résolution : Il faut bien entendu importer les modules nécessaires

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

puis créer les images

```
un = Image("00000:00000:00900:00000:00000")
deux = Image("00009:00000:00000:00000:90000")
trois = Image("90000:00000:00900:00000:00009")
quatre = Image("90009:00000:00000:00000:90009")
cinq = Image("90009:00000:00900:00000:90009")
six = Image("90009:00000:90009:00000:90009")
```

et enfin créer la boucle permettant d'afficher le résultat d'un tirage :

```
if button_a.get_presses():
    display.show(str(randint(1, 6)))
    sleep(800)
    display.clear()
```

### REMARQUE

Les deux dernières lignes ne sont pas indispensables, mais elles permettent de mieux différencier les tirages, surtout lorsque l'on tire le même nombre.



## NIVEAU INTERMÉDIAIRE

### Activité élève



Durée

0,5 h



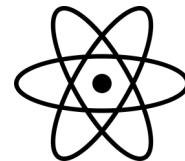
Public

2de



Maths

probabilités



Sciences

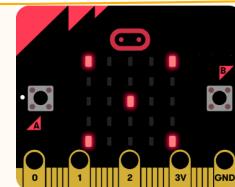


Algo

affichage  
boucles listes

#### ACTIVITÉ

TA MISSION : Utiliser une carte Micro:bit pour simuler un dé, et dénombrer les issues en utilisant le langage Python.



Comme pour l'activité précédente, nous utiliserons la bibliothèque `random` et la fonction `randint()`, le programme commencera donc aussi par :

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

Le programme de l'activité précédente devra être complété avec les éléments suivants, à vous de déterminer quelle doit être leur position dans le programme :

- 1) Nous allons devoir stocker les effectifs des différentes issues, pour cela il faudra créer un tableau de données que nous initialiserons avec des "0". Par exemple :

```
data = [0, 0, 0, 0, 0, 0]
```

- 2) Lorsqu'un nombre est tiré, il faudra augmenter l'effectif correspondant de 1. Étant donné que l'on se sert déjà du nombre tiré aléatoirement pour afficher l'image, il n'y a qu'une ligne à rajouter. Par exemple si le tableau s'appelle "data" :

```
data[i] = [data[i]+1
```

- 3) Pour déclencher l'affichage des effectifs, on utilisera le bouton B :

```
button_b.get_presses()
```

- 4) Pour parcourir le tableau, il faudra faire une boucle `for ... in ...` sur le tableau et afficher chaque élément du tableau avec `display.scroll(...)`.

```
for effectif in data :
    display.scroll(str(effectif)+" /")
```



## MÉTHODE

Proposition de résolution :

Le début du programme est inchangé.

```
from microbit import *
from random import randint

un = Image("00000:00000:00900:00000:00000")
deux = Image("00009:00000:00000:00000:90000")
trois = Image("90000:00000:00900:00000:00009")
quatre = Image("90009:00000:00000:00000:90009")
cinq = Image("90009:00000:00900:00000:90009")
six = Image("90009:00000:90009:00000:90009")

issues = [un, deux, trois, quatre, cinq, six]
```

Il faut ajouter l'initialisation du tableau d'effectifs :

```
data = [0, 0, 0, 0, 0, 0]
```

Dans la boucle permettant d'afficher le résultat d'un tirage, il faut inclure l'incrémentation :

```
while True:
    if button_a.get_presses():
        i = randint(0, 5)
        display.show(issues[i])
        data[i] = data[i] + 1
        sleep(1000)
        display.clear()
```

L'affichage des effectifs se fait par parcours du tableau et défilement :

```
if button_b.get_presses():
    for effectif in data:
        display.scroll(str(effectif)+" /")
```

### REMARQUE

Cette activité étant très guidée, la complexité réside dans le positionnement des instructions proposées dans le programme et dans la compréhension du fonctionnement des listes.

Les listes n'étant pas spécifiquement demandées dans le référentiel de seconde, elles sont à utiliser avec précaution, il est aussi envisageable de créer des variables pour enregistrer l'effectif de chaque issue, mais cela alourdi considérablement le programme.



## NIVEAU AVANCÉ

### Activité élève



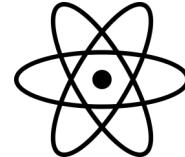
Durée



Public



Maths



Sciences



0,5 h

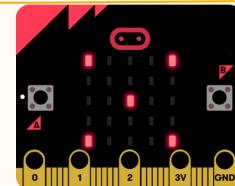
2de

probabilités

Algo  
affichage  
boucles listes  
fonctions

#### ACTIVITÉ

TA MISSION : Utiliser une carte Micro:bit pour simuler un dé, et observer la fluctuation des fréquences en utilisant le langage Python.



Comme pour l'activité précédente, nous utiliserons la bibliothèque `random` et la fonction `randint()`, le programme commencera donc aussi par :

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

Le programme de l'activité précédente devra être complété avec les éléments suivants, à vous de déterminer quelle doit être leur position dans le programme :

- 1) Nous allons ajouter une fonction qui nous permettra de calculer les fréquences de chaque issue.
- 2) Pour calculer les fréquences, il nous faut connaître l'effectif total des tirages, et donc pour cela nous devons créer une variable N.
- 3) Afin d'obtenir rapidement un grand nombre de tirage, nous allons inclure les instructions liés au tirage dans une boucle de 10 répétitions.
- 4) Pour déclencher l'affichage des fréquences, on utilisera le bouton B :  
`button_b.get_presses()`
- 5)



## MÉTHODE

Proposition de résolution :

Le début du programme est inchangé.

```
from microbit import *
from random import randint

un = Image("00000:00000:00900:00000:00000")
deux = Image("00009:00000:00000:00000:90000")
trois = Image("90000:00000:00900:00000:00009")
quatre = Image("90009:00000:00000:00000:90009")
cinq = Image("90009:00000:00900:00000:90009")
six = Image("90009:00000:90009:00000:90009")

issues = [un, deux, trois, quatre, cinq, six]
```

Il faut ajouter l'initialisation du tableau d'effectifs :

```
data = [0, 0, 0, 0, 0, 0]
```

Dans la boucle permettant d'afficher le résultat d'un tirage, il faut inclure l'incrémentation :

```
while True:
    if button_a.get_presses():
        i = randint(0, 5)
        display.show(issues[i])
        data[i] = data[i] + 1
        sleep(1000)
        display.clear()
```

L'affichage des effectifs se fait par parcours du tableau et défilement :

```
if button_b.get_presses():
    for effectif in data:
        display.scroll(str(effectif)+" /")
```

### REMARQUE

Cette activité étant très guidée, la complexité réside dans le positionnement des instructions proposées dans le programme et dans la compréhension du fonctionnement des listes.

Les listes n'étant pas spécifiquement demandées dans le référentiel de seconde, elles sont à utiliser avec précaution, il est aussi envisageable de créer des variables pour enregistrer l'effectif de chaque issue, mais cela alourdi considérablement le programme.



# Pile ou face communiquant avec Micro:bit (et python)

## DESCRIPTION

### Objectif

Le but de ce projet est de simuler une expérience aléatoire de lancer de pièce truquée avec une carte Micro:bit et d'envoyer par radio l'issue afin de centraliser les résultats. Cette activité permet de prendre en main facilement l'interface de développement en python

### Intérêt

Cette activité, idéale pour découvrir l'interface de programmation, propose d'utiliser python pour programmer une situation non-équiprobable et d'exploiter le module radio du Micro:bit.

**Production et traitement de données** Cette situation permet de travailler des notions de probabilités et de statistiques.

**Programmation Python** Cette activité nécessite que les élèves aient déjà été initié à Python, sans pour autant demandé une grande maîtrise.

### Matériel



- 1 x Micro:bit
- 1 x IDE programmation python (Mu) <https://codewith.mu/> ou interface de programmation ligne <https://python.microbit.org/v/1.1>

### Progression proposée

#### MÉTHODE

On propose ici d'aborder la problématique en deux temps :

##### 1) Tirage aléatoire et envoi des issues

Les élèves doivent modéliser une expérience aléatoire avec une probabilité donnée, et envoyer l'issue par radio.

##### 2) Réception et traitement des données

L'incrémentation des variables et le traitement des données est déclenchée par la réception d'un message radio.



## NIVEAU SIMPLE

### Activité élève



Durée

0,5 h



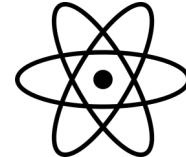
Public

2de

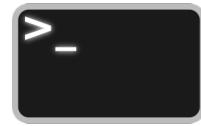


Maths

expérience aléatoire



Sciences



Algo

fonction;  
événement;  
boucle

### ACTIVITÉ

MISSION : PROGRAMME MICRO:BIT POUR SIMULER UN LANCER DE **Pile ou Face** TRUQUÉ !

On veut que notre **pièce virtuelle** ait 55% de chance de tomber sur "pile".

Le programme doit débuter avec les instructions ci-dessous afin :

- 1) d' importer les bibliothèques nécessaires;

```
from microbit import *
import random
import radio
```

- 2) allumer le mode radio; `radio.on()`

- 3) lancer une boucle infinie et détecter si le Micro:bit a été secoué;

```
while True:
    if accelerometer.was_gesture('shake'):
```

Il faut maintenant ajouter les fonctions pour :

- 1) effectuer un tirage aléatoire d'un nombre entre 1 et 100 avec `random.randint(,)` et comparer ce nombre à la probabilité voulue.
- 2) afficher P ou F **selon l'issue du tirage** avec `display.show('')` ;
- 3) envoyer la valeur 'P' ou 'F' par radio **selon l'issue du tirage** avec `radio.send('')`



Vérifie ton code avec



et flash-le sur la carte avec



## Notes pour l'enseignant

Ce premier niveau permet de se familiariser avec l'interface de l'IDE en produisant un premier programme fonctionnel et utile. Les notions de programmations utilisées sont :

- 1) l'appel de fonction
- 2) la boucle `while`
- 3) la boucle `if`

Dans cette première partie l'élève est aussi amené à concevoir une expérience aléatoire avec une probabilité donnée.

### MÉTHODE

Pour résoudre ce problème, il suffit de programmer les instructions de la façon suivante :

```
from microbit import *
import random
import radio

radio.on()

while True:
    if accelerometer.was_gesture('shake'):
        tirage = random.randint(1, 100)

        if tirage <= 55:
            display.show("P")
            radio.send('P')
        else:
            display.show("F")
            radio.send('F')
    sleep(10)
```

### REMARQUE

La variable `tirage` n'est pas indispensable, elle permet d'enregistrer le résultat du tirage aléatoire et de rendre le code plus lisible.

L'instruction `sleep(10)` permet de faire une pause de 10 ms dans le programme.

Afin de distinguer deux tirages successifs, on peut marquer une pause puis effacer l'écran à l'issue d'un tirage avec `display.clear()`, mais il peut-être amusant de montrer comment faire une animation simplement à partir d'une succession d'images.

Par exemple la fonction `anim()` ci-dessous pourra être définie au début du programme puis appeler avant chaque tirage.

```
while True:
    if accelerometer.was_gesture('shake'):
        tirage = random.randint(1, 100)
        anim()
```



## NIVEAU EXPERT

### Activité élève



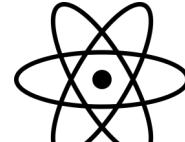
Durée



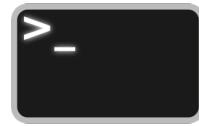
Public



Maths



Sciences

Algo  
fonction;  
événement;  
variable

0,5 h

2de

fréquences



#### ACTIVITÉ

MISSION : RÉCEPTIONNE LES DONNÉES ENVOYÉES PAR LES AUTRES MICRO:BIT ET EXPLOITE-LES POUR TRACER UN [graphique](#)

Dans l'activité précédente nous avons créer une [pièce virtuelle](#) qui affiche 'P' ou 'F' et qui envoie cette information par radio. Nous allons programmer le Micro:bit afin de recevoir, traiter et afficher les données.

Le programme précédent devra être complété avec les éléments suivants :

- 1) une déclaration de variables pour dénombrer les 'P' et les 'F', comme par exemple `p = 0`, qu'il faut écrire [avant](#) la boucle `while True:`
- 2) la réception d'un message radio (que l'on pourra enregistrer dans une variable 'issu') avec la fonction `radio.receive()` ;
- 3) l'incrémentation des variables décomptant les 'P' et les 'F' par exemple `p = p + 1` ;
- 4) le calcul de fréquence de 'P' et de 'F' (attention il faudra veiller à ne pas diviser par 0);
- 5) l'affichage des fréquences dans le plotter avec `print((a,b))` où '(a,b)' est le couple de fréquences de 'P' et de 'F' dont on souhaite voir l'évolution au cours du temps.



Vérifie ton code avec



et flashe-le sur la carte avec



Avant d'afficher le plotter avec , il faudra peut-être faire un reset de la carte.



## Notes pour l'enseignant

Cette deuxième activité nécessite impérativement l'usage de variables.

La difficulté peut provenir des différents cas de figure à traiter, mais à part l'instruction `else:`, il n'y a pas de nouveaux types d'instruction.

Les notions de programmations utilisées sont :

1) la déclaration et l'initialisation de variables

2) la boucle `if`, complété par `else`

Dans cette partie l'élève est aussi amené à calculer des fréquences, mais l'exécution de ce calcul doit être conditionné à la non nullité de la somme des variables.

### MÉTHODE

Pour recevoir et exploiter les données, il est possible de programmer les instructions de la façon suivante :

```
from microbit import *
import random
import radio

radio.on()

f = 0
p = 0

def anim():
    tab = [Image.ARROW_N, Image.ARROW_NE, Image.ARROW_E, Image.ARROW_SE,
           Image.ARROW_S, Image.ARROW_SW, Image.ARROW_W, Image.ARROW_NW]
    display.show(tab, delay=80)

while True:
    if accelerometer.was_gesture('shake'):
        tirage = random.randint(1, 100)
        anim()
        if tirage <= 55:
            display.show("P")
            radio.send('P')
        else:
            display.show("F")
            radio.send('F')
        sleep(10)

    issue = radio.receive()
    if issue == 'P':
        p += 1
    if issue == 'F':
        f += 1
    if p+f > 0:
        a = p/(p+f)
        b = f/(p+f)
    else:
        a, b = 0, 0
    print((a, b))
    sleep(10)
```



# Robot Maqueen et capteur de distance

## DESCRIPTION

### Objectif

Le but de cette séquence est d'**utiliser** et d'**étalonner** le capteur de distance ultrasonique du robot Maqueen. Ce capteur sera utilisé en tant que télémètre.

Après un étalonnage utilisant un modèle linéaire, nous proposons de programmer le robot afin qu'il s'arrête à une distance précise d'un obstacle.

### Intérêt

**Capteur piézoélectrique** La mise en œuvre du détecteur permet d'utiliser un capteur piézoélectrique de façon concrète.

**Propagation du son dans un milieu** Une explication théorique du fonctionnement du capteur permet de parler de la célérité du son dans l'air et de la notion d'ultrason.

**Régression linéaire** L'étalonnage permet de mettre en application sur un cas concret la régression linéaire.

**Mouvement du robot** La dernière partie permet de relier mouvement et détection, avec une automatisation basique.

### Matériel



— 1 x Micro:bit



— 1 x Maqueen (et son capteur intégré de distance ultrasonore)

— 1 x tableau magnétique

— 1 x règle graduée d'un mètre

— 1 x accès Internet :

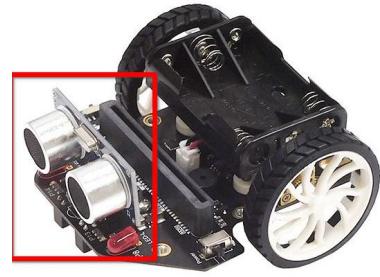
- 1 x accès Internet : IDE programmation par bloc <http://makecode.microbit.org/>
- 1 x accès en ligne à l'extension pour la programmation par bloc du robot Maqueen : <https://github.com/DFRobot/pxt-maqueen>



## Remarques

Le robot Maqueen est un robot pilotable à l'aide d'une carte Micro:bit.

Ce dernier est équipé de 2 roues motorisées, d'un buzzer, de 2 leds rouges de signalements, de 4 leds Neopixel, de 2 capteurs de lignes, d'un capteur de télécommande infrarouge et d'un **capteur de distance à ultrason**. C'est ce dernier capteur que nous allons utiliser, ainsi que les moteurs.



Le détecteur envoie une pulsation sonore de fréquence  $40\text{kHz}$  et mesure le temps que met l'onde pour faire l'aller-retour après réflexion sur un obstacle. Ce temps permet ainsi d'estimer une distance en reliant temps, célérité de l'onde et distance par la relation :  $d = \frac{c \times t}{2}$  avec la distance  $d$  en m, la célérité  $c$  du son dans l'air en m/s et  $t$  le temps en s.

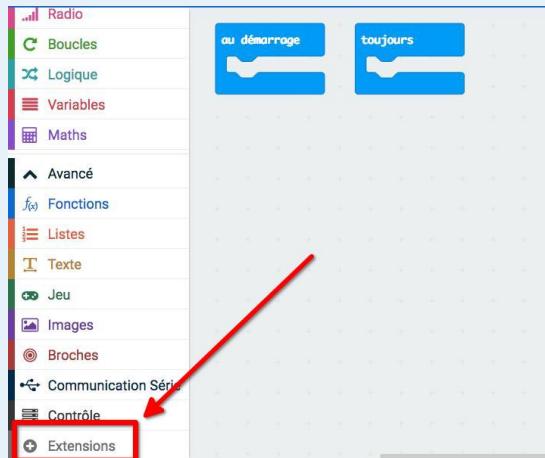
Pour chaque robot, un **étalonnage** est obligatoire à chaque séance car :

- la célérité dépend de la **température**
- l'**horloge interne** des Micro:bit n'est pas très précise.

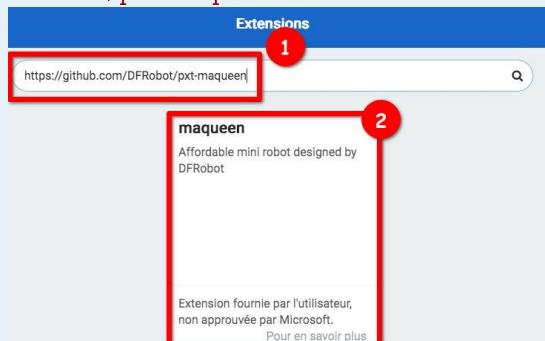
## MÉTHODE

### Installation de la bibliothèque Maqueen pour Makecode

- 1) Dans l'éditeur Makecode de Viens Coder, cliquer sur le bouton « Extension ».



- 2) Dans l'interface de recherche, entrer l'adresse suivante : <https://github.com/DFRobot/pxt-maqueen>



- 3) Cliquer sur *Maqueen* et une nouvelle famille de blocs apparaît.



## DESCRIPTION

The image shows a Scratch script for a Maqueen robot. The script is titled "maqueen". It contains the following blocks:

- on IR received message
- sensor unit cm
- Motor M1 dir CW speed 0
- Servo S1 angle 0
- Motor stop M1
- Read Patrol PatrolLeft
- led LEDLeft v ledswitch turnOn v

The "maqueen" category is highlighted with a red box and a red arrow points to it from the left sidebar.

4) Le bloc *sensor unit* permet de mesurer la distance, *Motor* à faire tourner les moteurs (**M1** gauche, **M2** droit) et *Motor stop* à les arrêter.



# NIVEAU INITIATION - UTILISATION DU CAPTEUR DE DISTANCE À ULTRASONS

## Activité élève



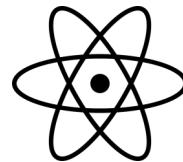
Durée  
0,5 h



Public  
2de



Maths



Sciences



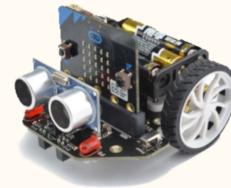
Algo



### ACTIVITÉ

Première programme avec le robot Maqueen.

TA MISSION : Sauras-tu utiliser le programme ci-dessous pour programmer ton robot Maqueen?



```

lorsque le bouton A est pressé
  montrer nombre arrondi
    sensor unit cm

```

## Notes pour l'enseignant

### MÉTHODE

Proposition de résolution :

- Télécharger le programme puis le téléverser sur la carte Micro:bit.
- Insérer la carte dans le robot Maqueen.
- Tester le programme en mesurant la distance (bouton A) à divers objets et chercher les limites du détecteur (angle de visée, objet non plat...)

### REMARQUE

Nous rappelons qu'il est indispensable d'installer l'extension Maqueen sur l'interface Make-code (<https://github.com/DFRobot/pxt-maqueen>)



## NIVEAU INTERMÉDIAIRE - ÉTALONNAGE DU CAPTEUR

### Activité élève



Durée

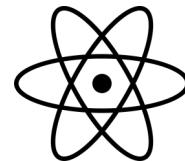
1,5 h



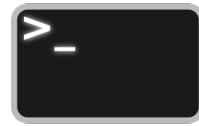
Public

2de, 1ère,  
Term

Maths

nuage de  
points,  
régression  
linéaire

Sciences

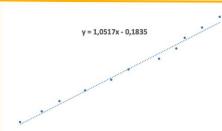


Algo



### ACTIVITÉ

Le capteur du robot Maqueen est très précis, mais malheureusement il donne souvent des valeurs faussées à cause de la chaleur et de la carte Micro:bit.



Pour corriger cela, il est indispensable de l'**étalonner**.

**TA MISSION :** Nous avons besoin de toi pour étalonner le robot Maqueen de la classe.

Si tu le désires, tu peux utiliser une règle, un obstacle (comme un tableau magnétique) et t'inspirer des deux indices ci-dessous...

$$y = 1,0517x - 0,1835$$

lorsque le bouton A est pressé

montrer nombre arrondi 1.0517 x sensor unit cm - 0.1835

### Notes pour l'enseignant

#### MÉTHODE

Proposition de méthode de résolution :

- 1) Placer le robot à 10 cm, 15 cm, 20 cm ... 100 cm d'un tableau de mécanique faisant obstacle.
- 2) Relever avec le capteur du robot les distances.



- 3) Renouveler l'opération pour chaque distance.
- 4) Dans un tableur, tracer la distance réelle en fonction de la distance mesurée par le capteur.
- 5) Modéliser le nuage de points par une régression linéaire.
- 6) Avec les blocs *Maths*, corriger le programme original pour afficher désormais une distance plus proche de la distance réelle prenant en compte la régression.



## NIVEAU EXPERT - ROBOT AUTONOME

### Activité élève



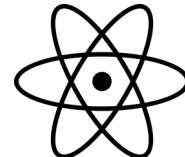
Durée  
1 h



Public  
1ère, Term



Maths



Sciences



Algo



**ACTIVITÉ**

TA MISSION : Programme le robot Maqueen pour qu'il avance quand on appuie sur le bouton A puis s'arrête à 20 cm d'un obstacle.



### Notes pour l'enseignant

#### MÉTHODE

Proposition de résolution :

```

lorsque le bouton A est pressé
  si arrondi (1.0517 * sensor unit cm) - 0.1835 > 20 alors
    Motor M1 dir CW speed 80
    Motor M2 dir CW speed 80
  tant que arrondi (1.0517 * sensor unit cm) - 0.1835 > 20
    faire
    Motor stop M1
    Motor stop M2
  
```

#### REMARQUE

Les élèves peuvent aussi programmer un **radar de recul** en utilisant aussi le **haut-parleur** du robot Maqueen.

```

toujours
  définir distance à arrondi (1.0517 * sensor unit cm) - 0.1835
  buzz (Hz) High G
  si distance < 10 alors
    tant que distance < 10
      faire
        définir distance à arrondi (1.0517 * sensor unit cm) - 0.1835
        buzz (Hz) 0 Hz
      sinon
        jouer ton (High G) pendant 1/16 temps
        pause (ms) distance * distance + 10
    
```



# Déterminer une vitesse avec MBot

## DESCRIPTION

### Objectif

Le but de ce projet, liées au programme de Maths-Sciences en CAP ou Bac Pro, est d'utiliser le robot MBot pour travailler sur le **mouvement** et la **mesure**. Outre la **vitesse**, qui est au centre du projet, une travail sur l'**erreur**, l'**incertitude** et les **statistique** est possible.

### Intérêt

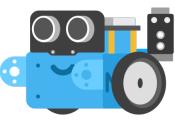
**Détermination de mesures physiques** Ces activités proposent d'étudier un questionnement essentiel de physique : **comment mesurer une vitesse**? Deux grandeurs sont donc à déterminer le plus précisément possible : une **distance** et un **temps**.

**Mouvement uniforme ou accéléré** Le robot MBot met un certain temps à démarrer : le mouvement est donc accéléré puis uniforme.

**Travail sur l'incertitude** Il est intéressant de déterminer la vitesse lorsque le **mouvement est uniforme**. Ainsi il sera obligatoire de lancer le chrono après le départ. Ce qui est intéressant car cela engendre un travail sur l'**erreur** et permet un calcul d'**incertitude**.

**Étude statistique** Un **travail statistique** pourra découler de ces mesures. Par exemple, il est intéressant de déterminer de moyenne et écart type.

### Matériel

- 1 x  MBot
- 1 x accès internet : IDE programmation par bloc <http://editor.makeblock.com/ide.html>

### Remarques

#### MÉTHODE

Dans la réalisation, nous suggérons d'utiliser un seul robot puis de demander aux élèves de mesurer temps et distance (donner 1 chronomètre par groupe ou utiliser les téléphones).

Récupérer ensuite l'ensemble des résultats (une dizaine) puis proposer de les exploiter afin de déterminer une **valeur précise et acceptable** de durée et distance. Il sera sans doute nécessaire de refaire les expériences au besoin.

L'objectif n'est pas d'obtenir des valeurs identiques, mais d'**analyser les conditions de l'expérience et de mesures** pour en améliorer la précision.



## NIVEAU INITIATION - PROGRAMMER LE DÉPLACEMENT DE MBOT

### Activité élève



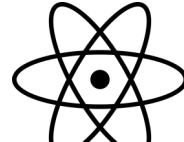
Durée



Public



Maths



Sciences



Algo

1 h

2de

translation

instructions de déplacement;  
temporisation;  
entrées clavier



#### ACTIVITÉ

Dans le cadre de la **conduite autonome d'un véhicule** le conducteur programme la vitesse à laquelle il souhaite rouler.

**TA MISSION :** Programmer le déplacement du robot MBot en utilisant le logiciel MBlock ?



#### ACTIVITÉ

**TA MISSION :** Stopper le déplacement du robot MBot si l'on presse la touche « Espace » du clavier.

### Notes pour l'enseignant

#### MÉTHODE

La plupart des élèves ne vont pas prévoir de stopper le robot.



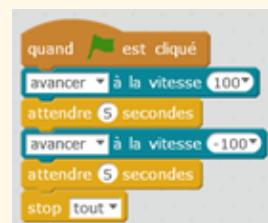
Ils devront trouver comment modifier leur programme.



#### REMARQUE

Concernant la **conduite autonome**, il est possible de montrer la première minute de [cette vidéo](#).

Il est possible d'utiliser des **vitesse différentes** que celles proposées ou encore des **vitesse négatives**.





## NIVEAU INTERMÉDIAIRE - DÉTERMINER LA VITESSE DE MBOT

### Activité élève



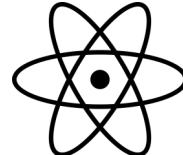
Durée



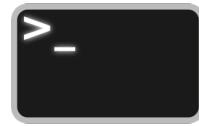
Public



Maths



Sciences



Algo

1 h

2de

conversions de grandeurs

mesure et incertitude; vitesse linéaire; temps; distance; mouvement



#### ACTIVITÉ

Karim et Julien programment le robot MBot pour qu'il se déplace. Julien annonce à Karim :

« Lorsqu'on sélectionne avancer à la vitesse 100 , cela ne correspond pas à 100 km/h mais à 100 m/s!!! »



TA MISSION : Comment vérifier l'**affirmation** de Julien ?

### Notes pour l'enseignant

#### MÉTHODE

En général, les élèves utilisent leur programme du fichier « MBot1 », leur téléphone (pour mesurer le temps) et une règle (pour déterminer une distance). La conversion des vitesses m/s et km/h est au centre de l'activité



#### REMARQUE

Très peu attendent que le robot ait déjà atteint une certaine vitesse avant de démarrer le chronomètre. Cela peut induire une réflexion sur le mouvement accéléré puis uniforme.



# Vitesse de rotation avec MBot

## DESCRIPTION

### Objectif

L'objectif du projet est de mettre en évidence, par l'expérimentation, l'influence du diamètre d'une roue sur la vitesse linéaire d'un véhicule.

Il faut arriver à observer que, pour un même distance, l'augmentation du diamètre des roues entraîne une diminution du temps de parcours du véhicule.

Autrement dit : la vitesse linéaire augmente lorsque le diamètre des roues augmente

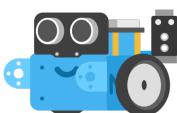
### Intérêt

Il est intéressant d'aborder le thème de sciences *Comment passer de la vitesse des roues à la vitesse du véhicule* avec un robot. En effet :

**Expérimentation et démarche scientifique** l'élève manipule tout au long de l'activité, fait des hypothèses et doit valider ses choix

**Objet attrayant** le robot Mbot reste attractif pour les élèves et sa programmation permet de valoriser le travail de l'élève

### Matériel



— 1 x MBot

— 1 x accès internet : IDE programmation par bloc <http://editor.makeblock.com/ide.html>

### Remarques

#### MÉTHODE

L'activité propose de mettre en évidence le lien entre variation du diamètre et variation de vitesse linéaire.

Cette activité peut être prolongée par une vérification de la relation entre vitesse linéaire ( $V$ ), rayon ( $r$ ) et fréquence de rotation ( $N$ ) :  $V = 2 \times \pi \times r \times N$ .

Pour cela il faudra utiliser un **tachymètre** permettant une mesure de la fréquence de rotation.



## NIVEAU INITIATION

### Activité élève



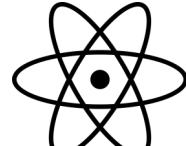
Durée



Public



Maths



Sciences



Algo

2 h

2de

mouvement,  
durée,  
distance,  
vitesse linéaireinstructions  
déplacement

#### ACTIVITÉ

Hélias veut booster son robot en changeant les roues. Il a trouvé sur Internet de jolies roues jaunes pouvant s'adapter sur le MBot.



Ces nouvelles roues sont plus grandes et Hélias s'exclame : "En plus, lorsque je sélectionnerai la vitesse 100, mon robot ira plus vite !".

TA MISSION : Comment vérifier si Hélias a raison, le robot ira-t-il vraiment plus vite ?

### Notes pour l'enseignant

#### MÉTHODE

- 1) La distance doit être **suffisamment grande** pour mettre en évidence la variation de vitesse et **suffisamment petite** pour éviter une dérive du robot.
- 2) Montrer l'intérêt de faire plusieurs fois la **même mesure** afin de mettre en évidence la variabilité d'une mesure.
- 3) Proposition de code :



#### REMARQUE

Les élèves utilisent leur téléphone (pour mesurer le temps) et une règle (pour déterminer une distance). L'utilisation de la barre d'espace pour arrêter le robot n'est pas indispensable mais est **très pratique** pour gérer l'arrêt du robot.



# Variation de vitesses avec MBot

## DESCRIPTION

### Objectif

Le but de ce projet est **d'abord** de déterminer par l'expérimentation la nature accéléré, décéléré ou uniforme du mouvement d'un mobile qui avance à l'aide d'un algorithme. Dans un **deuxième temps**, l'élève doit modifier l'algorithme pour animer le mobile d'un mouvement décéléré puis valider expérimentalement l'hypothèse.

### Intérêt

Il est intéressant d'aborder le thème de sciences *Comment peut-on décrire le mouvement d'un véhicule* avec un robot. En effet :

**Expérimentation et démarche scientifique** l'élève manipule tout au long de l'activité, fait des hypothèses et doit valider ses choix

**Analyse et modification d'un algorithme** l'algorithme de départ n'étant pas assez simple, l'élève doit émettre une hypothèse à partir de l'analyse de l'algorithme et ensuite le modifier pour changer le mouvement du robot

### Matériel



- 1 x MBot
- 1 x accès internet : IDE programmation par bloc <http://editor.makeblock.com/ide.html>

### Remarques

#### MÉTHODE

- 1) Lecture de l'algorithme par les élèves
- 2) Hypothèses des élèves (individuellement et/ou par groupe) sur la nature du mouvement
- 3) Proposition de protocole permettant de valider l'hypothèse (mesure de vitesse à partir de durée et distance)
- 4) Manipulation / Observation
- 5) Validation / Conclusion

#### MÉTHODE

- 1) Modification de l'algorithme pour obtenir un mouvement décéléré
- 2) Manipulation / Observation
- 3) Validation / Conclusion



## NIVEAU INITIATION - NATURE DU MOUVEMENT

### Activité élève



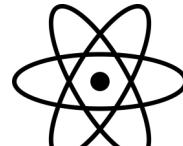
Durée



Public



Maths



Sciences



Algo

instruction  
déplacement,  
boucles, incrémentation



#### ACTIVITÉ

Un **véhicule autonome** est apte à avancer sans l'intervention du conducteur.



Il emploie des algorithmes pour décider de l'action à réaliser sur les commandes du véhicule.

Le robot MBot se déplace suivant le même principe. Il est aussi équipé d'un émetteur et d'un récepteur à ultrason qui permet de déterminer la distance du robot par rapport à un obstacle.

Le code ci-dessous a été envoyé dans le robot.

```

quand vert est cliqué
supprimer l'élément tout de la liste temps
supprimer l'élément tout de la liste distance
mettre d à distance mesurée par le capteur ultrasons du Port 3
mettre t à 1
mettre v à 100
répéter indéfiniment
  avancer à la vitesse v
  attendre 1 secondes
  ajouter t à temps
  mettre d à distance mesurée par le capteur ultrasons du Port 3
  ajouter d à distance
  ajouter à v 20
  ajouter à t 1

```

TA MISSION : Détermine la nature du mouvement.



## NIVEAU EXPERT - MOUVEMENT DÉCÉLÉRÉ

### Activité élève



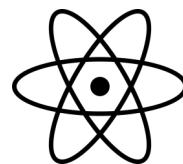
Durée



Public



Maths



Sciences



Algo

instruction  
déplacement,  
boucles, incrémentation



#### ACTIVITÉ

Quand le robot exécute l'algorithme ci-dessous, la distance en centimètre entre l'obstacle et le robot s'affiche en fonction du temps (en seconde).



```

quand vert est cliqué
supprimer l'élément tout de la liste temps
supprimer l'élément tout de la liste distance
mettre d à distance mesurée par le capteur ultrasons du Port 3
mettre t à 1
mettre v à 100
répéter indéfiniment
    avancer à la vitesse v
    attendre 1 secondes
    ajouter t à temps
    mettre d à distance mesurée par le capteur ultrasons du Port 3
    ajouter d à distance
    ajouter à v 20
    ajouter à t 1
}

```

TA MISSION : Modifie le code pour que ton robot MBot ait un mouvement décéléré. Vérifie alors expérimentalement que ton robot obéit !



# Normal ou truqué ? avec STM32

## Éducation

### DESCRIPTION

#### Objectif

Cette activité propose à l’élève de travailler sur la fluctuation d’échantillonnage. Sans accéder au code, seulement en manipulant la carte STM32 Éducation, l’élève doit déterminer si le jeu téléversé est un jeu normal ou truqué...

#### Intérêt

L’intérêt de cette activité est de pouvoir, en toute liberté, créer des situations équiprobales ou non. Même s’il est possible d’utiliser de vrais dés truqués, l’utilisation d’une carte STM32 Éducation permet un éventail très larges de situations.

Ici le choix a été fait de procéder à un tirage aléatoire (ou non;) d’un nombre entre 0 et 9 (inclus).

#### Matériel



- 1 x STM32 Éducation (facultatif car le simulateur peut suffire)
- 1 x accès internet : IDE programmation par bloc <https://makecode.st.com/>

#### Progression

L’activité se déroule en 2 temps :

**Analyse du programme** Dans cette partie, l’élève doit étudier les deux programmes possibles (truqué ou non) puis répondre à une série de questions de probabilités

**Expérience aléatoire** Ensuite l’élève manipule la carte STM32 Éducation et effectue un certain nombre d’expériences aléatoires. Dans cette partie l’analyse des échantillons et la création d’une représentation graphique permettra de conclure sur le type de programme téléversé.



## ACTIVITÉ

### Activité élève



Durée

2 h

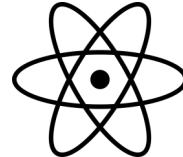


Public

2de ; term



Maths

fluctuation  
d'échantillage

Sciences



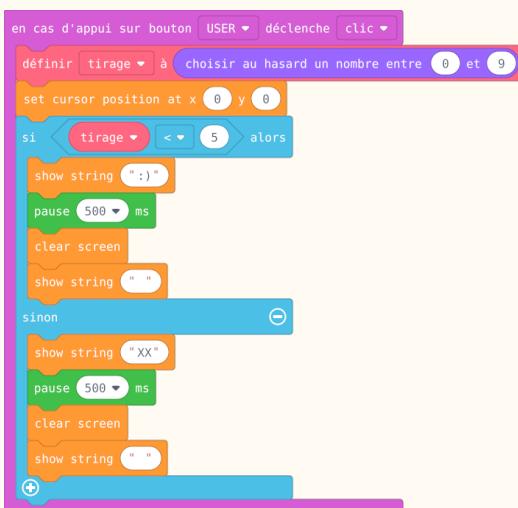
Algo



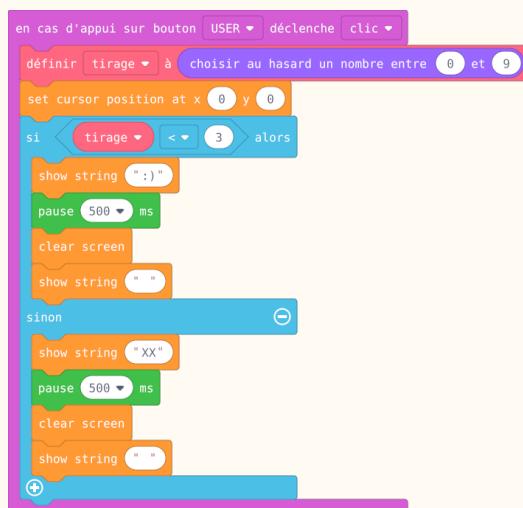
### ACTIVITÉ

NORMAL OU TRUQUÉ ?

Sur votre carte est téléchargé un des deux programmes de jeu suivant.  
Quand le joueur gagne, il a un smiley sourire, sinon il a une croix.



Jeu normal



Jeu truqué

### Analyse du programme

- 1) Si on choisit un nombre entier au hasard entre 0 et 9, combien y-a-t-il d'issues possibles ?
- 2) Si pour gagner il faut obtenir un nombre entre 0 et 4, combien y-a-t-il d'issues favorables ?
- 3) Calculer la probabilité de gagner avec la version normale du programme. Donner le résultat sous forme de fraction, de nombre décimal et de pourcentage.
- 4) Avec le même raisonnement, calculer la probabilité de gagner avec la version truquée du programme. Donner le résultat sous forme de fraction, de nombre décimal et de pourcentage.

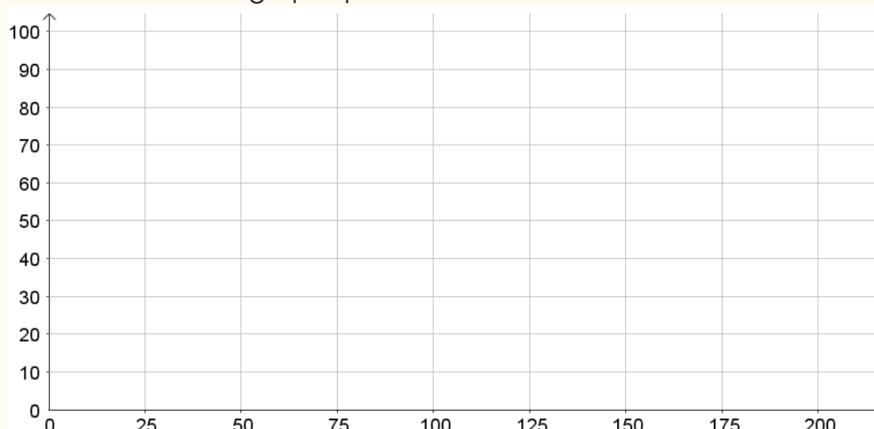


### Analyse du programme

- 5) Jouer des parties par séquence de 25 parties et noter G pour gagner et P pour perdu, dans votre cahier.
- 6) En utilisant vos résultats, compléter le tableau ci-dessous. **Attention!** Bien prendre son temps pour compter.

	25 parties	50 parties	75 parties	100 parties	125 parties	150 parties	175 parties	200 parties
Nombre de victoires								
Proportion de victoires en %	$\frac{\text{Nombre de victoires}}{25}$							

- 7) Compléter le graphique correspondant à la dernière ligne du tableau : placer un point par colonne, relier les points par des segments et légendrer les axes. Donner un **titre** au graphique.



- 8) Analyser le graphique et conclure sur le programme qui est téléchargé dans votre carte. Argumenter avec une ou plusieurs phrases.

### Notes pour l'enseignant

#### REMARQUE

Le code interactif est accessible en ligne :

Jeu normal      <http://url.univ-irem.fr/z>



Jeu truqué      <http://url.univ-irem.fr/A>





# Pile ou face avec STM32 Éducation

## DESCRIPTION

### Objectif

<https://www.overleaf.com/8232568565gwgstdqzhjfk> Le but de ce projet est de simuler une expérience aléatoire de lancer de pièce avec une carte STM32 Éducation.

### Intérêt

Bien évidemment, travailler avec une carte STM32 Éducation n'exclut pas de réaliser des expériences aléatoires réelles (pièces, dés, etc.). Cependant, il est très intéressant pour l'enseignant d'utiliser des STM32 Éducation dans cette partie du programme.

**Simplicité de la situation.** La situation est très simple à expliquer et les élèves comprennent le but à atteindre. L'absence de difficulté mathématique rend cette situation particulièrement simple à mettre en œuvre.

**Motivation des élèves.** L'envie de programmer un objet connecté est grande pour les élèves. Cette façon de programmer, **utile, concrète et appliquée**, leur correspond parfaitement.

**De nombreuses solutions/améliorations possibles.** Comme pour bien des projets, il y a plusieurs façons d'arriver à la solution. Par ailleurs, les élèves peuvent apporter ou proposer de nombreuses améliorations. La programmation par bloc, exempte de difficulté syntaxique, est particulièrement adaptée à la créativité.

**Travail mathématique sur la modélisation.** Cette compétence n'est pas facile à mettre en œuvre. Ici, l'absence de difficultés mathématiques rend ce travail beaucoup plus accessible à tous.

**Initiation aux différents capteurs** Aux travers cette activité on peut commencer à utiliser l'accéléromètre (en secouant la carte) , le magnétomètre (en approchant un aimant ) etc..

### Matériel



- 1 x STM32 Éducation (facultatif car le simulateur peut suffire)
- 1 x accès internet : IDE programmation par bloc <https://makecode.st.com/>



## PROGRESSION PROPOSÉE

### MÉTHODE

Voici la progression proposée aux élèves. L'objectif est atteint dès la troisième étape. Les étapes suivantes, de niveau expert, sont du bonus.

#### 1) Initiation - Premier modèle.



Afficher un 0 ou un 1 de façon aléatoire. Pour préparer la suite et pour plus de lisibilité, nous proposons dès maintenant de créer et d'utiliser la fonction `nTirage`.

#### 2) Intermédiaire - Pile ou Face.



Afficher un p ou un f de façon aléatoire.

#### 3) Intermédiaire - 16 tirages.



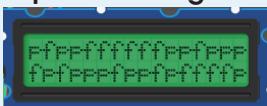
Afficher d'un coup 16 tirages aléatoires. Nous donnons à l'élève une nouvelle fonction : `efface`. Elle permet d'effacer l'écran LCD avant chaque tirage (sinon on voit rien). De plus, le fond d'écran change de couleur de façon aléatoire (c'est fun !)

#### 4) Expert - Initialisation.



Ajouter la fonction `initialisation` qui lance 10 fois la fonction `efface` (ce qui fait une petite animation de couleurs). Utiliser cette fonction pour initialiser STM32 Éducation au démarrage et lors d'un appui long sur le bouton.

#### 5) Expert - 32 tirages.



Afficher maintenant les tirages sur 2 lignes.

#### 6) Expert - Enregistrer les effectifs.



Création de deux variables `nP` et `nF` qui stockent le nombre total de piles et de faces obtenues. Afficher les valeurs de ces variables sur l'écran. Pour cela, effectuer seulement 10 tirages aléatoires par ligne (au lieu de 16) afin de laisser la place pour l'affichage de 5 caractères.



## NIVEAU INITIATION - PREMIER MODÈLE

### Activité élève



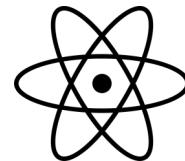
Durée



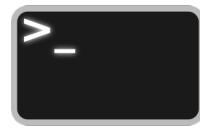
Public



Maths



Sciences



Algo

0,5 h

2de

expérience aléatoire

affichage ;  
événement ;  
fonction

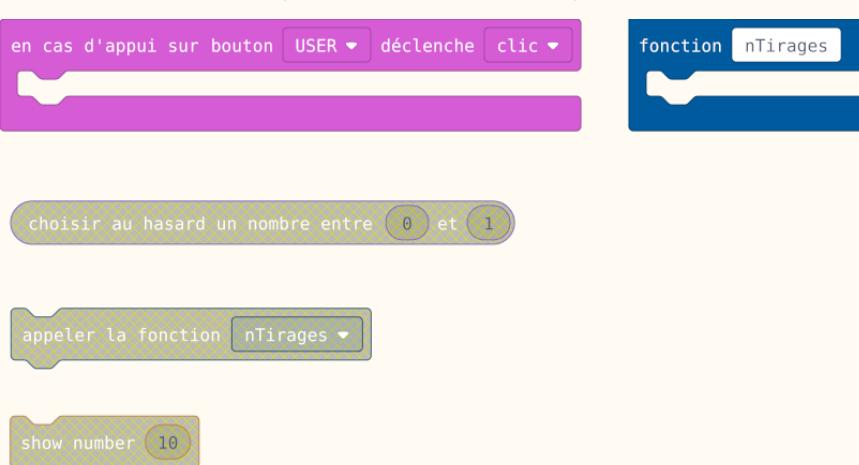
#### ACTIVITÉ

MISSION : utiliser vos STM32 Éducation pour jouer à [Pile ou Face!](#)



**Connectez vous** sur l'interface de programmation de votre STM32 Éducation (<https://makecode.st.com/>).

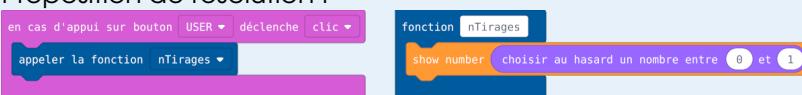
**Utiliser les instructions** ci-dessous pour simuler un jeu de pile ou face.



### Notes pour l'enseignant

#### MÉTHODE

Proposition de résolution :



#### REMARQUE

La proposition de solution **interactive** est accessible en ligne [https://makecode.com/\\_EMYMOYft11A6](https://makecode.com/_EMYMOYft11A6).



## NIVEAU INTERMÉDIAIRE - PILE OU FACE

### Activité élève



Durée

0,5 h



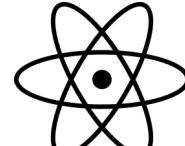
Public

2de



Maths

expérience aléatoire



Sciences



Algo

affichage ;  
événement ;  
fonction ;  
condition

### ACTIVITÉ

MISSION : améliorer votre simulateur



Utiliser les blocs ci-dessous pour remplacer les 0 et les 1 par des p ou f.

en cas d'appui sur bouton USER déclenche clic

appeler la fonction nTirages

fonction nTirages

show string "p"

choisir au hasard un nombre entre 0 et 1

show string "f"

0 = 0

si vrai alors

sinon



## NOTES POUR L'ENSEIGNANT

### MÉTHODE

Proposition de résolution :

```
en cas d'appui sur bouton USER déclenche clic
appeler la fonction nTirages
```

```
fonction nTirages
si choisir au hasard un nombre entre 0 et 1 = 0 alors
    show string "p"
sinon
    show string "f"
```

### REMARQUE

La proposition de solution **interactive** est accessible en ligne [https://makecode.com/\\_MysE701XAHpo](https://makecode.com/_MysE701XAHpo).



## NIVEAU INTERMÉDIAIRE - 16 TIRAGES

### Activité élève



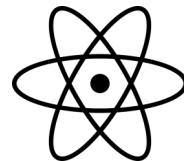
Durée



Public



Maths



Sciences



Algo

affichage ;  
événement ;  
fonction ;  
condition ;  
boucle

0,5 h

2de

expérience  
aléatoire

### ACTIVITÉ

MISSION FINALE : utiliser la puissance de l'électronique pour effectuer 16 tirages simultanés !



En vous aidant des instructions proposées, modifier votre programme pour afficher simultanément **16 tirages sur la première ligne**. N'oubliez d'ajouter la fonction efface qui permet d'effacer l'écran LCD de votre STM32 Éducation.

```

en cas d'appui sur bouton USER déclenche clic
appeler la fonction nTirages

fonction efface
set cursor position at x 0 y 0
clear screen
set backlight color hue choisir au hasard un nombre entre 0 et 255 sat 200 val 200
show string " "

fonction nTirages
si choisir au hasard un nombre entre 0 et 1 = 0 alors
show string "p"
sinon
show string "f"
+
répéter 16 fois faire
appeler la fonction efface

```



## MÉTHODE

Proposition de résolution :

```
en cas d'appui sur bouton USER déclenche clic
    appeler la fonction efface
    appeler la fonction nTirages
```

```
fonction efface
    set cursor position at x 0 y 0
    clear screen
    set backlight color hue choisir au hasard un nombre entre 0 et 255 sat 200 val 200
    show string ""
```

```
fonction nTirages
répéter 16 fois
    faire
        si choisir au hasard un nombre entre 0 et 1 = 0 alors
            show string "p"
        sinon
            show string "f"
```

## REMARQUE

La proposition de solution **interactive** est accessible en ligne [https://makecode.com/\\_711WJtAupKRX](https://makecode.com/_711WJtAupKRX).



## NIVEAU EXPERT - INITIALISATION

### Activité élève



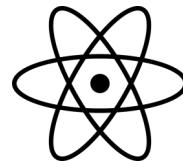
Durée



Public



Maths



Sciences



Algo

0,5 h

2de

expérience aléatoire

affichage ;  
événement ;  
fonction ;  
condition ;  
boucle



### ACTIVITÉ

#### MISSION EXPERTE 1/3 : initialiser la carte



Afin de préparer les améliorations à venir, créez une nouvelle fonction initialisation. Elle doit exécuter 10 fois la fonction efface, ce qui fait très jolie avec l'écran coloré ! Cette fonction initialisation sera appelée au démarrage de votre STM32 Éducation et aussi lors d'un appui long sur le bouton.

```

en cas d'appui sur bouton USER déclenche clic
    appeler la fonction efface
    appeler la fonction nTirages

fonction initialise
    appeler la fonction initialise

fonction efface
    set cursor position at x 0 y 0
    clear screen
    set backlight color hue choisir au hasard un nombre entre 0 et 255 sat 200 val 200
    show string " "

répéter 10 fois faire pause 100 ms
    appeler la fonction initialise

fonction nTirages
    répéter 16 fois faire
        si choisir au hasard un nombre entre 0 et 1 = 0 alors
            show string "p"
        sinon
            show string "f"
        +
    au démarrage
    en cas d'appui sur bouton USER déclenche clic long

```



## MÉTHODE

Proposition de résolution :

```

en cas d'appui sur bouton USER déclenche clic
    appeler la fonction efface
    appeler la fonction nTirages

fonction initialise
    répéter 10 fois
        faire appeler la fonction efface
        pause 100 ms

fonction efface
    set cursor position at x 0 y 0
    clear screen
    set backlight color hue choisir au hasard un nombre entre 0 et 255 sat 200 val 200
    show string " "

fonction nTirages
    répéter 16 fois
        faire si choisir au hasard un nombre entre 0 et 1 = 0 alors
            show string "p"
        sinon
            show string "f"
        + 

au démarrage
    appeler la fonction initialise

en cas d'appui sur bouton USER déclenche clic long
    appeler la fonction initialise

```

## REMARQUE

La proposition de solution **interactive** est accessible en ligne [https://makecode.com/\\_c3mKWL3fFX6A](https://makecode.com/_c3mKWL3fFX6A).



## NIVEAU EXPERT - 32 TIRAGES

### Activité élève



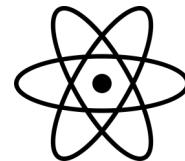
Durée



Public



Maths



Sciences



Algo

affichage ;  
événement ;  
fonction ;  
condition ;  
boucle

0,5 h

2de

expérience  
aléatoire

### ACTIVITÉ

MISSION EXPERTE 2/3 : Plus de tirages !



Utiliser les instructions ci-dessous pour effectuer les 32 tirages : 16 sur la première ligne de l'écran et 16 sur la seconde.

```

en cas d'appui sur bouton USER déclenche clic
    appeler la fonction efface
    appeler la fonction nTirages

fonction initialise
    répéter 10 fois
        faire appeler la fonction efface
        pause 100 ms

fonction efface
    set cursor position at x 0 y 0
    clear screen
    set backlight color hue choisir au hasard un nombre entre 0 et 255 sat 200 val 200
    show string " "

fonction nTirages
    répéter 16 fois
        faire
            si choisir au hasard un nombre entre 0 et 1 = 0 alors
                show string "p"
            sinon
                show string "f"
            +
au démarrage
    appeler la fonction initialise
en cas d'appui sur bouton USER déclenche clic long
    appeler la fonction initialise

set cursor position at x 0 y 1
set cursor position at x 0 y 0

appeler la fonction nTirages

```



## MÉTHODE

Proposition de résolution :

```
en cas d'appui sur bouton USER déclenche clic
    appeler la fonction efface
    set cursor position at x 0 y 0
    appeler la fonction nTirages
    set cursor position at x 0 y 1
    appeler la fonction nTirages
```

```
fonction initialise
    répéter 10 fois
        faire appeler la fonction efface
        pause 100 ms
```

### REMARQUE

La proposition de solution **interactive** est accessible en ligne [https://makecode.com/\\_F8MKCVD2jDY8](https://makecode.com/_F8MKCVD2jDY8).

```
fonction efface
    set cursor position at x 0 y 0
    clear screen
    set backlight color hue choisir au hasard un nombre entre 0 et 255 sat 200 val 200
    show string ""
```

```
fonction nTirages
    répéter 16 fois
        faire si choisir au hasard un nombre entre 0 et 1 = 0 alors
            show string "p"
        sinon
            show string "f"
        fin
    fin
```

```
au démarrage
    appeler la fonction initialise
    appeler la fonction initialise
```

```
en cas d'appui sur bouton USER déclenche clic long
    appeler la fonction initialise
```



## NIVEAU EXPERT - ENREGISTRER LES EFFECTIFS

### Activité élève



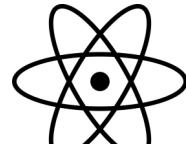
Durée



Public



Maths



Sciences



Algo

affichage;  
événement;  
fonction;  
condition;  
boucle;  
variables

0,5 h

2de

expérience  
aléatoire

### ACTIVITÉ

```
en cas d'appui sur bouton USER déclenche clic
appeler la fonction efface
set cursor position at x 0 y 0
appeler la fonction nTirages
set cursor position at x 0 y 1
appeler la fonction nTirages
```

```
fonction initialise
répéter 10 fois
faire
appeler la fonction efface
pause 100 ms
```



```
fonction efface
set cursor position at x 0 y 0
clear screen
set backlight color hue choisir au hasard un nombre entre 0 et 255 sat 200 val 200
show string
```

```
fonction nTirages
répéter 10 fois
faire
si choisir au hasard un nombre entre 0 et 1 alors
show string "P"
sinon
show string "F"
```

```
au démarrage
appeler la fonction initialise
en cas d'appui sur bouton USER déclenche clic long
appeler la fonction initialise
```

```
fonction effectifs
définir nP à 0
changer nF par 1
show value "nP" à nP
définir nF à 0
changer nP par 1
show value "nF" à nF
```

```
set cursor position at x 11 y 0
appeler la fonction effectifs
```

```
set cursor position at x 11 y 1
appeler la fonction effectifs
```

**MISSION EXPERTE 3/3 : La totale !**  
Pour finir, créer 2 variables nP et nF qui vont compter le nombre total de piles et de faces obtenus.  
Sur chaque ligne, n'effectuer plus que 10 tirages afin de laisser la place pour afficher les valeurs de ces variables.



## MÉTHODE

Proposition de résolution :

```

en cas d'appui sur bouton USER déclenche clic
    appeler la fonction efface
    set cursor position at x 0 y 0
    appeler la fonction nTirages
    set cursor position at x 0 y 1
    appeler la fonction nTirages
    appeler la fonction effectifs

fonction initialise
    répéter (10) [faire [appeler la fonction efface
        pause (100) ms
        définir [nP] à 0
        définir [nF] à 0
    ] appeler la fonction effectifs]
]

fonction efface
    set cursor position at x 0 y 0
    clear screen
    set backlight color [hue choisir au hasard un nombre entre 0 et 255 sat 200 val 200]
    show string " "

fonction nTirages
    répéter (10) [
        faire [si [choisir au hasard un nombre entre 0 et 1 = 0] alors
            changer [nP] par 1
            show string ("P")
        ]
        sinon
            changer [nF] par 1
            show string ("F")
        end
    ]
]

fonction effectifs
    set cursor position at x 11 y 0
    show value ("nP") = [nP]
    set cursor position at x 11 y 1
    show value ("nF") = [nF]
]

```

### REMARQUE

La proposition de solution **interactive** est accessible en ligne [https://makecode.com/\\_VEy38D1iC07g](https://makecode.com/_VEy38D1iC07g).