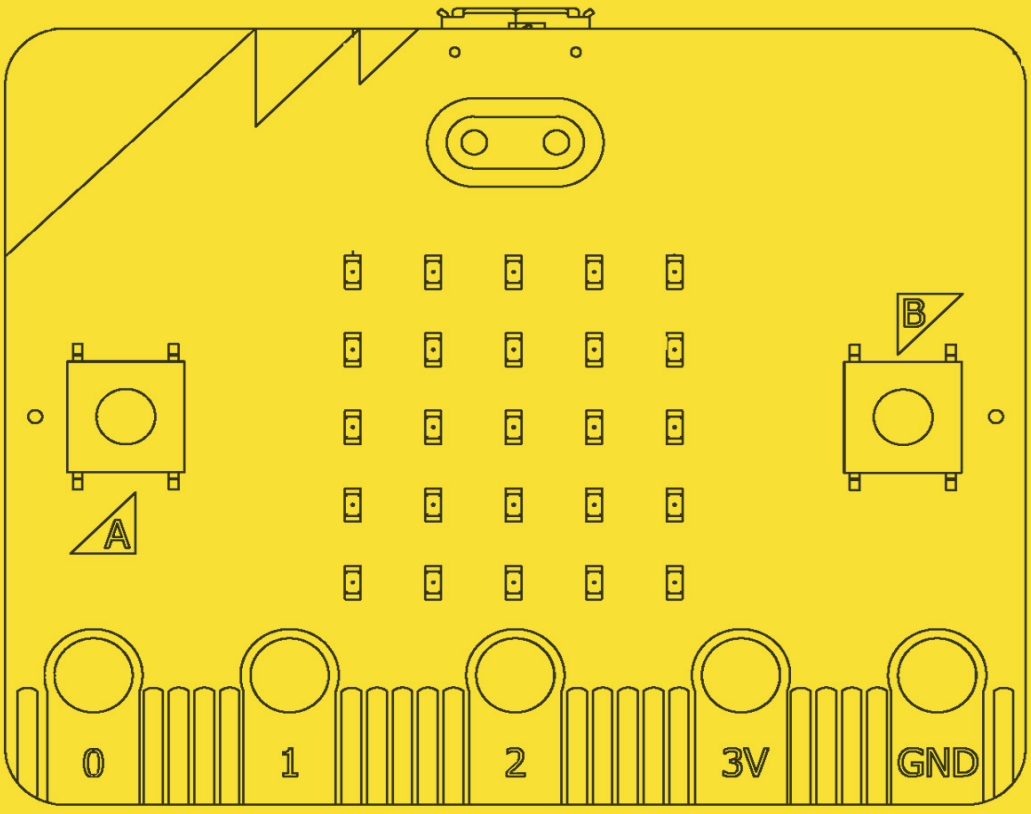
LE LIBRE SAVOIR ©

RIDHA NAIRI

ACTIVITÉS AVEC
la carte micro:bit



Programmation avec makeCode et microPython

TABLE DES MATIERES

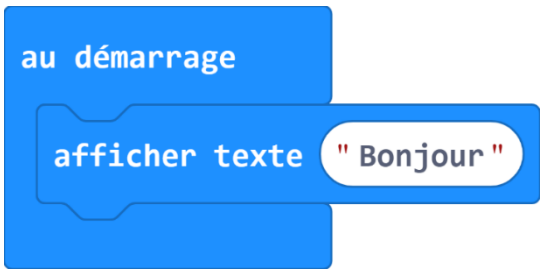
ACTIVITE 1

AFFICHER UN TEXTE A CHAQUE DEMARRAGE (VERSION 1).

La première fonction, toute simple, de la carte micro:bit consiste à afficher un texte qui va défiler lettre par lettre sur l'écran.

- Vous allez donc utiliser l'instruction "au démarrage" ainsi que l'instruction "afficher texte" qui se trouve dans le menu "Base".
- Vous pouvez placer le texte de votre choix. Nous vous proposons le classique « Bonjour» pour commencer.
- Vous constatez que la simulation se déclenche simultanément. Elle permet ainsi de vérifier le bon fonctionnement du programme.
- À présent, transférez ce programme dans votre carte micro:bit et visualisez le résultat.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



MicroPython est une version optimisée et allégée de Python 3 prévue spécifiquement pour les microcontrôleurs et les systèmes électroniques embarqués dans des machines.

DETAILLONS CES INSTRUCTIONS :

Dans l'instruction JavaScript `basic.showString("Bonjour")` nous appliquons la méthode `showString` à l'objet `basic`.
Dans l'instruction MicroPython `display.scroll("Bonjour")` nous appliquons la méthode `scroll` à l'objet `display`.

Les objets sont des conteneurs de données qui possèdent des propriétés (ce que sont ces données) et des méthodes (ce qu'elles font).

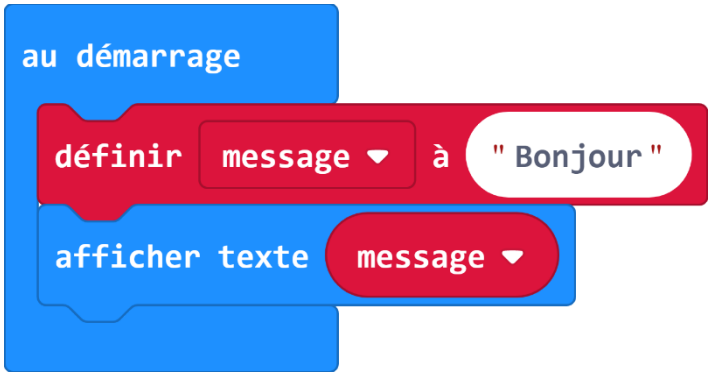
ACTIVITE 2

AFFICHER UN TEXTE A CHAQUE DEMARRAGE (VERSION 2).

Modifier la précédente solution (Activité 1) par la création d'une variable.

- Dans le menu "Variables", définissez une variable (appelée "message" ici).
- Dans l'onglet JavaScript initialiser la variable "message" avec la chaîne de caractères "Bonjour".
- Afficher ensuite le contenu de la variable "message".

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



Les variables sont des conteneurs dans lesquels se trouvent des données.

Une variable est caractérisée par : son nom (un identificateur unique), son type (un nombre entier ou réel, une chaîne de caractères, un booléen…) et sa valeur.

Le type booléen est formé de deux valeurs notées `false` et `true` (`False` et `True` Avec MicroPython).

On obtient un résultat de type booléen quand on est amené à comparer des expressions entre elles, au moyen des opérateurs relationnels (`<`, `<=`, `>`, `>=`, `==`, `!=`) et des opérateurs logiques (avec JavaScript : `&&`, `||`, `!`) (avec Python : `and`, `or`, `not`).

Une variable doit respecter quelques règles de syntaxe incontournables :

1. Le nom de la variable ne peut être composé que de lettres, majuscules ou minuscules, de chiffres et du symbole souligné « _ » (appelé underscore).
2. Le nom de la variable ne peut pas commencer par un chiffre.
3. Les langages JavaScript et MicroPython sont sensibles à la casse.

DETAILLONS CES INSTRUCTIONS :

Dans l'instruction JavaScript `let message = "Bonjour"` `let` permet de déclarer une variable dont la portée est celle du bloc courant et éventuellement d'initialiser sa valeur.

Dans notre exemple `message = "Bonjour"` une variable portant comme identifiant (nom) `message` initialisée à une chaine de caractères (`"Bonjour"`).

L'opérateur d'affectation représenté par le signe égale `=` affecte la valeur de l'opérande placé à droite à l'opérande de gauche.

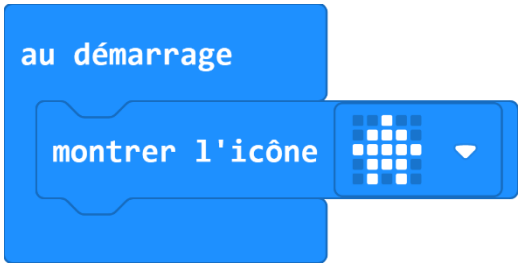
ACTIVITE 3

AFFICHER UNE ICONE PREDEFINIE A CHAQUE DEMARRAGE.

Il est possible d'afficher des icônes sur l'écran constitué d'une matrice de 5 x 5 LEDs grâce à l'instruction "montrer l'icône" du menu "Base". MakeCode vous propose 40 icônes à afficher.

Contrairement au texte, l'icône reste affichée.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



Sous MicroPython les icônes prédéfinies sur micro:bit font partie de l'objet `Image`, sont :

`HEART, HEART_SMALL, HAPPY, SMILE, SAD, CONFUSED, ANGRY, ASLEEP, SURPRISED, SILLY, FABULOUS, MEH, YES, NO, CLOCK12, CLOCK1, CLOCK2, CLOCK3, CLOCK4, CLOCK5, CLOCK6, CLOCK7, CLOCK8, CLOCK9, CLOCK10, CLOCK11, ARROW_N, ARROW_NE, ARROW_E, ARROW_SE, ARROW_S, ARROW_SW, ARROW_W, ARROW_NW, TRIANGLE, TRIANGLE_LEFT, CHESSBOARD, DIAMOND, DIAMOND_SMALL, SQUARE, SQUARE_SMALL, RABBIT, COW, MUSIC_CROTCHET, MUSIC_QUAVER, MUSIC_QUAVERS, PITCHFORK, XMAS, PACMAN, TARGET, ALL_CLOCKS, ALL_ARROWS, TSHIRT, ROLLERSKATE, DUCK, HOUSE, TORTOISE, BUTTERFLY, STICKFIGURE, GHOST, SWORD, GIRAFFE, SKULL, UMBRELLA` et `SNAKE`.

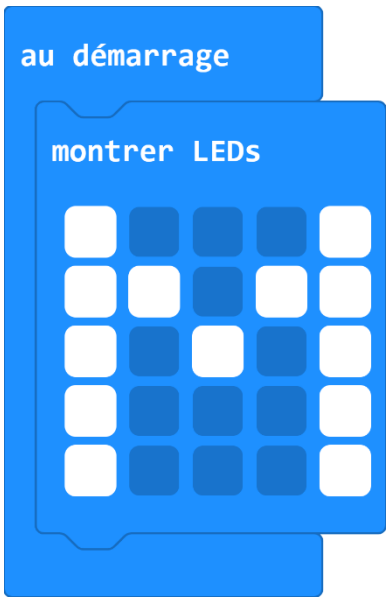
Code avec JavaScript	Code avec MicroPython
`basic.showIcon(IconNames.House)`	`display.show(Image.HOUSE)`

ACTIVITE 4

CREER ET AFFICHER VOTRE PROPRE IMAGE A CHAQUE DEMARRAGE.

Via le menu "Base", il est possible de créer sa propre icône avec l'instruction "montrer LEDs" Il suffit de cliquer sur les cases représentant les LEDs pour dessiner l'icône souhaitée.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



Pour créer une image sous MicroPython, il suffit de fixer l'intensité (comprise entre 0 et 9) de chacun des 25 pixels de l'écran.

Voici un exemple d'image personnalisée :

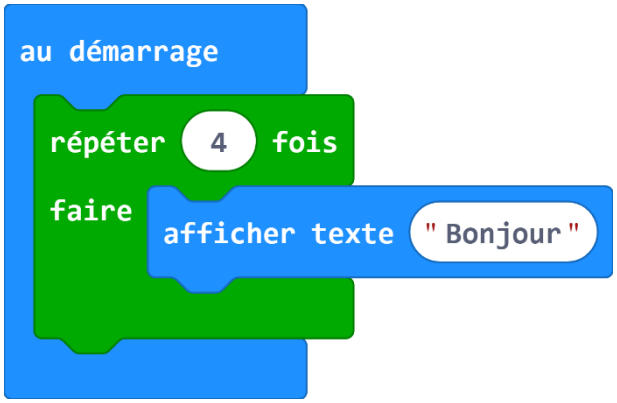
```
mon_image = Image(
      '03930:'
      '09490:'
      '95159:'
      '09490:'
      '03930')

display.show(mon_image)
```

ACTIVITE 5

AFFICHER UN TEXTE 4 FOIS A CHAQUE DEMARRAGE.

Si vous souhaitez que le texte s'affiche un nombre précis de fois, il faut utiliser l'instruction "répéter … fois faire" via le menu "Boucle", dans laquelle vous indiquez le nombre de répétitions.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



Une boucle sert à répéter une portion de code en fonction d'une condition (appelée aussi prédicat ou test).
On peut créer une boucle grâce au mot-clé `for` et `while`.

Syntaxe de la boucle for avec JavaScript	Syntaxe de la boucle for avec MicroPython

`for (initialisation ; condition ; incrémentation) {`
`  ...`
`}`
	`for variable in séquence:`
	`  ...`

DETAILLONS CES INSTRUCTIONS JAVASCRIPT :

```
for (let index = 0; index < 4; index++) {
    basic.showString("Bonjour")
}
```

Dans l'instruction JavaScript `for`, Il y a trois instructions, chacune séparée par un point-virgule :
1. La première est l'initialisation : cette première instruction est utilisée pour préparer la variable `index`. Dans notre cas, on initialise la variable `index` à `0`.
2. La seconde est la condition (une expression booléenne) : c'est elle qui dit si la boucle doit être répétée ou non.
   - Tant que la condition est vraie (le test renvoie `true`), la boucle `for` continue.
   - Si la condition n'est plus vérifiée (`index` vaut donc 4 dans cet exemple ; le test renvoie `false`), la boucle se termine.
3. Enfin, il y a l'incrémentation : cette dernière instruction est exécutée à la fin de chaque tour de boucle pour mettre à jour la variable `index`.

La quasi-totalité du temps on fera une incrémentation, mais on peut aussi faire une décrémentation (`variable--`) ou encore n'importe quelle autre opération (`variable += 2`).

DETAILLONS CES INSTRUCTIONS MICROPYTHON :

```
for index in range(4):
    display.scroll("Bonjour")
```

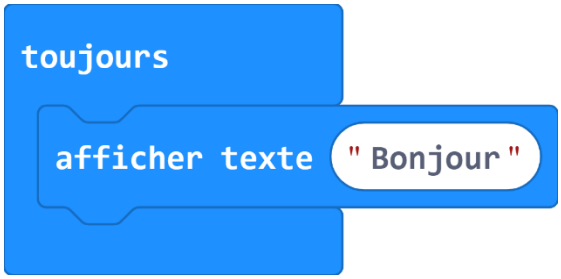
- La fonction `range()` permet d'énumérer le nombre de passage dans la boucle `for`.
- La fonction `range()` génère par défaut une séquence de nombres entiers de valeurs croissantes, et différant d'une unité.
- Si vous appelez `range()` avec un seul argument, la liste contiendra un nombre de valeurs égal à l'argument fourni, mais en commençant à partir de zéro (c'est-à-dire que `range(n)` génère les nombres de `0` à `n-1`).
- Dans notre exemple la variable index va recevoir à chaque itération respectivement les entiers 0, 1, 2 et 3.

ACTIVITE 6

FAIRE DEFILER UN TEXTE (VERSION 1).

Le texte de l'activité 1 ne s'est affiché qu'une seule fois, mais il est possible de reproduire cet affichage avec une instruction bien utile qui permet de répéter indéfiniment une action, l'instruction "toujours". Le texte va alors défiler sans s'arrêter.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



Sous MicroPython il est possible de faire défiler les différentes informations les unes après les autres sur l'écran de la carte micro:bit. Les données affichées ne restent donc pas fixes pendant un certain temps, elles défilent sur l'écran.

Pour ce faire, vous devrez utiliser la méthode `scroll` dont la syntaxe est la suivante :

```
display.scroll(string, delay = 300, wait = False, loop = True, monospace = False)
```

- `string` correspond à l'information à afficher, à savoir un texte, une liste, des images ...
- `delay` correspond à la vitesse de défilement, plus ou moins rapide.
- `wait` est un booléen : si sa valeur est True, on passe à l'instruction suivante uniquement si tout a été affiché ; si sa valeur est False, on passe à l'instruction suivante mais l'affichage se fait en même temps.
- `loop` est également un booléen : si sa valeur est True, l'affichage se fait en boucle.
- `monospace` est un booléen : si sa valeur est True, on affiche un pixel entre chaque caractère à afficher.

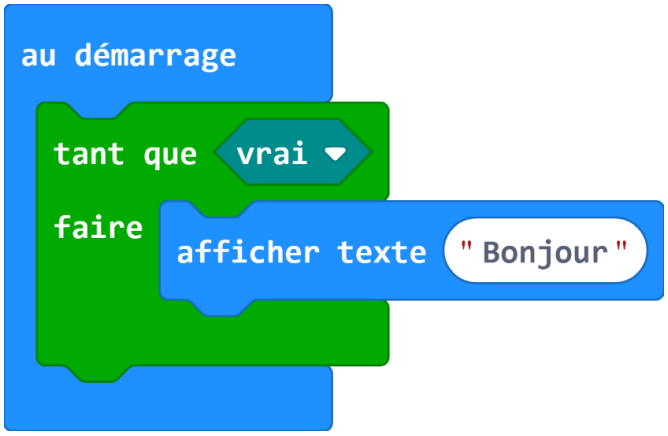
ACTIVITE 7

FAIRE DEFILER UN TEXTE (VERSION 1).

Refaite la même activité précédente (Activité 6) sans l'utilisation du bloc "toujours".

La solution consiste dans la création d'une boucle dite "infinie" avec le bloc "tant que … faire" du menu "Boucles".

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



L'instruction `while` permet de répéter une portion de code tant qu'une condition est vraie (`condition` est une variable ou expression booléenne).

Syntaxe de la boucle while avec JavaScript	Syntaxe de la boucle while avec MicroPython

`while (condition) {`
`  ...`
`}`
	`while (condition) :`
	`  ...`

1. La condition est testée (on dit aussi évaluée).
2. Si la condition est vraie (`condition==true`), l'instruction ou les instructions du bloc sont exécutées, puis on recommence à l'étape 1 (test de la condition).
3. Si la condition est fausse (`condition==false`), l'instruction ou les instructions du bloc ne sont pas exécutées et on passe aux instructions suivantes (après la structure de contrôle).

Code avec JavaScript	Code avec MicroPython

`while (true) {`
`  basic.showString("Bonjour")`
`}`
	`while (True) :`
	`  display.scroll("Bonjour")`

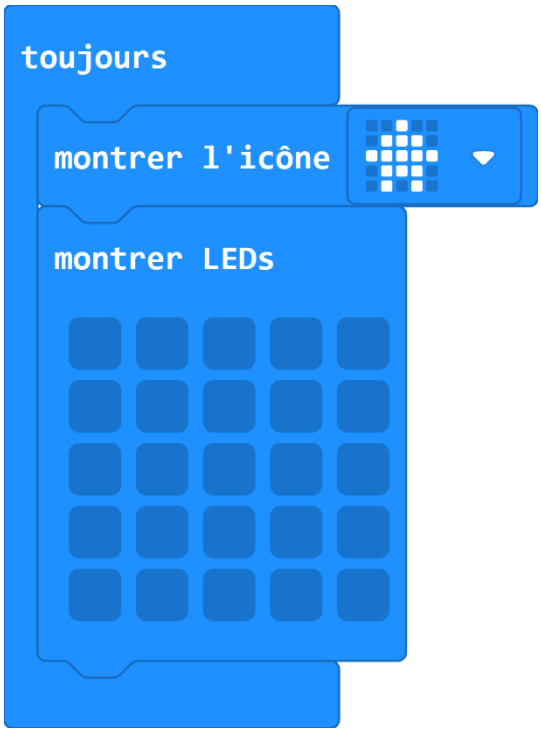
Il est possible d'interrompre une boucle `for` ou `while` par l'instruction `break`.
Il est possible d'interrompre une itération (elle ne provoque pas la fin de la boucle) par l'instruction `continue`.

ACTIVITE 8

FAIRE CLIGNOTER UNE ICONE.

Pour faire clignoter une icône, vous devez l'afficher, puis l'éteindre en montrant une matrice vide et recommencer.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.

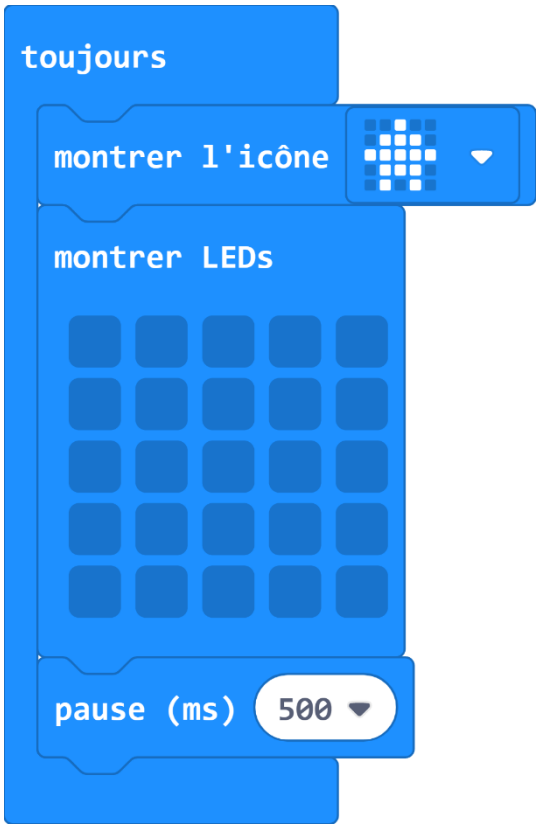


ACTIVITE 9

FAIRE CLIGNOTER UNE ICONE PLUS LENTEMENT (VERSION 1).

Si l'on veut un clignotement plus lent, il suffit d'ajouter une pause.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.

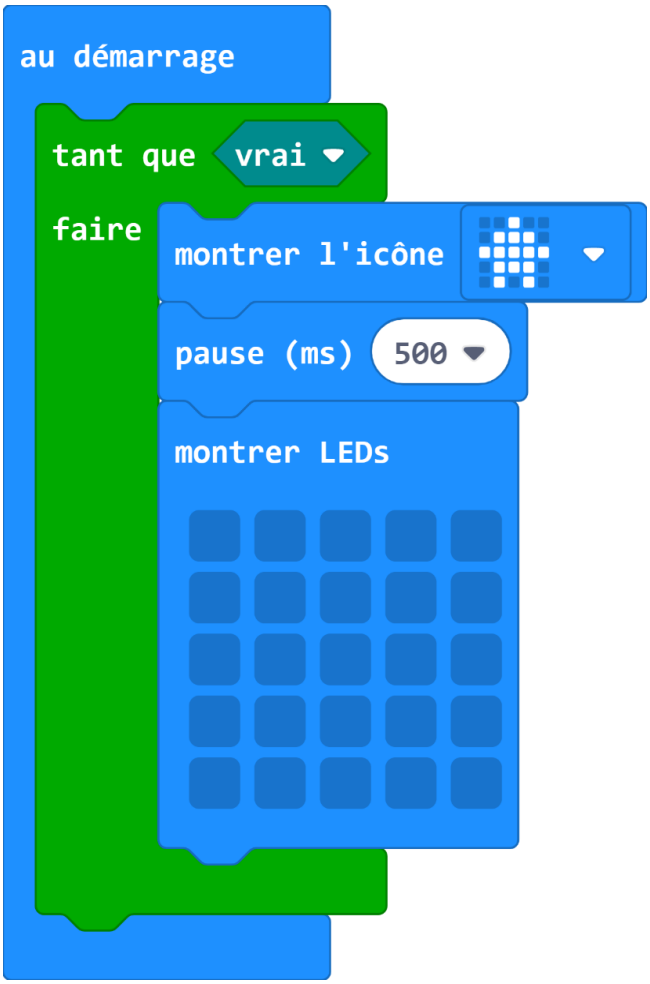


ACTIVITE 10

FAIRE CLIGNOTER UNE ICONE PLUS LENTEMENT (VERSION 2).

Modifier la précédente solution (Activité 9) sans l'utilisation du bloc "toujours".

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



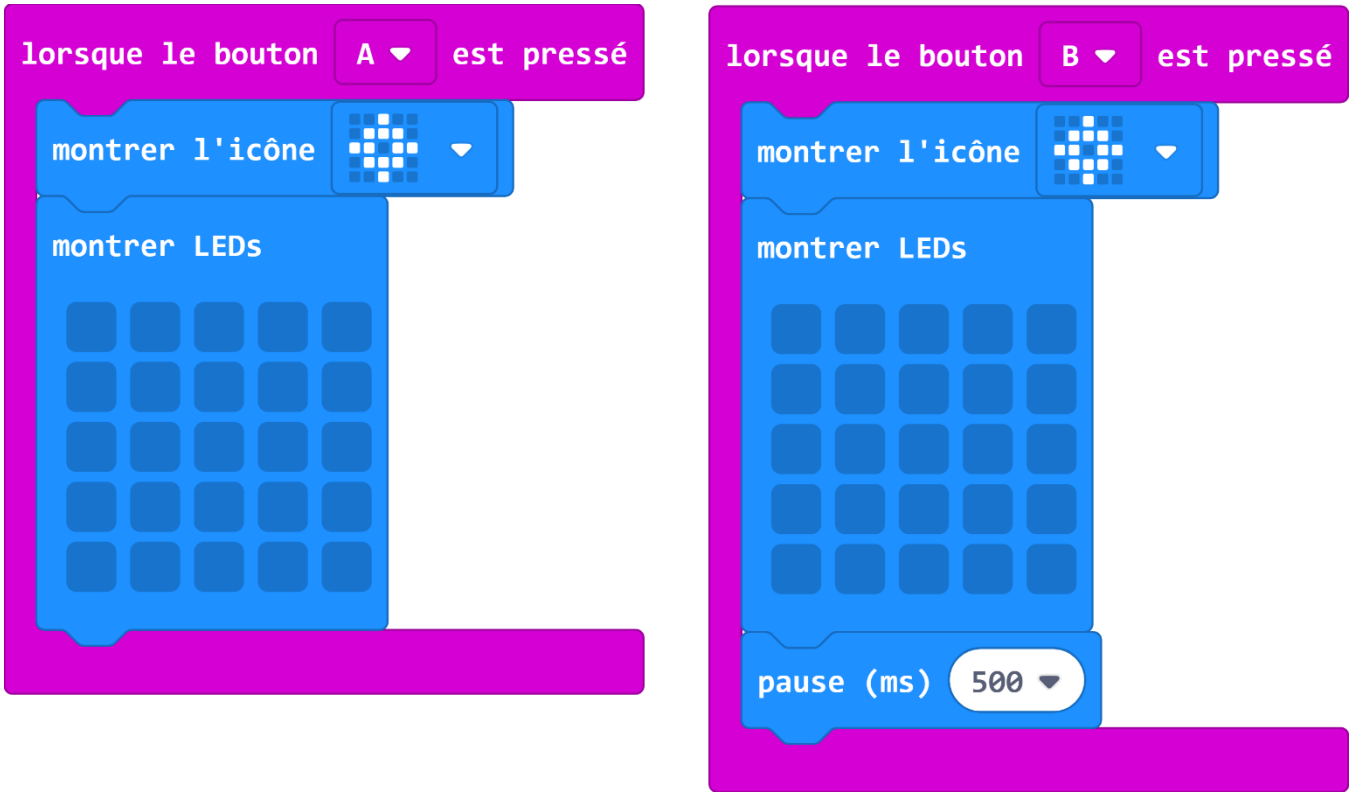
ACTIVITE 11

UTILISER LES BOUTONS POUR AFFICHER UNE ICONE.

La carte micro:bit dispose de deux boutons permettant de déclencher des actions : le bouton A, à gauche de la carte, et le bouton B, à droite. Les instructions concernant les boutons se trouvent dans le menu "Entrées".

Il est possible de programmer l'appui simultané sur les deux boutons, ce qui correspond à la fonction + : il faut que le bouton A + le bouton B soient appuyés ensemble pour que les instructions s'exécutent.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.

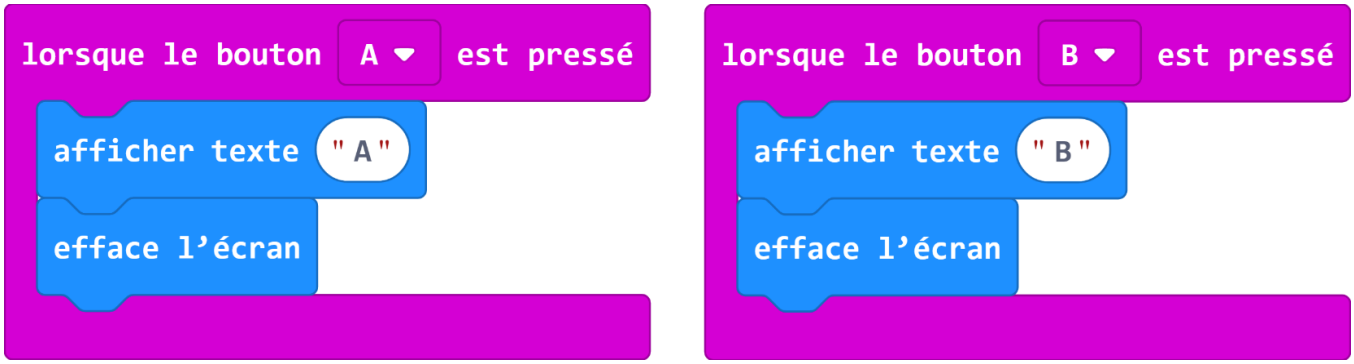


ACTIVITE 12

UTILISER LES BOUTONS POUR AFFICHER UN TEXTE (VERSION 1).

Afficher la lettre A lorsque le bouton A est pressé, et la lettre B si le bouton B est pressé sinon effacer l'écran.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.

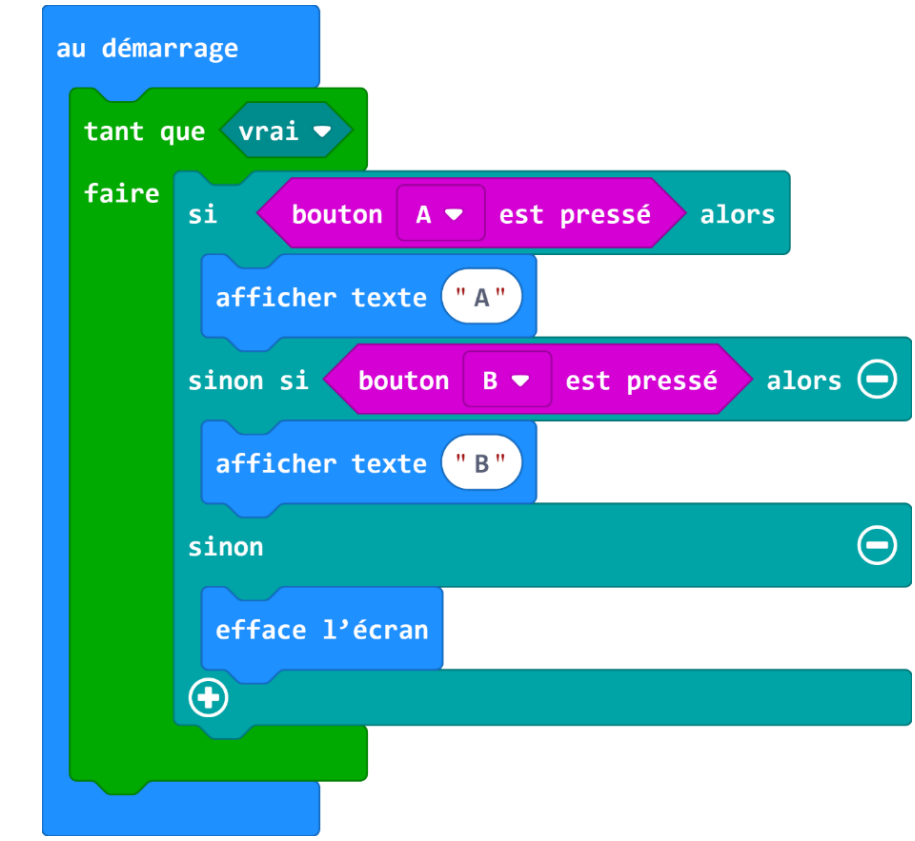


ACTIVITE 13

UTILISER LES BOUTONS POUR AFFICHER UN TEXTE (VERSION 2).

Toujours Afficher la lettre A lorsque le bouton A est pressé, et la lettre B si le bouton B est pressé sinon effacer l'écran.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



La structure conditionnelle `if` permet d'exécuter une portion de code seulement si une condition donnée est vérifiée (`==true`) (`condition` est une variable ou expression booléenne).

Syntaxe avec JavaScript	
Forme minimale	**Forme complète**

`if (condition) {`
`  ...`
`}`
	`if (condition) {`
	`  ...`
	`} else {`
	`  ...`
	`}`

Syntaxe avec MicroPython	
Forme minimale	**Forme complète**

`if (condition) :`
`  ...`
	`if (condition) :`
	`  ...`
	`else :`
	`  ...`

N.B. Sous MicroPython l'instruction « `elif (condition) :` » est équivalente à la contraction de « `else if (condition) :` ».

ACTIVITE 14

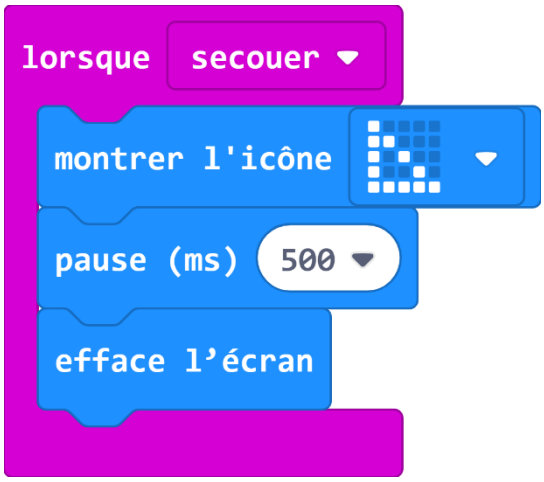
SECOUER LA CARTE POUR AFFICHER UNE ICONE.

La carte micro:bit dispose d'un accéléromètre qui offre plusieurs possibilités. Il faut utiliser l'instruction "lorsque secouer" du menu "Entrées" pour le déclencher.

Par exemple, lorsque vous secouez la carte, celle-ci affiche une icône, puis l'efface au bout d'une demi-seconde.

Cette instruction possède de multiples réglages permettant de gérer des situations plus précises, comme « logo vers le haut », « incliner à droite », etc., ainsi que des vitesses d'accélération.

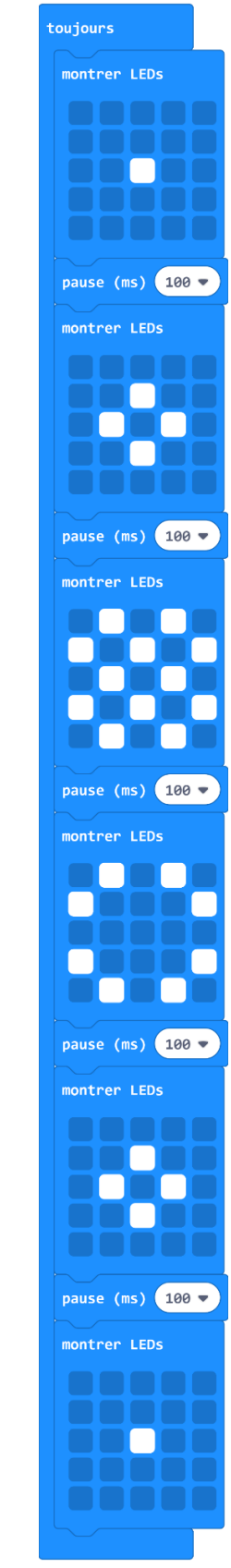
Refaite la même activité en utilisant le langage de programmation **MicroPython**.



ACTIVITE 15

CREER ET AFFICHER UNE ETOILE SCINTILLANTE.

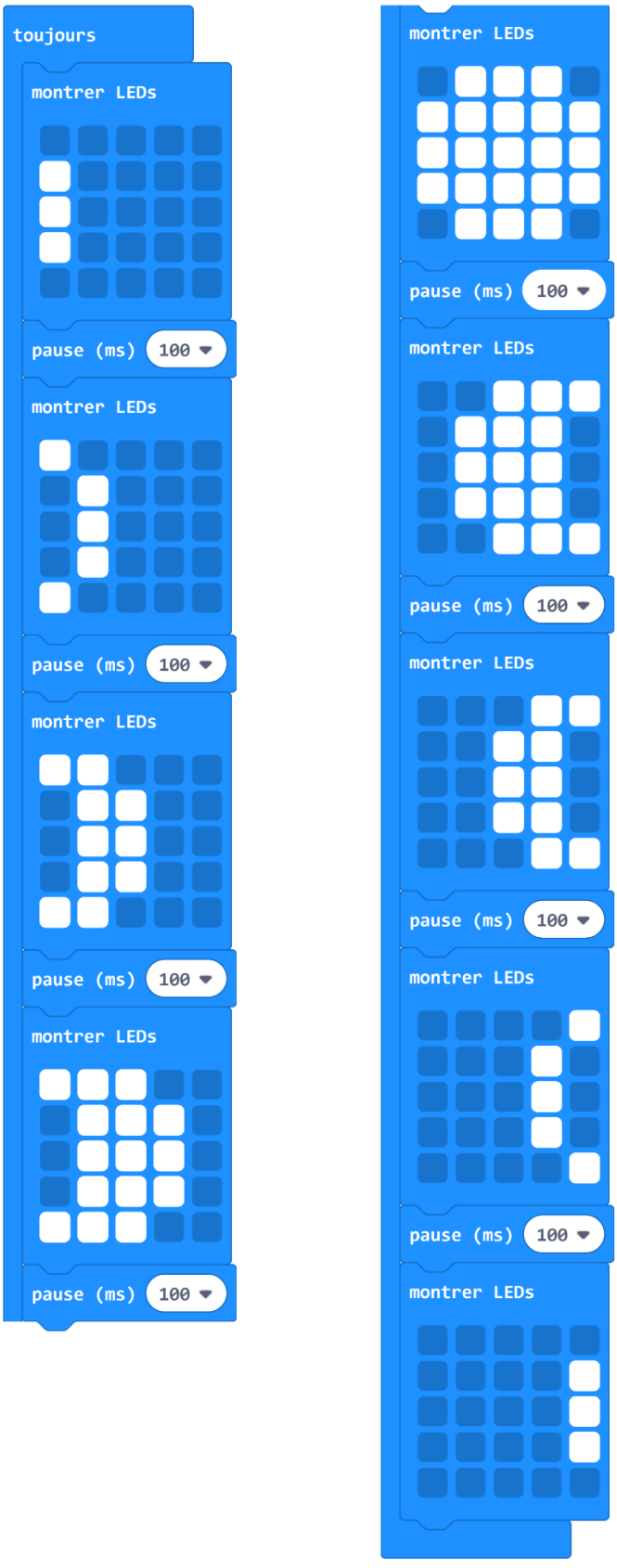
Refaite la même activité en utilisant le langage de programmation **MicroPython**.



ACTIVITE 16

CREER ET AFFICHER LES DIVERSES PHASES DE LA LUNE.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



ACTIVITE 17

CREER ET AFFICHER UNE NUIT ETOILEE.

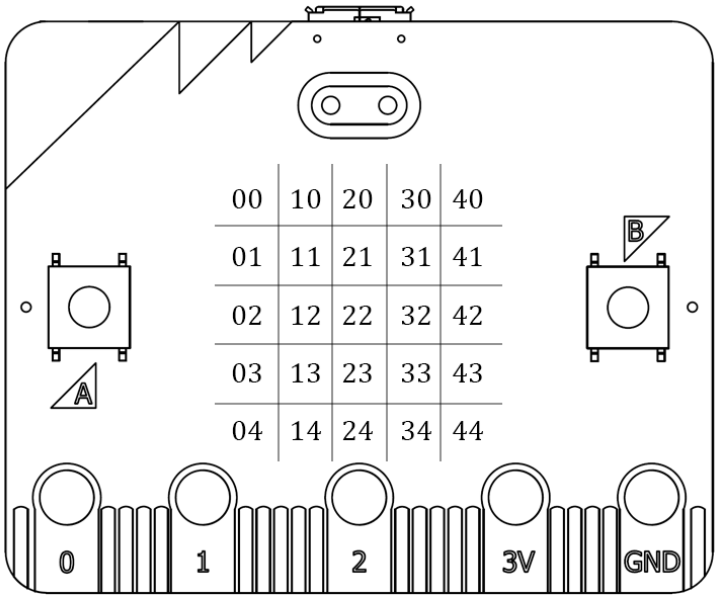
Le programme fait varier l'ètat des LEDs (allumer/éteindre) en X et en Y d'une manière aléatoire.

Via le menu "Led", utiliser les blocs "allumer x … y …" et "étendre x … y …".

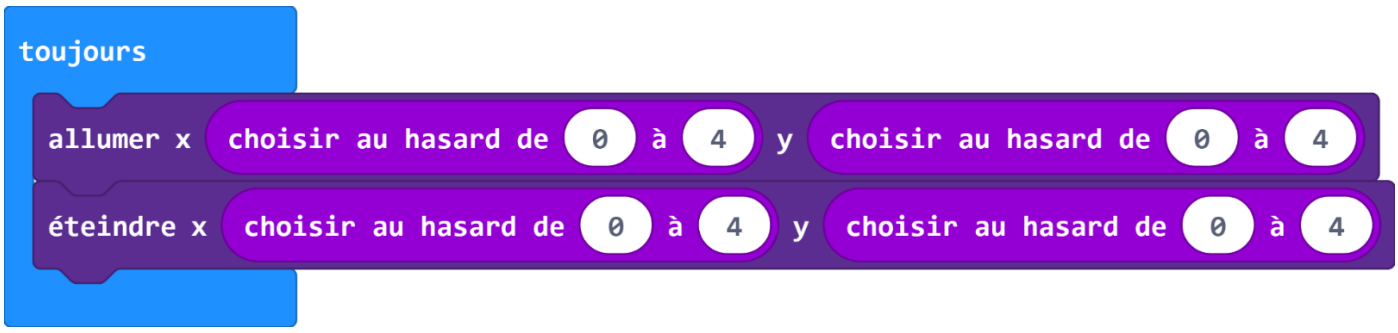
Pour les coordonnées des LEDs, l'axe des **X** est en horizontal, celui des **Y** en vertical.

Par exemple :

- **X** à 0 et **Y** à 0 correspond à 0-0.
- **X** à 3 et **Y** à 4 correspond à 3-4.



Refaite la même activité en utilisant le langage de programmation **MicroPython**.



Sous MicroPython il est possible de commander le flux lumineux des différentes LED de l'écran de la carte micro:bit avec la commande suivante :

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

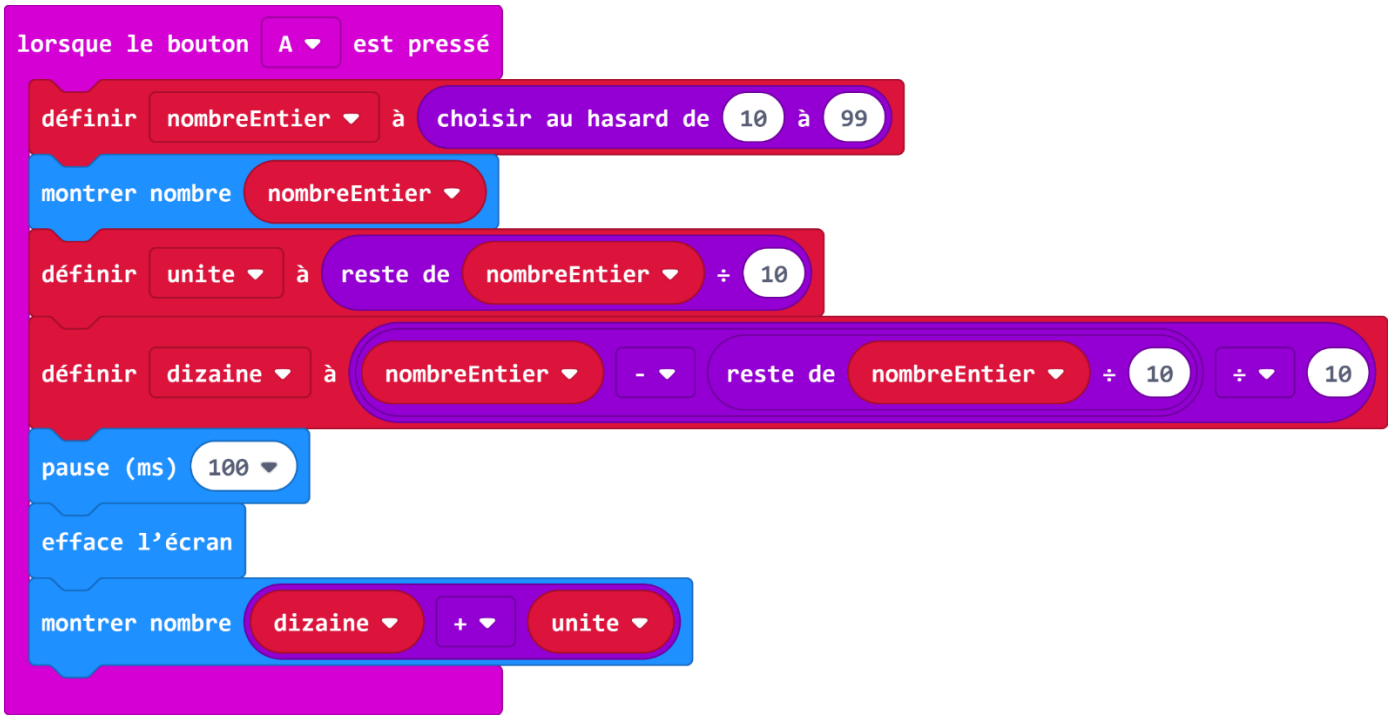
- `x` est la position sur la ligne;
- `y` la position dans la colonne;
- `intensité` correspond au flux lumineux : si sa valeur est 0, la LED est éteinte, tandis qu'une valeur de 9 correspond au maximum de l'intensité de la LED.

ACTIVITE 18

REALISER UN CALCUL.

En utilisant les boutons ; afficher la somme des chiffres d'un nombre entier pris au hasard formé par deux chiffres.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



ACTIVITE 19

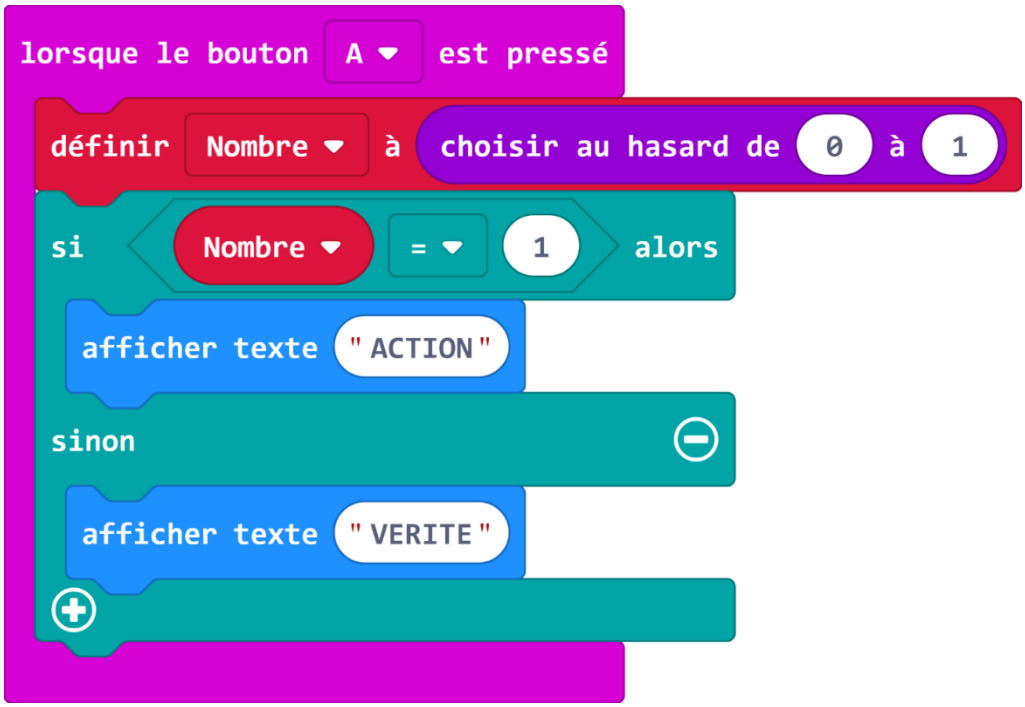
JOUER AU JEU "ACTION OU VERITE".

Le jeu de l'action ou vérité consiste à laisser le choix, à tour de rôle, aux joueurs entre deux options :

- Se voir demander de faire une action : Aller dire quelque chose à une personne, imiter une personnalité …
- Se voir poser une question (souvent embarrassante !).

En appuyant sur l'un des boutons, votre programme doit afficher d'une manière aléatoire le texte "ACTION ou "VERITE".

Refaite la même activité en utilisant le langage de programmation **MicroPython**.

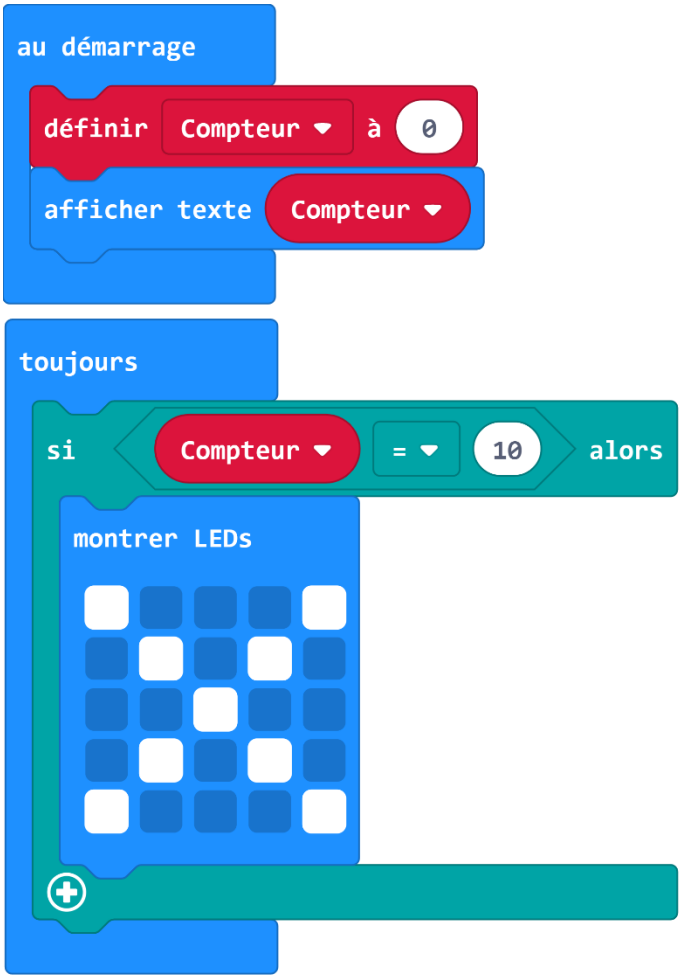


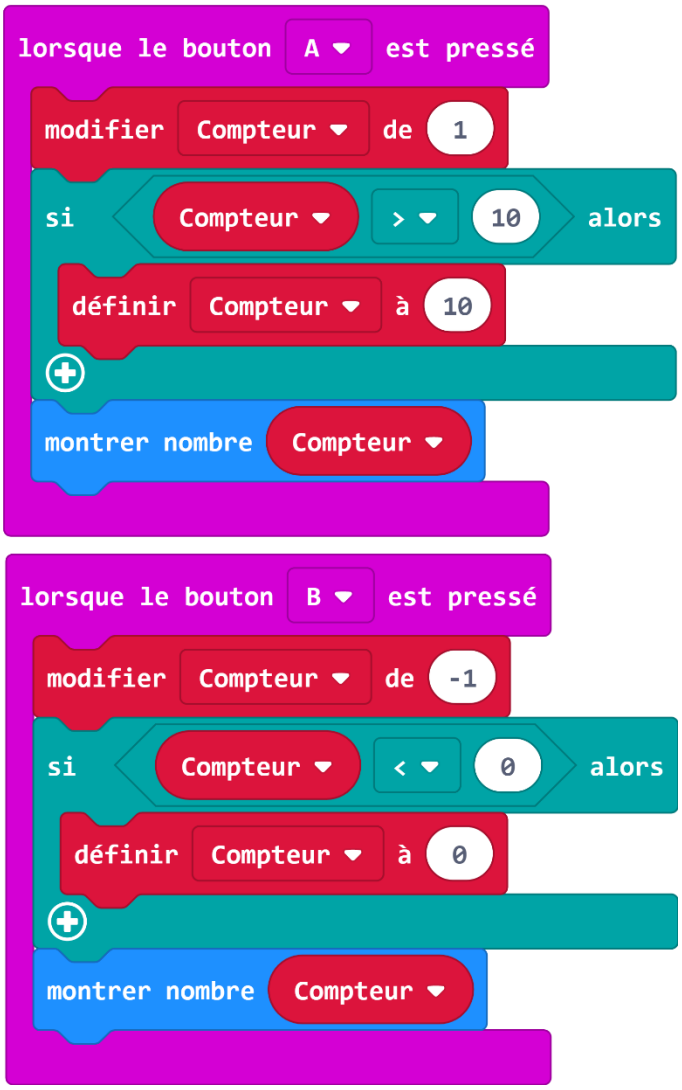
ACTIVITE 20

REALISER UN COMPTEUR.

Chaque appui sur le bouton A incrémentera le compteur (le nombre est limité à 10). Un appui sur le bouton B décrémentera le compteur.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.





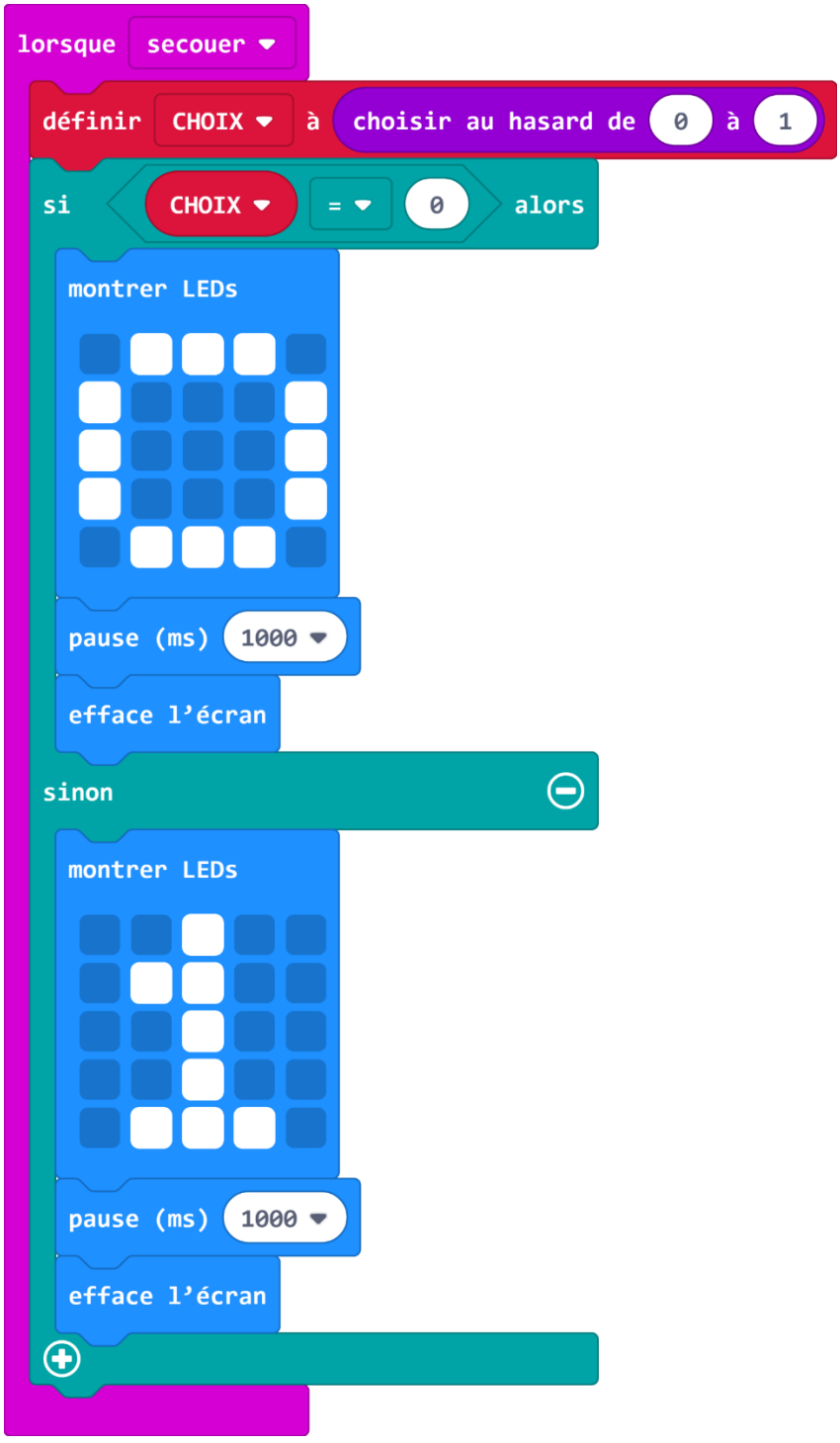
ACTIVITE 21

JOUER A "PILE OU FACE".

Le tirage au sort est souvent utilisé dans des jeux pour déterminer, par exemple, quel joueur commence ou pour effectuer un choix au hasard. Voici un programme qui permettra d'éviter toute tricherie grâce à la carte micro:bit !

- Lorsque la carte est secouée, un nombre au hasard est choisi, ici 0 ou 1.
- Si c'est le nombre 0, elle l'affiche pendant 1 seconde.
- Si c'est le nombre 1, elle l'affiche pendant 1 seconde.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



ACTIVITE 22

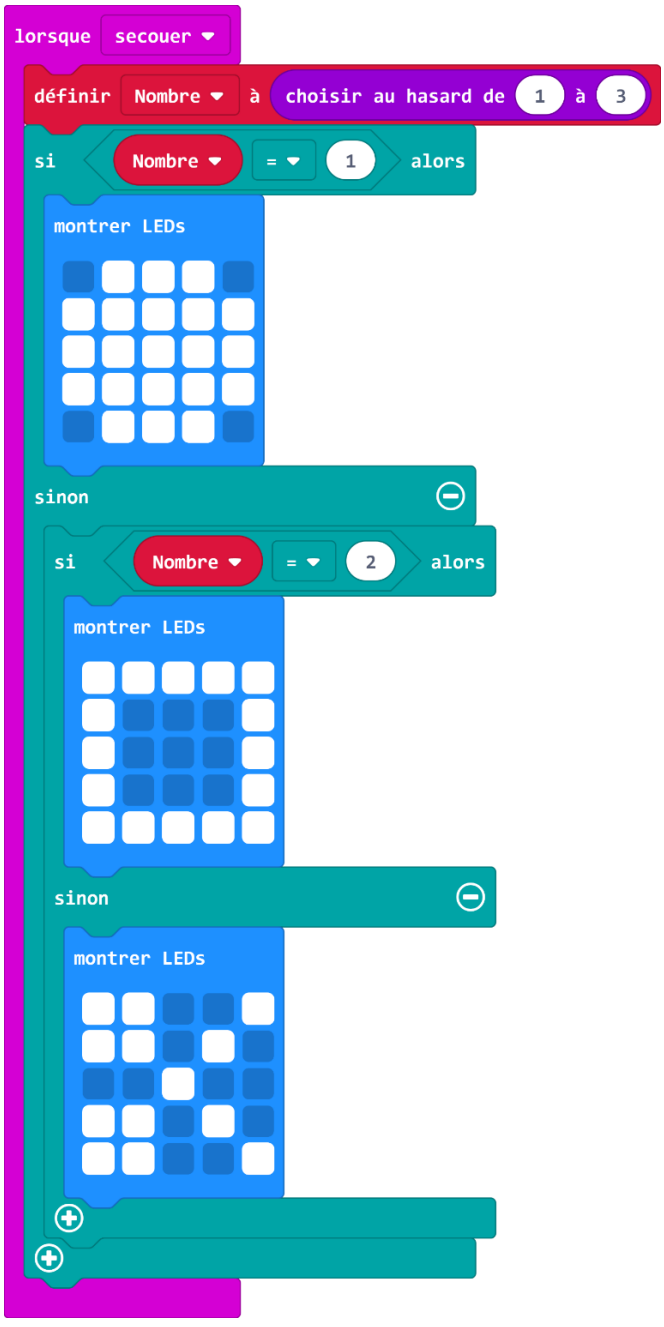
JOUER AU JEU "PIERRE, PAPIER, CISEAUX".

La carte micro:bit permet de jouer à de multiples jeux, comme le grand classique pierre papier ciseaux. Lorsque la carte est secouée, l'un des trois choix (pierre, papier, ciseaux) est sélectionné au hasard. Il est donc nécessaire de définir une variable qui sera un nombre compris entre 1 et 3, ce qui donnera trois choix possibles. Par exemple :

- le 1 pour la pierre ;
- le 2 pour le papier ;
- le 3 pour les ciseaux.

1. Dans le menu "Variables", définissez une variable (appelée « Nombre » ici). Ensuite, dans le menu "Maths", sélectionnez « Choisir au hasard ».
2. Il va falloir tester chaque nombre tiré au sort et afficher l'icône correspondante.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.

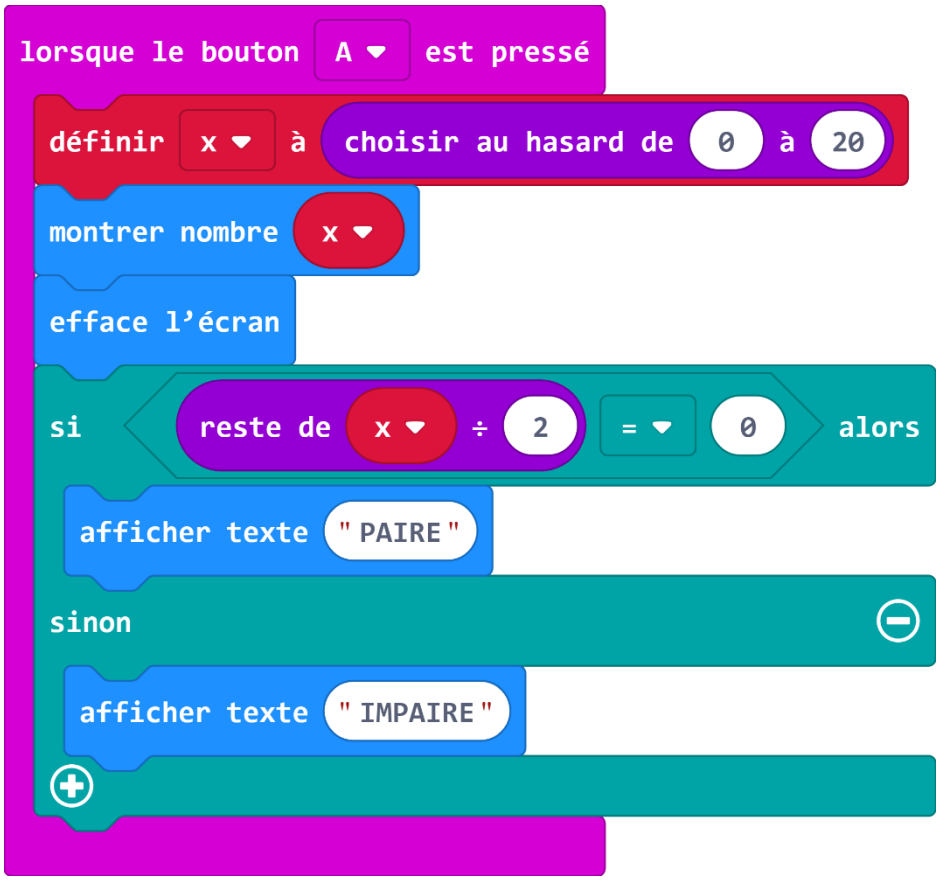


ACTIVITE 23

DETERMINER LA PARITE D'UN NOMBRE.

En utilisant les boutons ; afficher si un nombre entier compris entre 0 et 20 pris au hasard est paire ou impaire.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



ACTIVITE 24

CREER UNE CALCULATRICE SIMPLIFIEE.

Nous vous proposons ici de réaliser une calculatrice simple pour additionner deux nombres. Pour cela, nous allons utiliser des variables, à savoir des éléments qui vont varier en fonction d'événements ou d'une instruction (de calcul, par exemple).

Allez dans le menu "Variables", cliquez sur "Créer une variable ...", et choisissez son nom (ici, NOMBRE1) ; Répétez ensuite l'opération pour le second nombre.

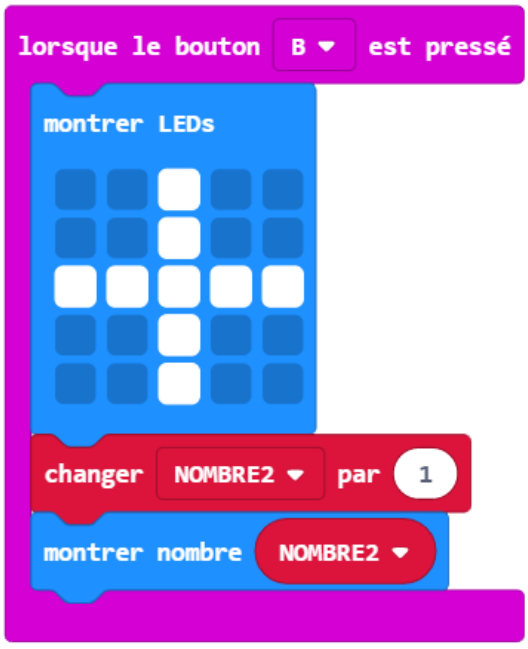
Le programme se déroule en trois parties :

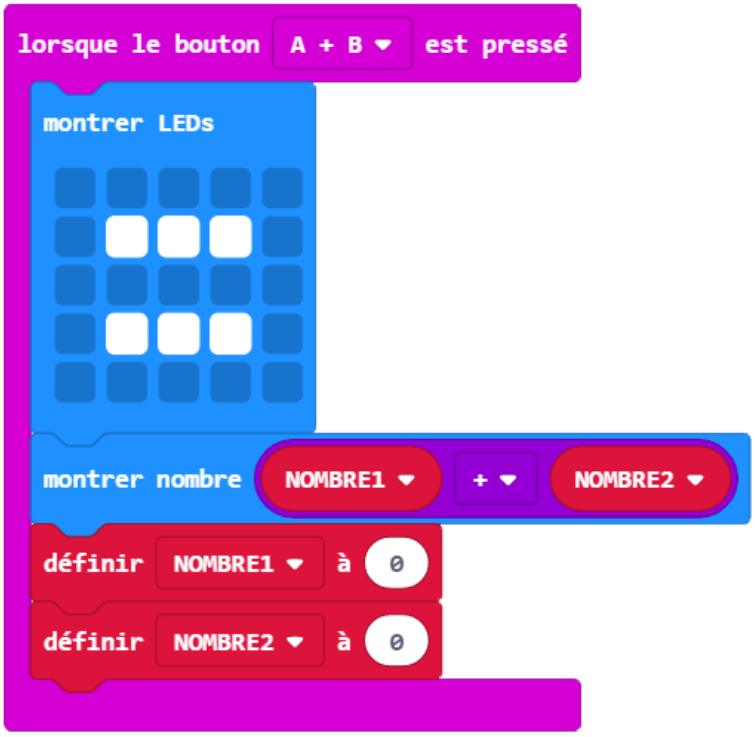
1. Une pression sur le bouton A permet d'afficher le premier nombre de l'addition.
2. Une pression sur le bouton B permet d'afficher le second nombre de l'addition.
3. Une pression sur les deux boutons simultanément permet d'obtenir le résultat.

Il faudra donc trois blocs d'instructions pour cette calculatrice.

Il est évidemment possible d'effectuer des soustractions, divisions, multiplications en changeant simplement le signe et son image. Notez que la simulation tient compte de la pression sur les deux boutons en ajoutant un bouton virtuel A+ B sous le bouton B.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.





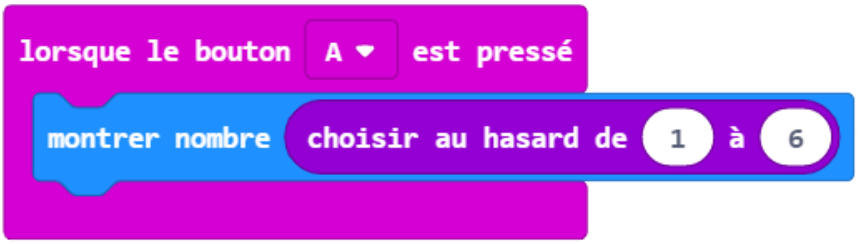
ACTIVITE 25

JOUER AU DE (VERSION 1).

Utilisez votre carte micro:bit pour jouer au dé, grâce à l'instruction "choisir au hasard de … à …" du menu "Maths", dans laquelle vous pouvez choisir l'étendue des nombres. Par défaut, l'instruction est réglée sur une plage de 0 à 10.

Pour un dé simple, nous avons choisi d'afficher au hasard un nombre compris entre 1 et 6. Pour la simulation, cliquez sur le point blanc Shake. Le nombre reste affiché tant que l'on ne réappuie pas sur le bouton.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.

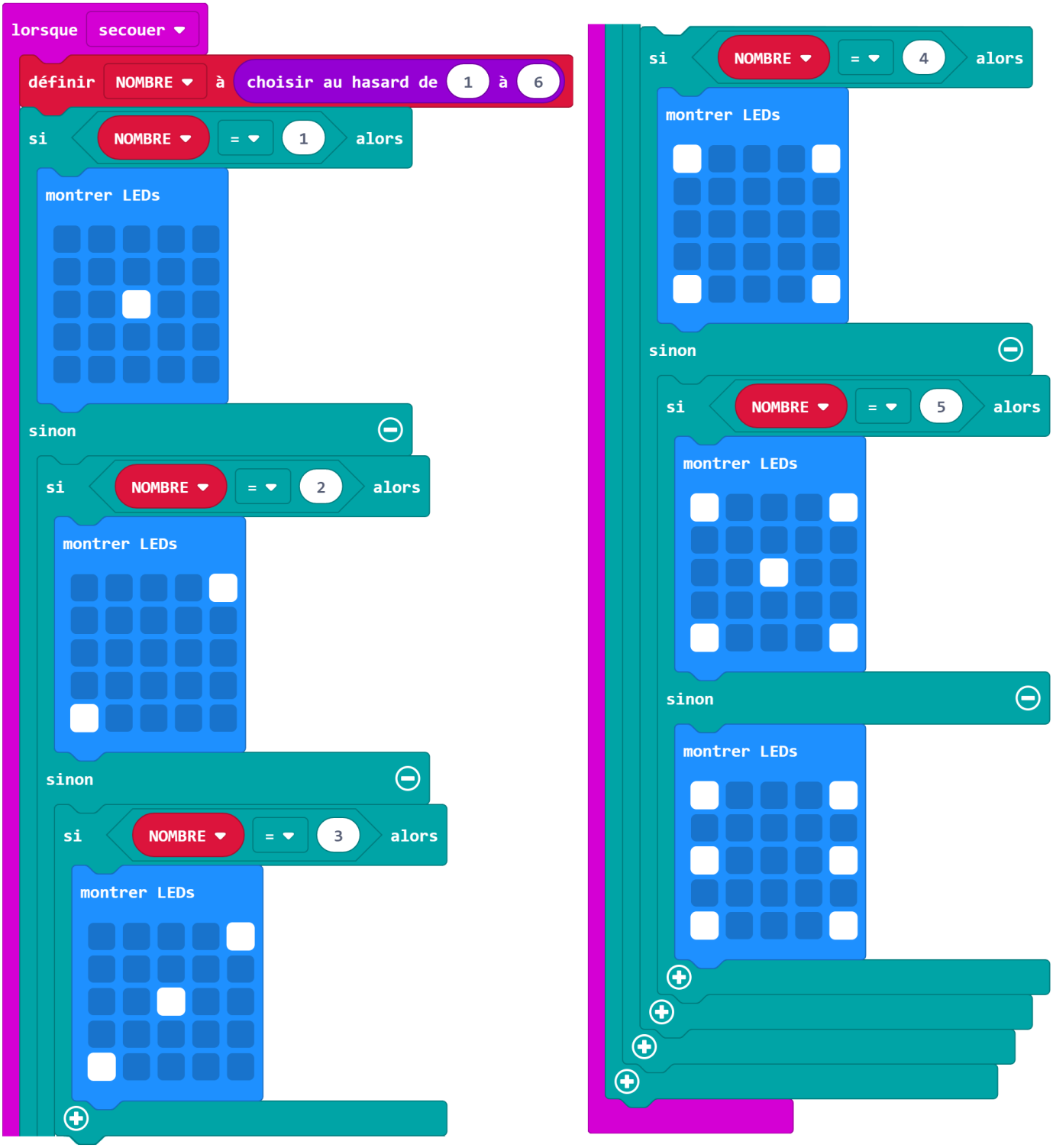


ACTIVITE 26

JOUER AU DE (VERSION 2).

Une variante de ce jeu consiste à afficher le dessin du dé.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



ACTIVITE 27

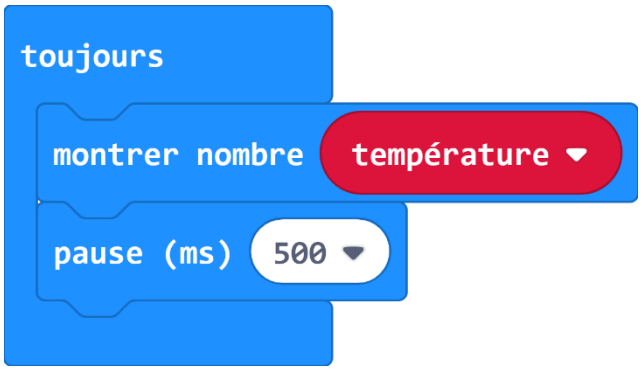
AFFICHER LA TEMPERATURE.

La carte micro:bit possède un capteur de température moyennement précis qui fournit un ordre de grandeur de la température du microprocesseur qui correspond généralement à la température ambiante car il chauffe très peu. L'instruction correspondante est dans le menu "Entrées".

Par défaut, lors de la simulation, MakeCode indique 21 °C. Pour obtenir une mesure plus précise, il faudra utiliser un module externe (de type Grove, par exemple).

Pour améliorer la lecture cette température, il est conseillé d'ajouter une pause pour que le texte cesse de défiler.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



ACTIVITE 28

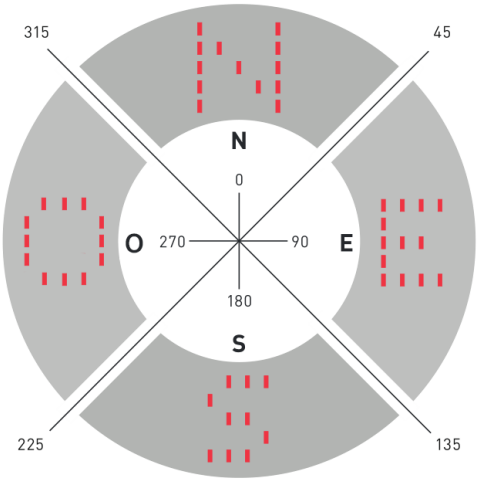
CREER UNE BOUSSOLE.

La carte micro:bit possède un capteur détectant le champ magnétique terrestre ; elle peut afficher un nombre compris entre 0 et 359 pour représenter tous les degrés d'une boussole. L'instruction "direction de la boussole (°)" se situe dans le menu Entrées.

Une fois le programme chargé sur la carte, le message "Tilt to fill screen" s'affichera, vous invitant à manipuler la carte dans tous les sens afin que toutes les LEDs soient allumées. Cette action sert à calibrer la boussole. Il se peut que la procédure s'arrête avant que vous n'ayez terminé. Pas de panique, le message revient et vous reprenez votre travail là où vous en étiez. Une fois cette étape effectuée, votre programme s'exécutera.

Pour une meilleure lisibilité, nous avons choisi de n'indiquer que les principales directions, d'où l'affichage des lettres N, E, S et 0, pour Nord, Est, Sud et Ouest.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



ACTIVITE 29

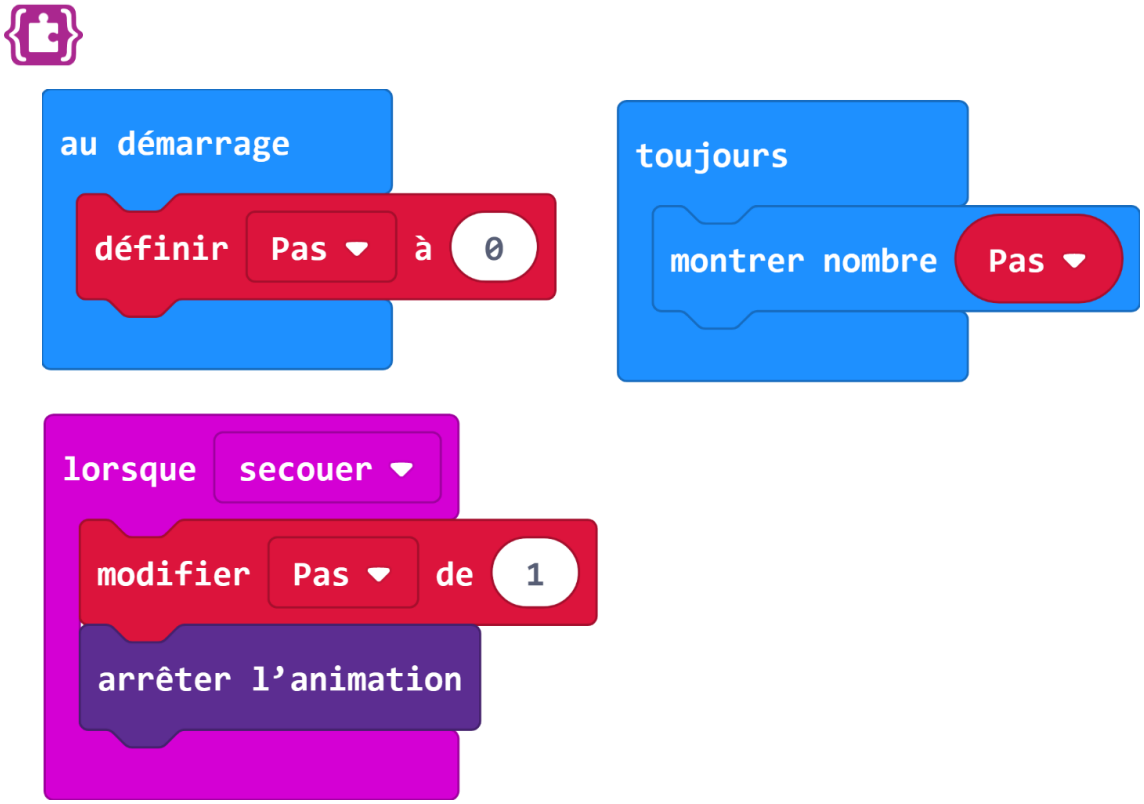
CONCEVOIR UN PODOMETRE (VERSION 1).

Un podomètre est un dispositif sensible au mouvement permettant de mesurer en temps réel le nombre de pas d'une personne.

Nous vous proposons d'utiliser l'accéléromètre de la carte pour fabriquer un podomètre simple qu'il suffira de fixer à votre cheville.

- Au démarrage, le nombre de pas est initialisé à 0. Dans le menu" Variables", nous avons créé la variable "PAS".
- À chaque secousse, la carte va incrémenter (ici, augmenter de la valeur 1) la variable "PAS" d'une unité.
- Puisque la valeur peut changer durant l'affichage, il est nécessaire d'ajouter l'instruction "arrêter l'animation" (présente dans l'onglet "Plus" du menu "LED") pour éviter un décalage.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



ACTIVITE 30

CONCEVOIR UN PODOMETRE (VERSION 2).

Une amélioration du podomètre de l'activité précédente (Activité 29) peut consister à indiquer la valeur du pas en mètres afin d'obtenir la distance parcourue.

- Ce réglage s'effectue via une pression sur le bouton A.
- Chaque pression augmentant de 0,25 mètre la longueur du pas.
- Fixez donc cette valeur en fonction de la longueur du vôtre.
- Cette dernière est ensuite multipliée par le nombre de pas.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.

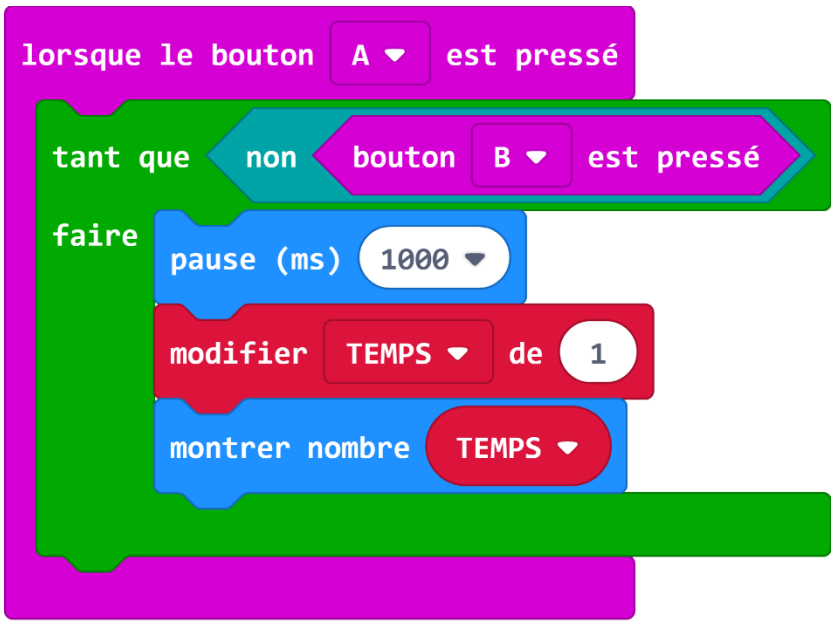
ACTIVITE 31

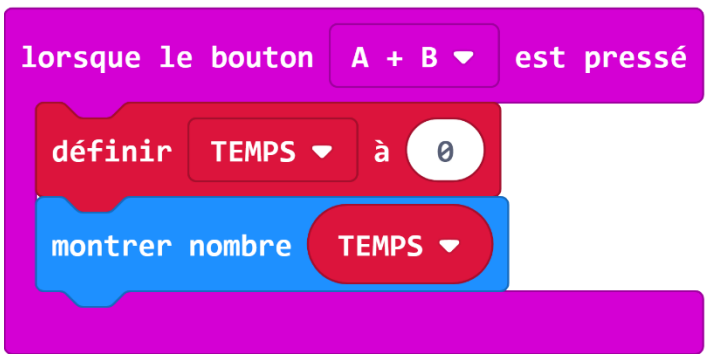
REALISER UN CHRONOMETRE.

La carte micro:bit permet de gérer le temps écoulé. Nous vous proposons ici de fabriquer un petit chronomètre :

- Via le menu "Variables", nous avons créé la variable "TEMPS",
- les boutons A + B permettent sa remise à zéro,
- le bouton A le démarrage et le bouton B l'arrêt.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.





ACTIVITE 32

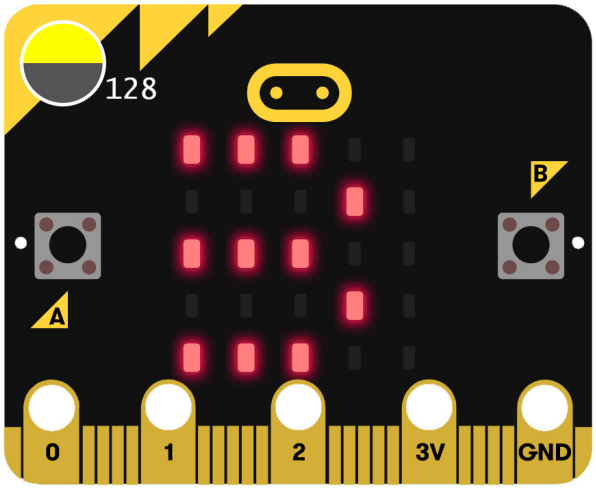
MESURER LA LUMINOSITE (VERSION 1).

Le capteur de lumière de la carte micro:bit permet d'estimer le niveau de lumière, en utilisant les LEDs de l'écran. Le niveau 0 correspond à l'obscurité et le niveau 255 à la lumière du jour.

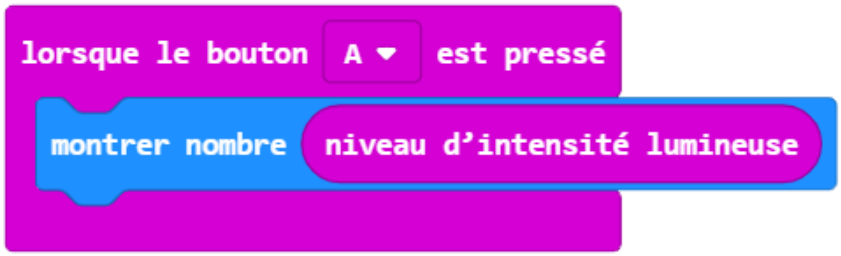
Le logiciel MakeCode vous offre deux possibilités d'affichage :

- soit un nombre compris entre 0 et 255 ;
- soit un affichage graphique à l'aide de barres.

Lorsqu'on appuie sur le bouton A, un indicateur chiffré de luminosité s'affiche.



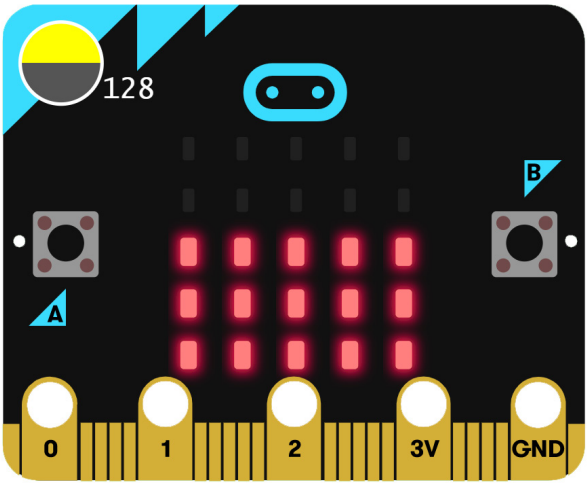
Refaite la même activité en utilisant le langage de programmation **MicroPython**.



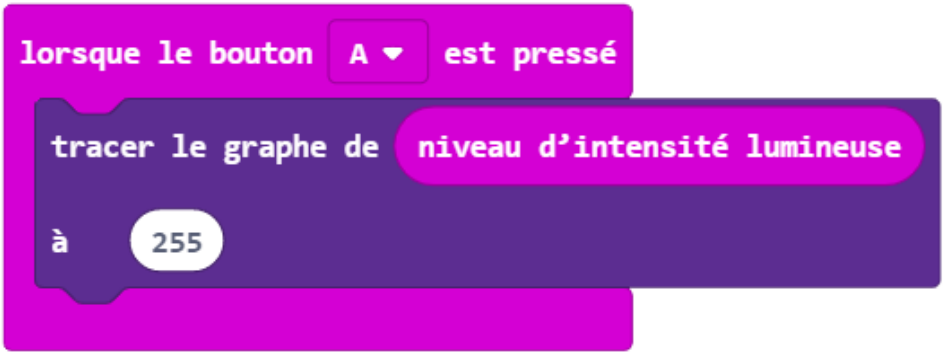
ACTIVITE 33

MESURER LA LUMINOSITE (VERSION 2).

Lorsqu'on déplace la carte micro:bit dans des endroits éclairés différemment, l'affichage indique graphiquement le niveau de lumière mesuré. Par défaut, la simulation indique la valeur 128.



Refaite la même activité en utilisant le langage de programmation **MicroPython**.



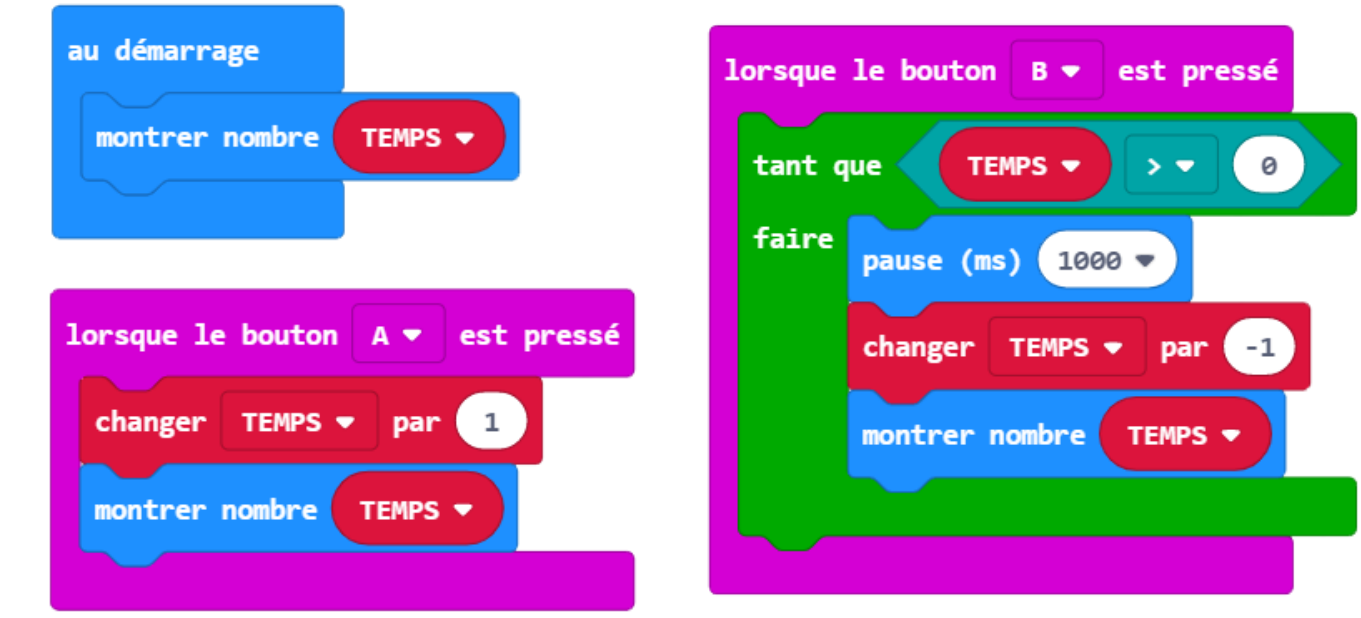
ACTIVITE 34

CREER UN MINUTEUR.

Il est souvent pratique, dans un jeu par exemple, de pouvoir disposer d'un compte à rebours. C'est l'intérêt de ce minuteur.

- Via le menu "Variables", on crée une variable" TEMPS" qui représente le temps s'écoulant.
- En appuyant sur le bouton A, vous définissez le temps souhaité pour le compte à rebours. Chaque pression correspond à 1 seconde. Ce temps sera affiché.
- Lorsque le bouton B est pressé, le compte à rebours démarre. L'instruction « changer TEMPS par -1 » constitue un compteur descendant. Elle permet de déclencher le compte à rebours.
- Vous pouvez compléter le programme en ajoutant, par exemple, une icône à la fin pour signaler qu'il est terminé.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



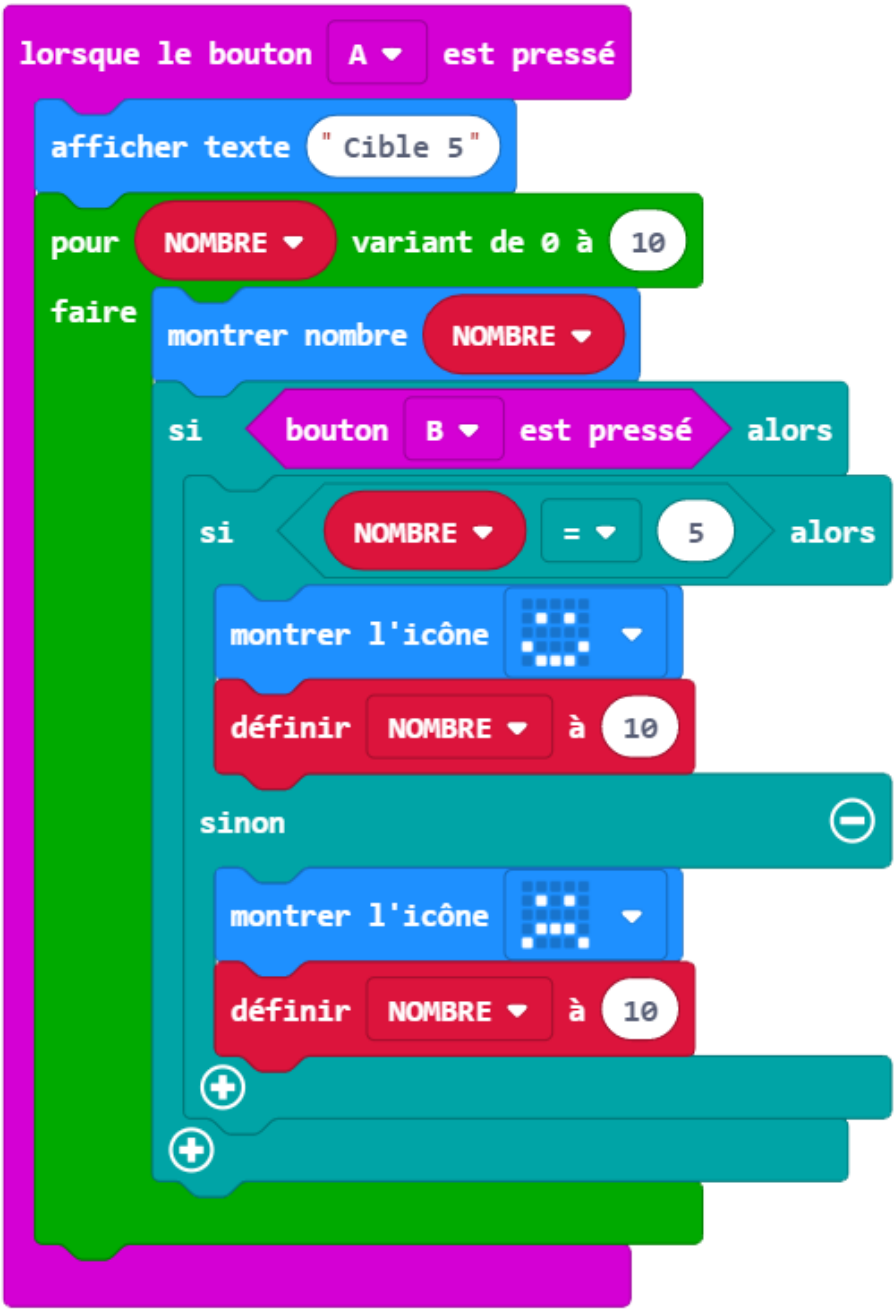
ACTIVITE 35

JOUER AU JEU DE "LA CIBLE".

La carte micro:bit est parfaitement adaptée pour concevoir de petits jeux. Vous en trouvez un grand nombre sur Internet. Nous vous proposons ici d'en créer un : le jeu de la cible.

- Une variable "NOMBRE" est utilisée, affichant les nombres de 0 à 10 lorsque le bouton A est pressé.
- Le jeu consiste à appuyer rapidement sur le bouton B quand le chiffre 5 apparaît (c'est la cible).
- En cas de réussite, un sourire s'affiche.
- Sinon, c'est un visage triste.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.

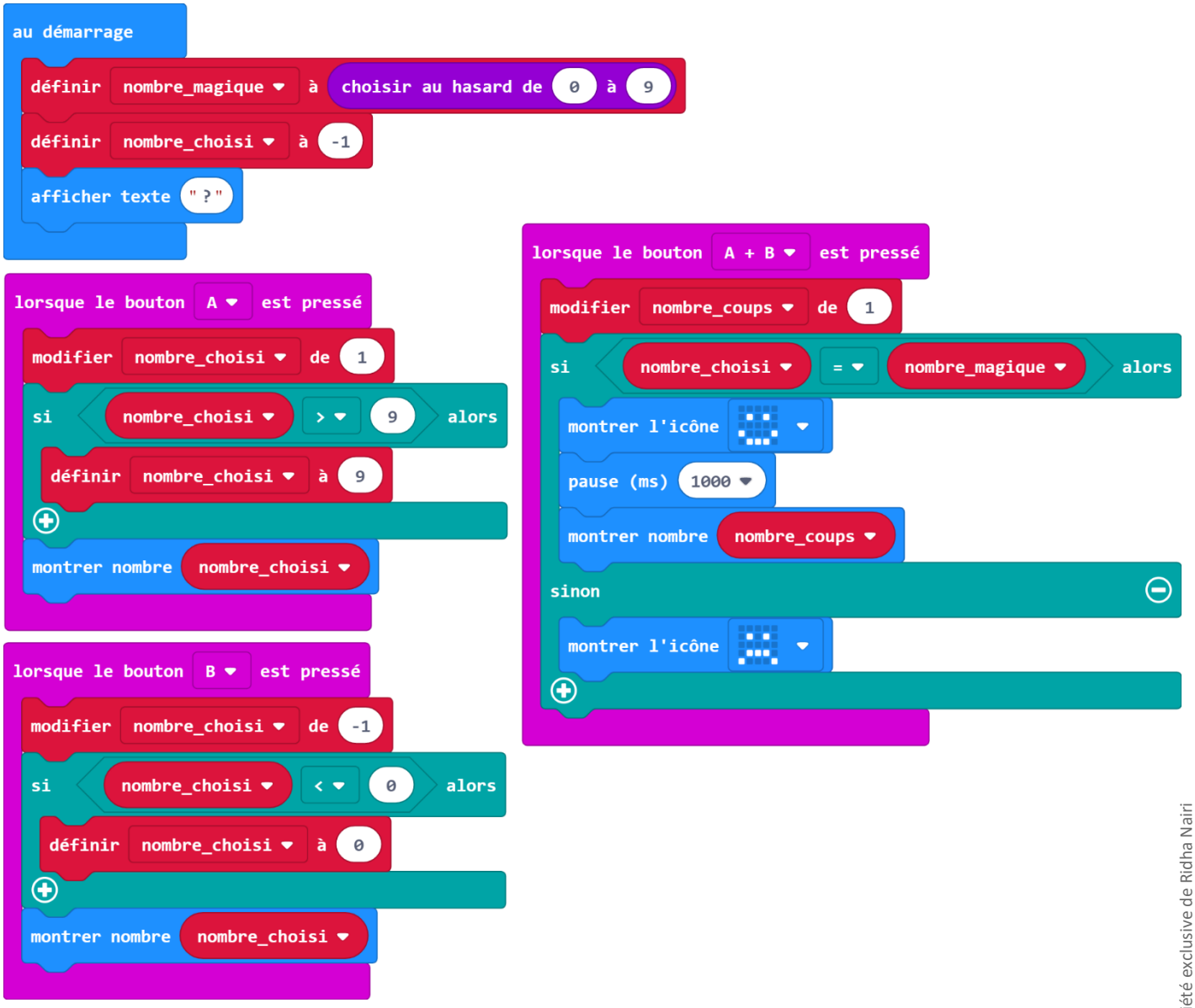


ACTIVITE 36

JOUER AU JEU DU "NOMBRE MAGIQUE".

L'objectif est ici de trouver le nombre compris entre 0 et 9 choisi aléatoirement par la carte micro:bit. À l'aide des boutons-poussoirs A et B, l'utilisateur pourra changer de nombre : il pressera le bouton A pour choisir un nombre plus grand et sur le bouton B pour un plus petit. Il devra valider son choix en pressant simultanément A et B. L'écran affichera le nombre de tentatives qui auront été nécessaires pour trouver le nombre choisi au départ.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



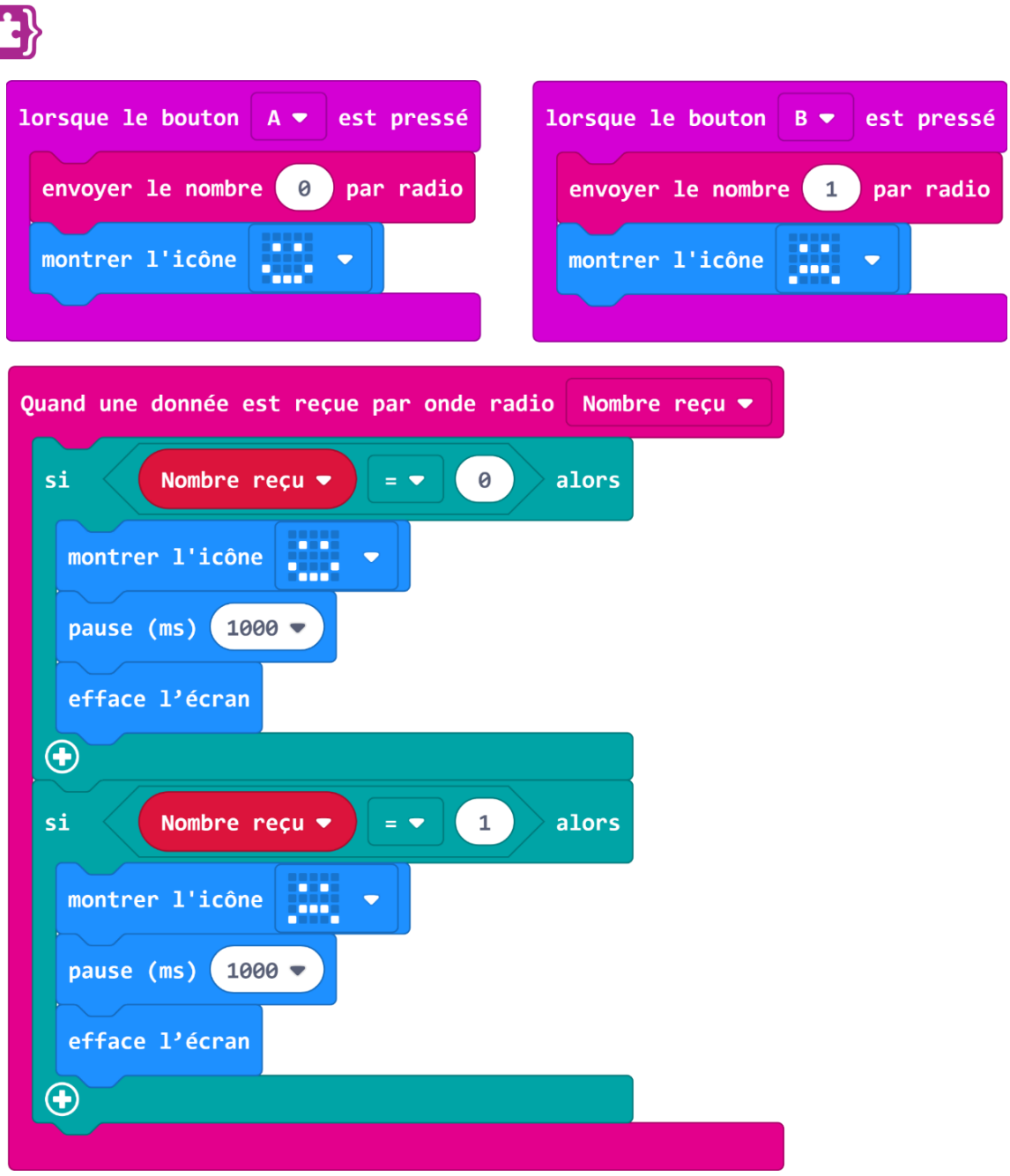
ACTIVITE 37

UTILISER LE MODE RADIO.

Il est possible de faire communiquer plusieurs cartes micro:bit entre elles grâce au mode radio. Il est pratique dans ce cas que les cartes soient alimentées par batterie, mais ce n'est pas indispensable.
La distance courante pour recevoir des informations par radio est d'environ 20 mètres mais peut aller jusqu'à 70 mètres. Il est théoriquement possible de faire communiquer ensemble un nombre illimité de cartes.
À l'aide de deux cartes, ce programme permet de signaler quelle est votre humeur à un ami en affichant un sourire ou une grimace. Votre ami pourra faire de même si le programme est chargé sur les deux cartes. Une variable "Nombre" reçu est créée dans le menu Variables.

- Si l'on presse le bouton A, on voit ce qui est envoyé à l'autre carte (ici, un message indiquant que l'on est heureux).
- Si l'on presse le bouton B, on voit aussi ce qui est envoyé à l'autre carte (ici, un message indiquant que l'on est triste).
- Lorsqu'une carte reçoit une donnée, elle affiche l'icône correspondante pendant un certain temps, puis l'efface.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.

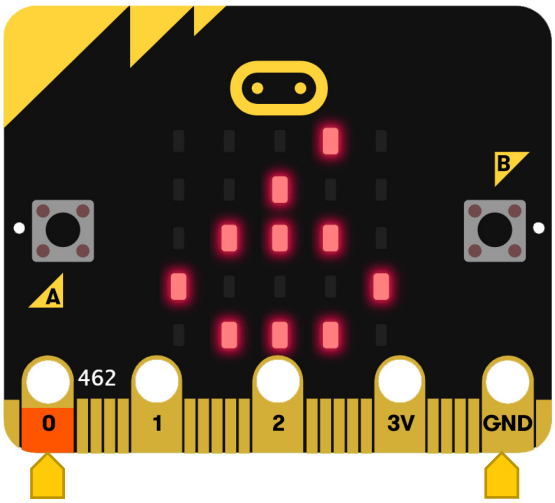


ACTIVITE 38

DETECTER DES EMOTIONS.

La carte micro:bit possède des contacts appelés broches (Pins) qui peuvent récupérer des signaux analogiques ou numériques. La broche P0 que nous allons utiliser est analogique et renvoie des valeurs comprises entre 0 et 1023.

- Pour s'amuser à « détecter » des émotions, il suffit, après avoir téléchargé le programme, de mettre les doigts sur la broche P0 et la broche GND.



- La résistance électrique de la peau varie avec la production de sueur (liée aux émotions subies). Ce que nous proposons n'est donc qu'un jeu très aléatoire et sans réelle valeur scientifique.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.

ACTIVITE 39

CREER DES FONCTIONS.

Via le menu Fonctions, il est possible de créer ses propres blocs dans MakeCode, ce qui évite d'avoir à réécrire certains programmes : il suffit juste d'utiliser le bloc de la fonction créée.

- Dans le menu "Fonctions", cliquez sur "Créer une fonction".
- Écrivez directement le nom que vous voulez donner à la fonction (ici, PLUIE) et cliquez sur Ok.

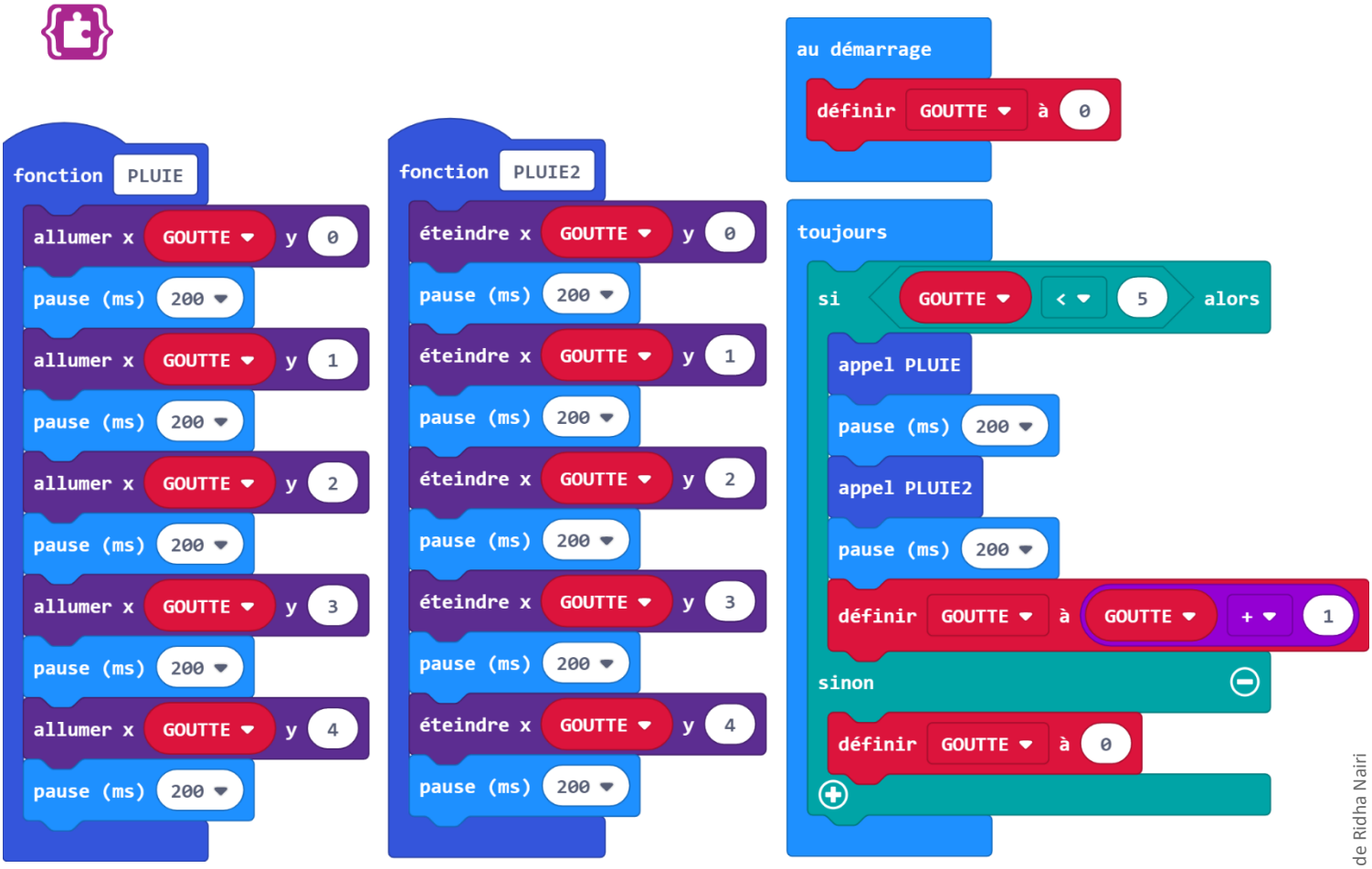
Nous allons créer une petite animation qui simule des gouttes de pluie. Pour cela, il faut créer une seconde fonction PLUIE2.

Le programme démarre de cette manière :

- La variable GOUTTE correspond à la position de la LED en X et Y.
- Le programme fait varier les positions des LEDs en X et en Y.

Vous pouvez le complexifier en appelant de nouveau les fonctions PLUIE et PLUIE2 sans avoir besoin de les réécrire.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.



Une fonction est un ensemble d'instructions regroupées sous un nom et s'exécutant à la demande (l'appel de la fonction).

Syntaxe de la déclaration d'une fonction avec JavaScript	Syntaxe de la déclaration d'une fonction avec MicroPython

`function identifiant (paramètres){`
`  ...`
`return valeur résultat de l'appel`
`}`
	`def identifiant (paramètres):`
	`  ...`
	`  return valeur résultat de l'appel`

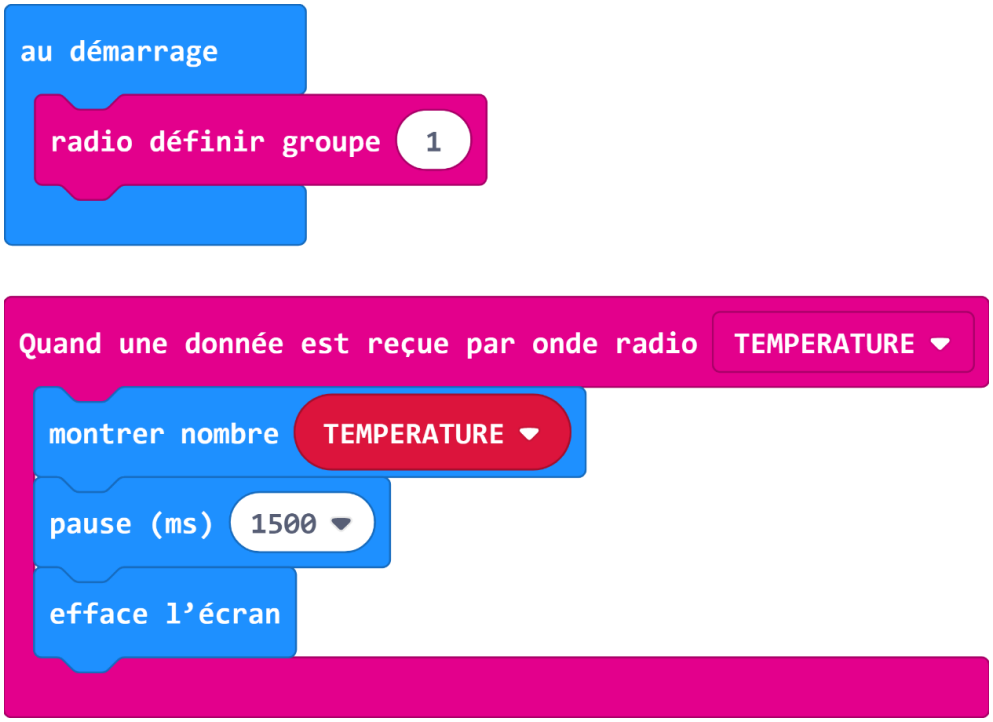
ACTIVITE 40

CREER UN THERMOMETRE D'EXTERIEUR.

Nous savons que le mode radio permet à deux cartes micro:bit de communiquer. Utilisons alors cette fonction associée au capteur de température interne pour réaliser un thermomètre d'extérieur avec affichage intérieur. Les deux cartes sont chacune alimentées par leur pack batterie pour être autonomes et possèdent leur programme propre.

- La carte extérieure envoie la valeur de la température, puis marque une pause avant de renvoyer une valeur. À noter que l'instruction « radio définir groupe » permet d'affecter les deux cartes à un seul groupe pour mieux communiquer.
- La carte intérieure va recevoir la donnée (renommée TEMPÉRATURE) qui sera affichée comme une variable durant 1,5 seconde avant d'effacer l'écran (pour économiser la batterie).
- Pour varier, il est possible de faire émettre une alerte lumineuse si la température atteint un certain seuil (une gelée, par exemple). Bien d'autres possibilités sont envisageables en fonction de votre imagination.

Refaite la même activité en utilisant le langage de programmation **MicroPython**.

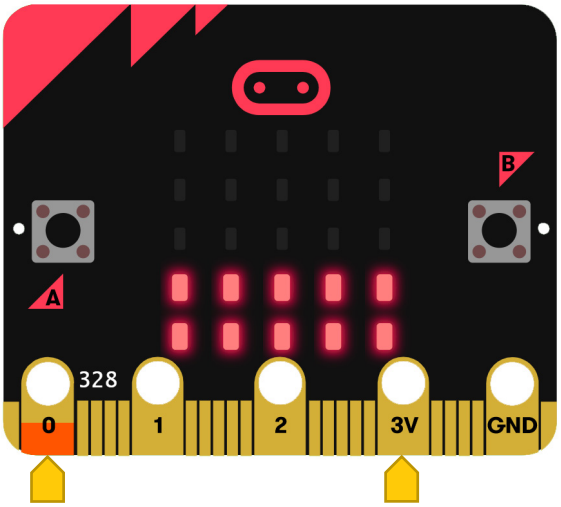


ACTIVITE 41

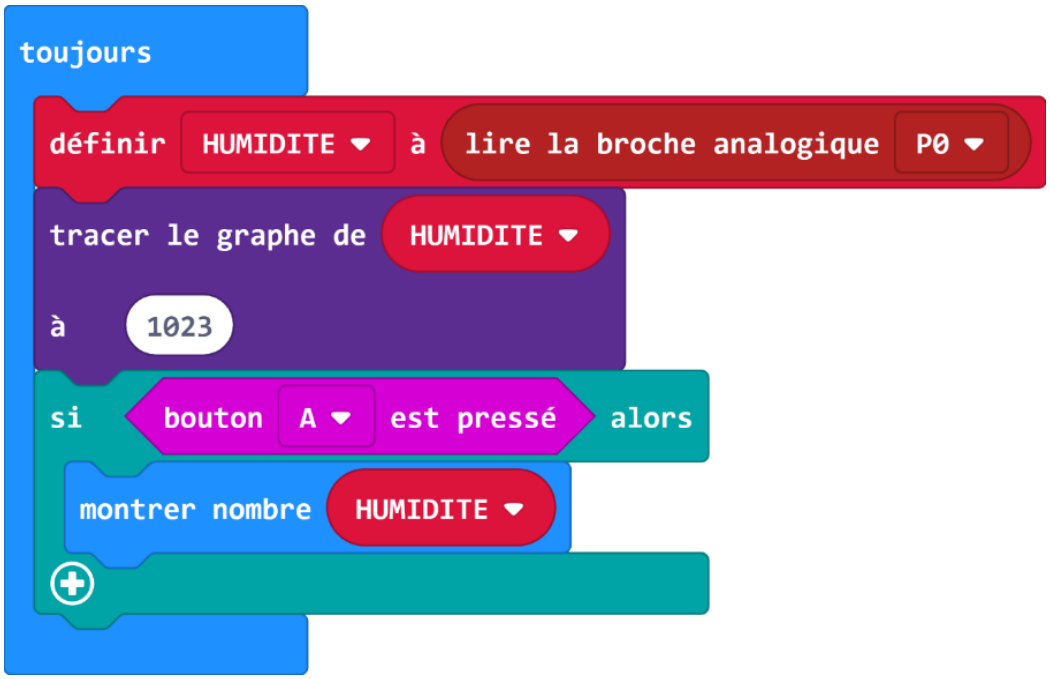
TESTER L'HUMIDITE D'UNE PLANTE (VERSION 1).

La carte micro:bit peut également prendre soin de vos plantes en faisant office de testeuse d'humidité.

La terre de vos plantes possède une certaine résistance électrique qui dépend de la quantité d'eau et des éléments qu'elle contient (nutriments, etc.). Cette résistance varie notamment en fonction du taux d'humidité : plus il y a d'eau et plus elle est faible. Pour obtenir la valeur de cette résistance, nous allons relier la carte à deux clous plantés dans la terre via des câbles avec pinces crocodiles.



Le programme trace un graphe en fonction de la valeur obtenue. Mais pour avoir une valeur précise, appuyez sur le bouton A. Celle-ci se situe aux alentours de 250 pour une terre sèche et de 1 000 pour une terre humide.

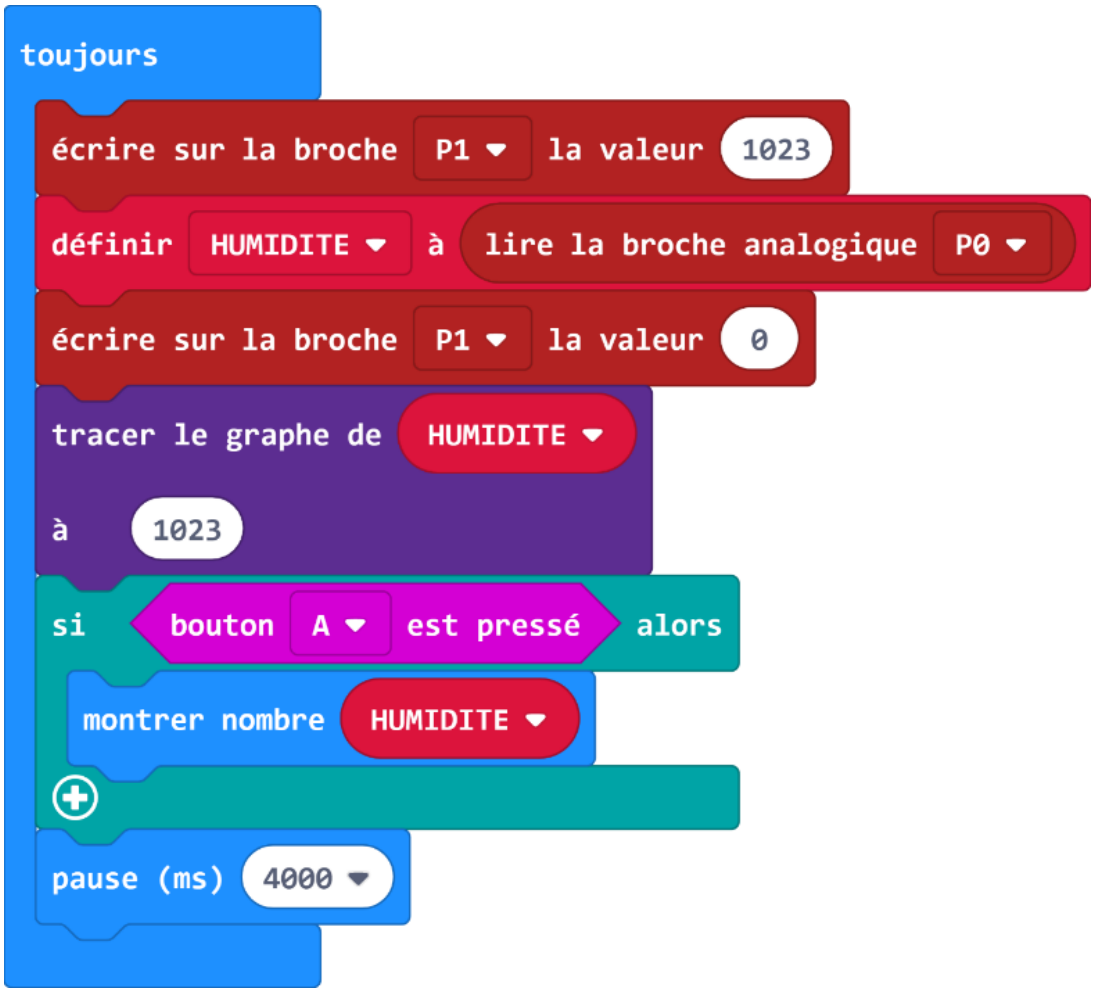


ACTIVITE 42

TESTER L'HUMIDITE D'UNE PLANTE (VERSION 2).

En l'état, le programme précédent (Activité 41) consomme du courant en permanence, mais il est possible d'économiser ce dernier en modifiant le programme :

- Débranchez la pince de la broche P0 et connectez-vous à la broche P1 juste le temps de prendre la valeur du taux d'humidité : vous économiserez l'énergie de vos piles.
- Le taux d'humidité variant lentement, une pause a aussi été ajoutée pour économiser de l'énergie.
- En cas de sécheresse de la terre, vous pouvez ajouter une alarme lumineuse.



V 019.12.14

http://profridha.blogspot.com

Edition le libre savoir ®